Publikation Nr 86/13 ISSN 02809265

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för
Tillämpad termodynamik och strömningslära

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Applied Thermodynamics
and Fluid Mechanics

A GENERAL COMPUTER PROGRAM FOR TRANSIENT, THREE-DIMENSIONAL, TURBULENT, RECIRCULATING FLOWS

by

Lars Davidson and Peter Hedberg

Göteborg December 1986

ABSTRACT

This report gives the main features of the general prediction method for heat mass and momentum transfer embodied in the TEACH3D Computer Program. The program was originally written for steady, two-dimensional, turbulent (or laminar) recirculating flows, by Gosman and Ideriah (1976). It has been extended to cover also transient and three-dimensional flows. We do not wish to make any secret of the fact that most of the material here was taken from Gosman and Ideriah's report. We have on the whole added relevant parts to describe the transient treatment and rewritten the two-dimensional formulations into three-dimensional forms. A few other extensions have also been made.

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Scope of Present Chapter

---

* Definition of objectives of the report
* Indication of Main features of the program
* Circumstances for application of Method
* Examples of numerical predictions
* Structure of the mathematical foundation

---

* This, the first chapter of the report, defines the objectives of the report and also gives a broad guide line to the computer program.
* Some circumstances in which the program can be adapted to predict the physical behavior of the flow fields will be given as well as a demonstration of examples of numerical predictions.
* Successful application of the program requires an understanding of the basic mathematical foundation, the structure of which will also be given in this introductory chapter.

1.2 Objectives

To Convey:

* Main features of the prediction method
* The Structure and use of the computer program

This report has been written with two specific aims:

* Firstly, to convey the essential feature of the general prediction method for heat, mass, and momentum transfer that is embodied in the computer program.
* Secondly, to give a clearer understanding of the structure and the use of the computer program to the intending user.

1.3 Main Features of the Program

---

* Solves conservation equations for heat, mass, momentum, etc. by a finite-difference method
* Uses "primitive" variables, with the velocities and pressure derived from SIMPLE algorithm (Patankar & Spalding, 1972), and all equations solved by line-by-line method of TDMA
* Treatment covers steady, unsteady, turbulent, laminar, 3D, incompressible or compressible flows in a cartesian coordinate system.

---

* The program operates by solving the relevant conservation equations by means of a hybrid finite-difference technique.
* The main hydrodynamic variables used in the program are velocities and pressure ("primitive" variables), and a special procedure, called SIMPLE algorithm, is employed to solve for the velocity and pressure fields, and each equation is solved by a Line-by-Line solution procedure using the Tri- Diagonal Matrix Algorithm (TDMA).
* In its standard form, the program is presented for steady/unsteady, compressible/incompressible, turbulent/laminar, three-dimensional flows, in a cartesian coordinate system.

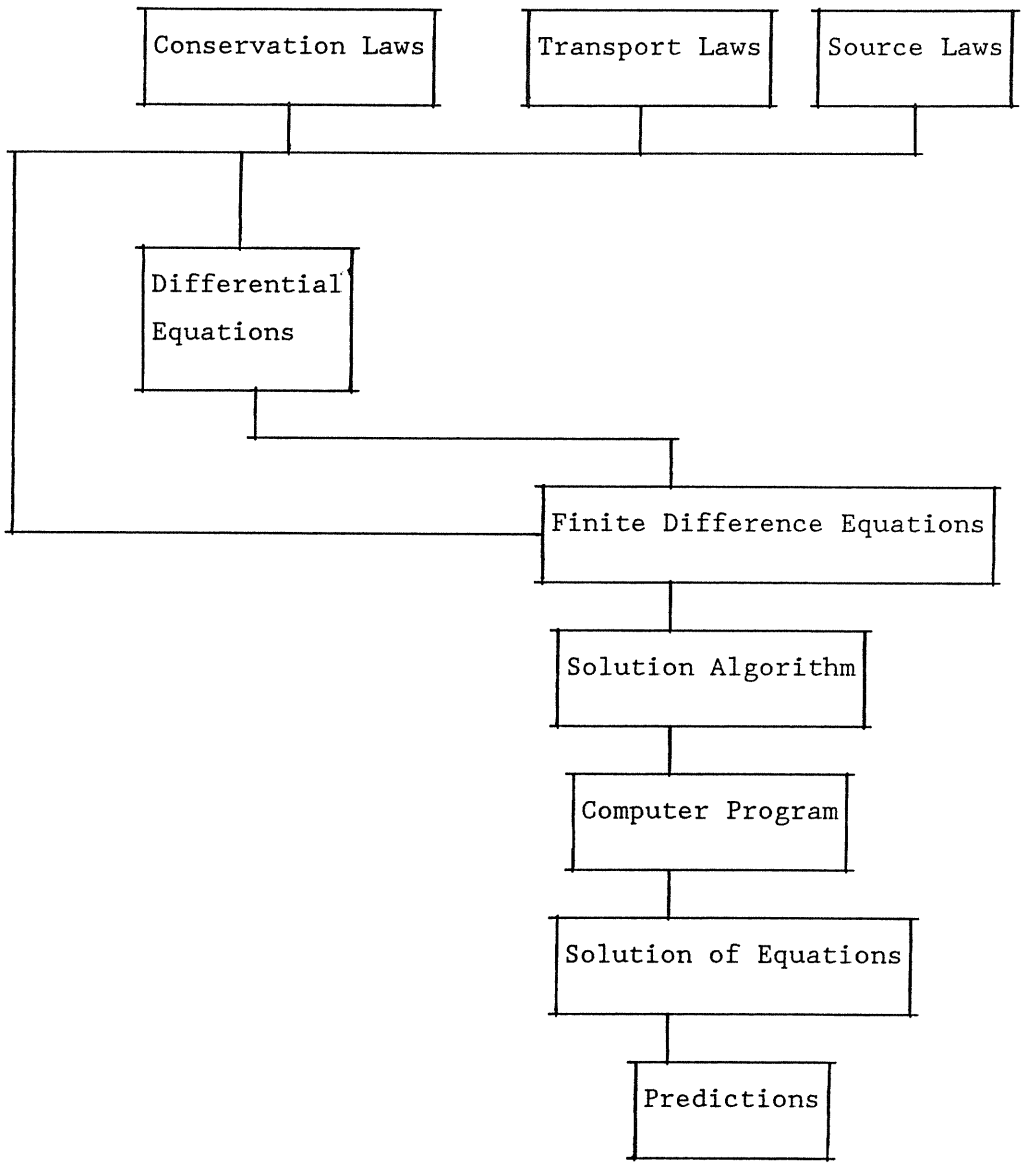
But the possibilities exist of extending this standard form to cylindrical polar coordinate system.

1.4 Circumstances for Application of Method

* Power generation
* Chemical Plants
* Environmental Studies
* Aerospace
* Domestic
* Physiological Studies

* In the field of power generation, the program can be used for the study of various flows in gas turbines, reciprocating engines, furnaces and boilers, and nuclear reactors.
* Chemical plants, like heat exchangers, blast furnaces, packed-reactors, and fluidised beds provide further areas for application.
* Predictions of pollution hazards due to disposal of thermal, chemical, and radioactive effluents into the atmosphere and rivers, as well as the prediction of weather, are areas where the program may be adapted in environmental studies.
* In the field of aerospace, drag and lift, kinetic heating, internal flow in aircraft and rockets are just a few of the circumstances where the program may be used.
* Heating, ventilation and air conditioning of homes and auditoria provide further areas where the program may be employed in domestic life.
* Airflow, and blood flow through veins and arteries are amenable to prediction by the program, in physiological studies.

1.5 Structure of the Mathematical Foundation



* As in most physical theories, the structure of the theory embodied in the code starts from the laws of nature as shown at the top of the boxes: conservation, transport, and source laws.
* These laws of nature may be combined into differential equations which may then be transformed into finite-difference forms.

* On the other hand, as will be shown later, the laws may be transformed directly into finite-difference forms using control-volume analysis. As the number of grid used tends to infinity, correctly formed finite-difference equations approach these of the differential ones.

* To achieve the solutions to the finite-difference equations a solution algorithm (i.e. a systematic set of operations from which values satisfying the equations may be obtained) is necessary. Since such an algorithm will involve thousands of operations, it is embodied in a computer program in order to take advantage of the capability of the computer to perform millions of operations in seconds.

* Thus, as is shown in the boxes, the computer program yields solutions to the set of equations. And, if the natural laws as well as the finite-difference equations have been adequately formulated, the predictions will depict physical reality.

CHAPTER 2 THE CONSERVATION EQUATIONS AND THEIR FINITE DIFFERENCE FORMS

2.1 Purpose and Scope of Chapter

---

1. Purpose : To demonstrate how the laws of nature may be combined into differential equations and their finite-difference forms.

2. Contents :

* Derivation of p.d.e.'s from combination of natural laws
* Derivation of f.d.e.'s from combination of natural laws
* Insertion of boundary conditions

---

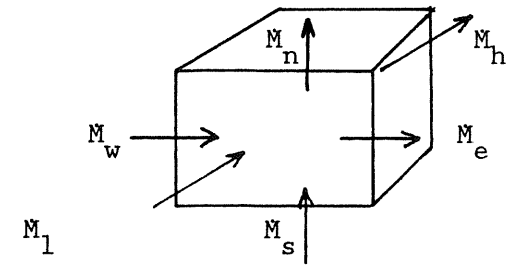
* The aim of this chapter is to demonstrate the manner in which both partial differential equations (p.d.e.'s) and finite difference equations (f.d.e.'s) may be obtained from the natural laws.
* The contents of the chapter therefore include brief illustration of the derivation of the p.d.e.'s from the conservation laws, and the manner in which the f.d.e.'s may also be obtained. In addition, the method of insertion of boundary conditions is also explained.

2.2 Combination of Natural Laws (Typical Example)

Conservation of momentum in direction x :

* Newton's Second Law:
  * Statement: rate of change of momentum plus sum of direction-i momentum flows (M_i) must equal net force in i-th direction (S_i)
  * Mathematical expression:

$$M_{i,t} + M_{i,w} - M_{i,e} + M_{i,s} - M_{i,n} + M_{i,l} - M_{i,h} = -S_i$$



* Change of momentum in time: $M_{x,t} = \frac{\partial}{\partial t}(\rho U)$.

* Transport Law: for Newtonian turbulent fluid, total flux is
  $M_x'' = \rho UU - \mu_t(\partial U/\partial x) + \ldots$. other terms

* Source Law: $S_x = -\frac{\partial p}{\partial x} + S_x'$

* Momentum equation for direction - x :

$$\frac{\partial}{\partial t}(\rho U) + \frac{\partial}{\partial x}(\rho UU) + \frac{\partial}{\partial y}(\rho UV) + \frac{\partial}{\partial z}(\rho UW) - \frac{\partial}{\partial x}(\mu_t \frac{\partial U}{\partial x}) - \frac{\partial}{\partial y}(\mu_t \frac{\partial U}{\partial y}) - \frac{\partial}{\partial z}(\mu_t \frac{\partial U}{\partial z}) =$$

$$-\frac{\partial p}{\partial x} + S_x'$$

---

* Here, we have taken the transport of momentum to illustrate how the laws of nature may be combined into a differential equation.
* The law governing the transport of momentum is the rather familiar Newton's second law of motion which has been expressed in the slide in both verbal and mathematical forms.

* The mathematical expression involves fluxes, M's, which express the rate of change of momentum and the transport of momentum by both convection and diffusion (viscous action). These are derived from "transport (or flux) laws", e.g. Newton's law of viscosity, etc.
* Furthermore, there are additional factors (or sources) which contribute to the transport of momentum. These are derived from "source laws" which cover, for example, contribution due to pressure gradient, buoyancy, etc.
* The differential equation for momentum for other directions, as well as for any other conserved property, may be derived in identical manner.

2.3 General Form of Conservation Equations

---

* The general form[#] of the equations to be solved, for unsteady 3D problems, is:

$$\frac{\partial}{\partial t}(\rho\phi)+\frac{\partial}{\partial x}(\rho U\phi)+\frac{\partial}{\partial y}(\rho V\phi)+\frac{\partial}{\partial z}(\rho W\phi)-\frac{\partial}{\partial x}(\Gamma\frac{\partial\phi}{\partial x})-\frac{\partial}{\partial y}(\Gamma\frac{\partial\phi}{\partial y})-\frac{\partial}{\partial z}(\Gamma\frac{\partial\phi}{\partial z})-S_\phi=0$$

* $\phi \equiv U,V,W,k,\epsilon,T$, etc.

  $\Gamma \equiv \mu_t,\Gamma_k,\Gamma_\epsilon,\Gamma_T$, etc.

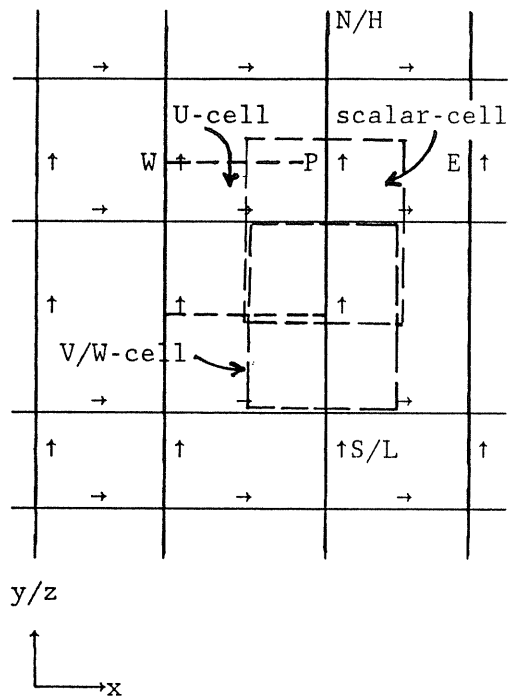
Continuity equation has special form

---

* The beauty of the transport equations for the conservation of various properties (except mass) is that they can all be written in a general form as shown in the panel.
* Here S_ϕ represents the "sources" relevant to the transport of the variable ϕ.
* The continuity or mass conservation equation has special form which will be dealt with later when the "pressure-correction" equation is derived.

2.4.1 Derivation of Finite Difference Equations (FDE's) : (I) The Grid, Storage, Locations, and Control Volumes

* Grid, shown by solid lines, is regular and rectangular with arbitrary spacing (i.e. $\delta x_{PW} \neq \delta x_{EP}$)
* Typical-control volumes or cells shown in dotted lines: each cell surrounds the point of location of the relevant variables.
* Scalar quantities (i.e. p,k,ϵ,T,etc) are located at intersection of grid nodes
* Velocities are located at boundaries of control volumes for scalar quantities



location	Variable stored
*	p,T,k,ϵ
→	U
↑	V,W

* The forms of the differential equations having been introduced, the stage is now set for the derivation of the fde's.
* The first step in deriving the fde's is the establishment of a suitable grid and storage locations of variables. The grid employed,viewed in the x - y/z plane, is regular and rectangular with arbitrary spacing and is shown by solid lines in the panel.
* A typical cluster of U- , V-, W-, and scalar-cells or control volumes is shown in dotted lines. Each cell surrounds the point of location of the relevant variable: note that the variables are stored at different locations of the grid.

* The pressure p and other scalar variables are located at intersection of grid nodes.
* The velocities are located at the boundaries of the scalar cells.
* This "staggered grid" system has the advantage that the variables U, V, and W are easy to evaluate; moreover, the velocities are located where they are needed for the calculation of convective fluxes.

2.4.2 Derivation of FDE's :(II) Expression of Conservation Laws by Control Volume Analysis

---

* For transport of any extensive property $\hat{\phi}$ (i.e. mass, momentum, energy, etc.):
rate of increase of $\hat{\phi}$ in cell = (net rate of inflow of $\hat{\phi}$ to cell by convective fluxes) + (net rate of inflow of $\hat{\phi}$ to cell by diffusive fluxes) + (rate of generation of $\hat{\phi}$ within cell)
* If ϕ is the corresponding intensive property of $\hat{\phi}$, above statement becomes

$$\frac{\Delta(\rho\phi)}{\Delta t}dV + \dot{q}_w - \dot{q}_e + \dot{q}_s - \dot{q}_n + \dot{q}_l - \dot{q}_h + \int_V S_\phi dV = 0$$

where ϕ=U,V,W,T,k,ϵ (or unity for $\hat{\phi}$=mass)

$\dot{q}$= total convective + diffusive fluxes

S_ϕ= generation per unit volume; $\frac{\Delta(\rho\phi)}{\Delta t}dV = 0$ for steady flow

* Next, the $\dot{q}$'s and generation term need to be determined

---

* Here, the basis of the control volume approach in deriving the fde's is shown. This approach is similar to the integral method, but it is more physical in its basis.

* A control volume or cell is pictured in space, and the node-point value of any property ϕ refers to the average over the control volume. The conservation law for the transport of ϕ may then be expressed both verbally and mathematically as shown in the panel. (Note that n,s,w,e,h,l denote cell boundaries)

* The convective and diffusive fluxes $\dot{q}$'s are represented as summation around the cell boundaries, thus aiding physical understanding and emphasizing conservation. These fluxes as well as the generation term next need to be determined, and this is done on the basis of further macroscopic physical laws (entailing some approximation) as is shown in the next three panels. (Note that in this approach no referens is made differential equations).

2.4.3 Derivation of FDE's :(III) "Exact" Convective and Diffusive Flux Expression

---

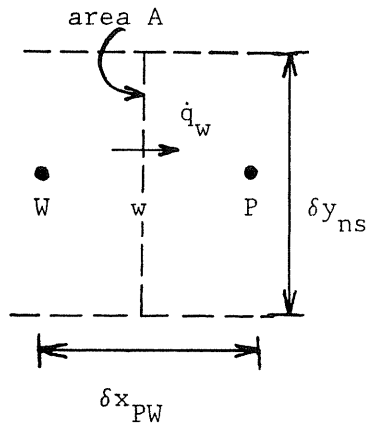
* Consider one-dimensional transport across cell boundaries
* Then, e.g. for the west boundary, the "exact" solution gives:

$$\dot{q}_w \approx \rho_w U_w A_w [f_w \phi_W + (1-f_w)\phi_P]$$

* Here, $f_w \equiv \exp(Pe_w)/[\exp(Pe_w)-1]$

$$Pe_w \equiv \rho_w U_w \delta x_{PW}/\Gamma_w \quad ; \quad A_w = \delta y_{ns} \times \delta z_{hl}$$

$$\rho_w \equiv (\rho_W+\rho_P)/2 \quad ; \quad \Gamma_w = (\Gamma_W+\Gamma_P)/2$$



---

* The convective and diffusive fluxes are derived with the aid of a one-dimensional analysis, since we require that, in 1-D limit, the fluxes should be accurately calculated.
* If transport across cell boundaries is regarded as 1-D, the "exact" solution, e.g. for the west boundary, is as shown in the panel: this gives the convected ϕ as weighted mean, with weighting factor depending on the local Peclet number.
* If ρ and Γ are non-uniform, average values are used in order to ensure continuity of flux.

2.4.4 Derivation of FDE's :(IV) Approximate Convective and Diffusive Flux Expression ("hybrid" Scheme)

---

* Exponential expensive to compute

* Hence "exact" expression approximated, with little loss of accuracy, by employing "central" differencing for $-2<Pe<2$ and "upwind" differencing for $Pe\geq 2$ or $Pe\leq -2$, resulting in

$$\frac{\dot{q}_w}{\rho_w U_w A_w} \equiv \begin{cases} 1/2[(1+2Pe_w^{-1})\phi_W+(1-2Pe_w^{-1})] \text{ , for } -2<Pe_w<2 \\ \phi_W \text{ , for } Pe_w\geq 2 \\ \phi_P \text{ , for } Pe_w\leq 2 \end{cases}$$

* Remaining $\dot{q}$'s are similarly treated

---

* in order to avoid expensive calculation of exponentials, a method using piece-wise linear fit is adopted to approximate the "exact" $\dot{q}_w \sim Pe_w$ relation with little loss of accuracy.

* In this method, a central-difference scheme is employed for low $|Pe_w|$, and an "upwind" difference scheme (asymptotes of upwind formula) is used for large $|Pe_w|$ — hence the method is called "hybrid" scheme

* $\dot{q}_e, \dot{q}_n, \dot{q}_s, \dot{q}_h$, and $\dot{q}_l$ are derived in a similar manner.

2.4.5 Derivation of FDE's :(V) Expression for Generation or Source Term

---

* Express the total generation or source term by a linear relation:

$$-\int_V S_\phi dV = b\phi + c$$

* b and c to be deduced from integrated and linearised form of the source

* Note: b and c will in general be functions of ϕ

---

* The total generation over the control volume cannot be derived without a knowledge of the particular expression of the source S_ϕ.
* However, whatever the form of the particular expression, the total generation can be reduced to a linearized form as shown in the panel. Of course, b and c will then be functions of ϕ in general.
* This approach proves to be advantageous and convenient in the setting up of a single computer program for various flow situations.

2.4.6 Derivation of FDE's :(VI) Assembly of Final Equation

---

* Substitution of the flux and generation expressions into the conservation law given in section 4.3.2 yields, with the aid of continuity:

$$(a_P - b + a_p^o)\phi_P = \sum_n a_n\phi_n + c + a_p^o\phi_P^o$$

* Here, $a_P = \sum_n a_n$; $\sum_n$ = summation over neighbours (H,L,N,S,E,W)

$$a_p^o = \rho_p^o \frac{\Delta x \Delta y \Delta z}{\Delta t}$$

$$a_W = \rho_w U_w A_w f_w$$

$$a_S = \rho_s V_s A_s f_s$$

etc. ^{o} = value at previous timestep

---

* The flux and generation expressions can now be assembled, resulting in a final fde.

* a_h, a_l, a_n, a_s, a_e, and a_w are combined convective/diffusive coefficients

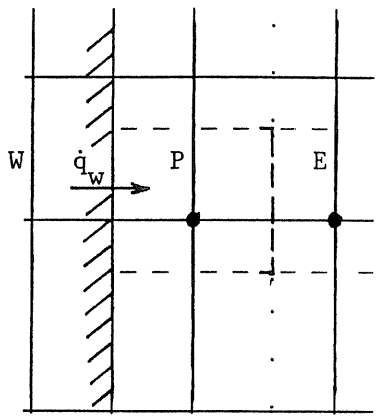
* With the aid of continuity, a_p turns out to be the summation of the combined-flux coefficients over H,L,N,S,E,W : thus, when b=c=0 and the flow is steady, ϕ_P represents weighted mean of neighbours.

