5-1987

An Application of the Finite Element Method and Two Equation (K and E) Turbulence Model to Two and Three Dimensional Fluid Flow Problems Governed by the Navier-Stokes Equations

John I. Finnie
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Civil and Environmental Engineering Commons

Recommended Citation
https://digitalcommons.usu.edu/etd/7350

This Dissertation is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
AN APPLICATION OF THE FINITE ELEMENT METHOD AND TWO EQUATION (K AND E) TURBULENCE MODEL TO TWO AND THREE DIMENSIONAL FLUID FLOW PROBLEMS GOVERNED BY THE NAVIER-STOKES EQUATIONS

by

John I. Finnie

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Civil and Environmental Engineering

UTAH STATE UNIVERSITY
Logan, Utah

1987
ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to Dr. Roland Jeppson for his guidance and help during this research project. Likewise my thanks go to Drs. Calvin Clyde, Chris Coray, Gordon Flammer, and Paul Tullis for their help and comments. I have been honored by their encouragement and friendship and I hope that I will come up to their confidence in me.

I also appreciate Steve Folkman's help with understanding Fortran. Thanks are due to Leslie Johnson for her excellent typing skill.

This research was made possible by teaching and research assistantships through the Department of Civil Engineering and by computer grants through the University. Kim Marshall was especially helpful with using the computer to its full advantage.

John P. Reeve, Weber County Surveyor, provided part and full-time employment to me for more than two years during my coursework and research. I enjoyed and appreciated working with him and the other employees.

My wife Ruth and sons Scott, Andrew, and Sean worked hard and made many sacrifices so that I could attend graduate school. No man ever had a better family.

To my parents I wish to say thanks. You gave me confidence in myself and provided a shining example of how to live life.

John I. Finnie
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>11</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>NOTATION</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xi</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>THEORY AND LITERATURE REVIEW</td>
<td>3</td>
</tr>
<tr>
<td>Theory of Turbulence Modeling</td>
<td>3</td>
</tr>
<tr>
<td>Reynolds's Stress Differential Equations</td>
<td>4</td>
</tr>
<tr>
<td>Turbulent Viscosity</td>
<td>6</td>
</tr>
<tr>
<td>Applications of the k-E Turbulence Model</td>
<td>16</td>
</tr>
<tr>
<td>Simplifications of the k-E Model</td>
<td>18</td>
</tr>
<tr>
<td>The Galerkin Finite Element Method</td>
<td>19</td>
</tr>
<tr>
<td>Finite Differences and Upwinding</td>
<td>19</td>
</tr>
<tr>
<td>Galerkin's Method</td>
<td>20</td>
</tr>
<tr>
<td>Weighting Functions</td>
<td>20</td>
</tr>
<tr>
<td>The Finite Element Equations</td>
<td>23</td>
</tr>
<tr>
<td>Numerically Integrating the Equations</td>
<td>31</td>
</tr>
<tr>
<td>Solving the Equations</td>
<td>32</td>
</tr>
<tr>
<td>Sparse Matrix Routines</td>
<td>33</td>
</tr>
<tr>
<td>THE COMPUTER PROGRAM</td>
<td>36</td>
</tr>
<tr>
<td>The Computer Code</td>
<td>36</td>
</tr>
<tr>
<td>Input Data</td>
<td>38</td>
</tr>
<tr>
<td>Shape Functions</td>
<td>40</td>
</tr>
<tr>
<td>Velocity-Pressure Phase</td>
<td>40</td>
</tr>
<tr>
<td>Non-zero Gradient Boundary Conditions</td>
<td>42</td>
</tr>
<tr>
<td>Subroutine SOLVE</td>
<td>42</td>
</tr>
<tr>
<td>Turbulent Kinetic Energy-Dissipation Rate Phase</td>
<td>43</td>
</tr>
<tr>
<td>Program Output</td>
<td>44</td>
</tr>
<tr>
<td>Data Generation Programs</td>
<td>44</td>
</tr>
<tr>
<td>RESULTS AND CONCLUSION</td>
<td>45</td>
</tr>
<tr>
<td>Laminar Problems</td>
<td>45</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbulent Problems</td>
<td>52</td>
</tr>
<tr>
<td>Boundary Conditions</td>
<td>72</td>
</tr>
<tr>
<td>Initial Conditions</td>
<td>75</td>
</tr>
<tr>
<td>Grid Definition</td>
<td>75</td>
</tr>
<tr>
<td>Wall Driven Flows</td>
<td>77</td>
</tr>
<tr>
<td>Problem Size</td>
<td>80</td>
</tr>
<tr>
<td>Performance of Harwell Subroutines</td>
<td>81</td>
</tr>
<tr>
<td>Computer Size</td>
<td>82</td>
</tr>
<tr>
<td>Solution Times</td>
<td>83</td>
</tr>
<tr>
<td>Conclusions</td>
<td>83</td>
</tr>
<tr>
<td>SUGGESTIONS FOR FURTHER STUDY</td>
<td>87</td>
</tr>
<tr>
<td>SELECTED BIBLIOGRAPHY</td>
<td>89</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>93</td>
</tr>
<tr>
<td>Appendix A. FEM2D.FOR</td>
<td>94</td>
</tr>
<tr>
<td>Appendix B. Data Files: FEM.DAT, GRID.DAT, INITIAL.DAT, and BC.DAT</td>
<td>116</td>
</tr>
<tr>
<td>Appendix C. Data Generation Programs: NODES4.FOR, SLUINIT.FOR, SLUBC.FOR</td>
<td>132</td>
</tr>
<tr>
<td>VITA</td>
<td>142</td>
</tr>
</tbody>
</table>
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Laminar and turbulent two and three dimensional flow problems</td>
<td>46</td>
</tr>
<tr>
<td>2. Effect of U and number of unknowns on matrix solution times on VAX 11/780 for Harwell sparse routines for two dimensional wall driven flow at Re = 1000</td>
<td>82</td>
</tr>
<tr>
<td>3. Total CPU time for solution of selected problems</td>
<td>84</td>
</tr>
<tr>
<td>4. Matrix solution times for two and three dimensional problems at various numbers of unknowns - on VAX 11/780 using Harwell subroutines at U = 0.1</td>
<td>85</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure | Page
---|---
1. Two dimensional finite elements | 22
2. Three dimensional finite elements | 22
3. Global node numbers for a three element grid | 29
4. Outline of the computer program Fem3d | 37
5. Problem 1. Velocity driven channel flow, 8 unknowns | 47
6. Problem 2. Pressure driven laminar channel flow, 40 unknowns | 47
7. Velocity (ft/sec) and pressure (lbs/ft²) results of problem 2 | 48
8. Problem 18. Velocity driven three dimensional laminar channel flow between parallel walls, 446 unknowns | 49
9. Problem 17. Pressure driven three dimensional laminar channel flow between parallel walls, 140 unknowns | 50
10. Pressure results (lbs/ft²) for three dimensional laminar velocity driven flow streamwise along center row of elements for problem 18 | 51
11. Pressure results (lbs/ft²) for three dimensional laminar pressure driven flow streamwise along center row of elements for problem 17 | 51
12. Problem 6. Two dimensional laminar wall driven flow | 53
13. Problem 14. Velocity results (ft/sec) of wall driven flow at Re=100 on 36 unevenly spaced elements | 53
14. Problem 16. Velocity results (ft/sec) of wall driven flow at Re=1000 on 36 unevenly spaced elements | 54
15. Problem 12. Velocity results (ft/sec) of wall driven flow at Re=1000 on 196 evenly spaced elements | 54
16. Problem 19. Two dimensional turbulent channel flow between parallel walls, 153 unknowns | 55
17. Problem 20. Two dimensional turbulent channel flow between parallel walls, 150 unknowns | 55
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.</td>
<td>Pressure results (lbs/ft²) for problem 20</td>
<td>57</td>
</tr>
<tr>
<td>19.</td>
<td>Turbulent viscosity results (ft²/sec) for problem 20</td>
<td>57</td>
</tr>
<tr>
<td>20.</td>
<td>Problems 21 to 25. Turbulent flow from a channel to a circular outlet in two dimensions, 587 unknowns</td>
<td>58</td>
</tr>
<tr>
<td>21.</td>
<td>Velocity results (ft/sec) for problem 25</td>
<td>59</td>
</tr>
<tr>
<td>22.</td>
<td>Pressure results (lbs/ft²) for problem 25</td>
<td>60</td>
</tr>
<tr>
<td>23.</td>
<td>Turbulent viscosity results (ft²/sec) for problem 25</td>
<td>60</td>
</tr>
<tr>
<td>24.</td>
<td>Problems 27 and 28. Three dimensional turbulent channel flow between parallel walls, 790 unknowns</td>
<td>62</td>
</tr>
<tr>
<td>25.</td>
<td>Velocity results (ft/sec) of problems 27 and 28</td>
<td>63</td>
</tr>
<tr>
<td>26.</td>
<td>Pressure results (lbs/ft²) of problem 27 streamwise along center row of elements</td>
<td>63</td>
</tr>
<tr>
<td>27.</td>
<td>Turbulent kinetic energy results (ft²/sec²) of problem 27 streamwise along center row of elements</td>
<td>64</td>
</tr>
<tr>
<td>28.</td>
<td>Turbulent kinetic energy as a percentage of inlet values for problem 27 streamwise along center row of elements</td>
<td>64</td>
</tr>
<tr>
<td>29.</td>
<td>Turbulent kinetic energy dissipation rate results (ft²/sec³) of problem 27 streamwise along center row of elements</td>
<td>65</td>
</tr>
<tr>
<td>30.</td>
<td>Turbulent kinetic energy dissipation rate as a percentage of inlet values for problem 27 streamwise along center row of elements</td>
<td>65</td>
</tr>
<tr>
<td>31.</td>
<td>Turbulent viscosity results (ft²/sec) of problem 27 streamwise along center row of elements</td>
<td>66</td>
</tr>
<tr>
<td>32.</td>
<td>Turbulent viscosity as a percentage on inlet values for problem 27 streamwise along center row of elements</td>
<td>66</td>
</tr>
<tr>
<td>33.</td>
<td>Problem 26. Two dimensional turbulent flow under a sluice gate, 1851 unknowns</td>
<td>67</td>
</tr>
<tr>
<td>34.</td>
<td>Finite element grid for problem 26</td>
<td>67</td>
</tr>
<tr>
<td>35.</td>
<td>Boundary conditions for problem 26</td>
<td>69</td>
</tr>
<tr>
<td>36.</td>
<td>Velocity results (ft/sec) for problem 26</td>
<td>69</td>
</tr>
</tbody>
</table>
LIST OF FIGURES (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>37. Pressure results (lbs/ft$^2$) for problem 26</td>
<td>70</td>
</tr>
<tr>
<td>38. Turbulent viscosity results (ft$^2$/sec) for problem 26</td>
<td>70</td>
</tr>
<tr>
<td>39. Plan view of three dimensional vortex flow</td>
<td>71</td>
</tr>
<tr>
<td>40. Problem 29. Three dimensional vortex flow, up to 2398 unknowns</td>
<td>71</td>
</tr>
<tr>
<td>41. Pressure results (lbs/ft$^2$) of problem 16</td>
<td>74</td>
</tr>
<tr>
<td>42. Pressure results (lbs/ft$^2$) of problem 12</td>
<td>74</td>
</tr>
<tr>
<td>43. Stream function results of wall driven flow at Re=100 [Olson and Tuann, 1979]</td>
<td>78</td>
</tr>
<tr>
<td>44. Stream function results of wall driven flow at Re=1000 [Olson and Tuann, 1979]</td>
<td>78</td>
</tr>
<tr>
<td>45. Comparison of results of wall driven flow at Re=100</td>
<td>79</td>
</tr>
<tr>
<td>46. Comparison of results of wall driven flow at Re=1000</td>
<td>79</td>
</tr>
</tbody>
</table>
The following symbols are used in this paper:

- **e** = local coordinates
- **g** = acceleration of gravity
- **h** = height above datum
- **k** = turbulent kinetic energy
- **k_s** = k on the surface of the finite element
- **l_m** = mixing length
- **n** = local coordinates
- **s** = local coordinates
- **t** = time
- **u_i** = fluctuating velocity in i direction
- **B** = matrix of derivatives
- **C** = constant in turbulence model
- **C^*** = constant in turbulence model
- **C_E** = constant in turbulence model
- **C_E1** = constant in turbulence model
- **C_E2** = constant in turbulence model
- **D** = Jacobian of vectors
- **E** = dissipation rate of turbulent kinetic energy
- **E'** = error
- **F** = vector of integrals
- **H** = local depth
- **J** = Jacobian of coordinate transform
- **L** = differential operator
- **M_r** = linear weighting function
\[ NR = \text{quadratic weighting function} \]
\[ P = \text{pressure} \]
\[ P' = \text{fluctuating pressure} \]
\[ P = \text{time averaged pressure} \]
\[ P^* = \text{pressure and gravity body force} \]
\[ P_{\text{KIN}} = \text{production of kinetic energy by shear} \]
\[ R = \text{space of weighting functions} \]
\[ U_i = \text{velocity in } i \text{ direction} \]
\[ \bar{U}_1 = \text{time averaged velocity in } 1 \text{ direction} \]
\[ U_* = \text{shear velocity} \]
\[ V = \text{mean fluctuating velocity} \]
\[ W(e,n) = \text{Gauss-Legendre weighting function} \]
\[ X_i = \text{coordinate in } i \text{ direction} \]
\[ Y = \text{wall distance} \]
\[ Y^+ = \text{normalized wall distance} \]
\[ Y = \text{specific weight of water} \]
\[ \delta_{ij} = \text{Kroneker's delta} \]
\[ \kappa = \text{von Kar'mens' constant} \]
\[ \mu = \text{dynamic viscosity} \]
\[ \nu = \text{kinematic viscosity} \]
\[ \nu_T = \text{turbulent (kinematic) viscosity} \]
\[ \rho = \text{density} \]
\[ \sigma_E = \text{turbulent Prandtl number for } E \]
\[ \sigma_k = \text{turbulent Prandtl number for } k \]
ABSTRACT

An Application of the Finite Element Method and Two Equation (k and E) Turbulence Model to Two and Three Dimensional Fluid Flow Problems Governed by the Navier-Stokes Equations

by

John I. Finnie, Doctor of Philosophy

Utah State University, 1987

Major Professor: Dr. Roland Jeppson
Department: Civil Engineering

Key Words: Fluid mechanics, Turbulence Model, Navier Stokes, Finite Element

Finite Element computer codes in two and three dimensions were written that solve both laminar and turbulent flow. These codes use the two equation (k and E) turbulence model to evaluate turbulent viscosity. They were tested with 29 different flow problems. The largest two dimensional turbulent problem solved is flow under a sluice gate. A three dimensional vortex flow problem was attempted but was not feasible due to the size of the available computer. The Harwell sparse matrix subroutines of the United Kingdom Atomic Energy Authority were used to solve the set of simultaneous equations. The performance of these subroutines is evaluated. The importance of defining adequate finite element grids and setting proper boundary and initial conditions is discussed.

(154 pages)
INTRODUCTION

Two-equation turbulence models have been developed in the last 10 to 15 years that solve 2 and 3 dimensional flow problems. These applications have been mainly performed on small scale flow fields, like boundary layers, jets, and wakes. The civil engineering profession has made limited use of these models in problems that are significant to them, yet they have the potential of becoming design tools for the modern civil engineer.

The past civil engineering applications of these models have included open channel flow around bends, under a sheet of ice, through expansions, and within sedimentation basins. Most of these have used a finite difference technique and made assumptions that don't allow for secondary currents. A common practice (called upwinding) has been to use a differencing equation that weights the upstream value of the variables more than the downstream values. This project will use a finite element method that allows secondary currents and does not resort to uneven weighting.

There are a total of 6 partial differential equations that together describe turbulent flow. They are the Navier-Stokes equations, the continuity equation, and 2 transport equations for turbulent viscosity. Galerkin's method will be used to develop the algebraic equations that replace these partial differential equations. Since these equations are non-linear, the Newton method will be used to find a solution.

Two assumptions are made that enable the solution of turbulent flow problems. The first is that the additional stresses that arise in the Navier-Stokes equations from turbulent flow (Reynold's stresses) can be
calculated from the mean flow stresses within the fluid, the turbulent viscosity \( \nu_T \), and the turbulent kinetic energy \( k \). The second is that this turbulent viscosity is a function of turbulent kinetic energy and the dissipation rate of turbulent kinetic energy \( \varepsilon \).

\[ \nu_T = C \frac{k^2}{\varepsilon} \]

The objectives of this dissertation are to write a finite element computer program that can solve the Navier-Stokes equations for turbulent flow, apply it to the solution of a significant problem in hydraulics, and compare the solution to experimental data. The problem chosen for solution was a three dimensional vortex in a round tank. This vortex was studied experimentally by Daggett and Kuelegan [1974] of the United States Army Corp of Engineers at Vicksburg, Mississippi. This problem was found to require too much time and storage for the available computer. The alternative problem chosen was two dimensional flow under a sluice gate.

This dissertation starts with a review of turbulence modeling, the Galerkin finite element method, and solution of sparse matrices. The computer program will be described, including its input requirement, variables, arrays, and general outline. Results from the program development and testing phase are presented as are the results of the sluice gate problem. The conclusion section discusses the performance and operational characteristics of the program.
THEORY AND LITERATURE REVIEW

The theory to be reviewed in this section falls into 3 main areas. These areas are: (1) the theory of turbulence modeling, (2) the Galerkin finite element method applied to the Navier-Stokes equations, (3) and sparse matrix routines.

Theory of Turbulence Modeling

The Navier-Stokes equations when combined with the equation of continuity provide a complete description of fluid flow at any Reynolds number. The Navier-Stokes equations are shown here using the Einstein summation wherein subscripts specify repeating of the variable i and j, usually taking a value of 1 to 3.

\[
\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial P^*}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_i \partial x_j}
\]

Components of velocities are the $U_i$ variables and $P^*$ is the sum of both pressure ($P$), a surface force, and $\gamma h$, the gravity body force. $\rho$ is the density of the fluid or mass per unit volume, $\nu$ is the kinematic viscosity, in length squared per second, and $\gamma$ is the specific weight of fluid or weight per unit volume.

The equation of continuity for incompressible fluids is shown.

\[
\frac{\partial U_i}{\partial x_i} = 0
\]

An important step towards a turbulent model was taken by Osborn Reynolds [1894]. To simplify the turbulent case he divided velocity and pressure into their average and fluctuating parts. The steady part was
chosen to represent the flow quantities during a time period longer than the smallest events while still allowing gross changes to occur to the flow. His equations are

\[ u_i = \overline{u_i} + u_i' \]

\[ p = \overline{p} + p' \]

in which the overbar denotes time averaging and the prime indicates the fluctuating part. Substituting these into the Navier-Stokes and continuity equations and time averaging the terms results in the Reynolds equations of motion. The Reynolds equations of motion are

\[ \frac{\partial \overline{u_i}}{\partial t} + \overline{u_j} \frac{\partial \overline{u_i}}{\partial x_j} = -\frac{1}{\rho} \frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j} \left( \nu \left( \frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) - \overline{u_i' u_j'} \right) \]

This representation produces additional stress terms called the Reynolds stresses. These are the \( \overline{u_i' u_j'} \) terms where \( i \) and \( j \) both take on values of 1, 2 and 3. These additional terms represent the temporal mean of the product of the fluctuations about the mean velocity and pressure and are always positive. These terms when multiplied by density are the additional stresses caused by turbulence in the flow. Turbulence modelling refers to the process of evaluating these stresses.

The Reynolds stresses can be solved for by two methods. One approach writes additional differential equations for these stresses. The other, which introduces the eddy (turbulent) viscosity approach, is used in this study.

Reynolds Stress Differential Equations

The drawback with the first approach is that it delays assumptions about stresses, since each additional equation adds as many unknowns as
it does equations. For example, the six Reynolds stress transport equations introduce the requirement for modeling the production, diffusion, and dissipation of the stresses, which include three way correlations between velocity, pressure, and their gradients. Models of up to 28 differential equations have been reported [Launder and Spalding, 1972, p. 20]. One of the advantages of differential stress equations is that they allow for the calculation of secondary flows in the cross section perpendicular to the main flow. These are seen in most channel flows. See for example the measurements of Nikuradse in Schlichting [1979, pp. 613-614]. Launder et al [1975], Hanjalic and Launder [1972a] and Daly and Harlow [1970] have investigated the solution of these secondary flows. Differential stress equations provide better solutions in channels with unequal wall roughness. They do not predict downstream reattachment of secondary currents behind bluff bodies any better than the turbulent viscosity approach [Pope and Whitelaw, 1976].

