Program NATCON: For the numerical solution of buoyancy-driven laminar and turbulent flows in differentially heated cavities

by

Mahesh Prakash.
CSIRO Mathematical and Information Services
Private Bag 10, Clayton South, 3169

Özden F. Turan, and Graham R. Thorpe.
School of Architectural, Civil and Mechanical Engineering
Victoria University
PO Box 14428
Melbourne, Australia, 8001

Occasional Paper Number 1

July 2006

PREFACE

Books on computational fluid dynamics (CFD) are often quite theoretical and general, and as such they do not provide users with definitive advice on how to translate the theory into a practical working computer code. On the other hand commercial CFD packages require users to have little or no theoretical knowledge, and they are menu-driven and applications orientated. There are therefore gaps between generalized theory, the writing of 'own-code' and commercial CFD packages. Furthermore, for all of their flexibility commercial CFD packages are often unable to solve the precise problem posed by the user, and user-defined functions have to be written. This requires at least some knowledge of how CFD codes are structured. Students and researchers new to the field of CFD need an interface that relates the differential equations that govern heat, mass and momentum transfer in fluids to CFD codes. If students had access to such an interface their rate of progress could be much higher. This report aims to bridge the gap between theory and application.

The report correlates the equations that govern fluid flow and heat transfer with a FORTRAN 90 code. The program uses the finite volume method, as this has become a widely used technique amongst CFD practitioners. Procedures for discretising the partial differential equations that govern the physics along with how the resulting linear algebraic equations are solved have been described in detail. The grid generation procedure has been discussed at some length, as this is important if the discretisation procedure is to be accurate. The implementation of the hybrid discretisation scheme is illustrated, and it is felt that this will facilitate users to experiment with other schemes. The effects of turbulence are captured using a k-ε model that has been modified to account for near wall effects.

It is strongly recommended that readers use this report along with the book by Patankar (1980) in order to maximize the benefits of this document. Before developing the code the authors had access to the TEACH code that has become ubiquitous, and it shares a similar structure and nomenclature of the TEAM code developed at the University of Manchester (Craft *et al.*, 2002). Users are advised to retain this structure when making modifications to the program so that it

retains a certain universality. The program has been validated against other programs and experimental data as described in Prakash's PhD thesis (2001).

The source code for the case of buoyancy-driven laminar and turbulent flows in differentially heated cavities may be obtained from the authors.

The authors would like to acknowledge Dr Yuguo Li, Dr Li Chen, Dr Jun-de Li and Dr Longde Zhao for their valuable contributions and comments.

M. Prakash
Ö. F. Turan
G. R. Thorpe

CONTENTS

1. INTRODUCTION

There are conceptual barriers between the mathematical formulation of fluid mechanics problems in terms of continuous equations, the discretisation of the equations and numerical methods to solve them, and their ultimate coding in a high-level computer language. This work is essentially didactic in that it aims to reduce these barriers and help students to understand how cfd codes actually work. They will then be in a good position to write their own codes, understand other people's codes, and commercial cfd packages will no longer appear to be solely menu-driven 'black boxes'.

This report contains a detailed description of the program NATCON that solves, using the finite volume method, the equations that govern two-dimensional buoyancy driven turbulent flows in a rectangular enclosure. Natural convection flow occurs due to a temperature difference imposed on the opposite walls of the enclosure. The problem description is presented in Section 2.

The program has a provision to solve steady and unsteady problems with laminar or turbulent flows. The standard k-ε model originally proposed by Harlow and Nakayama (1967) with some modification for natural convection flows (described in Section 3) is used as the turbulence model. Low Reynolds number k-ε models can also be used with some minor modifications to the program. This is also described in Section 3. A description of the non-dimensional equations is given in Section 4. A proper choice of the non-dimensional scheme can have a significant saving on the computer time by way of a reduction in the rounding off errors.

The concept of *staggered grid* to solve the discretized partial differential equations along with *grid generation* is described in Section 5. Section 5 also describes subroutines READDATA and INIT.

The *Gauss Seidel line by line solver*, used to solve all the partial differential equations is described in Section 7.

Section 8 describes subroutines CALCU and CALCV in which the momentum equations are encoded. The SIMPLE algorithm described in Patankar and Spalding (1972) is used to ensure that continuity of mass is conserved. The pressure correction equation forms the backbone of the SIMPLE algorithm, which, along with subroutine CALCP is described in Section 9.

Section 10 describes subroutine PROPS that can be used to make changes to the fluid properties. Section 11 describes subroutine CALCT for the thermal energy equation. Sections 12

and 13 describe subroutine CALCTE and CALCED for the turbulent kinetic energy and energy dissipation respectively.

Subroutine PROMOD that is used to assign boundary conditions to all the variables is described in Section 14. Section 15 describes subroutine UPDATE that is used for unsteady state calculations to update variables after each time iteration. Section 16 describes subroutine DUMP used to restart calculations using a previously calculated field.

The input required for the program and the output in numerical and graphical form are described in Section 17. The main program is listed in Section 18.

2. PROBLEM DESCRIPTION

Consider a closed rectangular cavity, which, is subjected to different thermal boundary conditions. The cavity can have a fluid heated from below with adiabatic vertical walls. This gives rise to a Rayleigh-Benard type of flow. One can also have the vertical walls at different temperatures with adiabatic horizontal walls. All other instances such as conducting horizontal walls with vertical walls at different temperatures, and a cavity with tilted axes are special cases which can be easily achieved with some minor modifications to the present program.

A particularly simple case that illustrates the key features of buoyancy driven flows is a cavity that has differentially heated vertical walls and floors that are adiabatic. **Figure 1** shows the heating from the side case as a representative system with the rectangular cavity filled with a fluid.

In the figure, Q is the heat flux and is zero for the adiabatic horizontal walls, T_h represents the temperature of the hot wall, T_c represents the temperature of the cold wall, H is the total height and L is the total length of the rectangular cavity. Vector $\boldsymbol{g}$ represents acceleration due to gravity. Since the heat flux Q is the first derivative of temperature with respect to space this condition can be mathematically represented as $\frac{\partial T}{\partial y} = 0$ at $y=0$ and $y=H$.

The problem satisfies the following conditions:

(a) The fluid is “almost” incompressible and satisfies the Boussinesq approximation [details can be found in Gray and Giorgini (1976)] which implies that the variation of

density with temperature is negligible except in the buoyancy term of the equation of motion. The buoyancy term occurs in the y-component equation of motion, Equation 3 in Section 3. The density in the buoyancy term is linearized according to

$$\frac{\rho(T)}{\rho(T_o)} = 1 - \beta(T - T_o) \qquad (1)$$

where, ρ is the fluid density, T is the local fluid temperature, T_o is a reference temperature and β is the thermal expansion coefficient of the fluid.

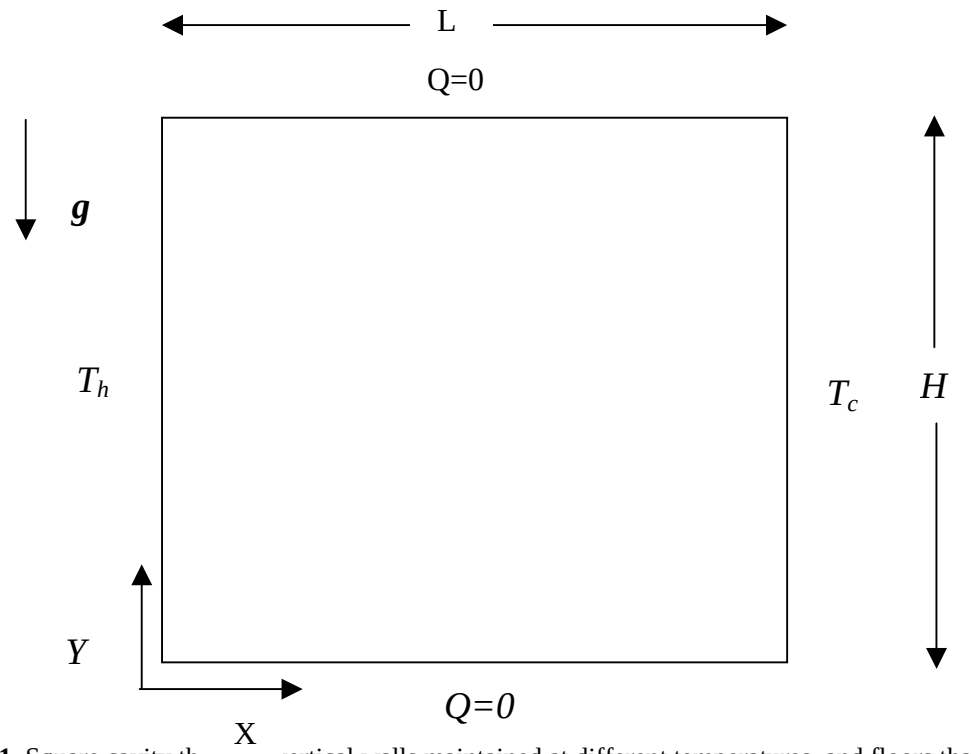

Figure 1. Square cavity th vertical walls maintained at different temperatures, and floors that are adiabatic.

(b) All other thermodynamic and transport properties of the fluid are constant.

(c) The z dimension is much greater than the x and y dimensions and thus the problem can be considered as essentially two-dimensional.

The requirements for these assumptions to be valid must be carefully examined before using the standard program. If any of these assumptions were not valid for a nonstandard problem the program would have to be modified so that it satisfies the specific requirements of the problem.

3. GOVERNING EQUATIONS AND BOUNDARY CONDITIONS

The following set of partial differential equations is solved in the present program.

1. Equation of continuity:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0 \tag{2}$$

in which t represents time, u and v are the components of the fluid velocity in the x and y directions respectively.

2. Momentum equation in the x direction:

$$\rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left[(\mu + \mu_t)\left(2\frac{\partial u}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left[(\mu + \mu_t)\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] \tag{3}$$

Unsteady term — *Advection* — *Pressure* — *Diffusion*

in which p is pressure, μ the fluid viscosity and μ_t the eddy or turbulent viscosity.

3. Momentum equation in the y direction:

$$\rho \frac{\partial v}{\partial t} + \rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial y}\left[(\mu + \mu_t)\left(2\frac{\partial v}{\partial y}\right)\right] + \frac{\partial}{\partial x}\left[(\mu + \mu_t)\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \rho g \beta (T - T_o) \tag{4}$$

Buoyancy

4. Thermal energy equation:

$$\rho \frac{\partial T}{\partial t} + \rho u \frac{\partial T}{\partial x} + \rho v \frac{\partial T}{\partial y} = \frac{\partial}{\partial x}\left[\left(\frac{\mu}{Pr} + \frac{\mu_t}{\sigma_T}\right)\frac{\partial T}{\partial x}\right] + \frac{\partial}{\partial y}\left[\left(\frac{\mu}{Pr} + \frac{\mu_t}{\sigma_T}\right)\frac{\partial T}{\partial y}\right] \tag{5}$$

T is the local fluid temperature, Pr is the fluid Prandtl number and σ_T is the turbulent Prandtl number for temperature.

5. Turbulent kinetic energy equation:

$$\rho\frac{\partial k}{\partial t}+\rho u\frac{\partial k}{\partial x}+\rho v\frac{\partial k}{\partial y}=\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x}\right]+\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial y}\right]+P_k+G_k-\rho\varepsilon+D \tag{6}$$

k is the turbulent kinetic energy, σ_k is the turbulent Prandtl number for k and ε is the rate of energy dissipation. D represents a term which arises when low Reynolds number turbulence models are implemented.

6. Equation for the rate of energy dissipation:

$$\rho\frac{\partial \varepsilon}{\partial t}+\rho u\frac{\partial \varepsilon}{\partial x}+\rho v\frac{\partial \varepsilon}{\partial y}=\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x}\right]+\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial y}\right]+$$
$$(c_{\varepsilon 1}f_1(P_k+c_{\varepsilon 3}G_k)-\rho c_{\varepsilon 3}f_2\varepsilon)\frac{\varepsilon}{k}+E \tag{7}$$

with

$$P_k=\mu_t\left(2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial v}{\partial y}\right)^2+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2\right)$$

$$G_k=-\frac{\mu_t}{\sigma_T}g\beta\frac{\partial T}{\partial y}$$

$$\mu_t=\rho c_\mu f_\mu\frac{k^2}{\varepsilon}$$

E is a term which occurs when low Reynolds number turbulence models are used.

σ_ε is the turbulent Prandtl number for ε.

The following values are empirical constants used in the standard k-ε model.

c_μ=0.09, $c_{\varepsilon 1}$=1.44, $c_{\varepsilon 2}$=1.92, σ_T=0.9, σ_k=1.0, σ_ε=1.3, $f_\mu=f_1=f_2$=1.0.

3.1 Laminar Solutions

For laminar solutions, Equations (6) and (7) are not used for calculations and the eddy viscosity, μ_t, is taken as zero. The variables u, v, p and T are instantaneous quantities for laminar calculations. One can either use a steady approach or a transient approach for laminar calculations. In the former case the time derivatives in Equations (2), (3), (4) and (5) are set to zero. Since no

modifications are carried out in arriving at the unsteady formulation, the solution obtained would approximate to a true transient solution.

3.2 Turbulent Solutions

For turbulent solutions, Equations (6) and (7) are solved simultaneously with Equations (2), (3), (4) and (5). Variables, *u*, *v*, *p* and *T* are, time averaged quantities for turbulent calculations. Here again one can either use a steady approach or approach the steady state solution by integrating through time.

The time derivatives in the time-averaged, Navier-Stokes equation represent the large time behaviour according to Henkes (1990). However the nature of the transient solution will depend on the type of turbulence model used. Thus the transient solution cannot be called a true transient. The eddy viscosity, μ_t, is introduced in the form of a modification to the fluid viscosity as described in Section 10. Quantities *D* and *E* represent terms that need to be added for low Reynolds number *k*-ε models. For the standard *k*-ε model, *D* and *E* are set equal to zero.

More recent experimental data on natural convection in a differentially heated cavity have been provided by Ampofo and Karayiannis (2003) against which the various models may be compared.

3.2a Modification for low Reynolds number models

Low Reynolds number models of Chien (1982) and Jones and Launder (1972) are given as examples.

1. Low Reynolds number *k*-ε model of Chien (1982)

   $c_\mu=0.09,\ c_{\varepsilon 1}=1.35,\ c_{\varepsilon 2}=1.8,\ \sigma_T=0.9,\ \sigma_k=1.0,\ \sigma_\varepsilon=1.3$

   $f_\mu=1-exp(-0.0115x^+),\ f_1=1.0,\ f_2=1-\frac{2}{9}exp(-(Re_t/6)^2),$

   $D=-2\mu\frac{k}{x_n^2},\ E=-\frac{2\mu\varepsilon}{x_n^2}exp(-0.5x^+).$

2. Low Reynolds number k-ε model of Jones and Launder (1972)

c_μ=0.09, $c_{\varepsilon 1}$=1.44, $c_{\varepsilon 2}$=1.92, σ_T=0.9, σ_k=1.0, σ_ε=1.3

$$f_\mu = exp\left(\frac{-2.5}{1+Re_t/50}\right),\ f_1=1.0,\ f_2 = 1-0.3\,exp(-Re_t^2),$$

$$D = -2\mu\left[\left(\frac{\partial\sqrt{k}}{\partial x}\right)^2 + \left(\frac{\partial\sqrt{k}}{\partial y}\right)^2\right],\ E = 2\mu\frac{\mu_t}{\rho}\left[\left(\frac{\partial^2 u}{\partial y^2}\right)^2 + \left(\frac{\partial^2 v}{\partial x^2}\right)\right].$$

3.3 Boundary Conditions

For calculations involving laminar flow natural boundary conditions are applied for *u*, *v* and *T*. The no-slip and impermeable boundary condition is applied to the *u* and *v* velocities.

For the temperature,

$$T=T_h \quad at\ x=0$$

$$T=T_c \quad at\ x=L$$

$$\frac{\partial T}{\partial y}=0 \ at\ y=0$$

$$\frac{\partial T}{\partial y}=0 \ at\ y=H$$

For calculations of turbulent flow wall functions can be introduced for velocities and temperature as well as for *k* and ε. However in the present formulation, wall functions are used only for *k* and ε and the other variables are solved up to the wall.

3.3a Boundary conditions for *k* and ε

1. Standard k-ε model.

$k = \frac{(u^*)^2}{\sqrt{c_\mu f_\mu}}$, $\varepsilon = \frac{(u^*)^3}{\kappa y}$ at the first inner grid point.

where u^* is friction velocity defined by $u^* = \sqrt{\frac{\tau_w}{\rho}}$

where τ_w is the wall shear stress calculated from $\tau_w = \frac{\mu}{\rho}\left(\frac{\partial u}{\partial y}\right)_w$

κ is Von Karman's constant=0.41

and y is the normal distance from the wall.

2. Low Reynolds number models of Chien and Jones and Launder.

$k = \varepsilon = 0$ at the wall.

4. NON-DIMENSIONAL EQUATIONS

A non-dimensional form of the equations reduces the number of independent parameters in the equations and makes the solutions more general for a given set of parameters. It aids in saving computer time by increasing the speed of convergence of the solution. The non-dimensional equations are derived in such a way that only the fluid Prandtl number and Rayleigh number are the dimensionless parameters.

The fluid Prandtl number is defined as $Pr = \frac{C_p \mu}{k_f}$, where C_p is the specific heat of the fluid and k_f is the fluid thermal conductivity. The Rayleigh number is defined as $Ra = \frac{\rho^2 g \beta \Delta T H^3 Pr}{\mu^2}$, where g is acceleration due to gravity and ΔT is the temperature difference between the hot and cold wall.

In case of natural convection flows, for low Prandtl number fluids like gases as well as low viscosity liquids, the convective acceleration term is balanced by the buoyancy term in the momentum equation. Let the subscript *ref*, represent a reference value for all variables and superscript * represent the non-dimensional variable.