2.5.1 Insertion of Boundary Conditions: (I) Outline of Procedure

---

* Grid arranged such that boundaries coincide with control-volume walls.

* Usual flux expression now inappropriate, hence suppress by setting coefficient to zero (e.g. a_W=0) which breaks normal $\phi_P \sim \phi_W$ link



* Correct expression may be conveniently as a "false" source by appropriate specification of b and c (e.g. $\dot{q}_w = b\phi_P + c$, where b and c will be functions of $\phi_{boundary}$)

---

* At the boundaries of the calculation domain, the general fde is not applicable. Hence special measures are required for the cells next to the boundaries.
* The grid arrangement is such that the boundaries coincide with the cell walls. This is advantageous for ensuring conservation and for flux calculation.
* Here, a typical cell whose west wall coincides with the boundary of the calculation domain is shown. There is now no link between ϕ_P and ϕ_W through the general fde. Hence, the usual $\phi_P \sim \phi_W$ link is suppressed by setting the coefficient a_W to zero.
* However, we still need to insert $\dot{q}_w$. Note that for the cell shown, normal expressions for $\dot{q}_e$, $\dot{q}_n$, $\dot{q}_s$, $\dot{q}_h$ and $\dot{q}_l$ are not affected.
* There are several ways of inserting $\dot{q}_w$. But that adopted is a "false" source treatment through specification of b and c. This approach is particularly convenient for programming.

2.5.2 Insertion of Boundary Conditions: (II) Some Examples

---

* Boundary flux, $\dot{q}_B$, prescribed

  Set $a_W=0$; $b=0$; $c=\dot{q}_B$

* Boundary value, ϕ_B, prescribed

  $\dot{q}_w$= may be written as $\dot{q}_w = a'_w(\phi_B - \phi_P)$

  Then, set $a_w=0$; $b = -a'_w$; $c = a'_w\phi_B$

* To fix ϕ_P for any internal node at ϕ_{fix}

  Set $b = -\gamma$; $c = \gamma\phi_{fix}$

  where $\gamma \equiv$ large number, e.g. 10^{30}

---

* Here are shown some typical examples of insertion of $\dot{q}_w$ of the previous panel. Three possible conditions are covered: prescribed boundary flux $\dot{q}_B$, or boundary value ϕ_B, or indeed a fixed value ϕ_{fix} for an internal node remote from boundaries.
* If the boundary flux $\dot{q}_B$ is prescribed, the problem is much simplified as $\dot{q}_w = \dot{q}_B$. Hence b=0 and $c = \dot{q}_B$.
* If the boundary value ϕ_B is prescribed, $\dot{q}_w$ may then be reduced to the linearised form shown, with b and c having the indicated values. If the relevant boundary of the calculation domain is a wall, a'_w will be obtained from wall functions.
* Sometimes (e.g. for flow past an obstruction) ϕ needs to be fixed within the flow field. The "false" source treatment becomes a very useful tool in such cases with the desired condition achieved by setting b and c as shown.

2.6.1 Finite Difference Momentum Equations: (I) Outline of Treatment

---

Development same as for scalar variables apart from:

* Control volumes centred around velocity locations (see section 4.4.1)
* Pressure-gradient purposely separated from remaining sources
* Interpolation practices for cell-boundary velocities, densities, etc. (to ensure continuity of total flux)
* E.g. u-equation pertaining to shaded volume is:

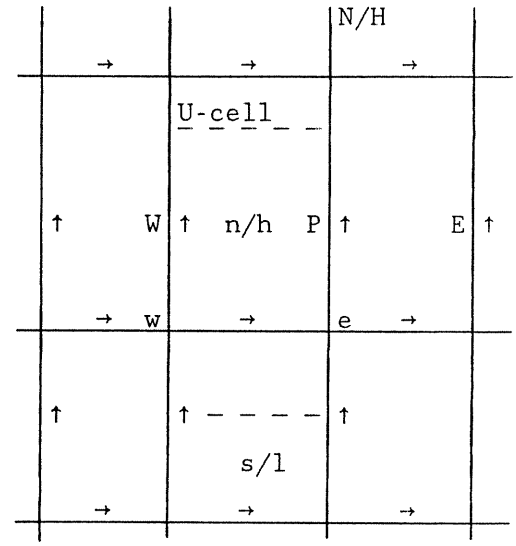
$$(a_P - b)U_P = \sum_n a_n U_n + A_{ew}(p_W - p_P) + c$$

* Here, e.g. $a_w \equiv \rho_w U_w A_{ew} f_w$

$f_w \equiv f_{HD}(Re_w)$; $Re_w = \rho_w U_w \delta x_{PW}/\mu_w$;

$\rho_w U_w = (\rho_W U_W + \rho_P U_P)/2$; $A_{ew} = \delta y_{ns} \delta z_{hl}$

suffix HD = hybrid difference



---

* While the foregoing derivation of the fde's is based on scalar variables, the finite-difference momentum equations are derived in similar manner except that the control volumes are displaced because the velocities are displaced. The convention is otherwise the same.
* However, the pressure gradients are singled out for later attention in the derivation of the "pressure-correction" equation. This is a consequence of the special solution procedure adopted, and it is dealt with in the next chapter.
* The cell-boundary velocities, densities, etc. are interpolated so as to satisfy continuity of total flux.
* Here, an example of the U-momentum equation is given. Note the similarity with panel (2.4.6). f_w is now function of the cell-boundary Reynolds number. Also note how $\rho_w U_w$ is evaluated.

2.6.2 Finite Difference Momentum Equations: (II) Insertion of Boundary Conditions

---

* Velocity tangential to boundary

  * Usual b.c.'s are: prescribed stress, τ_B; prescribed velocity U_B; or prescribed $\tau_B \sim U_B$ relation (i.e. drag law)

  * All may be inserted by manipulation of flux and source coefficients

* Velocity normal to boundary

  Standard practices apply again (see however 3.4.3)

---

* Various boundary conditions may be encountered. Where a tangential velocity is prescribed at the boundary, through flux or drag laws, for example, the appropriate value may be inserted by way of the usual source treatment.

* Velocities normal to the boundary may be prescribed as being fixed, or as a function of the boundary pressure. When the velocity is fixed, the correct value may be inserted through source treatment. However, when the velocity is a function of pressure, reference to the pressure equation is necessary — further comments on this point is made in panel 3.4.3.

2.7 Summary

---

* Use of "primitive" variables and Eulerian co-ordinates
* General form of pde's illustrated: all conservation equations of similar form except that for mass
* Staggered grid system
* Derivation of fde's by control volume analysis with emphasis on accuracy of total flux
* Use of hybrid scheme to procure numerical stability and accuracy
* Manner of insertion of various boundary conditions illustrated

---

* This chapter will have given the reader considerable insight into the manner in which both pde's and fde's may be derived from the physical laws. In setting up the equations, "primitive" variables (U,V,W,p) and Eulerian co-ordinates are employed, and it has been shown that all conservation equations of similar form (except that for mass).

* In the derivation of the fde's, a staggered grid system is used with the advantage that pressure gradients are easy to evaluate and velocities are most conveniently located for calculation of convective fluxes.

* The fde's are directly obtained from the conservation and macroscopic physical laws by a control volume analysis: this promotes correct expression of the conservation laws; also, the concentration on total-flux expressions ensures maximum accuracy for 1-D problems. The approach adopted highlights the fact that the conservation laws need not be expressed first as differential equations before deriving the fde's.

* A hybrid differencing scheme, in which central differencing is used at low |Pe| and upwind differencing at high |Pe|, is employed. This offers good compromise between accuracy and economy, and is numerically stable.

* The general method of inserting boundary conditions through linearized source treatment has also been illustrated.

CHAPTER 3 SOLUTION OF THE FINITE DIFFERENCE EQUATIONS

3.1 Purpose and Scope of Chapter

---

1. Purpose: To describe the general line-by-line iteration procedure of solving all the f.d.e.'s, and the special f.d. SIMPLE procedure for the hydrodynamic variables.

2. Contents:
   * The line-by-line iteration procedure
   * The SIMPLE procedure algorithm for the main hyrdodynamic equations
   * Miscellaneous matters: convergence, under-relaxation, etc.

---

* The aim of this chapter is to describe the method of solving the finite difference equations: this involves the brief description of the line-by-line procedure for all equations, and a special algorithm (called SIMPLE) for the hydrodynamic equations.
* The contents of the chapter include both the line-by-line procedure which employs a Tri-Diagonal Matrix Algorithm, and the SIMPLE algorithm. Some miscellaneous matters, including convergence and accuracy of the solution procedure as well as an under-relaxation method, are also covered.

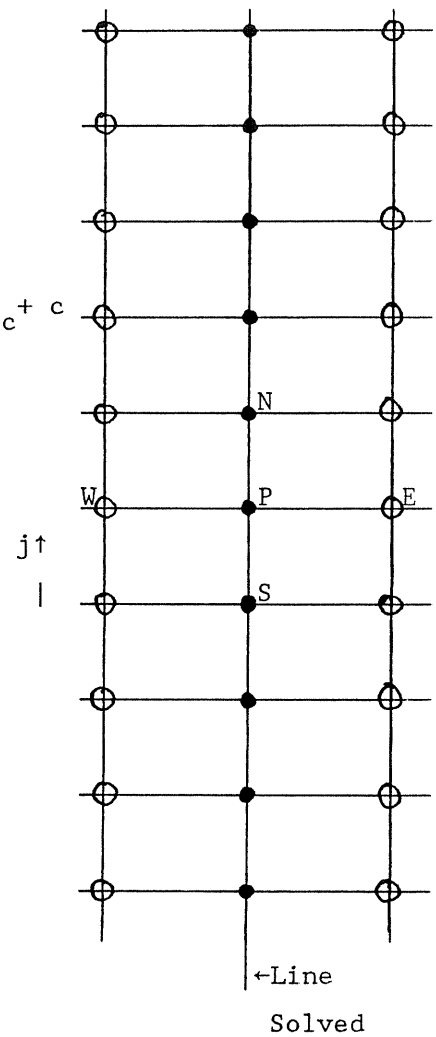
3.2 General Solution Procedure: Line-by-line Iteration (Background)

* Procedure is iterative, line-by-line method
* Re-cast equation for point P as follows:

$$a'_P\phi_P = a_N\phi_N + a_S\phi_S + c'$$

$$c' = a_E\phi_E + a_W\phi_W + a_H\phi_H + a_L\phi_L + a_P^o\phi_{Pc}^o + c$$

(known) and $a'_P = a_P - b + a_P^o$



○ = temporarily known

● = unknown

* Set of equations for points j=2 to j=n on N-S line becomes:

$$
\begin{aligned}
-\beta_2\phi_1 + D_2\phi_2 - \alpha_2\phi_3 &= c'_2 \\
-\beta_3\phi_2 + D_3\phi_3 - \alpha_3\phi_4 &= c'_3 \\
-\beta_j\phi_{j-1} + D_j\phi_j - \alpha_j\phi_{j+1} &= c'_j \\
-\beta_n\phi_{n-1} + D_n\phi_n - \alpha_n\phi_{n+1} &= c'_n
\end{aligned}
$$

where $D \equiv a'_P$, $\alpha \equiv a_N$, $\beta = a_S$; ϕ_1 and ϕ_{n+1} are known.

* Set of equations easily solved by Tri-Diagonal Matrix Algorithm (TDMA)

---

* The general solution employed is an iterative line-by-line (LBL) method: initial guess of values for the flow field is made, and these are improved upon from one line to the other.

* For the solution of the equations for points on each line (e.g. N-S line), values on neighbouring lines are assumed to be temporarily known. The equation for each point on the N-S line then reduce to one where only three values (ϕ_P, ϕ_N and ϕ_S) are unknown.

* The set of equations for all points on the N-S line then take a particularly simple form in which the non-zero coefficient matrix is tri-diagonal. Note that, generally, ϕ_1 and ϕ_{n+1} will be known in our application.

* Equations of this type are especially easy to solve by the Tri-Diagonal Matrix Algorithm (TDMA), the main features of which is subject of the next panel.

3.2.1 General Solution Procedure: Line-by-Line Iteration (TDMA)

---

* Re-arrange jth equation as

$$\phi_j = Q_j\phi_{j+1} + R_j\phi_{j-1} + Z_j$$

where

$$Q_j = \alpha_j/D_j \ , \ R_j = \beta_j/D_j \ , \ Z_j = c'_j/D_j$$

* Then equations become:

$$\phi_2 = Q_2\phi_3 + R_2\phi_1 + Z_2 \qquad \rightarrow \text{(i)}$$

$$\phi_3 = Q_3\phi_4 + R_3\phi_2 + Z_3 \qquad \rightarrow \text{(ii)}$$

$$\phi_4 = Q_4\phi_5 + R_4\phi_3 + Z_4 \qquad \rightarrow \text{(iii)}$$

$$\downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow$$

$$\phi_n = Q_n\phi_{n+1} + R_n\phi_{n-1} + Z_n$$

* ϕ_1 is known. Hence, eliminate ϕ_2 from (ii), and ϕ_3 from (iii), etc., yielding a general formula for ϕ_j:

$$\boxed{\phi_j = A_j\phi_{j+1} + c''_j} \qquad \rightarrow \text{recurrence relation}$$

where

$$A_j = \alpha_j/(D_j - \beta_j A_{j-1}) \ , \ c''_j = (\beta_j c''_{j-1} + c'_j)/(D_j - \beta_j A_{j-1})$$

$\rightarrow$ recurrence formulae

(Note: $A_1 = 0$, and $c''_1 = \phi_1$)

---

* Here, the highlights of the TDMA are illustrated.
* By straightforward algebraic manipulation, the set of equations is converted into one expressible by a general, recurrence relation for ϕ_j as indicated in the panel, with the coefficients A_j and c_j'' obtained from recurrence formulae.
* It is from this general recurrence relation for ϕ_j that all values for ϕ from j=2 to j=n are calculated, and the process is particularly easy as one only needs to evaluate the A's and c''s in order to get the ϕ's. Note that $A_1 = 0$ and $c_1'' = \phi_1$.

3.2.2 General Solution Procedure: Line-by-Line Iteration (Application)

---

* Calculate and store A's and c'''s from j=2 to j=n from recurrence formulae
* Finally, obtain ϕ_j's from general formula, starting with ϕ_n and up to ϕ_2 in that order. (Note: ϕ_1 and ϕ_{n+1} must be known)
* "Traverse" = movement along fixed grid line
  "Sweep" = movement from one grid line to the other
* Apply TDMA along N-S line ("traverse"). Proceed to neighbouring line, using most recently-calculated ϕ's in c'.
Scan through the whole volume.
* Apply TDMA along E-W line ("traverse"). Proceed to neighbouring line, using most recently-calculated ϕ's in c'.
Scan through the whole volume.
* Apply TDMA along H-L line ("traverse"). Proceed to neighbouring line, using most recently-calculated ϕ's in c'.
Scan through the whole volume.
* "Sweep" through grid, line-by-line, and repeatedly until desired solution obtained: many variants possible
* Next timestep

---

* To apply the TDMA to the N-S line, the A's and c'''s are calculated from j=2 to j=n from the recurrence formulae.
* Then, the ϕ_j's are obtained from the general recurrence relation, starting with ϕ_n and ending with ϕ_2 in that order.
* To apply the TDMA to the entire field, the process is started from the N-S line (traverse) at I=2, K=2. Next, it is repeated along successive neighbouring N-S lines, i.e. I=3, I=4 ..... I=NI-1 at K=2, with most recently-calculated ϕ's used in c' (see panel 3.2). Then the plane is switched to K=3 and the process continues at I=2, I=3 ..... The entire grid is "swept" through. After this TDMA is applied along W-E lines starting at I=2, K=2

and the process continues at the neighbouring line at I=3. And in a similar pattern the field is swept through.

Finally TDMA is applied along H-L lines starting at I=2, J=2 and the process continues at I=3.

* The entire grid is swept through, for the three traverse directions, and as many sweeps as necessary may be employed until the desired solution is obtained.

* Do the same procedure for the next timestep until final timestep is reached

3.3 Special Procedure for Main Hydrodynamic Equations: Introduction

---

* Why a Special Procedure?

* Unknown variables are: Main hydrodynamic variables - U,V,W,p
  Additional Scalar variables - k,ϵ,T
* Each unknown requires own equation
* k,ϵ,T satisfy this requirement
* U,V,W also satisfy this requirement (in momentum eqns.)
* p does not (remaining continuity eqn. does not contain p)
* Hence special measure required to obtain p

* The Special Procedure:

* Procedure employed is called SIMPLE (= Semi-Implicit Method for Pressure-Linked Equations) algorithm (Patankar and Spalding, 1972)

---

* The fde's for non-hydrodynamic variables (k,ϵ,T,etc.) can be solved for directly by the TDMA. However, an additional procedure is employed together with the TDMA to solve for the hydrodynamic variables: this panel seeks to satisfy the enquiring mind about why some special, additional measure is needed.

* To solve for a particular ϕ, an equation where that ϕ is the dominant variable is needed. All the non-hydrodynamic variables satisfy this requirement. U, V and W also satisfy this requirement via the momentum equations.

* However, the pressure has no equation of its own. There is an additional equation — the continuity equation — but pressure does not appear in it.

* Some special measure is therefore needed to obtain the pressure. The measure employed is one where the momentum equations are solved by first estimating a pressure field, the obtaining estimates of U,V and W, and finally correcting the pressure field

into conformity with the continuity equation. This procedure is known as SIMPLE, and it is the subject of the next few panels.

3.4.1 The SIMPLE Algorithm: (I) Algebraic Development of Procedure

---

* Definitions

$p = p^* + p'$; $G = G^* + G'$; $U = U^* + U'$ $(G = \rho U)$

* p^* is the "guessed" value of p; p' is correction on guessed value
* G^*, U^* correspond to guessed pressure field p^*, and G', U' are corrections (for nearly-incompressible flow $G' \simeq \rho^* U'$)

* Use linearized flux relation to obtain G''s in terms of p''s:

$$G' = - \frac{\lambda'_w (p'_P - p'_W)}{\delta x_{PW}} \text{ , with } \lambda' = - \rho^* \delta x_{PW} \frac{\partial U^*_w}{\partial (p^*_P - p^*_W)}$$

$$\text{or, } G'_w = \rho^* \frac{\partial U^*_w}{\partial (p^*_W - p^*_P)} (p'_W - p'_P) \qquad {}^*p'_W \xrightarrow{G'_w} {}^*p'_P \quad \delta x_{PW}$$

* From momentum equation (see panel 2.6.1), using * values,

$$D_w \equiv \frac{\partial U^*_w}{\partial (p^*_W - p^*_P)} = \frac{A_{ew}}{(a_P - b)} \text{ ; thus } G'_w = \rho^* D_w (p'_W - p'_P)$$

---

* This panel illustrates the preliminary algebraic manipulation involved in the development of the SIMPLE algorithm.
* First, the field of p is guessed (Starred values), and the momentum equations may then be solved by the LBL method to yeild corresponding values U^*,V^* and W^*. The resulting set of "incorrect" values (p^*,U^*,V^* and W^*) require the imposition of some corrections (primed values: p',U',V' and W'), as defined in the panel.
* Advantage is taken of the staggered grid system in deriving the corrections for the velocities or flux (G') by expressing the G''s

as coefficients λ''s times gradients of p''s. The coefficient λ' is further obtained from linearization of "resistance law" about p_P^*.

* Employing the momentum equations in terms of U^*, V^*, W^* and p^*, the expression for G_w' finally reduces to a simple form, with coefficient D_w in terms of a_P and b.

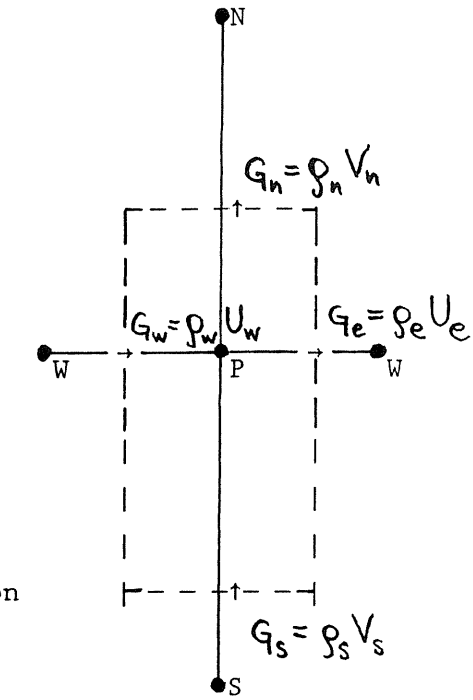
* Note: It is assumed that the fluid is nearly incompressible. If there are appreciable compressibility effects, care is needed in calculating the ρ's.

3.4.2 The SIMPLE Algorithm: (II) Pressure-Correcton Equation

---

* Mass Conservation equation

For a typical main control volume:



$$G_t \delta V + G_e A_e - G_w A_w + G_n A_n - G_s A_s + G_h A_h - G_l A_l = \dot{S}_m''' \delta V$$

where $G_t = (\rho_P - \rho_P^o)/\Delta t$

and $\dot{S}_m'''$ (ideally zero) is generation per unit volume

* Pressure-Correction Equation

Substituting $G_w = G^* + G'$ etc. in mass conservation equation gives:

$$(a_P - b) p_P' = \sum_n a_n p_n' + M_P + c$$

where

$$a_P = \sum_n a_n \quad ; \quad a_w = \rho_w D_w A_w \text{ , etc.}$$

$$M_P \equiv G_w^* A_w - G_e^* A_e + G_s^* A_s - G_n^* A_n + G_l^* A_l - G_h^* A_h - G_t \delta V$$

$\equiv$ residual mass source associated with guessed G^*'s

---

* If the velocities are correct, the mass conservation equation will be satisfied and the mass source $\dot{S}_m'''$ will be zero. However, the starred velocities (U^*, V^* and W^*), and hence G^*, will in

general not satisfy continuity, but will produce a net mass source.

* The major objective, therefore, is to correct the velocities and pressures so as to eliminate the mass source.

* The previous panel has already indicated how U^*, V^*, W^* and G^* as well as U', V', W' or G' may be obtained. Now, substituting $G_w = G^* + G'$, etc. into the mass conservation equation yields a Poisson equation for pressure-correction p'.

* Solution of the p' - equation, also by the LBL method, therefore completes the process, of seeking to obtain the desired set of corrections U', V', W' and p' which are required to make up for U^*, V^*, W^* and p^*.

