A Web-based finite element model for heat transfer

Trevor Allen Wilcox

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Part of the Higher Education Commons

Repository Citation

https://digitalscholarship.unlv.edu/rtds/1230

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600
NOTE TO USERS

Page(s) missing in number only; text follows. Page(s) were microfilmed as received.

39

This reproduction is the best copy available.

UMI
A WEB-BASED FINITE ELEMENT MODEL FOR HEAT TRANSFER

by

Trevor A. Wilcox

Bachelor of Science in Mathematics
University of Alaska, Anchorage
1997

Master of Science in Mathematics
University of Nevada, Las Vegas
1998

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree
Department of Mechanical Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2000

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The Thesis prepared by

Trevor Allen Wilcox

Entitled

A Web Based Finite Element Model for Heat Transfer

is approved in partial fulfillment of the requirements for the degree of

Masters of Science

Examination Committee Chair

Dean of the Graduate College

Graduate College Faculty Representative
ABSTRACT

A Web-Based Finite Element Model for Heat Transfer

by

Trevor A. Wilcox

Dr. Darrell Pepper, Examination Committee Chair
Professor and Chairman, Department of Mechanical Engineering
University of Nevada, Las Vegas

A web based Java program has been developed to solve 2-D transient or steady state heat transfer problems. The numerical solver is based on the finite element method. The program is designed to work with both triangle and quadrilateral elements. In this thesis the development of the finite element method, using the Galerkin weighted residuals method, is reviewed and the technique applied to the advection-diffusion equation for the transport of temperature. The Java code and graphical interface used to implement the solver for use on the Internet is discussed.
# TABLE OF CONTENTS

**ABSTRACT** ............................................................................................................................ iii

**LIST OF FIGURES** .................................................................................................................. vi

**LIST OF TABLES** .................................................................................................................. vii

**ACKNOWLEDGEMENTS** ........................................................................................................ viii

**CHAPTER 1** ............................................................................................................................ 1
  Introduction .............................................................................................................................. 1

**CHAPTER 2** ............................................................................................................................ 1
  Heat Equation ........................................................................................................................... 1
  Conduction Problem .............................................................................................................. 4
  Finite Difference Method ...................................................................................................... 6
  Finite Volume Method .......................................................................................................... 15
  Boundary Element Method .................................................................................................. 17
  Boundary Fitted Coordinates ............................................................................................... 22

**CHAPTER 3** ............................................................................................................................ 27
  Finite Element Method ......................................................................................................... 27
  Galerkin Formulation .......................................................................................................... 28
  Elements .................................................................................................................................. 31

**CHAPTER 4** ............................................................................................................................ 40
  Java History ........................................................................................................................... 40
  Applet ..................................................................................................................................... 41
  Abstract Window Toolkit ..................................................................................................... 43
  Swing ...................................................................................................................................... 44
  Java – Fortran Comparison ................................................................................................. 45
  Finite Element Program ...................................................................................................... 49

**CHAPTER 5** ............................................................................................................................ 51
  Numerical Results ................................................................................................................ 51

**CHAPTER 6** ............................................................................................................................ 71
  Conclusion ............................................................................................................................... 71
LIST OF FIGURES

Fig. 2.1 Analytic Test Problem Statement .................................................................5
Fig. 2.2 General FDM Mesh .........................................................................................7
Fig. 2.3 FDM Mesh for Analytic Test Problem ..........................................................11
Fig. 2.4 Generic Transform and Inverse Transform ..................................................15
Fig. 2.5 Generic FVM Mesh .........................................................................................16
Fig. 2.6 General Domain for BEM .............................................................................18
Fig. 2.7 Example Problem for BEM .........................................................................20
Fig. 2.8 Structured Mesh for Example Problem .......................................................21
Fig. 2.9 BEM mesh for the BEM Example Problem ..................................................22
Fig. 2.10 Domain in Curvilinear Coordinate .............................................................23
Fig. 2.11 BFC Transformed Domain .........................................................................25

Fig. 3.1 Three Node Triangular Element ....................................................................31
Fig. 3.2 Triangular Element Areas .............................................................................33
Fig. 3.3 Six Node Triangular Element .......................................................................34
Fig. 3.4 Four Node Quadrilateral Element ...............................................................35
Fig. 3.5 Quadrilateral Element in Natural Coordinates .............................................36
Fig. 3.6 Eight Node Quadratic Element .....................................................................37

Fig. 4.1 JFC, Swing, and JDK Relationship ...............................................................45

Fig. 5.1 Domain of Analytic Test Problem ...............................................................51
Fig. 5.2 Symmetry of Analytic Test Problem ............................................................52
Fig. 5.3 Mesh for Analytic Test Problem ..................................................................53
Fig. 5.4 Plot of Analytic Test Problem .....................................................................54
Fig. 5.5 Solution Values of Analytic Test Problem ...................................................59
Fig. 5.6 Percent Error of Analytic Test Problem .......................................................60
Fig. 5.7 Brick Wall Test Problem .............................................................................61
Fig. 5.8 Symmetry of Brick Wall Test Problem .......................................................62
Fig. 5.9 Mesh of Brick Wall Test Problem ................................................................62
Fig. 5.10 Solution Comparison of Brick Wall Test Problem ......................................64
Fig. 5.11 Flux and Convection Test Problem ............................................................65
Fig. 5.12 Mesh for Flux and Convection Test Problem .............................................66
Fig. 5.13 Solution Comparison for Flux and Convection Test Problem .....................68
Fig. 5.14 Fin Test Problem .........................................................................................69
Fig. 5.15 Mesh for Fin Test Problem .........................................................................69
Fig. 5.16 Solution Comparison for Fin Test Problem ................................................70
LIST OF TABLES

Table 5.1  Solution Comparison of Analytic Test Problem .............................................55
Table 5.2  3 Node Solution of Analytic Test Problem ......................................................55
Table 5.3  4 Node Solution of Analytic Test Problem ......................................................56
Table 5.4  6 Node Solution of Analytic Test Problem ......................................................56
Table 5.5  8 Node Solution of Analytic Test Problem ......................................................57
Table 5.6  FDM Solution of Analytic Test Problem .........................................................57
Table 5.7  Percent Error Comparison of Analytic Test Problem .....................................58
Table 5.8  Solution Comparison of Brick Wall Test Problem ...........................................63
Table 5.9  Solution Comparison for Flux and Convection Test Problem .........................67
Table 5.10 Solution Comparison for Fin Test Problem .....................................................70
ACKNOWLEDGEMENTS

I would like to express my most sincere thanks to Dr. Darrell W. Pepper for his guidance and constant interest. Without his supervision, I would have not been able to complete this thesis.

Also, I would like to thank the committee members, Dr. Mohamed Trabia, Dr. Yi-Tung Chen, and Dr. Laxmi Gewali, for their interest and assistance.

Furthermore, I would like to thank my parents, Rena Offutt and David Wilcox, for their continuing support in my education.

Thanks also to the many who have helped me throughout my quest for higher education.
CHAPTER 1

Introduction

For the purpose of this thesis we will concentrate on a unique numerical scheme for the solution of heat transfer in two dimensions. We first look at the equation for heat conduction in a general domain

$$\frac{\partial T}{\partial t} - \nabla(K \nabla T) = Q$$

For simple domains solutions to the conduction problem can be found in most heat transfer books or can be solved analytically. As the geometry or physics of the problem becomes more complex, the problem becomes more difficult or impossible to solve using exact methods. Since most practical problems of interest involve complex domains and boundary conditions, one must look to numerical methods to solve the problem.

The concept of present day finite elements dates back to the 1940's. It wasn't until the 1970's when an influx of contributions from other fields enlarged the types of problems being solved (Comini et al., 1994). Since the 1970's the finite element method has evolved into a powerful solver for solid mechanics, fluid mechanics, and heat transfer. We will employ the finite element method to solve the heat transfer problem in this thesis.
Before we develop the finite element method, we will first give a survey of other numerical methods and compare them to the finite element. Two of the most popular numerical methods are the finite difference and finite volume methods. The finite difference and finite volume approaches are fairly versatile and relatively easy to implement on regular domains. However, both methods become troublesome to implement when the domains are irregular. However, the finite element method works well for regular as well as irregular geometry.

To solve the general heat equation, a finite element program has been developed in the Java programming language. Java was created by Sun Microsystems engineers in 1996. Java is an object oriented programming language similar to C++. The reason for choosing Java is because it was designed, in part, to be deployed over the Internet. The purpose of this work was to develop a finite element program that is available over the Internet. The solver consists of standard techniques used in the finite element method. The graphical user interface makes use of Java’s Swing components. These Swing components allow the user to do preprocessing by viewing the mesh before the problem is solved. After the problem is solved the solution is displayed for the user.

The finite element program solves heat transfer problems involving conduction with convection. The user must establish the mesh for the domain using either triangles or quadrilateral elements. The program can solve regular and irregular domains with a variety of boundary conditions. As the program exists, all information about the problem to be solved must be stored in a data file. A sample data file can be found in the appendices.
CHAPTER 2

Heat Equation

The general, two-dimensional time dependent energy equation in Cartesian coordinates can be written as (Incropera and DeWitt, 1996)

\[
\rho c_p \left( \frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) = \frac{\partial}{\partial x} \left[ K_x \frac{\partial T}{\partial x} \right] + \frac{\partial}{\partial y} \left[ K_y \frac{\partial T}{\partial y} \right] + Q
\]

or, in vector notation,

\[
\rho c_p \left( \frac{\partial T}{\partial t} + \mathbf{V} \nabla T \right) = \nabla \left[ K \nabla T \right] + Q
\]

The governing equation we will discuss for illustrating development of the finite element procedure is the Poisson equation form (steady state) for diffusion (i.e., conduction) of heat. The form of the governing equation is

\[-\nabla (K \nabla T) = Q \quad (2.1)\]

where \(Q\) is a heat source or sink and \(K\) is the thermal conductivity. There are three types of boundary conditions associated with equation (2.1). The first is the Dirichlet condition given by

\[T = T_{\text{fixed}}\]

The second type of boundary condition is the Neumann, written as
\[ K \frac{\nabla T}{\partial n} = q \]

where \( q \) is the heat flow. The last type of boundary condition is the mixed, or Robin, which is normally associated with convection. For convection, the boundary condition is

\[ q = h(T - T_\infty) \]

where \( T_\infty \) is the ambient temperature. In order to solve the heat equation appropriate boundary, initial conditions, and time dependency need to be specified. These boundary conditions are stated in mathematical form based on the physical make up of the problem at hand. A general boundary condition form for (2.1) can be described as

\[ aT + b \frac{\partial T}{\partial n} + c = 0 \]  \hspace{1cm} (2.2)

Equation (2.2) gives Neumann (flux) condition when \( a = 0 \), mixed (Robin) condition if \( a \neq 0 \) and \( b \neq 0 \), and if \( b = 0 \) we have Dirichlet condition. The solution of the Poisson equation is dependent upon the type of boundary condition associated with the problem. In the next section we will look at a simple conduction problem.

Conduction Problem

A conduction problem with Dirichlet conditions will first be solved using the method of separation of variables. Using the method of separation of variables requires a relatively simple geometry (Myers, 1998). To use the method of separation of variables the partial differential equation must be linear and homogeneous. Also, the boundary conditions must be linear and partially homogeneous. For these reasons we have chosen the problem shown in Fig. 2.1 with constant \( K \).
Fig. 2.1 Analytic Test Problem Statement

\[ K \nabla^2 T + Q = 0 \]  \hspace{2cm} (2.3)

\[ \frac{\partial T}{\partial x} (0, y) = 0 \]  \hspace{2cm} (2.4)

\[ T(L, y) = T_L \]  \hspace{2cm} (2.5)

\[ \frac{\partial T}{\partial y} (x, 0) = 0 \]  \hspace{2cm} (2.6)

\[ T(x, L) = T_L \]  \hspace{2cm} (2.7)

To satisfy the necessary homogeneity assume the solution is of the following form
\[ T(x, y) = a(x) + b(y) + u(x, y) \quad (2.8) \]

It should be noted that not all problems of this type can be solved using the form of (2.8).