One promising area of this approach makes simplifying assumptions about the balance between generation and destruction of the stresses which turn the differential equations into algebraic equations. Frequently this requires differential equations for kinetic energy and its dissipation rate that are included in the next method discussed. [Launder et al 1975, Hanjalic and Launder, 1972b, and Daly and Harlow, 1970]. One weakness of differential stress models is that they lack the mathematical property of spatial invariance. Speziale [1979] made proposals to rectify this problem.
Turbulent Viscosity

The method used in this study employs the Boussinesq eddy viscosity concept. This relates the Reynolds stresses to the mean velocity gradients, \( \overline{\partial U_i / \partial x_j} \), the variable turbulent viscosity, \( v_T \), and the kinetic energy (k) of the flow (\( \text{ft}^2/\text{sec}^2 \)). The equation is

\[
-u_i u_j = v_T \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij}
\]

where \( \delta_{ij} \) is Kroneker's delta. The last term adds to the equation when \( i=j \) and the sum of these normal stresses equals twice the kinetic energy.

\[
k = \frac{1}{2} u_i u_i
\]

The problem is to determine the value of the turbulent viscosity \( v_T \). Prandtl [1925] drew a parallel with gas viscosity and hypothesized that the turbulent viscosity is proportional to the product of a mean fluctuating velocity, \( V \), and a mean free path. In fluids the "mixing length", \( l_m \), plays the part of the mean free path.

\( v_T \propto l_m V \)

In two dimensional shear layers he proposed that the mean fluctuating velocity equals the product of the mixing length and the mean velocity gradient.

\[
V = l_m \left| \frac{\partial U_1}{\partial x_2} \right|
\]

Combining these leads to the following mixing length expression for turbulent viscosity.
Mixing lengths have been determined for various types of shear flows including jets, wakes, and mixing layers [Rodi, 1980, p. 17]. Other algebraic expressions have been developed, including von Karman's [1930] similarity hypothesis, Pantaker and Spalding's [1970] ramp function, Van Driest's [1956] wall damping function, Nikuradse's equation, (see Schlichting, 1979, p.605), Prandtl's free shear layer equation, (see Rodi, 1980, p. 20), and the formulas of Cebeci and Smith [1968] and Mellor and Herring [1968]. All of these attempts, while valid, lack universality of application. Mixing length models do not account for convection and diffusion of turbulence, and do not predict shear stress where the velocity gradient is zero.

Other researchers have developed models using differential equations to evaluate turbulent viscosity. Nee and Kovasznay (see Launder and Spalding, 1972, pp. 13-14) used a transport equation to determine turbulent viscosity and an algebraic equation for the mixing length. Prandtl (see Launder and Spalding, 1972, pp. 13-14) related turbulent viscosity to the product of a mixing length and the square root of kinetic energy. He prescribed a transport equation to determine kinetic energy and an algebraic equation to determine the mixing length. Mellor and Herring [1968] related turbulent viscosity to the product of a mixing length and the square root of kinetic energy. They also used a differential equation to determine kinetic energy and an algebraic equation for mixing length [Cebeci and Smith, 1974, p. 175]. These methods still require that the mixing length or a form of it be determined algebraically. In order to increase the range of their
models, some researchers have defined an additional differential equation for mixing length. Kolmogorov (see Launder and Spalding, 1972, p. 15) related the mixing length to a characteristic frequency of the energy containing motion.

Many researchers have used differential equations that solve for variables that are related to mixing lengths. Ng and Spalding [1972] determined mixing length from an energy length differential equation. Harlow and Nakayama [1968] used a differential equation for kinetic energy dissipation, where

$$E = k^{1.5}/l_m$$

Other forms of mixing length variables include $k^{0.5}/l_m$, $kl_m$, and $k/l_m^2$. See Launder and Spalding [1972, p.95] and Rodi [1980, pp. 86-91] for a discussion of these equations and a summary of their applications.

The approach used in this study is to solve for both kinetic energy, $(k)$, and the Harlow and Nakayama model of kinetic energy dissipation, $(E=k^{1.5}/l_m)$, with differential equations. Viscosity is then determined from the relationship

$$\nu_t = C^* k^{2}/E$$

$$= Ck^{1/2} l_m$$

The result is called the k-E model or two equation model and has been successfully used in small scale flows such as boundary layer, jet, mixing layer, and wakes as well as in larger scale flows such as curved open channels and conduits. This model can handle more complicated flows than an algebraic mixing length model. Rodi [1980] provided a summary of its use. Most of these applications used the finite difference method.
Hanjalic and Launder [1972b] give an exact transport equation for $E$, or rather its equivalent, fluctuating vorticity, $\nu(\partial u_i / \partial x_j)^2$. Davidov in 1961 developed this equation by manipulating the Navier-Stokes equations.

\[
\frac{DE}{Dt} = -2\nu \frac{\partial u_i}{\partial x_j} \left( \frac{\partial u_i}{\partial x_l} \frac{\partial u_j}{\partial x_l} + \frac{\partial u_j}{\partial x_i} \frac{\partial u_l}{\partial x_j} \right) \quad (i)
\]

\[
-2\nu \frac{\partial u_i}{\partial x_j} \frac{\partial u_j}{\partial x_l} \frac{\partial u_l}{\partial x_i} \quad (ii)
\]

\[
-2 [\nu \frac{\partial^2 u_i}{\partial x_j \partial x_l}]^2 \quad (iii)
\]

\[
- \frac{\partial}{\partial x_j} u_j' E' \quad (iv)
\]

\[
- \frac{\nu}{\rho} \frac{\partial}{\partial x_i} [\frac{\partial p}{\partial x_l} \frac{\partial u_i}{\partial x_l}] \quad (v)
\]

The overbars indicate a correlation between the various variables and gradients. These terms arise in the derivation of the equation from dividing variables into their steady and fluctuating parts as was done to obtain the Reynolds stresses.

While dissipation actually occurs at the smallest eddy sizes, its rate is controlled by the energy cascade or vortex stretching. The first term describes generation of vorticity by shear. They evaluate this term as a function of the Reynolds stresses, which are shear stresses.

\[
(i) = C_{E1}(E/k)(\partial u_i / \partial x_j)u_i' u_j'
\]

where $C_{E1}$ is a constant.
The Reynolds stresses are replaced by the Bousinesq eddy viscosity term already given.

\[(i) = C_{E1} (E/k) \left( \frac{\partial U_i}{\partial x_j} \right) [v_T (\frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j}) - 2k \delta_{ij}/3] \]

The second and third terms represent the destruction of vorticity by viscosity. The destruction rate is controlled by the energy cascade process, since dissipation occurs at eddy sizes which are much smaller than the energy containing motions. This process is independent of viscosity at high Reynolds numbers. Because it is a function of the flow characteristics it is assumed to be related to \( E \) and \( k \). From dimensional considerations, they are modelled as

\[(ii) + (iii) = C_{E2} E^2/k \]

where \( C_{E2} \) is a constant.

The fourth and fifth terms represent transport by diffusion and pressure fluctuations. Integrating these terms across a boundary layer reveals that they do not change the dissipation rate, they only redistribute it. Like many diffusion processes its rate is assumed to be related to spatial gradients. Term iv and v are modeled as

\[(iv) + (v) = C_{E} \frac{\partial}{\partial x_j} ((k/E)u_j'u_l'(\partial E/\partial x_l)) \]

where \( C_{E} \) is a constant.

Substitution for the Reynolds stresses with the equivalent turbulent viscosity expression results in the following equation (see Rodi, 1980, p. 28)

\[
\frac{D E}{D t} = \frac{\partial}{\partial x_i} \left( E_k \frac{\partial E}{\partial x_i} \right) + C_{E1} E \frac{P_{KIN}}{k} - C_{E2} \frac{E^2}{k}
\]

where \( E_k \) is the turbulent Prandtl number for diffusion of \( E \) and is usually assumed to be 1.3 and

\[P_{KIN} = \text{production of kinetic energy by shear}\]
Launer and Spalding [1972] give the following exact form of the kinetic energy differential equation. This equation is derived from the sum of all normal Reynolds stress terms.

\[ \frac{\partial k}{\partial t} = \frac{1}{2} \frac{\partial}{\partial x_i} \left[ u_i u_i \right] \]

The first term describes diffusion of kinetic energy. Launder and Spalding assume that it is proportional to the gradient of \( k \) and evaluate it by

\[ \frac{\partial k}{\partial x_i} = (C_k \frac{\partial k}{\partial \sigma_k}) \frac{\partial k}{\partial x_i} \]

where \( C \) is a constant and \( \sigma_k \) is the turbulent Prandtl number for diffusion of kinetic energy and is usually assumed equal to 1.0. The second term accounts for production of kinetic energy by shear. If a transport equation for the total normal stresses were developed from the steady velocities, the same term would occur but with opposite sign. Thus it represents the transfer of energy from the mean to the turbulent flow. It is modelled by the equivalent turbulent viscosity expression for the Reynolds stresses.

\[ \text{(ii)} = \nu (\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}) \frac{\partial U_i}{\partial x_j} \]
The third term is by definition the viscous dissipation of kinetic energy. The following equation for \( k \) results.

\[
\frac{Dk}{Dt} = \frac{\partial}{\partial x_i} \left( \frac{\nu_T}{k} \frac{\partial k}{\partial x_i} \right) + \nu_T \left( \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - E
\]

The final \( k \) and \( E \) equations are solved based upon the velocity and pressure found previously from the Reynolds equation of motion and the continuity equation. Once a new turbulent viscosity is found the equations of motion and continuity are solved again. The process is repeated until a consistent set of velocities, pressures, \( k \)'s and \( E \)'s is found.

**Evaluating constants.** The turbulence model has introduced 5 constants. Their values have been estimated by assumption and experiment and in some cases later optimized to fit the data [Launder and Spalding, 1972]. The constant \( C_{E2} \) was determined by measurements of grid produced turbulence. In this type of turbulence the production and diffusion terms in the \( E \) equation are zero so the constant was measured directly. It was found to range from 1.8 to 2.0. The constant \( C^* \) was found from experiments on local equilibrium shear layers, where production of kinetic energy equals its dissipation. Substituting the turbulent viscosity equation into the differential equation for kinetic energy and using these assumptions results in the following expression.

\[
C^* = \frac{\bar{u}_1 \bar{u}_2 \bar{u}_1}{k^2}
\]

From this it was found that \( C^* \) is about 0.09. The turbulent Prandtl numbers were set by Launder and Spalding [1972] at 1.0 for kinetic energy and 1.3 for its dissipation rate. They report that these are the result of optimization. The value of \( C_{E1} \) was estimated from flow near a
In this region the production of kinetic energy is equal to its dissipation rate and convection of the dissipation rate is neglected. The log law for velocity and these assumptions are used in the differential equation for $E$ and the following equation results.

$$C_{E_1} = C_{E_2} - \kappa^2 / \sigma_E (C^*)^{0.5}$$

The value of $C_{E_1}$ was equivalent to 1.44. These values have been used by many investigators on many flow patterns. Rodi [1980] summarized its use. See the applications section for a discussion of the use of the two equation model since Rodi's review.

**Boundary conditions.** Due to the elliptic nature of the equations, boundary conditions must be set on all variables. The two types of useable boundary conditions are known values and gradients. The gradient boundary conditions for the Reynolds equations of motion at walls are

- **normal:** $-P + \left( 2 \bar{\frac{U_n}{x_n}} \right) \mu$
- **traction:** $\left( \bar{\frac{U_n}{x_t}} + \bar{\frac{U_t}{x_n}} \right) \mu$

where $n$ indicates normal to the boundary and $t$ indicates tangential to the boundary, and $\mu$ is the dynamic viscosity.

$$\mu/\rho = \nu$$

These gradients represent the shear stress at the boundary.

The inlet velocities will either be specified or a pressure gradient will be applied by the normal gradient. The outlet velocities will either be set or their gradients will be zero, whichever is easier to solve. The wall boundary conditions will use either a known velocity, a zero gradient (zero stress) boundary, or a stress that was calculated using the law of the wall. The law of the wall will supply a
shear stress at a given distance from the wall if the turbulent kinetic energy is known there. The law of the wall is used to move the first node into the turbulent range of flow. The law of the wall is

\[ \frac{U}{U_*} = \frac{1}{\kappa} \ln (Y^+) + 5.45 \]

\( U_* \) is the friction velocity. It is defined as

\[ U_* = \sqrt{\frac{\text{WALL STRESS}}{\rho}} \]

\( \kappa \) is von Kar'man's constant and is usually given as .4 to .435. \( Y^+ \) is the distance from the wall that is normalized by a viscosity and wall stress.

\[ Y^+ = \frac{Y U_*}{\nu} \]

In the near wall region the production of turbulence approximately equals its dissipation rate and the convection and diffusion terms can be neglected. In this region the equations for transport of turbulent kinetic energy and its dissipation rate reduce to algebraic equations. These equations are

\[ k = U_*^2 / \sqrt{c_*} \]

\[ E = U_*^3 / \kappa Y \]

From this equation the wall stress can be found from a known kinetic energy, and vise versa. The law of the wall can also be used to predict velocity if empirical equations, such as Manning's or Chezy's, are available to determine traction stress.

The boundary conditions for the longitudinal and transverse velocity at the free surface are zero gradient. The vertical velocity will be zero. These conditions are only approximately true since a free
surface produces an inflection in the velocity profile. This discrepancy is not expected to effect the value of the calculations.

The boundary conditions for k and E are similar. Alfrink and van Rijn, [1983] suggest the following inlet boundary conditions for k and E in developed flows.

\[
\begin{align*}
    k_{INLET} &= \frac{U_*^2}{\sqrt{C_n}} (1 - \frac{Y}{H}) \\
    E_{INLET} &= \frac{U_*^3}{\kappa_y} (1 - \frac{Y}{H})
\end{align*}
\]

where \(Y\) is distance from the wall and \(H\) is the local depth. Inlet conditions for \(k\) and \(E\) don't seem to have a large effect on the outcome [Leschziner and Rodi, 1979, Alfrink and van Rijn, 1983, and Sala et al, 1980]. The outlet conditions are zero gradient. At the free surface a zero gradient for \(k\) and \(E\) is used.

So far this discussion has not mentioned boundary conditions for pressure. In setting up the matrix equations for an element the continuity equation is adjacent to the pressure in the unknowns. Setting the pressure at a node deletes the continuity equation there. Many researchers have indicated that setting pressures, especially at inlets and outlets, leads to poor convergence and sometimes strange results, [Martin, 1979], [Chung, 1978]. Jackson [1984] points out that the effect of deleting continuity at a point is to collect all round off errors at that point. Pressure anomalies or discontinuities should be expected at that location. For these reasons the pressure is usually set to zero at one wall node where all velocities are zero or in the interior of the flow [Gresho, Lee and Sani, 1980] and [Schamber and
Larock, 1981]. The rest of the boundary conditions for pressure are included in "normal" stress conditions.

Applications of the k-E Turbulence Model

The use of two and three dimensional turbulence models was summarized by Rodi [1980]. These applications include open channel flow, spillways, secondary flows in the horizontal plane, curved flow, stratified flow, internal hydraulic jumps, and jets of many descriptions. Ideriah [1978] applied this model to forced convection in a square cavity. HaMinh and Chassaing [1978] predicted flow in a sudden expansion. Alfrink and van Rijn [1983] applied it to expanding river sections, Krishnappen and Lau [1984] applied it to ice covered rivers. Naot studied non-symmetric roughness [1984]. Schamber and Larock [1981] investigated two dimensional flow in a sedimentation basin. Sala et al [1980] applied the model to two dimensional flow in a diffuser. Humphrey et al [1981] used it to calculate three dimensional flow in a square duct. Demuren and Rodi [1983] calculated the effect of a side inlet on the three dimensional flow pattern of a river. This last calculation used finite difference techniques and required from 400 to 500 iterations between the velocity and viscosity phase. The total time per iteration was 50 cpu-seconds on a UNIVAC 1100/82. This is one of the few three dimensional applications that did not make simplifying assumptions that produce a set of parabolic equations. With the exception of Schamber and Larock all of these were done using a finite difference technique.

While this model is more general than a mixing length model, it still cannot reproduce certain secondary motions in the plane perpendicular to the main flow. These secondary motions are called
Prandtl’s second type, [Prandtl, 1952] and are caused by unequal Reynolds stresses [Brundett and Baines, 1964]. However, the Bousinesq eddy viscosity concept assumes that these stresses are equal (isotropic). To obtain these secondary currents previous workers have had to resort to solving the Reynolds stress differential equations, for example [Launer, Reece and Rodi, 1975]. Rodi [1980] also gives examples of these models to channels, ducts, and jets. Hanjalic and Launder [1972b] modelled flow in a channel with different wall roughness and predicted stress levels within the flow and displacement of the point of maximum velocity. They also predicted the effect of wall damping on a flat boundary layer, flow in an annulus, and mixing layers. In order to simplify the calculation effort, algebraic equations for the Reynolds stress equations have been developed. These have been applied to jets [Ljuboja and Rodi, 1980], curved boundary layers [Irwin and Smith, 1975] and secondary channel currents [Naot and Rodi, 1982].

Smith [1984a] pointed out that there is no unique solution of $k$ and $E$ for a given velocity field. What makes the solution unique is the specification of boundary and initial conditions. For example, unless the initial estimate for $k$ and $E$ are close to the solution, the Newton method may converge to incorrect results, such as negative values for these two variables. Smith illustrates this point with a simple problem by showing multiple solutions with identical boundary conditions. The only differences were the initial conditions. He claims that this proves that the $k$ and $E$ model is mathematically improper. He has proposed a new turbulence model that is claimed to overcome these difficulties [Smith, 1984b]. The $k$-$E$ equations are non-linear and like all non-linear equations they are naturally subject to multiple
solutions. For example, in open channel hydraulics the third solution for depth with the specific energy held constant is negative. While this is a solution of a third order polynomial equation it is an unrealistic one. Due to the non-linear nature of the k and E equations the best estimate of the correct answer should be used to initialize the solution and the solution grid should be as fine as possible to minimize the chance of a solution wandering to an incorrect result. Perhaps the only effect of this problem is to over or under calculate the friction loss which will change the pressure and redistribute the velocity vectors. The total flow would not change. Data on k and E levels in turbulence is very scarce and its lack impedes proper investigation of this problem. More experience with the k and E model is required before discarding it in favor of yet another turbulence model.

Simplifications of the k-E Model

Due to the large amount of computer time involved in solution of three dimensional models, nearly all turbulence investigators have employed a "parabolizing" of the equations. This involves removing all terms that allow downstream conditions to affect the calculations. The numerical scheme proceeds downstream from the initial conditions. The terms neglected include shear stress and diffusion fluxes in the transverse plane. Also neglected is the downstream pressure. This requires that the water surface be known before the calculations start. The calculation time for a parabolized three dimensional finite difference calculation of curved open channel flow was about 30 minutes on a UNIVAC 1108 [Leschziner and Rodi, 1979]. Parabolizing doesn't allow for flow reversals to occur and will not be used in this study.
Finite Differences and Upwinding

Of the two major methods for solving partial differential equations, the finite difference method is older and is the method used primarily in the past for solving turbulence problems. Finite difference users have found that when solving Navier-Stokes type equations the use of central differences results in oscillations of the solution, particularly at large Reynolds numbers. They have resorted to a process called upwind differencing. This consists of using backward difference equations for the convection (acceleration) terms of the equations of motion. Which surrounding nodes are employed in the backward differencing depends on the direction of the velocity at the node. The effect of this operation is to damp oscillations, allow coarser grids for a given problem and make the truncation error relative to $O(h)$. The terms truncated during the differencing operation are similar to a first order diffusion equation so upwinding is said to produce false diffusion of turbulence. Davis and Mallinson [1976] concluded that upwinding introduces errors in the solution that can only be overcome by use of finer grids.

Since finite element solutions exhibit oscillations also, upwind weighting functions for finite elements have been proposed [Heinrich, et al 1977]. However, Gresho and Lee [1981] and Schamber and Larock [1981] discourage it because it hides problems caused by too large elements and can give a false impression of solution stability. It is their contention that making the grid fine enough will produce a stable solution. Because of the ease of using varying element size with the finite element method and the possible problems with false stability it
was decided that this project would use finite elements with traditional shape functions.