3.4.3 The SIMPLE Algorithm: (III) Boundary Conditions for Pressure Correction

---

* When normal velocity U_B is prescribed
  * U_B must remain unchanged, hence:
    $U'_w = U_B - U^*_B = D_w(p'_W - p'_P) = 0$
  * Hence set $D_w = 0$ (or, $a_W = 0$ in p'_P)
  * Amounts to prescription of zero normal gradient on p'
  * Boundary pressure therefore not required
* When pressure p_B is prescribed
  * Now $p_W = p_B$; consequently $p'_W = 0$
  * U_w and D_w either obtained in usual way or from special momentum equation, e.g. linearised Bernoulli:
    $U_w = \alpha p_P + B$; hence $U'_w = \alpha p'_P$
  * Insertion by standard procedure

---

* Here, an illustration of some possible boundary conditions are given.

* Where normal velocities are prescribed, no further pressure corrections are necessary. Hence, e.g. for a west boundary, the coefficient D_w must be zero, and this may be achieved by setting $a_W = 0$ in the pressure equation.

* When the boundary pressure is prescribed, e.g. $p_W = p_B$, the pressure correction p'_W must be zero. Then U_w and D_w may be obtained in the usual way, or from special momentum equation: for example, linearization of Bernoulli's equation

($p_{stag} = p_B + 1/2\rho U_w^2$)

* The final form may often be inserted through the usual source treatment or by other more convenient forms.

3.4.4 General, Unified, Solution Procedure: Outline of Steps in Complete Solution Procedure

---

(1) Guess or initialise fields of all variables

(2) Assemble coefficients of momentum equations and solve for U^*, V^* and W^*, by means of LBL procedure, using prevailing pressures:

$$\text{e.g. } (a_P - b)U_P^* = \sum_n a_n U_n^* + A_{ew}(p_W^* - p_P^*) + c$$

(3) Calculated coefficients and mass sources for p' by LBL procedure. Evaluate U''s : e.g. $U_w' = D_w(p_W' - p_P')$

(4) Obtain new values of p,U,V,W from

$$p = p^* + p' \quad ; \quad U = U^* + U' \quad ; \quad V = V^* + V' \quad ; \quad W = W^* + W'$$

(5) Assemble coefficients and solve equations for other variables by means of LBL procedure

(6) Test for convergence: if not attained, use prevailing fields as new guesses and repeat from (2).

(7) Next timestep. Use calculated fields from previous timestep and repeat from (2) until final timestep.

---

* The various stages in the SIMPLE algorithm may now be combined with the solution of equations for the non-hydrodynamic variables in order to form a general, unified, solution procedure.

* The fields of all variables (U,V,W,p,T,k,ϵ,etc.) are guessed or initialised.

* The coeeficients of the momentum equations are assembled, and the improved values U^*, V^*, W^* are then solved for by means of the LBL procedure, using prevailing pressures. More than one "sweep" may be made, but without updating the coefficients. Note: At this point, the momentum equations are satisfied but not the continuity equation.

* The coefficients of the p' - equation are next assembled and p' solved for by the LBL method. Usually more than one sweep may be necessary without updating the coefficients.

* The U''s ,V''s and W''s are next evaluated. Then, new values of p, U, V, and W are obtained from $p = p^* + p'$, etc.

Note: At this point continuity is exactly satisfied, but the momentum equations are no longer satisfied.

* Next, the coefficients of the non-hydrodynamic equations are assembled (one variable at a time), and the relevant ϕ's solved for by means of the LBL procedure. The number of sweeps required without updating the coefficients for each ϕ may depend on the nature of the problem, but 1 to 3 sweeps may often be sufficient.

* A test for convergence is made (see next panel), and if this is not attained, the prevailing fields are used as new guesses and the process repeated is from (2).

* Next Timestep. The prevailing fields are used as old timestep values and as the initially guessed values for the next timestep. The process is repeated from (2) until final timestep is reached.

3.5.1 Miscellaneous Matters

---

(1) Convergence

* All equations satisfy conditions for convergence of matrix (Scarborough, 1966) that

$$|a_P - b| \leq \sum_n |a_n| \quad , \quad \text{provided } b \leq 0$$

* Main convergence test based on "residual sources" (R_ϕ's) defined by: $R_\phi = (a_P - b)\phi_P - \sum_n a_n \phi_n - c$

We require that $\sum |R_\phi| < \lambda R_{\phi,ref}$; λ, $R_{\phi,ref}$ are constants

(2) Under-Relaxation

* Non-linearity of equations necessitates under-relaxation: $\phi_P^R = f\phi_P + (1-f)\phi_P^{old}$;

where f = under-relaxation factor; ϕ_P^R, $\phi_P = \phi_P$ of present iteration with and without under-relaxation;

$\phi_P^{old} = \phi_P$ of previous iteration

---

* All our equations satisfy the conditions under which a successive substitution method can converge (Sometimes called the Scarborough Criterion), since b is usually ≤ 0.

* In the process of the solution procedure, convergence is assessed at the end of each iteration on the basis of the "Residual-Source" criterion which compares the residual sources of each fde with some reference value $R_{\phi,ref}$ (typically the fixed flux of the relevant extensive property fde into the domain of calculation). By so doing, it is ensured that the fde's are solved. (Note: λ is typically of order 10^{-3}).

* By the use of an appropriate relaxation method for an iterative process, convergence may be improved, and, in some instances, divergence may be avoided. Since the equations solved are non-linear, under-relaxation as illustrated in this panel proves to be a very useful tool.

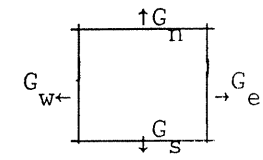
3.5.2 Miscellaneous Matters (Continued)

---

(3) Allowance for mass-flow imbalance

* If mass flows do not satisfy continuity, all a_n's may be zero
* Equations the singular, since $a_P = \sum_n a_n$
* Add "false" linearized source, S_f :

$$S_f \equiv |\dot{m}_{net}|(\phi_P^{old} - \phi_P) = b_f\phi_P + c_f$$



where $\dot{m}_{net} = \sum_n \dot{m}_n$, with $\dot{m}_w = G_w A_w$, etc. ; suffix f = false

* Form of fde solved the becomes

$$(a_P - b + a_p^o - b_f)\phi_P = \sum_n a_n\phi_n + c + a_p^o\phi_p^o + c_f$$

(4) Accuracy

* Function of degree of convergence and grid
* Main source of error is false diffusion, which occurs when Peclet number is large and flow is inclined to mesh

---

* If the mass flows do not satisfy continuity, e.g. as shown in the diagram in this panel, a situation may arise where all a_n's are zero. The fde's may then become singular.
* The solution is to add a "false" source through the linearized source treatment. The form of the final fde then has additional constants b_f and c_f. Note: While this addition of false source makes provision for stabilizing the solution procedure, it has no effect on the final solution.
* The accuracy of the solution procedure will in general be a function of the number of grid nodes employed. For each flow configuration, a grid-independent solution is sought by increasing the number of grid lines until no further changes are observed in the final solution.
* A major source of error at high Peclet number in all finite difference schemes is "false" diffusion which occurs owing to

evaluation of ϕ_P as a weighted mean of surrounding ϕ's. However, its effect may be made considerably small by arranging the grid such that the stream lines are parallel to the mesh, or reducing the cell size which leads to smaller Peclet number.

3.6 Summary

---

* Difference equations solved by LBL iteration procedure
* Using primitive variables, solution of momentum equations similar to that for scalar transport, provided pressure field is available.
* Pressure recovered by "guess and correct" procedure called SIMPLE algorithm which employs a pressure-correction equation based on continuity and momentum equations together with linearized resistance law.
* Major source of error is "false diffusion" which affects all fd schemes at high Pe.

---

* All the fde's are solved by LBL iteration method which employs the TDMA.
* A consequence of using "primitive" variables (U,V,W and p) is the need to obtain the pressure field by some special method. The procedure employed in doing this is the SIMPLE algorithm: this involves a "guess and correct" method, with the pressures obtained by solving a pressure-correction equation whose basis are the continuity and momentum equations coupled with linearized resistance law.
* "False diffusion" poses a major threat to accuracy at high Peclet number for all fd schemes. Solution is to set grid mesh parallel to streamlines, or to make cell Pe's small.

CHAPTER 4 INCORPORATION OF TWO-EQUATION TURBULENCE MODEL

4.1 Purpose and Scope of Present Chapter

---

1. Purpose: to illustrate the incorporation into the numerical procedure of a turbulence model of the two-equation, effective-viscosity variety.

2. Contents:

* Basis of the turbulence model
* Differential equations for steady and transient 3D turbulent flows
* Finite difference forms of the equations and boundary conditions
* Factors influencing stability, accuracy and economy

---

* This chapter seeks to illustrate the incorporation into the solution procedure of a turbulence model in which closure of the time-averaged equations of mean flow is obtained from two turbulence quantities k and ϵ which are derived from their own transport equations.

* The contents of the chapter include a brief introduction to the turbulence model, statements of the differential and finite-difference equations, and the boundary conditions. Also discussed are the factors influencing stability, and economy.

4.2 Two-Equation Model: An Introduction

---

* Instantaneous: $\tilde{U}_i = U_i + u_i$; $\tilde{\phi} = \phi + \varphi$

* Time-averaged Equations of mean motion:

  * $$\frac{\partial \rho}{\partial \tau} + \frac{\partial}{\partial x_i}(\rho U_i) = 0$$

  * $$\frac{\partial}{\partial \tau}(\rho U_i) + \frac{\partial}{\partial x_i}(\rho U_j U_i) = \frac{\partial}{\partial x_j}(-\rho\overline{u_i u_j}) - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_i}\left(\mu \frac{\partial U_i}{\partial x_i}\right) + S_{U_i}$$

  * $$\frac{\partial}{\partial \tau}(\rho\phi) + \frac{\partial}{\partial x_i}(\rho U_i \phi) = \frac{\partial}{\partial x_i}(-\rho\overline{u_i \varphi}) + \frac{\partial}{\partial x_i}\left(\frac{\mu}{\sigma_\phi}\frac{\partial \phi}{\partial x_i}\right) + S_\phi$$

  where μ = laminar viscosity; σ_ϕ = laminar Prandtl number

* Closure: 'gradient transport hypotesis' (Hinze, 1976):

$$-\rho\overline{u_i u_j} = \mu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \; ; \; -\rho\overline{u_i \varphi} = \frac{\mu_t}{\sigma_{\phi,t}}\frac{\partial \phi}{\partial x_i}$$

where μ_t = turbulent viscosity ; $\sigma_{\phi,t}$ = turbulent Prandtl number

  * Dimensional analysis gives $\mu_t = C_\mu \rho k^2/\epsilon$; C_μ = constant at high Reynolds number

  * k and ϵ from their transport equations

---

* This panel illustrate the basis of the turbulence model employed in TEACH3D.

* Using the familiar Reynolds method, the instantaneous values $(\tilde{U}_i, \tilde{\phi})$ and fluctuating components (u_i, φ).

* Using cartesian tensor notation the time-averaged equations for continuity, momentum and scalar transport take the form given in the panel. Unfortunately, these equations contain unknown Reynolds stresses $\rho\overline{u_i u_j}$ and scalar fluxes $\rho\overline{u_i \varphi}$: these turbulent diffusional

fluxes play a significant role in determinating the flow behaviour as the fine-scale effects are expressed through them.

* The method employed in obtaining closure is an 'effective viscosity' model in which the unknown turbulent diffusional fluxes are expressed by means of 'gradient transport hypotesis' wherein the fluxes are assumed proportional to the gradient of mean flow properties. The constant of proportionality is μ_t or $\mu_t/\sigma_{\phi,t}$.

* $\sigma_{\phi,t}$ is often assumed known, and from dimensional analysis, μ_t turns out to be a function of the turbulent kinetic energy k and its dissipation rate ϵ. k and ϵ are derived from their own transport equations: hence the name 2-equation model. Note: the form of μ_t shown in the panel is for high Reynolds number.

4.3 The Equations of Mean Motion in 3D

---

* Continuity: $\frac{\partial \rho}{\partial \tau} + \frac{\partial}{\partial x}(\rho U) + \frac{\partial}{\partial y}(\rho V) + \frac{\partial}{\partial z}(\rho W) = 0$

* Momentum, x-direction:

$$\frac{\partial}{\partial \tau}(\rho U) + \frac{\partial}{\partial x}(\rho UU) + \frac{\partial}{\partial y}(\rho UV) + \frac{\partial}{\partial z}(\rho UW) =$$

$$- \frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu_{eff} \frac{\partial U}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu_{eff} \frac{\partial U}{\partial y}\right) + \frac{\partial}{\partial z}\left(\mu_{eff} \frac{\partial U}{\partial z}\right) + S_U$$

* Scalar transport (e.g. ϕ = T, k etc.)

$$\frac{\partial}{\partial \tau}(\rho \phi) + \frac{\partial}{\partial x}(\rho U \phi) + \frac{\partial}{\partial y}(\rho V \phi) + \frac{\partial}{\partial z}(\rho W \phi) =$$

$$+ \frac{\partial}{\partial x}\left(\Gamma_{eff} \frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma_{eff} \frac{\partial \phi}{\partial y}\right) + \frac{\partial}{\partial z}\left(\Gamma_{eff} \frac{\partial \phi}{\partial z}\right) + S_\phi$$

* Auxiliary: S_{U_i} is given by

$$S_{U_i} \equiv \frac{\partial}{\partial x_j}\left(\mu_{eff} \frac{\partial U_j}{\partial x_i}\right)$$

* $\mu_{eff} = \mu + \mu_t$; $\Gamma_{eff} = \mu/\sigma_\phi + \mu_t/\sigma_{\phi,t}$

---

* It may be useful to recast the time-averaged equations for continuity, momentum (x-direction only) and scalar transport for 3D, transient turbulent flows to which TEACH3D adresses itself.

* Here, μ_{eff} and Γ_{eff} are the effective exchange coeffients which represent summation of both laminar and turbulent transport effects. At high Reynolds number (i.e. fully turbulent flows), to which the k-ϵ model is restricted, the molecular transport effects μ and Γ are comparatively neglible.

* The source S_{U_i} covers additional terms associated with non-uniform viscosity. Their influence is generally small except where

changes in fluid property have considerably effects. The source S_{U_i} is deactivated in TEACH3D; it is, however, easily activated by replacing the 'C' comment characters in column 1 of each line with blanks in the subroutines CALCU, CALCV and CALCW.

4.4 The k and ϵ Equations

---

* Turbulence Energy k:

$$\frac{\partial}{\partial\tau}(\rho k) + \frac{\partial}{\partial x}(\rho U k) + \frac{\partial}{\partial y}(\rho V k) + \frac{\partial}{\partial z}(\rho W k) = G - C_D \rho \epsilon$$

$$+ \frac{\partial}{\partial x}\left(\mu_{eff}/\sigma_k \frac{\partial k}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu_{eff}/\sigma_k \frac{\partial k}{\partial y}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}/\sigma_k \frac{\partial k}{\partial z}\right)$$

* Energy dissipation Rate, ϵ:

$$\frac{\partial}{\partial\tau}(\rho \epsilon) + \frac{\partial}{\partial x}(\rho U \epsilon) + \frac{\partial}{\partial y}(\rho V \epsilon) + \frac{\partial}{\partial z}(\rho W \epsilon) = C_1 \frac{\epsilon}{k} G - C_2 \rho \epsilon^2/k$$

$$+ \frac{\partial}{\partial x}\left(\mu_{eff}/\sigma_\epsilon \frac{\partial \epsilon}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu_{eff}/\sigma_\epsilon \frac{\partial \epsilon}{\partial y}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}/\sigma_\epsilon \frac{\partial \epsilon}{\partial z}\right)$$

where

$$G=\mu_t\{\, 2\left[\left(\frac{\partial U}{\partial x}\right)^2 + \left(\frac{\partial V}{\partial y}\right)^2 + \left(\frac{\partial W}{\partial z}\right)^2\right] + \left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x}\right)^2 + \left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right)^2$$

$$+ \left(\frac{\partial V}{\partial z} + \frac{\partial W}{\partial y}\right)^2 \,\}$$

* C's are constants at high Reynolds number.
* Constants (Launder and Spalding, 1974):
  $C_\mu = 0.09$; $C_D = 1.0$; $C_1 = 1.44$; $C_2 = 1.92$; $\sigma_k = 1.0$; $\sigma_\epsilon = 1.3$

  which correspond to the FORTRAN symbols CMU, CD, C1, C2, PRTE and PRED.

---

* This panel shows the forms of the required additional k- and ϵ-equations. Note the similarity between these equations and those of the mean flow (panel 4.3) when the substitutions $S_k = G - C_D \rho \epsilon$ and $S_\epsilon = C_1 \frac{\epsilon}{k} G - C_2 \rho \frac{\epsilon^2}{k}$ are made.

* G represents generation of k from the mean flow by turbulent shear stress and, to be precise, ϵ is the rate of viscous dissipation of k to heat by the smallest turbulent eddies.

* The C's and σ's are generally emperical functions, but they turn out to be constants for high Reynolds number flows (note corresponding FORTRAN symbols).

4.5 Finite Difference Equations for Turbulent Flows

---

1. Equations of Mean Motion: U, V, W and ϕ as for lamimar flows apart from

   * replacement of μ's by μ_{eff}, and Γ's by Γ_{eff}

   * insertion of additional source terms in momentum equation (which have to be set explicitely by the user, see panel 4.3)

2. Turbulent Equations: treatment of k and ϵ as for ϕ's, with S_k and S_ϵ as follows

$$* \int S_k dV = b\, k_p + C, \text{ with } b \equiv - \frac{C_\mu C_D \rho^2 k_P^* \delta V}{\mu_t^*} , \quad C \equiv G\, \delta V$$

$$* \int S_\epsilon dV = b\, \epsilon_P + C, \text{ with } b \equiv - \frac{C_2 \rho\, \epsilon_P^* \delta V}{k_P^*} , \quad C \equiv \frac{C_1 \epsilon_P^* G\, \delta V}{k_P^*}$$

where

* δV = cell volume; k_P^* , ϵ_P^* and μ_t^* = previous values

$$* \; G = \mu_t \{ 2 \left[\left(\frac{U_e - U_w}{\delta x_{ew}}\right)^2 + \left(\frac{V_n - V_s}{\delta y_{ns}}\right)^2 + \ldots . \right] + \ldots \}$$

---

* Well, one might ask if our finite difference procedure developed in Chapter 2 and 3 do change for turbulent flows. The answer is, of course, no.

* For the equations of mean flow, all that needs to be done is to replace μ's by μ_{eff} ans Γ's by Γ_{eff}, and to insert any additional source terms by means of the linearized source treatment.

* Indeed, the k and ϵ equations are not even a shade different from the other scalar-transport equations (see panel 4.3) if we note that S_k and S_ϵ (panel 4.4) are just additional source terms

which are conveniently introduced through the normal source treatment.

* A point to note about the treatment of S_k and S_ϵ is that in either case b is negative which promotes stability.

4.6 Boundary Conditions: General Remarks

---

1. At Inlets: distributions of U, V, W, ϕ, k and ϵ are specified (or estimated, e.g. $\epsilon = \text{const } k^{3/2}/L$)

2. At Outlets: specification is normally unimportant (at large Reynolds number); usual practice is to set normal gradients to zero, and to obtain exit velocities from mass balance.

3. At Walls: special formulae (wall functions) and other modifications necassary due to steep variation in properties, and inadequacy of the turbulence model at low Reynolds number.

---

* Having got to grips with the equations of turbulent transport, both in the finite-difference forms and otherwise, we now turn our attention to the boundary conditions.

* At inlets to computation domains, the distribution of all variables is specified or estimated. More often than not, ϵ will be estimated from dimensional analysis based on the fact that turbulence is characterized by its energy k and a length scale L representing the size of the energy containing eddies. The other variables can often be prescribed from knowledge of the particular flow situation.

* At outlets of the computation domains, and at large Reynolds number, upwind difference renders specification of variables unimportant. The usual practice is to set normal gradients to zero, and to obtain exit velocities from mass balance.

* Near walls, the local Reynolds number becomes very small and our turbulence model, which is designed for high Reynolds number, becomes inadequate. Both this fact and the steep variation of properties near walls necessitate special attention for grid nodes

close to walls. This rather important aspect forms the basis of the next few panels.

4.7.1 Basis of Wall Boundary Conditions: i) Equations of Mean Motion

---

1. General: one-dimensional, constant shear-stress and heat transfer layer

2. Momentum Equations: $\tau = (\mu + \mu_t) \frac{\partial U}{\partial y}$,or $\frac{\tau}{\tau_w} = (1 + \frac{\mu_t}{\mu}) \frac{\partial U^+}{\partial y^+}$

   * For $y^+ \leq 11.63$, $\mu_t / \mu << 1$, $\tau \simeq \tau_w$; thus $U^+ = y^+$

   * For $y^+ > 11.63$, $\mu_t / \mu >> 1$, $\tau \simeq \tau_w$; thus $U^+ = \kappa\ y\ U_\tau$

   (see Hinze, 1976)

   thus $U^+ = \frac{1}{\kappa} \log_e(y^+) + \text{const} \equiv \frac{1}{\kappa} \log_e(E\ y^+)$

   where $y^+ = \frac{U_\tau y}{\nu}$; $U^+ = \frac{U}{U_\tau}$; $U_\tau = \sqrt{(\tau_w/\rho)}$; κ = von Karman constant (=0.435) and E = func{roughness, shear stress variation} (=9.0).

3. Wall Shear Stress: $U_\tau = \left[\frac{\kappa\ V}{\log_e(Ey^+)} \right]$

   where V= resultant velocity paralell to the wall

---

* This and the next panel give the background to our wall treatment for the the equations of mean flow. Generally, close to the walls, a 1-D couette flow analysis is made. The layer is assumed to be one of constant shear stress ($\tau = \tau_w$) and constant heat flux

($q = q_w$): these conditions are, however, true only for an impermeable wall, with zero or neglible stream-wise pressure gradient (i.e. $\tau_w/|dP/dx| >> y$)

* The momentum equation can then be reduced to a particularly simple non-dimensional form as shown in the panel.

* The region close to the wall is one where the local Reynolds number changes considerably, and the approach adopted is dependent upon the value of the local Reynolds number, y^+, based on the distance from the wall and the friction velocity, U_τ.