Following the development in the appendix, the analytical solution is

\[ T(x, y) = T_e + \frac{Q}{2K}(L^2 - x^2) + 2\frac{Q L^3}{K} \sum_{n=1}^{\infty} \frac{\sin(\lambda_n x) \cosh(\lambda_n y)}{(\lambda_n L)^3} \cosh(\lambda_n L) \quad (2.9) \]

where \( \lambda_n = \frac{(2n-1) \pi}{2L} \).

Finite Difference Method

As one can see from the previous development of the conduction problem, analytical solutions are limited and mathematically rigorous, and in most practical instances impossible to develop solutions. Numerical methods are techniques used to approximate problems when their analytical solutions are difficult or intractable. Instead of yielding exact temperatures, numerical methods give approximate values at discrete points within the domain. One method that has become popular with the widespread use of computers is the Finite Difference Method (FDM).

To implement the finite difference method the domain is meshed into an array of grid points. Fig. 2.2 shows a region that has been divided into a mesh. The spacing between nodes is denoted by \( \Delta x \) and \( \Delta y \) in the \( x \) and \( y \) directions. Each node is identified by using subscripts \( i \) and \( j \). The \( x \)-coordinate corresponds to \( i \) while the \( y \)-coordinate corresponds to \( j \). A finite difference formula for temperature must be specified for an interior node. To derive this formula we choose an arbitrary node and assign it an \((i, j)\).
The node to the left and the right of \((i, j)\) are assigned \((i-1, j)\) and \((i+, j)\). Likewise, the nodes above and below \((i, j)\) are assigned \((i, j+1)\) and \((i, j-1)\).

![General FDM Mesh](https://via.placeholder.com/150)

**Fig. 2.2** General FDM Mesh

Before we derive the formulas for the finite difference method we should review some background material. Recall the Taylor series expansion of the function \(f(x)\)

\[
f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \cdots
\]

Truncating the series after two terms we have the approximation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The expression given by (2.10) is called the forward difference. The backward difference is obtained by replacing $h$ by $-h$ in (2.10) and is given by

\[
    f'(x) \equiv \frac{f(x+h) - f(x)}{h} \tag{2.11}
\]

Subtracting (2.11) from (2.10) we obtain the central difference approximation

\[
    f'(x) \equiv \frac{1}{2h} \left[ f(x+h) - f(x-h) \right]
\]

By retaining another term in the Taylor series, this type of analysis can be extended to derive the central difference approximation for $f''(x)$ as

\[
    f''(x) \equiv \frac{1}{h^2} \left[ f(x+h) - 2f(x) + f(x-h) \right]
\]

This analysis can be extended to partial derivatives giving

\[
    T_x(x,y) \equiv \frac{1}{2h} \left[ T(x+h,y) - T(x-h,y) \right]
\]

\[
    T_{xx}(x,y) \equiv \frac{1}{h^2} \left[ T(x+h,y) - 2T(x,y) + T(x-h,y) \right]
\]

\[
    T_y(x,y) \equiv \frac{1}{2k} \left[ T(x,y+k) - T(x,y-k) \right]
\]

\[
    T_{yy}(x,y) \equiv \frac{1}{k^2} \left[ T(x,y+k) - 2T(x,y) + T(x,y-k) \right]
\]

It is convenient to express the above equations in the following notation

\[
    T(x,y) = T_{i,j}
\]

\[
    T(x+h,y) = T_{i+1,j}
\]
\[ T(x-h, y) = T_{i-1,j} \]
\[ T(x, y+k) = T_{i,j+1} \]
\[ T(x, y-k) = T_{i,j-1} \]

Thus, rewrite the partial derivatives as

\[ T_x(x, y) = \frac{1}{2\Delta x} (T_{i+1,j} - T_{i-1,j}) \]
\[ T_y(x, y) = \frac{1}{2\Delta y} (T_{i,j+1} - T_{i,j-1}) \]
\[ T_{xx}(x, y) = \frac{1}{\Delta x^2} (T_{i+1,j} - 2T_{i,j} + T_{i-1,j}) \quad (2.12) \]
\[ T_{yy}(x, y) = \frac{1}{\Delta y^2} (T_{i,j+1} - 2T_{i,j} + T_{i,j-1}) \quad (2.13) \]

where \( h \) and \( k \) have been replaced with \( \Delta x \) and \( \Delta y \). To solve the heat conduction problem we replace the partial derivatives in

\[ T_{xx} + T_{yy} = -\frac{Q}{K} \]

with (2.12) and (2.13).

Doing this and taking \( \Delta x = \Delta y = \delta \) we have

\[ T_{i,j} = \frac{1}{4} \left( T_{i+1,j} + T_{i,j+1} + T_{i-1,j} + T_{i,j-1} + \frac{\delta^2 Q}{K} \right) \quad (2.14) \]

Here, \( T_{i,j} \) is a solution to an interior node. This interior node we have just analyzed is one of many, perhaps hundreds or thousands of nodes in a 2-D region. To solve for the temperature of every node in a domain, a separate finite difference equation must be derived for each node. These nodal equations make up a system of equations that have as many unknowns as the number of equations.
For illustrative purposes we will look at the same problem we solved analytically earlier. The problem is restated here and shown in Fig. 2.1.

\[ KV^2T + Q = 0 \]  \hspace{1cm} (2.15)

\[ \frac{\partial T}{\partial x} (0, y) = 0 \]  \hspace{1cm} (2.16)

\[ T(L, y) = T_L \]  \hspace{1cm} (2.17)

\[ \frac{\partial T}{\partial y} (x, 0) = 0 \]  \hspace{1cm} (2.18)

\[ T(x, L) = T_L \]  \hspace{1cm} (2.19)

We will divided the arbitrary domain into a 5x5 grid with equal spacing as shown in Fig. 2.3.
Next, we write a single linear equation of the form (2.14) for each of the nine interior grid points.

\[-T_{12} - T_{32} - T_{23} - T_{21} + 4T_{22} = \frac{\delta^2 Q}{K}\]

\[-T_{22} - T_{42} - T_{33} - T_{31} + 4T_{32} = \frac{\delta^2 Q}{K}\]

\[-T_{32} - T_{52} - T_{43} - T_{41} + 4T_{42} = \frac{\delta^2 Q}{K}\]

\[-T_{13} - T_{33} - T_{24} - T_{22} + 4T_{23} = \frac{\delta^2 Q}{K}\]
This system of equations can be solved by Gaussian elimination. There are 45
coefficients and since \( u \) is known at the boundary points we move these 12 terms to the
right hand side. After moving the terms we have 33 nonzero interior coefficients out of
81 in a 9x9 system. To show the sparse structure of the system of equation we will now
write the system in matrix form.

\[
AT = b
\]

Using natural ordering for the unknowns we have

\[
T = [T_{22}, T_{32}, T_{42}, T_{23}, T_{33}, T_{43}, T_{24}, T_{34}, T_{44}]^T
\]

The coefficient matrix is
$A = \begin{bmatrix}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 \\
-1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \\
\end{bmatrix}$

and the right hand side is

$\begin{bmatrix}
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\frac{\delta^2 Q}{K} \\
\end{bmatrix}$

Because the equations are similar in form, iterative methods can be use to solve the sparse system. One such solver is the Gauss-Seidel method (Gerald, 1978). Solvers and complete programs for solving finite difference problems can be found on the web and in the literature. Thus,
In implementing the finite difference method we discretized the region using a set of regularly spaced nodes. The spacing in the x-direction is uniform and equal to the spacing in the y-direction, which gives a square pattern for the nodal points. A rectangular pattern can also be easily implemented. However, using non-square patterns makes the problem considerably more difficult. Since non-orthogonal meshes are prohibitive it can be difficult to create a mesh for a domain that is not orthogonal. For irregular domains the boundary of the region can be approximated as closely as possible using an orthogonal mesh. Where the mesh does not fall on the boundary the boundary conditions must be incorporated into the surrounding nodes. In some cases the domain may be transformed from the geometric domain into the computational domain, see Fig. 2.4. As shown in Fig. 2.4, \(\mathcal{M}\) denotes the forward transform and \([\mathcal{M}]^{-1}\) denotes the inverse transform. Once in computational space the problem is easily solved by the computer. The biggest difficulty with transforming a domain into computational space is deriving the transform and inverse transforms.

\[
\frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} + \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{Q_{i,j}}{K} = 0
\]

(2.20)
Finite Volume Method

The next numerical method we will look at is the Finite Volume Method (FVM). The finite volume method is based on the conservation of a physical property such as thermal energy. Thus the finite volume method always starts with an integral conservation formulation. First, the governing conservation principle is stated in integral form. The integral form of the steady state heat equation over a fixed volume is

$$\Omega \nabla^2 T dV + \int_y \frac{Q}{K} dV = 0$$  \hspace{1cm} (2.21)

Using the mean value theorem of integrals the integral over $\frac{Q}{K}$ can be evaluated. Also the volume integral over $\nabla^2 T$ can be transformed to a surface integral by applying the divergence theorem (Özişik, 1993).
\[ \int_s \nabla T \cdot n dS + V \frac{Q}{K} \]

where

\[ \nabla T \cdot n = \frac{\partial T}{\partial n} \]

Here \( V \) denotes a small control volume

Similar to the finite difference method, the domain is subdivided into a regular mesh using non-overlapping control volumes. However, the node locations are located at the centers of each grid instead of at the intersections as seen in Fig. 2. 5

![Generic FVM Mesh](image)

**Fig. 2. 5** Generic FVM Mesh
The space between each grid line is denoted by $\Delta x$ and $\Delta y$. Each grid line makes up a control volume face, which lies midway between each node at $(i \pm 1/2, j)$ and $(i, j \pm 1/2)$.

Once the control volumes are defined, we can apply the conservation statement to a typical control volume. After applying equation (2.21) to the $(i, j)$ control volume we have

$$0 = -\int_{y-j/2}^{y+(j+1)/2} \frac{\partial T}{\partial x} dy - \int_{x-i/2}^{x+(i+1)/2} \frac{\partial T}{\partial y} dx + \int_{x-i/2}^{x+(i+1)/2} \frac{\partial T}{\partial x} dx + \int_{y-j/2}^{y+(j+1)/2} \frac{\partial T}{\partial y} dy + \int_{\Delta x \Delta y} \frac{Q}{K} dxdy$$

Here we assign unit depth, so the integrals are evaluated over the volumes $\Delta x\Delta y(1)$.

Using simple finite difference expressions, the heat balance on the control volume can be written as

$$\frac{T_{i,j} - T_{i-1,j}}{\Delta x} - \frac{T_{i,j} - T_{i,j+1}}{\Delta y} + \frac{T_{i+1,j} - T_{i,j}}{\Delta x} + \frac{T_{i,j+1} - T_{i,j}}{\Delta y} + \frac{Q}{K} \Delta x \Delta y = 0$$

Dividing by $\Delta x \Delta y$ and rearranging we get

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} + \int_{\Delta x \Delta y} \frac{Q}{K} dxdy = 0 \quad (2.22)$$

which looks like the finite difference expression, equation (2.20). Equation (2.22) can be solved in a similar manner to the finite difference method.

Boundary Element Method

The Boundary Element Method (BEM) is now being used as an alternative to the finite element method (Brebbia and Dominguez, 1989) for simple problem definitions. Studies have found that the BEM can yield accurate results on problems that involve
stress and fracture analysis. The BEM is also being used for domains that are infinite or where rapid prototyping is being used. When using boundary elements only the surface of the domain needs to be meshed. The method allows for easier mesh regeneration in the event that the geometry of the problem changes. Although it should be noted that in its ease of use one must be careful when developing codes, as the BEM is prone to numerical errors (Brebbia and Dominguez, 1989).

To illustrate the BEM we will again use Poisson's form of the heat equation given by

$$\nabla^2 T = -\frac{Q}{K}$$

and defined over the domain $\Omega$ with boundary $\Gamma$. A generic domain can be seen in Fig. 2.6

![Fig. 2.6 General Domain for BEM](image)

The boundary conditions are of two types: the 'essential' type $T$ on $\Gamma_1$ and the 'natural' type $\frac{\partial T}{\partial n}$ on $\Gamma_2$ (Brebbia and Dominguez, 1989). The boundary is given by $\Gamma = \Gamma_1 + \Gamma_2$.