Galerkin's Method

The first step in solving the partial differential equations is to employ Galerkin's method to develop the finite element equations. Galerkin's method is used on differential equations of the general form

\[ Lu - f(x_j) = 0 \]

where \( L \) is a differential operator. The method works by forcing the error of the approximation to zero. The error is defined as

\[ Lu_{\text{approx}} - f(x_j) = E' = \text{error} \]

The error is forced to zero by making it orthogonal to the set of \( r \) linearly independent weighting functions, \( (N_r) \). Since the weighting functions span the space, the only function that is orthogonal to all of them is zero. Mathematically, an inner product is formed between the error and all of the weighting functions and is set equal to zero.

\[(N_r, E') = 0 \]

\[ \int_R N_r (Lu_{\text{approx}} - f(x_j)) \, dR = 0 \]

Weighting Functions

These same weighting functions can also be used to approximate the value of variables within a finite element grid, that is

\[ u_{\text{approx}} = N_i U_i \]

\[ p_{\text{approx}} = M_i U_i \]

where the repeating subscript indicates that the product is summed for all \( i \) in conformance with the Einstein summation convention. These shape functions are evaluated by the coordinate positions of the nodes.
of the elements. Figure 1 shows the two dimensional elements while Figure 2 shows the three dimensional version. The velocities are approximated in two dimensions by the eight node quadratic element and the pressure is approximated by the four node bi-linear element. In three dimensions, a twenty node quadratic and an eight node linear element is used. Both a quadratic and bi-linear element were used because previous investigators found that this combination prevents ill conditioning of the solution matrix [Chung 1978, p. 208]. The letters e, n, and s are the natural coordinates of the elements. They vary from -1 to 1 with the origin in the center of the element. The two and three dimensional shape functions for pressure are

\[ M_r = \frac{(1+e_0)(1+n_0)}{4} \]
\[ M_r = \frac{(1+e_0)(1+n_0)(1+s_0)}{8} \]

where r goes from 1 to 8 or from 1 to 20 respective for two and three dimensional elements. The subscript o indicates the product of the natural coordinate and the value of the node’s natural coordinate, e_b. For example, for shape function 1, e_0 is the product -1 x e, since the natural coordinate e_b at node number 1 is -1. Linear shape function number 1 for the two and three dimensional elements are

\[ M_1 = \frac{(1-e)(1-n)}{4} \]
\[ M_1 = \frac{(1-e)(1-n)(1-s)}{8} \]

The two dimensional quadratic shape functions at corner nodes is given by

\[ N_r = \frac{(1+e_0)(1+n_0)(e_0+n_0-1)}{4} \]

and at mid-side nodes they are given by

\[ (1-e^2)(1+n_0)/2 \]

where e_b=0, and
Figure 1. Two dimensional finite elements.

Figure 2. Three dimensional finite elements.
\[(1+e_0)(1-n^2)/2\]

where \(n_b=0\).

The three dimensional quadratic shape functions for corner nodes is given by

\[N_r=(1+e_0)(1+n_0)(1+s_0)(e_0+n_0+s_0-2)/8\]

The shape functions at mid-side nodes are given by

\[\begin{align*}
(1-e^2)(1+n_0)(1+s_0)/4 \\
(1+e_0)(1-n^2)(1+s_0)/4 \\
(1+e_0)(1+n_0)(1-s^2)/4
\end{align*}\]

where \(e_b=0\)

where \(n_b=0\)

where \(s_b=0\)

The use of natural coordinates make integration of the equations easier but requires a coordinate transformation operation for the numerical integration. The numerical integration will be of the form

\[\iiint f(e,n,s)dx_1dx_2dx_3 = \iiint f(e,n,s)|\det J|\,de\,dnd\,ds\]

where \(J\) is the Jacobian of the coordinate transform between the local and the cartesian coordinate system.

The Finite Element Equations

The equations of interest are the steady state Reynolds's equations of motion, the continuity equation, and the transport equations for kinetic energy, \(k\), and kinetic energy dissipation rate, \(E\). The equations are given here using the Einstein summation convention. The overbars have been dropped.
Reynolds equations of motion.

\[ \frac{\partial U_i}{\partial x_j} \quad \text{(convection)} \]
\[ = -\frac{1}{\rho} \frac{\partial}{\partial x_i} (P+\gamma h) \quad \text{(pressure)} \]
\[ + \frac{\partial}{\partial x_j} \left[ \nu_T \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3} \kappa \delta_{ij} \right] \quad \text{(dissipation)} \]

Continuity equation.
\[ \frac{\partial U_1}{\partial x_1} + \frac{\partial U_2}{\partial x_2} + \frac{\partial U_3}{\partial x_3} = 0 \]

\( k \) equation.
\[ U_i \frac{\partial k}{\partial x_i} = \frac{\partial}{\partial x_i} \left( \nu_T \frac{\partial k}{\partial x_i} \right) + \nu_T \frac{\partial U_i}{\partial x_j} \frac{\partial U_j}{\partial x_i} - E \]

\( E \) equation.
\[ U_i \frac{\partial E}{\partial x_i} = \frac{\partial}{\partial x_i} \left( \frac{\nu_T}{\nu} \frac{\partial E}{\partial x_i} \right) + C_{E1} E \quad \frac{\partial U_i}{\partial x_j} \frac{\partial U_j}{\partial x_i} \frac{\partial U_i}{\partial x_j} - C_{E2} \frac{E^2}{k} \]

Applying the Galerkin method to these equations results in the following equations. The quadratic weighting function is used for the inner product involving the Reynold's equations of motion and the \( k \) and \( E \) transport equations. The linear shape function is used for the continuity equation.

Reynolds equations of motion.

For \( i = 1 \)
\[ 0 = \int \int \int N_r \left[ U_1 \frac{\partial U_1}{\partial x_1} + U_2 \frac{\partial U_1}{\partial x_2} + U_3 \frac{\partial U_1}{\partial x_3} + \frac{1}{\rho} \frac{\partial}{\partial x_1} (P+\gamma h) \right] \]
\[-\frac{1}{\partial x_1} ((v+v_T)(2 \frac{\partial u_1}{\partial x_1} - \frac{2}{3} k)) - \frac{\partial}{\partial x_2} ((v+v_T)(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1})) \]
\[- \frac{\partial}{\partial x_3} ((v+v_T)(\frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1})) \] 
\[dx_1 dx_2 dx_3 \]

For \( i = 2 \)

\[0 = \iiint N_r \left[ U_1 \frac{\partial u_2}{\partial x_1} + U_2 \frac{\partial u_2}{\partial x_2} + U_3 \frac{\partial u_2}{\partial x_3} + \frac{1}{\rho} \frac{\partial}{\partial x_2} (P + \gamma h) \right. \]
\[- \frac{\partial}{\partial x_1} ((v+v_T)(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2})) - \frac{\partial}{\partial x_2} ((v+v_T)(2 \frac{\partial u_2}{\partial x_2} - \frac{2}{3} k)) \]
\[- \frac{\partial}{\partial x_3} ((v+v_T)(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2})) \] 
\[dx_1 dx_2 dx_3 \]

For \( i = 3 \)

\[0 = \iiint N_r \left[ U_1 \frac{\partial u_3}{\partial x_1} + U_2 \frac{\partial u_3}{\partial x_2} + U_3 \frac{\partial u_3}{\partial x_3} + \frac{1}{\rho} \frac{\partial}{\partial x_2} (P + \gamma h) \right. \]
\[- \frac{\partial}{\partial x_1} ((v+v_T)(\frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3})) - \frac{\partial}{\partial x_2} ((v+v_T)(\frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3})) \]
\[- \frac{\partial}{\partial x_3} ((v+v_T)(2 \frac{\partial u_3}{\partial x_3} - \frac{2}{3} k)) \] 
\[dx_1 dx_2 dx_3 \]

Continuity Equation.

\[0 = \iiint M_r \left[ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \right] dx_1 dx_2 dx_3 \]

\( k \) equation.

\[0 = \iiint N_r \left[ U_1 \frac{\partial k}{\partial x_1} - \frac{\partial}{\partial x_i} (\frac{v_T}{\sigma} \frac{\partial k}{\partial x_i}) - v_T (\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) \frac{\partial u_i}{\partial x_j} \right] dx_1 dx_2 dx_3 \]
The equations that result from the Galerkin process may contain derivatives higher than first order. Since the weighting functions are continuous in the variable but not in its derivatives, any second and higher order derivatives must be manipulated to first order. For example, the dissipation part of the equations of motion are:

For \( i = 1 \)

\[
N_r \left[ \frac{\partial}{\partial x_1} ((v+v_T)(2 \frac{\partial U_1}{\partial x_1}) - \frac{2}{3} k) + \frac{\partial}{\partial x_2} ((v+v_T)(\frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1})) 
+ \frac{\partial}{\partial x_3} ((v+v_T)(\frac{\partial U_1}{\partial x_3} + \frac{\partial U_3}{\partial x_1})) \right] = \cdots, r = 1, \ldots, 20
\]

The part of the equations of motion that contain second order derivatives are shown for the case \( i=1 \).

Integration by parts is applied next.

For \( i = 1 \)

\[
\int \int \int \left\{ \frac{\partial}{\partial x_1} \left[ (N_r((v+v_T)(2 \frac{\partial U_1}{\partial x_1}) - \frac{2}{3} k) \right] + \frac{\partial}{\partial x_2} \left[ N_r((v+v_T)(\frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1})) \right] \right\}
\]
The Green-Gauss theorem reduces the order of 1/2 of these terms. Happily this results in naturally occurring gradient boundary conditions. This operation is also done on the pressure terms which causes the necessary boundary conditions to occur.

For i = 1

\[
\int \int N_r ((v + v_T) (2 \frac{\partial U_1}{\partial x_1}) - \frac{2}{3} k) dx_2 dx_3 + \int \int N_r ((v + v_T) (\frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1})) dx_1 dx_3 \\
+ \int \int N_r ((v + v_T) (\frac{\partial U_1}{\partial x_3} + \frac{\partial U_3}{\partial x_1})) dx_1 dx_2 - \int \int \frac{\partial N}{\partial x_j} ((v + v_T) (\frac{\partial U_1}{\partial x_j} + \frac{\partial U_1}{\partial x_1})) dx_1 dx_2 \\
- \frac{2}{3} k \delta_{1j} dx_1 dx_2 dx_3
\]

The next step is to substitute expressions incorporating the shape functions for the variables. For example,

\[ u_1 = N_1*U1(1) + ... + N_{20}*U1(20) \]

where U1(r) is the value of the variable at local node r. Derivatives are approximated as,

\[ \frac{\partial u_1}{\partial x_j} = (\frac{\partial N_1}{\partial x_j})*U1(1) + ... + (\frac{\partial N_{20}}{\partial x_j})*U1(20) \]
All the variables and non-constant knowns in these equations can be approximated in terms of shape functions and derivatives of shape functions. Upon substitution a system of non-linear equations occurs with the value of the velocity components and pressure at the nodes as the unknowns. The boundary conditions become the known or force vector. These equations are assembled for all of the elements. This requires that global node numbers are used. Figure 1 shows the local node numbers for a two dimensional element. Figure 3 shows the global node numbers for a small problem. The integral equations that result from the Galerkin method for the Reynolds' equations of motion, the continuity equation, and the $k$ and $E$ transport equation are given below.

Reynolds' equations of motion.

\[ i = 1 \]

\[
\iiint N_r \left[ U_1 \frac{\partial U_1}{\partial x_1} + U_2 \frac{\partial U_1}{\partial x_2} + U_3 \frac{\partial U_1}{\partial x_3} + g \frac{\partial h}{\partial x_1} \right] - \frac{\partial N_r}{\partial x_1} \frac{1}{\rho}(P) \\
+ \frac{\partial N_r}{\partial x_1} \left[ (v+v_T) \left( 2 \frac{\partial U_1}{\partial x_1} - \frac{2}{3} k \right) \right] + \frac{\partial N_r}{\partial x_2} \left[ (v+v_T) \left( \frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1} \right) \right] \\
+ \frac{\partial N_r}{\partial x_3} \left[ (v+v_T) \left( \frac{\partial U_1}{\partial x_3} + \frac{\partial U_2}{\partial x_1} \right) \right] \right] dx_1 dx_2 dx_3 \\
= \iiint N_r \left[ \frac{-1}{\rho} (P) + 2(v+v_T) \frac{\partial U_1}{\partial x_1} - \frac{2}{3} k_s \right] dx_2 dx_3 \\
+ \iiint N_r \left[ (v+v_T) \left( \frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1} \right) \right] dx_1 dx_3 \\
+ \iiint N_r \left[ (v+v_T) \left( \frac{\partial U_1}{\partial x_3} + \frac{\partial U_2}{\partial x_1} \right) \right] dx_1 dx_2 \\
r = 1, \ldots, 20\]
Figure 3. Global node numbers for a three element grid.

\[ i = 2 \]

\[ \iiint \left\{ N_r \left[ \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2} + u_3 \frac{\partial u_2}{\partial x_3} + g \frac{\partial h}{\partial x_2} \right] - \frac{\partial N_r}{\partial x_2} \frac{1}{\rho} f(P) \right. \]

\[ + \frac{\partial N_r}{\partial x_1} \left[ (v+\nu_T) \frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \right] + \frac{\partial N_r}{\partial x_2} \left[ 2(v+\nu_T) \frac{\partial u_2}{\partial x_2} - \frac{2}{3} k \right] \]

\[ + \frac{\partial N_r}{\partial x_3} \left[ (v+\nu_T) \frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2} \right] \right\} dx_1 dx_2 dx_3 \]

\[ = \int N_r \left[ (v+\nu_T) \left( \frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \right) \right] dx_2 dx_3 \]

\[ + \int N_r \left[ \frac{-1}{\rho} f(P) + 2(v+\nu_T) \frac{\partial u_2}{\partial x_2} - \frac{2}{3} k \right] dx_1 dx_3 \]

\[ + \int N_r \left[ (v+\nu_T) \frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2} \right] dx_1 dx_2 \quad r = 1, \ldots, 20 \]
\[ i = 3 \]

\[
\iiint N_r \left[ \frac{\partial u_3}{\partial x_1} + u_2 \frac{\partial u_3}{\partial x_2} + u_3 \frac{\partial u_3}{\partial x_3} + g \frac{\partial h}{\partial x_3} \right] - \frac{\partial n_r}{\partial x_3} \frac{1}{\rho} \exp \{ p \}
\]

\[
+ \frac{\partial n_r}{\partial x_1} \left[ (v + v_T) \left( \frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3} \right) \right] + \frac{\partial n_r}{\partial x_2} \left[ (v + v_T) \left( \frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3} \right) \right]
\]

\[
+ \frac{\partial n_r}{\partial x_3} \left[ 2(v + v_T) \frac{\partial u_3}{\partial x_3} - \frac{2}{3} k \right] \right) \right) \right) dx_1 dx_2 dx_3
\]

\[ = \iiint N_r \left[ (v + v_T) \left( \frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3} \right) \right] dx_2 dx_3
\]

\[ + \iiint M_r \left[ (v + v_T) \left( \frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3} \right) \right] dx_1 dx_3
\]

\[ + \iiint N_r \left[ -\frac{1}{\rho} \exp \{ p \} + (v + v_T) \left( \frac{2}{3} \frac{\partial u_3}{\partial x_3} - \frac{2}{3} k \right) \right] dx_1 dx_2 dx_3
\]

\[ H \text{ is height above a datum and } k_s \text{ is } k \text{ on the surface.} \]

The continuity equation becomes

\[
\iiint M_r \left[ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \right] dx_1 dx_2 dx_3 = 0 \quad r = 1, \ldots, 8
\]

The K equation.

\[ 0 = \iiint \left\{ N_r \left[ U_1 \frac{\partial k}{\partial x_1} + U_2 \frac{\partial k}{\partial x_2} + U_3 \frac{\partial k}{\partial x_3} \right] + \frac{1}{\sigma_k} \left[ C^* \frac{k^2}{E} \frac{\partial k}{\partial x_1} \frac{\partial n_r}{\partial x_1} \right.
\]

\[ + C^* \frac{k^2}{E} \frac{\partial k}{\partial x_2} \frac{\partial n_r}{\partial x_1} + C^* \frac{k^2}{E} \frac{\partial k}{\partial x_3} \frac{\partial n_r}{\partial x_1} \right] \right) \right) \right) \right) \right) \right) \left[ C^* \frac{k^2}{E} \frac{\partial u_1}{\partial x_1} \frac{\partial n_r}{\partial x_1} \]

\[ + N_r E \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \right) \r
The E equation.

\[ 0 = \iint [N_r C^* \frac{k^2}{E} \frac{\partial E}{\partial x_2} dx_1 dx_3 + \iint N_r C^* \frac{k^2}{E} \frac{\partial E}{\partial x_3} dx_1 dx_2] \quad r = 1, \ldots, 20 \]

Numerically Integrating the Equations

The following example shows the process used in this project to numerically integrate the two-dimensional continuity equation.

\[ \iint [M] \{\partial u_1/\partial x_1\} dx_1 dx_2 = \iint [M]^T [B] \{u\} |\det J| W(e,n) d\eta \]

where [M] is the vector of the linear shape functions, [B] is the matrix of derivatives of the quadratic shape function, W(e,n) is the Gauss-Legendre weighting function, J is the Jacobian of the coordinate transform, and superscript T denotes transpose.

\[ [M] = \left[ (1-e)(1-n) (1+e)(1-n) (1+e)(1+n) (1-e)(1+n) \right] / 4 \]
$W(e,n)$ is assigned according to the value of the local coordinate. All of these variables are in terms of the spatial coordinates and the local coordinates. In this study fifth order numerical integration is used, so 25 locations within each element are evaluated for each of the above terms and are multiplied by their respective weights. The sum of these operations is the value of the integral for the element. A three dimensional numerical integration is done at 125 locations within each element.

**Solving the Equations**

The solution to the equations is sought by using Newton's method. The current values of the variables is used to evaluate all the integrals (the functions, $\{F\}$). The Jacobian matrix $[D]$ is formed by calculating the derivatives of all the functions with respect to each variable. This is a different Jacobian from the one mentioned above. The next value of the variables is calculated by the matrix equation

$$u^{n+1} = u^n - [D]^{-1}[F].$$

The corrections are made and the process is repeated until the corrections become less than some tolerance. Problems will arise with
Convergence of Newton's method if the initial guess is not "close enough".

The Jacobian matrix that arises during the solution of the Navier-Stokes equations is unsymmetric, due to the non-linear convection terms, non-positive definite, since the continuity equation doesn't contain pressure as a variable, is not diagonally dominant and can contain zeros on the diagonal [Olson, 1977]. It is non-singular if proper boundary conditions are applied.

**Sparse Matrix Routines**

The matrix encountered in solving the Navier-Stokes equations is very sparse. In two dimensional problems the non-zeros in the solution matrix can be as few as 12 percent [Schamber and Larock, 1981]. Routines are available to take advantage of this sparsity. The United Kingdom Atomic Energy Authority (Harwell) has produced two versions of their sparse matrix routines [Duff, 1980]. The current version solves the matrix in four steps. It first reorders the matrix to block lower triangular form. It then does a row permutation to obtain a non-zero diagonal. Next it performs a LU factorization on the diagonal blocks and solves the LU decompositions. Since repeated solutions of similarly sparse matrices is desired, the routine saves the procedure it used and can apply it to subsequent matrices. Its big disadvantage is that it requires a lot of integer arrays to keep track of variable location and the operations performed on them. For example, a two dimensional problem with 1290 unknowns with 213830 "non-zeros" in the matrix required the following integer*2 arrays: IRN(575,000), ICN(1,750,000), IKEEP(1300,5), IVECT(575,000), JVECT(575,000), an integer*4 array,
IW(1290), and the real*4 array that holds the matrix, A(1,750,000). The previous and unsuccessful arrays were about forty percent smaller. While Harwell recommends 2 to 4 times the number of non-zeros as the size of the largest array, they also point out that the array size is a function of the size of the permuted matrix and can exceed these numbers.

The intent of the Harwell authors was to pass only the non-zeros numbers of the matrix to their subroutines. The programs developed in this study pass all of the elements of the matrix rows that lie between the first and last non-zero. The reasons for doing this are: (1) to increase the likelihood of obtaining a series of similarly sparse matrices, (2) the intended method was tried on a two dimensional problem and it required about the same size matrices. It is important to obtain similarly sparse matrices since the first solution of the matrix takes more time to solve than the second and subsequent matrices. Passing some zeros does not increase the required array size since this array must hold the decomposition elements also.