* The wall region is made up of three zones (Hinze, 1976): the viscous sublayer ($0 < y^+ < 5$) where viscous effects dominate, the inertial sublayer where the flow is assumed to be completly turbulent but $\tau \simeq \tau_w$, and the transition (or 'buffer') zone ($5 < y^+ < 30$) of vigorous turbulence dynamics where the flow is neither completly dominated by viscous effects nor completly turbulent. Our approach, as is indeed done in many engineering calculation, is to dispose of the 'buffer' layer by defining a point $y^+ = 11.63$ (where the linear velocity profile in the viscous sublayer meets the logarithmic velocity profile in the inertial sublayer) below which the flow is assumed to be purely viscous and above which it is purely turbulent.

* The forms to which the momentum equation finally reduces for $y^+ \leq 11.63$ and $y^+ > 11.63$ are rather familiar. Note that κ is the von Karman constant, and E is integration constant that depends on the magnitude of the variation of shear stress across the layer and on the roughness of the wall. The value of E given in the panel is for smooth walls with constant shear stress.

* Effects of mass transfer across the layer (suction or transpiration), and severe pressure gradients may be incorporated by modifying E which will then no longer be constants.

* The wall shear stress is obtained from the log-law of the wall from which both k and ϵ is calculated (see i.e. Rodi, 1980). It may be noted that this type of wall function is different from that commonly used [also in TEACH-T; see Gosman and Ideriah (1976)] where the friction velocity U_τ is calculated from a, near

the wall, modified k-equation. The advantage of using the present type of wall function is that it can be used in connection with zero- and one-equation turbulence models.

4.7.2 Basis of Wall Boundary Conditions: ii) Equations of Mean Motion (continued)

---

4. Scalar Transport (e.g. ϕ=T): $q = (\Gamma + \Gamma_t)\, c_p \frac{dT}{dy}$, or

$$\frac{q}{q_w} = \left(\frac{\Gamma}{\mu} + \frac{\Gamma_t}{\mu_t}\right) \frac{dT^+}{dy^+}$$

* For $y^+ \leq 11.63$, $\Gamma >> \Gamma_t$, $q \simeq q_w$; thus $T^+ = \sigma_\phi\, y^+$

* For $y^+ > 11.63$, $\Gamma << \Gamma_t$, $q \simeq q_w$, $\Gamma_t/\rho = \nu_t/\sigma_{\phi,t} \simeq \kappa\, y\, U_\tau/\sigma_{\phi,t}$;

$$\text{thus } T^+ = \frac{\sigma_{\phi,t}}{\kappa} \log_e(y^+) + C_T(\sigma_\phi) \equiv \sigma_{\phi,t}\left[U^+ + P\left(\frac{\sigma_\phi}{\sigma_{\phi,t}}\right)\right]$$

where * $T^+ = \rho\, U_\tau\, c_p (T_w - T)/q_w$; $\sigma_\phi = c_p \mu/\lambda$;

$\sigma_{\phi,t}$ = turbulent Prandtl number

* $P(r) = 9.24(r^{3/4} - 1)[1 + 0.28 \exp(-0.007/r)]$

where $r = \frac{\sigma_\phi}{\sigma_{\phi,t}}$ (see Jayatillika, 1969)

---

* It is also of main engineering interest to be able to predict heat-transfer characteristics of walls. The same treatment goes for heat (or other scalar) transport as for the momentum transport illustrated in the last panel: the corresponding non-dimensional T-equation is shown in the panel.

* Note that constant heat-flux across the layer is assumed. Following similar lines as before, the distinguishing wall Reynolds number y^+ is 11.63. For $y^+ \leq 11.63$ transport is assumed to

be due solely to molecular activity, and the expression for the heat flux parameter T^+ is a simple one.

* For $y^+ > 11.63$, transport is assumed to be due entirely to turbulence. The heat flux parameter T^+ becomes a logarithmic function of y^+ with the constant of integration C_T expressible as a P-function. There are many forms of the P-function, but the particular form employed (shown in the panel) is due to Jayatillaka (1969) and is valid only for impermable, smooth walls.

* For rough, impermeable walls, Jayatillaka also indicated how E and the P-function may be made as functions of the 'roughness height'. However, there is yet no information on how E and the P-function may be modified to take account of simultaneous effects of mass transfer, roughness and pressure gradient.

4.8 Basis of Wall Boundary Conditions: iii) Turbulence Energy and Dissipation

---

1. General: 1-D, constant shear stress, 'equilibrium' layer (Inertial Sublayer)

2. k-equation: production ≃ dissipation ;thus $-\overline{uv}\,\frac{dU}{dy} = \epsilon$, yielding

* $k = \frac{\tau_w}{\rho\, C_\mu^{1/2}}$

* $\epsilon = U_\tau^{\,3}/(\kappa y)$

3. ϵ-equation: * Reduces to $C_1 = C_2 - \kappa^2/(\sigma_\epsilon\, C_\mu^{\,1/2})$

* Thus $\sigma_\epsilon = \kappa^2/(C_2 - C_1)/C_\mu^{\,1/2}$

---

* In this panel, the basis of the wall treatment for k- and ϵ-equations is presented. Again, the approach is based on 1-D, constant shear-stress couette layer.

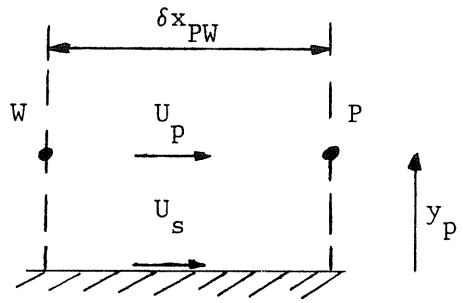
* The approach adopted is strictly valid only for the inertial sublayer where the flow is assumed to be completely turbulent, approximately $30<y^+<100$ (see Rodi, 1980). In this region, the local rate of production of turbulence is balanced by the viscous dissipation rate ϵ. This local equilibrium forms the main basis for our wall treatment.

* The ϵ-equation reduces to a form which indicates modification of σ_ϵ for this region.

4.9 Incorproration of Wall Boundary Conditions: i) Momentum Equations.

---

1. Tangential Velocity:



* U_p from usual momentum balance, but with usual shear force (F_s) expression supressed by setting $a_s = 0$; then

  * For P within the turbulent region ($y^+ > 11.63$):

$$F_s = \tau_s \delta x_{PW} \delta_{hl} = - \rho U_\tau^2 \delta x_{PW} \delta_{hl}$$

  * For P within viscous sublayer ($y^+ \leq 11.63$):

$$F_s = \tau_s \delta x_{PW} \delta_{hl} = -\mu U_p \delta x_{PW} \delta_{hl} / y_p$$

* F_s incorporated through source coefficients b and C

2. Normal Velocity

* No special practice required (nor for pressure)

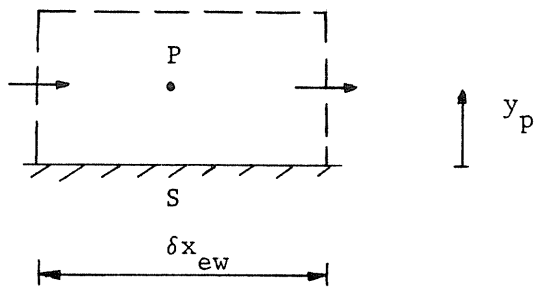
---

* Attention is now directed towards the incorporation of the wall boundary conditions dealt with in the last few panels.
* A tangential velocity U_p for a node next to a wall is obtained from usual momentum balance. For the configuration shown in the panel, the usual expression for a_s is no longer valid; therefore,

it is set to zero. The correct shear force expression is now inserted from $U^+ \sim y^+$ relations of panel 4.7. The incorporation is made via the source treatment.

* For velocities normal to a wall, no special wall treatment is necessary.

4.10 Incorporation of Wall Boundary Conditions: ii) Passive Scalar Property (e.g. T)



* T_P from the usual energy balance, but usual flux (Q) expression supressed by setting $a_S=0$; then

* For P within turbulent region ($y^+ > 11.63$):

$$Q_S = q\ \delta x_{ew}\delta_{hl} = -\ \rho U_\tau (T_P - T_S)\ \delta x_{ew}\delta_{hl}/T^+$$

where $T^+ = \sigma_{\phi,t}[\ U^+ + P(\sigma_\phi/\sigma_{\phi,t})]$

* For P within viscous sublayer ($y^+ \leq 11.63$):

$$Q_S = q\ \delta_{ew}\delta_{hl} = -\ \frac{\mu}{\sigma_\phi}(T_P - T_S)\ \delta x_{ew}\delta_{hl}/y_P$$

* Q_S incorporated through source coeffients b and C

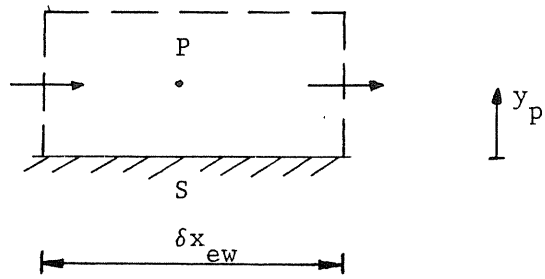
---

* The incorporation of scalar wall boundary conditions follows similar lines as for the momentum case: the usual flux expression a_S is supressed and the scalar flux Q contributed by the wall is derived from the $T^+ \sim y^+$ relation of panel 4.7

* Again, the incorporation is made by way of the source treatment

4.11 Incorporation of Wall Boundary Conditions: iii) Turbulence Equations

---



* k_P not from usual k-balance, but fixed according to experimental data for a flat-plate boundary layer:

$$k_P = c_\mu^{-1/2} U_\tau^2$$

* Effective replacement by overwriting through source coefficients, i.e. $b=-10^{30}$; $C=k_P 10^{30}$

* ϵ_P not from usual ϵ-balance, but fixed by equilibrium relations:

$$\epsilon_P = U_\tau^3/(\kappa\, y_p)$$

* Effective replacement by overwriting through source coefficients, i.e. $b=-10^{30}$; $C=\epsilon_P 10^{30}$

* U_τ obtained from the log-law of the wall

---

* The turbulent energy k is fixed according to experimental data where it is found that k/U_τ^2 is constant in a inertial sublayer. k is fixed using the usual source treatment.

* The expression for ϵ_P is obtained from a simpified form of the k-equation for a boundary layer (production=dissipation), $U_\tau^2 = -\overline{uv}$

and the log-law of the wall. ϵ is also fixed using the source treatment.

* The friction velocity is obtained from the log-low of the wall (see panel 4.7).

4.12 Numerical Stability (Convergence)

Instabilities may be provoked by:

* Bad specification of initial fields.

Cure: improve, or under-relax.

* Inappropriate choice of under-relaxtion factors.

Cure: adjust and re-run.

* Incomplete solution of equations during iteration.

Cure: increase number of application of line-iteration procedure.

---

* Numerical instabilities may be encountered for many complex flows. It requires much computational art to achive convergence for such complex flows. However, there are some three major cases of instability for even simple flows:

* Firstly, a bad specification of the initial field may result in instability. This case of instability may be eliminated by improving the initial field or by using improved under-relaxation factors.

* Secondly, an inappropriate choice of under-relaxation factors may in itself be a source of instability. The cure in such cases is to adjust the relaxation factors.

* Thirdly, an incomplete solution of the finite difference equations during iteration may also provoke instability. The P'-equation is the most sensitive in this case because for each iteration the starting P'-field is zero. Increasing the number of application of the LBL method eliminates this instability.

4.13 Accuracy

Accuracy is influenced by:

* Degree to which solution satisfies fde's.

To asses: examine residual sources.

* Degree to which solution satisfies the pde's.

To asses: refine grid.

* Location of, and conditions imposed at, boundaries.

To asses: adjust conditions and locations.

* Adequacy of turbulence model.

To asses: compare with experiments.

---

* Many factors contribute towards inaccurate solutions as shown in the panel.

* Accuracy is influenced by the extent to which the solution satisfies the fde's. The degree of satisfaction is mirrored by the residual sources. For a satisfactory solution, the residual sources should be of the order 10^{-3} of the reference values (see panel 3.5.1)

* The degree to which the solution of the fde's satisfies the partial differential equations also strongly influences the accuracy of the solution. Generally, smaller and smaller grid size should be used until grid-independent solution is obtained.

* In addition, both the conditions imposed at boundaries and the location of the boundaries may affect accuracy of the solution. Improvement can be obtained by adjusting the conditions and/or the location.

* Finally, given a fully converged, grid-independent solution based on satisfactory boundary conditions and location, how very well the predictions reflect reality as compared with experiments depends on the adequacy of the turbulence model. Indeed, in some complex flows, inadequacy of the turbulence model may in itself

also provoke severe instability. Note: it is important to distinguish between computational errors/instability (which can be eliminated) from those due to physical modelling.

4.14 Economy

Demands on computing time and storage may be minimized by:

* Good specification of initial fields (e.g. values from previous calculation)
* Optimisation of grid: concentrate nodes in regions of steep gradients and reduce elsewhere; minimize extent of calculation domain.
* Optimisation of under-relaxation factors (trial and error)
* Realistic specification of convergence criterion (about 1% on residual sources)

---

* Computer resources regarding computing time and storage are limited, and measures should always be taken to minimize them. This panel shows some of the major techniques in doing this.
* Good specification of initial field, e.g. starting from previous calculation, reduces the computing time considerably.
* In addition, the grid should be arranged so that the nodes are concentrated in areas where steep gradients occur, and reduced where the gradient are nearly uniform. Reduction in the size of the computing domain also proves very useful for the grid-economy.
* Also, by trial-and-error, the under-relaxation factors may be improved upon in order to achive faster convergence.
* Finally, while convergence criteria in general should be based on the residual sources being about 0.1 % of the reference values, it should be noted (especially for complex flows) that about 1% may suffice.

4.15 Summary

---

* Two-equation viscosity model used, employing transport equations for k and ϵ
* Main step required are:
    * derivation of fde's for k and ϵ
    * linearization of sources in appropriate fashion
    * use of wall functions to calculate shear stresses, heat flux, etc near walls
    * solution of fde's as usual, with under-relaxation

---

* The practical route to prediction of turbulent flows is the use of turbulence models to achive closure of the mean flow equations. Here, a two-equation viscosity model, which employs transport equations for k and ϵ, is used.
* The additional steps involved in the prediction of turbulent flows are the derivation of the fde's for k and ϵ with source linearized in appropriate manner, the use of wall functins for grids in the vicinity of walls, and the solution of the equations by the usual LBL method with appropriate under-relaxation for k, ϵ and μ_{eff}.

CHAPTER 5 THE STRUCTURE OF TEACH3D COMPUTER PROGRAM

5.1 Purpose and Scope of Present Chapter

---

1. Purpose: to describe the capabilities, principles of organisation and structure of the TEACH3D program for calculation of 3-D transient, turbulent and recirculating flows.

2. Contents:
   * Outline of main features of the program
   * Details of grid specification, storage of variables, control of iteration, etc.
   * Description of individual subroutines

---

* This chapter aims at describing the capabilities and limitations, and of the structure of the TEACH3D program.
* The contents include the main features of the program (capabilities and limitations, programming philosophy, etc.), and of the details of grid specification, storage of variables, etc. Also included is a description of the structure and functions of the individual subroutines.

5.2 Capabilities and Limitations

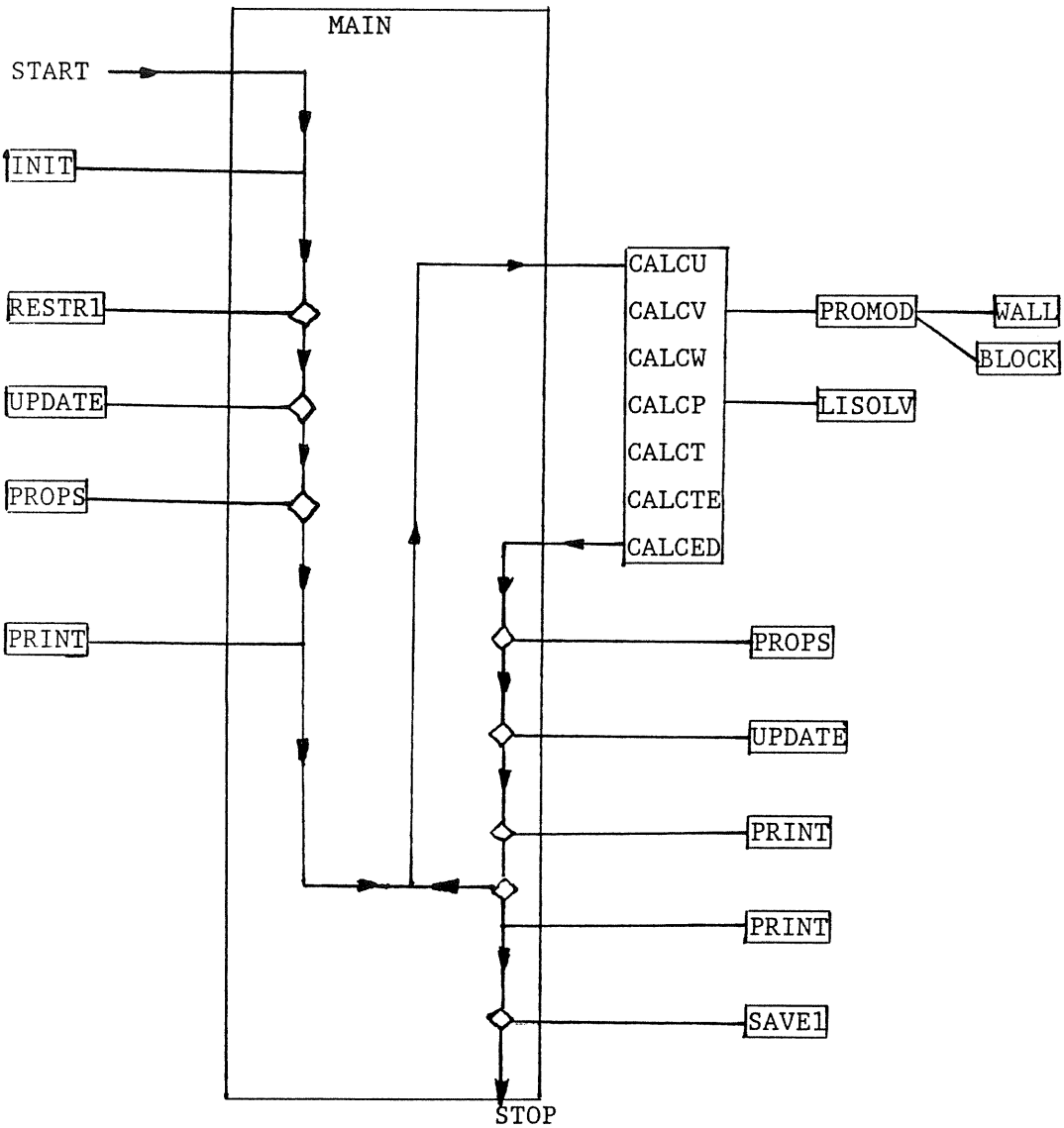
---

1. Class of flows: transient or steady, 3D, variable-property, laminar or turbulent.
2. Geometry and Grid: cartesian, arbitrary spacing.
3. Dependent variables: U, V, W, T, p, k and ϵ (others may be added).
4. Programming language and computers: FORTRAN 77; various computers e.g. VAX, CDC, IBM, UNIVAC etc.
5. Programming philosophy: teaching-oriented, modular structure, most subroutines independent of the type of problem.

---

* TEACH3D is essentially a general program for steady or transient, 3D flows. The flows may be laminar or turbulent, and of variable properties - it can certainly be made to handle compressible flows.
* The program is for flows which can be represented in cartesian coordinates, and the grid may be non-uniform.
* While the standard TEACH3D embodies equations for only U, V, W, T, p, k and ϵ, any other variables may be added with ease.
* The programming language is FORTRAN 77, and the program may be run on various computers including VAX, CDC, and UNIVAC.
* As the program is teaching-oriented it is written in a very simple a straightforward form which is really amenable to modifications. Most subroutines are independent of the flow type.

5.3 Program Flow Chart



* Here the overall structure of TEACH3D is illustrated.

* There are eight general subroutines relevant for any particular variable to be solved, namely, INIT, RESTR1, UPDATE, PROPS, SAVE1, PROMOD, LISOLV, PRINT, WALL and BLOCK. In addition, there is a

major set of CALCϕ subroutines, where ϕ is the particular variable solved.

* The inter-connection between various subroutines is shown in the panel.

* Overall control is exerted by MAIN which performs the initial and final operations, and also controls the iteration. The CALCϕ subroutines make the main calculations of the fde for each variable ϕ. Modification of sources and boundary conditions are made in PROMOD.

* In subroutine WALL, which should be be used in connection with PROMOD, boundary conditions according to the wall functions in panels 4.7-4.11 are set; zero flux boundary conditions is also provided as an option. The fluid properties (viscosity, density, etc.) are calculated in subroutine PROPS.

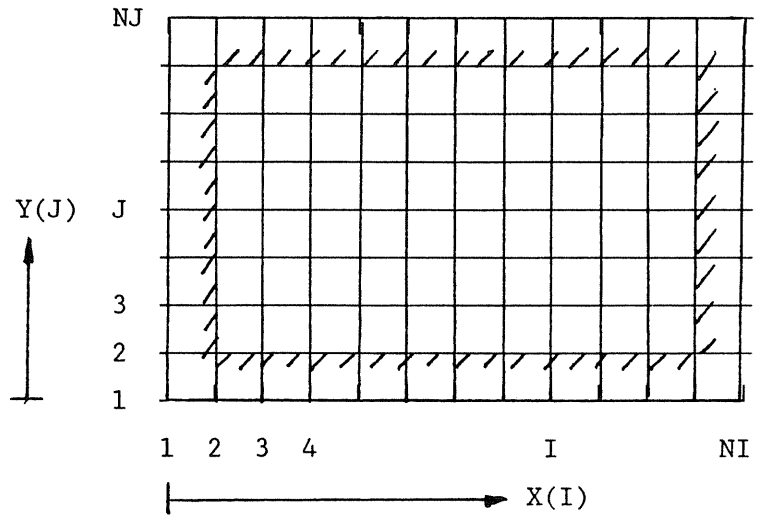
* BLOCK sets, for a prescribed region, the variable for which it is called to zero and applies boundary conditions (by calling WALL) at the boundaries of the 'blocked' region.