Thus one can write,

$$u^* = \frac{u}{u_{ref}},\ v^* = \frac{v}{u_{ref}},\ T^* = \frac{T - T_{ref}}{T_h - T_c},\ x^* = \frac{x}{H},\ y^* = \frac{y}{H},\ p^* = \frac{p}{p_{ref}},\ \varepsilon^* = \frac{\varepsilon}{\varepsilon_{ref}},\ k^* = \frac{k}{k_{ref}},$$

$$\rho^* = \frac{\rho}{\rho_{ref}},\ \mu^* = \frac{\mu}{\mu_{ref}},\ \mu_t^* = \frac{\mu_t}{\mu_{ref}},\ t^* = \frac{t}{t_{ref}} \quad .$$

Using the above non-dimensional variables one can equate the convective acceleration term and buoyancy term in Equation (3) and arrive at: $u_{ref} = \sqrt{g\beta\Delta TH}$. By equating the convective acceleration term with the pressure term one then obtains: $p_{ref} = \rho u_{ref}^2$. The reference temperature is taken as $(T_c+T_h)/2$. The reference density and viscosity are taken as the fluid density and viscosity respectively.

The reference time t_{ref}, is taken as the ratio of the reference length scale and the reference velocity scale, i.e. $t_{ref} = \dfrac{H}{\sqrt{g\beta\Delta TH}}$.

The reference values for turbulent kinetic energy and energy dissipation are derived with the aid of perturbation theory which is described in Wilcox (1993) and are respectively given as:

$$k_{ref} = u_{ref}^2 \,,\ \varepsilon_{ref} = \frac{u_{ref}^3}{H} \,.$$

Using the non-dimensional parameters and dropping the superscript * from all the variables, Equations (2) through (7) can be written as,

1. Equation of continuity:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \qquad (8)$$

2. Momentum equation in the *x* direction:

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[(\mu+\mu_t)\left(2\frac{\partial u}{\partial x}\right)\right] + \sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[(\mu+\mu_t)\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right] \qquad (9)$$

3. Momentum equation in the *y* direction:

$$\rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[(\mu+\mu_t)\left(2\frac{\partial v}{\partial y}\right)\right] + \sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[(\mu+\mu_t)\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]$$

$$+(T-T_o) \tag{10}$$

3. Thermal energy equation:

$$\rho\frac{\partial T}{\partial t}+\rho u\frac{\partial T}{\partial x}+\rho v\frac{\partial T}{\partial y}=\frac{1}{\sqrt{Pr\,Ra}}\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t\,Pr}{\sigma_T}\right)\frac{\partial T}{\partial x}\right]+\frac{1}{\sqrt{Pr\,Ra}}\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t\,Pr}{\sigma_T}\right)\frac{\partial T}{\partial y}\right] \tag{11}$$

4. Turbulent kinetic energy equation:

$$\rho\frac{\partial k}{\partial t}+\rho u\frac{\partial k}{\partial x}+\rho v\frac{\partial k}{\partial y}=\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x}\right]+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial y}\right]+P_k+G_k - \rho\varepsilon \tag{12}$$

5. Equation for energy dissipation:

$$\rho\frac{\partial \varepsilon}{\partial t}+\rho u\frac{\partial \varepsilon}{\partial x}+\rho v\frac{\partial \varepsilon}{\partial y}=\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x}\right]+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial y}\right]+(c_{\varepsilon 1}f_1(P_k+c_{\varepsilon 3}G_k)-\rho c_{\varepsilon 2}f_2\varepsilon)\frac{\varepsilon}{k} \tag{13}$$

with

$$P_k=\mu_t\sqrt{\frac{Pr}{Ra}}\left(2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial v}{\partial y}\right)^2+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2\right)$$

$$G_k=-\frac{1}{\sqrt{Pr\,Ra}}\frac{\mu_t}{\sigma_T}\frac{\partial T}{\partial y}$$

$$\mu_t=\sqrt{\frac{Ra}{Pr}}\rho c_\mu f_\mu\frac{k^2}{\varepsilon}$$

The non-dimensional forms of the equations are now used along with their boundary conditions. The temperatures T_h and T_c become equal to 1 and 0 on a non-dimensional scale.

5. SUBROUTINES INIT AND READDATA (GRID GENERATION, INITIALIZATION AND READING THE INPUT DATA FILE)

The calculation of all variables (i.e., vectors *u* and *v* and scalars *p, T, k* and ε) at one point leads to a non-uniform pressure filed being represented as a uniform pressure field. Also, a physically unrealistic velocity field seems to satisfy the discretized continuity equation. These problems associated with the primitive variable formulation have been described in Patankar

(1980). The problem is overcome by using a different set of points to calculate vectors and scalars. This is called the staggered grid concept where the calculation points for vectors are staggered with respect to the calculation points for scalars. Such a staggered grid for velocity components was first used by Harlow and Welch (1965).

In the staggered grid, the velocity components are calculated for the points that lie on the faces of a control volume. Thus, the *x*-component of velocity *u* is calculated at the faces that are normal to the *x*-direction. The locations for *u* are shown in **Figure 2** by short arrows, while the grid points (hereafter called the main grid points) are shown by the intersections of the solid lines; the dashed lines indicate the control-volume faces.

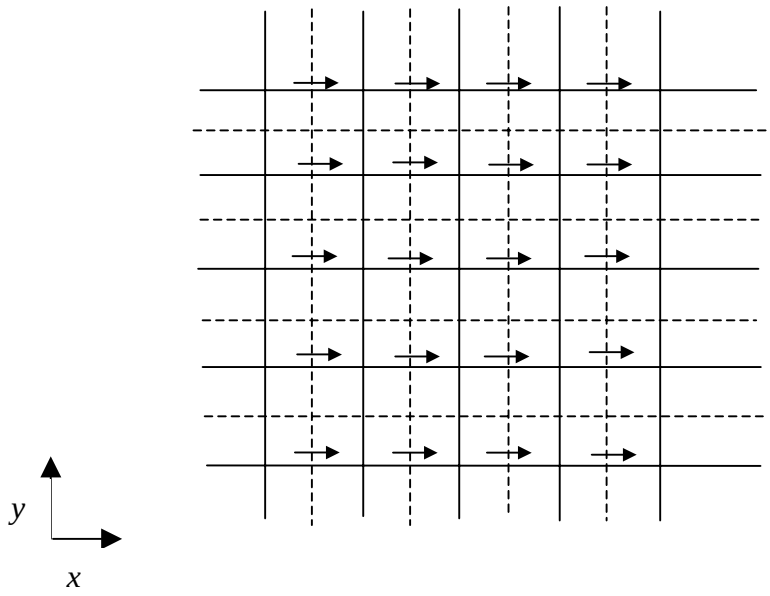

Figure 2. Staggered locations for *u*

Note that with respect to the main grid points, the *u* locations are staggered only in the *x* direction. Similarly the *v* locations are staggered only in the *y* direction. Scalar variables like *T*, *p*, *k* and ε are calculated at the main grid points.

Grids are developed, by using algebraic functions for grid spacing (non-uniform or uniform grid spacing). The staggered grid points are first developed. They are represented as *xu(i*) and *yv(j)* for *x* and *y* directions respectively. The main grid points are then calculated by using the staggered grid locations. **Figure 3** shows the staggered and main grid locations *xu(i)* and *x(i)* respectively for the *x* direction on a 7x7 grid. Note that the boundary of the diagram is the physical boundary of the cavity. The staggered grid starts with *xu(2)* whereas the main grid starts with *x(1)*. Note that *xu(2) = x(1)* and *xu(ni) = x(ni)* with *ni = 7*. This representation allows the imposition of natural boundary conditions for scalar and vector quantities. In the *x* direction, calculations for *u* velocity starts at *xu(3)* and ends at *xu(ni-1)=xu(6)* whereas calculations for scalars and *v* velocity start at *x(2)* and end at *x(ni-1) = x(6)*. Similarly in the *y* direction, calculations for *v* velocity starts at *yv(3)* and ends at *yv(nj-1) = yv(6)* whereas calculations for scalars and *u* velocity start at *y(2)* and end at *y(nj-1) = y(6)*.

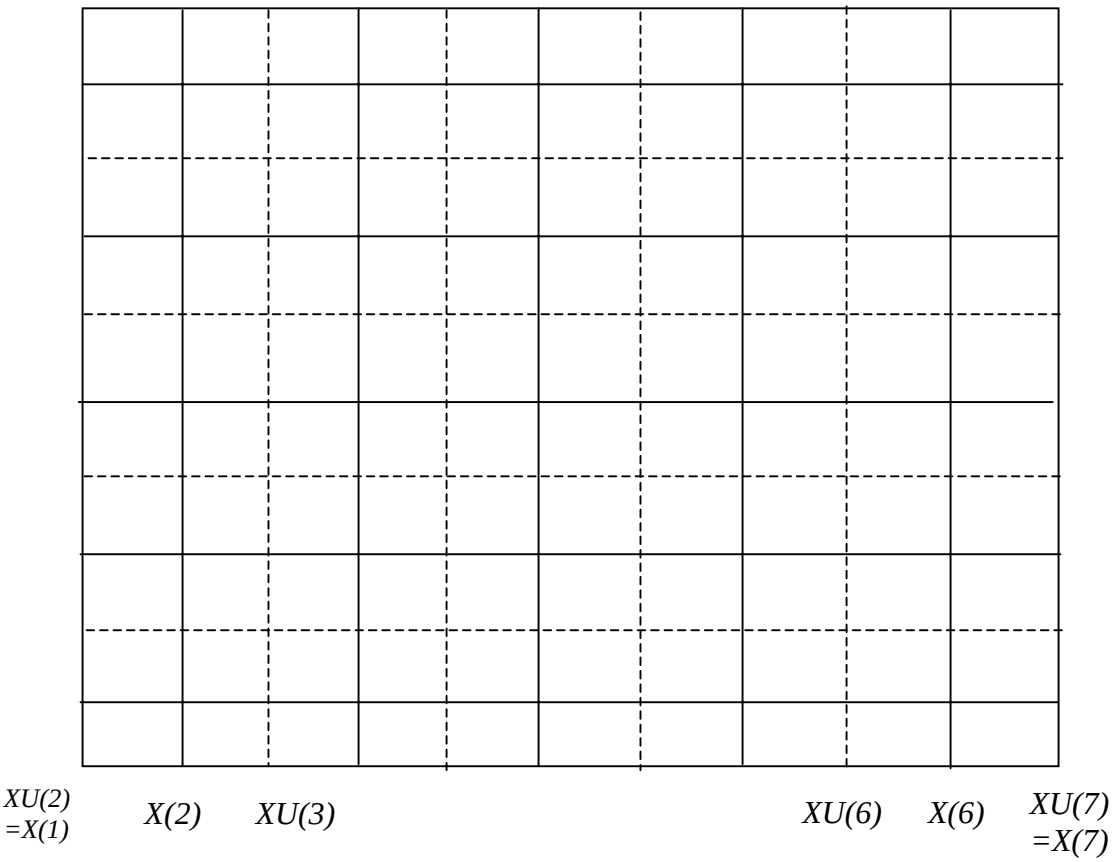

Figure 3. Main and Staggered grid locations for a 7x7 grid

The staggered grid generation is given as GRID GENERATION FUNCTIONS in SUBROUTINE READDATA and the development of the main grids from the staggered grids is shown as CALCULATE GEOMETRICAL QUANTITIES in SUBROUTINE INIT.

This part of the program is given below:

```
NIM1=NI-1
NJM1=NJ-1
NIM2=NI-2
NJM2=NJ-2
```

```
C       GRID GENERATION FUNCTIONS (development of the staggered grid. This is a part C  of
SUBROUTINE READDATA)
     DO 101 I=2,NI
     XU(I)=ELBYH*((I-2)/FLOAT(NIM2)-1/(2*3.14159)*SIN(2*3.14159*(I-2)
    1/FLOAT(NIM2)))
  101 CONTINUE

     DO 105 J=2,NJ
     YV(J)=((J-2)/FLOAT(NJM2)-1/(2*3.14159)*SIN(2*3.14159*(J-2)
    1/FLOAT(NJM2)))
  105 CONTINUE
```

In the example presented above a sine function is used for generating the staggered grid in the *x* and *y* directions. This function can be expressed mathematically as:

$$\frac{xu(i)}{H} = \frac{i-2}{i\max} - \frac{1}{2\pi}\sin\left(2\pi\frac{i}{i\max}\right) \qquad i=imin,\ imax$$

$$\frac{yv(j)}{H} = \frac{j-2}{j\max} - \frac{1}{2\pi}\sin\left(2\pi\frac{i}{i\max}\right) \qquad j=jmin,jmax$$

where *imin=jmin=2*, *imax=NI-2* and *jmax=NJ-2* .

The sine function gives rise to a non-uniform grid which is closely spaced near the wall and sparsely spaced away from the wall. Similarly any other function can be used to define the staggered grid. ELBYH represents the ratio of the length to the height of the cavity. Once the staggered grid is generated, the main grids are created by using the staggered grid co-ordinates in SUBROUTINE INIT. X(I) and Y(J) represent the main grid locations.

```
      SUBROUTINE INIT

          INCLUDE 'common.h'

C     CALCULATE GEOMETRICAL QUANTITIES
          X(1)=XU(2)
          X(NI)=XU(NI)
          DO 101 I=2,NIM1
        101 X(I)=0.5*(XU(I+1)+XU(I))

          Y(1)=YV(2)
          Y(NJ)=YV(NJ)

          DO 102 J=2,NJM1
        102 Y(J)=0.5*(YV(J+1)+YV(J))

          DXPW(1)=0.0
C         (DXPW(I), distance between two consecutive main grid points in the x-direction
C         starting from X(2) to X(NI))

          DXEP(NI)=0.0
C         (DXEP(I), distance between two consecutive main grid points in the x-direction
C         starting from X(1) to X(NIM1))

          DO 103 I=1,NIM1
          DXEP(I)=X(I+1)-X(I)
        103 DXPW(I+1)=DXEP(I)

          DYPS(1)=0.0
C         (DYPS(J), distance between two consecutive main grid points in the y-direction
C         starting from Y(2) to Y(NJ))

          DYNP(NJ)=0.0
C         (DYNP(J), distance between two consecutive main grid points in the y-direction
C         starting from Y(1) to Y(NJM1))

          DO 104 J=1,NJM1
          DYNP(J)=Y(J+1)-Y(J)
        104 DYPS(J+1)=DYNP(J)
```

```
      DXPWU(1)=0.0
      DXPWU(2)=0.0
C     (DXPWU(I), distance between two consecutive staggered grid locations in the x-
C     direction starting from XU(3) to XU(NI))

      DXEPU(1)=0.0
      DXEPU(NI)=0.0
C     (DXEPU(I), distance between two consecutive staggered grid locations in the x-
C     direction starting from XU(2) to XU(NIM1))

      DO 105 I=2,NIM1
      DXEPU(I)=XU(I+1)-XU(I)
  105 DXPWU(I+1)=DXEPU(I)

      DYPSV(1)=0.0
      DYPSV(2)=0.0
C     (DYPSV(J), distance between two consecutive staggered grid locations in the y-
C     direction starting from YV(3) to YV(NJ))

      DYNPV(1)=0.0
      DYNPV(NJ)=0.0
C     (DYNPV(J), distance between two consecutive staggered grid locations in the y-
C     direction starting from YV(2) to YV(NJM1))

      DO 106 J=2,NJM1
DYNPV(J)=YV(J+1)-YV(J)
  106 DYPSV(J+1)=DYNPV(J)

      DO 107 I=1,NI
  107 SEW(I)=DXEPU(I)
C     (SEW(I), area associated with the non-staggered control volume in the x-direction)

      DO 108 J=1,NJ
  108 SNS(J)=DYNPV(J)
C     (SNS(J), area associated with the non-staggered control volume in the y-direction)

      DO 109 I=1,NI
  109 SEWU(I)=DXPW(I)
C     (SEWU(I), area associated with the staggered control volume in the x-direction)

      DO 110 J=1,NJ
  110 SNSV(J)=DYPS(J)
```

```
C       (SNSV(J), area associated with the staggered control volume in the y-direction)
```

As already mentioned the walls of the cavity are located at the staggered locations in order to facilitate the application of the no-slip and impermeable boundary conditions. Thus XU(2), XU(NI), YV(2) and YV(NJ) are located on the cavity walls. The main grid locations, X(1), X(NI), Y(1) and Y(NJ) are set equal to XU(2), XU(NI), YV(2) and YV(NJ) respectively. X(1), X(NI), Y(1) and Y(NJ) are dummy points and are not used for calculations. Such an allocation also enables the use of natural boundary conditions for temperature at the wall. All other non-staggered locations are positioned in between the staggered locations.