Let

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ T - \overline{T} \neq 0 \]

\[ \frac{\partial T}{\partial n} - \frac{\partial \overline{T}}{\partial n} \neq 0 \]

be the residuals where \( \overline{T} \) and \( \frac{\partial \overline{T}}{\partial n} \) are approximations. Take \( W \) to be an arbitrary weighting function with derivatives on the boundary given by

\[ \frac{\partial W}{\partial n} \]

The governing equation in integral form is

\[
\int_{\Omega} \left( \nabla^2 T + \frac{Q}{K} \right) W d\Omega = \int_{\Gamma} \left( \frac{\partial T}{\partial n} - \frac{\partial \overline{T}}{\partial n} \right) W d\Gamma - \int_{\Gamma} (T - \overline{T}) \frac{\partial \overline{T}}{\partial n} d\Gamma \tag{2.23}
\]

Integrating equation (2.23) by parts once yields

\[
\int_{\Omega} \left( \frac{\partial T}{\partial x} \frac{\partial W}{\partial x} - \frac{\partial T}{\partial y} \frac{\partial W}{\partial y} + \frac{Q}{K} W \right) d\Omega = -\int_{\Gamma} \frac{\partial \overline{T}}{\partial n} W d\Gamma - \int_{\Gamma} \frac{\partial T}{\partial n} W d\Gamma \\
- \int_{\Gamma} T \frac{\partial \overline{T}}{\partial n} d\Gamma + \int_{\Gamma} \overline{T} \frac{\partial T}{\partial n} d\Gamma
\]

Integration by parts again gives

\[
\int_{\Omega} (\nabla^2 W) \, T d\Omega + \int_{\Omega} \frac{Q}{K} W d\Omega = -\int_{\Gamma} \frac{\partial \overline{T}}{\partial n} W d\Gamma - \int_{\Gamma} \frac{\partial T}{\partial n} W d\Gamma \\
+ \int_{\Gamma} T \frac{\partial \overline{T}}{\partial n} d\Gamma + \int_{\Gamma} \overline{T} \frac{\partial T}{\partial n} d\Gamma \tag{2.24}
\]

Equation (2.24) is the basic statement for the boundary element formulation of Poisson’s form of the heat equation. By passing the unknowns to the left hand side one can write

(Brebbia and Dominguez, 1989)

\[ KT = F \]
where $T$ is a vector of the unknowns.

The elements used in the boundary element method are similar to elements used in the finite element method. These elements include linear, bilinear, quadratic, and biquadratic. To illustrate the difference in meshes between the FDM and BEM, consider the domain shown in Fig. 2.7.

![Fig. 2.7 Example Problem for BEM](image)

A structured mesh is shown in Fig. 2.8.
The mesh for the domain is shown above using rows and columns common to FDM, FVM, and FEM (although the FEM permits unstructured elements). The mesh, using linear elements for the BEM, can be seen in Fig. 2.9.
The mesh for the BEM uses considerable less nodes than that of the FDM, FVM, or even the FEM. The interesting feature to notice is the absence of interior nodes – the BEM has transformed the solution integral from an area to a line, i.e. a 1-D problem. As one can imagine, there are considerable savings on the number of nodes and element for large domains. The accuracy of the BEM is as good as the FEM, and in some cases may provide the only solution.

Boundary Fitted Coordinates

To overcome some of the problems of irregular geometries one can use the method of boundary fitted coordinates (BFC). The method consists of transforming irregular
geometries into a rectangular computational domain. The computational domain may be composed of rectangular sub-domains. One begins by using a curvilinear coordinate system to create the mesh of the domain. The mesh should be smooth and a sufficiently fine grid should be used in areas where the solution has large variations.

To provide a foundation for the boundary fitted coordinate method we will look at a transformation of an annular ring domain in cylindrical coordinates (Thompson et al, 1985). In this case the curvilinear coordinate system is taken to be $r$ and $\theta$. The annular region and coordinate orientation can be seen in Fig. 2. 10.

![Fig. 2. 10 Domain in Curvilinear Coordinate](image-url)
The well known relationship between the curvilinear coordinates and the Cartesian coordinates is given by
\[ x(r, \theta) = r \cos \theta \]
\[ y(r, \theta) = r \sin \theta \]
The inverse transform is then defined as
\[ r(x, y) = \sqrt{x^2 + y^2} \]
\[ \theta(x, y) = \tan^{-1}\left( \frac{y}{x} \right) \]
Using the transform we can represent the annular ring as a rectangle. However, for the purpose of programming it is convenient to normalize the region. To normalize the curvilinear coordinates to an interval of \([0,1]\), we introduce the coordinates \(\xi\) and \(\eta\).

The relationship between the two coordinate systems is defined as
\[ \xi = \frac{\theta}{2\pi} \]
\[ \eta = \frac{r - r_1}{r_2 - r_1} \]
or
\[ \theta(\xi) = 2\pi \xi \]
\[ r(\eta) = r_1 + (r_2 - r_1) \eta \]
The transform can be rewritten as
\[ x(\xi, \eta) = (r_1 + (r_2 - r_1) \eta) \cos(2\pi \xi) \]
\[ y(\xi, \eta) = (r_1 + (r_2 - r_1) \eta) \sin(2\pi \xi) \]
where $0 \leq \xi \leq 1$ and $0 \leq \eta \leq 1$. The transformed domain is shown in Fig. 2.11

![Fig. 2.11 BFC Transformed Domain](image)

The relationship of interior grid points between the computational space and the physical domain can be expressed as

$$
\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = 0 
$$ 

(2.25)

$$
\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = 0 
$$ 

(2.26)

with appropriate boundary conditions. Equations (2.25) and (2.26) are in physical space, the corresponding equations in computational space are (Fletcher, 1991)
\[
\alpha \frac{\partial^2 x}{\partial \xi^2} - 2\beta \frac{\partial^2 x}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 x}{\partial \eta^2} = 0 
\]

(2.27)

\[
\alpha \frac{\partial^2 y}{\partial \xi^2} - 2\beta \frac{\partial^2 y}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 y}{\partial \eta^2} = 0 
\]

(2.28)

where

\[
\alpha = \left( \frac{\partial x}{\partial \xi} \right)^2 + \left( \frac{\partial y}{\partial \eta} \right)^2
\]

\[
\beta = \frac{\partial x}{\partial \xi} \frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi} \frac{\partial y}{\partial \eta}
\]

\[
\gamma = \left( \frac{\partial x}{\partial \xi} \right)^2 + \left( \frac{\partial y}{\partial \xi} \right)^2
\]

Once the boundary conditions are known in the computational domain, equations (2.27) and (2.28) can be solved using simple finite difference techniques. The concept of boundary fitted coordinates can be extended to 3-D regions of any shape. Although this method can be used on almost any geometry, it may be very difficult to derive the transform and its inverse.
Finite Element Method

The finite element method (FEM) is similar to the finite difference scheme in that it uses numerical approximation to provide solutions to physical problems. The finite element method dates back to the 1940's and the 1950's. However, there is general agreement that the term "finite element method" actually appeared in a paper by Clough in 1960 (Pepper and Heinrich, 1992). This is also the time when digital computers were starting to be used to solve matrix type problems. The early use of finite elements was aimed at problems of structural analysis. Starting in the 1970's the use of finite elements for non-structural problems began in earnest.

The governing equation we will be using for illustrating the finite element method is the Poisson equation for heat conduction with an internal source,

\[ \nabla^2 T = \frac{-Q}{K} \]  

We now look to formulate the "weak statement" for this problem using the method of weighted residuals (MWR). Greens theorem is implemented to reduce the problem from second order to first order over the domain. To apply the MWR the domain must be discretized into elements. For each of these elements the temperature will be approximated by using known functions \( N_i(x, y) \). The functions \( N_i(x, y) \) are called
shape functions. We can approximate the temperature distribution using the trial
approximation

\[ T(x, y) = \sum_{i=1}^{n} N_i(x, y) T_i \]  
(3. 2)

where \( n \) is the number of nodes in the element. We will use the Galerkin form of the
MWR to formulate the finite element method. In general, when using equation (3. 2) to
approximate the left hand side of equation (3. 1) a true solution is not achieved. The best
we can hope for is a "residual" function representing the error of the approximation. Let
the error for the heat conduction problem with constant conductivity be

\[ R(T, x_i) = -K \nabla^2 T - Q \]

Here \( T \) is an approximation to the analytical solution and cannot be made to equal the true
solution unless the error vanishes over the whole domain. Since the error cannot be made
to go to zero, we seek to minimize the error. To minimize the error we multiply by a
weighting function and force the integral to vanish as given by

\[ \int_{\Omega} W_i R(T, x_i) \, d\Omega = 0 \]

Galerkin Formulation

Now we will formulate the Galerkin weighted residual by setting the weight equal to
the interpolating function. As before, let

\[ \nabla^2 T = -\frac{Q}{K} \]

be defined over the domain \( \Omega \) with boundary \( \Gamma \). We will use the MWR as described
earlier (dropping subscript \( i \) for simplicity)
\[ \int_{\Omega} W R d\Omega = 0 \quad (3.3) \]

where
\[ R = -K \left[ \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right] - Q \quad (3.4) \]

and \( W \) is the weighting function. Rewriting equation (3.4) and substituting into equation (3.3) we have
\[ \int_{\Omega} W \left[ -K \nabla^2 T - Q \right] d\Omega = \int_{\Omega} \left[ -KW \nabla^2 T - WQ \right] d\Omega \]
\[ = \int_{\Omega} K \left[ -W \nabla^2 T \right] d\Omega - \int_{\Omega} WQ d\Omega \quad (3.5) \]

Using the well known properties of derivatives
\[ \nabla \cdot (W \nabla T) = \nabla W \cdot \nabla T + W \nabla^2 T \]

or
\[ -W \nabla^2 T = \nabla W \cdot \nabla T - \nabla \cdot (W \nabla T) \quad (3.6) \]

Substituting equation (3.6) into equation (3.5) we have
\[ \int_{\Omega} K \left[ \nabla W \cdot \nabla T - \nabla \cdot (W \nabla T) \right] d\Omega - \int_{\Omega} WQ d\Omega = 0 \]

or
\[ \int_{\Omega} K \nabla W \cdot \nabla T d\Omega - \int_{\Omega} K \nabla \cdot (W \nabla T) d\Omega - \int_{\Omega} WQ d\Omega \]

Recalling Gauss's theorem,
\[ \int_{\Omega} \nabla u d\Omega = \int_{\Gamma} u \cdot nd\Gamma \]

we have
\[ \int_{\Omega} K \nabla W \cdot \nabla T d\Omega - \int_{\Omega} WQ d\Omega - \int_{\Gamma} KW \nabla T \cdot nd\Gamma = 0 \]

or
\[
\left[\int_{\Omega} K \left( \nabla W \cdot \nabla T - W Q \right) d\Omega + \int_{\Gamma} W \left( -K \frac{\partial T}{\partial n} \right) d\Gamma \right] = 0
\]  
(3.7)

We now approximate the temperature over the domain using the trial approximation defined by equation (3.2). For the Galerkin formulation we set \( W_i = N_i \). Thus equation (3.7) is

\[
\sum_{j=1}^{\mathcal{M}} \left[ \int_{\Omega} K \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \right] T_j = \int_{\Omega} N_i Q d\Omega - \int_{\Gamma} N_i \left( K \frac{\partial T}{\partial n} \right) d\Gamma
\]  
(3.8)

Equation (3.8) can be expressed in matrix notation as

\[
\mathbf{K} \mathbf{T} = \mathbf{F}
\]

where

\[
\mathbf{K} = \left[ k_{ij} \right] = \int_{\Omega} K \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega
\]

and

\[
\mathbf{F} = \left[ f_i \right] = \int_{\Omega} N_i Q d\Omega - \int_{\Gamma} N_i q d\Gamma
\]

where

\[
q = K \frac{\partial T}{\partial n}
\]

The \( \mathbf{K} \) matrix is commonly called the stiffness matrix while \( \mathbf{F} \) is called the load vector (Pepper and Heinrich. 1992). If advection is present, the matrix form can be expressed as

\[
(\mathbf{U} + \mathbf{K}) \mathbf{T} = \mathbf{F}
\]

where

\[
\mathbf{U} = \left[ u_i \right] = \int_{\Omega} N_i \left[ \left( N_k u_k \right) \frac{\partial N_i}{\partial x} + \left( N_k v_k \right) \frac{\partial N_i}{\partial y} \right]
\]
The functions \( u \) and \( v \) have been interpolated using the shape functions, and summation is implied by the index \( k \).