* INIT performs initialization tasks, PRINT provides output of variable arrays, and LISOLV performs the LBL iteration.

* UPDATE update variables (including the density) in transient calculations.

* RESTR1 and SAVE1 makes it possible to save calculated results on a disc-file and to make a restart using the previous calculated results as initial fields.

5.4 Domain of Solution and Grid



1. Co-ordinates:
   * X(I), I=1,NI and
   * Y(J), J=1,NJ and
   * Z(K), K=1,NK.

2. Domain of Solution: rectangular region bounded by I=2 to (NI-1), and J=2 to (NJ-1), and K=2 to (NK-1)

3. Domain of Flow: flow bounding surfaces coincide with boundaries of main control volumes (i.e. storage locations of normal velocities).

---

* The manner in which the co-ordinates are specified is shown in the panel. The domain of calculation is the entire cubic region bounded by I=2 to (NI-1), and J=2 to (NJ-1), and K=2 to (NK-1). However the region of calculation may always be altered if desirable.

* Note that the boundaries of the flow domain (shaded lines) always coincide with boundaries of main control volumes.

5.5 Conventions and Notation: i) Storage of dependent Variables

---

1. Scalar Variables (main nodes):

$T \rightarrow T(I,J,K)$

$p \rightarrow P(I,J,K)$

$p' \rightarrow PP(I,J,K)$

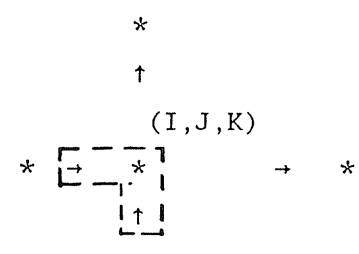
$k \rightarrow TE(I,J,K)$

$\epsilon \rightarrow ED(I,J,K)$

$\mu_{eff} \rightarrow VIS(I,J,K)$

$\Gamma_{eff} \rightarrow GAMH(I,J,K)$

$\rho \rightarrow DEN(I,J,K)$



2. Velocities (displaced locations):

$U \rightarrow U(I,J,K)$

$V \rightarrow V(I,J,K)$ } (I,J,K) refers to main node in direction of

$W \rightarrow W(I,J,K)$ increasing X, Y or Z.

Note: U calculations start at I=3, V at J=3 and W at K=3

---

* Some conventions and notation regarding storage of dependent variables are illustrated here. Subscript (I,J,K) refers to the node whose co-ordinates are X(I), Y(J) and Z(K).

* The diagram in the panel futher illustrates the 'staggered grid' system: the scalar variables are stored at the main nodes (*) while the associated velocities are displaced (→) and (↑). Hence the storage locations of the variables form a triad as shown by the dotted lines.

* Note further that while calculations for scalar variables, except modified by the user, start at point (2,2,2), U-calculation start at (3,2,2), that for V at (2,3,2) and that for W at (2,2,3).

5.6 Conventions and Notation: ii) Selection of Variables, Number of Sweeps, and Under-relaxation

---

1. Selection of Variables:

* Set INCALU=.TRUE., if U-momentum is to be solved;
and INCALU=.FALSE., if its calculation is to be supressed.
* Similarily, other equations are controlled by
INCALV, INCALW, INCALP, INCALK, INCALD, INCALT, INPRO.

2. Number of Sweeps:

* Set NSWPU=1, if line-iteration of U-momentum is once.
* For other variables: NSWPV, NSWPW, NSWPP, NSWPK, NSWPD, NSWPT.

3. Under-relaxation:

* Set URFU=0.5, if under-relaxation of U is 0.5
* For other variables: URFV, URFW, URFP, URFK, URFD, URFT.

---

* This panel illustrates some futher conventions and notations, but this time relating to selection of variables, number of sweeps, and under-relaxation.
* The solution of the fde for any variable ϕ (=U, V, W, p, etc) is obtained by setting INCALϕ=.TRUE.; if INCALϕ is set to .FALSE. the calculation of the particular ϕ-equation is supressed. INPRO similarily controls the calculation of fluid properties.
* The notation used for setting the number of application of the LBL procedure without updating the coefficients for any particular variable ϕ is NSWPϕ. Often, this will take values between 1 and 6, depending on the particular ϕ and the flow.
* In much the same manner, the under-relaxation for any variable ϕ is set by specifying values for URFϕ. This will generally have values between 0.3 and 1.0, according to the particular ϕ.

5.7 Iterating Monitoring, Control, and Preliminary Results

---

1. Iterating Monitoring:

* Niter=cumulative number of iterations at the current execution

* RESORU, RESORV, etc.=absolute sum of residual sources (RESORM for mass in p'-equation).

2. Iteration Control: calculation terminated if

* SORCE > 10^4*SORMAX (divergent), or

* SORCE ≤ SORMAX (converged), or NITER=MAXIT

Note: SORCE=max{ RESORM, RESORU, RESORV, RESORW, RESORT}

3. Pressure Level:

* P(IPREF,JPREF,KPREF) pre-specified and remains unaltered

4. Printout:

* Every INDMON iteration: NITER+FITER (FITER=First ITERation for the current run; it may be larger than 1 when restarts are made), RESOR's (usually normalized), all variables at location (IMON, JMON, KMON).

* Of variable arrays:

  i) before and after iteration sequence

  ii) every INDPRI iteration

  iii) during transient runs every INDPRT time-step

---

* Iteration monitoring, control, and printout of results give the user a better understanding of the succes/failure of the calculation procedure for any particular problem.

* For iteration monitoring, the cumulative number of iterations performed for the current run is calculated and stored as NITER and the absolute sum of the residual sources are stored as RESORϕ (ϕ=U, V, W, p, etc.) - for the p'-equation the absolute mass sources are stored as RESORM. The RESOR's are usually normalized.

* The iteration is controlled such that the calculations may be terminated for three reasons: the maximum residual source SORCE is excessively large after 20 iterations (i.e. a diverging solution); SORCE has fallen below a maximum acceptable value, specified as SORMAX; or NITER has reached a maximum value allowed, MAXIT.

* The p'-equation can be satisfied by several pressure fields. Hence, the pressure is pre-specified at the location (IPREF, JPREF, KPREF) and fixed at this value, with all other pressures measured relative to it: thus, if this location is within the flow domain, the pressure level is thereby fixed.

* At every INDMON iteration, output is provided of NITER+FITER, RESOR's and all variables at a specified location (IMON, JMON, KMON). The variable arrays are printed out before and after the iteration sequence - values during the sequence are printed out at intervals of INDPRI iteration, and, in the case of transient calculations, every INDPRT time step.

5.8.1 FORTRAN Variables for Coeffients of Finite-difference Equations

---

1. Finite-difference Equations: form of programmed equations:

* $(a_P-b+C_P)\phi_P = a_W\phi_W+a_E\phi_E+a_S\phi_S+a_N\phi_N+a_L\phi_L+a_H\phi_H+a_P^o\phi_P^o+C+C_P\phi_P^{old}$

* Here, $a_P \equiv a_W+a_E+a_S+a_N+a_L+a_H+a_P^o$; $C_P \equiv \max(0,M_P)$

2. FORTRAN Symbols:

* $a_P \rightarrow$ AP(I,J,K); $a_P^o \rightarrow$ APO(I,J,K); $a_W \rightarrow$ AW(I,J,K); etc.

* $(C+C_P) \rightarrow$ SU(I,J,K); $(b-C_P) \rightarrow$ SP(I,J,K).

* In velocity-correction formula: $D_u \rightarrow$ DU(I,J,K); $D_v \rightarrow$ DV(I,J,K); $D_w \rightarrow$ DW(I,J,K).

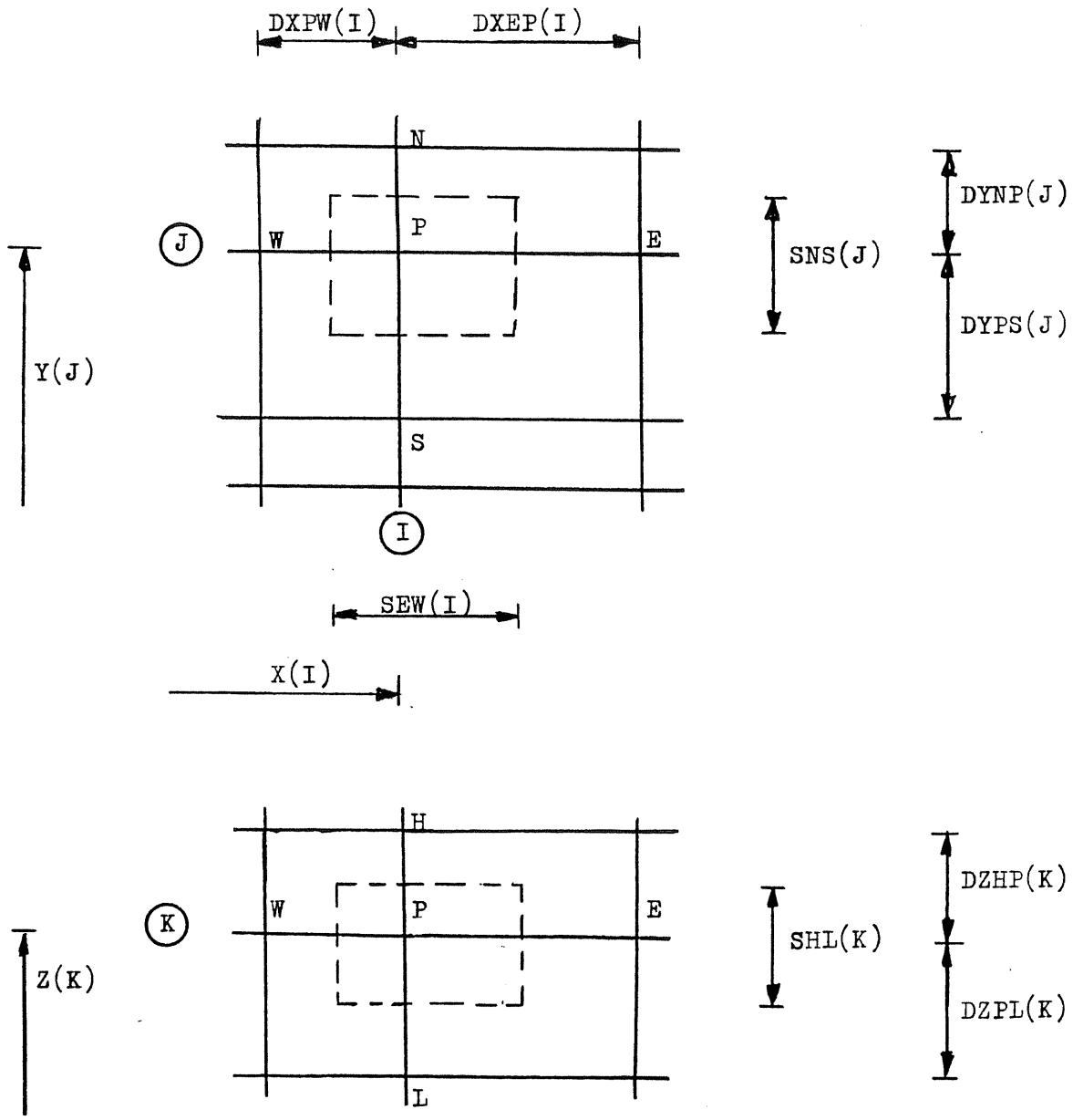
---

* Here the FORTRAN symbols used in the programmed fde's are explained. Note that the final form of the fde's solved is that given in panel 3.5.2. M_P is the net outflow from the control volume. It should be remembered that C_P is employed only as an artifice which ensures stability: through its use, the coeffient of ϕ_P remains finite if there is a net outflow.

* The FORTRAN symbols have been carefully chosen so as to bear direct relationship with their equivalents in the fde's (e.g. AS(I,J,K) for a_S, etc.)

5.8.2 FORTRAN Variables related to the grids: i) Scalar-cell

i) Scalar-cell (p, T, k, ϵ, etc.)



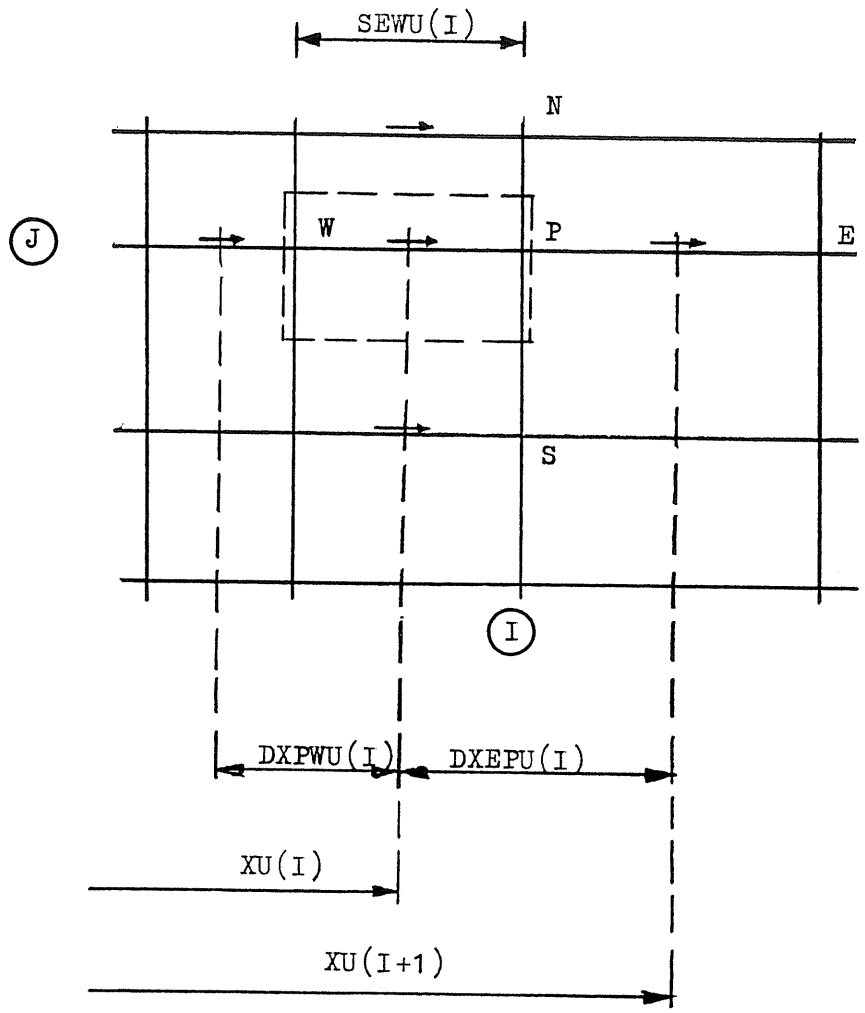
* This and the next two panels illustrate the FORTRAN variables relating to the dimensions/co-ordinates of the various cells. The scalar-cell is the subject of this panel.

* The panel is self-explanatory, and the user will find it (as well as the next three panels) a useful companion in adapting TEACH3D for various problems.
* Note that the boundaries of the scalar cell lie mid-way between main grid nodes.

5.8.3 FORTRAN Variables related to the grid: ii) U-cell

---

ii) U-cell.



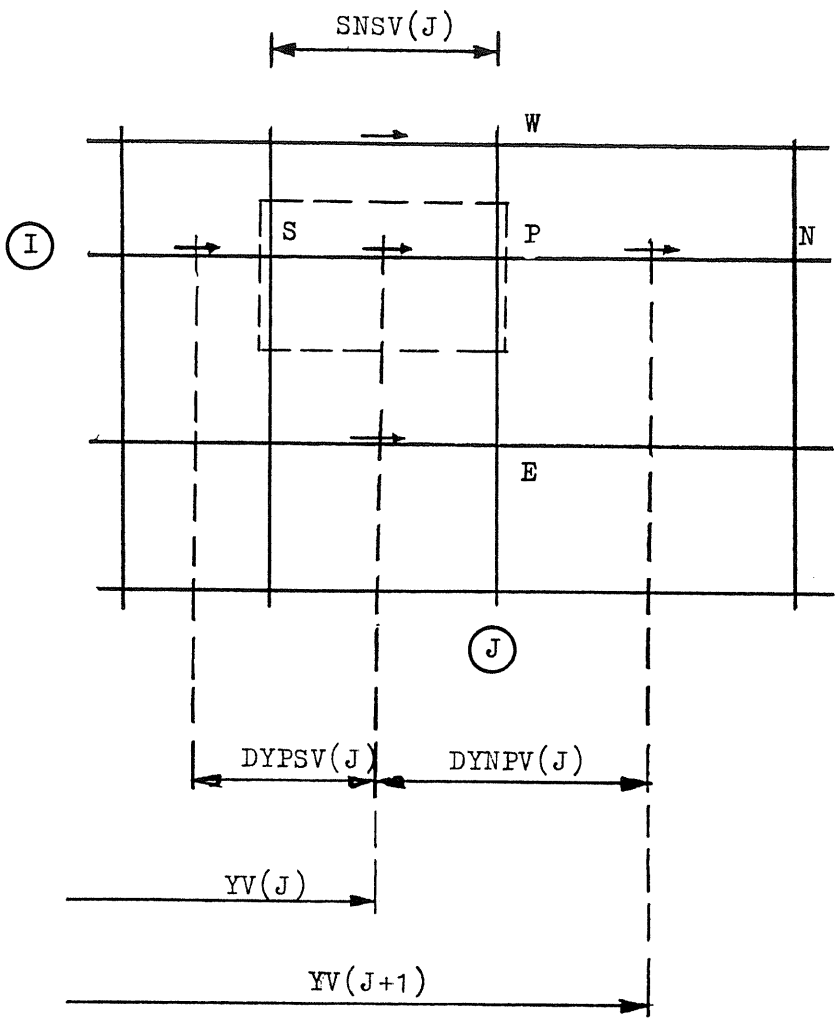
---

* This is another self-explanatory panel relating to the grid, but this time for a typical U-cell.

* Note that the west wall of the cell does not lie mid-way of DXPWU(I); neither does the east cell lie mid-way of DXEPU(I). The south, north, low and high walls lie mid-way of distances PS, NP, PL and HP respectively.

5.8.4. FORTRAN Variables related to the grid: iii) V-cell

iii) V-cell.

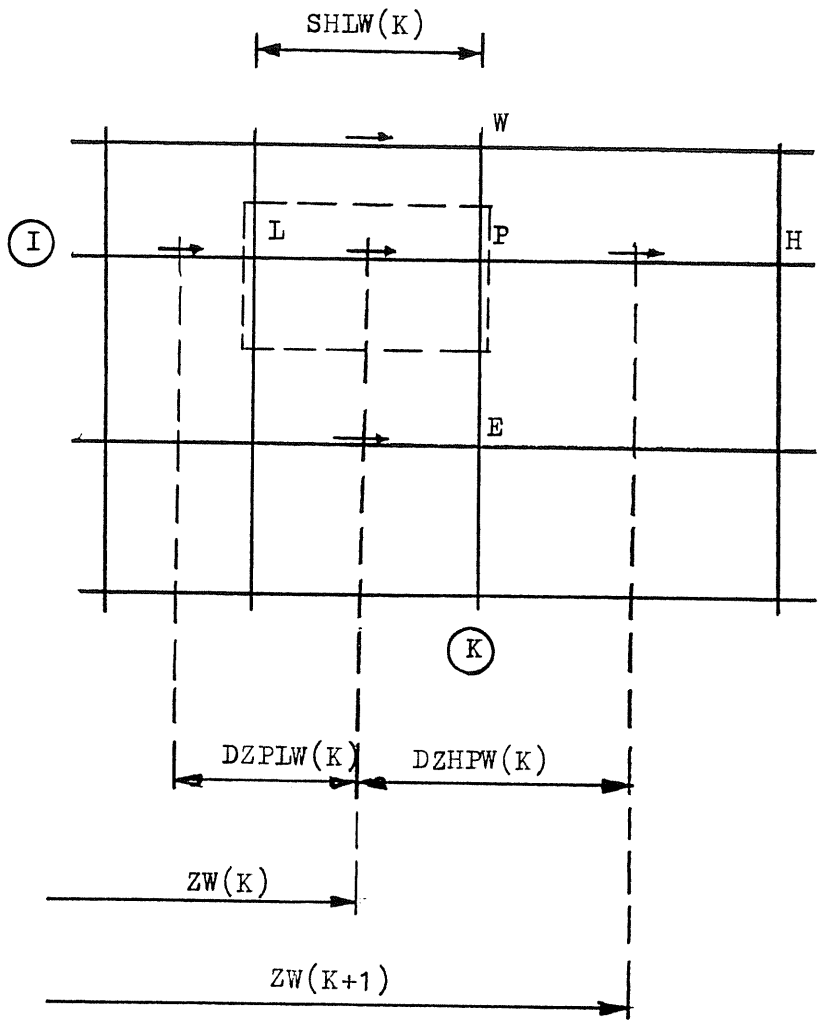


* This is another self-explanatory panel relating to the grid: the V-cell.

* Note that the north boundary of the cell does not lie mid-way of DYNPV(J). Neither does the south boundary lie mid-way of DYPSV(J). The west, east, low and high boundaries lie mid-way of WP, PE, LP and HP respectively.

5.8.5. FORTRAN Variables related to the grid: iv) W-cell

W-cell.



---

* This panel completes the picture of the FORTRAN variables relating to the grid, by illustrating the case for a typical W-cell.
* Note that the high boundary of the cell does not lie mid-way of DZHPW(K). Neither does the low boundary lie mid-way of DZPLW(K). The west, east, south and boundaries lie mid-way of WP, PE, SP and PS respectively.

5.9 Structure and Functions of MAIN

---

* Chapter 1: specification of grid, control parameters, constants of problem, etc.
* Chapter 2: calculation of grid parameters, initialisation of arrays (via INIT or from disc-file via RESTR1), prescription of fixed boundary values, preliminary output, etc.
* Chapter 3: iteration and output control.
* Chapter 4: final output and, if prescribed, results saved on disc-file.

---

* The remaining part of this Chapter is devoted to the structure and functions of the various subroutines of TEACH3D. Generally, each subroutine is divided into chapters to faciliate easy understanding of the entire program. In this panel, the functions of the various chapters of MAIN are given.

* Chapter 1 of MAIN carries out the initial specification of the grid, as well as the control parameters, constants of the problem, and any other relevant specification.