Before carrying out calculations all the necessary data are read in by using SUBROUTINE READDATA. This subroutine in turn reads in the data file “IN.DAT”.

```
SUBROUTINE READDATA
    INCLUDE 'common.h'

C       The include statement in FORTRAN does away with all common statements.  This
C       information is stored in the include file common.h.
    LOGICAL INCALU,INCALV,INCALP,INPRO,INCALK,INCALD,INCALM
1       ,INCALT,INHY,INCEN,STEADY

C       These are logicals and are defined at the end of this listing.

    OPEN(2,FILE='in.dat')

C       The file in.dat contains input parameters and is given in Section 17.

C       GRID, ITERATION AND COMPARISON PARAMETERS
            READ(2,'(////)')
    READ(2,*)GREAT,NITER,SMALL,NFTSTP,NLTSTP,STEADY,TFIRST
    WRITE(*,*)"GREAT NITER SMALL NFTSTP NLTSTP STEADY TFIRST"
    WRITE(*,*)GREAT,NITER,SMALL,NFTSTP,NLTSTP,STEADY,TFIRST
    READ(2,*)
    IF(STEADY)NFTSTP=1
    IF(STEADY)NLTSTP=1
    IF(STEADY) DT(1)=GREAT
    READ(2,*)IT,JT
    WRITE(*,*)"IT   JT"
    WRITE(*,*)IT,JT
    READ(2,'(/)')
```

```
      READ(2,*)NSWPU,NSWPV,NSWPP,NSWPK,NSWPD,NSWPT
      WRITE(*,*)"NSWPU  NSWPV  NSWPP  NSWPK  NSWPD  NSWPT"
      WRITE(*,*)NSWPU,NSWPV,NSWPP,NSWPK,NSWPD,NSWPT
      READ(2,'(/)')
      READ(2,*)NI,NJ,ELBYH
      WRITE(*,*)"NI   NJ  ELBYH"
      WRITE(*,*)NI,NJ,ELBYH

C     TIME STEP FOR UNSTEADY CALCULATIONS
      READ(2,'(/)')
      READ(2,*)TSTEP
      WRITE(*,*)"TSTEP"
      WRITE(*,*)TSTEP

C     DEPENDENT VARIABLE, DISCRETIZATION AND RESTART OPTIONS
      READ(2,'(/)')
      READ(2,*)INCALU,INCALV,INCALP,INCALK,INCALD,INPRO,INCALT
      WRITE(*,*)"INCALU INCALV INCALP INCALK INCALD INPRO INCALT"
      WRITE(*,*)INCALU,INCALV,INCALP,INCALK,INCALD,INPRO,INCALT
      READ(2,*)
      READ(2,*)INCALB,INHY,INCEN,VALUE
      WRITE(*,*)"INCALB INHY INCEN VALUE"
      WRITE(*,*)INCALB,INHY,INCEN,VALUE


C     FLUID PROPERTIES
      READ(2,'(/)')
      READ(2,*)DENSIT,PRANDL,VISCOS,CPP
      WRITE(*,*)"DENSIT  PRANDL  VISCOS  CPP"
      WRITE(*,*)DENSIT,PRANDL,VISCOS,CPP
C     ALPHAF represents the thermal diffusivity of the fluid and is defined as  α = μ/(ρ Pr)
      ALPHAF=VISCOS/(DENSIT*PRANDL)

C     TURBULENCE CONSTANTS
      READ(2,'(/)')
      READ(2,*)CMU,CD,C1,C2,CAPPA,ELOG,PRTE,PRANDT
      WRITE(*,*)"CMU  CD  C1  C2  CAPPA  ELOG  PRTE  PRANDT"
      WRITE(*,*)CMU,CD,C1,C2,CAPPA,ELOG,PRTE,PRANDT
      READ(2,*)
      READ(2,*)F1,F2
      WRITE(*,*)"F1,F2"
      WRITE(*,*)F1,F2

C     PRED represents σε, the turbulent Prandtl number for ε.

      PRED=CAPPA*CAPPA/(C2-C1)/(CMU**.5)
```

```
      PFUN=PRANDL/PRANDT
      PFUN=9.24*(PFUN**0.75-1.0)*(1.0+0.28*EXP(-0.007*PFUN))

C     BOUNDARY VALUES
      READ(2,'(/)')
      READ(2,*)TH,TC
      WRITE(*,*)"TH TC"
      WRITE(*,*)TH,TC

C     INTERNAL HEAT GENERATION AND RAYLEIGH NUMBER
      READ(2,'(/)')
      READ(2,*)QGENER,RALI
      WRITE(*,*)"QGENER RALI"
      WRITE(*,*)QGENER,RALI

C     TREF represents the reference temperature.

C     BEITA represents β, the thermal expansion coefficient of the fluid.

C     DELT represents ΔT.


      TREF=(TC+TH)/2
      BEITA=1/(273.15+TREF)
      DELT=TH-TC

C     PRESSURE CALCULATION
      READ(2,'(/)')
      READ(2,*)IPREF,JPREF
      WRITE(*,*)"IPREF JPREF"
      WRITE(*,*)IPREF,JPREF

C     PROGRAM CONTROL AND MONITOR
      READ(2,'(/)')
            READ(2,*)MAXIT,IMON,JMON,URFU,URFV
      WRITE(*,*)"MAXIT  IMON  JMON  URFU  URFV"
      WRITE(*,*)MAXIT,IMON,JMON,URFU,URFV
      READ(2,*)
      READ(2,*)URFP,URFE,URFK,URFT
      WRITE(*,*)"URFP URFE URFK URFT"
      WRITE(*,*)URFP,URFE,URFK,URFT
      READ(2,*)
      READ(2,*)URFG,URFVIS,INDPRI,SORMAX
      WRITE(*,*)"URFG URFVIS INDPRI  SORMAX"
      WRITE(*,*)URFG,URFVIS,INDPRI,SORMAX

C     CAVITY DIMENSIONS
      H=((RALI*VISCOS*ALPHAF)/(DENSIT*9.81*BEITA*DELT))**0.3333
C     EL represents L the length of the cavity
```

```
      EL=H*ELBYH

C     GRID GENERATION FUNCTIONS

      NIM1=NI-1
      NJM1=NJ-1
      NIM2=NI-2
      NJM2=NJ-2

            DO 101 I=2,NI
     XU(I)=ELBYH*((I-2)/FLOAT(NIM2)-1/(2*3.14159)*SIN(2*3.14159*(I-2)
    1/FLOAT(NIM2)))
  101 CONTINUE

      DO 105 J=2,NJ
      YV(J)=((J-2)/FLOAT(NJM2)-1/(2*3.14159)*SIN(2*3.14159*(J-2)
    1/FLOAT(NJM2)))
  105 CONTINUE

C     NON-DIMENSIONALISATION

C     UREF represents u_ref, the reference value for velocity.

            UREF=ALPHAF*(PRANDL*RALI)**0.5/H
C     R1 and R2 are the non-dimensional numbers given by sqrt(Pr/Ra) and sqrt(Pr Ra)
            R1=(PRANDL/RALI)**0.5
      R2=(PRANDL*RALI)**0.5
           CLOSE(2)
      RETURN
      END
```

Following is a listing of the quantities read in from the input data file in.dat.

C GREAT represents a large number that is sometimes used for comparison or for some special purpose like assigning the boundary condition for $\varepsilon = \infty$.

C NITER represents the iteration counter for iterations in a single time step.

C SMALL represents a small number that is used for some special purpose in the program such as preventing division by zero.

C NFTSTP represents the first iteration step for time iterations.

C NLTSTP represents the last iteration step for time iterations.

C STEADY is a LOGICAL . IF STEADY is TRUE then the unsteady terms are omitted from the calculation procedure.

C TFIRST represents the starting value assigned to time *t*.

C IT and JT represent the maximum values that NI and NJ can have. If NI and NJ exceed the value of IT and JT respectively, new values have to be assigned to IT and JT. The program should then be recompiled.

C NSWPU, NSWPV, NSWPP, NSWPK, NSWPD, NSWPT are the total number of internal iterations used to calculate u, v, p', k, ε and T respectively.

C NI and NJ are the total number of grids in the x and y directions respectively.

C ELBYH represents the ratio of length to height of the cavity.

C TSTEP represents the time step for unsteady calculations.

C LOGICALS INCALU, INCALV, INCALP, INCALK, INCALD, INPRO, INCALT activate SUBROUTINES CALCU, CALCV, CALCP, CALCTE, CALCED, PROPS, CALCT respectively.

C LOGICAL INCALB activates the buoyancy terms.

C LOGICALS INHY and INCEN activate the hybrid and central schemes respectively.

C If VALUE equals one, the program uses an initial field that has been fed in by the user. If VALUE equals zero, the program uses the solution that has been dumped in the DUMP file as the initial field. Thus for any fresh calculations, VALUE should always be one.

C DENSIT-fluid density.

C PRANDL-fluid Prandtl number.

C VISCOS-fluid viscosity.

C CMU-turbulence model constant, c_μ.

C CD-damping factor, f_μ.

C C1-turbulence model constant, $c_{\varepsilon 1}$.

C C2-turbulence model constant, $c_{\varepsilon 2}$.

C CAPPA-Von Karman's constant, κ.

C ELOG- represents c^{κ} where c is given by $lnc=5.5$ and κ is Von Karman's constant.

C PRTE-represents σ_{κ}.

C PRANDT-represents turbulent Prandtl number, σ_T.

C F1-damping factor, f_1.

C F2-damping factor, f_2.

C TH-temperature of the hot wall, T_h.

C TC-temperature of the cold wall, T_c.

C QGENER-internal heat generation equals zero for the present problem.

C CPP-specific heat of the fluid, C_P.

C RALI-Rayleigh number.

C IPREF, JPREF-position of reference value for guessed pressure.

C MAXIT-maximum number of space iterations (i.e., number of iterations inside one time step).

C IMON, JMON- monitoring location for different variables.

C URFU-under-relaxation factor for u.

C URFV-under-relaxation factor for v.

C URFP-under-relaxation factor for p.

C URFE-under-relaxation factor for ε.

C URFK-under-relaxation factor for k.

C URFT-under-relaxation factor for T.

C URFG-under-relaxation factor for μ/Pr or $(\mu+\mu_t)/Pr$.

C URFVIS-under-relaxation factor for μ *or* $(\mu+\mu_t)$.

C INDPRI-number of iterations after which labels are printed on the screen.

C SORMAX-convergence criterion.

The variables are initialized in SUBROUTINE INIT immediately after the subsection CALCULATE GEOMETRICAL QUANTITIES.

C Note that the following is a part of SUBROUTINE INIT

C SET VARIABLES TO SMALL VALUE

```
C       UO(I,J), VO(I,J), PO(I,J), TO(I,J), TEO(I,J), EDO(I,J), DENO(I,J) represent the old value
        (i.e.,values at the previous time iteration for the respective variables)
            DO 200 I=1,NI
      DO 200 J=1,NJ

C       SMALL is used as an initial field to prevent division by zero.

      U(I,J)=SMALL
      UO(I,J)=SMALL
      V(I,J)=SMALL
      VO(I,J)=SMALL
      P(I,J)=SMALL
      PO(I,J)=SMALL
      PP(I,J)=SMALL
      T(I,J)=0.5
      TO(I,J)=0.5
      TE(I,J)=SMALL
      TEO(I,J)=SMALL
      ED(I,J)=SMALL
      EDO(I,J)=SMALL
      DEN(I,J)=1.0+SMALL
      DENO(I,J)=1.0+SMALL
      VIS(I,J)=1.0+SMALL
      GAMH(I,J)=1.0+SMALL
      DU(I,J)=0.0
      DV(I,J)=0.0

C       DU(I,J) and DV(I,J) are quantities associated with the velocity correction equation.
C       The velocity correction equation is discussed in Section 9.
      SU(I,J)=0.0
C       SU(I,J) represents the overall source term and is equivalent to term b in Patankar
C       (1980).
      SP(I,J)=0.0
C       SP(I,J) represents SP in S=SC+SP.
  200 CONTINUE
      DO 201 J=1,NJ
      T(1,J)=1.0
  201 T(NI,J)=0.0
      RETURN
      END
```

6. PROGRAM FLOW CHART

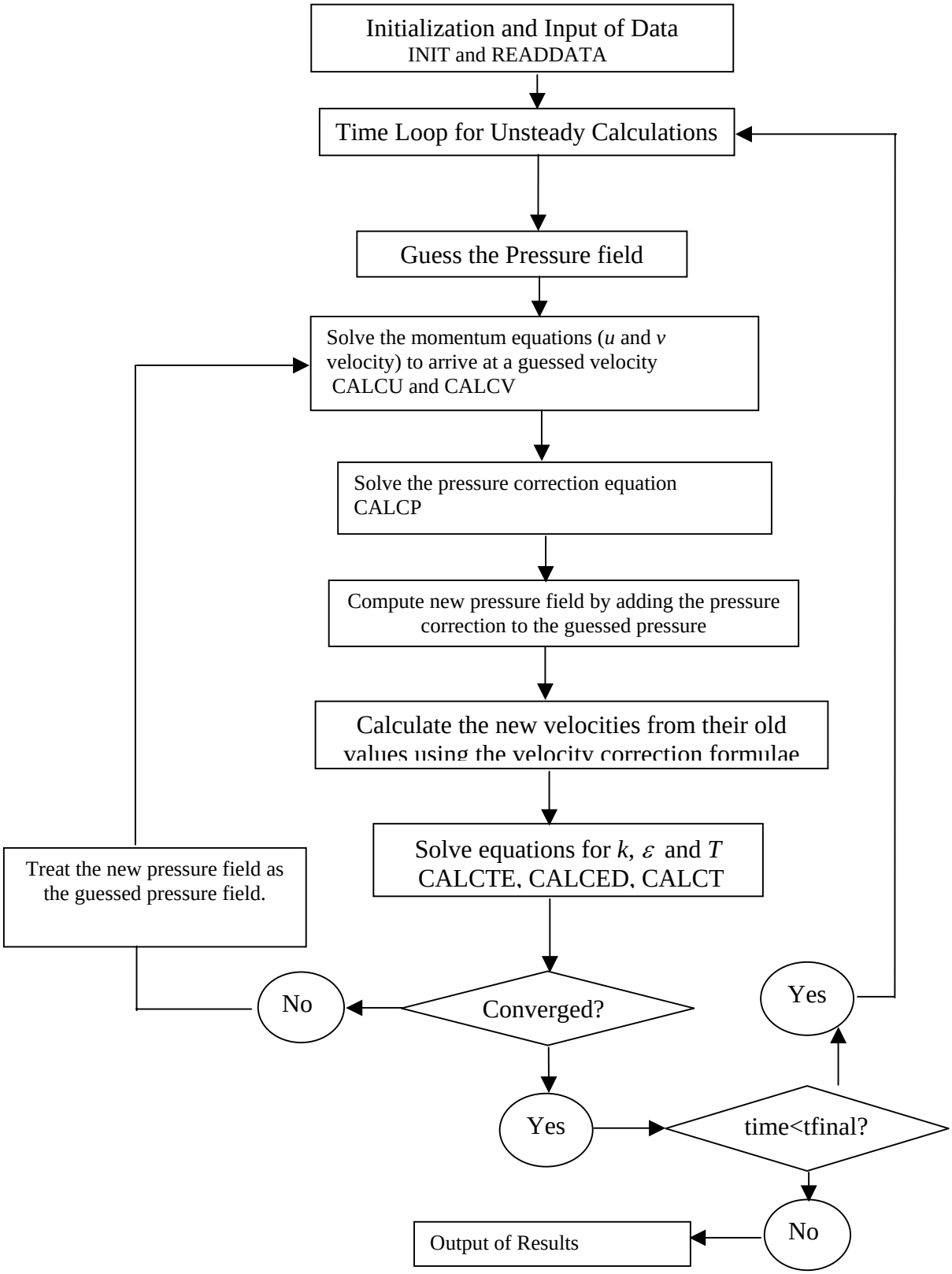

Figure 4. Flow chart explaining details of the solution procedure.
(Names in block letters are those of subroutines.)

The SIMPLE algorithm which stands for *Semi-Implicit Method for Pressure-Linked Equations* is used for calculation of the flow field. The procedure has been described in Patankar and Spalding (1972). The flow chart described in **Figure 4** gives a detailed description of the steps used in calculating the flow field along with the temperature field for the general unsteady turbulent solution. The pressure correction equation is used to incorporate the continuity equation in the solution procedure. The pressure correction equation is described in Section 9.

7. (SUBROUTINE LISOLV) THE GAUSS SEIDEL LINE BY LINE SOLVER

Including the pressure correction equation, there are now six partial differential equations to be solved. The following subroutines represent the six partial differential equations in their discretized form:

CALCU	*x-directional momentum equation*
CALCV	*y-directional momentum equation*
CALCP	*pressure correction equation*
CALCTE	*equation for turbulence kinetic energy*
CALCED	*equation for energy dissipation*
CALCT	*thermal energy equation*

These equations are solved by means of a line by line Gauss-Seidel solver that employs a combination of the *Tri-Diagonal-Matrix Algorithm* (TDMA) for one-dimensional situations and the point by point Gauss-Seidel iterative method.

Following is a description of the TDMA for one dimensional situations:
The one dimensional discretized equation for a variable ϕ can be written as,

$$d_j \phi_j = a_j \phi_{j+1} + b_j \phi_{j-1} + c_j \qquad (14)$$

Where *a, b, c* and *d* represent coefficients of the discretized equation for variable ϕ. Subscript *j* represents a counter for space, *j=jmin, jmax.* The TDMA algorithm consists of a recurrence formula for the variable in question so that one can obtain the new value for ϕ with the help of the boundary conditions.

For the forward substitution process one seeks a relation,

$$\phi_j = P_j \phi_{j+1} + Q_j \tag{15}$$

With *j=j-1* in the above relationship one can arrive at an equation for ϕ_{j-1},

$$\phi_{j-1} = P_{j-1} \phi_j + Q_{j-1} \tag{16}$$

Substitution of Equation (16) into Equation (14) leads to,

$$d_j \phi_j = a_j \phi_{j+1} + b_j \left(P_{j-1} \phi_j + Q_{j-1} \right) + c_j \tag{17}$$

If Equation (17) is rearranged to take the form of Equation (15) and the coefficients are compared, one arrives at a recurrence relationship of the form,

$$P_j = \frac{a_j}{d_j - b_j P_{j-1}} \tag{18}$$

$$Q_j = \frac{c_j + b_j Q_{j-1}}{d_j - b_j P_{j-1}} \tag{19}$$

For $j=jmin$, the recurrence relation (18) and (19) gives a definite value for P_{min} and Q_{min}. Similarly for $j=jmax$, the recurrence relation gives a definite value for P_{max} and Q_{max}. An explanation for a specific boundary condition with temperature as the variable is given in Patankar (1980).

Summary of the algorithm

1. Calculate P_{min} and Q_{min} using the left boundary conditions (i.e., for $j=j_{min}$)
2. Use the recurrence relations (18) and (19) to obtain P_j and Q_j for $j=jmin+1, jmax$.
3. Equate the right boundary conditions (i.e., for $j=j_{max}$) with P_{max} and Q_{max}.
4. Use Equation 15 for $j=jmax-1, jmin$ to obtain $\phi_{jmax-1,}$ ϕ_{jmin}.

For the two dimensional situation one needs to use the Gauss-Seidel point by point method along with the TDMA. The general discretized equation in two dimensions can be written as:

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b \qquad (20)$$

where a_P, a_E, a_W, a_N and a_S represent coefficients associated with the variable ϕ and b represents the source term. In order to be able to use the TDMA one has to choose a particular direction for one sweep and assume the other direction to be a constant. In the present program, the S-N direction is chosen for calculations, and the W-E direction is assumed to be constant for every

sweep. Thus a new source term b_0 is introduced as part of the terms in the W-E direction. Equation (20) is thus modified into,

$$a_P\phi_P = a_N\phi_N + a_S\phi_S + b_0 \qquad (21)$$

where $b_0 = a_E\phi_E + a_W\phi_W + b$.

Discussion on the line by line Gauss-Seidel method

The line by line scheme can be visualized with reference to **Figure 5**. The discretization equations for the grid points along a chosen line are considered first. These contain the values of ϕ at the grid points (shown by squares) along two adjacent lines. If these ϕ*'s* are substituted from their latest values, the equations for the grid points (shown by circles) along the chosen line would look like one-dimensional equations and could be solved by the TDMA. This procedure is carried out for all the lines in the S-N direction.