Elements

The first type of element we will discuss is the three node triangular element. Although there are different ways to number the nodes, the general practice is to number the nodes of an element in the counterclockwise order (Pepper and Heinrich, 1992). We will denote the node locations as shown in Fig. 3.1.

![Three Node Triangular Element](image.png)

Fig. 3.1  Three Node Triangular Element

The linear triangular element is represented by the following polynomial

\[
\gamma = \alpha_1 + \alpha_2 x + \alpha_3 y
\]
The linear shape functions for the triangular element in $x$ and $y$ coordinates are

\begin{align*}
N_1 &= \frac{1}{2A} \left[ (x_2 y_3 - x_3 y_2) + (y_2 - y_3) x + (x_3 - x_2) y \right] \\
N_2 &= \frac{1}{2A} \left[ (x_3 y_1 - x_1 y_3) + (y_3 - y_1) x + (x_1 - x_3) y \right] \\
N_3 &= \frac{1}{2A} \left[ (x_1 y_2 - x_2 y_1) + (y_1 - y_2) x + (x_2 - x_1) y \right]
\end{align*}

where

\[ A = \frac{1}{2} \left[ (x_1 y_2 - x_2 y_1) + (x_3 y_1 - x_1 y_3) + (x_2 y_3 - x_3 y_2) \right] \]

For computational purposes it is convenient to rewrite the shape functions into a natural coordinate system. The natural coordinate variables are defined as

\begin{align*}
\xi_1 &= \frac{A_1}{A} \\
\xi_2 &= \frac{A_2}{A} \\
\xi_3 &= \frac{A_3}{A}
\end{align*}

where $A_1$, $A_2$, and $A_3$ are the areas as shown in Fig. 3.2. The coordinates $x$ and $y$ are related to the coordinates $\xi_1$, $\xi_2$, and $\xi_3$ by

\begin{align*}
x &= \xi_1 x_1 + \xi_2 x_2 + \xi_3 x_3 \\
y &= \xi_1 y_1 + \xi_2 y_2 + \xi_3 y_3 \\
1 &= \xi_1 + \xi_2 + \xi_3
\end{align*}
Fig. 3.2 Triangular Element Areas

The shape functions in the natural coordinate system are defined as

\[ N_1 = \xi_1 \]
\[ N_2 = \xi_2 \]
\[ N_3 = \xi_3 \]

The derivatives of the shape function can be calculated using the chain rule.

The quadratic triangular element is made up of six nodes as shown in Fig. 3.3. The polynomial representing the six node triangle is an extension of the linear interpolating polynomial and is given by

\[ \gamma = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 xy + \alpha_6 y^2 \]
The shape functions in the natural coordinate system are given by

\[ N_1 = \xi_1 (2\xi_1 - 1) \]
\[ N_2 = \xi_2 (2\xi_2 - 1) \]
\[ N_3 = \xi_3 (2\xi_3 - 1) \]
\[ N_4 = 4\xi_2\xi_3 \]
\[ N_5 = 4\xi_1\xi_3 \]
\[ N_6 = 4\xi_1\xi_2 \]

The derivatives for the shape functions are calculated in the same manner as those for the linear triangle.
The second type of element commonly used in the finite element method is the quadrilateral. The quadrilateral element is a four sided polygon usually consisting of four, eight, or nine nodes. Typically the higher the order of the element the more accurate the results, but such elements consume substantially more computational resources. We will discuss the four and eight node elements.

The four node quadrilateral element and node numbering are shown in Fig. 3.4

\[ y = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy \]

Using the relation given by
\[ x = x_1 N_1 + x_2 N_2 + x_3 N_3 + x_4 N_4 \]

\[ y = y_1 N_1 + y_2 N_2 + y_3 N_3 + y_4 N_4 \]

the element can be transformed into the natural coordinate system as shown in Fig. 3.5 (Pepper and Heinrich, 1992). Hence,

\[ N_1 = \frac{1}{4} (1 - \xi)(1 - \eta) \]
\[ N_2 = \frac{1}{4} (1 + \xi)(1 - \eta) \]
\[ N_3 = \frac{1}{4} (1 + \xi)(1 + \eta) \]
\[ N_4 = \frac{1}{4} (1 - \xi)(1 + \eta) \]

where \(-1 \leq \xi \leq 1\) and \(-1 \leq \eta \leq 1\).

Fig. 3.5 Quadrilateral Element in Natural Coordinates

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The calculation of the derivative is analogous to the triangular element.

The eight node quadratic element and node numbering are shown in Fig. 3.6.

Fig. 3.6 Eight Node Quadratic Element

The shape functions for the eight node quadratic are as follows:

\[ N_1 = -\frac{(1-\xi)(1-\eta)(1+\xi+\eta)}{4} \]

\[ N_2 = \frac{(1-\xi^2)(1-\eta)}{2} \]
\[ N_3 = \frac{(1+\xi)(1-\eta)(\xi-\eta-1)}{4} \]

\[ N_4 = \frac{(1-\eta^2)(1+\xi)}{2} \]

\[ N_5 = \frac{(1+\xi)(1+\eta)(\xi+\eta-1)}{4} \]

\[ N_6 = \frac{(1-\xi^2)(1+\eta)}{2} \]

\[ N_7 = -\frac{(1-\xi)(1+\eta)(1+\xi-\eta)}{4} \]

\[ N_8 = \frac{(1-\eta^2)(1-\xi)}{2} \]

where \(-1 \leq \xi \leq 1\) and \(-1 \leq \eta \leq 1\). Again, the calculation of the derivatives is similar to that of the triangular element.
NOTE TO USERS

Page(s) missing in number only; text follows. Page(s) were microfilmed as received.

39

This reproduction is the best copy available.

UMI
CHAPTER 4

Java History

Java stems from Sun Microsystems engineers Patrick Naughton and James Gosling who wanted to develop a language that could be used for consumer electronics in 1991. The language was needed to generate compact code that would run on different central processing units (CPUs) with little memory. This project was dubbed the “Green” Project (Horstmann, 1999). The Green project engineers designed a portable language that uses intermediate code that is interpreted by a virtual machine. This code could run on any hardware that had the proper interpreter. Since the code is machine independent a programmer can be confident that a program developed in Java will run on any hardware or platform, provided it has the correct interpreter.

Sun Microsystems first release of Java was in 1996. The release was followed by Java 1.02a shortly after the initial release. It was found that these versions of Java were not able to produce serious applications. The first major revision was Java 1.1. Although Java 1.1 was a major step in the right direction, it was the release of Java 1.2 that provided the means to produce sophisticated code. Release 1.2 provided a wealth of new components. These components allow for greater flexibility in programming Applets.
Applet

Java programs that run on the web are called applets. Byte-codes are downloaded form the Internet and then run on the client’s system. For an applet to work the user must have a Java-enabled web browser to interpret the byte-codes. Applets provide a versatile way to present programs and data across the Internet. Applets can be used to add active content to web pages. Also, applets can be used to convey pseudo-applications. A pseudo-applications is an applet that looks and feels like a stand-alone application, but does not have the functionality of an application.

There are some drawbacks to using applets. For example, downloading applets over a dial-up connection may be very slow. Also, problems can arise when web browsers do not run the most current version of Java. Sun, the creators of Java, have provided a solution to the problem called a Java Plug-in. Java Plug-in is software that allows the programmer to direct applets on their web pages to run Sun’s Java Runtime Environment (JRE) instead of the web browsers default Java virtual machine. The current release provides support for Microsoft Internet Explorer and Netscape Navigator on various Win32 platforms. There is also support for Compaq Tru64™ Unix(r), Linux, and HPUX. Work is under way to support other platforms.

Using the following HTML tag one can automatically force the browser to download the correct Java Plug-in if it is not currently installed on the system viewing the applet. This ensures that a client will be able to correctly view the applet. To use Java Plug-in in IE on Windows 95, Windows 98 or Windows NT 4.0, use the OBJECT tag. The following is an example of mapping an APPLET tag to a Java Plug-in tag:

Original APPLET tag:
<APPLET code="HeatXfer.class" codebase="html/"
align="baseline"
width="200" height="200">
<PARAM NAME="model" VALUE="models/HeatXfer.xyz">
No Java 2 SDK, Standard Edition v 1.3 support for APPLET!!
</APPLET>

New OBJECT tag:

<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="200" height="200" align="baseline"
codebase="http://Java.sun.com/products/plugin/1.3/jinstall-13-win32.cab#Version=1,3,0,0">
<PARAM NAME="code" VALUE="HeatXfer.class">
<PARAM NAME="codebase" VALUE="html/">
<PARAM NAME="type" VALUE="application/x-Java-applet;version=1.3">
<PARAM NAME="model" VALUE="models/models/HeatXfer.xyz">
<PARAM NAME="scriptable" VALUE="true">
No Java 2 SDK, Standard Edition v 1.3 support for APPLET!!
</OBJECT>

To use Java Plug-in in Netscape Navigator 3 or 4 on Windows 95, Windows 98,
Windows NT 4.0, or Solaris operating environments, use the EMBED tag. The following
example maps an APPLET tag to a Java Plug-in EMBED tag:

Original APPLET tag:

<APPLET code="HeatXfer.class" codebase="html/"
align="baseline"
width="200" height="200">
<PARAM NAME="model" VALUE="models/models/HeatXfer.xyz">
No Java 2 SDK, Standard Edition v 1.3 support for APPLET!!
</APPLET>

New EMBED tag:

<EMBED type="application/x-Java-applet;version=1.3"
width="200"
height="200" align="baseline" code="HeatXfer.class"
codebase="html/" model="models/models/HeatXfer.xyz"
pluginspage="http://Java.sun.com/products/plugin/1.3/plugin-install.html"
There are other drawbacks in using applets. First, because of the inherent security associated with applets one cannot access a client's system with an applet. This means that a client cannot upload a data file to the solver via direct access to the client's hard drive. Second, there isn't a readily available way to directly access the server's hard drive to save information to the server for later use. Lastly, since the applet physically runs on the client system there is a possibility for the client system to become overloaded. For example, if the client system is an old computer with little memory the system may bog down while trying to solve a large problem.

Abstract Window Toolkit

To develop applets and applications that use graphical presentations one must use the Abstract Window Toolkit (AWT) (Geary, 1999). The AWT is part of the Java Development Kit (JDK), which is distributed free from Sun Microsystems. The AWT components are platform independent and supply basic user interface components, such as lists, menus, buttons, text fields, etc. AWT also has event handling, data transfer, and image manipulation.

AWT makes use of classes known as peers, which are the client's Graphical User Interface (GUI) components manipulated by the classes. For example, suppose an instance of the menu class is created causing the Java runtime to intern create an instance of menu peer. For the case of Solaris JDK would create a Motif menu peer while Microsoft windows JDK version would create a window specific menu peer. Since AWT
is rendered in their clients native window, they are called heavy weight components. These heavyweight components consume the clients resources and are difficult to subclass in order to modify their default behavior.

Swing

Swing was built on top of the AWT infrastructure (Geary, 1999). Swing overcomes some of the problems with heavyweight components by providing lightweight components. There are lightweight replacement for AWT heavyweight components and at the same time provides a wealth of new components. Also, Swing provides support for pluggable look and feel were the look and feel of a window can be changed to MS Windows, Metal, Motif, or Macintosh. Pluggable look and feel, if incorporated, allow the user to change the look and feel on the fly.