* In Chapter 2 the calculation of the grid parameters and initialization of arrays are made via INIT. When it has been prescribed that a restart is to be made, these arrays are initialized from a disc-file via RESTR1; in the latter case and when the run is transient, the variables are updated in the subroutine UPDATE so that the arrays read from the disc-file are stored as 'old' values. After the initialization via INIT, improved initial field may be specified as well as fixed boundary conditions. Before actual iteration starts, an output of the initial variable fields (or fields stored on the disc-file) as well as other preliminary outputs may be necassary, and this is also done in Chapter 2.

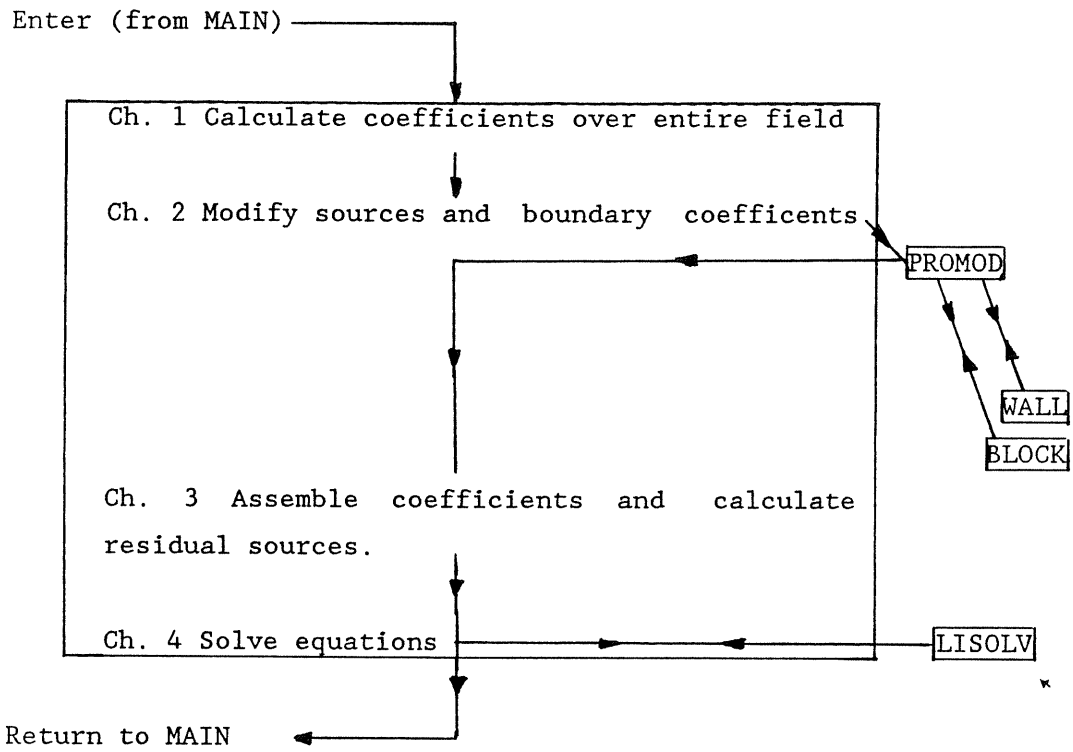
* Chapter 3 initiates and controls the iteration. It also gives intermediate outputs of NITER+FITER, RESOR's, ϕ(IMON,JMON,KMON) every INDMON iteration. The ϕ-fields are printed out every INDPRN

iteration, and, in transient runs, every INDPRT time step. In transient runs ITSTEP, DT and TIME are printed out every time step.

* Final operations, like saving the results on a disc-file, as well as the final output, are carried out in Chapter 4.

5.10 Structure and Functions of CALCϕ Subroutines

General structure for all but CALCP:



* This panel illustrates the general structure of all CALCϕ subroutines, with the exception of only CALCP.
* Entry to each of these subroutines is made from, and finally exit also made to, MAIN.
* Chapter 1 calculates the coefficients over the entire field. This is done from the standard total-flux expressions. The calculations made here is irrespective of the type of problem.
* Then, in Chapter 2 a call is made to PROMOD in order to modify the sources and boundary coeffients to suit the particular problem. The subroutine WALL can be called from PROMOD where boundary conditions according to the wall-functions in panels 4.7-4.11 are set. BLOCK may also be called from PROMOD where a

particular variable is set to zero in a prescribed region; boundary conditions are automatically applied at the boundaries of the 'blocked' region by use of WALL.

* Chapter 3 then gathers all the coefficients, and also calculates the residual sources RESORϕ's from the ϕ of the previous iteration.

* Finally, a call is made to LISOLV for the application of the LBL procedure before a return to MAIN is made.

5.11 Structure and functions of CALCP Subroutine

Special features of CALCP:

1. a_n's are unlikely all to become zero, so no special precautions are taken.
2. In Chapter 1 absolute mass sources are summed and stores as RESORM.
3. Residual source of p'-equation provide no useful information, so they are not calculated.
4. Corrections are applied to pressure and velocities in (additional) Chapter 5.

---

* The special features of CALCP which distinguish it from the other CALCϕ subroutines are given in this panel.
* The P'-equation is unlikely to become singular since the a_n's are unlikely to become zero. Hence no special precautions are taken regarding introduction of 'false' sources.
* Chapter 1 has an additional feature: the absolute mass sources are summed and stored as RESORM.
* No residual sources of P'-equation are calculated because they provide no useful information.
* Finally, an additional chapter (Chapter 5) is provided to carry out the calculation of the pressure- and velocity-corrections.

5.12 Structure and function of PROMOD

---

1. Function: to enable problem-dependent sources and boundary conditions to be embodied in fde's.
2. Structure:
   * Subdivided into chapters, each pertaining to particular variable.
   * Each chapter has individual ENTRY and RETURN point: former are labelled MODU for U, MODV for V, etc.
   * User must supply all instructions.

---

* PROMOD plays a very important role in the program as it is the major area where the sources and boundary conditions can be modified to suit individual problems. The reader might want to study this subroutine carefully, to make sure modifications for the standard case are well understood.
* It is divided into chapters, each chapter pertaining to a particular variable ϕ. There are separate ENTRY and RETURN point for each chapter. The entry point are labelled MODϕ where ϕ reflects the relevent variable (U, V, etc.)
* The user is advised to make in PROMOD all mayor modifications necassary for any particular problem, and he must be aware he bears full responsibility for such modifications.

5.13 Structure and functions of WALL

---

* This subroutine should normally be called from PROMOD.

* Boundary conditions according to the wall functions described in panels 4.7-4.11 are set here.

* Zero gradient boundary conditions are also possible to prescribe with this subroutine.

* Arguments: PHI, FACE, ISTART, IEND, JSTART, JEND, KSTART, KEND and VALUE.

---

* Boundary conditions according to the wall functions for the dependent variables U, V, W, T, k and ϵ described in panels 4.7-4.11 are conveniently set by using this subroutine. WALL is really an extension of PROMOD for the convenience of the user and can maybe serve to reduce some errors in prescribing boundary conditions; WALL should normally be used only in connection with PROMOD.

* As an option, zero gradient boundary conditions may conveniently be chosen; the argument VALUE is then given a value LE.-100. (see below).

* The arguments in the CALL WALL statement are:

PHI=name of dependent variable, i.e. U, V, W, T, TE, or EP

FACE=side of the cell(s) which faces the boundary, i.e. WEST, EAST, SOUTH, NORTH, LOW or HIGH

ISTART, IEND, JSTART, JEND, KSTART, KEND=the nodes (note: scalar nodes) which are near the boundary of the region for which the boundary condition is to be applied. Note: the nodes ISTART, IEND, etc. are in the region.

VALUE: For any other variable than T the exact value of VALUE is of no meaning exept that if VALUE.LE.-100. zero gradient boundary condition is applied, otherwise boundary conditions according to the wall functions. For the variable T VALUE=wall temperature; if VALUE.LE.-100. adibatic wall is prescribed.

* The FORTRAN types of the arguments are: PHI and FACE are CHARACTER variables, VALUE is REAL and the rest are INTEGER variables.

* The user is advised to study the subroutine in order to fully understand what is carried out in it.

5.14 Structure and functions of BLOCK

---

* This subroutine should normally be called from PROMOD.

* Regions within the calculation domain are conveniently 'blocked' by using this subroutine.

* Boundary conditions are automatically applied at the boundaries of the 'blocked' region.

* Arguments: PHI, ISTART, IEND, JSTART, JEND, KSTART, KEND and VALUE.

---

* Regions are 'blocked' by setting the variable, for which BLOCK is called, to zero. BLOCK is an extension of PROMOD for the convenience of the user and should normally be used only in connection with PROMOD.

* As an option, symmetry plane boundary conditions may conveniently be chosen; the argument VALUE is then given a value LE.-100. (see below).

* Note that if a region with zero velocities is prescribed using BLOCK, the subroutine BLOCK must also be called for the pressure correction equation, PP, in order to avoid divison by zero in LISOLV.

* The arguments in the CALL BLOCK statement are:

PHI=name of dependent variable, i.e. U, V, W, PP, T, TE, or EP

ISTART, IEND, JSTART, JEND, KSTART, KEND=the nodes (note: scalar nodes) which are near the boundary of the region which the variable PHI (ϕ) is set to zero. Note: the nodes ISTART, IEND, etc. are in the region.

VALUE: For any other variable than T the exact value of VALUE is of no meaning exept that if VALUE.LE.-100. symmetry boundary condition is applied, otherwise boundary conditions according to the wall functions. For the variable T VALUE=wall temperature; if VALUE.LE.-100. adibatic wall is prescribed.

* The FORTRAN types of the arguments are: PHI and FACE are CHARACTER variables, VALUE is REAL and the rest are INTEGER variables.

* It may be noted that when VALUE.LE.-100 symmetry boundary conditions are applied whereas in WALL zero gradient are applied; the difference is that in the former case the normal velocity component is set to zero at the boundaries while in the latter case the normal gradient of the normal velocity component is set to zero.

* The user is advised to study the subroutine in order to fully understand what is carried out in it.

5.15 Structure and functions of PROPS and LISOLV

---

1. PROPS: calculates values over entire field of thermodynamic and transport properties (e.g. ρ, μ_{eff}, Γ_{eff}, etc.)

2. LISOLV: applies line-iteration algorithm, arranged to solve along, consequtively, E-W lines, N-S lines and finally H-L lines sweeping over the entire field. Important arguments are:

PHI(I,J,K) - array containing the variable to be computed.

ISTART, JSTART and KSTART - starting indices of traverses and sweeps. This means that for every NSWPϕ (see panel 5.6) three sweeps are performed; one in each direction.

---

* Subroutine PROPS evaluates the fluid properties based on user-supplied formulae. The form of PROPS in the standard TEACH3D will, however, be found useful for many problems.

* The LBL iteration method is performed in subroutine LISOLV which is arranged so as to solve along, consequtively, E-W lines, N-S lines and H-L lines. It may be noticed that the LBL procedure thus is applicated three times when NSWPϕ=1 (see panel 5.6). The most important arguments of LISOLV are PHI(I,J,K), ISTART, JSTART, and KSTART which are defined in the panel.

5.16 Structure and functions of INIT and PRINT

---

1. INIT:

* Calculates from grid co-ordinates, inter-cell distances, cell dimensions, etc. (Chapter 1)

* Initialize dependent variable arrays (Chapter 2)

2. PRINT:

Prints out dependent-variable arrays, according to specification of arguments:

PHI(I,J,K) - array in question

X(I), Y(J), Z(K) - co-ordinates of storage locations

HEAD - alphanumeric array containing variable name

ISTART, JSTART, KSTART - starting values of indices I,J,K

ILAST, JLAST, KLAST - last values of indices I,J,K

---

* Subroutine INIT calculates from grid co-ordinates, inter-node distances, cell dimensions and so on in Chapter 1. In Chapter 2, initial values of the dependent variable arrays are specified: specially, the starting variable fields (except ρ and μ) are set to a small value (=SMALL).

* The printing of the dependent variable arrays is performed by subroutine PRINT. The particular arguments to note are PHI(I,J,K), X(I), Y(J), Z(K), HEAD, ISTART, JSTART, KSTART, ILAST, JLAST and KLAST, all of which are defined in the panel.

5.17 Structure and functions of SAVE1 and RESRT1

---

* SAVE1 writes the dependent variable arrays as well as X, Y and Z on logical unit 17.
* RESTR1 reads the variables (except X, Y and Z) from logical unit 15.
* All dependent variables (i.e. U, V, W, T, k, ϵ and p) including X, Y, Z are written (SAVE1) and read (RESTR1) irrespectively of if they are being solved or not.

---

* The subroutines SAVE1 and RESTR1 handle the restart facilities; this means that the user can make a number of (say) 50 iterations, store the results on a disc-file, examine the results, and, if they are to his satisfaction, making further calculations using the calculated results as initial fields.
* All dependent variables are stored on the disc-file irrespectively if they are solved or not. This may be convenient if, for example, the user wants to make thermal calculations using previously iso-thermal calculated results as initial fields; the temperature field is then stored from the iso-thermal calculations on the disc-file.
* The results are written/read on logical unit number 17/15; these are preferably assigned to appropriate files.
* The grid arrays X, Y and Z are also stored on the disc-file in order to make it convenient for the user when he is plotting his results. X, Y and Z are not read by RESTR1 since they have allready been specified by the user in MAIN.

5.18 Summary

---

1. Capability of TEACH3D:
   * 3D flows in cartesian co-ordinates.
   * Transient or steady, laminar or turbulent.
   * Constant- or variable-property.
2. Standard form:
   * Solves for U, V, W, T, p, k and ϵ.
   * Additional variables may be added.
3. Subroutines:
   * MAIN, PROPS and PROMOD to be modified to suit particular problem.
   * All other subroutines are general-purpose.

---

* TEACH3D is a program for 3D, transient or steady, laminar or turbulent flows in cartesian co-ordinates.
* In its standard form, it solves for the variables U, V, W, T, p, k and ϵ: any unlimited set of additional variables may conveniently be added.
* The program is particularly written to aid quick understanding, and most of the subroutines (except MAIN, PROPS, and PROMOD) require no modifications for all types of problems.

FORTRAN SYMBOLS

A(I)	coefficient of recurrence formula
AE, AH, AL, AN, AS AW(I,J,K)	coefficients of convective/diffusive flux through east, high, low, north wall of control volume
AP(I,J,K)	sum of coefficients AE, AW, AN, AS, AH, AL and APO and source SP
APO(I,J,K)	coefficient for old time step
B, C(I)	coefficients of recurrence formula
C1, C2	constants of turbulence model (=1.44 and 1.92)
CAPPA	von Karman's constant (=0.435)
CD	constant of turbulence model (=1.0)
CE, CH, CL	coefficient of convective flux through east, high and low wall of control volume
CMU	constant of turbulence model (=0.09)
CMUCD	constant of turbulence model (=CMU*CD)
CN	coefficient of convective flux through north wall of control volume
CP	maximum of zero and net outflow (SMP) from control volume
CPO	=CP

CS, CW	coefficient of convective flux through south and west wall of control volume
D(I)	coefficient of recurrence formula
DEN(I,J,K)	density of fluid
DENSIT	constant density of fluid set in INIT
DFE, DFH, DFL, DFN, DFS, DFW	coefficient of diffusive flux through east, high, low, north, south and west wall of control volume
DU(I,J,K)	coefficient of velocity-correction term for U velocity
DUDX, DUDY DUDZ	$\partial U/\partial x$, $\partial U/\partial y$ and $\partial U/\partial z$ at main grid node (I,J,K)
DUDXM, DUDXP	$\partial U/\partial x$ at main grid nodes [(I-1,J,K) and (I,J,K)]
DUDYM, DUDYP	$\partial U/\partial y$ at midpoint of south and north face of the U-velocity control volume
DUDZM, DUDZP	$\partial U/\partial z$ at midpoint of low and high face of the U-velocity control volume
DV(I,J,K)	coefficient of velocity-correction term for V velocity
DVDX, DVDY DVDZ	$\partial V/\partial x$, $\partial V/\partial y$ and $\partial V/\partial z$ at main grid node (I,J,K)
DVDXM, DVDXP	$\partial V/\partial x$ at midpoint of west and east face of the V-velocity control volume

DVDYM, DVDYP $\partial V/\partial y$ at main grid nodes [(I,J-1,K) and (I,J,K)]

DVDZM, DVDZP $\partial U/\partial z$ at midpoint of low and high face of the V-velocity control volume

DW(I,J,K) coefficient of velocity-correction term for W velocity

DWDX, DWDY
DWDZ $\partial W/\partial x$, $\partial W/\partial y$ and $\partial W/\partial z$ at main grid node (I,J,K)

DWDXM, DWDXP $\partial W/\partial x$ at midpoint of west and east face of the W-velocity control volume

DWDYM, DWDYP $\partial W/\partial y$ at midpoint of south and north face of the W-velocity control volume

DWDZM, DWDZP $\partial W/\partial z$ at main grid nodes [(I,J,K-1) and (I,J,K)]

DXEP(I) =X(I+1)-X(I)

DXEPU(I) =XU(I+1)-XU(I)

DXPW(I) =X(I)-X(I-1)

DXPWU(I) =XU(I)-XU(I-1)

DYNP(J) =Y(J+1)-Y(J)

DYNPV(J) =YV(J+1)-YV(J)

DYPS(J) =Y(J)-Y(J-1)

DYPSV(J)= =YV(J)-YV(J-1)

DZHP(K)	=Z(K+1)-Z(K)
DZHPW(K)	=ZW(K+1)-ZW(K)
DZPL(K)	=Z(K)-Z(K-1)
DZPLW(K)	=ZW(K)-ZW(K-1)
ED	energy dissipation rate, ϵ
ELOG	constant of P-function for heat transfer at walls
FITER	first iteration
FLOWIN	mass flux at inlet of domain
GAMM, GAMP	viscosity at mid-point of downstream and upstream wall of cell
GAME, GAMHX, GAML, GAMN, GAMS, GAMW	coefficients of diffusion for scalar variables at east, high, low, north, south and west wall
GAMH(I,J,K)	coefficient of diffusion for temperature
GE,GH,GL GN,GS,GW	mass flux through east, high, low, north, south and west wall of cell
GEN(I,J,K)	generation of turbulence by shear from mean flow
GP	mass flux at location of velocity
GREAT	a very large value (i.e. 10^{10})

HEATIN heat flux at inlet of domain

HEDD heading 'Energy Dissipation'

HEDK heading 'Turbulent Energy'

HEDM heading 'Viscosity'

HEDP heading 'Pressure'

HEDT heading 'Temperature'

HEDU heading 'U-Velocity'

HEDV heading 'V-Velocity'

HEDW heading 'W-Velocity'

IMON I-index of monitoring location

INCALD, INCALK
INCALP, INCALT
INCALU, INCALV
INCALW logical parameter for solution of ϵ, k, P', T, U, V, W-equation

INDMON monitoring output each INDMON iteration

INDPRI intermediate output of variable fields each INDPRI iteration

INDPRT intermediate output of variable fields each INDPRT time step (transient runs)

IPREF I-index of location where pressure is fixed

INPRO	logical variable for updating of fluid properties
IT	I-index of maximum dimension of dependent variable
JMON	J-index of monitoring location
JPREF	J-index of location where pressure is fixed
JT	J-index of maximum dimension of dependent variable
KMON	K-index of monitoring location
KPREF	K-index of location where pressure is fixed
KT	K-index of maximum dimension of dependent variable
MAXIT	maximum number of iterations to be completed in the current run if iteration is not stopped by test on value of SORCE
NI	maximum value of I-index for the calculation domain
NIM1	=NI-1
NITER	number of iterations completed
NJ	maximum value of J-index for the calculation domain
NJM1	=NJ-1
NK	maximum value of K-index for the calculation domain
NKM1	=NK-1
NSWPD,NSWPK NSWPP,NSWPT	

NSWPU,NSWPV
NSWPW — number of application of line iteration for ϵ, k, P′, T, U, V and W-equation

P(I,J,K) — pressure

PHI(I,J,K) — general representation for all dependent variables

PP(I,J,K) — pressure-correction, P′

PRANDL — laminar Prandtl number for temperature

PRANDT,PRTE — turbulent Prandtl number for temperature and turbulent kinetic energy

RESOR — residual source for individual control volume

RESORE,RESORK
RESORM,RESORT
RESORU,RESORV
RESORW — sum of absolute residual sources within calculation domain for ϵ, k, P′, T, U, V and W-equation

SEW(I) — 0.5*[DXEP(I)+DXPW(I)]

SEWU(I) — 0.5*[DXEPU(I)+DXPWU(I)]

SMP — net outflow from control volume

SHL(K) — 0.5*(DZHP(K)+DZPL(K)]

SHLW(K) — 0.5*[DZHPW(K)+DZPLW(K)]

SMALL — a very small value (i.e. 10^{10})

SNS(J) — 0.5*[DYNP(J)+DYPS(J)]

SNSV(J)	0.5*[DYNPV(J)+DYPSV(J)]
SORCE	maximum of RESORM, RESORU, RESORV and RESORW
SORMAX	maximum acceptable value of SORCE for converged solution
SP,SU(I,J,K)	coefficient b and C of linearized source treatment
TAUE,TAUW(J,K) TAUS,TAUN(I,K) TAUL,TAUH(I,J)	shear stress at east, west, south, north, low and high boundary of flow domain
TE(I,J,K)	turbulent kinetic energy
TMULT	coefficient of wall shear expression
U(I,J,K)	U-velocity
URFE,URFK URFP,URFT URFU,URFV URFW	under-relaxation factor for ϵ, k, P', T, U, V and W-equation
V(I,J,K)	V-velocity
VIS(I,J,K)	effective viscosity ($\mu+\mu_t$)
VISCOS	laminar viscosity (μ)
VISE,VISH VISL,VISN VISS,VISW	effective viscosity at midpoint of east, high, low, north, south and west wall of cell

VISOLD	value of effective viscosity before under-relaxation
VOL	volume of control volume
X(I)	x co-ordinate of main cells
XMOMIN	momentum of fluid at inlet of flow domain
XPLUSE XPLUSW(J,K)	local Reynolds number based on the friction velocity and distance from east and west wall-boundary of flow domain
XU(I)	x co-ordinate at storage location of U
Y(J)	y co-ordinate of main cells
YPLUSN YPLUSS(I,K)	local Reynolds number based on the friction velocity and distance from north and south wall-boundary of flow domain
YV(J)	y co-ordinate at storage location of V
Z(K)	z co-ordinate of main cells
ZPLUSH ZPLUSL(I,J)	local Reynolds number based on the friction velocity and distance from high and low wall-boundary of flow domain
ZW(K)	z co-ordinate at storage location of W

REFERENCES

Davidson, L. and Olsson, E. (1986), "A new one-equation turbulence model applied to some parabolic and elliptical flows", Rept. 86/8, Dept. of Applied Thermodynamics and Fluid Mechanics, Chalmers Univ. of Tech., Gothenburg

Gosman, A. D. and Ideriah, F. J. K. (1976), "TEACH-T: A general computer program for two-dimensional, turbulent, recirculating flows", Dept. of Mechanical Engineering, Imperial College, London.