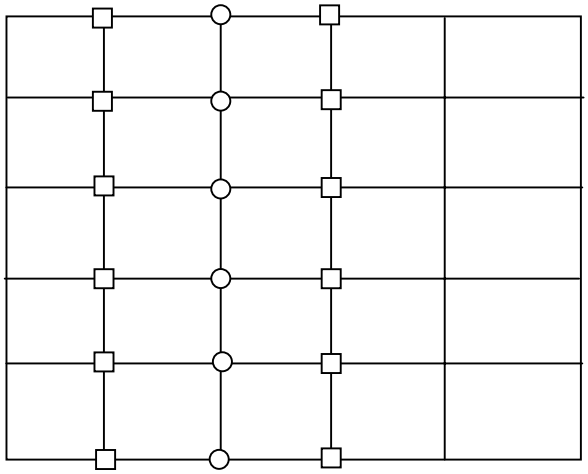

Figure 5. Representation of the line by line method.

In the program, subroutine LISOLV represents the line by line Gauss-Seidel solver.

```
SUBROUTINE LISOLV(ISTART,JSTART,NI,NJ,IT,JT,PHI)

    DIMENSION PHI(IT,JT),A(90),B(90),C(90),D(90)
    COMMON
   1/COEF/AP(80,80),AN(80,80),AS(80,80),AE(80,80),AW(80,80),SU(80,80),
   1     SP(80,80)


    NIM1=NI-1
    NJM1=NJ-1
    JSTM1=JSTART-1
    A(JSTM1)=0.0
C       COMMENCE W-E SWEEP
    DO 100 I=ISTART,NIM1
    C(JSTM1)=PHI(I,JSTM1)

C       COMMENCE S-N TRAVERSE
    DO 101 J=JSTART,NJM1

C       ASSEMBLE TDMA COEFFICIENTS
    A(J)=AN(I,J)
C       (A(J) represents a_j in Equation (14))

    B(J)=AS(I,J)
C       (B(J) represents b_j in Equation (14))

    C(J)=AE(I,J)*PHI(I+1,J)+AW(I,J)*PHI(I-1,J)+SU(I,J)
C       (C(J) represents c_j in Equation (14))

    D(J)=AP(I,J)
C       (D(J) represents d_j in Equation (14))

C       CALCULATE COEFFICIENTS OF RECURRENCE FORMULA
    TERM=1./(D(J)-B(J)*A(J-1))
    A(J)=A(J)*TERM
 101 C(J)=(C(J)+B(J)*C(J-1))*TERM
C       The recurrence formulae (18) and (19) for P_j and Q_j are stored in A(J) and C(J) here.

C       OBTAIN NEW PHI"S
    DO 102 JJ=JSTART,NJM1
    J=NJ+JSTM1-JJ
 102 PHI(I,J)=A(J)*PHI(I,J+1)+C(J)
```

```
100 CONTINUE
    RETURN
    END
```

8. SUBROUTINE CALCU AND CALCV (MOMENTUM EQUATIONS)

Subroutines CALCU and CALCV representing the discretized form of the momentum equations are described here. The momentum equation in the *x* direction can be written as:

$$\rho\frac{\partial u}{\partial t}+\rho u\frac{\partial u}{\partial x}+\rho v\frac{\partial u}{\partial y}=-\frac{\partial p}{\partial x}+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[(\mu+\mu_t)\left(2\frac{\partial u}{\partial x}\right)\right]+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[(\mu+\mu_t)\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]$$

Modifying the diffusion term on the right hand side one can rewrite the equation as follows:

$$\rho\frac{\partial u}{\partial t}+\rho u\frac{\partial u}{\partial x}+\rho v\frac{\partial u}{\partial y}=-\frac{\partial p}{\partial x}+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[(\mu+\mu_t)\frac{\partial u}{\partial x}\right]+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[(\mu+\mu_t)\frac{\partial u}{\partial y}\right]+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[(\mu+\mu_t)\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right] \quad (22)$$

For an incompressible fluid since the density does not change with time, the term:

$$\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[(\mu+\mu_t)\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right] \quad (22a)$$

equals zero due to continuity. The retention of this term increases the numerical accuracy in some types of flows. Therefore this term is included in our formulation as a source term. For a description of the discretization procedure one can refer to Patankar (1980). The initial and final discretized forms in two dimensions is presented here.

Initial discretized form:

$$\frac{(\rho_P u_P-\rho_P^o u_P^o)\Delta x\Delta y}{\Delta t}+J_e-J_w+J_n-J_s=(S_C+S_P u_P)\Delta x\Delta y \quad (23)$$

where

$$J_e=\left\{(\rho u)_e u-(\mu+\mu_t)_e\frac{\partial u}{\partial x}\right\}\Delta y$$

$$J_w = \left\{ (\rho u)_w u - (\mu + \mu_t)_w \frac{\partial u}{\partial x} \right\} \Delta y$$

$$J_n = \left\{ (\rho v)_n u - (\mu + \mu_t)_n \frac{\partial u}{\partial y} \right\} \Delta x$$

$$J_s = \left\{ (\rho v)_s u - (\mu + \mu_t)_s \frac{\partial u}{\partial y} \right\} \Delta x$$

$S = S_C + S_p u_P$ represents the source term. Terms arising due to the non-dimensional form have been omitted for ease of understanding. The *old* values (i.e., the values at the beginning of the time step) are denoted by the superscript *o*.

Final discretized form:

$$a_P u_P = a_E u_E + a_W u_W + a_N u_N + a_S u_S + b \qquad (24)$$

where

$$a_E = D_e A(|P_e|) + [[-F_e, 0]]$$
$$a_W = D_w A(|P_w|) + [[F_w, 0]]$$
$$a_N = D_n A(|P_n|) + [[-F_n, 0]]$$
$$a_S = D_s A(|P_s|) + [[F_s, 0]]$$

{The symbol [[ ]] represents the largest of the quantity contained within it}

$$a_P^o = \frac{\rho_P^o \Delta x \Delta y}{\Delta t}$$

$$b = S_C \Delta x \Delta y + a_P^o u_P^o + M$$

$$a_P = a_E + a_W + a_N + a_S + a_P^o - S_P \Delta x \Delta y \qquad (24a)$$

with $F_e = (\rho u)_e \Delta y$, $D_e = \dfrac{(\mu + \mu_t)_e \Delta y}{(\delta x)_e}$, $P_e = \dfrac{F_e}{D_e}$

$$F_w = (\rho u)_w \Delta y, \; D_w = \frac{(\mu + \mu_t)_w \Delta y}{(\delta x)_w}, \; P_w = \frac{F_w}{D_w}$$

$$F_n = (\rho v)_n \Delta x, \; D_n = \frac{(\mu + \mu_t)_n \Delta x}{(\delta y)_n}, \quad P_n = \frac{F_n}{D_n}$$

$$F_s = (\rho v)_s \Delta x, \; D_s = \frac{(\mu + \mu_t)_s \Delta x}{(\delta y)_s}, \quad P_s = \frac{F_s}{D_s}$$

F represents the strength of convection or the mass flow rates through the faces of the control volume, *D* represents the strength of diffusion and *P* represents the Peclet number which is a ratio of the strengths of convection and diffusion. As shown in **Figure 6**, the subscripts in lower case represent values at the faces of the control volume.

$A(|P|)$ represents a function which assumes different forms for different discretization schemes. The central difference scheme and the hybrid scheme are used in the present program.

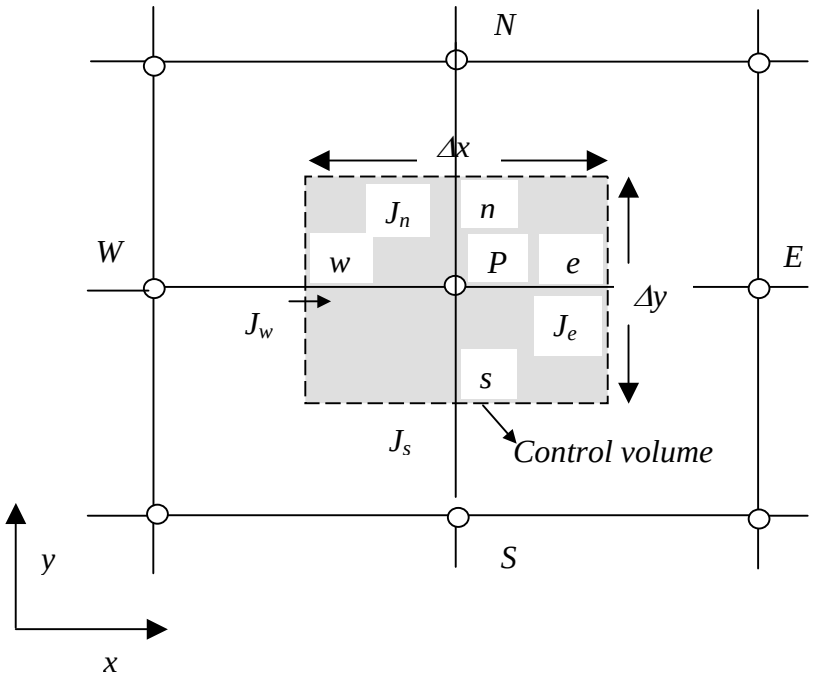

Figure 6. Control volume for a two-dimensional situation.

Other schemes include the upwind scheme, the power law scheme, the exponential or exact scheme and are described in detail in Patankar (1980) and the QUICK scheme of Leonard (1979). The term M in the source term represents modifications to the momentum equation such as the inclusion of the term (22a). The eddy viscosity μ_t is represented with the help of a modification to the fluid viscosity. This modification is carried out through subroutine PROPS described in Section 10. Subroutine PROPS can also be used to modify any other fluid property such as density, for example, variation of density with temperature can be accounted for by using subroutine PROPS. The listing of subroutine CALCU is given below with descriptions in the form of comment statements.

```
SUBROUTINE CALCU

      INCLUDE 'common.h'
      LOGICAL INHY,INCEN,STEADY
C     Note that I starts from 3.  Due to staggering, I=2 represents dummy points.
      DO 100 I=3,NIM1
      DO 101 J=2,NJM1
C     COMPUTE AREAS AND VOLUME
      AREANS=SEWU(I)
C     represents staggered area in the x-direction and applies to fluid in the y-direction
      AREAEW=SNS(J)
C     represents non-staggered area in the y-direction and applies to fluid in the x-direction
      VOL=SEWU(I)*SNS(J)
C     represents the control volume.
C     CALCULATE CONVECTION COEFFICIENTS
C     represents F in Equations (24a).  Note that the variables are to be evaluated at the
C     faces of the control volume.  The U velocity is staggered in the x-direction.  Thus
C     the appropriate interpolated values for V velocity and density need to be taken.
      GN=0.5*(DEN(I,J+1)+DEN(I,J))*V(I,J+1)
      GNW=0.5*(DEN(I-1,J)+DEN(I-1,J+1))*V(I-1,J+1)
      GS=0.5*(DEN(I,J-1)+DEN(I,J))*V(I,J)
      GSW=0.5*(DEN(I-1,J)+DEN(I-1,J-1))*V(I-1,J)
      GE=0.5*(DEN(I+1,J)+DEN(I,J))*U(I+1,J)
      GP=0.5*(DEN(I,J)+DEN(I-1,J))*U(I,J)
      GW=0.5*(DEN(I-1,J)+DEN(I-2,J))*U(I-1,J)


      CN=0.5*(GN+GNW)*AREANS
      CS=0.5*(GS+GSW)*AREANS
      CE=0.5*(GE+GP)*AREAEW
      CW=0.5*(GP+GW)*AREAEW

C     CALCULATE DIFFUSION COEFFICIENTS

C     represents D in Equations (24a).  Appropriate interpolated values need to be taken
C     for viscosity, VIS(I,J).  VIS(I,J) represents either the fluid viscosity (laminar
C     flow) or the total of fluid viscosity and eddy viscosity (turbulent flow).  R1
C     represents the factor sqrt(Pr/Ra) which arises due to non-dimensionalization.
      VISN=0.25*(VIS(I,J)+VIS(I,J+1)+VIS(I-1,J)+VIS(I-1,J+1))
```

```
      VISS=0.25*(VIS(I,J)+VIS(I,J-1)+VIS(I-1,J)+VIS(I-1,J-1))
      DN=R1*VISN*AREANS/DYNP(J)
      DS=R1*VISS*AREANS/DYPS(J)
      DE=R1*VIS(I,J)*AREAEW/DXEPU(I)
      DW=R1*VIS(I-1,J)*AREAEW/DXPWU(I)

C     CALCULATE COEFFICIENTS OF SOURCE TERMS

C     the coefficients of the source term S=SC+SPuP are calculated here

C     CPO*U(I,J) represents SC ΔxΔy and SP(I,J) represents SP ΔxΔy

      SMP=CN-CS+CE-CW
      CP=AMAX1(0.0,SMP)
      CPO=CP

C     ASSEMBLE MAIN COEFFICIENTS
C     the main coefficients aE, aW, aN and aS are evaluated depending on the type of

C     discretization used.  The hybrid scheme (INHY) or the central scheme

C     (INCEN) is used here.

C     For the hybrid scheme the function A(|P|) = [[0,1 – 0.5|P|]]

      IF (INHY) THEN
      AN(I,J)=DN*AMAX1(0.,1-0.5*ABS(CN/DN))+AMAX1(-CN,0.)
      AS(I,J)=DS*AMAX1(0.,1-0.5*ABS(CS/DS))+AMAX1(CS,0.)
      AE(I,J)=DE*AMAX1(0.,1-0.5*ABS(CE/DE))+AMAX1(-CE,0.)
      AW(I,J)=DW*AMAX1(0.,1-0.5*ABS(CW/DW))+AMAX1(CW,0.)
      END IF

C     For the central scheme the function A(|P|) = 1 – 0.5|P|


       IF (INCEN) THEN
      AN(I,J)=AMAX1(-CN,0.)+DN-0.5*ABS(CN)
      AS(I,J)=AMAX1(CS,0.)+DS-0.5*ABS(CS)
      AE(I,J)=AMAX1(-CE,0.)+DE-0.5*ABS(CE)
      AW(I,J)=AMAX1(CW,0.)+DW-0.5*ABS(CW)
      END IF

C     Logical STEADY =TRUE implies that the steady state problem is solved and
C     the unsteady term ρ ∂u/∂t is omitted.

     IF(STEADY) THEN
```

```
      APO(I,J)=0.0
      ELSE
      APO(I,J)=DEN(I,J)*VOL/DT(ITSTEP)
      END IF

C     The pressure gradient is not included in the momentum source term S=S_C+S_P u_P.
C     This is because the pressure field needs to be ultimately calculated .
C     Thus the pressure gradient is included as a separate source term in SU(I,J).
C     It is given here as DU(I,J)*(P(I-1,J)-P(I,J)).
C     (Refer to Section 9 for the pressure correction equation.)

      DU(I,J)=AREAEW
      SU(I,J)=CPO*U(I,J)+DU(I,J)*(P(I-1,J)-P(I,J))+APO(I,J)*UO(I,J)
      SP(I,J)=-CP

C     Extra term to improve numerical stability: sqrt(Pr/Ra) d/dx [ (mu + mu_t)(du/dx + dv/dy) ]

      DUDXP  =(U(I+1,J)-U(I,J))/DXEPU(I)
      DUDXM  =(U(I,J)-U(I-1,J))/DXPWU(I)
      SU(I,J)=R1*(VIS(I,J)*DUDXP-VIS(I-1,J)*DUDXM)/SEWU(I)*VOL+SU(I,J)
      GAMP   =0.25*(VIS(I,J)+VIS(I-1,J)+VIS(I,J+1)+VIS(I-1,J+1))
      DVDXP  =(V(I,J+1)-V(I-1,J+1))/DXPW(I)
      GAMM   =0.25*(VIS(I,J)+VIS(I-1,J)+VIS(I,J-1)+VIS(I-1,J-1))
      DVDXM  =(V(I,J)-V(I-1,J))/DXPW(I)
      SU(I,J) =SU(I,J)+R1*(GAMP*DVDXP-GAMM*DVDXM)/SNS(J)*VOL

  101 CONTINUE
  100 CONTINUE

C     ENTRY MODU in SUBROUTINE PROMOD contains information about the
C     boundary conditions for u-velocity (Section 14).

      CALL MODU


C     The residual source term RESORU gives an idea about the convergence of the
C     solution.  RESORU is the difference in the total source term between two
C     consecutive iteration steps.

      RESORU=0.0
      DO 300 I=3,NIM1
      DO 301 J=2,NJM1
      AP(I,J)=AN(I,J)+AS(I,J)+AE(I,J)+AW(I,J)+APO(I,J)-SP(I,J)
      DU(I,J)=DU(I,J)/AP(I,J)
```

```
      RESOR=AN(I,J)*U(I,J+1)+AS(I,J)*U(I,J-1)+AE(I,J)*U(I+1,J)
     1     +AW(I,J)*U(I-1,J)-AP(I,J)*U(I,J)+SU(I,J)
      VOL=SEW(I)*SNS(J)
      SORVOL=GREAT*VOL
      IF(-SP(I,J).GT.0.5*SORVOL)  RESOR=RESOR/SORVOL
      RESORU=RESORU+ABS(RESOR)

C     UNDER-RELAXATION
C     In an iterative procedure it is often desirable to speed up or slow down changes in
C     the dependent variable from iteration to iteration in order to avoid divergence.
C     The former is achieved by over-relaxation and the latter is achieved by under-
C     relaxation.  The under-relaxation method is used in the present program.  URFU
C     represents the under-relaxation factor used for the u-velocity. The value of under-
C     relaxation factor is always between 0 and 1.

      AP(I,J)=AP(I,J)/URFU
      SU(I,J)=SU(I,J)+(1.-URFU)*AP(I,J)*U(I,J)
      DU(I,J)=DU(I,J)*URFU
  301 CONTINUE
  300 CONTINUE

C     SUBROUTINE LISOLV (Section 7) is used to solve the x-directional momentum
C     equation.  NSWPU represents the number of internal iterations used for u.

      DO 400 N=1,NSWPU
  400 CALL LISOLV(3,2,NI,NJ,IT,JT,U)
      RETURN
      END
```