Swing was released as part of the Java Foundation Class (JFC). JFC is part of the Java Development Kit (JDK) 1.2 released in December of 1998. Since Swing components are enhancements to the existing AWT components, both AWT and Swing components can be used in the same interface. Fig. 4.1 shows how JFC and Swing are related to JDK 1.2 (also called Java 2). Swing is the primary component used for the development of the GUI in this thesis.
Java – Fortran Comparison

The code bit listed below is a method for the program written for this thesis. The code that operates on data is called a Method. Methods are like subroutines or functions in Fortran. In Java methods are public, private, or protected. This allows the programmer to set access control on how a method is used. Each method is contained in a class. One of the main differences between Fortran and Java is that Java is an object oriented programming language (OOP). This means that everything in Java, with a few exceptions, is an object. Object oriented design allows for easier development of sophisticated projects. Also, unlike Fortran, Java is designed to integrate with graphical user interfaces. The graphical user interface can be stand-alone applications or applets.
design to run on the Internet. Although Java is a better choice for programming applications it does have one major disadvantage. The disadvantage is that Java at the present time does not supply a robust mathematical component. The mathematical component is sparse compared to Fortran. There are considerable more differences that are not addressed in this thesis.

```java
public void ASSEMB(){
    int l, il, i;
    int JSTR, JEND, NGI;
    double RHS;
    double B[] = new double[maxNode];
    double FIXED[] = new double[maxNode];
    CLEAR(B, NNODE);

    // Construct RHS
    for(l = 1; l <= NNODE; l++){
        B[l] = F[l];
        JSTR = JMIN[l];
        JEND = JMAX[l];

        for(il = JSTR; il <= JEND; il++){
            NGI = NFR[l] + il - JSTR;
            RHS = (AFM*C[NGI] + P[NGI]/DT)*COLD[il];
        }
    }

    for(i = 1; i <= NNST; i++)
        FIXED[i] = COLD[NTS[i]];

    //private void SKYLIN(double B[], double VAR[], int NB[], double FIXED[], int NBOUND)
    SKYLIN(B, CNEW, NTS, FIXED, NNST);
} // End ASSEMB()
```

Below is the same code bit from the Fortran program.

```fortran
SUBROUTINE ASSEMB
COMMON/VA/COLD(201),CNEW(201)
COMMON/VK/NNODE,NELEM,NTYPE,NTIME,TIME,NFRMAX
```
COMMON/VC/Q(201), F(201), NTS(201), NQS(201), DX(201), DY(201)

COMMON/VD/AF, AFM, DT, NNQS, NUMN, NUM, NGAUS, NNHC, NNST, NSTOP, KPR
   NT, IAXI
   COMMON/SK/NFR(201), JMIN(201), JMAX(201)
   DIMENSION B(201), FIXED(201)

C
   CALL CLEAR(B, NNODE)
C
   CONSTRUCT RHS
C
   DO 4 L=1, NNODE
   B(L)=F(L)
   JSTR=JMIN(L)
   JEND=JMAX(L)
   DO 4 I1=JSTR, JEND
   NGI=NFR(L)+I1-JSTR
   RHS=(AFM*C(NGI)+P(NGI)/DT)*COLD(I1)
   4 B(L)=B(L)+RHS
C
   DO 5 I=1, NNST
   5 FIXED(I)=COLD(NTS(I))
C
   CALL SKYLIN(B, CNEW, NTS, FIXED, NNST)
C
   RETURN
   RETURN
   END

The Fortran and Java version of the heat transfer code produces almost the identical results. The results of the Java versus Fortran comparison for the analytical test problem presented in Chapter 5 can be found in Table 4.1 and Table 4.2.
Java - FORTRAN 77 Comparison

<table>
<thead>
<tr>
<th>Node</th>
<th>FEM 3 Node (Java)</th>
<th>FEM 4 Node (Java)</th>
<th>FEM 6 Node (Java)</th>
<th>FEM 8 Node (Java)</th>
<th>FEM 3 Node (FORTRAN 77)</th>
<th>FEM 4 Node (FORTRAN 77)</th>
<th>FEM 6 Node (FORTRAN 77)</th>
<th>FEM 8 Node (FORTRAN 77)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3012</td>
<td>0.2982</td>
<td>0.2945</td>
<td>0.2945</td>
<td>0.3012</td>
<td>0.2982</td>
<td>0.2945</td>
<td>0.2945</td>
</tr>
<tr>
<td>2</td>
<td>0.2804</td>
<td>0.2823</td>
<td>0.2787</td>
<td>0.2787</td>
<td>0.2804</td>
<td>0.2823</td>
<td>0.2787</td>
<td>0.2787</td>
</tr>
<tr>
<td>3</td>
<td>0.2291</td>
<td>0.2321</td>
<td>0.2292</td>
<td>0.2292</td>
<td>0.2291</td>
<td>0.2321</td>
<td>0.2292</td>
<td>0.2292</td>
</tr>
<tr>
<td>4</td>
<td>0.1392</td>
<td>0.1413</td>
<td>0.1396</td>
<td>0.1396</td>
<td>0.1392</td>
<td>0.1413</td>
<td>0.1397</td>
<td>0.1396</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6</td>
<td>0.2644</td>
<td>0.2674</td>
<td>0.2640</td>
<td>0.2640</td>
<td>0.2644</td>
<td>0.2674</td>
<td>0.2640</td>
<td>0.2640</td>
</tr>
<tr>
<td>7</td>
<td>0.2171</td>
<td>0.2205</td>
<td>0.2176</td>
<td>0.2177</td>
<td>0.2171</td>
<td>0.2205</td>
<td>0.2177</td>
<td>0.2177</td>
</tr>
<tr>
<td>8</td>
<td>0.1326</td>
<td>0.1350</td>
<td>0.1332</td>
<td>0.1332</td>
<td>0.1326</td>
<td>0.1350</td>
<td>0.1333</td>
<td>0.1332</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.1800</td>
<td>0.1837</td>
<td>0.1811</td>
<td>0.1810</td>
<td>0.1800</td>
<td>0.1837</td>
<td>0.1811</td>
<td>0.1810</td>
</tr>
<tr>
<td>11</td>
<td>0.1116</td>
<td>0.1145</td>
<td>0.1127</td>
<td>0.1127</td>
<td>0.1116</td>
<td>0.1145</td>
<td>0.1127</td>
<td>0.1127</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>13</td>
<td>0.0714</td>
<td>0.0750</td>
<td>0.0729</td>
<td>0.0724</td>
<td>0.0714</td>
<td>0.0750</td>
<td>0.0729</td>
<td>0.0724</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 4.1 Java vs. Fortran Comparison for the Analytic Test Problem

Also, the runtime for the Java and Fortran programs are comparable. If one is interested in learning to program in Java, there are many good books on the subject. One such set of books is the Sun Microsystems series published by Prentice Hall.
### Java - FORTRAN 77 Comparison

<table>
<thead>
<tr>
<th>Node</th>
<th>Absolute Difference 3 Node</th>
<th>Absolute Difference 4 Node</th>
<th>Absolute Difference 6 Node</th>
<th>Absolute Difference 8 Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>4</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>7</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>8</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>11</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>13</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 4.2 Java vs. Fortran Difference for the Analytic Test Problem

### Finite Element Program

The finite element solver was programmed using Java 1.2 and has been "compiled" on Microsoft Windows; 98, NT, NT Server, 2000 Professional, and 2000 Server. Also, it has been "compiled" on Linux versions 5.2 and 6.1. The program uses the Galerkin MWR as a basis for the solver. It can solve problems using 3 node triangles, 6 node triangles, 4 node quadrilaterals, and 8 node quadrilaterals. The maximum number of elements is currently 1000, but this can be easily changed. The solver is currently 2000 plus lines of Java code.
The GUI is written using Swing components and has been tested on Internet Explorer 4.0 and above. Also, it has been tested on Netscape 4.0 and above. The HTML code is written to specify that the browsers use the Java 1.3 Plug-in. If the browser is not currently using the 1.3 Plug-in, the browser is directed to the Sun Microsystems web site to download and install it.

The GUI is displayed in a separate window form that of the web browser and can be scaled to the user’s liking. The GUI consists of three tabbed panels. The first panel is used to displays the mesh and the boundary conditions associated with the problem. The second panel displays the solution contour lines overlaid on the mesh. The user can change the number of contour lines from 10 to 75 lines. The last panel gives the nodal temperatures overlaid on the mesh. Screen shots of the program can be found in the appendices.
CHAPTER 5

Numerical Results

We will use the following problem to compare the FDM and FEM to the analytical solution. The solution of the two methods will be compared to the analytical solution.

The problem to be solved is shown in Fig. 5.1.
Next the domain is meshed as seen in Fig. 5. 2.

Fig. 5. 2 Symmetry of Analytic Test Problem

Using symmetry we can simplify the problem as shown in Fig. 5. 3.
Fig. 5.3 Mesh for Analytic Test Problem
The plot of the analytical solution using Mathematica is shown in Fig. 5.4.

Fig. 5.4 Plot of Analytic Test Problem

The numerical data from the test problem is given in the following tables.
## Solution Comparison

<table>
<thead>
<tr>
<th>Node</th>
<th>FDM</th>
<th>FEM 3 Node (Java)</th>
<th>FEM 4 Node (Java)</th>
<th>FEM 6 Node (Java)</th>
<th>FEM 8 Node (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2911</td>
<td>0.3012</td>
<td>0.2982</td>
<td>0.2945</td>
<td>0.2945</td>
</tr>
<tr>
<td>2</td>
<td>0.2755</td>
<td>0.2804</td>
<td>0.2823</td>
<td>0.2787</td>
<td>0.2787</td>
</tr>
<tr>
<td>3</td>
<td>0.2266</td>
<td>0.2291</td>
<td>0.2321</td>
<td>0.2292</td>
<td>0.2292</td>
</tr>
<tr>
<td>4</td>
<td>0.1381</td>
<td>0.1392</td>
<td>0.1413</td>
<td>0.1396</td>
<td>0.1396</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6</td>
<td>0.2609</td>
<td>0.2644</td>
<td>0.2674</td>
<td>0.2640</td>
<td>0.2640</td>
</tr>
<tr>
<td>7</td>
<td>0.2151</td>
<td>0.2171</td>
<td>0.2205</td>
<td>0.2176</td>
<td>0.2177</td>
</tr>
<tr>
<td>8</td>
<td>0.1317</td>
<td>0.1326</td>
<td>0.1350</td>
<td>0.1332</td>
<td>0.1332</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.1787</td>
<td>0.1800</td>
<td>0.1837</td>
<td>0.1811</td>
<td>0.1810</td>
</tr>
<tr>
<td>11</td>
<td>0.1110</td>
<td>0.1116</td>
<td>0.1145</td>
<td>0.1127</td>
<td>0.1127</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>13</td>
<td>0.0711</td>
<td>0.0714</td>
<td>0.0750</td>
<td>0.0729</td>
<td>0.0724</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 5.1 Solution Comparison of Analytic Test Problem

## Finite Element Method 3 Node (Java)

<table>
<thead>
<tr>
<th>Node</th>
<th>Finite Element</th>
<th>Exact</th>
<th>Absolute Error</th>
<th>Percent Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3012</td>
<td>0.2947</td>
<td>0.0065</td>
<td>2.2056</td>
</tr>
<tr>
<td>2</td>
<td>0.2804</td>
<td>0.2789</td>
<td>0.0015</td>
<td>0.5378</td>
</tr>
<tr>
<td>3</td>
<td>0.2291</td>
<td>0.2293</td>
<td>0.0002</td>
<td>0.0872</td>
</tr>
<tr>
<td>4</td>
<td>0.1392</td>
<td>0.1397</td>
<td>0.0005</td>
<td>0.3579</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6</td>
<td>0.2644</td>
<td>0.2641</td>
<td>0.0003</td>
<td>0.1136</td>
</tr>
<tr>
<td>7</td>
<td>0.2171</td>
<td>0.2178</td>
<td>0.0007</td>
<td>0.3214</td>
</tr>
<tr>
<td>8</td>
<td>0.1326</td>
<td>0.1333</td>
<td>0.0007</td>
<td>0.5251</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.1800</td>
<td>0.1811</td>
<td>0.0011</td>
<td>0.6074</td>
</tr>
<tr>
<td>11</td>
<td>0.1116</td>
<td>0.1127</td>
<td>0.0011</td>
<td>0.9760</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>13</td>
<td>0.0714</td>
<td>0.0728</td>
<td>0.0014</td>
<td>1.9500</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 5.2 3 Node Solution of Analytic Test Problem

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
### Finite Element Method 4 Node (Java)