Hinze, J. O. (1976), "Turbulence", McGraw-Hill, 2nd ed.

Jayatillika, C. L. V. (1969), Progr. in Heat Mass Transfer, 1, 193

Launder, B. E. and Spalding, D. B. (1974), "The numerical computation of turbulent flows", Comp. Methods in Appl. Mech. and Eng., 3, 269

Patankar, S. V. and Spalding, D. B. (1972), "A calculation procedure for heat, mass, and momentum transfer in three-dimensional parabolic flows", Int. J. Heat Mass Transfer, 15.

Restivo, A. (1979), "Turbulent flows in ventilated rooms", PhD thesis, Dept. of Mech. Eng., Imperial College, London.

Rodi, W. (1980), "Turbulence models and their application in hydraulics", International Association of Hydraulic Research, Monograph, Delft.

Rosten, H. I. and Spalding, D. B. (1985), "PHOENICS-84 reference handbook", CHAM TR/100.

Scarborough, J. B. (1966), "Numerical mathematical analysis", 6th ed., The John Hopkins Press, Baltimore, Oxford Univ. Press, London.

Spalding, D. B. (1981), "A general purpose computer program for multi-dimensional one- and two-phase flow", Mathematics and Computers in Simulation, IAMCS, XXlll, 267.

APPENDIX 1. A THREE-DIMENSIONAL ROOM

The flow in a three-dimensional isothermally ventilated room, Fig. 1, is calculated using two different turbulence models: the standard k-ϵ model (which is incorporated in TEACH3D) and a one-equation model (herafter denoted by KL1), see Davidson and Olsson (1986). The predictions are compared with experimental data by Restivo (1979). The plane y=0 is a symmetry plane which means that it suffices to calculated the flow in one half of the room.

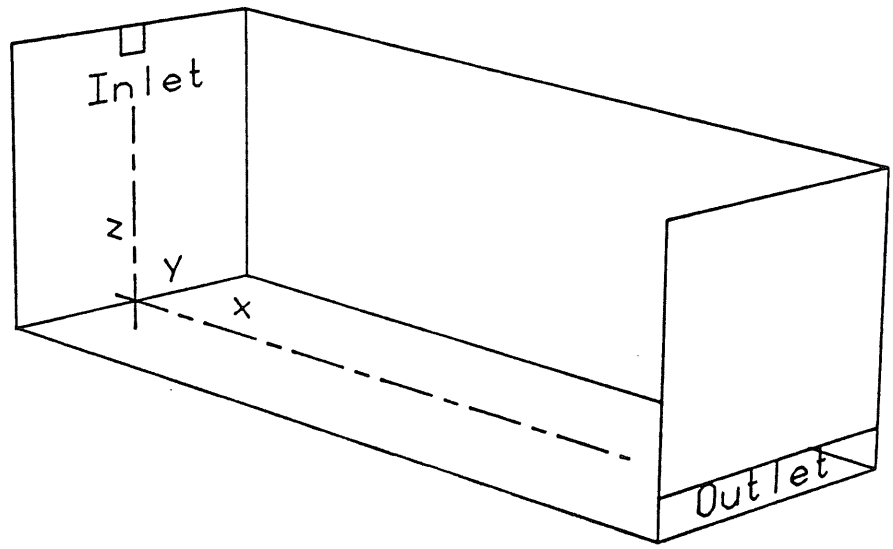


Figure 1. Flow configuration.

Listings of MAIN, the subroutines PROMOD and PROPS, and the common-block KASECOM are to be found at the end of this appendix.

Modifications are made in PROPS because a one-eqation turbulence model is used; when the k-ϵ model is used no modifications in PROPS is necessary.

In PROMOD boundary conditions are set. In MODPRO (MODify PROperties) the turbulence viscosity is set in the wall jet when the KL1 model is used. In MODϕ zero streamwise gradient is imposed at the outlet for all variables using WALL (VALUE.LE.-100.). This

treatment is not necassary for this problem because the Peclet number at the outlet is larger than two, which means that the coefficients at the east boundary of the cells at the outlet [AE(NIM1,J,K), J=2,JOUTLET, K=KOUTLET,NKM1] are zero anyway.

Results

Velocity vectors are presented in Figs. 2-3. Profiles of the U-velocity are compared with experimental data in Fig. 4 (y=0) and Fig. 5 (y=0.4 x width of the room). As can be seen the predictions are in good agreement with the experiments. The required CPU time on a VAX-750 machine for obtaining a converged solution was 2 hours and 14 minutes with the k-ϵ model; the number of iterations was 290. A run was also made setting NSWPU=NSWPV=NSWPW=NSWPK=NSWPE=2, and NSWPP=5; the CPU time was then 3 hours and 6 minutes and the number of iterations was 295.

This problem was also, for comparison, solved using the PHOENICS computer program, see Spalding (1981) and Rosten and Spalding (1985). The velocity profiles predicted with PHOENICS were more or less identical with those predicted by TEACH3D in Figs. 4-5. The required CPU time was, however, considerably larger, namely 4 hours and 55 minutes. For readers familiar to PHOENICS it may be interesting to know which relaxation parameters, etc. that was used. LITER(P1)=5 and LITER(ϕ)=2, DTFALS=0.5 for all variables (except P1), whole-field solution of P1, and z-axis in the direction of the inlet velocity. LITER(P1)=10 and LITER(ϕ)=1, DTFALS=0.1 and 1, z-axis as in Fig. 1 was also tested. It may be mentioned that the option "whole-field solution of all variables" does not work (confirmed by CHAM) in the current version of PHOENICS-84; use of this option may reduce the CPU time.

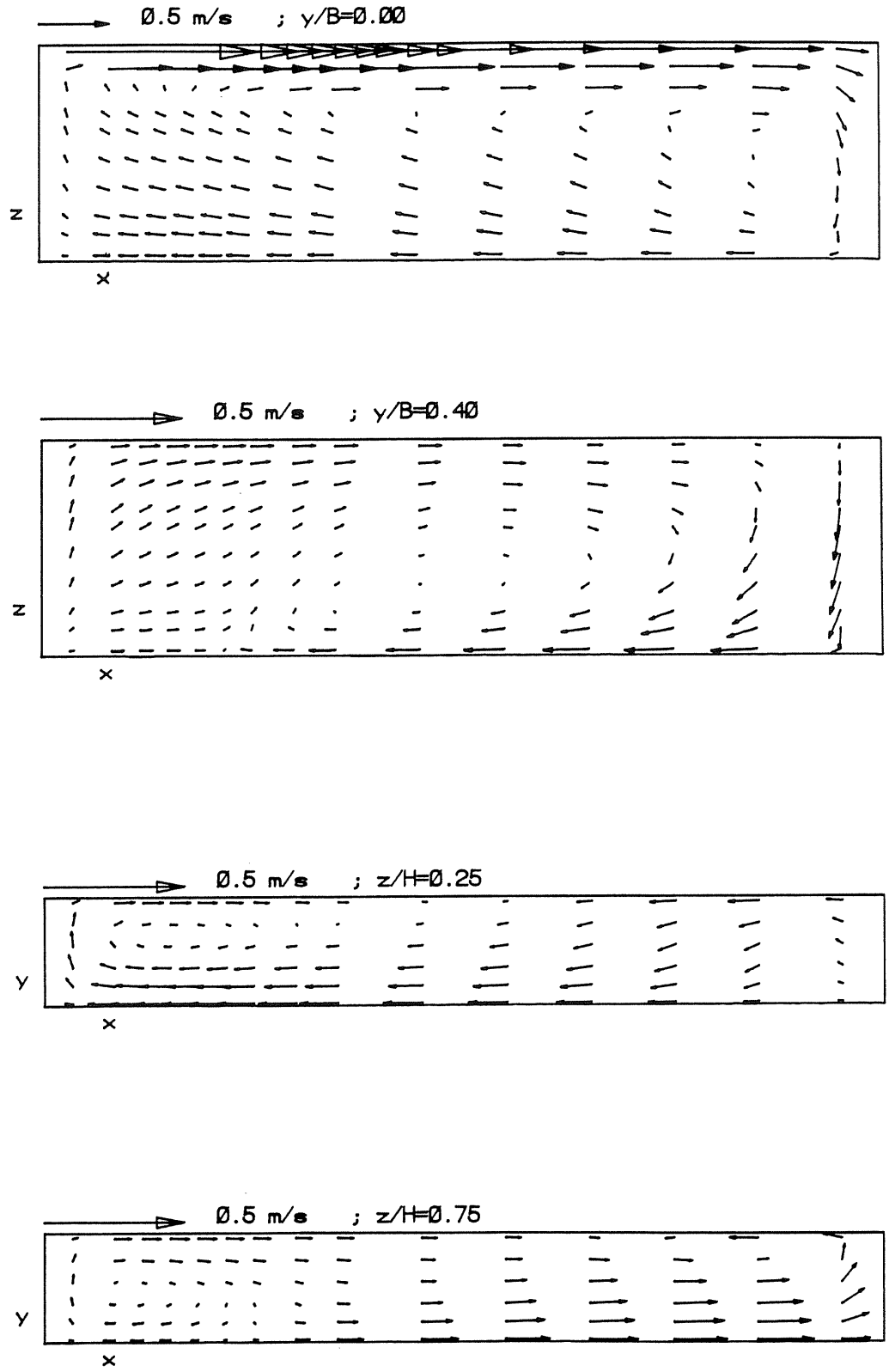


Figure 2. Predicted velocity vectors. k-ϵ model.

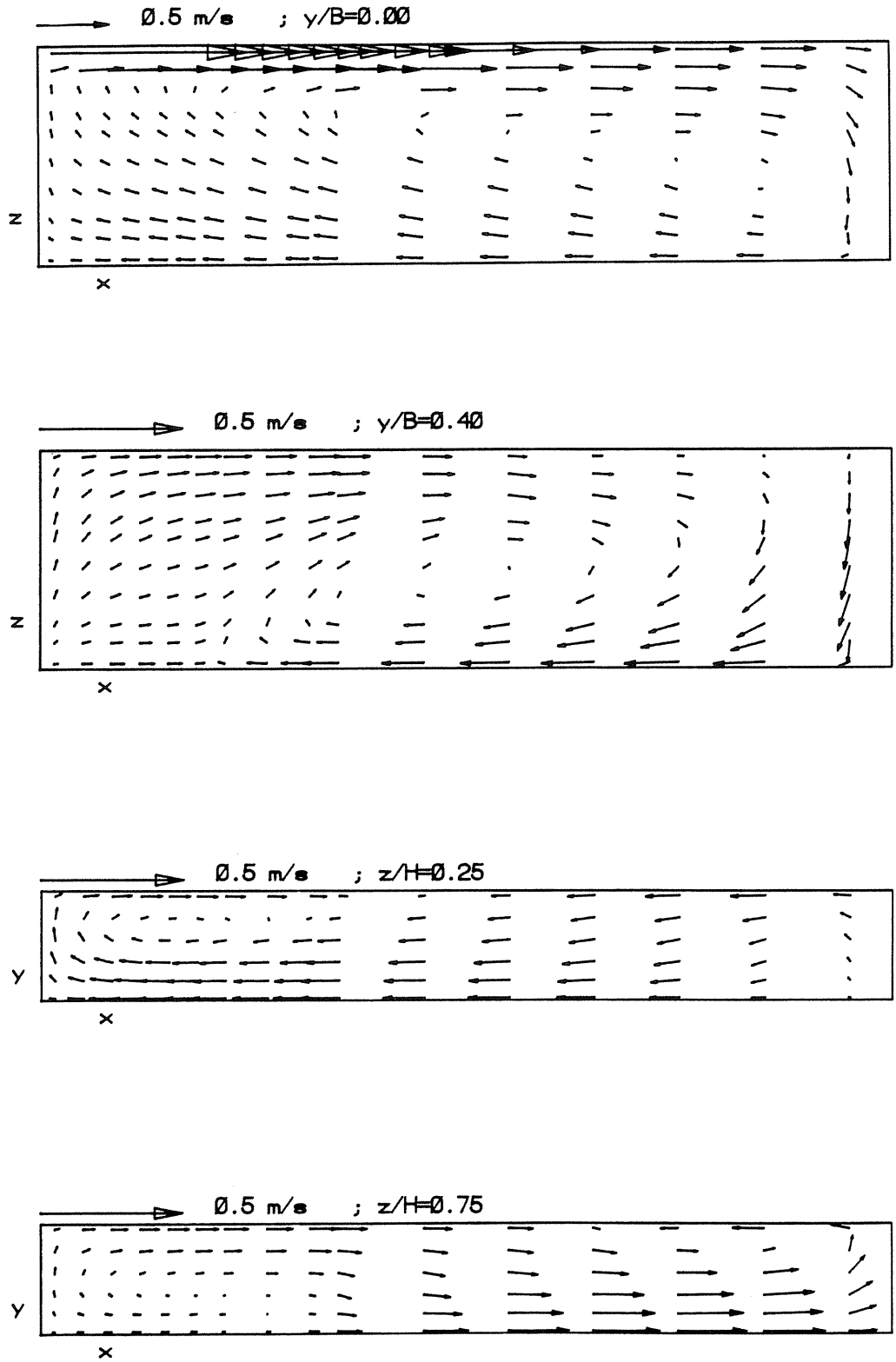


Figure 3. Predicted velocity vectors. KL1 model.

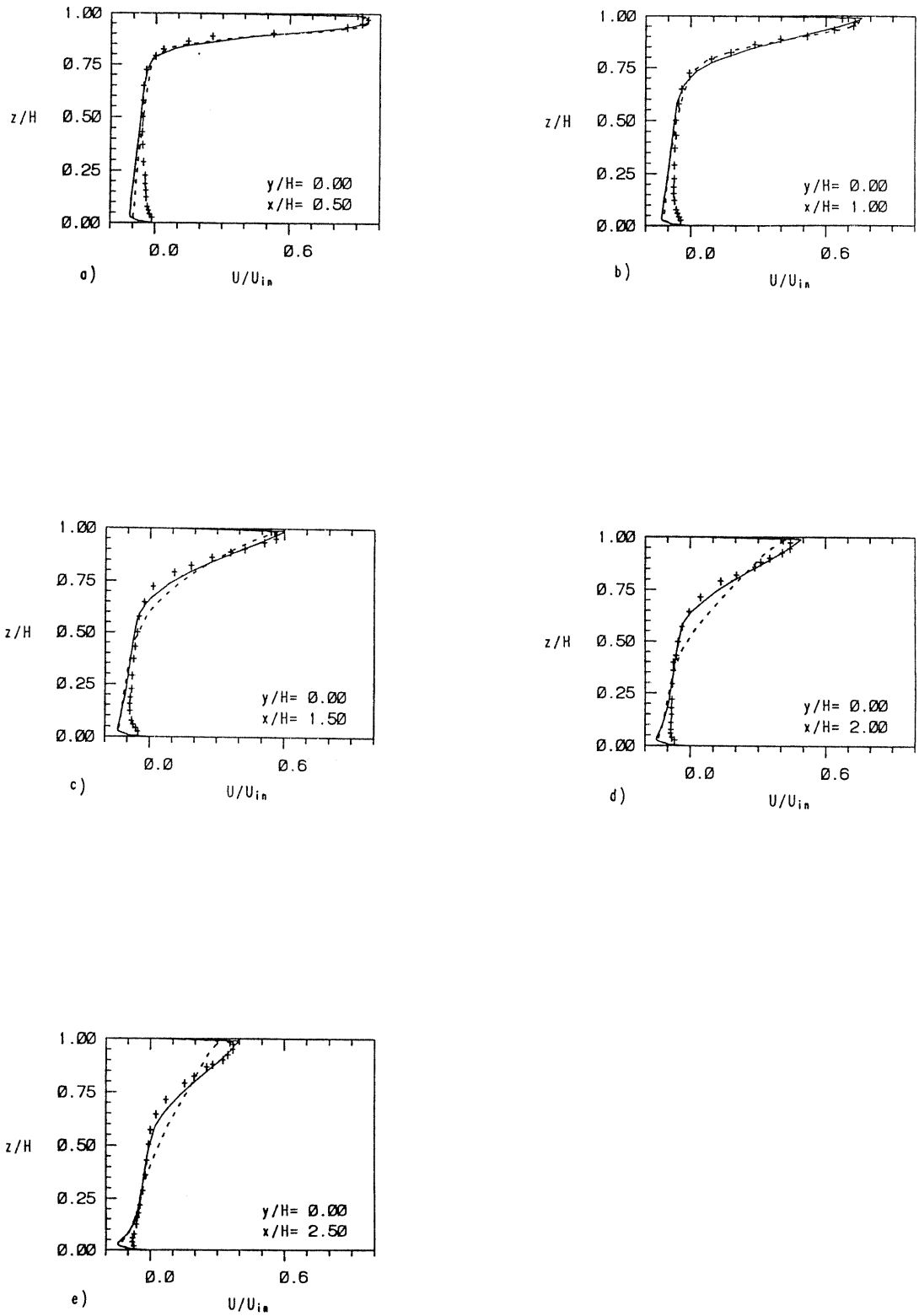
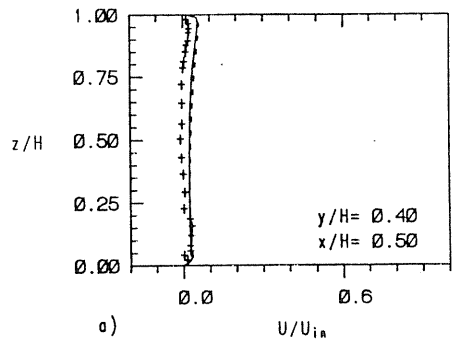
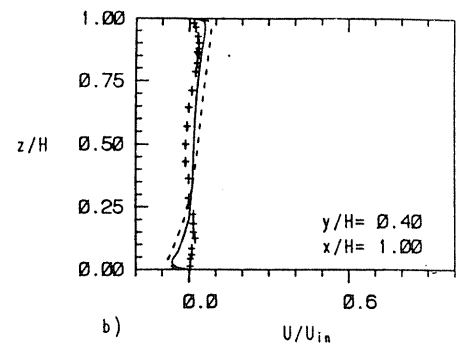
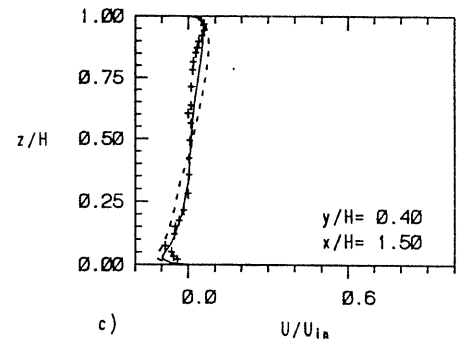
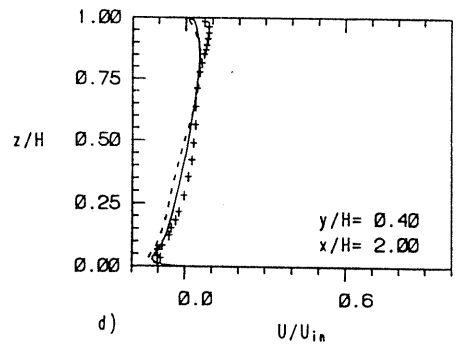


Figure 4. Profiles of the U-velocity. Plane y=0. Solid lines: k-ϵ model. Dashed lines: KL1 model. Expts. by Restivo (1979).









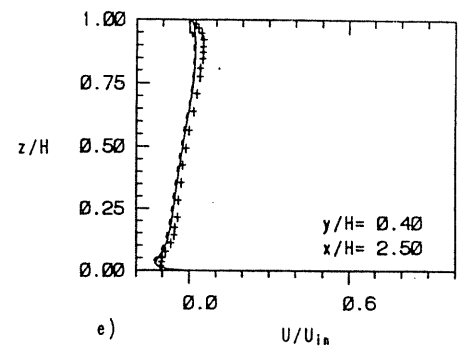


Figure 5. Profiles of the U-velocity. Plane y=0.4 x (width of the room). Solid lines: k-ϵ model. Dashed lines: KL1 model. Expts. by Restivo (1979).

File KASECOM.FOR

In this file the problem specific COMMON block KASE is found. The meaning of the FORTRAN-variables are explained below.