The current Harwell subroutine requires 37,000 bytes of Fortran 4 code on an IBM 370/168. The older version is smaller, requiring 15,500 bytes [Duff, 1980]. At 1795 fortran statements the current version is a substantial subroutine. The older version lacks the block triangular function and requires much smaller index arrays. The 1290 unknown problem mentioned previously only required the following integer*4 arrays: IP(1300,13) and IX(1,000,000). The A array was sized at A(500,000). If further space reductions are required the arrays can be converted to integer*2. However, use of integer*2 arrays limits the number of non-zeros less than 27000. In this study the older version
could not be used since it failed to complete even one iteration of this problem in 6 cpu-hours on a VAX 11/780.

During the matrix permutations the routine will compress out zero valued elements in an attempt to keep storage requirements to less than the size of the array ICN. If ICN has been undersized the computation time can be increased dramatically during the compression phase [Duff, 1980].

These subroutines use pivoting during the LU decomposition phase to minimize round off error. The strategy for choosing the pivot has two criteria. The first is to choose the one with the smallest Markowitz count. That is, the smallest product of non-zero entries in the pivot's row and column. The second criteria is that this pivot not be less than the variable U times the largest value in its row. The variable U is set by the user, usually to a value between .1 and .25. A smaller value of U doesn't constrain the choice of pivot as much. A larger value improves solution stability but increases storage requirements for the permuted matrix and the array ICN and solution times. Error statements are printed if U is too small while extreme computation times result if U is too large. See the conclusions section for a discussion of the effect of U's size on computation time.
THE COMPUTER PROGRAM

Separate computer codes for both two and three dimensional laminar and turbulent flow were developed during this project. FEM2D is given in appendix A. These codes prepare the matrix and known vector that is passed to the Harwell sparse matrix subroutines. While other matrix solving routines could be used to solve the system of equations, the current Harwell version is the only one capable of solving the largest problem encountered during this project in a feasible time.

The Computer Code

Figure 4 gives the outline of the steps taken in the three dimensional computer program to solve the differential equations. The two dimensional program is similar. Rather than solve for all variables in one operation, the programs first use an assumed constant turbulent viscosity to solve the Reynolds equations of motion and continuity equations. The velocities that result from these calculations are then assumed known and a solution for turbulent viscosity is found. Dividing the six equations for each node into two sets reduces the number of unknowns by almost one half. This has been done in every previous turbulence investigation. The major advantage of this is to decrease the amount of storage required for the problem. A drawback of this procedure is that the convergence rate drops from quadratic to linear. A computer program was developed to solve a small two dimensional problem for all five variables at once. The maximum number of unknowns increased from 303 to 587. While the number of iterations decreased, the total cpu time was about the same (1 hour).
Figure 4. Outline of the computer program Fem3d.
Since some of the terms contain the product of a velocity and a derivative of a velocity the equations are non-linear and an implicit method such as Newton's must be used.

**Input Data**

The input for the program is held in four data files. FEM3D.DAT contains nineteen constants. DELU1, DELU2, DELU3, DELP, DELK, and DELE determine the size of the change in variable that is used to calculate derivatives for the Jacobian. MAX1, MAX2, and MAX3 control the number of iterations of the velocity-pressure phase, the k-E phase, and the number of passes between them. TOL is set to the value of the maximum acceptable Newton method corrections. C1, C2, C3, CE1, and CE2 are constants in the turbulence model. LAM tells the program whether the problem is laminar or turbulent and which phase starts the turbulent computations. ALPHAI1 and ALPHAI2 are relaxation factors for the corrections calculated by Newton's method. OUT controls the printing of input values. The value of the variable OUT may be set at a value less than 11 to obtain an echo of all of the input data for checking. UVAR (U) is a factor that controls decomposition sparsity and stability in the Harwell subroutines.

INITIAL3D.DAT contains the initial value of the variables by global node number. The order of the data is: the global node number, the three velocities (U1, U2, U3), pressure (P), turbulent kinetic energy (k), k's dissipation rate (E), and the three shear stress values for gradient boundary conditions at walls (T1, T2, T3).

GRID3D.DAT contains three sets of data. The first contains eight constants. These are the number of elements, the number of nodes,
kinematic viscosity, density, number of known boundary conditions, and
number of gradient boundary conditions. The second set lists the
element and local node number for each global node number. These are
stored in the array GNN. For example, if the value stored in GNN(15,19)
is 200, then local node number 19 of element 15 is global node number
200. The third set gives the global node number with the X1, X2, and X3
coordinates and height of the water surface above the elevation datum.

BC3D.DAT contains two types of data. The first set gives the known
boundary conditions. These are, in order, the global node number, the
variable number, and the value of the boundary condition. U1 is
variable 1, P is variable 4, and E is variable 6. Variable numbers 7
and 8 can be specified to set initial values of k and E. This tells the
program to first use the specified value as a boundary condition and
then calculate k and E on the next pass from previously calculated
velocity data at the node. Known boundary conditions are stored in the
array DUIT. For example, if the value held in DUIT(41,2) is 1, then the
second variable (U2) at global node number 41 is a known boundary
condition.

The second data set is the non-zero gradient boundary conditions.
These can be applied in three directions on six faces. For example,
Faces 1 and 2 are perpendicular to the direction of U1, while faces 3
through 6 are parallel to it. The order for this data is: data set
number, direction of the momentum equation, face number, total number of
elements having this condition, the sign of the gradient (+/- 1), and a
list of these elements. Following this is the value of the applied
normal gradients. These are constant values and this is where the
pressure gradient is applied. The normal gradient is
The non-zero gradient boundary conditions are limited to element faces that are parallel to the coordinate axis. Sample input files are shown in the appendix B.

**Shape Functions**

The program next evaluates shape functions using the local coordinates, called XI, ETA, and ZETA in the program, and derivatives of shape functions with respect to the local coordinates. The linear shape functions are SFL and the quadratic shape functions are SF5. The derivatives are DNDC5, Dnda5, and DNDZ5, for the derivatives with respect to XI, ETA, and ZETA. Their two dimensional counterparts that are used for non-zero gradient boundary conditions are SFBC, DNDABC, and DNDZBC.

If the problem being solved is laminar, then the variable LAM (in FEM3D.DAT) is set to an integer greater than 0. The program will proceed to the velocity-pressure phase and end once the convergence criteria have been satisfied or the maximum number of iterations exceeded. If the problem is turbulent, then LAM should be set equal to 0 if the program should start with the velocity-pressure phase and set to an integer less than 0 if it should start with the k-E phase.

**Velocity-Pressure Phase**

The following describes the calculations of the velocity-pressure phase. The calculations proceed by element through the finite element grid. The value of the integrals and the contribution to the Jacobian
are calculated for each element before moving on. For each new element the program calls the subroutine DERIVE. This subroutine returns the derivatives of the shape functions with respect to the global coordinates X1, X2, and X3. These derivatives are the variables DNDX1, DNDX2 and DNDX3. It then evaluates the variables EWE1, EWE2, EWE3, and PEE. These are the sum of the shape functions times the value of U1, U2, U3, and P at each node in the element. The derivatives of these variables with respect to the global coordinates, such as DU1DX1 and DU3DX3 are also calculated here. The variables are combined into the integral functions developed by Galerkin's method. These functions are called F1, F2, F3, and F4, and represent the three Reynolds' equations of motion and the continuity equation. The functions are numerically integrated at five locations in each direction. The derivatives of these functions with respect to each variable are evaluated next. These derivatives are defined by a finite difference,

$$\frac{\partial F}{\partial V} = \frac{F_{\text{CHANGED}} - F}{\text{del } V}$$

The derivatives are done in the following order. First U1 is changed at a node. The changed functions F1 through F4 are calculated by the subroutine DF1. The derivatives of F1 through F4 at each node with respect to U1 at node L1 is calculated and added to the one dimensional array A. A stores the two dimensional Jacobian array in row order. U1 is restored to its original value and the next U1 is changed. After the U1's have all been changed the U2's, U3's, and P's are changed in order. After this the program moves to a new element and starts to calculate the functions and their derivatives for that element.
Non-zero Gradient Boundary Conditions

Once this has been done for each element the program calculates the non-zero gradient boundary conditions. These are of two types, parallel to the element surface and normal to it. The normal stress (BCNAT) is inputed. The parallel, or traction, stress (TAU) is calculated from shear stress values given or previously calculated from kinetic energy at the node. In the three dimensional program both normal and shear stress gradient boundary conditions are numerically integrated to find the boundary condition. These are added to their respective functions F1 through F4.

Subrouting SOLVE

Subroutine SOLVE is called next. It reads each row of the Jacobian (A), identifies the first and last non-zero in each row, and saves them and all elements between them. All of the zeros that are external to this series are compressed out. It also evaluates the integer arrays IRN, ICN, IVECT, JVECT, IW, and creates the real array W. The first four of these hold the row and column position of all of the saved elements. If this is the first iteration SOLVE calls the Harwell subroutines MA28A and MA28C. If the second or later matrix has the same sparsity as the previous one SOLVE will call MA28B and MA28C. MA28B will decompose the matrix using the procedure worked out by MA28A. Error flags from the Harwell subroutines are passed through SOLVE to the main program and error statements are printed. Calculations will halt if it is a fatal error. The solutions are passed from MA28C through SOLVE to the main program. The corrections are made to the value of the variables. If the corrections are greater than the desired tolerance
(TOL) and if the maximum number of iterations (MAX1) has not been exceeded, the program returns to the start of the velocity-pressure phase and starts the cycle over. If the corrections are smaller than the variable TOL then the iterations cease. The program ends here if the problem is laminar. If it is turbulent, then the program shifts to the k-E phase.

**Turbulent Kinetic Energy-Dissipation Rate Phase**

The structure of the k-E phase is similar to the velocity-pressure phase. The variables are KAY and EE, the derivatives are of the form DKDX1 or DEDX3, and the functions are F5 and F6. Boundary conditions for this phase are all knowns or zero gradients. The program can calculate k and E at a wall from the previous velocity at the node using the logarithmic velocity profile and the simplified k and E differential equations. Once the functions and their derivatives are calculated the subroutine SOLVE is called and the corrections to the variables are made. Once this phase has converged or iterated the desired number of times the program returns to the velocity-pressure phase. The number of passes between the two phases is limited by the variable MAX3.

If desired, the corrections made during each phase can be altered by the relaxation factors ALPHA1 and ALPHA2 for V-P and k-E phase respectively. This was useful only on the larger problems.

The size of the corrections is measured with two norms (ANORM and BNORM). ANORM is the absolute value of the sum of all of the corrections (not relaxed) and BNORM is the absolute value of the largest correction. Both of these are printed to a "log" file when the program
is operated in the batch mode and can be viewed during calculations to check progress. This allows divergent problems to be killed before they waste too much cpu time. The program also has a subroutine (CPU) that calculates the present cpu time usage.

The program uses single precision but could easily be converted to double precision. The Harwell subroutines come with substitute fortran code to make the change easier. Single precision was found adequate during this project.

Program Output

The program outputs to the file FEM3D.OUT. It prints the updated values of each iteration. It also prints the number of the iteration, total cpu-seconds at each step, the actual bandwidth of each Jacobian, the number of variables being solved, the total number of non-zeros being passed to the Harwell subroutines, and the two convergence norms. The graphics package "DISSPLA" by Issco Software generated contour, vector, and grid point plots from the output of these programs. These were plotted on a Calcomp model 1051 plotter.

Data Generation Programs

Programs had to be written to generate the data files GRID.DAT, INITIAL.DAT, and BC.DAT and their three dimensional versions GRID3D.DAT, INITIAL3D.DAT, and BD3D.DAT. NOD.FOR creates GRID.DAT. INIT.FOR generates the file INITIAL.DAT and BC.FOR develops BC.DAT. These programs generate data for rectangular areas. The vortex problem required a cylindrical grid so a special data program (VORDAT.FOR) was written for it. Typical data generation program are shown in appendix C.
RESULTS AND CONCLUSIONS

Many different flow problems in two and three dimensions have been attempted during the preparation of the computer program. Table 1 summarizes 29 of these problems. The acceleration of gravity has been ignored in all problems with the exception of the three-dimensional vortex.

Laminar Problems

The first two dimensional laminar problem is shown in Figure 5. It represents flow between two parallel walls. This is considered a velocity driven flow since the velocities are set on the boundaries. While it is out of the laminar range, with a Reynolds number of 8217, it was calculated using the laminar program. The second problem also represents flow between two parallel walls and is shown in Figures 6 and 7. This flow was caused by application of a pressure (normal) gradient so it is considered a pressure driven flow. The velocity results of this problem agree with the analytical solution. The pressure field shows slightly inconsistent pressures vertically but a reasonable horizontal gradient has developed. The pressure was set only at the node in the lower right corner.

The three dimensional velocity driven version of this problem is shown in Figure 8. The three dimensional pressure driven version of this problem is shown in Figure 9. Both velocity solutions agree with the two dimensional results. The pressure results for the velocity driven problem is shown in Figure 10. The pressure field for this and other velocity driven flows is often inconsistent. In this case the
### Table 1. Laminar and turbulent two and three dimensional flow problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Number of Unknowns</th>
<th>Number of Nodes</th>
<th>Number of Elements</th>
<th>Converge?</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laminar two dimensional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Velocity driven channel flow</td>
<td>46</td>
<td>20</td>
<td>5</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>2. Pressure driven channel flow</td>
<td>46</td>
<td>20</td>
<td>5</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Wall driven flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Even Spaced Grid</td>
<td>46</td>
<td>20</td>
<td>5</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>4. Even Spaced Grid</td>
<td>47</td>
<td>47</td>
<td>9</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>5. Even Spaced Grid</td>
<td>47</td>
<td>47</td>
<td>9</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>6. Even Spaced Grid</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td>Initial condition from previous uneven grid solution</td>
</tr>
<tr>
<td>7. Uneven Spaced Grid</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td>Results unreasonable</td>
</tr>
<tr>
<td>8. Uneven Spaced Grid</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>9. Uneven Spaced Grid</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>10. Uneven Spaced Grid</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>11. Uneven Spaced Grid</td>
<td>1,290</td>
<td>649</td>
<td>196</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>12. Uneven Spaced Grid</td>
<td>1,290</td>
<td>649</td>
<td>196</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>13. Uneven Spaced Grid No. 1</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>14. Uneven Spaced Grid No. 2</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>15. Uneven Spaced Grid No. 2</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>16. Uneven Spaced Grid No. 2</td>
<td>216</td>
<td>133</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>17a. Pressure Driven Channel Flow</td>
<td>360</td>
<td>140</td>
<td>12</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>17b. Velocity Driven Channel Flow</td>
<td>360</td>
<td>140</td>
<td>12</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>18a. Pressure Driven Channel Flow</td>
<td>360</td>
<td>140</td>
<td>12</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>18b. Velocity Driven Channel Flow</td>
<td>360</td>
<td>140</td>
<td>12</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Turbulent two dimensional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel Flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91. Velocity Set at Inlet</td>
<td>284</td>
<td>264</td>
<td>35</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>92. Pressure 0 at exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td>65</td>
<td>51</td>
<td>12</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td>51</td>
<td>12</td>
<td>45,500</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td>219</td>
<td>80</td>
<td>280</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td>303</td>
<td>147</td>
<td>40</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Turbulent three dimensional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel Flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>301</td>
<td>147</td>
<td>40</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>24.</td>
<td>301</td>
<td>147</td>
<td>40</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>25.</td>
<td>301</td>
<td>147</td>
<td>40</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>26.</td>
<td>301</td>
<td>147</td>
<td>40</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>27.</td>
<td>301</td>
<td>264</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>28.</td>
<td>301</td>
<td>264</td>
<td>36</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>29a. Vortex Flow</td>
<td>307</td>
<td>160</td>
<td>20</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>29b.</td>
<td>420</td>
<td>193</td>
<td>28</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>29c.</td>
<td>1369</td>
<td>602</td>
<td>9</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>29d.</td>
<td>1191</td>
<td>517</td>
<td>77</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
**Figure 5.** Problem 1. Velocity driven channel flow, 8 unknowns.

\[ \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} = 0 \]

\[ -p + 2 \mu \frac{\partial u_1}{\partial x_1} = 0.00001 \]

\[ -p + 2 \mu \frac{\partial u_1}{\partial x_1} = 0 \]

\[ u_2 = 0 \]

**ALL BOUNDARIES**

**Figure 6.** Problem 2. Pressure driven laminar channel flow, 40 unknowns.
Figure 7. Velocity (ft/sec) and pressure (lbs/ft²) results of problem 2.
Boundary Conditions for problem 18

<table>
<thead>
<tr>
<th>UPSTREAM</th>
<th>DOWNSTREAM</th>
<th>RIGHT</th>
<th>LEFT</th>
<th>TOP</th>
<th>BOTTOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>Set</td>
<td>$\frac{\partial u_1}{\partial x_1} = 0$</td>
<td>$\frac{\partial u_1}{\partial x_2} = 0$</td>
<td>$\frac{\partial u_1}{\partial x_3} = 0$</td>
<td>Set</td>
</tr>
<tr>
<td>U2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U3</td>
<td>0</td>
<td>0</td>
<td>$\frac{\partial u_3}{\partial x_2} = 0$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$P = 0$ on upstream face on bottom row

Figure 8. Problem 18. Velocity driven three dimensional laminar channel flow between parallel walls, 446 unknowns.
Boundary Conditions for problem 17

<table>
<thead>
<tr>
<th>UPSTREAM</th>
<th>DOWNSTREAM</th>
<th>RIGHT</th>
<th>LEFT</th>
<th>TOP</th>
<th>BOTTOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>$\frac{p}{\rho} - 2v \frac{\partial u_1}{\partial x_1} = 0$</td>
<td>$\frac{p}{\rho}$</td>
<td>$\frac{\partial u_1}{\partial x_1} = 0$</td>
<td>$\frac{\partial u_1}{\partial x_2} = 0$</td>
<td>$\frac{\partial u_1}{\partial x_3} = 0$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$u_3$</td>
<td>0</td>
<td>0</td>
<td>$\frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3} = 0$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$P = 0$ on downstream face on bottom row

Figure 9. Problem 17. Pressure driven three dimensional laminar channel flow between parallel walls, 140 unknowns.
Figure 10. Pressure results (lbs/ft^2) for three dimensional laminar velocity driven flow streamwise along center row of elements for problem 18. Pressure does not change with X_2.

Figure 11. Pressure results (lbs/ft^2) for three dimensional laminar pressure driven flow streamwise along center row of elements for problem 17. Pressure does not change with X_2.
pressure results may have improved if the pressure specification point had been moved to the downstream end. The pressure results of the pressure driven flow are shown in Figure 11. The pressure results for this problem are much better.

The third problem is laminar two dimensional wall driven flow at Reynolds numbers \( (Re=vL/v) \) from 100 to 3300. Convergence was accomplished at Reynolds numbers of 100 and 1000. The general problem is shown in Figure 12. In these problems the top wall moves and the fluid viscosity causes the flow to circulate. The grids for these problems varied from nine elements to 196 elements. The solution for the Re=100 problem on the 36 element unevenly spaced grid (problem 14) is shown in Figure 13. The vectors are proportional to the wall velocity vector shown at the top. The tail of the vector indicates the finite element grid point. Very small, low velocity eddies have formed in the lower corners and near the upper left corner. The Re=1000 results on a different 36 element unevenly spaced grid (problem 16) are shown on Figure 14. The same problem on 196 evenly spaced elements (problem 12) is shown in Figure 15. Only one half of the velocity results are shown because of the number of unknowns and the figure's size. While both of the Re=1000 solutions show that the eddies have grown in size, the 36 element solution shows them as being larger.

### Turbulent Problems

The first and second turbulence problems (19 and 20) were two dimensional channel flow between two parallel walls. The problems are shown in Figures 16 and 17. Both of these had velocities set at the inlet with zero velocity gradient at the outlet. Problem 19 had both
Figure 12. Problem 6. Two dimensional laminar wall driven flow.

Figure 13. Problem 14. Velocity results (ft/sec) of wall driven flow at Re=100 on 36 unevenly spaced elements.
Figure 14. Problem 16. Velocity results (ft/sec) of wall driven flow at Re=1000 on 36 unevenly spaced elements.

Figure 15. Problem 12. Velocity results (ft/sec) of wall driven flow at Re=1000 on 196 evenly spaced elements. Only one-half of results shown.
Figure 16. Problem 19. Two dimensional turbulent channel flow between parallel walls, 153 unknowns.