The subroutine used to calculate the *y*-directional momentum equation, CALCV, is very similar to CALCU. However one has to remember that the calculation points for *v* velocity are staggered in the *y*-direction. An extra source term is added to *b* in the form of the buoyancy term. Following is a listing of SUBROUTINE CALCV.

```
SUBROUTINE CALCV

      INCLUDE 'common.h'
      LOGICAL INCALB,INHY,INCEN,STEADY

C     Note that J starts from 3.  Due to staggering, J=2 represents dummy points.
      DO 100 I=2,NIM1
```

```
      DO 101 J=3,NJM1
C       COMPUTE AREAS AND VOLUME
      AREANS=SEW(I)
C       represents non-staggered area in the x-direction and applies to fluid in the y-direction.

      AREAEW=SNSV(J)
C       represents staggered area in the y-direction and applies to fluid in the x-direction.

      VOL=SEW(I)*SNSV(J)
C       CALCULATE CONVECTION COEFFICIENTS
      GN=0.5*(DEN(I,J+1)+DEN(I,J))*V(I,J+1)
      GP=0.5*(DEN(I,J)+DEN(I,J-1))*V(I,J)
      GS=0.5*(DEN(I,J-1)+DEN(I,J-2))*V(I,J-1)
      GE=0.5*(DEN(I+1,J)+DEN(I,J))*U(I+1,J)
      GSE=0.5*(DEN(I,J-1)+DEN(I+1,J-1))*U(I+1,J-1)
      GW=0.5*(DEN(I,J)+DEN(I-1,J))*U(I,J)
      GSW=0.5*(DEN(I,J-1)+DEN(I-1,J-1))*U(I,J-1)
      CN=0.5*(GN+GP)*AREANS
      CS=0.5*(GP+GS)*AREANS
      CE=0.5*(GE+GSE)*AREAEW
      CW=0.5*(GW+GSW)*AREAEW
C       CALCULATE DIFFUSION COEFFICIENTS
      VISE=0.25*(VIS(I,J)+VIS(I+1,J)+VIS(I,J-1)+VIS(I+1,J-1))
      VISW=0.25*(VIS(I,J)+VIS(I-1,J)+VIS(I,J-1)+VIS(I-1,J-1))
      DN=R1*VIS(I,J)*AREANS/DYNPV(J)
      DS=R1*VIS(I,J-1)*AREANS/DYPSV(J)
      DE=R1*VISE*AREAEW/DXEP(I)
      DW=R1*VISW*AREAEW/DXPW(I)
C       CALCULATE COEFFICIENTS OF SOURCE TERMS
      SMP=CN-CS+CE-CW
      CP=AMAX1(0.0,SMP)
      CPO=CP
C       ASSEMBLE MAIN COEFFICIENTS
      IF (INHY) THEN
      AN(I,J)=DN*AMAX1(0.,1-0.5*ABS(CN/DN))+AMAX1(-CN,0.)
      AS(I,J)=DS*AMAX1(0.,1-0.5*ABS(CS/DS))+AMAX1(CS,0.)
      AE(I,J)=DE*AMAX1(0.,1-0.5*ABS(CE/DE))+AMAX1(-CE,0.)
      AW(I,J)=DW*AMAX1(0.,1-0.5*ABS(CW/DW))+AMAX1(CW,0.)
      END IF



      IF (INCEN) THEN
      AN(I,J)=AMAX1(-CN,0.)+DN-0.5*ABS(CN)
      AS(I,J)=AMAX1(CS,0.)+DS-0.5*ABS(CS)
      AE(I,J)=AMAX1(-CE,0.)+DE-0.5*ABS(CE)
```

```
      AW(I,J)=AMAX1(CW,0.)+DW-0.5*ABS(CW)
      END IF
      IF(STEADY) THEN
      APO(I,J)=0.0
      ELSE
      APO(I,J)=DEN(I,J)*VOL/DT(ITSTEP)
      END IF
      DV(I,J)=AREANS
      SU(I,J)=CPO*V(I,J)+DV(I,J)*(P(I,J-1)-P(I,J))+APO(I,J)*VO(I,J)

C     BUOYANCY TERM

C     Buoyancy term is included as a source term in SU(I,J).  The reference

C     temperature, TREF, is given the value 0.5 which represents (Th+Tc)/2.

C     Depending on the value assigned to TREF the approach to a steady solution would be

C     different.  However the final steady solution will always remain the same.  Note that the

C     temperature, T, has an interpolated value in the buoyancy term BOUYA to account for the

C     staggering.

      TREF=0.0
      IF (INCALB) THEN
      BOUYA=(0.5*(T(I,J)+T(I,J-1))-TREF)
      SU(I,J)=SU(I,J)+BOUYA*VOL
      END IF
      SP(I,J)=-CP

C     Extra term to improve numerical stability: sqrt(Pr/Ra) d/dx [ (mu + mu_t)(du/dx + dv/dy) ]

      DUDYP  =(U(I+1,J)-U(I+1,J-1))/DYPS(J)
      GAMP   =0.25*(VIS(I,J)+VIS(I+1,J)+VIS(I,J-1)+VIS(I+1,J-1))
      GAMM   =0.25*(VIS(I,J)+VIS(I-1,J)+VIS(I,J-1)+VIS(I-1,J-1))
      DUDYM  =(U(I,J)-U(I,J-1))/DYPS(J)
      SU(I,J)=SU(I,J)+R1*(GAMP*DUDYP-GAMM*DUDYM)/SEW(I)*VOL
      DVDYP  =(V(I,J+1)-V(I,J))/DYNPV(J)
      RGAMP  =VIS(I,J)
      DVDYM  =(V(I,J)-V(I,J-1))/DYPSV(J)
      RGAMM  =VIS(I,J-1)
      SU(I,J) =SU(I,J)+R1*(RGAMP*DVDYP-RGAMM*DVDYM)/SNSV(J)*VOL
  101 CONTINUE
  100 CONTINUE

C     ENTRY MODV has information regarding boundary conditions for v (Section 14).
```

```
      CALL MODV
C     RESORV represents the residual source term for the y-directional momentum
C     equation.
      RESORV=0.0
      DO 300 I=2,NIM1
      DO 301 J=3,NJM1
      AP(I,J)=AN(I,J)+AS(I,J)+AE(I,J)+AW(I,J)+APO(I,J)-SP(I,J)
      DV(I,J)=DV(I,J)/AP(I,J)
      RESOR=AN(I,J)*V(I,J+1)+AS(I,J)*V(I,J-1)+AE(I,J)*V(I+1,J)
     1     +AW(I,J)*V(I-1,J)-AP(I,J)*V(I,J)+SU(I,J)
      VOL=SEW(I)*SNS(J)
      SORVOL=GREAT*VOL
      IF(-SP(I,J).GT.0.5*SORVOL)  RESOR=RESOR/SORVOL
      RESORV=RESORV+ABS(RESOR)

C     UNDER-RELAXATION
C     URFV represents under-relaxation factor for v velocity.
      AP(I,J)=AP(I,J)/URFV
      SU(I,J)=SU(I,J)+(1.-URFV)*AP(I,J)*V(I,J)
      DV(I,J)=DV(I,J)*URFV
  301 CONTINUE
  300 CONTINUE


C     Subroutine LISOLV (Section7) is used to solve the y-directional momentum equation.

C     NSWPV represents the number of internal iterations used for v.

      DO 400 N=1,NSWPV
  400 CALL LISOLV(2,3,NI,NJ,IT,JT,V)
      RETURN
      END
```

9. SUBROUTINE CALCP (THE PRESSURE CORRECTION EQUATION)

The continuity equation is included in the solution procedure through the introduction of the pressure correction equation in case of the SIMPLE ALGORITHM that is used in the present program. A relationship between pressure and velocity is derived. This is used in the continuity equation to derive the pressure correction equation. A detailed derivation of the pressure correction equation is given in Patankar (1980). The main steps in the derivation are given here.
Let p^* be the guessed pressure, p the corrected pressure and p' the pressure correction. Then one can write:

$$p = p^* + p' \tag{25}$$

A similar equation can be written for the corrected velocities, *u* and *v*:

$$u = u^* + u', \quad v = v^* + v' \tag{26}$$

where u^* and v^* are the guess velocities and u' and v' are the velocity corrections.

The velocity correction formulae can be written as:

$$u' = d_w(p'_W - p'_P), \quad v' = d_s(p'_S - p'_P) \tag{27}$$

where

$$d_w = \frac{A_{ew}}{a_P} \text{ and } d_s = \frac{A_{ns}}{a_P}.$$

A_{ew} and A_{ns} represent areas associated with the East-West and North-South directions respectively. In Patankar (1980) a slightly different formulation is given for the velocity correction formulae but both the formulations have the same meaning. Thus Equation (27) gives a relationship between the velocity correction and pressure correction. One can now write Equation (26) as follows:

$$u = u^* + d_w(p'_W - p'_P), \; v = v^* + d_s(p'_S - p'_P) \tag{28}$$

The continuity equation can be written as:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \tag{29}$$

This equation is integrated over the shaded control volume in **Figure 7.**

For the integration of the term $\partial\rho/\partial t$, the density, ρ_P, is assumed to prevail over the control volume. Since a fully implicit procedure is used for time, the new values of velocity and density (i.e., those at time $t + \Delta t$) are assumed to prevail over the time step; the old density, ρ_P^o (i.e., at time *t*), will appear only through the term $\partial\rho/\partial t$. Thus the integrated form of Equation (29) becomes

$$\frac{\left(\rho_P - \rho_P^o\right)\Delta x \Delta y}{\Delta t} + \left[(\rho u)_e - (\rho u)_w\right]\Delta y + \left[(\rho v)_n - (\rho v)_s\right]\Delta x = 0 \qquad (30)$$

Equation (30) is now converted to the pressure correction equation, using Equation (27). The final discretized form of the pressure correction equation can be written as:

$$a_P p_P^{'} = a_E p_E^{'} + a_W p_W^{'} + a_N p_N^{'} + a_S p_S^{'} + b \qquad (31)$$

where

$$a_E = \rho_e d_e \Delta y ,$$
$$a_W = \rho_w d_w \Delta y ,$$
$$a_N = \rho_n d_n \Delta x ,$$

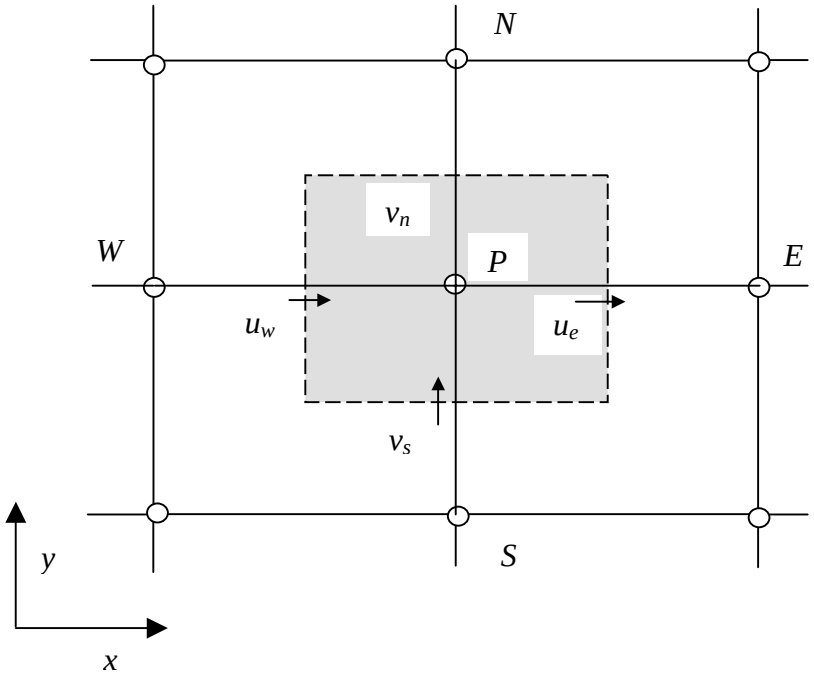

Figure 7. Control volume for the continuity equation.

$$a_S = \rho_s d_s \Delta x ,$$
$$a_P = a_E + a_W + a_N + a_S ,$$
$$b = \frac{\left(\rho_P^o - \rho_P\right)\Delta x \Delta y}{\Delta t} + \left[\left(\rho u^*\right)_w - \left(\rho u^*\right)_e\right]\Delta y + \left[\left(\rho v^*\right)_s - \left(\rho v^*\right)_n\right]\Delta x . \qquad (31a)$$

Equation (31) is now solved as the pressure correction equation. The following is a listing of SUBROUTINE CALCP that solves Equation (31). PP(I,J) represents the pressure correction p'.

```
SUBROUTINE CALCP

INCLUDE 'common.h'
      LOGICAL STEADY
      RESORM=0.0

      DO 100 I=2,NIM1
      DO 101 J=2,NJM1
C         COMPUTE AREAS AND VOLUME

C         Areas and volume are non-staggered.
      AREANS=SEW(I)
      AREAEW=SNS(J)
      VOL=SNS(J)*SEW(I)

C         CALCULATE COEFFICIENTS
C         Interface densities are required but densities are available only at the main grid
C         points.  Therefore they need to be interpolated
      DENN=0.5*(DEN(I,J)+DEN(I,J+1))
      DENS=0.5*(DEN(I,J)+DEN(I,J-1))
      DENE=0.5*(DEN(I,J)+DEN(I+1,J))
      DENW=0.5*(DEN(I,J)+DEN(I-1,J))
      AN(I,J)=DENN*AREANS*DV(I,J+1)
      AS(I,J)=DENS*AREANS*DV(I,J)
      AE(I,J)=DENE*AREAEW*DU(I+1,J)
      AW(I,J)=DENW*AREAEW*DU(I,J)

C         CALCULATE SOURCE TERMS
      CN=DENN*V(I,J+1)*AREANS
      CS=DENS*V(I,J)*AREANS
      CE=DENE*U(I+1,J)*AREAEW
      CW=DENW*U(I,J)*AREAEW
      SMP=CN-CS+CE-CW

C         Note that there is no SP term in Equation (31).  Thus SP(I,J)=0.
      SP(I,J)=0.0

      IF(STEADY) THEN
      SU(I,J)=-SMP
      ELSE

C         In the present problem the unsteady term can be dropped because the fluid is
```

```
C       incompressible.  However this term is retained to maintain generality.
      SU(I,J)=-SMP+(DENO(I,J)-DEN(I,J))*VOL/DT(ITSTEP)
      END IF
C       COMPUTE SUM OF ABSOLUTE MASS SOURCES

C       RESORM represents the residual mass source.

      RESORM=RESORM+ABS(SMP)
  101 CONTINUE
  100 CONTINUE

C       ENTRY MODP can have information about any modifications to conditions in
C       the pressure field.  However here ENTRY MODP does not introduce any changes
C       in the pressure field (Section 14).

        CALL MODP

      DO 300 I=2,NIM1
      DO 301 J=2,NJM1
  301 AP(I,J)=AN(I,J)+AS(I,J)+AE(I,J)+AW(I,J)-SP(I,J)
  300 CONTINUE

C       SUBROUTINE LISOLV is used to solve the pressure correction equation.
C       NSWPP represents the number of internal iterations applied to the pressure
C       correction equation
      DO 400 N=1,NSWPP
  400 CALL LISOLV(2,2,NI,NJ,IT,JT,PP)

C       VELOCITIES
      DO 500 I=2,NIM1
      DO 501 J=2,NJM1
      IF(I.NE.2) U(I,J)=U(I,J)+DU(I,J)*(PP(I-1,J)-PP(I,J))
      IF(J.NE.2) V(I,J)=V(I,J)+DV(I,J)*(PP(I,J-1)-PP(I,J))

C       Represents Equation (28)
  501 CONTINUE
  500 CONTINUE

C       PRESSURES (WITH PROVISION FOR UNDER-RELAXATION)
C       IPREF and JPREF are reference values for pressure for the guess pressure field.


C       URFP represents under-relaxation factor for pressure.
```

```
      PPREF=PP(IPREF,JPREF)
      DO 502 I=2,NIM1
      DO 503 J=2,NJM1
      P(I,J)=P(I,J)+URFP*(PP(I,J)-PPREF)
      PP(I,J)=0.0
  503 CONTINUE
  502 CONTINUE
        RETURN
        END
```

10. SUBROUTINE PROPS (MODIFICATION TO FLUID PROPERTIES)

SUBROUTINE PROPS is used to make modifications to the fluid properties. In the present problem all fluid properties are constant. However it is very convenient to express the turbulent or eddy viscosity as a part of the fluid viscosity for turbulent flow calculations. For laminar flow this term is set to zero. The following is a listing of SUBROUTINE PROPS.

```
      SUBROUTINE PROPS

      INCLUDE 'common.h'

      DO 100 I=2,NIM1
      DO 100 J=2,NJM1
C     GAMH(I,J) represents the ratio of fluid viscosity and fluid Prandtl number.
C     If the fluid properties are variable then VISOLD and GAMHOLD store the values
C     of VIS(I,J) and GAMH(I,J) from the previous iteration.  For laminar flow there is
C     no such change since the fluid viscosity, is not a function of any other variable
C     like temperature.  However for turbulent flow the eddy viscosity, μt, is
C     incorporated in the definition of VIS(I,J).  Thus VIS(I,J) and GAMH(I,J) vary
C     with every iteration.
      VISOLD=VIS(I,J)
      GAMHOLD=GAMH(I,J)
C     If ED(I,J) equals zero, there is no turbulence.  Thus VIS(I,J) equals the fluid
C     viscosity.
      IF(ED(I,J).EQ.0) GOTO 102
C     The eddy viscosity has the following form: μt = sqrt(Ra/Pr) ρ cμ fμ k^2/ε
C     Note that VIS(I,J) represents the sum total of fluid viscosity and eddy viscosity.
```

```
      VIS(I,J)=(1/R1)*DEN(I,J)*TE(I,J)**2*CMU*CD/ED(I,J)+1.0

     GO  TO  101
  102 VIS(I,J)=1.0

C        UNDER RELAX VISCOSITY
C        URFVIS represents the under-relaxation factor for viscosity
C        URFG represents the under-relaxation factor for GAMH(I,J)
  101 VIS(I,J)=URFVIS*VIS(I,J)+(1.-URFVIS)*VISOLD
      GAMH(I,J)=1.0+(VIS(I,J)-1.0)*PRANDL/PRANDT
      GAMH(I,J)=URFG*GAMH(I,J)+(1.-URFG)*GAMHOLD
  100 CONTINUE
      RETURN
      END
```