<table>
<thead>
<tr>
<th>Node</th>
<th>Finite Element</th>
<th>Exact</th>
<th>Absolute Error</th>
<th>Percent Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2982</td>
<td>0.2947</td>
<td>0.0035</td>
<td>1.19</td>
</tr>
<tr>
<td>2</td>
<td>0.2823</td>
<td>0.2789</td>
<td>0.0034</td>
<td>1.22</td>
</tr>
<tr>
<td>3</td>
<td>0.2321</td>
<td>0.2293</td>
<td>0.0028</td>
<td>1.22</td>
</tr>
<tr>
<td>4</td>
<td>0.1413</td>
<td>0.1397</td>
<td>0.0016</td>
<td>1.15</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>0.2674</td>
<td>0.2641</td>
<td>0.0033</td>
<td>1.25</td>
</tr>
<tr>
<td>7</td>
<td>0.2205</td>
<td>0.2178</td>
<td>0.0027</td>
<td>1.24</td>
</tr>
<tr>
<td>8</td>
<td>0.1350</td>
<td>0.1333</td>
<td>0.0017</td>
<td>1.28</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>0.1837</td>
<td>0.1811</td>
<td>0.0026</td>
<td>1.44</td>
</tr>
<tr>
<td>11</td>
<td>0.1145</td>
<td>0.1127</td>
<td>0.0018</td>
<td>1.60</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>13</td>
<td>0.0750</td>
<td>0.0728</td>
<td>0.0022</td>
<td>2.99</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 5.3 4 Node Solution of Analytic Test Problem

### Finite Element Method 6 Node (Java)

<table>
<thead>
<tr>
<th>Node</th>
<th>Finite Element</th>
<th>Exact</th>
<th>Absolute Error</th>
<th>Percent Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2945</td>
<td>0.2947</td>
<td>0.0002</td>
<td>0.0679</td>
</tr>
<tr>
<td>2</td>
<td>0.2787</td>
<td>0.2789</td>
<td>0.0002</td>
<td>0.0717</td>
</tr>
<tr>
<td>3</td>
<td>0.2292</td>
<td>0.2293</td>
<td>0.0001</td>
<td>0.0436</td>
</tr>
<tr>
<td>4</td>
<td>0.1396</td>
<td>0.1397</td>
<td>0.0001</td>
<td>0.0716</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6</td>
<td>0.2640</td>
<td>0.2641</td>
<td>0.0001</td>
<td>0.0379</td>
</tr>
<tr>
<td>7</td>
<td>0.2176</td>
<td>0.2178</td>
<td>0.0002</td>
<td>0.0918</td>
</tr>
<tr>
<td>8</td>
<td>0.1332</td>
<td>0.1333</td>
<td>0.0001</td>
<td>0.0750</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.1811</td>
<td>0.1811</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>11</td>
<td>0.1127</td>
<td>0.1127</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>13</td>
<td>0.0729</td>
<td>0.0728</td>
<td>0.0001</td>
<td>0.1099</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 5.4 6 Node Solution of Analytic Test Problem

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Finite Element Method 8 Node (Java)

<table>
<thead>
<tr>
<th>Node</th>
<th>Finite Element</th>
<th>Exact</th>
<th>Absolute Error</th>
<th>Percent Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2945</td>
<td>0.2947</td>
<td>0.0002</td>
<td>0.0679</td>
</tr>
<tr>
<td>2</td>
<td>0.2787</td>
<td>0.2789</td>
<td>0.0002</td>
<td>0.0717</td>
</tr>
<tr>
<td>3</td>
<td>0.2292</td>
<td>0.2293</td>
<td>0.0001</td>
<td>0.0436</td>
</tr>
<tr>
<td>4</td>
<td>0.1396</td>
<td>0.1397</td>
<td>0.0001</td>
<td>0.0716</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6</td>
<td>0.2640</td>
<td>0.2641</td>
<td>0.0001</td>
<td>0.0379</td>
</tr>
<tr>
<td>7</td>
<td>0.2177</td>
<td>0.2178</td>
<td>0.0001</td>
<td>0.0459</td>
</tr>
<tr>
<td>8</td>
<td>0.1332</td>
<td>0.1333</td>
<td>0.0001</td>
<td>0.0750</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.1810</td>
<td>0.1811</td>
<td>0.0001</td>
<td>0.0552</td>
</tr>
<tr>
<td>11</td>
<td>0.1127</td>
<td>0.1127</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>13</td>
<td>0.0724</td>
<td>0.0728</td>
<td>0.0004</td>
<td>2.34</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 5.5 8 Node Solution of Analytic Test Problem

Finite Difference Method

<table>
<thead>
<tr>
<th>Node</th>
<th>Finite Difference</th>
<th>Exact</th>
<th>Absolute Error</th>
<th>Percent Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2911</td>
<td>0.2947</td>
<td>0.0036</td>
<td>1.22</td>
</tr>
<tr>
<td>2</td>
<td>0.2755</td>
<td>0.2789</td>
<td>0.0034</td>
<td>1.22</td>
</tr>
<tr>
<td>3</td>
<td>0.2266</td>
<td>0.2293</td>
<td>0.0027</td>
<td>1.18</td>
</tr>
<tr>
<td>4</td>
<td>0.1381</td>
<td>0.1397</td>
<td>0.0016</td>
<td>1.15</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>0.2609</td>
<td>0.2641</td>
<td>0.0032</td>
<td>1.21</td>
</tr>
<tr>
<td>7</td>
<td>0.2151</td>
<td>0.2178</td>
<td>0.0027</td>
<td>1.24</td>
</tr>
<tr>
<td>8</td>
<td>0.1317</td>
<td>0.1333</td>
<td>0.0016</td>
<td>1.20</td>
</tr>
<tr>
<td>9</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>0.1787</td>
<td>0.1811</td>
<td>0.0024</td>
<td>1.33</td>
</tr>
<tr>
<td>11</td>
<td>0.1110</td>
<td>0.1127</td>
<td>0.0017</td>
<td>1.51</td>
</tr>
<tr>
<td>12</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>13</td>
<td>0.0711</td>
<td>0.0728</td>
<td>0.0017</td>
<td>2.34</td>
</tr>
<tr>
<td>14</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 5.6 FDM Solution of Analytic Test Problem
### Percent Error Comparison

<table>
<thead>
<tr>
<th>Node</th>
<th>FDM</th>
<th>FEM 3 Node (Java)</th>
<th>FEM 4 Node (Java)</th>
<th>FEM 6 Node (Java)</th>
<th>FEM 8 Node (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.222</td>
<td>2.206</td>
<td>1.188</td>
<td>0.068</td>
<td>0.068</td>
</tr>
<tr>
<td>2</td>
<td>1.219</td>
<td>0.538</td>
<td>1.219</td>
<td>0.072</td>
<td>0.072</td>
</tr>
<tr>
<td>3</td>
<td>1.177</td>
<td>0.087</td>
<td>1.221</td>
<td>0.044</td>
<td>0.044</td>
</tr>
<tr>
<td>4</td>
<td>1.145</td>
<td>0.358</td>
<td>1.145</td>
<td>0.072</td>
<td>0.072</td>
</tr>
<tr>
<td>5</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>1.212</td>
<td>0.114</td>
<td>1.250</td>
<td>0.038</td>
<td>0.038</td>
</tr>
<tr>
<td>7</td>
<td>1.240</td>
<td>0.321</td>
<td>1.240</td>
<td>0.092</td>
<td>0.046</td>
</tr>
<tr>
<td>8</td>
<td>1.200</td>
<td>0.525</td>
<td>1.275</td>
<td>0.075</td>
<td>0.075</td>
</tr>
<tr>
<td>9</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>10</td>
<td>1.325</td>
<td>0.607</td>
<td>1.436</td>
<td>0.000</td>
<td>0.055</td>
</tr>
<tr>
<td>11</td>
<td>1.508</td>
<td>0.976</td>
<td>1.597</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>12</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>13</td>
<td>2.335</td>
<td>1.950</td>
<td>2.994</td>
<td>0.110</td>
<td>0.577</td>
</tr>
<tr>
<td>14</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>15</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 5.7  Percent Error Comparison of Analytic Test Problem

The nodal temperature distributions are shown graphically in Fig. 5.5. The percent error for each node can be found in Fig. 5.6.
Fig. 5.5 Solution Values of Analytic Test Problem
Fig. 5.5 Solution Values of Analytic Test Problem
In the next test case we look at the analysis of a furnace wall (Incropera and DeWitt, 2001) shown in Fig. 5. 7.
The temperature distribution in the brick wall will be found using the program created for this thesis and a FEM program distributed by Incropera and DeWitt (2001). Using symmetry the problem can be simplified as shown in Fig. 5. 8.
Next the domain is meshed using triangular elements as seen in Fig. 5.9 (based on mesh configuration in Incropera and DeWitt, 2000).

Fig. 5.9 Mesh of Brick Wall Test Problem
The temperature values from the Incropera and DeWitt finite element program and the Java based program are listed in Table 5.8.

<table>
<thead>
<tr>
<th>Node</th>
<th>Temp Incropera and DeWitt</th>
<th>Temp Thesis</th>
<th>Absolute Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>530.00</td>
<td>532.70</td>
<td>2.70</td>
</tr>
<tr>
<td>2</td>
<td>306.00</td>
<td>309.43</td>
<td>3.43</td>
</tr>
<tr>
<td>3</td>
<td>90.30</td>
<td>95.90</td>
<td>5.60</td>
</tr>
<tr>
<td>4</td>
<td>530.00</td>
<td>528.45</td>
<td>1.55</td>
</tr>
<tr>
<td>5</td>
<td>87.50</td>
<td>80.67</td>
<td>6.83</td>
</tr>
<tr>
<td>6</td>
<td>530.00</td>
<td>533.55</td>
<td>3.55</td>
</tr>
<tr>
<td>7</td>
<td>530.00</td>
<td>526.78</td>
<td>3.22</td>
</tr>
<tr>
<td>8</td>
<td>488.00</td>
<td>483.34</td>
<td>4.66</td>
</tr>
<tr>
<td>9</td>
<td>257.00</td>
<td>254.78</td>
<td>2.22</td>
</tr>
<tr>
<td>10</td>
<td>308.00</td>
<td>308.14</td>
<td>0.14</td>
</tr>
<tr>
<td>11</td>
<td>258.00</td>
<td>255.93</td>
<td>2.07</td>
</tr>
<tr>
<td>12</td>
<td>61.10</td>
<td>62.78</td>
<td>1.68</td>
</tr>
<tr>
<td>13</td>
<td>91.50</td>
<td>84.96</td>
<td>6.54</td>
</tr>
<tr>
<td>14</td>
<td>88.00</td>
<td>87.26</td>
<td>0.74</td>
</tr>
<tr>
<td>15</td>
<td>60.80</td>
<td>62.36</td>
<td>1.56</td>
</tr>
<tr>
<td>16</td>
<td>28.10</td>
<td>29.03</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Table 5.8  Solution Comparison of Brick Wall Test Problem

The nodal temperature values are shown graphically in Fig. 5.10.
The next test case incorporates convection, flux, and Dirichlet boundary conditions. The domain is depicted in Fig. 5. 11.
The temperature distribution will be found using the program created for this thesis. We will then compare the distribution with the results from Hagen's finite difference analysis (Hagen, 1999). The mesh for the domain is shown in Fig. 5. 12
Fig. 5. 12 Mesh for Flux and Convection Test Problem