Figure 17. Problem 20. Two dimensional turbulent channel flow between parallel walls, 150 unknowns.
velocities set at all boundaries while \( k \) and \( E \) were set only at the wall. Problem 20 had the velocities, \( k \), and \( E \) set only at the inlet and wall. The rest of the boundary conditions were zero gradients. Inlet velocities and \( k \) and \( E \) at the wall were set by the law of the wall and the simplified \( k \) and \( E \) differential equations. Because turbulent viscosity varies as the square of kinetic energy this created an uneven turbulent viscosity gradient at the inlet. The pressure and turbulent viscosity results for problem 20 are shown in Figures 18 and 19. Turbulent viscosity shows only slight decrease downstream while maintaining the pattern set by the boundary conditions. This indicates that there is not a unique turbulent viscosity field for a given velocity field.

The next five turbulent problems (21 - 25) were two dimensional channels with circular outlets. These outlets represent idealized pump inlet pipes. Velocities into the "pumping pit" were set while the walls were zero gradient (slip) boundary conditions. The outlet boundary conditions were zero stress. This boundary condition is quite useful and allows the flow to adjust itself to the outlet. The general problem is shown in Figure 20. The velocity solution to problem #25 is shown in Figure 21. The pressure and turbulent viscosity results are shown in Figures 22 and 23. The pressure contours show some flow occurring against the pressure gradient. Turbulent viscosity increased in the downstream direction in this problem.

The solutions progressed from lower to higher velocity. The results of the previous solution were used to estimate the initial conditions for the succeeding problem. The inlet \( k \) and \( E \) values required for convergence increased faster than the inlet velocity.
Figure 18. Pressure results (lbs/ft²) for problem 20.

Figure 19. Turbulent viscosity results (ft²/sec) for problem 20.
\[
\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} = u_2 = \frac{\partial k}{\partial x_2} = \frac{\partial e}{\partial x_2} = 0
\]

Wall

\[
\frac{\partial k}{\partial x_2} - \frac{\partial e}{\partial x_2} = \frac{\partial u_1}{\partial x_2} - \frac{\partial u_2}{\partial x_1} = u_2 = 0
\]

\[
\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} = \frac{\partial k}{\partial x_1} = \frac{\partial e}{\partial x_1} = \frac{\partial e}{\partial x_2} = 0
\]

Inlet

Center line

Outlet

Wall

<table>
<thead>
<tr>
<th>Problem #</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>0.05</td>
<td>0.10</td>
<td>0.15</td>
<td>0.20</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( k )</td>
<td>0.007</td>
<td>0.014</td>
<td>0.07</td>
<td>1.4</td>
</tr>
<tr>
<td>( E )</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.001</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Figure 20. Problems 21 to 25. Turbulent flow from a channel to a circular outlet in two dimensions, 587 unknowns.
Figure 21. Velocity results (ft/sec) for problem 25.
Figure 22. Pressure results (lbs/ft\(^2\)) for problem 25.

Figure 23. Turbulent viscosity results (ft\(^2\)/sec) for problem 25.
The next two turbulent problems (27, 28) were three dimensional versions of the velocity driven turbulent channel flow between two parallel walls. The problem is shown in Figure 24. Problem 26 had constant values of $k$ and $E$ specified at the bottom wall while problem 27 used wall shear stresses to calculate their values. No changes in the value of the variables occurred across the flow. Since the results were nearly identical they are shown as one line in Figure 25. The pressure, turbulent kinetic energy, dissipation rate, and turbulent viscosity results are shown in Figures 26 to 32. The contours tend to have square corners due to the small number of plotting points. Turbulent kinetic energy and dissipation rate show a decrease in the downstream direction. The turbulent viscosity shows a slight decrease near the center line (top of figure) and an increase at the wall. Overall it has changed very little in the horizontal direction. This was the largest three dimensional problem to converge. While it is not a complicated problem it does show that the three dimensional computer code does not diverge from the solution.

The largest two dimensional turbulent problem is flow under a sluice gate. The physical problem is shown in Figure 33. The depth of the incoming flow was 1.0 feet at an average velocity of 2.0 feet per second. The sluice gate was set at 0.5 feet above the floor. The free surface elevation downstream of the sluice gate was set according to the potential flow solution for this problem. The finite element grid has 140 elements and is shown in Figure 34. The first formulation assumed no stress (slip) boundary conditions along the sluice gate and the floor. The velocity-pressure phase would not converge until the boundary and initial values of $k$ and $E$ were set above a lower limit.
Figure 24. Problems 27 and 28. Three dimensional turbulent channel flow between parallel walls, 790 unknowns.
Figure 25. Velocity results (ft/sec) of problems 27 and 28.

Figure 26. Pressure results (lbs/ft²) of problem 27 streamwise along center row of elements.
Figure 27. Turbulent kinetic energy results (ft²/sec²) of problem 27 streamwise along center row of elements.

Figure 28. Turbulent kinetic energy as a percentage of inlet values for problem 27 streamwise along center row of elements.
Figure 29. Turbulent kinetic energy dissipation rate results (ft$^2$/sec$^3$) of problem 27 streamwise along center row of elements.

Figure 30. Turbulent kinetic energy dissipation rate as a percentage of inlet values for problem 27 streamwise along center row of elements.
Figure 31. Turbulent viscosity results (ft²/sec) of problem 27 streamwise along center row of elements.

Figure 32. Turbulent viscosity as a percentage of inlet values for problem 27 streamwise along center row of elements.
Figure 33. Problem 26. Two dimensional turbulent flow under a sluice gate, 1551 unknowns.

Figure 34. Finite element grid for problem 26.
Upon trying to solve for $k$ and $E$ their values were small enough that some became negative during the Newton iterations and ANORM did not get smaller than about 0.02 before it became very large. The negative turbulent viscosity caused the problem to diverge at this point. The small turbulent viscosity values for the solution show that the boundary conditions basically created a potential flow problem, as expected.

The second formulation added shear stress along the floor and sluice gate. Values of $k$ and $E$ were also set along the free surface downstream of the sluice gate. The boundary conditions for this problem are shown in Figure 35. These steps kept the turbulent viscosity from going to zero. The solution is shown in Figures 36 to 38. The next step on this problem would be to reduce the boundary and initial conditions for $k$ and $E$ at all points and set zero gradient boundary conditions for $k$ and $E$ along the free surface.

The last turbulent problem is a three dimensional free surface vortex. It represents the flow of a free surface vortex that was created in a 2.08 foot diameter circular tank by Daggett and Keulegan [1974]. The tank had variable vanes around its circumference to set tangential velocity and a 4 inch diameter hole in the center of the floor for an outlet. The depth of the vortex was 1.0 feet and its total flow was 0.671 cfs. Because of the number of unknowns it was decided to try to solve only one quarter of the flow field with zero tangential velocity. The problem is shown in Figures 39 and 40. No valid solution of this problem was obtained. Three different grids were tried. These had up to 91 elements, 602 nodes, and 1364 unknowns. A solution for the velocity variables of the largest of these could not be attempted because the virtual memory of the VAX 11/780 was exceeded. The largest
\frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1} = 0\quad U_2 = \frac{\partial k}{\partial x_2} - \frac{\partial E}{\partial x_2} = 0

U_1 = 0

\frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1} = f_1(k,E)

k = 2.19
E = 6.35

\frac{\partial U_1}{\partial x_2} + \frac{\partial U_2}{\partial x_1} = f_1(k,E)

U_2 = 0
k = f_2(U_1)
E = f_3(U_1)

Figure 35. Boundary conditions for problem 26.

Figure 36. Velocity results (ft/sec) for problem 26.
Figure 37. Pressure results (lbs/ft²) for problem 26.

Figure 38. Turbulent viscosity results (ft²/sec) for problem 26.
Figure 39. Plan view of three dimensional vortex flow. Diameter is 2.08 feet, depth is 1.0 feet, and total flow is .671 cfs.

Figure 40. Problem 29. Three dimensional vortex flow, up to 2398 unknowns.
number of unknowns (k and E) that were successfully iterated was 1034. The first iteration of this many unknowns took about one hour and 50 minutes of cpu time.

Because of these difficulties the problem was further reduced to include just the area in the rotational core of the vortex. Because of the time and storage requirements it was decided to abandon the vortex problem until a larger and faster computer is available.

Approximately 430 runs were made on these problems and variations of them. Performance tests and graphic displays took another 75 runs. The large number of runs include those to debug the programs, learn how to pose each problem correctly, how to use the Harwell subroutines, and find elements of appropriate size to adequately solve the problem.

**Boundary Conditions**

Posing the problems correctly includes proper boundary and initial conditions. The most troublesome boundary values to obtain are pressures. In the two dimensional problems pressure is only specified at one node. In the three dimensional channel problems pressures were specified at nodes in a horizontal line from side to side. Without specifying these pressures the three dimensional problems would not converge. The best place for the pressure specification point was usually at the down stream end of the flow. This allows a smoother pressure gradient along the flow lines. The purpose of setting pressure at these nodes is to provide a pressure datum. If it is desired to drive the flow by a pressure gradient, the normal gradient boundary condition is applied. This results in pressures that are very close to the expected pressures. For example the pressures calculated in the second of the three dimensional turbulent problems (27) were very close
to the applied normal gradient boundary condition. Setting pressures as a known boundary condition deletes the continuity equation at that node. Some of the literature refers to this relationship as "pressure enforcing continuity".

In velocity driven problems pressures will sometimes be inconsistent. For example, flow will occur against a pressure gradient or the pressure will alternate positive and negative across an inlet or along a streamline. Since the velocity boundary conditions dominate the flow the pressure discontinuities don't really affect the outcome. Even so, this is a numerical weakness.

The ratio of surface nodes to interior nodes in these problems is quite large. With the exterior velocities specified, the smallest values inside the flow, i.e. the pressure, could be far from their solution values while the norm of the solution is small.

Inconsistent pressures could also be due to the finite element formulation used. While velocity is evaluated at eight nodes in the two dimensional element, pressure is only evaluated at four. In three dimensional elements the numbers are twenty and eight. The effect of the number of nodes on the pressure field is seen in the two dimensional wall driven flows. Solutions were found for Reynolds number of 1000 with grids of 36 and 196 elements. The pressure field of the 36 element problem was based on 49 pressure nodes and was inconsistent. The pressure field of the 196 element problem was based on 225 pressure nodes and was very well defined and reasonable. The difference is due to the effect of the small number of pressure nodes on the solution as well as on the plotting routine. The contour plots showing pressure for these two are shown in Figures 41 and 42.
Figure 41. Pressure results (lbs/ft$^2$) of problem 16.

Figure 42. Pressure results (lbs/ft$^2$) of problem 12.
Initial Conditions

Laminar problems will often converge if the initial velocity and pressure are set to zero. The two dimensional wall driven flow would only converge from initial velocity and pressure settings of zero. This flow has a large central circular current with small secondary currents in the corners. Other initial conditions with a linear velocity change away from the wall were tried but would not converge.

The turbulent two dimensional circular outlet problem required a trial and error process to find the correct initial conditions. The problem was started from smaller velocities and after convergence was obtained, the results were used to estimate the values for the next solution. All turbulent problems require initial non-zero conditions for the $E$ variable, since it appears in the denominator of the model. The two and three dimensional channel problems were all started from initial velocity and pressure values given by the analytical solution or from experimental results. The sluice gate problem used a logarithmic inlet velocity profile and trial and error estimates for the initial $k$ and $E$ conditions. The vortex problems were started from the data given by Daggett and Kuelegan [1974]. Their data only included the velocity in the outer half of the problem domain so all initial velocities inside the rotational part of the vortex were estimated. The initial conditions for $k$ and $E$ were estimated from their values at the outer floor.

Grid Definition

It is typical of standard Galerkin finite element programs for fluid flow problems to converge only if the grid is fine enough, especially in areas of rapid change of variable value [Gresho, Lee, and
Sani, 1980]. This is not surprising since finite difference techniques are also sensitive to grid spacing. An approach in finite difference solutions is to increase the number of grid points until the solution does not change from that obtained using the larger grid.

Success in solving the wall driven flow problem was associated with the grid size. At Reynolds number 1000 the 36 uniformly sized element problem diverged. The norms of the iterations (ANORM) went as follows.

1.695
3.237
12.510
8.820
7.647
33.732
18.086
19.849
12.435
57.366

This pattern of approaching convergence and then diverging is typical of problems where the grid is too coarse for the applied change of variables.

Once the same 36 elements were redistributed to give a fine spacing at the walls the solution converged. The iteration sequence for the successful problem was

2.127
2.845
2.080
1.651
1.304
0.5796
0.2600
0.04765
0.003326
0.00002503

Convergence was not obtained for the uniformly spaced grid even when initial values used for this problem were interpolated from the solution. Grids using 100 and 144 uniformly sized elements were tried
but would not converge. The number of uniform elements was increased to 196 and convergence was obtained. The grid spacing at the walls was the same for the unevenly spaced 36 element grid as the evenly spaced 196 element problem. The probable explanation for this is that the size of the element must be inversely proportional to the rapidity of the variable changes. Small eddies form in the corners of this problem and the elements there must be small enough to describe the changes.

The wall driven flow was also tried at a Reynolds number of 100. Corner eddies are seen in this problem and the evenly spaced 36 element grid was successful.

Grid problems were also experienced with the turbulent circular outlet flow. The first attempt using 24 elements would not converge. Convergence was obtained with a 40 element grid.

Wall Driven Flows

Wall driven flows have been solved extensively in the literature. Most of these used the stream function and vorticity variable. The stream function solution obtained by Olson and Tuann [1979] for Reynolds number 100 and 1000 are shown in Figures 43 and 44. The centerline velocity calculated from their solution and the results for the finite element solution are shown in Figures 45 and 46. At Re=100, the 36 element grid solution agrees with the results of Olson and Tuann. However, at Re=1000 it diverges from their results at the lower wall. The 196 element solution agrees well with their results.
Figure 43. Stream function results of wall driven flow at Re=100 [Olson and Tuann, 1979].

Figure 44. Stream function results of wall driven flow at Re=1000 [Olson and Tuann, 1979].
Figure 45. Comparison of results of wall driven flow at Re=100. U1 velocity along vertical center line, [Olson and Tuann, 1979].

Figure 46. Comparison of results of wall driven flow Re=1000. U1 velocity along vertical center line, [Olson and Tuann, 1979]
Problem Size

Major problems were encountered during attempts to solve the largest two dimensional wall driven flow. This flow had 1290 unknowns. Since the program version in use at that time needed 12,000,000 bytes of storage, the first difficulty was storage limitations. While the Vax 11/780 has sufficient memory for this problem, the University computer center doesn't normally allow any single user that much memory. This problem was solved with a request to the proper person. Once the program was set to work it ran for six hours without returning one solution of the matrix. Different matrix solving algorithms were tried. The IMSL subroutine LINV3F also worked for six hours without success. The older version of the Harwell subroutine (MA18) was tried for three hours without returning one iteration. MA18 had been successful on smaller problems with 218 unknowns. Two questions appeared concerning the earlier successes. The first question was whether the Jacobian matrix was solvable at all. The matrix was printed out and the entries checked for comparative size. The entries were small (10^-4) but seemed close in magnitude. Calculating the determinant of the Jacobian was tried. This was possible on the smaller problems, but overflow occurred during calculations on the larger ones. While the determinants of the smaller problems were small numbers, they weren't zero. Both the Harwell and IMSL subroutines will indicate if a matrix is singular. Since neither had, it was concluded that the Jacobian was non-singular.

The second question was whether the output obtained was really a solution of the matrix equation. The Jacobian was multiplied by the solution and the result was within the limits of machine accuracy of the known vector.
Since these last two tests indicated success on small problems, a large finite difference groundwater problem with 1275 unknowns was tried. The older version of the Harwell subroutines solved this rapidly. The matrix for this problem is easier to solve since it is diagonally dominant and symmetrical and has more zeros on the off diagonals.

Performance of Harwell Subroutines

Since number of unknowns was not the problem, it was decided to vary some of the input values the Harwell subroutines use. Table 2 shows the results of these tests. The variable U has a large effect on the solution times of both versions of these subroutines. For example, with U equal to .1, the newer version of the Harwell subroutines did the first iteration of a 642 unknown problem in 179 cpu-seconds. With U equal to .25, it used 1403 cpu-seconds. At U equal to .15, it used 1060 cpu-seconds. With U held constant at .1, the newer Harwell version obtained the first iteration of the 642 unknown problem in 178 cpu-seconds and did the 938 unknown problem in 755 cpu-seconds. The next iteration took 63 and 132 cpu-seconds, respectively. Once the effect of U was learned, the wall driven flow problem was attempted. At U=.1 the 1290 unknown problem converged to a solution after 7 iterations. The first iteration took 1030 cpu-seconds while the subsequent ones took 240 cpu-seconds. Table 2 also shows the time saved on the second and subsequent iterations. This savings is only available if each succeeding matrix has the same sparsity.
Table 2. Effect of U and number of unknowns on matrix solution times on VAX 11/780 for Harwell sparse routines for two dimensional wall driven flow at Re = 1000.

<table>
<thead>
<tr>
<th>Number of Unknowns</th>
<th>First Iteration</th>
<th>Subsequent Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MA28A-MA28C</td>
<td>MA28B-MA28C</td>
</tr>
<tr>
<td></td>
<td>(cpu seconds)</td>
<td>(cpu seconds)</td>
</tr>
<tr>
<td>218 0.1 24.30</td>
<td>6.85</td>
<td></td>
</tr>
<tr>
<td>642 0.1 219 213 178.6</td>
<td>65.0 66.3 63.4</td>
<td></td>
</tr>
<tr>
<td>0.20 962.94</td>
<td>88.2</td>
<td></td>
</tr>
<tr>
<td>0.25 1403.4</td>
<td>84.89</td>
<td></td>
</tr>
<tr>
<td>938 0.1 755.07</td>
<td>132.0</td>
<td></td>
</tr>
<tr>
<td>1290 0.1 1029.6</td>
<td>238.7</td>
<td></td>
</tr>
</tbody>
</table>

Computer Size

During the difficulty with solution times the possibility of using a larger computer was investigated. The University of Utah computer was available only to their faculty. The Corp of Engineers at Vicksburg, Mississippi was interested in solving the vortex problem but could only help after this project was done. The National Science Foundation has a program to allow access to super computers but that process requires a proposal and at least five months to accomplish. The Cray computers used in the NSF program will not have adequate storage for the larger problems in this project until the newer Cray machines are installed in September, 1986.
Solution Times

Tables 3 and 4 present solution times for problems at various numbers of unknowns. Table 3 presents total solution times while Table 4 shows time for each iteration. These were done on the University's VAX 11/780 computer. The small three dimensional turbulent problems (430 velocity and pressure unknowns) took over 10 hours to converge. The VAX is considered a mini-computer and is slower than most of the machines cited by other turbulence investigators. For example, the CDC 7600 used by Schamber and Larock [1981] completed 53 passes between the v-p and k-e phase of a 1783 unknown problem in 6.2 cpu minutes. The VAX is adequate for the two dimensional problems attempted. However, it has too little storage and calculates too slowly for all but the simplest three dimensional problems. Since development of programs and trying to find the right boundary and initial conditions takes so much time a slow machine is a handicap. A large portion of the time used for this project was spent in adjusting to the size and time limitations of the VAX. This did make the code more efficient with storage.

Conclusions

Finite element computer codes in two and three dimensions have been written that solve both laminar and turbulent flow. These codes use the two equation (k & E) turbulence model to evaluate a turbulent viscosity. These codes have been tested with 29 different flow problems ranging from Re=100 to Re=330,000. The largest Reynolds number flow that was successfully solved (2 x 10^5) was the sluice gate problem. The limitations of time and computer size have prevented successful completion of the three dimensional vortex problem originally sought.
Table 3. Total CPU time for solution of selected problems.