11. SUBROUTINE CALCT (THERMAL ENERGY EQUATION)

The temperature field is resolved by using the thermal energy equation. It is a scalar variable and is thus calculated at non-staggered locations. If the temperature field does not affect the flow field this equation can be solved after a convergent flow field is obtained. However in the present problem the temperature field does affect the flow field. The thermal energy equation in non-dimensional form is:

$$\rho\frac{\partial T}{\partial t}+\rho u\frac{\partial T}{\partial x}+\rho v\frac{\partial T}{\partial y}=\frac{1}{\sqrt{Pr\,Ra}}\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_{t\,Pr}}{\sigma_T}\right)\frac{\partial T}{\partial x}\right]+\frac{1}{\sqrt{Pr\,Ra}}\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_{t\,Pr}}{\sigma_T}\right)\frac{\partial T}{\partial y}\right] \quad (32)$$

Following is a listing of SUBROUTINE CALCT, which solves the thermal energy equation.

```
SUBROUTINE CALCT

      INCLUDE 'common.h'
      LOGICAL INHY,INCEN,STEADY

      DO 100 I=2,NIM1
      DO 101 J=2,NJM1

C        COMPUTE AREAS AND VOLUME
C        non-staggered areas and volume have been used.
      AREANS=SEW(I)
      AREAEW=SNS(J)
      VOL=SNS(J)*SEW(I)
```

```
C       CALCULATE CONVECTION COEFFICIENTS
      GN=0.5*(DEN(I,J)+DEN(I,J+1))*V(I,J+1)
      GS=0.5*(DEN(I,J)+DEN(I,J-1))*V(I,J)

      GE=0.5*(DEN(I,J)+DEN(I+1,J))*U(I+1,J)
      GW=0.5*(DEN(I,J)+DEN(I-1,J))*U(I,J)
      CN=GN*AREANS
      CS=GS*AREANS
      CE=GE*AREAEW
      CW=GW*AREAEW

C       CALCULATE DIFFUSION COEFFICIENTS
C       R2 represents the non-dimensional factor sqrt(Pr Ra) .

      GAMN=0.5*(GAMH(I,J)+GAMH(I,J+1))
      GAMS=0.5*(GAMH(I,J)+GAMH(I,J-1))
      GAME=0.5*(GAMH(I,J)+GAMH(I+1,J))
      GAMW=0.5*(GAMH(I,J)+GAMH(I-1,J))
      DN=(1/R2)*GAMN*AREANS/DYNP(J)
      DS=(1/R2)*GAMS*AREANS/DYPS(J)
      DE=(1/R2)*GAME*AREAEW/DXEP(I)
      DW=(1/R2)*GAMW*AREAEW/DXPW(I)

C       SOURCE TERMS
      SMP=CN-CS+CE-CW
      CP=AMAX1(0.0,SMP)
      CPO=CP

C       ASSEMBLE MAIN COEFFICIENTS
      IF (INHY) THEN
      AN(I,J)=DN*AMAX1(0.,1-0.5*ABS(CN/DN))+AMAX1(-CN,0.)
      AS(I,J)=DS*AMAX1(0.,1-0.5*ABS(CS/DS))+AMAX1(CS,0.)
      AE(I,J)=DE*AMAX1(0.,1-0.5*ABS(CE/DE))+AMAX1(-CE,0.)
      AW(I,J)=DW*AMAX1(0.,1-0.5*ABS(CW/DW))+AMAX1(CW,0.)
      END IF

      IF (INCEN) THEN
      AN(I,J)=AMAX1(-CN,0.)+DN-0.5*ABS(CN)
      AS(I,J)=AMAX1(CS,0.)+DS-0.5*ABS(CS)
      AE(I,J)=AMAX1(-CE,0.)+DE-0.5*ABS(CE)
      AW(I,J)=AMAX1(CW,0.)+DW-0.5*ABS(CW)
      END IF
      IF(STEADY) THEN
      APO(I,J)=0.0

      ELSE
      APO(I,J)=DEN(I,J)*VOL/DT(ITSTEP)
```

```
      END IF
      SU(I,J)=CPO*T(I,J)+APO(I,J)*TO(I,J)
      SP(I,J)=-CP
  101 CONTINUE
  100 CONTINUE


C     ENTRY MODT  contains information about boundary conditions for T  (Section 14).
      CALL MODT
C         RESORT represents the residual source for thermal energy.
      RESORT=0.0
      DO 300 I=2,NIM1
      DO 301 J=2,NJM1
      AP(I,J)=AN(I,J)+AS(I,J)+AE(I,J)+AW(I,J)+APO(I,J)-SP(I,J)
      RESOR=AN(I,J)*T(I,J+1)+AS(I,J)*T(I,J-1)+AE(I,J)*T(I+1,J)
     1      +AW(I,J)*T(I-1,J)-AP(I,J)*T(I,J)+SU(I,J)
      VOL=SEW(I)*SNS(J)
      SORVOL=GREAT*VOL
      IF(-SP(I,J).GT.0.5*SORVOL)  RESOR=RESOR/SORVOL
      RESORT=RESORT+ABS(RESOR)

C         UNDER-RELAXATION
C         URFT represents under-relaxation factor for temperature.
      AP(I,J)=AP(I,J)/URFT
      SU(I,J)=SU(I,J)+(1.0-URFT)*AP(I,J)*T(I,J)
  301 CONTINUE
  300 CONTINUE

C         NSWPT represents the number of internal iterations applied to the thermal energy
C         equation.
      DO 400 N=1,NSWPT
  400 CALL LISOLV(2,2,NI,NJ,IT,JT,T)
      RETURN
      END
```

12. SUBROUTINE CALCTE (EQUATION FOR TURBULENT KINETIC ENERGY)

The turbulent kinetic energy, *k*, is a scalar variable and is thus calculated on a non-staggered grid. The non-dimensional form of the equation can be written as:

$$\rho\frac{\partial k}{\partial t}+\rho u\frac{\partial k}{\partial x}+\rho v\frac{\partial k}{\partial y}=\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x}\right]+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial y}\right]+P_k+G_k$$

$$-\rho\varepsilon \qquad (33)$$

with,

$$P_k = \mu_t \sqrt{\frac{Pr}{Ra}} \left(2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 \right)$$

$$G_k = -\frac{1}{\sqrt{Pr\,Ra}} \frac{\mu_t}{\sigma_T} \frac{\partial T}{\partial y} \qquad \text{and}$$

$$\mu_t = \sqrt{\frac{Ra}{Pr}} \rho c_\mu f_\mu \frac{k^2}{\varepsilon} \qquad (33a)$$

Following is a listing of SUBROUTINE CALCTE, which solves the turbulent kinetic energy equation.

```
SUBROUTINE CALCTE

      INCLUDE 'common.h'
      LOGICAL INCALB,INHY,INCEN,STEADY

C     PRTE represents σk.
      PRTE=1.0
      DO 100 I=2,NIM1
      DO 101 J=2,NJM1
C     COMPUTE AREAS AND VOLUME
      AREANS=SEW(I)
      AREAEW=SNS(J)
      VOL=SNS(J)*SEW(I)
C     CALCULATE CONVECTION COEFFICIENTS
      GN=0.5*(DEN(I,J)+DEN(I,J+1))*V(I,J+1)
      GS=0.5*(DEN(I,J)+DEN(I,J-1))*V(I,J)
      GE=0.5*(DEN(I,J)+DEN(I+1,J))*U(I+1,J)
      GW=0.5*(DEN(I,J)+DEN(I-1,J))*U(I,J)
      CN=GN*AREANS
      CS=GS*AREANS
      CE=GE*AREAEW
      CW=GW*AREAEW

C     CALCULATE DIFFUSION COEFFICIENTS

C     VIS(I,J) represents the total of the fluid and eddy viscosity, whereas the diffusion
C     term: √(Pr/Ra) ∂/∂x[(μ+μt/σk)∂k/∂x] + √(Pr/Ra) ∂/∂y[(μ+μt/σk)∂k/∂y] has got the fluid viscosity
```

```
C       and the eddy viscosity as separate entities.

      GAMN=R1*0.5*(VIS(I,J)+VIS(I,J+1)-2.0)/PRTE+R1
      GAMS=R1*0.5*(VIS(I,J)+VIS(I,J-1)-2.0)/PRTE+R1
      GAME=R1*0.5*(VIS(I,J)+VIS(I+1,J)-2.0)/PRTE+R1
      GAMW=R1*0.5*(VIS(I,J)+VIS(I-1,J)-2.0)/PRTE+R1
      DN=GAMN*AREANS/DYNP(J)
      DS=GAMS*AREANS/DYPS(J)
      DE=GAME*AREAEW/DXEP(I)
      DW=GAMW*AREAEW/DXPW(I)



C       SOURCE TERMS
      SMP=CN-CS+CE-CW
      CP=AMAX1(0.0,SMP)
      CPO=CP
      DUDX=(U(I+1,J)-U(I,J))/SEW(I)
      DVDY=(V(I,J+1)-V(I,J))/SNS(J)
      DUDY=((U(I,J)+U(I+1,J)+U(I,J+1)+U(I+1,J+1))/4.-(U(I,J)+U(I+1,J)+
     1U(I,J-1)+U(I+1,J-1))/4.)/SNS(J)
      DVDX=((V(I,J)+V(I,J+1)+V(I+1,J)+V(I+1,J+1))/4.-(V(I,J)+V(I,J+1)+V(
     1I-1,J)+V(I-1,J+1))/4.)/SEW(I)
C       GEN(I,J) represents the term: P_k = μ_t sqrt(Pr/Ra) ( 2(∂u/∂x)^2 + 2(∂v/∂y)^2 + (∂u/∂y + ∂v/∂x)^2 )
C       Note that only the eddy viscosity is used for multiplication.  The factor 1.E-8 is
C       added in order to avoid multiplication by zero.

      GEN(I,J)=R1*(2.*(DUDX**2+DVDY**2)+(DUDY+DVDX)**2)
     1 *(VIS(I,J)-1.0+1.E-8)

C       BUOYANCY TERM
      DTDY=(T(I,J+1)-T(I,J))/DYPS(J)
C       GENB(I,J) represents the term: G_k = -1/sqrt(Pr Ra) (μ_t/σ_T) (∂T/∂y).
      GENB(I,J)=-(1/R2)*(VIS(I,J)-1.0+1.E-8)*PRANDL*DTDY/PRANDT

C       ASSEMBLE MAIN COEFFICIENTS
      IF (INHY) THEN
      AN(I,J)=DN*AMAX1(0.,1-0.5*ABS(CN/DN))+AMAX1(-CN,0.)
      AS(I,J)=DS*AMAX1(0.,1-0.5*ABS(CS/DS))+AMAX1(CS,0.)
      AE(I,J)=DE*AMAX1(0.,1-0.5*ABS(CE/DE))+AMAX1(-CE,0.)
      AW(I,J)=DW*AMAX1(0.,1-0.5*ABS(CW/DW))+AMAX1(CW,0.)
      END IF

      IF (INCEN) THEN
```

```
      AN(I,J)=AMAX1(-CN,0.)+DN-0.5*ABS(CN)
      AS(I,J)=AMAX1(CS,0.)+DS-0.5*ABS(CS)
      AE(I,J)=AMAX1(-CE,0.)+DE-0.5*ABS(CE)
      AW(I,J)=AMAX1(CW,0.)+DW-0.5*ABS(CW)
      END IF
      IF(STEADY) THEN
      APO(I,J)=0.0
      ELSE

      APO(I,J)=DEN(I,J)*VOL/DT(ITSTEP)
      ENDIF
      SU(I,J)=CPO*TE(I,J)+APO(I,J)*TEO(I,J)
      SU(I,J)=SU(I,J)+GEN(I,J)*VOL
      IF (INCALB) THEN

      SU(I,J)=SU(I,J)+GENB(I,J)*VOL
      END IF
      SP(I,J)=-CP
C     The term ρε is inlcuded as a variable part of the source term, i.e., as SP. Note that
C     the relationship: μt = sqrt(Ra/Pr) ρ cμ fμ k^2/ε is used to express ρε as:
C     sqrt(Ra/Pr) cμ fμ ρ^2 k^2/μt. This procedure is adopted to increase numerical stability.
C     The factor 1.E-8 is included in μt =(VIS(I,J)-1.0) in order to avoid division by
C     zero.

      SP(I,J)=SP(I,J)-(1/R1)*CD*CMU*DEN(I,J)**2*TE(I,J)*VOL
     1/(VIS(I,J)-1.0+1.E-8)
  101 CONTINUE
  100 CONTINUE


C     ENTRY MODTE has information regarding boundary conditions for turbulence
C     kinetic energy (Section 14).
      CALL MODTE
C     RESORK represents the residual source term for turbulent kinetic energy.
      RESORK=0.0
      DO 300 I=2,NIM1
      DO 301 J=2,NJM1
      AP(I,J)=AN(I,J)+AS(I,J)+AE(I,J)+AW(I,J)+APO(I,J)-SP(I,J)
      RESOR=AN(I,J)*TE(I,J+1)+AS(I,J)*TE(I,J-1)+AE(I,J)*TE(I+1,J)
     1     +AW(I,J)*TE(I-1,J)-AP(I,J)*TE(I,J)+SU(I,J)
      VOL=SEW(I)*SNS(J)
```

```
      SORVOL=GREAT*VOL
      IF(-SP(I,J).GT.0.5*SORVOL)  RESOR=RESOR/SORVOL
      RESORK=RESORK+ABS(RESOR)

C     UNDER-RELAXATION

C     URFK represents under-relaxation factor for turbulent kinetic energy.

      AP(I,J)=AP(I,J)/URFK
      SU(I,J)=SU(I,J)+(1.-URFK)*AP(I,J)*TE(I,J)
  301 CONTINUE
  300 CONTINUE

C     NSWPK represents the number of internal iterations used to solve the turbulent

C     kinetic energy equation.


      DO 400 N=1,NSWPK
  400 CALL LISOLV(2,2,NI,NJ,IT,JT,TE)
      DO 401 I=2,NIM1
      DO 401 J=2,NJM1
  401 TE(I,J)=AMAX1(TE(I,J),SMALL)
      RETURN
      END
```

13. SUBROUTINE CALCED (ENERGY DISSIPATION EQUATION)

Energy dissipation, ε, is yet another scalar quantity. Thus the calculation points for this quantity are also on a non-staggered grid. The non-dimensional form of this equation can be written as:

$$\rho\frac{\partial\varepsilon}{\partial t}+\rho u\frac{\partial\varepsilon}{\partial x}+\rho v\frac{\partial\varepsilon}{\partial y}=\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial\varepsilon}{\partial x}\right]+\sqrt{\frac{Pr}{Ra}}\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial\varepsilon}{\partial y}\right]+$$
$$(c_{\varepsilon 1}f_1(P_k+c_{\varepsilon 3}G_k)-\rho c_{\varepsilon 2}f_2\varepsilon)\frac{\varepsilon}{k} \qquad (34)$$

with,

$$P_k=\mu_t\sqrt{\frac{Pr}{Ra}}\left(2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial v}{\partial y}\right)^2+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2\right)$$

$$G_k=-\frac{1}{\sqrt{Pr\,Ra}}\frac{\mu_t}{\sigma_T}\frac{\partial T}{\partial y}$$

$$\mu_t = \sqrt{\frac{Ra}{Pr}}\rho c_\mu f_\mu \frac{k^2}{\varepsilon} \qquad (34a)$$