The nodal temperature values and absolute differences are listed in Table 5.9.
<table>
<thead>
<tr>
<th>Node</th>
<th>Temp Hagen FDM</th>
<th>Temp Thesis</th>
<th>Absolute Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>120.00</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>120.00</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>120.00</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>120.00</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>120.00</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>120.00</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>97.24</td>
<td>98.59</td>
<td>1.35</td>
</tr>
<tr>
<td>8</td>
<td>113.06</td>
<td>110.52</td>
<td>2.54</td>
</tr>
<tr>
<td>9</td>
<td>120.73</td>
<td>116.87</td>
<td>3.86</td>
</tr>
<tr>
<td>10</td>
<td>124.85</td>
<td>119.48</td>
<td>5.37</td>
</tr>
<tr>
<td>11</td>
<td>120.89</td>
<td>116.02</td>
<td>4.87</td>
</tr>
<tr>
<td>12</td>
<td>104.69</td>
<td>99.75</td>
<td>4.94</td>
</tr>
<tr>
<td>13</td>
<td>93.61</td>
<td>92.19</td>
<td>1.42</td>
</tr>
<tr>
<td>14</td>
<td>109.63</td>
<td>106.62</td>
<td>3.01</td>
</tr>
<tr>
<td>15</td>
<td>123.25</td>
<td>117.46</td>
<td>5.79</td>
</tr>
<tr>
<td>16</td>
<td>135.07</td>
<td>125.04</td>
<td>10.03</td>
</tr>
<tr>
<td>17</td>
<td>138.84</td>
<td>124.84</td>
<td>14.00</td>
</tr>
<tr>
<td>18</td>
<td>103.70</td>
<td>91.84</td>
<td>11.86</td>
</tr>
<tr>
<td>19</td>
<td>92.72</td>
<td>90.69</td>
<td>2.03</td>
</tr>
<tr>
<td>20</td>
<td>109.63</td>
<td>106.30</td>
<td>3.33</td>
</tr>
<tr>
<td>21</td>
<td>126.80</td>
<td>121.33</td>
<td>5.47</td>
</tr>
<tr>
<td>22</td>
<td>146.20</td>
<td>138.39</td>
<td>7.81</td>
</tr>
<tr>
<td>23</td>
<td>178.98</td>
<td>163.80</td>
<td>15.18</td>
</tr>
<tr>
<td>24</td>
<td>92.67</td>
<td>90.53</td>
<td>2.14</td>
</tr>
<tr>
<td>25</td>
<td>109.93</td>
<td>106.57</td>
<td>3.36</td>
</tr>
<tr>
<td>26</td>
<td>128.44</td>
<td>123.15</td>
<td>5.29</td>
</tr>
<tr>
<td>27</td>
<td>152.29</td>
<td>143.38</td>
<td>8.91</td>
</tr>
<tr>
<td>28</td>
<td>182.92</td>
<td>173.59</td>
<td>9.33</td>
</tr>
</tbody>
</table>

Table 5.9 Solution Comparison for Flux and Convection Test Problem

The nodal temperature values are represented graphically in Fig. 5.13.
In the next test case we will look at convection on a fin. The domain is depicted in Fig. 5. 14.
The temperature distribution will be found using the program created for this thesis. We will then compare the distribution with the one dimensional fin equation. The mesh for the domain is shown in Fig. 5. 15.

The temperature values and absolute differences are listed in Table 5. 10.
Table 5. 10 Solution Comparison for Fin Test Problem

<table>
<thead>
<tr>
<th>x coordinate</th>
<th>1-D Fin Equation</th>
<th>Temp Nodes 1 to 9</th>
<th>Temp Nodes 10 to 18</th>
<th>Abs Difference Nodes 1 to 9</th>
<th>Abs Difference Nodes 10 to 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>90.000</td>
<td>90.000</td>
<td>90.000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.015</td>
<td>80.996</td>
<td>81.491</td>
<td>80.556</td>
<td>0.4954</td>
<td>0.9351</td>
</tr>
<tr>
<td>0.030</td>
<td>73.675</td>
<td>74.083</td>
<td>73.366</td>
<td>0.4081</td>
<td>0.7172</td>
</tr>
<tr>
<td>0.045</td>
<td>67.819</td>
<td>68.204</td>
<td>67.563</td>
<td>0.3857</td>
<td>0.641</td>
</tr>
<tr>
<td>0.060</td>
<td>63.250</td>
<td>63.612</td>
<td>63.040</td>
<td>0.3624</td>
<td>0.572</td>
</tr>
<tr>
<td>0.075</td>
<td>59.832</td>
<td>60.176</td>
<td>59.655</td>
<td>0.3448</td>
<td>0.5212</td>
</tr>
<tr>
<td>0.090</td>
<td>57.461</td>
<td>57.793</td>
<td>57.307</td>
<td>0.3321</td>
<td>0.4858</td>
</tr>
<tr>
<td>0.105</td>
<td>56.066</td>
<td>56.391</td>
<td>55.926</td>
<td>0.3246</td>
<td>0.465</td>
</tr>
<tr>
<td>0.120</td>
<td>55.606</td>
<td>55.928</td>
<td>55.470</td>
<td>0.3221</td>
<td>0.4582</td>
</tr>
</tbody>
</table>

The temperature values are shown graphically in Fig. 5. 16

Fig. 5. 16 Solution Comparison for Fin Test Problem
CHAPTER 6

Conclusion

The finite element method is a powerful tool for solving partial differential equations (PDE) with complicated problems domains. The method has been used to solve complex problems in fluid flow, heat transfer, species transport, and a wide variety of structural problems. As technology in the computer industry becomes more advanced and less costly, we are able to solve larger and more sophisticated problems faster than ever before. Also, with the increased computing power the meshes can be refined to yield more accurate solutions without significantly increasing the computing time.

There are some drawbacks to using Applets. One possible solution to the problem is Java Servlets. Using servlets one can still build cross-platform Internet solutions. Also, with servlets there is the ability of direct access to the client and server machines. In addition, since a servlet is running on the server side the clients hardware does not become an issue in performance of the program. This eliminates the problem of low-end client machines not being able to solve large problems.

The current program should provide a basis for improvement. Future plans are to add mesh adaptation to the program. Adaptation allows the mesh to be refined in places where there may be a steep gradient. Also, interaction with the mesh and the user will be
added. This interaction will allow the user to move nodes and specify boundary conditions on the fly. Lastly, radiation and 3-D capability will be added to the program.
Notation

\[ u_x = \frac{\partial u}{\partial x} \]
\[ u_{xx} = \frac{\partial^2 u}{\partial x^2} \]
\[ a'(x) = \frac{d}{dx} a(x) \]
\[ a''(x) = \frac{d^2}{dx^2} a(x) \]
\[ \nabla T \equiv \frac{\partial T}{\partial x} + \frac{\partial T}{\partial y} \]
\[ \nabla^2 T \equiv \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \]
\[ \frac{\partial T}{\partial n} = \frac{\partial T}{\partial x} n_x + \frac{\partial T}{\partial y} n_y \]

Here \( n_x \) and \( n_y \) are the direction cosines of the unit outward vector, normal to a given boundary.
Analytic Solution

Substituting (2. 8) into the governing equation (2. 3) we get

\[ K \left[ a''(x) + u_{xx} + b''(y) + u_{yy} \right] + Q = 0 \]

A homogeneous equation for \( u(x, y) \) can be obtained by taking

\[ u_{xx} + u_{yy} = 0 \]  \hspace{1cm} (1)

Equation (1) implies that

\[ K \left[ a''(x) + b''(y) + Q \right] = 0 \]

where \( a''(x) = \frac{d^2}{dx^2} a(x) \). Assuming \( Q \) is a constant we can satisfy the above equation by letting

\[ Ka''(x) + Q = 0 \]  \hspace{1cm} (2)

and

\[ b''(y) = 0 \]  \hspace{1cm} (3)

Substituting (2. 8) into the boundary condition (2. 4) gives

\[ a'(0) + u_x(0, y) = 0 \]

where \( a'(x) = \frac{d}{dx} a(x) \). To obtain a homogeneous boundary condition for \( u(x, y) \) let

\[ u_x(0, y) = 0 \]  \hspace{1cm} (4)

The above dictates that we take

\[ a'(0) = 0 \]  \hspace{1cm} (5)

Substituting (2. 8) into (2. 5) gives

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ a(L) + b(y) + u(L, y) = T_L \]

To obtain a homogeneous boundary condition for \( u(x, y) \) let

\[ u(L, y) = 0 \]  \hspace{1cm} (6)

This implies that we must also satisfy

\[ a(L) + b(y) = T_L \]

From the above equation we see \( b(y) \) is a constant satisfying (3). Now we take

\[ a(L) = T_L \]  \hspace{1cm} (7)

and

\[ b(y) = 0 \]  \hspace{1cm} (8)

Substituting (2.8) into (2.6) yields

\[ b'(0) + u_y(x, 0) = 0 \]

Because we took \( b(y) = 0 \) in (8), we see that \( b'(0) = 0 \) and

\[ u_y(x, 0) = 0 \]  \hspace{1cm} (9)

Substituting (2.8) into the last boundary condition we have

\[ a(x) + b(L) + u(x, L) = T_L \]

Solving for \( u(x, L) \) and letting \( b(L) = 0 \) from (8) we have

\[ u(x, L) = T_L - a(x) \]  \hspace{1cm} (10)

We now summarize the sub problems and the boundary conditions that must be solved...
\[ K a''(x) + Q = 0 \]  \hspace{1cm} (2)

\[ a'(0) = 0 \]  \hspace{1cm} (5)

\[ a(L) = T_L \]  \hspace{1cm} (7)

\[ b''(y) = 0 \]  \hspace{1cm} (3)

\[ b(y) = 0 \]  \hspace{1cm} (8)

\[ u_{xx} + u_{yy} = 0 \]  \hspace{1cm} (1)

\[ u_x(0, y) = 0 \]  \hspace{1cm} (4)

\[ u(L, y) = 0 \]  \hspace{1cm} (6)

\[ u_y(x, 0) = 0 \]  \hspace{1cm} (9)

\[ u(x, L) = T_L - a(x) \]  \hspace{1cm} (10)

Equation (2) is an ordinary differential equation and can be solved by integration twice, applying the boundary conditions (5) and (7). This yields the following result

\[ a(x) = T_L + \frac{Q}{2K} \left( L^2 - x^2 \right) \]  \hspace{1cm} (11)

Substituting the result into (10) we have

\[ u(x, L) = T_L - T_L - \frac{Q}{2K} \left( L^2 - x^2 \right) \]

\[ = \frac{Q}{2K} \left( x^2 - L^2 \right) \]  \hspace{1cm} (12)

Equation (8) is the solution for \( b(y) \) and satisfies (3). Substituting (11) and (8) into (2. 8) we get
\[ T(x, y) = T_L + \frac{Q}{2K} (L^2 - x^2) + u(x, y) \]  
\hspace{1cm} (13)

For readability we will summarize the sub problems in \( T(x, y) \) and \( u(x, y) \) that are left to solve.

\[ T(x, y) = T_L + \frac{Q}{2K} (L^2 - x^2) + u(x, y) \]  
\hspace{1cm} (13)

\[ u_{xx} + u_{yy} = 0 \]  
\hspace{1cm} (1)

\[ u_x(0, y) = 0 \]  
\hspace{1cm} (4)

\[ u(L, y) = 0 \]  
\hspace{1cm} (6)

\[ u_y(x, 0) = 0 \]  
\hspace{1cm} (9)

\[ u(x, L) = \frac{Q}{2K} (x^2 - L^2) \]  
\hspace{1cm} (12)

The method of separation of variables can now be used to find \( u(x, y) \). To use the method assume the solution has the form of

\[ u(x, y) = Y(y) X(x) \]  
\hspace{1cm} (14)

Substituting the assumed solution into (1) yields

\[ X''(x) Y(y) + X(x) Y''(y) = 0 \]

After rearranging the equation we have

\[ \frac{X''(x)}{X(x)} = -\frac{Y''(y)}{Y(y)} \]

The only way for the equality to hold is if both sides are equal to a constant. Taking the constant to be \(-\lambda^2\) we let
\begin{equation}
X''(x) + \lambda^2 X(x) = 0
\end{equation}

and

\begin{equation}
Y''(y) - \lambda^2 Y(y) = 0
\end{equation}

Substituting (14) into the boundary condition (4) we have

\begin{equation}
X'(0)Y(y) = 0
\end{equation}

which can be satisfied if we take

\begin{equation}
X'(0) = 0
\end{equation}

Substituting (14) into the boundary condition (6) we get

\begin{equation}
X(L)Y(y) = 0
\end{equation}

In order to satisfy the above condition we take

\begin{equation}
X(L) = 0
\end{equation}

We now have the eigenproblem for \( X(x) \) given by (15), (16), and (18). The well-known general solution for (15) is (Powers, 1972)

\[ X(x) = A \sin(\lambda x) + B \cos(\lambda x) \]

We can satisfy (17) by taking \( A = 0 \). To satisfy (18) take \( \cos(\lambda L) = 0 \). For

\[ \cos(\lambda L) = 0 \]

to hold true \( \lambda L \) must be odd multiples of \( \frac{\pi}{2} \). Thus the eigenfunctions are given by

\[ X_n(x) = B_n \cos(\lambda_n L) \]

where \( \lambda_n L = (2n-1)\frac{\pi}{2} \). The well-known solution to (16) is
\[ Y_n(y) = C_n \sinh(\lambda_n y) + D_n \cosh(\lambda_n y) \quad (20) \]