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Total # of Unknowns</th>
<th>Total # of passes</th>
<th>Total # of iterations</th>
<th>Total CPU time hours:minutes:seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>218</td>
<td>NA</td>
<td>5</td>
<td>00:09:51</td>
</tr>
<tr>
<td>9</td>
<td>938</td>
<td>NA</td>
<td>6</td>
<td>00:36:28</td>
</tr>
<tr>
<td>11</td>
<td>1290</td>
<td>NA</td>
<td>10</td>
<td>01:42:05</td>
</tr>
<tr>
<td>12</td>
<td>1290</td>
<td>NA</td>
<td>10</td>
<td>01:46:22</td>
</tr>
<tr>
<td>14</td>
<td>218</td>
<td>NA</td>
<td>5</td>
<td>00:12:05</td>
</tr>
<tr>
<td>16</td>
<td>218</td>
<td>NA</td>
<td>10</td>
<td>00:21:58</td>
</tr>
<tr>
<td>17</td>
<td>72</td>
<td>NA</td>
<td>10</td>
<td>00:20:52</td>
</tr>
<tr>
<td>18</td>
<td>72</td>
<td>NA</td>
<td>4</td>
<td>00:02:10</td>
</tr>
<tr>
<td>19</td>
<td>153</td>
<td>10</td>
<td>60</td>
<td>00:13:40</td>
</tr>
<tr>
<td>20</td>
<td>150</td>
<td>9</td>
<td>41</td>
<td>00:10:52</td>
</tr>
<tr>
<td>22</td>
<td>587</td>
<td>10</td>
<td>56</td>
<td>01:54:34</td>
</tr>
<tr>
<td>23</td>
<td>587</td>
<td>4</td>
<td>29</td>
<td>N/A</td>
</tr>
<tr>
<td>24</td>
<td>587</td>
<td>8</td>
<td>36</td>
<td>01:15:08</td>
</tr>
<tr>
<td>25</td>
<td>587</td>
<td>6</td>
<td>24</td>
<td>00:52:05</td>
</tr>
<tr>
<td>26</td>
<td>1851</td>
<td>18</td>
<td>183</td>
<td>14:04:07</td>
</tr>
<tr>
<td>27</td>
<td>790</td>
<td>6</td>
<td>43</td>
<td>20:00:21</td>
</tr>
<tr>
<td>28</td>
<td>790</td>
<td>4</td>
<td>26</td>
<td>13:00:12</td>
</tr>
</tbody>
</table>

Two dimensional laminar flows

Wall driven trench flow

Three dimensional laminar channel flow

Two dimensional turbulent channel flow

Circular outlet from a channel

Flow under a sluice gate

Three dimensional turbulent channel flow
Table 4. Matrix solution times for two and three dimensional problems at various numbers of unknowns - on VAX 11/780 using Harwell subroutines at \( U = 0.1 \).

<table>
<thead>
<tr>
<th>Number of Unknowns</th>
<th>First Iteration (cpu seconds)</th>
<th>Second Iteration (cpu seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Two dimensional</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>218</td>
<td>24.30</td>
<td>6.85</td>
</tr>
<tr>
<td>642</td>
<td>218.8</td>
<td>65.0</td>
</tr>
<tr>
<td></td>
<td>212.8</td>
<td>66.28</td>
</tr>
<tr>
<td></td>
<td>178.6</td>
<td>63.41</td>
</tr>
<tr>
<td>938</td>
<td>755.07</td>
<td>132.01</td>
</tr>
<tr>
<td></td>
<td>307.04</td>
<td>102.46</td>
</tr>
<tr>
<td></td>
<td>486.26</td>
<td>110.21</td>
</tr>
<tr>
<td>1290</td>
<td>972.1</td>
<td>224.8</td>
</tr>
<tr>
<td></td>
<td>516.95</td>
<td>267.17</td>
</tr>
<tr>
<td></td>
<td>1029.59</td>
<td>238.72</td>
</tr>
<tr>
<td>822</td>
<td>122.71</td>
<td>43.96</td>
</tr>
<tr>
<td>912</td>
<td>171.49</td>
<td>92.05</td>
</tr>
<tr>
<td>1029</td>
<td>402.29</td>
<td>122.16</td>
</tr>
<tr>
<td>1029</td>
<td>480.67</td>
<td>133.28</td>
</tr>
<tr>
<td><strong>Three dimensional</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>25.03</td>
<td>9.2</td>
</tr>
<tr>
<td>430</td>
<td>433</td>
<td>218.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>218.0</td>
</tr>
<tr>
<td>360</td>
<td>201.17</td>
<td>92.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>93.05</td>
</tr>
<tr>
<td>446</td>
<td>696</td>
<td>229</td>
</tr>
<tr>
<td>474</td>
<td>765.8</td>
<td>233.8</td>
</tr>
</tbody>
</table>
While not enough is known about the vortex problem or the capabilities of super computers to guarantee the success of this problem it should be attempted. The two dimensional sluice gate problem is about the size of the largest feasible problem with the current program and computer.

While the Harwell subroutines are fast they also require a lot of storage. The largest storage array must be approximately 5 times the number of non-zeros in the Jacobian matrix. This has required the use of a single precision array of size 1,750,000 for the two dimensional wall driven flow problem. This is 7,000,000 bytes of storage. The required integer*2 index arrays took another 7,000,000 bytes. The two dimensional sluice gate problem would require about two-thirds of this amount. It is hard to estimate storage requirements since they depend on the LU decomposition process that the Harwell subroutines employ. The solution times of the Harwell subroutines are also very sensitive to the value of the variable U.

These finite element computer codes are very sensitive to boundary and initial conditions and element size. The number of successes to attempts was about 1 to 50 for the largest problems. This poor showing was due to the number of possible combinations of boundary and initial conditions and finding a grid with small enough elements. As experience is gained the ratio will improve.

These results should be seen as encouragement to the further development and improvement of finite element solutions to turbulent flow problems.
SUGGESTIONS FOR FURTHER STUDY

1. The sluice gate problem should be completed by reducing the boundary and initial conditions for k and E at all points and changing the k and E boundary conditions along the free surface downstream of the sluice gate to zero gradient.

2. The three dimensional vortex problem should be continued on a faster computer. An axisymmetric solution of this problem could also be attempted.

3. The model should be tried on many other flow problems and the results compared to experimental data. These problems could include open channel transitions, junctions, constrictions, hydraulic jumps, and gate structures. A two dimensional depth-averaged model could be used to study currents in large bays or reservoirs.

4. With very fine grids (and many unknowns) the program could be used to study pressure phenomena such as cavitation.

5. Data on the levels of turbulent kinetic energy and its dissipation rate is very scarce. Obtaining this data would require sophisticated laboratory equipment not yet available at Utah State University. Perhaps a series of experiments that studied the relationship between turbulent viscosity and empirical pressure loss equations would help provide guidance in specifying input values of the k and E variables. Any data in this area would be referred to often by turbulence modelers.

6. The possibility of non-unique solutions of the k-E equations needs to be investigated.
7. A separate sediment mechanics model could be combined with this program to study the formation of sediment bedforms and erosion in channels and around obstructions.

8. Other methods of solving the non-linear equations that are not as sensitive to initial conditions should be investigated.
SELECTED BIBLIOGRAPHY


Appendix A

FEM2D.FOR
Job FEM20 (1524) queued to VAXA_TXAO on 26-NOV-1986 09:18 by user UF7105, UIC [UF7105], under account 000744 at priority 100, started on printer _VAXA$TXAO: on 26-NOV-1986 09:53 from queue VAXA_TXAO.
**2-D TURBULENT FLOW**

**USING THE HARWELL SPARSE MATRIX Routines IN SINGLE PRECISION**

---

**VARIABLES IN "FEM.OAT"**

DELU1, DELU2, DELU, DEL

CHANGES MADE TO THESE VARIABLES TO CALCULATE THE DERIVATIVES FOR THE JACOBIAN FOR NEWTON'S METHOD

MAX1 - THE MAXIMUM ITERATIONS ALLOWED PER CYCLE ON THE VELOCITY & PRESSURE VARIABLES

MAX2 - THE MAXIMUM ITERATIONS ALLOWED PER CYCLE ON THE K AND E VARIABLES

MAX3 - THE MAXIMUM PASSES BETWEEN THE V-P AND K-E PHASE

TOL - THE VALUE OF THE NORM OF THE CORRECTIONS AT WHICH CONVERGENCE IS ACCEPTED

C1, C2, C3, C4, C5 - CONSTANTS IN THE K-E TURBULENCE MODEL

LAM -

>0 LAMINAR PROBLEMS

=0 TURBULENT PROBLEMS - START CALCULATIONS WITH KINEMATIC VISCOSITY

<0 TURBULENT PROBLEM - START CALCULATIONS WITH KINETIC ENERGY/DISSIPATION

ALPHA1 - RELAXATION FACTOR FOR THE V-P CORRECTIONS

ALPHA2 - RELAXATION FACTOR FOR THE K-E CORRECTIONS

OUT = 11 REGULAR OUTPUT, NO DATA ECHO

<11 IN ADDITION TO OUTPUT, PROGRAM ECHOES DATA

UVAR = A FACTOR IN THE HARWELL SUBROUTINE THAT CONTROLS SPARCITY AND STABILITY OF THE LU DECOMPOSITION

---

**VARIABLES IN "INITIAL.DAT"**

FACTOR1, FACTOR 2, AND FACTOR 3 MULTIPLY THE INITIAL CONDITIONS TO ALLOW CHANGING THE PROBLEM

NGN1 = GLOBAL NODE NUMBER

U1 = INITIAL GUESS FOR U1 AT GLOBAL NODE NUMBER NGN1

U2 = INITIAL GUESS FOR U2 AT GLOBAL NODE NUMBER NGN1

P = INITIAL GUESS FOR P AT GLOBAL NODE NUMBER NGN1

E = INITIAL GUESS FOR E AT GLOBAL NODE NUMBER NGN1

T1 = INITIAL GUESS FOR TAU (SHEAR STRESS) AT NODE NUMBER NGN1 IN THE U1 DIRECTION

T2 = INITIAL GUESS FOR TAU (SHEAR STRESS) AT NODE NUMBER NGN1 IN THE U2 DIRECTION

---

**VARIABLES IN "GRID.DAT"**

NELT = NUMBER OF ELEMENTS IN FINITE ELEMENT GRID

---

**VARIABLES IN "BC.OAT"**

PHI = DISTANCE FROM WALL TO NEAREST ELEMENT GRID POINT. USED IN APPLYING "LAW OF THE WALL" TO OBTAIN VELOCITY AT THE NEAREST NODE.

FACTOR1 AND FACTOR 2 MULTIPLY THE B.C.'S

NODE = GLOBAL NODE AT WHICH KNOWN BC IS APPLIED

NVARK = NUMBER OF VARIABLE THAT IS KNOWN (1=U1, 6=E)

1U1, 2U2, 5P, 6E, 6K

7=K TO BE SET FROM PREVIOUS VELOCITIES

8=E TO BE SET FROM PREVIOUS VELOCITIES

BCX = VALUE OF THE KNOWN BOUNDARY CONDITION (DIRICHLET)

THE FOLLOWING ONLY IF NBCG .GT. 0

DIR = DIRECTION OF THE MOMENTUM EQUATION 1=U1

FACE = SIDE OF ELEMENT ON WHICH BC ACTS

1U1 POSITIVE IN DOWNSWEEP DIRECTION

U2 POSITIVE UPWARD

1 UPSTREAM FACE NORMAL TO U1

2 DOWNSTREAM FACE NORMAL TO U1

3 BOTTOM FACE

4 UPPER FACE

NBCEL = TOTAL NUMBER OF ELEMENTS ON WHICH THE BC OPERATES

NBCL = NUMBER OF THE ELEMENT ON WHICH THE BC OPERATES

BCNAT = IS THE NON-ZERO VALUE OF THE NORMAL CONSTANT BOUNDARY CONDITION THAT INVOLVES PRESSURE AND THE PARTIAL DERIVATIVE OF THE NORMAL VELOCITY IN THE NORMAL COORDINATE DIRECTION

---

**FUNCTIONS**

GNN(200, 8), N201(700, 6), N20E, NVARK, NVARG, NBCI(5003, 8, CC), NBCG.
DO 1=1,NODE
READ(1,*,END=107)NODE1,A1,A2,A3,A4,A5,A6,A7,A8,
U1(NODE)=A1*FACTOR1
U2(NODE)=A2*FACTOR1
P(NODE)=A3*FACTOR1
EKIN(NODE)=A4*FACTOR2
E(NODE)=A5*FACTOR2
T1(NODE)=A6*FACTOR3
T2(NODE)=A7*FACTOR3
IF(OUT .LT. 11)THEN
WRITE(4,204)NODE1,U1(NODE),U2(NODE),P(NODE),
EKIN(NODE),E(NODE),T1(NODE),T2(NODE),END IF END DO
FORMAT(10X, I6, 3X, 7F11.7)
CONTINUE
WRITE(4,*) 'THE NUMBER OF INITIAL NODE VALUES IS'
II=II+1
WRITE(4,*) III
IF(II .LT. NODE) THEN
WRITE(4,*) ' THIS IS LESS THAN SPECIFIED'
GO TO 9999
END IF
C READ AND ASSIGN THE KNOWN BOUNDARY CONDITIONS
FIVE=5 SIX=6
READ(7,*),PHI,FACTOR4,FACTOR5
WRITE(4,*)
WRITE(4,*) PHI, FACTOR4, FACTOR5 ARE'
WRITE(4,*) PHI, FACTOR4, FACTOR5 ARE'
WRITE(4,*) IF(OUT .LT. 11) THEN
WRITE(4,*) ' THIS IS LESS THAN SPECIFIED'
GO TO 9999
END IF
NBCK=0
DO 1=1,NBC
READ(7,*,END=105)NODE,NVARK,BCK(NODE,NVARK)
IF(NVARK .EQ. 7) THEN
CUTOUC(NODE,FIVE)=1
GO TO 100
END IF
IF(NVARK .EQ. 8) THEN
CUTOUC(NODE,SIX)=1
GO TO 100
END IF
CUTOUC(NODE,NVARK)=1
100 CONTINUE
GO TO 101,102,103,104,105,106,107,108
101 U1(NODE)=BCK(NODE,NVARK)*FACTOR1
GO TO 109
102 U2(NODE)=BCK(NODE,NVARK)*FACTOR1
26 NOV 1986 09:16
CONTINUE
WRITE(*,*) 'THE NUMBER OF NON-ZERO GRADIENTS IS:'
II=1
IF(I I I LT . NBCG) THEN
WRITE(*,*) 'THIS IS LESS THAN SPECIFIED'
GO TO 999
END IF
WRITE(*,*) DATA IS ENTERED
CONTINUE

C LET THE FUN BEGIN
C THE BAND WIDTH
C ASSIGN THE TURBULENT VISCOSITY FROM THE INPUT VARIABLES
IF(LAM .LE. 0) THEN
DO I=1,NKODE
VISTURB(I) = EKIN(I)**2/E(I)
END DO
ELSE
DO I=1,NKODE
EKIN(I) = 0.
END DO
END IF
DO I=1,NKODE
M=I+2
CALL GNM(I,M,3)=1
END DO

C ASSIGN THE GAUSS LEGENDRE WEIGHTS
C AND THE LOCAL COORDINATES
H(1)= 2369258680
H(2)= 4762628704
H(3)= 5688000000
H(4)= (2)
H(5)= (1)
DO J=1,5

CALCULATE SHAPE FUNCTIONS
SF5(1,J,K) = - . 25*(1.·XI)* (1.· ETA)* (XI+ETA+1.)
SF5(2,J,K) = . 5*(1.·XI)* (1.· ETA)
SF5(3,J,K) = - .25*(1.+XI)* (1.· ETA)* (XI· ETA+1.)
SF5(4,J,K) = . 5*(1.· ETA)* (1.·XI)
SF5(5,J,K) = .25*(1.+XI)* (1.· ETA)* (XI+ETA+1.)
SF5(6,J,K) = .25*(1.·XI)* (1.· ETA)
SF5(7,J,K) = .25*(XI+1.)* (1.· ETA)* (XI+ETA+1.)
SF5(8,J,K) = .5*(1.· ETA)* (1.· XI)

SFL(1,J,K) = .25*(1.·XI)* (1.· ETA)
SFL(2,J,K) = .25*(1.+XI)* (1.· ETA)
SFL(3,J,K) = .25*(1.+XI)* (1.· ETA)
SFL(4,J,K) = .25*(1.·XI)* (1.· ETA)

CALCULATE THE DERIVATIVES OF THE SHAPE FUNCTION WITH RESPECT
C TO THE LOCAL COORDINATES
DNDC5(1,J,K) = .25* (1.· ETA) *(2.· XI· ETA)
DNDC5(2,J,K) = (ETA·1.)* XI
DNDC5(3,J,K) = - .25* (1.· ETA) *(2.· XI· ETA)
DNDC5(4,J,K) = (ETA·1.)* XI
DNDC5(5,J,K) = .25*(1.· ETA) *(1.· ETA)
DNDC5(6,J,K) = (1.· ETA)* XI
DNDC5(7,J,K) = .25*(1.· ETA) *(2.· XI· ETA)
DNDC5(8,J,K) = .5*(ETA· ETA)· XI
DO NDAS(1,J,K)=.25*(XI)**2*(XI+2.*ETA)
NDAS(2,J,K)=.5*(XI**2-1.)
NDAS(3,J,K)=.25*(1.+XI)*(2.*ETA*XI)
NDAS(4,J,K)=-1.*ETA*XI
END DO
END DO

THE CYCLE OF ITERATIONS STARTS HERE
FIRST, ASSUME THE VISCOSITY AND CALCULATE THE VELOCITY
AND PRESSURE. SECOND, FROM V AND P CALCULATE THE K AND E VARIABLES TO DETERMINE TURBULENT VISCOSITY

NCTBI3=NCTBI3+1
NCTBIG4=0
IF(LAM.LT.0) THEN
NCTBI3=3
END IF
3 CONTINUE
CALCULATE THE NUMBER OF UNKNOWNS
WRITE(4,*)
NCTV=0
DO I=1,NNODE
DO J=1,3
IF(DUIT(I,J).LT.0) THEN
NCTV=NCTV+1
END IF
END DO
WRITE(4,*)
WRITE(4,*)' THE NUMBER OF UNKNOWNS IS '
WRITE(4,*')NCTV
NCTBI3=NCTBI3+1
NCTBIG4=0

THE FIRST CYCLE OF THE NEWTON METHOD RETURNS UPDATED VALUES OF U1, U2, AND U3 HERE
C EQUAL TO ZERO
DO J=1,5
DO K=1,5
EWE(J,K)=0.0
EWEZ(J,K)=0.0
DUDX(K,J)=0.0
DUDXZ(J,K)=0.0
DUDXZ(J,K)=0.0
PEE(J,K)=0.0
VISTURB(J,K)=0.0
ERK(J,K)=0.0
DHDX(J,K)=0.0
DHDX2(J,K)=0.0
END DO
END DO

C CALCULATE EWE1,EWE2,DUDX,DUDX2,DUDX3
DO J=1,5
DO K=1,5
DO M=1,8
EWE(J,K)=EWE(J,K)+S7S(M,J,K)*STURN(J,M)
EWEZ(J,K)=EWEZ(J,K)+S7S(M,J,K)*STURNZ(J,M)
DUDX(K,J)=DUDX(K,J)+S7S(M,J,K)*STURNX(J,M)
DUDXZ(J,K)=DUDXZ(J,K)+S7S(M,J,K)*STURNXZ(J,M)
DUDXZ(J,K)=DUDXZ(J,K)+S7S(M,J,K)*STURNXZ(J,M)
VISTURB(J,K)=VISTURB(J,K)+S7S(M,J,K)*STURNURB(J,M)
ERK(J,K)=ERK(J,K)+S7S(M,J,K)*STURNK(J,M)
DHDX(J,K)=DHDX(J,K)+S7S(M,J,K)*STURNHDX(J,M)
DHDX2(J,K)=DHDX2(J,K)+S7S(M,J,K)*STURNHDX2(J,M)
END DO
END DO

C CALCULATE THE VALUE OF THE CONTRIBUTION TO THE "F + M" FROM
THE CONVECTION, DISSIPATION, AND PRESSURE AREA INTEGRALS
DO J=1,8
DO K=1,5
DO L=1,8
FTEMP(J,M,K)=FTEMP(J,M,K)+FVAR(J,L)*STURN(J,L)
END DO
END DO

C SUM UP THE ELEMENT CONTRIBUTION TO F1, F2, AND F3
DO J=1,8
DO L=1,8
IF (VAR(J,L))
F1(J,M,N)=F1(J,M,N)+FTEMP(J,M,N)
F2(J,M,N)=F2(J,M,N)+FTEMP(J,M,N)
END DO

* CALCULATE THE VALUE OF THE DERIVATIVES
* L1 IS THE NODE AT WHICH THE VARIABLE IS CHANGED
* L2 IS THE NODE AT WHICH THE F IS CHANGED
* JAC IS THE DERIVATIVE OF F(I) WITH RESPECT TO
* THE VARIABLE (J)
* IT IS STORED IN THE ONE DIMENSIONAL ARRAY "A"

C IF VARIABLE "U" AT ELEMENT J, NODE L1 IS NOT KNOWN
C CHANGE UT THERE WHILE RETAINING THE UNCHANGED VALUE
IF (VAR(J,L1))
RR=RR+0.1
END IF

C IF VARIABLE AT ELEMENT J, NODE L1 IS KNOWN
C CHANGE UT THERE WHILE RETAINING THE UNCHANGED VALUE
IF (VAR(J,L1))
RR=RR+0.1
END IF
CALL DF1(I,F10,F20,F30,EWE1,EWE2,P1,DU0X1,DU0X2,DU0X3,DU0X2)
END IF  
IF(DUIT(GNN(1,L1),2),3), NE. 1) THEN  
II=NUNK(GNN(1,L1),3)  
NUM=II*(II+1)*1500  
A(KM)=A(MM)+F30*(GNN(1,L2)-FTEMP(GNN(1,L2)))/DELP  
END IF  
END IF  
CONTINUE  
C  
RESTORE U2  
U2(GNN(1,L1))=U2HOLD  
DO J=1,5  
DO K=1,5  
U2(J,K)=HOLD2(J,K)  
END DO  
C  
END IF  
C  
IF(MISS P AT NODE GNN(1,L1) IS KNOWN &  
C  
SAVE THE OLD VALUE OF P  
IF(DUIT(GNN(1,L1),3), .EQ. 0) THEN  
JH=NUNK(GNN(1,L1),3)  
P(0)HOLD=P(GNN(1,L1))  
DO J=1,5  
DO K=1,5  
P(HOL2(J,K))=PEE(J,K)  
END DO  
END IF  
C CHANGE P  
P(GNN(1,L1))=P(GNN(1,L1))+DELP  
DO J=1,5  
DO K=1,5  
DO M=1,4  
MM=2*M-1  
PEE(J,K)+PEE(J,K)+SFM(M,J,K)*P(GNN(1,M))  
END DO  
END DO  
CALL DIF(1,FT3,F20,F30,EWE1,EWE2,PEE,DUXD2,DUXD2)  
C  
EVALUATE THE DERIVATIVES OF F1,F2, AND F3 AT L2 WITH RESPECT TO
THIS CHANGE IS FOR THE SLUICE GATE AREA OF THE SLUICE GATE PROBLEM

IF(FACE(II).EQ.4)
  XL2=PM(II)*X2(GNN(IA,BCNOD(1))); X2(GNN(IA,BCNOD(3)));
  BCC(1)=2*X2(GNN(IA,BCNOD(1)))*XL2/6.
  BCC(2)=2*X2(GNN(IA,BCNOD(2)))*XL2/6.
  BCC(3)=2*X2(GNN(IA,BCNOD(3)))*XL2/6.
END IF

END IF DON=1,3 ILCNOD ( N)

XL2=PM(II)*X2(GNN(IA,BCNOD(1)));
BC(1)=T2(GNN(IA,BCNOD(1)))*XL2/6.
BC(2)=4*T2(GNN(IA,BCNOD(2)))*XL2/6.
BC(3)=T2(GNN(IA,BCNOD(3)))*XL2/6.