Following is a listing of SUBROUTINE CALCED which calculates energy dissipation.

```
SUBROUTINE CALCED
    INCLUDE 'common.h'
    LOGICAL INCALB,INHY,INCEN,STEADY

    DO 100 I=2,NIM1
    DO 101 J=2,NJM1
C       COMPUTE AREAS AND VOLUME
    AREANS=SEW(I)
    AREAEW=SNS(J)
    VOL=SNS(J)*SEW(I)



C       CALCULATE CONVECTION COEFFICIENTS
    GN=0.5*(DEN(I,J)+DEN(I,J+1))*V(I,J+1)
    GS=0.5*(DEN(I,J)+DEN(I,J-1))*V(I,J)
    GE=0.5*(DEN(I,J)+DEN(I+1,J))*U(I+1,J)
    GW=0.5*(DEN(I,J)+DEN(I-1,J))*U(I,J)
    CN=GN*AREANS
    CS=GS*AREANS
    CE=GE*AREAEW
    CW=GW*AREAEW

C       CALCULATE DIFFUSION COEFFICIENTS
C       The diffusion term: sqrt(Pr/Ra) d/dx[(mu + mu_t/sigma_e) d(eps)/dx] + sqrt(Pr/Ra) d/dy[(mu + mu_t/sigma_e) d(eps)/dy] has the fluid
C       viscosity and eddy viscosity as separate identities.

    GAMN=R1*0.5*(VIS(I,J)+VIS(I,J+1)-2.0)/PRED+R1
    GAMS=R1*0.5*(VIS(I,J)+VIS(I,J-1)-2.0)/PRED+R1
    GAME=R1*0.5*(VIS(I,J)+VIS(I+1,J)-2.0)/PRED+R1
    GAMW=R1*0.5*(VIS(I,J)+VIS(I-1,J)-2.0)/PRED+R1
    DN=GAMN*AREANS/DYNP(J)
    DS=GAMS*AREANS/DYPS(J)
    DE=GAME*AREAEW/DXEP(I)
    DW=GAMW*AREAEW/DXPW(I)

C       SOURCE TERMS
    SMP=CN-CS+CE-CW
```

```
      CP=AMAX1(0.0,SMP)
      CPO=CP

C     ASSEMBLE MAIN COEFFICIENTS
      IF (INHY) THEN
      AN(I,J)=DN*AMAX1(0.,1-0.5*ABS(CN/DN))+AMAX1(-CN,0.)
      AS(I,J)=DS*AMAX1(0.,1-0.5*ABS(CS/DS))+AMAX1(CS,0.)
      AE(I,J)=DE*AMAX1(0.,1-0.5*ABS(CE/DE))+AMAX1(-CE,0.)
      AW(I,J)=DW*AMAX1(0.,1-0.5*ABS(CW/DW))+AMAX1(CW,0.)
      END IF

      IF (INCEN) THEN
      AN(I,J)=AMAX1(-CN,0.)+DN-0.5*ABS(CN)
      AS(I,J)=AMAX1(CS,0.)+DS-0.5*ABS(CS)
      AE(I,J)=AMAX1(-CE,0.)+DE-0.5*ABS(CE)
      AW(I,J)=AMAX1(CW,0.)+DW-0.5*ABS(CW)
      END IF

      IF(STEADY) THEN
      APO(I,J)=0.0
      ELSE
      SU(I,J)=CPO*ED(I,J)+APO(I,J)*EDO(I,J)

      END IF
C     The coefficient c_ε3 does not have a universally acceptable form.  The form
C     suggested by Henkes (1990) i.e., c_ε3 = tanh|v/u| is used here.

      C3=ABS((V(I,J)+V(I,J+1))/(U(I,J)+U(I+1,J)))
      C3=TANH(C3)

C     Represents the addition of the term: c_ε1 f_1 P_k ε/k to SU(I,J).

C     Note that the term ε/k is not directly used. Instead the relationship:

C     μ_t = sqrt(Ra/Pr) ρ c_μ f_μ k^2/ε is used to write ε/k = sqrt(Ra/Pr) ρ c_μ f_μ k/μ_t.

C     This procedure is adopted to improve numerical stability.

      SU(I,J)=SU(I,J)+(1/R1)*C1*F1*CMU*CD*DEN(I,J)*GEN(I,J)*VOL
     1 *TE(I,J)/(VIS(I,J)-1.0+1.E-8)

C     Represents the addition of the term: c_ε1 f_1 c_ε3 G_k ε/k to SU(I,J)

C     Here again the same representation is used for the term ε/k.

      IF (INCALB) THEN
```

```
      SU(I,J)=SU(I,J)+(1/R1)*C1*F1*C3*CMU*CD*GENB(I,J)*DEN(I,J)*VOL
     1 *TE(I,J)/(VIS(I,J)-1.0+1.E-8)
      END IF
      SP(I,J)=-CP
C     Represents the addition of the term:  ρc_ε2 f_2 ε ε/k
C     No changes have been made in representing this term.

       SP(I,J)=SP(I,J)-C2*F2*DEN(I,J)*ED(I,J)*VOL/TE(I,J)
  101 CONTINUE
  100 CONTINUE

C     ENTRY MODED has information regarding boundary conditions for energy
C     dissipation (Sectio 14).
      CALL MODED

C     RESORE represents the residual source term for energy dissipation.
      RESORE=0.0

      DO 300 I=2,NIM1
      DO 301 J=2,NJM1
      AP(I,J)=AN(I,J)+AS(I,J)+AE(I,J)+AW(I,J)+APO(I,J)-SP(I,J)

     RESOR=AN(I,J)*ED(I,J+1)+AS(I,J)*ED(I,J-1)+AE(I,J)*ED(I+1,J)
     1     +AW(I,J)*ED(I-1,J)-AP(I,J)*ED(I,J)+SU(I,J)
      VOL=SNS(J)*SEW(I)
      SORVOL=GREAT*VOL
      IF(-SP(I,J).GT.0.5*SORVOL)  RESOR=RESOR/SORVOL
      RESORE=RESORE+ABS(RESOR)

C     UNDER-RELAXATION
C     URFE represents under-relaxation factor employed for energy dissipation ED(I,J)
      AP(I,J)=AP(I,J)/URFE
      SU(I,J)=SU(I,J)+(1.-URFE)*AP(I,J)*ED(I,J)
  301 CONTINUE
  300 CONTINUE

C     NSWPD represents the number of internal iterations used for calculating ED(I,J)
      DO 400 N=1,NSWPD
  400 CALL LISOLV(2,2,NI,NJ,IT,JT,ED)
      DO 401 I=2,NIM1
      DO 401 J=2,NJM1
  401 ED(I,J)=AMAX1(ED(I,J),SMALL)
      RETURN
```

```
      END
```

14. SUBROUTINE PROMOD (BOUNDARY CONDITIONS)

Boundary conditions are specified as PROBLEM MODIFICATIONS through SUBROUTINE PROMOD. The SUBROUTINE has different subs-sections called ENTRY (which is a FORTRAN command) and are used to specify boundary conditions for specific variables. The following is a listing of SUBROUTINE PROMOD.

```
SUBROUTINE PROMOD

      INCLUDE 'common.h'
C         For the fluid properties.  No changes are required for this part.
      ENTRY MODPRO
     RETURN
C         represents boundary conditions for u-velocity. No slip and impermeable boundary
C         conditions are applied at the walls.
      ENTRY MODU

C         TOP WALL
      J=NJM1
      DO  210 I=3,NIM1
  210 U(I,J+1)=0.0

C         WEST WALL
C         I=2 represents the west wall because u-velocity is calculated on a staggered grid
C         in the x-direction
      I=3
      DO 213 J=2,NJM1
  213 U(I-1,J)=0.0

C         BOTTOM WALL
      J=2
      DO 214 I=3,NIM1
  214 U(I,J-1)=0.0

C         EAST WALL
      I=NIM1
      DO 217 J=2,NJM1
  217 U(I+1,J)=0.0
      RETURN
```

```
C       represents boundary conditions for v-velocity. No slip and impermeable boundary
C       conditions are applied at the wall.
      ENTRY MODV
C       WEST WALL
      I=2
      DO  310 J=3,NJM1
  310 V(I-1,J)=0.0

C       TOP WALL
      J=NJM1
      DO  313  I=2,NIM1
  313 V(I,J+1)=0.0

C       EAST WALL
      I=NIM1
      DO 314 J=3,NJM1
  314 V(I+1,J)=0.0



C       BOTTOM WALL
C       J=2 represents the bottom wall because v-velocity is calculated on a staggered
C       grid in the y-direction
       J=3
      DO 317 I=2,NIM1
  317 V(I,J-1)=0.0
      RETURN

C       A pressure boundary condition is not applied here.  Therefore no modifications
C       are required for the pressure correction equation.
      ENTRY MODP
      RETURN

C       represents thermal boundary conditions.  The horizontal walls (top and bottom
C       walls) are adiabatic whereas the vertical walls (east and west walls) are at a
C       constant temperature.

       ENTRY MODT
C       TOP WALL (ADIABTIC)
C       The adiabatic boundary condition:  ∂T / ∂y = 0  at y=0 and y=H, are approximated
```

```
C       by a first order Taylor series approximation.  A better approximation was not
C       required because of the fine grids close to the wall.
      J=NJM1
      DO 500 I=2,NIM1
      T(I,J+1)=T(I,J)
  500 AN(I,J)=0.0

C       WEST WALL (CONSTANT TEMPERATURE TH)
      I=2
      DO 501 J=2,NJM1
      T(I-1,J)=1.0
  501 CONTINUE

C       BOTTOM WALL (ADIABATIC)
      J=2
      DO 504 I=2,NIM1
      T(I,J-1)=T(I,J)
  504 AS(I,J)=0.0


C       EAST WALL (CONSTANT TEMPERATURE TC)
      I=NIM1
      DO 505 J=2,NJM1
      T(I+1,J)=0.0
  505  CONTINUE
      RETURN
C       represents boundary conditions for the turbulent kinetic energy.  One can use the
C       natural boundary condition for k which is zero at the wall. This is quite


C       straightforward.  The boundary condition arising out of perturbation which is
C       derived in Wilcox (1993) is used in the present program.  This boundary condition
C       is given by the equation: k = (u*)^2 / sqrt(c_mu f_mu)  (where u* is the friction velocity) close to
C       the wall.  The boundary condition is applied at the first inner grid point.
      ENTRY MODTE
C       TOP WALL
      J=NJM1
      YP=YV(NJ)-Y(NJM1)
      DO 610 I=2,NIM1
      TAU=1.0*ABS(U(I,J))/YP
      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  610 TE(I,J)=USTAR**2/SQRT(CMU*CD)
```

```
C       WEST WALL
      I=2
      XP=X(2)-XU(2)
      DO 620 J=2,NJM1
      TAU=1.0*ABS(V(I,J))/XP
      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  620 TE(I,J)=USTAR**2/SQRT(CMU*CD)

C       BOTTOM WALL
      J=2
      YP=Y(2)-YV(2)
      DO 630 I=2,NIM1
      TAU=1.0*ABS(U(I,J))/YP
      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  630 TE(I,J)=USTAR**2/SQRT(CMU*CD)

C       EAST WALL
      I=NIM1
      XP=XU(NI)-X(NIM1)
      DO 640 J=2,NJM1
      TAU=1.0*ABS(V(I,J))/XP

      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  640 TE(I,J)=USTAR**2/SQRT(CMU*CD)
      RETURN


C       represents boundary conditions for energy dissipation.  There are no natural
C       boundary conditions for ε.  Therefore the boundary condition from perturbation
C       theory is used here.  ε = (u*)^3/(κy)  (where κ is Von Karman's constant and y
C       is the normal distance from the wall at which the boundary condition is applied) is
C       the boundary condition for ε and occurs close to the wall.  This boundary
C       condition is again applied at the first inner grid point.


      ENTRY MODED
C       TOP WALL
      YP=YV(NJ)-Y(NJM1)
      J=NJM1
      DO  710  I=2,NIM1
      TAU=1.0*ABS(U(I,J))/YP
      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  710 ED(I,J)=USTAR**3/(CAPPA*YP)
```

```
C       WEST WALL
      XP=X(2)-XU(2)
      I=2
      DO  720  J=2,NJM1
      TAU=1.0*ABS(V(I,J))/XP
      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  720 ED(I,J)=USTAR**3/(CAPPA*XP)

C       BOTTOM WALL
      YP=Y(2)-YV(2)
      J=2
      DO 730 I=2,NIM1
      TAU=1.0*ABS(U(I,J))/YP
      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  730 ED(I,J)=USTAR**3/(CAPPA*YP)

C       EAST WALL
      XP=XU(NI)-X(NIM1)
      I=NIM1
      DO 740 J=2,NJM1
      TAU=1.0*ABS(V(I,J))/XP
      USTAR=R1**0.5*SQRT(TAU/DEN(I,J))
  740 ED(I,J)=USTAR**3/(CAPPA*XP)
      RETURN
      END
```

15. SUBROUTINE UPDATE (UNSTEADY CALCULATIONS)

For unsteady calculations, the converged solution from the previous time iteration is also required for calculations. The results of the converged solution from the previous iteration are stored as old values and are represented as UO(I,J) for *u*-velocity, VO(I,J) for *v*-velocity, PO(I,J)

for pressure, DENO(I,J) for density, TO(I,J) for temperature, TEO(I,J) for turbulent kinetic energy and EDO(I,J) for energy dissipation. Thus the variables are updated after each time iteration for use in the next time iteration. Following is the listing of SUBROUTINE UPDATE

```
SUBROUTINE UPDATE(PHI,PHIO,NI,NJ,IT,JT)
C       PHI(I,J) stands for any variable u, v, T, p, k, ε or the fluid density. PHIO(I,J)
C       stands for the value of the same variable at the previous time step.
```

```
      DIMENSION PHI(80,80),PHIO(80,80)
      NIM1=NI-1
      NJM1=NJ-1
      DO 100 I=2,NIM1
      DO 100 J=2,NJM1
  100 PHIO(I,J)=PHI(I,J)
      RETURN
      END
```

16. SUBROUTINE DUMP (RESTARTING CALCULATIONS)

Since the number of iterations required for a converged solution cannot be known beforehand, one needs a facility by which a previously calculated solution field can be used for further iterations. This facility is provided through the SUBROUTINE DUMP. The solution after a particular number of iterations or after satisfying a particular convergence criterion, whichever occurs earlier, is stored in *binary* form in a file called the DUMP file. Storage in *binary* form is essential to prevent any loss of information due to truncation. This DUMP file can then be recalled for further calculations. Following is the listing of SUBROUTINE DUMP

```
SUBROUTINE DUMP(NI,NJ,U,V,P,T,TE,ED,DEN,GAMH,VIS)
      DIMENSION U(80,80),V(80,80),P(80,80),T(80,80),TE(80,80)
     1,ED(80,80),DEN(80,80),GAMH(80,80),VIS(80,80)
      WRITE(10)((U(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((V(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((P(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((T(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((TE(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((ED(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((DEN(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((GAMH(I,J),I=1,NI),J=1,NJ)
      WRITE(10)((VIS(I,J),I=1,NI),J=1,NJ)

     RETURN
      END
```

17. INPUT AND OUTPUT

17.1. Input

Input to the program is carried out through an external input file ‘in.dat’. This input file is in turn read in through SUBROUTINE READDATA discussed in Section 5. The meaning of every input parameter is listed in Section 5.

```
----------------------------------------------------------------------
This data file is for program NATCON
----------------------------------------------------------------------

C   GRID, ITERATION AND COMPARISON PARAMETERS
      GREAT    NITER  SMALL  NFTSTP  NLTSTP  STEADY  TFIRST
      1.E20      0      1.E-20     1       100       .TRUE.     0.
      IT   JT
      80   80
      NSWPU  NSWPV  NSWPP  NSWPK  NSWPD  NSWPT
      1          1          3          1        1        1
      NI   NJ   ELBYH
      60   60     1.0
C   TIME STEP FOR UNSTEADY CALCULATIONS
      TSTEP
      0.25
C   DEPENDENT VARIABLE, DISCRETIZATION  AND RESTART OPTIONS
      INCALU INCALV INCALP INCALK INCALD INPRO INCALT
      .TRUE. .TRUE. .TRUE. .FALSE. .FALSE. .FALSE. .TRUE.
      INCALB  INHY    INCEN   VALUE
      .TRUE.  .FALSE. .TRUE.     1.
C   FLUID PROPERTIES
      DENSIT   PRANDL  VISCOS   CPP
      1.2          0.71       1.85E-5   1006.
C   TURBULENCE CONSTANTS
      CMU   CD   C1   C2   CAPPA   ELOG   PRTE  PRANDT
      0.09    1.00  1.44  1.92   .41       10.0       1.0     0.9
       F1   F2
      1.00 1.00
C   BOUNDARY VALUES
      TH  TC
      40  30
C   INTERNAL HEAT GENERATION AND RAYLEIGH NUMBER
      QGENER     RALI
      0.              1.E5


C   PRESSURE CALCULATION
      IPREF JPREF
      2         2
C   PROGRAM CONTROL AND MONITOR
      MAXIT    IMON  JMON  URFU URFV
      200        30       30       0.4     0.4
      URFP   URFE   URFK    URFT
      0.5      0.3      0.3       0.8
      URFG  URFVIS INDPRI  SORMAX
```

0.5 0.5 100 0.0000000001

17.2. Output

Numerical and graphical outputs are dependent on the nature of outputs required. The principal outputs are given through the output data file 'TEA.OUT' which contains numerical output and 'TEC.DAT' which contains data for graphical output using the software TECPLOT. The *streamfunction* is calculated using the formula:

$u = \frac{\partial \Psi}{\partial y}, v = -\frac{\partial \Psi}{\partial x}$ with the boundary condition $\Psi=0$ at the walls where Ψ is the streamfunction.

The lisiting for calculating the streamfunction is given below:

```
C       CALCULATION OF STREAM FUNCTION
      NIH=NI/2
      NIHP=NI/2+1
      SF(1,J)=0.0
C       SF(I,J) represents the streamfunction.

      SF(NI,J)=0.0
      DO 102 I=2,NIH
      DO 103 J=2,NJM1
      VN(I,J)=0.5*(V(I,J)+V(I,J+1))
C       VN(I,J) represents the non-staggered v-velocity.

      SF(I,J)=VN(I,J)*SEW(I)+SF(I-1,J)
  103 CONTINUE
  102 CONTINUE
      DO 104 I=NIM1,NIHP,-1
      DO 105 J=2,NJM1
      VN(I,J)=0.5*(V(I,J)+V(I,J+1))
      SF(I,J)=VN(I,J)*SEW(I)+SF(I+1,J)
  105 CONTINUE
  104 CONTINUE

      DO 106 I=2,NIM1
      DO 106 J=2,NJM1
  106 SF(I,J)=ABS(SF(I,J))

C       Prints the streamfunction using SUBROUTINE PRINT

      CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,SF,HEDSF)
```

The output data TEA.OUT is printed through SUBROUTINE PRINT. The listing for

SUBROUTINE PRINT is given below.

```
SUBROUTINE PRINT(ISTART,JSTART,NI,NJ,IT,JT,X,Y,PHI,HEAD)
    DIMENSION PHI(80,80),X(80),Y(80),STORE(500)
    CHARACTER*24 F,F4,HI,HY,HEAD
    DATA F/'(3X,A3,I5,10I10,8X,A3)'/
    DATA F4/'1 2 3 4 5 6 7 8 9 10 11'/
C       Values for each variable are printed in groups of 11 columns.  A typical
C       output is shown after the subroutine.
    DATA HI,HY/6H  I = , 6H Y =  /
    ISKIP=1
    JSKIP=1
    ISTA=ISTART-11
  100 ISTA=ISTA+11
    IEND=ISTA+10
    IEND=MIN0(NI,IEND)
    IEL=IEND-ISTA
    INUM=2*IEL-1
    IF(ISTA.EQ.ISTART)THEN
    WRITE(6,115)
    WRITE(6,110)HEAD
    ELSE
    WRITE(6,115)
    ENDIF
    IF(IEL.GT.1) THEN
    F(11:12)=F4(INUM:INUM+1)
    WRITE(6,F)HI,(I,I=ISTA,IEND),HY
    ELSE
    WRITE(6,111)HI,ISTA,HY
    END IF
    WRITE(6,112)
    DO 101 JJ=JSTART,NJ,JSKIP
    J=JSTART+NJ-JJ
    DO 120 I=ISTA,IEND
    A=PHI(I,J)
    IF(ABS(A).LT.1.E-20) A=0.0
  120 STORE(I)=A
  101 WRITE(6,113) J,(STORE(I),I=ISTA,IEND,ISKIP),Y(J)

    WRITE(6,114) (X(I),I=ISTA,IEND,ISKIP)
    IF(IEND.LT.NI)GO TO 100
    RETURN
  110 FORMAT(1H0,20(2H'-),7X,6A6,7X,20(2H'*))
  111 FORMAT(3X,A3,I5,8X,A3)
  112 FORMAT(3H  J)
  113 FORMAT(1H ,I3,1P13E13.4,0PF7.3)
  114 FORMAT(4H0X= ,1P12E13.4)
```

```
 115 FORMAT(///)
     END
```

Typical Output for a 6x6 grid:

The maximum number of columns in one row is 13. The last column represents the *y*-value. Thus for a 6x6 grid, the 1st column represents the counter for J, columns 2 to 7 represent the values of the calculated variable and the 8th column represents the corresponding *y*-value. Similarly the 1st row represents the counter for I, rows 2 to 7 represent the values of the calculated variable and the 8th row represents the corresponding *x*-value.