Substituting (19) and (20) into (14) we obtain
\[ u_n(x, y) = \cos(\lambda_n x) \left[ C_n \sinh(\lambda_n y) + D_n \cosh(\lambda_n y) \right] \]

where \( B_n \) is now part of \( C_n \) and \( D_n \). By summing over all the eigenvalues a more general solution can be given by
\[ u(x, y) = \sum_{n=1}^{\infty} \cos(\lambda_n x) \left[ C_n \sinh(\lambda_n y) + D_n \cosh(\lambda_n y) \right] \quad (21) \]

The expression for \( C_n \) and \( D_n \) will be taken to make (21) satisfy the boundary conditions (9) and (12). After differentiating (21) with respect to \( y \), we have
\[ u_y(x, y) = \sum_{n=0}^{\infty} \cos(\lambda_n x) \left[ C_n \lambda_n \cosh(\lambda_n y) + D_n \lambda_n \sinh(\lambda_n x) \right] \]

Then by setting \( y = 0 \) and substituting into (9) we have
\[ \sum_{n=1}^{\infty} \cos(\lambda_n x) C_n \lambda_n = 0 \]

The only way for this equation to be satisfied for all \( x \) is to take \( C_n = 0 \). Thus (21) reduces to
\[ u(x, y) = \sum_{n=1}^{\infty} \cos(\lambda_n x) D_n \cosh(\lambda_n y) \quad (22) \]

Letting \( y = L \) and substituting (22) into (12) we get
\[ \sum_{n=1}^{\infty} \cos(\lambda_n x) D_n \cosh(\lambda_n y) = \frac{Q}{2K} \left( x^2 - L^2 \right) \]

Multiplying both sides by \( \cos(\lambda_n x) \, dx \) and integrating from \( x = 0 \) to \( x = L \) we obtain
\[ D_n \]. Thus we have
\[ \int_0^L \sum_{n=1}^{\infty} \cos(\lambda_n x) D_n \cosh(\lambda_n y) \cos(\lambda_m x) \, dx = \int_0^L \frac{Q}{2K} (x^2 - L^2) \cos(\lambda_m x) \, dx \]

From orthogonality we know that only terms where \( n = m \) remain. Factoring \( D_m \cosh(\lambda_m L) \) out of the integral we have

\[ D_m \cosh(\lambda_m L) \int_0^L \cos^2(\lambda_m x) \, dx = \int_0^L \frac{Q}{2K} (x^2 - L^2) \int_0^L \cos(\lambda_m x) \, dx \]

Carrying out the integration and solving for \( D_m \) yields

\[ D_m = 2 \frac{QL^2}{K} \frac{(-1)^m}{(\lambda_m L)^3 \cosh(\lambda_m L)} \]

Upon substituting the Fourier coefficient into (22) gives us \( u(x, y) \), which can then be substituted into (13) to give (2.9).
Units

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>SI</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>$A$</td>
<td>$m^2$</td>
<td>$ft^2$</td>
</tr>
<tr>
<td>Convective transfer coefficient</td>
<td>$h$</td>
<td>$W/m^2 , ^\circ C$</td>
<td>$Btu/hr , ft^2 , ^\circ F$</td>
</tr>
<tr>
<td>Density</td>
<td>$\rho$</td>
<td>$kg/m^3$</td>
<td>$lb_m/ft^3$</td>
</tr>
<tr>
<td>Heat energy</td>
<td>$\bar{q}$</td>
<td>$kJ$</td>
<td>$Btu$</td>
</tr>
<tr>
<td>Force</td>
<td>$F$</td>
<td>$N$</td>
<td>$lb_f$</td>
</tr>
<tr>
<td>Gravity</td>
<td>$g$</td>
<td>$m/s^2$</td>
<td>$ft/s^2$</td>
</tr>
<tr>
<td>Heat flux per unit area</td>
<td>$q/A$</td>
<td>$W/m^2$</td>
<td>$Btu/hr , ft^2$</td>
</tr>
<tr>
<td>Heat generation per unit length</td>
<td>$Q$</td>
<td>$W/m^3$</td>
<td>$Btu/hr , ft^3$</td>
</tr>
<tr>
<td>Length</td>
<td>$L$</td>
<td>$m$</td>
<td>$ft$</td>
</tr>
<tr>
<td>Mass</td>
<td>$m$</td>
<td>$kg$</td>
<td>$lb_m$</td>
</tr>
<tr>
<td>Pressure</td>
<td>$p$</td>
<td>$N/m^2$</td>
<td>$lb_f/\text{in}^2$</td>
</tr>
<tr>
<td>Specific heat</td>
<td>$c_p$</td>
<td>$kJ/kg , ^\circ C$</td>
<td>$Btu/lb_m , ^\circ F$</td>
</tr>
<tr>
<td>Thermal conductivity</td>
<td>$\kappa$</td>
<td>$W/m , ^\circ C$</td>
<td>$Btu/hr , ft , ^\circ F$</td>
</tr>
<tr>
<td>Thermal diffusivity</td>
<td>$\alpha(\kappa/\rho c_p)$</td>
<td>$m^2/s$</td>
<td>$ft^2/s$</td>
</tr>
<tr>
<td>Velocity</td>
<td>$v$</td>
<td>$m/s$</td>
<td>$ft/s$</td>
</tr>
</tbody>
</table>
Program Screen Shot of Tab One

Domain Mesh
Program Screen Shot of Tab Two

Solution Contours
Program Screen Shot of Tab Three

Solution Values
Gauss Points for Triangular Elements

3 Node Triangular Element

<table>
<thead>
<tr>
<th>Gauss Point</th>
<th>( L_1 )</th>
<th>( L_2 )</th>
<th>( L_3 )</th>
<th>( W )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
<td>0.33333333</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0.33333333</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.33333333</td>
</tr>
</tbody>
</table>

6 Node Triangular Element

<table>
<thead>
<tr>
<th>Gauss Point</th>
<th>( L_1 )</th>
<th>( L_2 )</th>
<th>( L_3 )</th>
<th>( W )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.333333333</td>
<td>0.333333333</td>
<td>0.333333333</td>
<td>0.225</td>
</tr>
<tr>
<td>2</td>
<td>0.47014206</td>
<td>0.05971587</td>
<td>0.47014206</td>
<td>0.13239415</td>
</tr>
<tr>
<td>3</td>
<td>0.47014206</td>
<td>0.47014206</td>
<td>0.05971587</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.05971587</td>
<td>0.47014206</td>
<td>0.47014206</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.10128651</td>
<td>0.79742699</td>
<td>0.10128651</td>
<td>0.12593918</td>
</tr>
<tr>
<td>6</td>
<td>0.10128651</td>
<td>0.10128651</td>
<td>0.79742699</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.79742699</td>
<td>0.10128651</td>
<td>0.10128651</td>
<td></td>
</tr>
</tbody>
</table>
Gauss Points for Quadrilateral Elements

### 4 Node Quadrilateral Elements

<table>
<thead>
<tr>
<th>Gauss Points</th>
<th>$\xi_i$</th>
<th>$\eta_i$</th>
<th>$W_i$</th>
<th>$W_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{-1}{\sqrt{3}}$</td>
<td>$\frac{-1}{\sqrt{3}}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{-1}{\sqrt{3}}$</td>
<td>$\frac{1}{\sqrt{3}}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{1}{\sqrt{3}}$</td>
<td>$\frac{-1}{\sqrt{3}}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{1}{\sqrt{3}}$</td>
<td>$\frac{1}{\sqrt{3}}$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### 8 Node Quadrilateral Elements

<table>
<thead>
<tr>
<th>Gauss Points</th>
<th>$\xi_i$</th>
<th>$\eta_i$</th>
<th>$W_i$</th>
<th>$W_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{-\sqrt{3}}{5}$</td>
<td>$\frac{-\sqrt{3}}{5}$</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{-\sqrt{3}}{5}$</td>
<td>0</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{8}{9}$</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{-\sqrt{3}}{5}$</td>
<td>$\frac{\sqrt{3}}{5}$</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>$\frac{-\sqrt{3}}{5}$</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>$\frac{\sqrt{3}}{5}$</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
<tr>
<td>7</td>
<td>$\frac{\sqrt{3}}{5}$</td>
<td>$\frac{-\sqrt{3}}{5}$</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
<tr>
<td>8</td>
<td>$\frac{\sqrt{3}}{5}$</td>
<td>0</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
<tr>
<td>9</td>
<td>$\frac{\sqrt{3}}{5}$</td>
<td>$\frac{\sqrt{3}}{5}$</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{5}{9}$</td>
</tr>
</tbody>
</table>
for NSTEP = 1 to NSTOP

KOUNT = KPRNT

Yes

PRINT KOUNT = 0

No

TIME = TIME + DT
KOUNT = KOUNT + 1
NTIME = NTIME + 1
RESID

L = 1 to NNODE

Output

End
Sample Input Data File

Data set Title: LIN-2D4.DAT
1 MTYPE
2 NUMDIM Number of dimensions
25 NNODE Number of nodes
16 NELEM Number of elements
4 NUMN Number of Nodes per Element
100 NSTOP Max No. of iterations
50 KPRNT Printing interval
1 NVEL (0) no velocities or (1) velocity data set exists
2 NTYPE (1) Steady-state or (2) time-dependent
0.10 DT Time step
1.0 AF(1.0) Implicit or (0.5) Crank-Nicolson
0.0 TO
1.0 RHO
1.0 CP
0 TAXI (0) Cartesian or (1) Axisymmetric
1.0 max x
1.0 max y
nodal coordinates
 1  0.00000  0.00000
 2  0.00000  0.25000
 3  0.00000  0.50000
 4  0.00000  0.75000
 5  0.00000  1.00000
 6  0.25000  0.00000
 7  0.25000  0.25000
 8  0.25000  0.50000
 9  0.25000  0.75000
10  0.25000  1.00000
11  0.50000  0.00000
12  0.50000  0.25000
13  0.50000  0.50000
14  0.50000  0.75000
15  0.50000  1.00000
16  0.75000  0.00000
17  0.75000  0.25000
18  0.75000  0.50000
19  0.75000  0.75000
20  0.75000  1.00000
21  1.00000  0.00000
22  1.00000  0.25000
23  1.00000  0.50000
24  1.00000  0.75000
25  1.00000  1.00000
1  .00  15.00  15.00  1  6  7  2
2  .00  15.00  15.00  2  7  8  3
3  .00  15.00  15.00  3  8  9  4
4  .00  15.00  15.00  4  9  10  5
5  .00  15.00  15.00  6  11  12  7
6  .00  15.00  15.00  7  12  13  8
7  .00  15.00  15.00  8  13  14  9
8  .00  15.00  15.00  9  14  15  10
9  .00  15.00  15.00 11  16  17  12
10 .00  15.00  15.00 12  17  18  13
11 .00  15.00  15.00 13  18  19  14
12 .00  15.00  15.00 14  19  20  15
13 .00  15.00  15.00 16  21  22  17
14 .00  15.00  15.00 17  22  23  18
15 .00  15.00  15.00 18  23  24  19
16 .00  15.00  15.00 19  24  25  20

1  1000.00
2  1000.00
3  1000.00
4  1000.00
5  1000.00

DIRC
FLUX
120.00  1500.00  1  4
120.00  1500.00  4  2
120.00  1500.00  5  4
120.00  1500.00  8  2
120.00  1500.00  9  4
120.00  1500.00 12  2
120.00  1500.00 13  3
120.00  1500.00 13  4
120.00  1500.00 16  2
120.00  1500.00 16  3

CONV
VELC
BIBLIOGRAPHY


VITA

Graduate College
University of Nevada, Las Vegas

Trevor A. Wilcox

Local Address:
110 Quiet Desert Ln
Henderson, NV 89014

Home Address:
6635 Lunar Drive
Anchorage, AK 99504

Degrees:
Bachelor of Science, Mathematics, 1997
University of Alaska, Anchorage

Master of Science, Mathematics, 1998
University of Nevada, Las Vegas

Thesis Title: A Web Based Finite Element Model for Heat Transfer

Thesis Examination Committee:
Chairperson, Dr. Darrell Pepper
Committee Member, Dr. Mohamed Trabia
Committee Member, Dr. Yi-Tung Chen
Graduate Faculty Representative, Dr. Laxmi Gewali