IF(DIR(II).EQ.1)
  IC=GNN(IA,IB) IF(DIR(II).EQ.1)
  F1( IC)=F1( IC).
END IF
IF(DIR(II).EQ.2)
  F2( IC)=F2( IC).
END IF
END DO
END DO

END IF
C THIS END IF IS FOR GRADIENT BOUNDARY CONDITIONS

C SET THE F VECTOR (VALUE OF THE FUNCTION AT THE PRESENT VALUE OF
C THE VARIABLES

NCT4=0
DO I=1,NNODE
  IF(DUIT(I,1).EQ.0)
    NCT4=NCT4+1
    FF(NCT4)=F1(I)
  END IF
  IF(DUIT(I,2).EQ.0)
    NCT4=NCT4+1
    FF(NCT4)=F2(I)
  END IF
  IF(DUIT(I,3).EQ.0)
    NCT4=NCT4+1
    FF(NCT4)=F3(I)
  END IF
END DO

CALL SOLVE(NCTBIG1,NCT4,A,FF,ICN,IRN,IVECT,JEVT,LEND)

IF(NORM LoT).
  WRITE(4,'(A)') 'THE SOLUTION IS '
  GO TO 9997
END IF
IF(NCTBIG1 .GE. MAX1) THEN
  WRITE(4,'(A)') 'THE LAST VALUE OF THE VARIABLES ARE '
  END IF
  WRITE(4,'(A)')
CONTINUE
DO I=1,NNODE
  WRITE(4,'(I,1X,F10.8)') I, Ul(I)
  WRITE(4,'(I,1X,F10.8)') I, U2(I)
  WRITE(4,'(I,1X,F10.8)') I, P(I)
END DO
IF(LAH .GT. 0) GO TO 9999
IF(LAH .GT. 0) GO TO 9999

ELSEIF(LT. MAX1 .AND. ANORM .GT. TOL)
  GO TO 1
END IF
RESET THE VALUE OF A, IRN, IVECT, ICN, AND JVECT TO ZERO

C THIS END IF IS FOR GRADIENT BOUNDARY CONDITIONS

C SET THE F VECTOR (VALUE OF THE FUNCTION AT THE PRESENT VALUE OF
C THE VARIABLES

NCT4=0
DO I=1,NNODE
  IF(DUIT(I,1).EQ.0)
    NCT4=NCT4+1
    FF(NCT4)=F1(I)
  END IF
  IF(DUIT(I,2).EQ.0)
    NCT4=NCT4+1
    FF(NCT4)=F2(I)
  END IF
  IF(DUIT(I,3).EQ.0)
    NCT4=NCT4+1
    FF(NCT4)=F3(I)
  END IF
END DO

CALL SOLVE(NCTBIG1,NCT4,A,FF,ICN,IRN,IVECT,JEVT,LEND)

IF(NORM LoT).
  WRITE(4,'(A)') 'THE SOLUTION IS '
  GO TO 9997
END IF
IF(NCTBIG1 .GE. MAX1) THEN
  WRITE(4,'(A)') 'THE LAST VALUE OF THE VARIABLES ARE '
  END IF
  WRITE(4,'(A)')
CONTINUE
DO I=1,NNODE
  WRITE(4,'(I,1X,F10.8)') I, Ul(I)
  WRITE(4,'(I,1X,F10.8)') I, U2(I)
  WRITE(4,'(I,1X,F10.8)') I, P(I)
END DO
IF(LAH .GT. 0) GO TO 9999
IF(LAH .GT. 0) GO TO 9999

ELSEIF(LT. MAX1 .AND. ANORM .GT. TOL)
  GO TO 1
END IF
RESET THE VALUE OF A, IRN, IVECT, ICN, AND JVECT TO ZERO
DO II=1,4*NZZ
ICN(II) = 0
END

DO II=1,NZZ+350000
RN(II) = 0
END

DO II=1,NZZ
I Vect(II) = 0
J Vect(II) = 0
END

I I=1,NCT4
!START(II) = 0
I ASTART(II) = 0
FF(II) = 0.
!END(II) = 0
IAEND(II) = 0
END

II=1,NNOOE
U1DI F(II) = 0.
UZDI F(II) = 0.
POI F(II) = 0.
END

...........................................................................

C SET KNOWN K AND E BOUNDARY CONDITIONS FROM PREVIOUS VELOCITIES
...........................................................................

C
C THE SECOND CYCLE OF THE
C NEUTON METHOD RETURNS UPDATED VALUES OF K, E AND V1 HERE
C

4 CONTINUE
NCTBIG2 = 0

IF(ABS(FBC1/DIBC) .GT. TOLBC) THEN
NCT7 = NCT7 + 1
IF(NCT7 .LT. 20) THEN
WRITE(4,*') USTAR DID NOT CONVERGE AT NODE I
WRITE(4,*') USTAR
GO TO 377
END IF
END IF

CONTINUE
IF(NCT7 .GT. 0) THEN
WRITE(4,*') USTAR, EKIN(I), E(I)
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') USTAR, K, E ARE OUT OF RANGE
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF NON-CONVERGED BCS (K-E) IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
IF(NCT6 .GT. 0) THEN
WRITE(4,*') NUMBER OF OUT OF RANGE USTARS IS'
WRITE(4,*') NCT6
END IF

CONTINUE
C THIS END IF FOR NBC2T .GT. 0
...
CALCULATE THE VALUE OF THE DERIVATIVES

L1 IS THE NODE AT WHICH THE VARIABLE IS CHANGED
L2 IS THE NODE AT WHICH THE F IS CHANGED
JAC IS THE DERIVATIVE OF F(I) WITH RESPECT TO THE VARIABLE (J)
IT IS STORED IN THE ARRAY "A"

DO 1500 L1=1,8
IF VARIABLE K AT ELEMENT I, NODE L1 IS NOT KNOWN CHANGE K THERE WHILE RETAINING THE UNCHANGED VALUE
IF(DUIT(GNN(I,L1),5).EQ.0) THEN
J=UNK(GNN(I,L1),5)
EKKHOLD=EK(GNN(I,L1))
DO J=1,5
DO K=1,5
HOLD(J,K)=KAY(J,K)
HOLD(J,K)=HOLD2(J,K)
HOLD2(J,K)=HOLD3(J,K)
KAY(J,K)=0.0
D0X(J,K)=0.0
DOX2(J,K)=0.0
END DO
END IF
IF(DUIT(GNN(I,L1),6).EQ.0) THEN
J=UNK(GNN(I,L1),6)
EKKHOLD=EK(GNN(I,L1))
DO J=1,5
END DO
END IF

END IF CHECK IF VARIABLE E AT NODE GNN(I,L1) IS A KNOWN
IF(DUIT(GNN(I,L1),6).EQ.0) THEN
J=UNK(GNN(I,L1),6)
EKKHOLD=EK(GNN(I,L1))
DO J=1,5
ENDIF
DO K=1,5
  HOLD1(J,K)=EE(J,K)
  HOLD2(J,K)=DEDX1(J,K)
  HOLD3(J,K)=DEDX2(J,K)
  EE(J,K)=0.0
  DEDX1(J,K)=0.0
  DEDX2(J,K)=0.0
END DO
END DO

CHANGE E AND CALCULATE THE NEW VALUE OF EE,DEDX1,DEDX2

DO J=1,5
  DO M=1,8
  EE(J,K)=EE(J,K)+S5(M,J,K)*E(GNN(I,M))
  DEDX1(J,K)=DEDX1(J,K)+S5(M,J,K)*E(GNN(I,M))
  DEDX2(J,K)=DEDX2(J,K)+S5(M,J,K)*E(GNN(I,M))
END DO
END DO
END DO

C CALL DF2(I,F5,F6,EWE1,EWE2,KAY,EE,DEDX1,DEDX2,DEOX1,DEOX2,DEOX3,CONSTA,SFS,
  C1,G2,G3,EZ1,EZ2,GAN)

C CALL SOLVE(NCT4,NCT5,F5,F6,ICN,IRM,JVECT,JEND,JENO,JIRONG,ANORM,BNORM,NNZ)

C EVALUATE THE DERIVATIVES OF F5 & F6 AT L2 WITH RESPECT TO E AT NODE L1

DO 1460 L2=1,8
  IF(DUIT(GNN(I,L2),5).EQ.0) THEN
    NCT4=NCT4+1
    IF(DUIT(GNN(I,L2),6).EQ.0) THEN
      NCT5=NCT5+1
      IF(JIRONG(JIIRONG).EQ.0) THEN
        WRITE(4,*)JIIRONG
        GO TO 9997
      END IF
    END IF
  END IF
1460 CONTINUE

C RESTORE E

E(GNN(I,L1))=E1HOL

WRITE(6,*)J
GO TO 9997
END IF

C END IF FOR TESTING IF VARIABLE E (6) AT L 1 IS NOT A "KNOWN"

CONTINUE
C THIS 'END DO' FOR 'I' (THE ELEMENT COUNTER)

NCT4=0
DO I=1,NNODE
  IF(DUIT(I,5).EQ.0) THEN
    NCT4=NCT4+1
    IF(DUIT(I,6).EQ.0) THEN
      NCT5=NCT5+1
    END IF
  END IF
END DO

C CALL SOLVE(NCT102,NCT4,F5,F6,ICN,IRM,JVECT,JEND,JENO,JIRONG,ANORM,BNORM,NNZ)

C IF(JIRONG .NE. 0) THEN
  WRITE(6,*)JIRONG
  GO TO 9997
END IF

C END IF FOR TESTING IF VARIABLE E (6) AT L 1 IS NOT A "KNOWN"
SUBROUTINE DERIV(I,JX1,JX2,GNN,DNDX,DNOA,HH)
6 DNDX,DNOA,CONST
C THIS SUBROUTINE CALCULATES THE JACOBIAN MATRIX OF THE
C COORDINATE TRANSFORMATION OF EACH ELEMENT
C AND CALCULATES THE VALUE OF THE DERIVATIVE OF X1 AND X2
C WITH RESPECT TO XI AND ETA (LOCAL COORDINATES) AT EACH
C INTEGRATION POINT AS WELL AS THE DERIVATIVE OF THE
C SHAPE FUNCTIONS WITH RESPECT TO THE X1 AND X2 COORDINATES

INTEGER*2 GNN(200,8) REAL*4 DNDX0(3,5,5),DNOA0(3,5,5)
6 DNOA5(8,5,5),X1(700),X2(700),OETA, J11(5,5), J12(5,5), J21(5,5)
6 J22(5,5),DET(5,5),CONSTA(5,5),HH(5,5)
6 DNDX1(8,5,5),DNDX2(8,5,5) DO I=1,5
DO K=1,5
DO M=1,2
DNDX0(M,J,K)=0.0
DNOA0(M,J,K)=0.0
END DO END DO END DO
DO J=1,5
DO K=1,5
DO M=1,5
DNOA0(J,K,M)=0.0
DNDX0(J,K,M)=0.0
END DO END DO END DO
DO J=1,5
END
DO I=1,5
DO K=1,5
DO M=1,5
DNDX0(I,J,K)=0.0
DNOA0(I,J,K)=0.0
END DO END DO END DO
END

C CALCULATE THE INVERSE OF THE JACOBIAN AS ADJOINT(J)/DET(J)
C J(22) -J(12) / /
C -J(21) J(11) / /
C (P, 51 OF MATRICES)
SUBROUTINE SOLVE(NCTBIG,NCT4,A, FF, ICN, IRN, IVECT, JVECT, IENO, IAST, IAENO, ISTART, IAEND, UVAR, IIIRONG, ANORM, BNORM, NZZ)

THIS ACCOMODATES THE HARWELL SPARSE MATRIX ROUTINE

C FIND THE FIRST AND LAST ENTRY ON ROW I THAT IS NON-ZERO
C FOR COMPRESSION OF ZEROS TO MAKE THE JACOBIAN ONE DIMENSIONAL
C & FOR LATER COMPARISON TO DETERMINE IF THE SUCCEEDING MATRICES
C HAVE THE SAME SPARITY

INTEGER*2 ISTART(1300), IASTART(1300), IEN(1300), IAEND(1300),
6 IINC(250000), IEN(1750000), IKEEP(1300,5), JVEC(250000),
6 JVECT(250000),
6 NEN, MCT810, MCT4, IIIRONG

INTEGER N4(1300,5), N2Z, N4W, BW, BW2, N2Z4, MCT8
REAL*4 FFN(1300), BNMORM, ANORM, A(1750000), W1300, AHOLD

WRITE(4,*)' NCTBIG,MCT4,UVAR,IIIRONG, AND N2Z AT THE CALL ARE'
WRITE(4,*)MCT810,MCT4,UVAR,IIIRONG,N2Z

SUBROUTINE SOLVE IS CALLED
CALL CPU
WRITE(4,*)
MITYPE=1
ANORM=0.
NEWA=0

C WRITE AND SAVE THE JACOBIAN
IF(NCTBIG .GT. 1) THEN
N2Z=0
DO I=1,NCT4
DO J=1,NCT4
N4=I+J-1
IF(N4 .LT. 1) THEN
GO TO 900
END IF
END DO
CONTINUE
J=1,NCT4
IF(A(N2Z) .NE. 0) THEN
GO TO 920
END IF
CONTINUE
END DO
CONTINUE
END
C
C DO I=1,NCT4
C IF (N4 .LT. 1) THEN
C N4=I+J-1
C IF(A(N2Z) .NE. 0) THEN
C GO TO 920
C END IF
C CONTINUE
C END DO
C IF(NCTBIG .GT. 1) THEN
C N2Z=N2Z+1
C IF(N2Z .GT. 557) THEN
C WRITE(4,*) N2Z TO SS7
C DO I=1,NCT4
C IF(N4 .LT. 1) THEN
C N4=I+J-1
C IF(A(N2Z) .NE. 0) THEN
C GO TO 920
C END IF
C CONTINUE
C END DO
C IF(NCTBIG .GT. 1) THEN
C N2Z=N2Z+1
C IF(N2Z .GT. 557) THEN
C WRITE(4,*) N2Z TO SS7
C END DO
C END DO
C 100 CONTINUE
C
SET "A" MATRIX AND ROW AND COLUMN INDICES

IF (A(NUM).NE.0) THEN
   ISTART(I)=J
   GO TO 930
END IF
END DO

CONTINUE

DO J=NCT4,1,-1 NUM=J+(J·1)*1300 IF(A(NUM).NE.0) THEN ENDDO CONTINUE

END

DO K=NZZ,1,1750000 A(K)=0 END DO

CALL THE HARRIET "SPARSE MATRIX SOLVER"

THESE CALLS REPLACE THE FF(VALUE OF FUNCTION) WITH A NEW
IF WHICH IS THE CORRECTIONS TO BE MADE TO THE VARIABLES

END

WRITE(*,100)BWA

FORMAT(*,10K,'MAX ACTUAL BANDWIDTH IS',7X,110,/)
IF(IFLAG .NE. 0)THEN
WRITE(*,*) 'ERROR'
END IF
IF(IFLAG .LT. 0)THEN
WRITE(*,*) IFLAG IS
WRITE(*,*) MINCN, MINIRN ARE
WRITE(*,*) MINCN, MINIRN
SWONG=1
GO TO 49
END IF
IF(NCTB < 1 .AND. NEIIA .EQ. 0)THEN
WRITE(*,*) SUBROUTINE MA28B IS CALLED'
CALL CPU
WRITE(*,*)
CALL MA28B(NCT4,NZ,A,ICN,IVECT,JVECT,ICN,KEEP,IV,IFLAG)
END IF
IF(IFLAG .NE. 0)THEN
WRITE(*,*) 'ERROR'
END IF
IF(IFLAG .LT. 0)THEN
WRITE(*,*) IFLAG IS
WRITE(*,*) MINCN, MINIRN ARE
WRITE(*,*) MINCN, MINIRN
SWONG=1
GO TO 49
END IF
CALL MA28C(NCT4,A,ICN,KEEP,IV,JVECT,JVECT,ICN,IV,IFLAG)
IF(IFLAG .NE. 0)THEN
WRITE(*,*) 'ERROR'
END IF
IF(IFLAG .LT. 0)THEN
WRITE(*,*) IFLAG IS
WRITE(*,*) MINCN, MINIRN ARE
WRITE(*,*) MINCN, MINIRN
SWONG=1
GO TO 49
END IF
CALL CPU
WRITE(*,*) SUBROUTINE MA28C IS CALLED'
CALL CPU
WRITE(*,*)
CALL MA28C(NCT4,A,ICN,KEEP,IV,JVECT,JVECT,ICN,IV,IFLAG)
WRITE(*,*) 'SUBROUTINE MA28C IS DONE'
CALL CPU
WRITE(*,*)
CALL MA28C(NCT4,A,ICN,KEEP,IV,JVECT,JVECT,ICN,IV,IFLAG)
WRITE(*,*) 'SUBROUTINE MA28C IS DONE'
Integer Grp,
Integer Mem,
Integer Sys$GetJpi, Sys$Stat
Integer List(0:2,1:6), '02030000', '000', '04070004', 'X', 0, 0,
 '02030000', 'X', 0, 0, '04070004', 'X', 0, 0,
 List(1,1) = %Loc(Grp)
 List(1,2) = %Loc(CPUTime)
 List(1,3) = %Loc(UserName)
 List(1,4) = %Loc(Account)
 List(1,5) = %Loc(Mem)
 Sys$Stat = Sys$GetJpi(,,,,List,)
 Time = FLOAT(CPUTime)/100
 WRITE(A,*),TIME
 Return
End
Appendix B