Output for temperature *T*. The output is non-dimensional. The *x* and *y* directions are non-staggered.

```
0'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-'-
  I =       1          2             3             4             5             6             Y=
 J
  6  1.0000E+00  9.6501E-01  7.6606E-01  3.9682E-01  6.4164E-02  0.0000E+00
1.0000E+00
  5  1.0000E+00  9.6501E-01  7.6606E-01  3.9682E-01  6.4164E-02  0.0000E+00  9.5458E-01
  4  1.0000E+00  9.6278E-01  7.5405E-01  3.9943E-01  6.2888E-02  0.0000E+00  7.0458E-01
  3  1.0000E+00  9.3711E-01  6.0057E-01  2.4596E-01  3.7221E-02  0.0000E+00  2.9542E-01
  2  1.0000E+00  9.3584E-01  6.0318E-01  2.3394E-01  3.4988E-02  0.0000E+00  4.5422E-02
  1  1.0000E+00  9.3584E-01  6.0318E-01  2.3394E-01  3.4988E-02  0.0000E+00
0.0000E+00
X=  0.0000E+00  4.5422E-02   2.9542E-01  7.0458E-01  9.5458E-01  1.0000E+00
```

This output is stored in the data file TEA.OUT.

The graphical output is obtained through the software TECPLOT. Data for this output are stored in TEC.DAT. Before plotting the velocities one has to convert the staggered velocities into non-staggered velocities. This is acheived by carrying out appropriate interpolations for *u* and *v* velocities. Custom outputs like *convergence of a variable at a particular monitoring location with respect to time* can be printed on a data file easily.

18. MAIN PROGRAM

The main program is used to link all the subroutines listed above by using the SIMPLE algorithm (Refer to the flow chart in **Figure 4**). The main program is also used to generate the desired output i.e., numerical or graphical. Following is a listing of the main program.

```
PROGRAM  MAIN
**********************************************************************
C      Natural Convection flow in a square cavity  (in two dimensions)
C      Non-dimensional version, unsteady state calculations
C      Common file= common.h, Input data file= in.dat
**********************************************************************
     CHARACTER*24 HEDU,HEDV,HEDP,HEDT,HEDK,HEDD,HEDM,HEDSF
     INCLUDE 'common.h'
     LOGICAL INCALU,INCALV,INCALP,INPRO,INCALK,INCALD,
    1        INCALB,INCALT,INHY,INCEN,STEADY
     OPEN(6,FILE='TEA.OUT',STATUS='OLD')
C       File TEA.OUT stores the numerical output.

     OPEN(7,FILE='CONV.DAT',STATUS='OLD')
C       File CONV.DAT stores the convergence with respect to time data for unsteady calculations.

     OPEN(10,FILE='DUMP',STATUS='OLD',FORM='UNFORMATTED')
C       File DUMP stores the entire solution field in binary form for later use.

CALL READDATA
C       Reads the subroutine READDATA

C       CALCULATE GEOMETRICAL QUANTITIES AND SET VARIABLES TO ZERO
     CALL INIT
C       Reads the subroutine INIT
     IF(VALUE.EQ.0) THEN
C       If value equals zero, the initial field is read from the DUMP file.
     READ(10)((U(I,J),I=1,NI),J=1,NJ)
     READ(10)((V(I,J),I=1,NI),J=1,NJ)
     READ(10)((P(I,J),I=1,NI),J=1,NJ)
     READ(10)((T(I,J),I=1,NI),J=1,NJ)
     READ(10)((TE(I,J),I=1,NI),J=1,NJ)
     READ(10)((ED(I,J),I=1,NI),J=1,NJ)
     READ(10)((DEN(I,J),I=1,NI),J=1,NJ)
     READ(10)((GAMH(I,J),I=1,NI),J=1,NJ)
     READ(10)((VIS(I,J),I=1,NI),J=1,NJ)
     END IF
     REWIND 10
```

```
      RESORU=0.
      RESORV=0.
      RESORM=0.
      RESORT=0.
      RESORK=0.
      RESORE=0.
C     The resdiual source values are initialised

C     INITIAL OUTPUT
      WRITE(6,210)
      WRITE(6,230) RALI
      WRITE(6,223) PRANDL
      WRITE(6,260) DENSIT
      WRITE(6,250) VISCOS
      WRITE(6,222) TH,TC
      IF(INCALU) CALL PRINT(1,1,NI,NJ,IT,JT,XU,Y,U,HEDU)
      IF(INCALV) CALL PRINT(1,1,NI,NJ,IT,JT,X,YV,V,HEDV)
      IF(INCALP) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,P,HEDP)
      IF(INCALK) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,TE,HEDK)
      IF(INCALD) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,ED,HEDD)
      IF(INCALT) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,T,HEDT)
C     Outputs are printed in the file TEA.OUT.  This corresponds to the field before
C     calculations are started.

C     CALCULATE RESIDUAL SOURCES NORMALIZATION FACTORS

      RESORM=RESORM/((NI-2)*(NJ-2))
      RESORU=RESORU/((NI-3)*(NJ-2))
      RESORV=RESORV/((NI-2)*(NJ-3))
      RESORT=RESORT/((NI-2)*(NJ-2))
      RESORK=RESORK/((NI-2)*(NJ-2))
      RESORE=RESORE/((NI-2)*(NJ-2))

C     BEGIN ITERATION LOOP
      TIME=TFIRST
      DO 3000 ITSTEP=NFTSTP,NLTSTP
      DT(ITSTEP)=TSTEP
      IF(.NOT.STEADY) TIME=TIME+DT(ITSTEP)
C     Time step loop for unsteady calculations

C     INNER ITERATION LOOP
      WRITE(*,310) IMON,JMON

C     Prints the position of the monitoring location on the screen.

      300 NITER=NITER+1
```

```
C       Internal iterations within one time step.


      IF(INCALU) CALL CALCU
      IF(INCALV) CALL CALCV
      IF(INCALP) CALL CALCP
      IF(INCALK) CALL CALCTE
      IF(INCALD) CALL CALCED
      IF(INCALT) CALL CALCT
C       Subroutines are read in to solve the discretized partial differential equations.  Note the
C       sequence in which the subroutines are read in and compare them with the flow chart in
C       Figure 4.
C       UPDATE FLUID PROPERITIES
      IF(INPRO) CALL PROPS
C       Fluid properties are updated.  In case of turbulent flow this means the introduction of
C       turbulent viscosity.  Since the actual fluid properties are constant, INPRO can be taken
C       as FALSE for laminar flow calculations.
C       INTERMEDIATE OUTPUT
       WRITE(*,311) NITER,RESORU,RESORV,RESORM,RESORT,RESORK,RESORE
     1         ,U(IMON,JMON),V(IMON,JMON),P(IMON,JMON),T(IMON,JMON),
     1               TE(IMON,NJM1),ED(IMON,NJM1)

C       These outputs are printed on the screen after every iteration.

      IF(MOD(NITER,INDPRI).NE.0) GO TO 301

C       TERMINATION TESTS
  301 SORCE=AMAX1(RESORM,RESORU,RESORV,RESORT)

C       SORCE is the maximum of mass, momentum and thermal energy resdiuals.

      IF(NITER.EQ.MAXIT) GOTO 302

C       MAXIT represents the maximum number of internal iterations within a time step.

C       For steady calculations, MAXIT represents the maximum number of iterations.

      IF(SORCE.GT.SORMAX) GOTO 300
C       SORCE is compared with the convergence criterion SORMAX.  This procedure is
C       carried out for every time step.


      IF(NITER.LE.20) GOTO 300
```

```
C       The value of residual source sum can be lower than SORMAX at the beginning of the
C       iteration process.  Therefore NITER is allowed to go to a value of atleast 20
C       irrespective of the value of SORCE.   The least value assigned to NITER is arbitrary but
C       generally 20 to 50 iterations are found to be sufficient.


  302 IF(.NOT.STEADY) THEN
C------INTERMEDIATE OUTPUT FOR TRANSIENT CALCULATIONS
     WRITE(*,*)"TIME STEP"
     WRITE(*,407)ITSTEP
C       Printed on the screen after each time iteration.

C       Printed in the output file TEA.OUT
      IF(INCALU) CALL PRINT(1,1,NI,NJ,IT,JT,XU,Y,U,HEDU)
      IF(INCALV) CALL PRINT(1,1,NI,NJ,IT,JT,X,YV,V,HEDV)
      IF(INCALP) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,P,HEDP)
      IF(INCALK) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,TE,HEDK)
      IF(INCALD) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,ED,HEDD)
      IF(INCALT) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,T,HEDT)
      IF(INPRO ) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,VIS,HEDM)

C       Outputs for checking convergence
     DO  3001 J=2,NJM1
     I=(NI+2)/2
     UC(I,J)=ABS(U(I,J))
     UM(J)=U(I,J)
     UMAX=MAXVAL(UM)
C       UMAX represents the maximum of u velocity at the vertical midplane of the cavity.

 3001 CONTINUE
     WRITE(7,406) TIME,UMAX
C       Printed as convergence with respect to time data in the output data file CONV.DAT

C       UPDATE VARIABLES FOR THE NEXT TIME STEP
     IF(INCALU) CALL UPDATE(U,UO,NI,NJ,IT,JT)
     IF(INCALV) CALL UPDATE(V,VO,NI,NJ,IT,JT)
     IF(INCALP) CALL UPDATE(P,PO,NI,NJ,IT,JT)
     IF(INCALK) CALL UPDATE(TE,TEO,NI,NJ,IT,JT)
     IF(INCALD) CALL UPDATE(ED,EDO,NI,NJ,IT,JT)
     IF(INCALT) CALL UPDATE(T,TO,NI,NJ,IT,JT)
     IF(INCALT) CALL UPDATE(DEN,DENO,NI,NJ,IT,JT)
     END IF
 3000 CONTINUE
C       This is the end of time iterations.

C       Printed in the output data file TEA.OUT as final output.
```

```
      IF(INCALU) CALL PRINT(1,1,NI,NJ,IT,JT,XU,Y,U,HEDU)
      IF(INCALV) CALL PRINT(1,1,NI,NJ,IT,JT,X,YV,V,HEDV)
      IF(INCALP) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,P,HEDP)
      IF(INCALK) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,TE,HEDK)
      IF(INCALD) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,ED,HEDD)
      IF(INCALT) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,T,HEDT)
      IF(INPRO ) CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,VIS,HEDM)


CALL DUMP(NI,NJ,U,V,P,T,TE,ED,DEN,GAMH,VIS)

C     The dump file is called and the final field is stored as a binary output for further use.

C     CALCULATION OF STREAM FUNCTION
      NIH=NI/2
      NIHP=NI/2+1
      SF(1,J)=0.0
      SF(NI,J)=0.0
      DO 102 I=2,NIH
      DO 103 J=2,NJM1
      VN(I,J)=0.5*(V(I,J)+V(I,J+1))
      SF(I,J)=VN(I,J)*SEW(I)+SF(I-1,J)
  103 CONTINUE
  102 CONTINUE
      DO 104 I=NIM1,NIHP,-1
      DO 105 J=2,NJM1
      VN(I,J)=0.5*(V(I,J)+V(I,J+1))
      SF(I,J)=VN(I,J)*SEW(I)+SF(I+1,J)
  105 CONTINUE
  104 CONTINUE
      DO 106 I=2,NIM1
      DO 106 J=2,NJM1
  106 SF(I,J)=ABS(SF(I,J))

      CALL PRINT(1,1,NI,NJ,IT,JT,X,Y,SF,HEDSF)

C     CALCULATION OF NUSSELT NUMBER, UMAX AND VMAX
C     UMAX is the maximum of u velocity at the vertical midplane of the cavity.
C     VMAX is the maximum of v velocity at the horizontal midplane of the cavity.
C     Nusselt number is the non-dimensional heat flux from the hot wall and is defined as:
C     Nu = -(dT/dx)_{y=0} H/dT .  The negative sign is included to make the nusslet number
C     positive. The average nusselt number is given by Nu_avg = -∫(dT/dx)_{y=0} dy H/dT .
```

```
      DO 451 I=1,NI
      DO 452 J=1,NJ
      T(I,J)=T(I,J)*DELT+TC
  452 CONTINUE
  451 CONTINUE

      DO 453 I=1,NI
  453 X(I)=X(I)*H
      DX1=X(2)-XU(2)
      DX2=X(3)-X(2)

      DX=DX1+DX2
      WRITE(6,402)

     DO  401  J=2,NJM1
      I=(NI+2)/2
      U(I,J)=ABS(U(I,J))
      UM(J)=U(I,J)
      UMAX=MAXVAL(UM)
      HFLUXN=(T(2,J)*DX**2-T(3,J)*DX1**2-TH*(DX**2-DX1**2))
     1/(DX1*DX**2-DX*DX1**2)
C        The heat flux, HFLUXN, is calculated using a second order Taylor series
C        approximation.
      ANUN=-HFLUXN*H/DELT
      ANUN=ABS(ANUN)
C        ANUN represents the local nusselt number at the hot wall.
      ANUN1=ANUN*SNS(J)
C        SNS(J) represents dy.
      SUMN=SUMN+ANUN1
C        SUMN represents the average nusselt number
      ANUN=ANUN/RALI**0.25
      WRITE(6,403)Y(J),ANUN
  401 CONTINUE
      DO 415 I=2,NIM1
      J=(NJ+2)/2
      V(I,J)=ABS(V(I,J))
      VM(I)=V(I,J)
      VMAX=MAXVAL(VM)
  415 CONTINUE
      WRITE(6,404)
      WRITE(6,405)SUMN,UMAX,VMAX,H

C        OUTPUTS FOR PLOTTING USING TECPLOT
      DO 421 I=1,NI
  421 X(I)=X(I)/H
      WRITE(8,*)'TITLE="TECPLOT PLOTS"'
      WRITE(8,*)'VARIABLES="X" "Y" "T" "U" "V" "P" "SF" "DEN"'
```

```
      WRITE(8,*)'ZONE F=POINT, I=', NI, ', J=', NJ

      DO 502 J=1,NJ
      DO 502 I=1,NI
  502 WRITE(8,503)X(I),Y(J),T(I,J),UN(I,J),VN(I,J),P(I,J),SF(I,J),
     1 DEN(I,J)
C         These outputs are written in the data file TEC.DAT.  TEC.DAT can then be loaded
C         into TECPLOT using the Load data file command.
      STOP



C         FORMAT STATEMENTS
  210 FORMAT(1H0,47X,'TURBULENT FLOW IN A CAVITY '/////)
  222 FORMAT(///1H0,15X,'THERMAL BOUNDARY CONDITIONS ARE - - -'//
     11H0,25X,'SIDE WALL TEMPERATURES = '2(1PE11.3)//
     11H0,25X,'ADIABATIC TOP AND BOTTOM WALLS '//)
  223 FORMAT(1H0,15X,'PRANDTL NUMBER',T60,1H=,3X,1PE11.3)
  230 FORMAT(1H0,15X,'RAYLEIGH NUMBER ',T60,1H=,3X,1PE11.3)
  250 FORMAT(1H0,15X,' LAMINAR VISCOSITY ',T60,1H=,3X,1PE11.3)
  260 FORMAT(1H0,15X,'FLUID DENSITY ',T60,1H=,3X,1PE11.3)
  310   FORMAT(1H0,'ITER   ','I----------ABSOLUTE RESIDUAL SOURCE SUM
     1S----------I   I-------FIELD VALUES AT MONITORING LOCATION','
     2(',I2,',',I2,')','--------I' / 2X,'NO.',3X,'UMOM',6X,'VMOM',6X,'MA
     3SS',6X,'ENER',6X,'TKIN',6X,'DISP',10X,'U',9X,'V',9X,'P',9X,'T',9X,
     4'K',9X,'D'/)
  311   FORMAT(1H ,I8,4X,1P6E10.3,3X,1P6E10.3)
  402 FORMAT(///5X,1HI,7X,5HYV(I),6X,10HS.S.COEFF.,'NUSSELT NO. ',
     25X,'Y(I)')
  403 FORMAT(/5X,1PE11.3,2X,1PE11.3)
  404 FORMAT('AVERAGE NUSSELT NUMBER',5X,'UMAX',5X,'VMAX',5X,'H')
  405 FORMAT(/5X,1PE11.3,4X,1PE11.3,4X,1PE11.3,4X,1PE11.3)
  406 FORMAT(1H ,1PE11.3,4X,1PE11.3)
  407 FORMAT(1H ,I6)
  503 FORMAT(1PE11.3,2X,1PE11.3,2X,1PE11.3,2X,1PE11.3,2X,1PE11.3,
     12X,1PE11.3,2X,1PE11.3)
      END
```

19. REFERENCES

Ampofo, F. and Karayiannis, T. G. (2003) Experimental benchmark data for turbulent natural convection in an air filled square cavity. Int. J. Heat and Mass Transfer, **46**, 3551-3572.

Chien, K. Y., (1982) Predictions of channel and boundary layer flows with a low Reynolds number turbulence model. *AIAA Journal*, **20**, 33-38.

Craft T. J., Gerasimov A. V., Iacovides H. and Launder B. E. (2002) Progress in the Generalization of Wall-Function Treatments, *Int. J. Heat and Fluid Flow*, **23**, No. 2, pp. 148-160.

Gray, D. D. and Giorgini, A. (1976) The validity of the Boussinesq approximation for liquids and gases. *International Journal of Heat and Mass Transfer*, **24**, 125-131.

Harlow, F. H. and Nakayama, P. (1967) Turbulence transport equations. *Physics of Fluids*, **11**, 2323-2332.

Harlow, F. H. and Welch, J. E. (1965) Numerical calculation for time dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids,* **8**, 2182-2189.

Henkes, R. A. W. M. (1990) Natural-Convection Boundary Layers. *Ph.D. thesis, Technical University of Delft,* The Netherlands.

Jones, W. P. and Launder, B. E. (1972) The Prediction of laminarization with a two-equation model of turbulence. *International Journal of Heat and Mass Transfer,* **15**, 301-314.

Patankar, S. V. and Spalding, D. B., (1972) A Calculation Procedure for Heat, Mass and Momentum Transfer in Three Dimensional Parabolic Flows, *International Journal of Heat and Mass Transfer*, **15**, p.1787.

Patankar, S. V. (1980) Numerical Heat Transfer and Fluid Flow. *Hemisphere Publishing Corporation*, Washington.

Wilcox, D. C. (1993) Turbulence Modelling for CFD. *DCW Industries*, La Canada.