Data Files: FEM.DAT, GRID.DAT,

INITIAL.DAT, and BC.DAT
Job FEM (1514) queued to VAXA_TXAO on 26-NOV-1986 09:09 by user UF7105, UIC (UF7105), under account 000744 at priority 100, started on printer _VAXASTXAO on 26-NOV-1986 09:10 from queue VAXA_TXAO.
Job GRID (1515) queued to VAXA_TXAO on 26-NOV-1986 09:09 by user UF7105, UIC (UF7105), under account 003744 at priority 100, started on printer _VAXASTXAO: on 26-NOV-1986 09:11 from queue VAXA_TXAO.
<table>
<thead>
<tr>
<th>USERSOISK</th>
<th>DAT;56</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>2 415</td>
</tr>
<tr>
<td>129</td>
<td>3 435</td>
</tr>
<tr>
<td>129</td>
<td>4 434</td>
</tr>
<tr>
<td>129</td>
<td>5 433</td>
</tr>
<tr>
<td>129</td>
<td>6 414</td>
</tr>
<tr>
<td>129</td>
<td>7 401</td>
</tr>
<tr>
<td>129</td>
<td>8 402</td>
</tr>
<tr>
<td>130</td>
<td>1 405</td>
</tr>
<tr>
<td>130</td>
<td>2 416</td>
</tr>
<tr>
<td>130</td>
<td>3 437</td>
</tr>
<tr>
<td>130</td>
<td>4 436</td>
</tr>
<tr>
<td>130</td>
<td>5 435</td>
</tr>
<tr>
<td>130</td>
<td>6 415</td>
</tr>
<tr>
<td>130</td>
<td>7 403</td>
</tr>
<tr>
<td>130</td>
<td>8 404</td>
</tr>
<tr>
<td>131</td>
<td>1 419</td>
</tr>
<tr>
<td>131</td>
<td>2 439</td>
</tr>
<tr>
<td>131</td>
<td>3 451</td>
</tr>
<tr>
<td>131</td>
<td>4 450</td>
</tr>
<tr>
<td>131</td>
<td>5 449</td>
</tr>
<tr>
<td>131</td>
<td>6 438</td>
</tr>
<tr>
<td>131</td>
<td>7 417</td>
</tr>
<tr>
<td>131</td>
<td>8 418</td>
</tr>
<tr>
<td>132</td>
<td>1 421</td>
</tr>
<tr>
<td>132</td>
<td>2 440</td>
</tr>
<tr>
<td>132</td>
<td>3 453</td>
</tr>
<tr>
<td>132</td>
<td>4 452</td>
</tr>
<tr>
<td>132</td>
<td>5 451</td>
</tr>
<tr>
<td>132</td>
<td>6 439</td>
</tr>
<tr>
<td>132</td>
<td>7 419</td>
</tr>
<tr>
<td>132</td>
<td>8 420</td>
</tr>
<tr>
<td>133</td>
<td>1 423</td>
</tr>
<tr>
<td>133</td>
<td>2 441</td>
</tr>
<tr>
<td>133</td>
<td>3 455</td>
</tr>
<tr>
<td>133</td>
<td>4 454</td>
</tr>
<tr>
<td>133</td>
<td>5 453</td>
</tr>
<tr>
<td>133</td>
<td>6 440</td>
</tr>
<tr>
<td>133</td>
<td>7 421</td>
</tr>
<tr>
<td>133</td>
<td>8 422</td>
</tr>
<tr>
<td>134</td>
<td>1 425</td>
</tr>
<tr>
<td>134</td>
<td>2 442</td>
</tr>
<tr>
<td>134</td>
<td>3 456</td>
</tr>
<tr>
<td>134</td>
<td>4 456</td>
</tr>
<tr>
<td>134</td>
<td>5 455</td>
</tr>
<tr>
<td>134</td>
<td>6 441</td>
</tr>
<tr>
<td>135</td>
<td>7 423</td>
</tr>
<tr>
<td>135</td>
<td>8 424</td>
</tr>
<tr>
<td>135</td>
<td>9 427</td>
</tr>
<tr>
<td>135</td>
<td>10 443</td>
</tr>
<tr>
<td>135</td>
<td>11 459</td>
</tr>
<tr>
<td>135</td>
<td>12 458</td>
</tr>
<tr>
<td>135</td>
<td>13 457</td>
</tr>
<tr>
<td>135</td>
<td>14 442</td>
</tr>
<tr>
<td>135</td>
<td>15 425</td>
</tr>
<tr>
<td>135</td>
<td>16 426</td>
</tr>
<tr>
<td>136</td>
<td>1 429</td>
</tr>
<tr>
<td>136</td>
<td>2 444</td>
</tr>
</tbody>
</table>

26 NOV 1986 09:09

Page 10
Job INITIAL (1517) queued to VAXA_TXA0 on 26-NOV-1986 09:10 by user UF7105, UIC (UF7105), under account 000744 at priority 100, started on printer _VAXA$TXA0: on 26-NOV-1986 09:14 from queue VAXA_TXA0.
<table>
<thead>
<tr>
<th>Number</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
<th>Value 7</th>
<th>Value 8</th>
<th>Value 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>456</td>
<td>6.241639144</td>
<td>0.20292133</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>457</td>
<td>6.33865372</td>
<td>0.07257570</td>
<td>0.60375265</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>458</td>
<td>6.47199106</td>
<td>-0.09356218</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>459</td>
<td>6.55225150</td>
<td>-0.24304001</td>
<td>0.17241596</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>460</td>
<td>6.57185361</td>
<td>-0.38847031</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>461</td>
<td>6.5657590</td>
<td>-0.30939065</td>
<td>-1.87698564</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>462</td>
<td>6.59771575</td>
<td>-0.22043803</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>463</td>
<td>6.5944726</td>
<td>-0.12175006</td>
<td>-2.52095222</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>464</td>
<td>6.55399010</td>
<td>0.04728621</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>465</td>
<td>6.53498786</td>
<td>0.00000000</td>
<td>-2.50367944</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>466</td>
<td>6.55883312</td>
<td>0.05082873</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>467</td>
<td>6.56420344</td>
<td>-0.14026146</td>
<td>-3.0168128</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>468</td>
<td>6.56742434</td>
<td>-0.12576058</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>469</td>
<td>6.56990147</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.02200000</td>
<td>0.06300000</td>
<td>0.01200000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Job BC (1513) queued to VAXA_TXAO on 26-NOV-1986 09:08 by user UF7105, UIC [UF7105], under account 000744 at priority 100, started on printer VAXA$TXAO: on 26-NOV-1986 09:10 from queue VAXA_TXAO.
<table>
<thead>
<tr>
<th>USERDISK: [UF7105]BC.DAT</th>
<th>26-NOV-1986 09:08</th>
<th>Page 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.000000E-02</td>
<td>1</td>
<td>0.0002</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>2.22990</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>2.21416</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>1</td>
<td>2.19702</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>2.17150</td>
</tr>
<tr>
<td>65</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
<td>2.14260</td>
</tr>
<tr>
<td>86</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>96</td>
<td>1</td>
<td>2.10870</td>
</tr>
<tr>
<td>97</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>117</td>
<td>1</td>
<td>0.06810</td>
</tr>
<tr>
<td>118</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>128</td>
<td>1</td>
<td>2.05115</td>
</tr>
<tr>
<td>129</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>169</td>
<td>1</td>
<td>1.99265</td>
</tr>
<tr>
<td>160</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>161</td>
<td>1</td>
<td>1.92760</td>
</tr>
<tr>
<td>181</td>
<td>1</td>
<td>0.84548</td>
</tr>
<tr>
<td>192</td>
<td>1</td>
<td>1.73020</td>
</tr>
<tr>
<td>213</td>
<td>1</td>
<td>1.43780</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>85</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>96</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>117</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>128</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>169</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>160</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>181</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>192</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

| 213 | 2 | 0 |
| 224 | 7 | 2.191299E-02 |
| 245 | 7 | 2.191299E-02 |
| 256 | 7 | 2.191299E-02 |
| 277 | 7 | 2.191299E-02 |
| 288 | 7 | 2.191299E-02 |
| 309 | 7 | 2.191299E-02 |
| 320 | 7 | 2.191299E-02 |
| 341 | 7 | 2.191299E-02 |
| 352 | 7 | 2.191299E-02 |
Appendix C

Data Generation Programs:

NODES4.FOR

SLUINIT.FOR

SLUBC.FOR
Job NODE54 (1519) queued to VAXA_TXAO on 26·NOV·1986 09:14 by user UF7105, UIC (UF7105), under account 000744 at priority 100, started on printer _VAXA_TXAO_ on 26·NOV·1986 09:16 from queue VAXA_TXAO.
INTEGER E(2000,8), DU1
REAL (1000,4), XI(1000), X2(1000), XI(1000), X2(1000), X1A(1000),
X2A(1000), X2B(1000), HT(1000)
OPEN (UNIT=3, NAME='MODELS.DAT', STATUS='OLD')
OPEN (UNIT=4, NAME='GRID.DAT', STATUS='NEW')
READ (3, *) DU1
READ (3, *) NROW, NCOL, NW, NNODES, NEL
READ (3, *) VIS, RHO, NBCK, NBCG
READ (3, *, END=998) (X1A(J), X1B(J), X2B(J), J=1, NCOL+1)
CONTINUE
WRITE (4, *) NEL, NNODES, VIS, RHO, NBCK, NBCG
100 FORMAT (/I8, 2X, F10.8, 2X, F10.8, 2X, F10.8, 2X, 2X)

DO K=1, NCOL
 B=K-1*NWC
 A=(K-2)*NROW+1
 C(B+1,1)=.3
 C(B+1,2)=.5
 C(B+1,3)=.1
 C(B+1,4)=A+NROII+.7
 C(B+2,1)=A+.4
 C(B+2,2)=A+NROII+.8
 END DO

DO I=1, NROW
 C(B+2*I-1,1)=A+1-2*3
 C(B+2*I-1,2)=A+1+1.5
 C(B+2*I-1,3)=A+1-2*NROW+.1
 C(B+2*I-1,4)=A+1-2*NROW+.7
 C(B+2*I-1,5)=A+1+.6
 C(B+2*I-2)=A+1+NROW+.8
 END DO

COMMENCE THE "INTERIOR" NODES

DO I=2, NROW
 C(B+2*I*NROW+1,1)=A+NROW+1.3
 C(B+2*I*NROW+1,2)=.5
 C(B+2*I*NROW+1,3)=A+NROW+2+.1
 C(B+2*I*NROW+1,4)=.7

C(B+2*I*NROW+2,1)=.2
 C(B+2*I*NROW+2,2)=A+NROW+.6
 END DO

134
COMENCE THE LAST COLUMN

B=NCOL*NC
A=(NERR,1)*NROW+1
C(B+1,1)=.3
C(B+1,2)=A+.5
C(B+1,3)=.1
C(B+1,4)=.7
C(B+2,1)=A+.4
C(B+2,2)=.8
DO I=2,NROW
C(B+2*I-1,1)=A+.3
C(B+2*I-1,2)=A+.5
C(B+2*I-1,3)=.1
C(B+2*I-1,4)=.7
C(B+2*I,1)=A+.4
C(B+2*I,2)=.8
END DO
C(B+2*NROW+1,1)=A+NROW+.3
C(B+2*NROW+1,2)=.5
C(B+2*NROW+1,3)=.1
C(B+2*NROW+1,4)=.7
DO I=1,NNODES
DO J=1,4
IF(C(I,J).LE.1C(I,J)=0.0
WRITE(*,'(5X,E16.15)')
END DO
END DO
DO K=1,NEL
DO L=1,8
DO I=1,NNC
DO J=1,4
IF(C(J,J).EQ.D)THEN
C(I,J)=I
WRITE(*,'(5X,E16.15)')
END IF
END DO
END DO
END DO
NUTS=NROW*2+1
DO I=1,NCOL+1
DIST=XTA(I)*X18(I)
XT(I)*NNC+1)=XTA(I)
XT(I)*NNC+2)=XTA(I)*DIST*.005
XT(I)*NNC+3)=XTA(I)*DIST*.010
XT(I)*NNC+4)=XTA(I)*DIST*.015
XT(I)*NNC+5)=XTA(I)*DIST*.025
XT(I)*NNC+6)=XTA(I)*DIST*.050
XT(I)*NNC+7)=XTA(I)*DIST*.060
XT(I)*NNC+8)=XTA(I)*DIST*.075
XT(I)*NNC+9)=XTA(I)*DIST*.085
XT(I)*NNC+10)=XTA(I)*DIST*.095
END DO
X1((1)*NNC+9)*XTA(1) DIST=0.080
X1((1)*NNC+10)*XTA(1) DIST=0.115
X1((1)*NNC+11)*XTA(1) DIST=0.150
X1((1)*NNC+12)*XTA(1) DIST=0.250
X1((1)*NNC+13)*XTA(1) DIST=0.350
X1((1)*NNC+14)*XTA(1) DIST=0.475
X1((1)*NNC+15)*XTA(1) DIST=0.600
X1((1)*NNC+16)*XTA(1) DIST=0.700
X1((1)*NNC+17)*XTA(1) DIST=0.800
X1((1)*NNC+18)*XTA(1) DIST=0.875
X1((1)*NNC+19)*XTA(1) DIST=0.95
X1((1)*NNC+20)*XTA(1) DIST=0.963333
X1((1)*NNC+21)*XTA(1) DIST=0.966667
X1((1)*NNC+22)*XTA(1) DIST=0.993333
X1((1)*NNC+23)*XTA(1) DIST=0.986667

DIST=X2A(I+1)-X2B(I)
X2((1)*NNC+1)*X2A(I)
X2((1)*NNC+2)*X2A(I) DIST=0.05
X2((1)*NNC+3)*X2A(I) DIST=0.10
X2((1)*NNC+4)*X2A(I) DIST=0.175
X2((1)*NNC+5)*X2A(I) DIST=0.25
X2((1)*NNC+6)*X2A(I) DIST=0.325
X2((1)*NNC+7)*X2A(I) DIST=0.40
X2((1)*NNC+8)*X2A(I) DIST=0.60
X2((1)*NNC+9)*X2A(I) DIST=0.80
X2((1)*NNC+10)*X2A(I) DIST=1.15
X2((1)*NNC+11)*X2A(I) DIST=1.50
X2((1)*NNC+12)*X2A(I) DIST=2.50
X2((1)*NNC+13)*X2A(I) DIST=3.50
X2((1)*NNC+14)*X2A(I) DIST=4.75
X2((1)*NNC+15)*X2A(I) DIST=6.00
X2((1)*NNC+16)*X2A(I) DIST=7.00
X2((1)*NNC+17)*X2A(I) DIST=8.00
X2((1)*NNC+18)*X2A(I) DIST=8.75
X2((1)*NNC+19)*X2A(I) DIST=9.50
X2((1)*NNC+20)*X2A(I) DIST=9.683333
X2((1)*NNC+21)*X2A(I) DIST=9.666667
X2((1)*NNC+22)*X2A(I) DIST=9.933333
X2((1)*NNC+23)*X2A(I)

END DO

DO I=1,NCOL.
DIST=XTA(I)+XTA(I+1)-XTB(I)-XTB(I+1)/2.
X1((1)*NNC+1)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+2)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+3)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+4)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+5)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+6)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+7)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+8)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+9)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+10)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+11)*XTA(I+1)-XTA(I)/2.
X1((1)*NNC+12)*XTA(I+1)-XTA(I)/2.

DIST=X2A(I+1)-X2B(I)
X2((1)*NNC+1)*X2A(I+1)
X2((1)*NNC+2)*X2A(I+1)
X2((1)*NNC+3)*X2A(I+1)
X2((1)*NNC+4)*X2A(I+1)
X2((1)*NNC+5)*X2A(I+1)
X2((1)*NNC+6)*X2A(I+1)
X2((1)*NNC+7)*X2A(I+1)
X2((1)*NNC+8)*X2A(I+1)
X2((1)*NNC+9)*X2A(I+1)
X2((1)*NNC+10)*X2A(I+1)

END DO
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+2) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 0.10
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+3) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 0.25
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+4) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 0.40
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+5) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 0.80
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+6) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 1.50
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+7) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 3.00
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+8) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 5.00
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+9) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 10.00
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+10) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 25.00
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+11) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 50.00
X2((1\cdot 1)\cdot \text{NINC}+\text{NUTS}+12) = \frac{x2a(I) \cdot x2a(I+1)}{2} \cdot \text{DIST} \cdot 100.00
END DO

IF (CUT \leq 1) THEN
    DO I = 1, NWODES
        \text{HT}(I) = x2(I)
    END DO
END IF

DO I = 1, NWODES
    WRITE(*,*) x1(I), x2(I), HT(I)
END DO

END
Job SLUINIT (1521) queued to VAXA_TXAO on 26-NOV-1986 09:16 by user UF7105, UIC (UF7105), under account 000744 at priority 100, started on printer VAXASTXA0: on 26-NOV-1986 09:18 from queue VAXA_TXAO.
C THIS PROGRAM APPLIES CONSTANT VELOCITIES TO NODES FOR THE PROBLEM INITIALIZATION

REAL K
OPEN(UNIT=3, NAME='SLUNIT.DAT', STATUS='OLD')
OPEN(UNIT=4, NAME='SLUNIT.OUT', STATUS='NEW')
READ(3,*,END=98)
CONTINUE
READ(3,*)FACTOR1, FACTOR2, FACTOR3
WRITE(4,*)FACTOR1, FACTOR2, FACTOR3
DO I=1,NSTEPS
READ(3,*,END=99)NODE1, NODE2, U1, U2, P, K, E, T1, T2
98 CONTINUE
DO J=NODE1, NODE2
WRITE(4,*)J, U1, U2, P, K, E, T1, T2
99 CONTINUE
END DO
END DO
END
Job SLUC (1520) queued to VAXA_TXAO on 26-NOV-1986 09:15 by user UF7105, UIC (UF7105), under account 000744 at priority 100, started on printer _VAXASTXA_ on 26-NOV-1986 09:17 from queue VAXA_TXAO.
INTEGER NN(500),ZERO,ONE,TWO,THREE,FIVE,SIX,SEVEN,EIGHT
REAL X1(500),X2(500),KAY,EE,UT(500)
OPEN(UNIT=3,NAME='SLUBC.OUT',STATUS='OLD')
OPEN(UNIT=4,NAME='SLUBC.OUT',STATUS='NEW')
ZERO=0 ONE=1 TWO=2 THREE=3 FIVE=5 SIX=6 SEVEN=7 EIGHT=8
NCT=0 NUM=0
READ(3,*,'(X1,I1,X2)',I=1,13)
DO 1=1,469
READ(3,*,'(I1),X1(I),X2(I),DIP')
END DO CONTINUE
DO 1=1,469
IF(X1(I).EQ.0)THEN
NUM=NUM+1 WRITE(4,*,'(I1)',I=1,13)
WRITE(4,*,'(I1),ONE,ZERO')
NCT=NCT+1 END IF
END DO
DO 1=1,469
IF(X1(I).EQ.0)THEN
NUM=NUM+1 WRITE(4,*,'(I1),ONE,ONE')
NCT=NCT+1 END IF
END DO
DO 1=1,469
IF(X1(I).EQ.0)THEN
NUM=NUM+1 WRITE(4,*,'(I1),ONE,KAY')
NCT=NCT+1 END IF
END DO
C WRITE(4,1000),UT(I),UE(I),UE(I),ONE,ONE,ONE
C FORMAT(5X,18,9F12.8)
END
VITA

John I. Finnie

Candidate for the Degree of

Doctor of Philosophy

Dissertation: An Application of the Finite Element Method and Two Equation (k and E) Turbulence Model to Two and Three Dimensional Fluid Flow Problems Governed by the Navier-Stokes Equations.

Major Field: Civil Engineering

Biographical Information:

Personal Data: Born at Vancouver, British Columbia, Canada, December 24, 1951; married to Ruthellen Walker on August 10, 1972; children--Scott, Andrew, Sean

Education:
Ph.D. Utah State University, 1987, in Hydraulics Division of Civil Engineering Department.
M.S. Utah State University, 1985, in Hydraulics Division of Civil Engineering Department.
B.S. California State Polytechnic University, 1975, Agricultural Engineering

Professional Experience:
1984-present Civil engineer with Weber County Surveyor's Office. Performed surface water hydrology and contract document preparation both full and part time.
1982-present Graduate research and teaching assistant in Hydraulics Division, Utah State University
1980 Assistant project engineer at Lompoc, California, with Soil Conservation Service. Helped administer construction contract on concrete flood control channel.
1978-1980 Agricultural engineer with Soil Conservation Service at Auburn, California. Designed irrigation systems and structures. Temporary duty as construction inspector and assistant project engineer on flood control projects.