EDIN	inlet turbulent dissipation
FLOWIN	inlet mass flow
ITURB	ITURB=0: selects the k-ϵ model; ITURB=2: selects the KL1 model
JINLET	the inlet is covered by the nodes between J=2 and J=JINLET in the y-direction
KINLET	the inlet is covered by the nodes between K=KINLET and NKM1 in the z-direction
NIHALF,NJHALF NKHALF	number of cells from the wall to the centre of the room in the x, y and z direction
TEIN	inlet turbulent kinetic energy
UIN	inlet U-velocity
XL, YL, ZL	turbulent length scale in the x, y, and z direction respectively, used in the KL1 model
XYZL	resulting turbulent length scale used in the KL1 model

Program listings

```
      PROGRAM MAIN
C
C
C*************************************************************************
C
C.............................TEACH3D.....................................
..
C     * A COMPUTER PROGRAM FOR THE CALCULATION OF                       **
C     * THREE-DIMENSIONAL TURBULENT RECIRCULATING                       **
C     * TRANSIENT FLOWS.                                                **
C*************************************************************************
CHAPTER  0  0  0  0  0  0  0  0  PRELIMINARIES  0  0  0  0  0  0  0  0
C
      INCLUDE 'COMMON.FOR'
      INCLUDE 'KASECOM.FOR'
      DIMENSION HEDU(6),HEDV(6),HEDW(6),HEDP(6),HEDT(6),HEDK(6)
     1    ,HEDD(6),HEDM(6),HEDA(6),HEDB(6)
     2    ,HEDX(6),HEDY(6),HEDZ(6),HEDXYZ(6),HEDR(6)
      DATA HEDU/4HU  V,4HELOC,4HITY ,4H    ,4H    ,4H    /
      DATA HEDV/4HV  V,4HELOC,4HITY ,4H    ,4H    ,4H    /
      DATA HEDW/4HW  V,4HELOC,4HITY ,4H    ,4H    ,4H    /
      DATA HEDP/4HPRES,4HSURE,4H    ,4H    ,4H    ,4H    /
      DATA HEDT/4HTEMP,4HERAT,4HURE ,4H    ,4H    ,4H    /
      DATA HEDK/4HTURB,4HULEN,4HCE E,4HNERG,4HY   ,4H    /
      DATA HEDD/4HENER,4HGY D,4HISSI,4HPATI,4HON  ,4H    /
      DATA HEDM/4HVISC,4HOSIT,4HY   ,4H    ,4H    ,4H    /
      DATA HEDX/4HLENG,4HTH S,4HCALE,4H/X  ,4H    ,4H    /
      DATA HEDY/4HLENG,4HTH S,4HCALE,4H/Y  ,4H    ,4H    /
      DATA HEDZ/4HLENG,4HTH S,4HCALE,4H/Z  ,4H    ,4H    /
      DATA HEDR/4HDENS,4HITY ,4H    ,4H    ,4H    ,4H    /
      DATA HEDXYZ/4HLENG,4HTH S,4HCALE,4H/XYZ,4H    ,4H    /
      LOGICAL INCALU,INCALV,INCALW,INCALP,INPRO,INCALK,INCALD,
     1        INCALM,INCALA,INCALB,INCALT,RESTRT,SAVEM
C**** k-eps
      ITURB=0
C*** KL1
C      ITURB=2
C-----STORE/RESTART PARAMETERS
      RESTRT=.TRUE.
      SAVEM=.TRUE.
C-----UNSTEADY/STEADY
      STEADY=.TRUE.
      TFIRST=0.
      NFTSTP=1
      NLTSTP=1
      IF(STEADY)NFTSTP=1
      IF(STEADY)NLTSTP=1
C-----ALL
      DATA GREAT/1.E10/
      DATA SMALL/1.E-10/
C-----ITERATION CONTROL PARAMETERS
      NFITER=1
```

```
      NLITER=300
      MAXIT=NLITER-NFITER
      NSWPU=1
      NSWPV=1
      NSWPW=1
      NSWPP=3
      NSWPK=1
      NSWPD=1
      NSWPT=1
C
CHAPTER  1  1  1  1  1  PARAMETERS AND CONTROL INDICES  1  1  1  1  1  1
C
C-----GRID
      NI=18
      NJ=15
      NK=19
      NIM1=NI-1
      NJM1=NJ-1
      NKM1=NK-1
      KINLET=14  !lowest cell in the inlet
      KOUTLET=4  !highest cell in the outlet
      JINLET=4   !northest ell in the inlet
      DATA X/-.05,.05,.15,.25,.35,.45,.55,.65,.8,.95,1.05,1.35,
     &1.65,1.95,2.25,2.55,2.85,3.15,7*0.0/
      DATA Y/-0.01,.01,.025,0.04,.06,0.09,
     &0.13,0.175,0.225,0.275,0.325,0.375,0.425,0.475,0.525,10*0.0/
      DATA Z/-0.025,0.025,0.075,0.12,0.2,0.33,0.46,0.59,
     &0.67,0.74,0.79,0.84,0.89,0.91,0.93,0.95,0.97,0.99,1.01,6*0.0/
C-----DEPENDENT VARIABLE SELECTION
      INCALU=.TRUE.
      INCALV=.TRUE.
      INCALW=.TRUE.
      INCALP=.TRUE.
      INCALK=.TRUE.
      INCALD=.TRUE.
      IF(ITURB.NE.0)INCALD=.FALSE.
      INPRO=.TRUE.
      INCALT=.FALSE.
C-----FLUID PROPERTIES
      DENSIT =1.189
      PRANDL=0.72
      VISCOS =18.1E-6
C-----TURBULENCE CONSTANTS
      IF(ITURB.EQ.0)THEN
      CMU=0.09
      CD=1.00
      END IF
      IF(ITURB.NE.0)THEN
      CMU=0.5477
      CD=0.1643
      END IF
      CMUCD=CMU*CD
      C1=1.44
      C2=1.92
```

```
      CAPPA=0.435
      ELOG=9.0
      PRED=CAPPA*CAPPA/(C2-C1)/(CMU**.5)
      PRTE=1.0
      PRANDT=0.9
      PFUN=PRANDL/PRANDT
      PFUN=9.24*(PFUN**0.75-1.0)*(1.0+0.28*EXP(-0.007*PFUN))
C-----BOUNDARY VALUES
      UIN=1.368
      TURBIN=0.04
      TEIN=(TURBIN*UIN)**2
      EDIN=0.1643*TEIN**1.5/0.01
C-----PRESSURE CALCULATION
      IPREF=10
      JPREF=5
      KPREF=5
C-----PROGRAM CONTROL AND MONITOR
      IMON   =5
      JMON   =5
      KMON   =5
      URFU=0.5
      URFV=0.5
      URFW=0.5
      URFP=1.0
      URFE=0.5
      URFK=0.5
      URFT=1.0
      URFVIS=0.5
      INDMON=10
      INDPRT=1
      INDPRI =1000
      SORMAX =0.01
C
CHAPTER  2  2  2  2  2  2  INITIAL OPERATIONS  2  2  2  2  2  2  2  2  2
C
C-----CALCULATE GEOMETRICAL QUANTITIES AND SET VARIABLES TO ZERO
      CALL INIT
C**** search NIHALF & NKHALF
      DO 2004 I=1,NI
 2004 IF(X(I).GE.XU(NI)/2.AND.X(I-1).LT.XU(NI)/2)NIHALF=I
      DO 2006 K=1,NK
 2006 IF(Z(K).GE.ZW(NK)/2.AND.Z(K-1).LT.ZW(NK)/2)NKHALF=K
C-----INITIALISE VARIABLE FIELDS
      DO 2000 I=1,NI
      DO 2000 J=1,NJ
      DO 2000 K=1,NK
      TE(I,J,K)=1.E-3
      ED(I,J,K)=3.E-4
 2000 CONTINUE
      DO  203  K=1,NK
      DO  203  J=1,NJ
      XPLUSE(J,K)=11.0
 203  XPLUSW(J,K)=11.0
      DO  204  K=1,NK
```

```
      DO  204  I=1,NI
      YPLUSS(I,K)=11.0
 204  YPLUSN(I,K)=11.0
      DO  205  J=1,NJ
      DO  205  I=1,NI
      ZPLUSL(I,J)=11.0
 205  ZPLUSH(I,J)=11.0
      IF(RESTRT)CALL RESTR1(INPRO)
      IF(INCALU.AND..NOT.STEADY) CALL UPDATE(U,UO,NI,NJ,NK,IT,JT,KT)
      IF(INCALV.AND..NOT.STEADY) CALL UPDATE(V,VO,NI,NJ,NK,IT,JT,KT)
      IF(INCALV.AND..NOT.STEADY) CALL UPDATE(W,WO,NI,NJ,NK,IT,JT,KT)
      IF(INCALK.AND..NOT.STEADY) CALL UPDATE(TE,TEO,NI,NJ,NK,IT,JT,KT)
      IF(INCALD.AND..NOT.STEADY) CALL UPDATE(ED,EDO,NI,NJ,NK,IT,JT,KT)
      IF(INCALT.AND..NOT.STEADY) CALL UPDATE(T,TO,NI,NJ,NK,IT,JT,KT)
      IF(INCALT.AND..NOT.STEADY) CALL UPDATE(DEN,DENO,NI,NJ,NK,IT,JT,KT)
C-----INITIAL OUTPUT
C      OPEN(UNIT=16,FILE='OUTDAT',FORM='FORMATTED',STATUS='NEW')
      WRITE(6,210) DENSIT
      WRITE(6,211) VISCOS
      WRITE(6,212)
      WRITE(6,213)
      WRITE(6,214)URFU,URFV,URFW,URFP,URFT,URFK,URFE
      IF(INCALU) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,XU,Y,Z,U,HEDU)
      IF(INCALV) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,YV,Z,V,HEDV)
      IF(INCALW) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,ZW,W,HEDW)
      IF(INCALP) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,P,HEDP)
      IF(INCALK) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,TE,HEDK)
      IF(INCALD) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,ED,HEDD)
      IF(INPRO) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,VIS,HEDM)
      IF(INCALT) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,T,HEDT)
      IF(INCALP) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,DEN,HEDP)
C.......CALCULATE RESIDUAL SOURCES NORMALIZATION FACTORS.............
      FLOWIN=DENSIT*0.1*0.05*UIN
      XMOMIN=FLOWIN*UIN
      HEATIN=0.002
C
CHAPTER  3  3  3  3  3  3  3  ITERATION LOOP  3  3  3  3  3  3  3  3  3
C
C-----TIME STEP LOOP
      TIME=TFIRST
      DO 3000 ITSTEP=NFTSTP,NLTSTP
      NITER=0
      IF(.NOT.STEADY)TIME=TIME+DT(ITSTEP)
      IF(.NOT.STEADY)WRITE(6,404)
      IF(.NOT.STEADY)WRITE(6,405)TIME,DT(ITSTEP),ITSTEP
C-----ITERATION LOOP
  300 NITER=NITER+1
      IF(INCALU) CALL CALCU
      IF(INCALV) CALL CALCV
      IF(INCALW) CALL CALCW
      IF(INCALP) CALL CALCP
      IF(INCALK) CALL CALCTE
      IF(INCALD) CALL CALCED
      IF(INCALT) CALL CALCT
```

```
C-----UPDATE FLUID PROPERITIES
      IF(INPRO) CALL PROPS
C-----INTERMEDIATE OUTPUT
      RESORM =RESORM/FLOWIN
      RESORU =RESORU/XMOMIN
      RESORV  =RESORV/XMOMIN
      RESORW  =RESORW/XMOMIN
      RESORK  =RESORK/FLOWIN
      RESORE  =RESORE/FLOWIN
      RESORT =RESORT/HEATIN
      IF(MOD(NITER+NFITER,INDMON).NE.0) GO TO 320
      WRITE(6,310)NITER+NFITER,RESORU,RESORV,RESORW,RESORM,RESORT,
     &RESORK,RESORE
      WRITE(6,312) IMON,JMON,KMON,U(IMON,JMON,KMON),V(IMON,JMON,KMON),
     2W(IMON,JMON,KMON),P(IMON,JMON,KMON),T(IMON,JMON,KMON),
     3TE(IMON,JMON,KMON),ED(IMON,JMON,KMON)
 320  IF(MOD(NITER+NFITER,INDPRI).NE.0.OR.NITER.EQ.MAXIT) GO TO 301
      IF(INCALU) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,XU,Y,Z,U,HEDU)
      IF(INCALV) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,YV,Z,V,HEDV)
      IF(INCALW) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,ZW,W,HEDW)
      IF(INCALP) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,P,HEDP)
      IF(INCALK) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,TE,HEDK)
      IF(INCALD) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,ED,HEDD)
      IF(INPRO) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,VIS,HEDM)
      IF(INCALT) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,T,HEDT)
C-----TERMINATION TESTS
  301 SORCE=AMAX1(RESORM,RESORU,RESORV,RESORW,RESORT)
      IF(NITER.EQ.20.AND.SORCE.GT.1.0E4*SORMAX) GO TO  3002
      IF(NITER.EQ.MAXIT) GO TO 302
      IF(SORCE.GT.SORMAX) GO TO 300
  302 IF(STEADY)GOTO 3002
C-----INTERMEDIATE OUTPUT OF FIELDS (TRANSIENT CALC.)
      IF(MOD(ITSTEP,INDPRT).NE.0.OR.ITSTEP.EQ.NLTSTP) GO TO 3003
      IF(INCALU) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,XU,Y,Z,U,HEDU)
      IF(INCALV) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,YV,Z,V,HEDV)
      IF(INCALW) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,ZW,W,HEDW)
      IF(INCALP) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,P,HEDP)
      IF(INCALK) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,TE,HEDK)
      IF(INCALD) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,ED,HEDD)
      IF(INPRO) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,VIS,HEDM)
      IF(INCALT) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,T,HEDT)
C-----UPDATE VARIABELS
 3003 IF(INCALU) CALL UPDATE(U,UO,NI,NJ,NK,IT,JT,KT)
      IF(INCALV) CALL UPDATE(V,VO,NI,NJ,NK,IT,JT,KT)
      IF(INCALV) CALL UPDATE(W,WO,NI,NJ,NK,IT,JT,KT)
      IF(INCALK) CALL UPDATE(TE,TEO,NI,NJ,NK,IT,JT,KT)
      IF(INCALD) CALL UPDATE(ED,EDO,NI,NJ,NK,IT,JT,KT)
      IF(INCALT) CALL UPDATE(DEN,DENO,NI,NJ,NK,IT,JT,KT)
      IF(INCALT) CALL UPDATE(T,TO,NI,NJ,NK,IT,JT,KT)
 3000 CONTINUE
C
CHAPTER  4  4  4  4  4  4  FINAL OPERATIONS AND OUTPUT  4  4  4  4  4  4
C
 3002 CONTINUE
```

```
      IF(.NOT.STEADY)WRITE(6,404)
      IF(.NOT.STEADY)WRITE(6,405)TIME,DT(NLTSTP),NLTSTP
      IF(SAVEM)CALL SAVE1
      IF(INCALU) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,XU,Y,Z,U,HEDU)
      IF(INCALV) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,YV,Z,V,HEDV)
      IF(INCALW) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,ZW,W,HEDW)
      IF(INCALP) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,P,HEDP)
      IF(INCALK) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,TE,HEDK)
      IF(INCALD) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,ED,HEDD)
      IF(INPRO) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,VIS,HEDM)
      IF(INCALT) CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,T,HEDT)
      CALL PRINT(2,2,2,NI,NJ,NK,IT,JT,KT,X,Y,Z,XYZL,HEDXYZ)
C      CLOSE(UNIT=16)
      STOP
C-----FORMAT STATEMENTS
  210 FORMAT(1X,15X,'FLUID DENSITY ',T60,1H=,3X,1PE11.3)
  211 FORMAT(1X,14X,' LAMINAR VISCOSITY ',T60,1H=,3X,1PE11.3)
  212 FORMAT(1X,//'I',28('-'),'RELAXATION PARAMETERS',29('-'),'I')
  213 FORMAT(6X,'URFU',5X,'URFV',5X,'URFW',5X,'URFP',5X,'URFT',5X,
     &'URFTE',5X,'URFE')
  214 FORMAT(5X,7(F5.2,4X))
  310 FORMAT(1X,///1X,79('*')/'ITER    ','I',17('-'),'ABSOLUTE RESIDUAL
     1 SOURCE SUMS',25('-'),'I' / 2X,'NO.',5X,'UMOM',6X,'VMOM',6X,
     2'WMOM',6X,'MASS',6X,'TEMP',6X,'TKIN',6X,'DISP',
     3/1X,I4,2X,1P7E10.3)
  312 FORMAT(1X,/'I',15('-'),'FIELD VALUES AT MONITORING LOCATION',
     12X,'(',I2,',',I2,',',I2,')',16('-'),'I' / 6X,'U',10X,'V',10X,'W',
     210X,'P',10X,'T',10X,'K',10X,'D'/1X,1P7E11.3)
  404 FORMAT(//1X,5('*'),1X,'TIME',7X,'DELTA T',6X,'TIME STEP NO.',
     11X,5('*'))
  405 FORMAT(/3X,1PE10.2,3X,1PE10.2,6X,I5)
      END
      SUBROUTINE PROPS
C
CHAPTER  0  0  0  0  0  0  0  0  PRELIMINARIES  0  0  0  0  0  0  0  0
C
      INCLUDE 'COMMON.FOR'
      INCLUDE 'KASECOM.FOR'
      IF(ITURB.EQ.0.)GOTO 200
CHAPTER   1   1   1    LENGTH SCALE   1   1   1
C BOUNDARY CONDITIONS
      BCW=(X(2)-XU(2))*CAPPA
      BCE=(XU(NI)-X(NIM1))*CAPPA
      BCS=(Y(2)-YV(2))*CAPPA
      BCN=(YV(NJ)-Y(NJM1))*CAPPA
      BCL=(Z(2)-ZW(2))*CAPPA
      BCH=(ZW(NK)-Z(NKM1))*CAPPA
      DO 1020 K=2,NKM1
      DO 1020 J=2,NJM1
C WEST WALL
      XL(2,J,K)=BCW
      YL(2,J,K)=BCW
      ZL(2,J,K)=BCW
C EAST WALL
```

```
      ZL(NIM1,J,K)=BCE
      YL(NIM1,J,K)=BCE
 1020 XL(NIM1,J,K)=BCE
      DO 103 K=2,NKM1
      DO 103 I=2,NIM1
C NORTH WALL
      ZL(I,NJM1,K)=BCN
      XL(I,NJM1,K)=BCN
 103  YL(I,NJM1,K)=BCN
      DO 104 J=2,NJM1
      DO 104 I=2,NIM1
C LOW WALL
      XL(I,J,2)=BCL
      YL(I,J,2)=BCL
      ZL(I,J,2)=BCL
C NORTH WALL
      XL(I,J,NKM1)=BCH
      YL(I,J,NKM1)=BCH
 104  ZL(I,J,NKM1)=BCH
      XLMAX=.09*XU(NI)
      YLMAX=.09*YV(NJ)
      ZLMAX=.09*ZW(NK)
      DO 105 J=3,NJ-2
      DO 105 K=3,NK-2
C WEST WALL
      GI=0.
      DO 110 I=3,NIHALF
      GI=GI+SEW(I)/SQRT(TE(I,J,K))
 110  XL(I,J,K)=AMIN1(SQRT(TE(I,J,K))*GI+BCW,XLMAX)
C EAST WALL
      GI=0.
      DO 111 I=NI-2,NIHALF+1,-1
      GI=GI+SEW(I)/SQRT(TE(I,J,K))
 111  XL(I,J,K)=AMIN1(SQRT(TE(I,J,K))*GI+BCE,XLMAX)
 105  CONTINUE
      DO 120 K=3,NK-2
      DO 120 I=3,NI-2
C NORTH WALL
      GI=0.
      DO 131 J=NJ-2,2,-1
      GI=GI+SNS(J)/SQRT(TE(I,J,K))
 131  YL(I,J,K)=AMIN1(SQRT(TE(I,J,K))*GI+BCN,YLMAX)
 120  CONTINUE
      DO 140 J=3,NJ-2
      DO 140 I=3,NI-2
C LOW WALL
      GI=0.
      DO 150 K=3,NKHALF
      GI=GI+SHL(K)/SQRT(TE(I,J,K))
 150  ZL(I,J,K)=AMIN1(SQRT(TE(I,J,K))*GI+BCL,ZLMAX)
C HIGH WALL
      GI=0.
      DO 151 K=NK-2,NKHALF+1,-1
      GI=GI+SHL(K)/SQRT(TE(I,J,K))
```

```
 151  ZL(I,J,K)=AMIN1(SQRT(TE(I,J,K))*GI+BCH,ZLMAX)
 140  CONTINUE
C IN THE CENTRE
      DO 134 K=1,NK
      DO 134 J=1,NJ
      XLCENTRE=0.5*(XL(NIHALF,J,K)+XL(NIHALF-1,J,K))
      XL(NIHALF,J,K)=XLCENTRE
 134  XL(NIHALF-1,J,K)=XLCENTRE
      DO 136 J=1,NJ
      DO 136 I=1,NI
      ZLCENTRE=0.5*(ZL(I,J,NKHALF)+ZL(I,J,NKHALF-1))
      ZL(I,J,NKHALF)=ZLCENTRE
 136  ZL(I,J,NKHALF-1)=ZLCENTRE
C CALCULATE XYZL
      DO 191 K=1,NK
      DO 191 J=1,NJ
      DO 191 I=1,NI
 191  XYZL(I,J,K)=AMIN1(XL(I,J,K),YL(I,J,K),ZL(I,J,K),0.09*0.5)
C
CHAPTER   1   1   1   VISCOSITY   1   1   1
C
 200  CONTINUE
      DO 100 K=2,NKM1
      DO 100 J=2,NJM1
      DO 100 I=2,NIM1
      VISOLD=VIS(I,J,K)
      IF(ITURB.EQ.0.AND.ED(I,J,K).EQ.0.) GO TO 102
      IF(ITURB.EQ.0)
     &VIS(I,J,K)=DEN(I,J,K)*TE(I,J,K)**2*CMU/ED(I,J,K)+VISCOS
      IF(ITURB.GT.0)
     &VIS(I,J,K)=DEN(I,J,K)*SQRT(TE(I,J,K))*CMU*XYZL(I,J,K)+VISCOS
      GO  TO  101
  102 VIS(I,J,K)=VISCOS
C-----UNDER-RELAX VISCOSITY
  101 VIS(I,J,K)=URFVIS*VIS(I,J,K)+(1.-URFVIS)*VISOLD
      GAMH(I,J,K)=VISCOS/PRANDL+(VIS(I,J,K)-VISCOS)/PRANDT
  100 CONTINUE
CHAPTER 2  2  2  2  2  PROBLEM MODIFICATIONS  2  2  2  2  2  2  2  2  2  2
      CALL MODPRO
      RETURN
      END
      SUBROUTINE PROMOD
C
CHAPTER  0  0  0  0  0  0  0  PRELIMINARIES  0  0  0  0  0  0  0  0  0
C
      INCLUDE 'COMMON.FOR'
      INCLUDE 'KASECOM.FOR'
CHAPTER  1  1  1  1  1  1  1  1  PROPERTIES  1  1  1  1  1  1  1  1  1
C
      ENTRY MODPRO
      IF(ITURB.EQ.0)RETURN
      DO 100 K=KINLET-3,NKM1
      DO 100 J=2,JINLET+3
      DO 100 I=2,11
```

```
      VIST=0.003*X(I)*(U(I,J,NKM1)+U(I+1,J,NKM1))/2.
      VIS(I,J,K)=VIST+VISCOS
  100 CONTINUE
      RETURN
C
CHAPTER  2  2  2  2  2  2  2  2  U MOMENTUM  2  2  2  2  2  2  2  2  2
C
      ENTRY MODU
C
C SYMMETRY PLANE
      CALL WALL('U','SOUTH',2,NIM1,2,2,2,NKM1,-101.)
C
      CALL WALL('U','NORTH',2,NIM1,NJM1,NJM1,2,NKM1,0.)
C
      CALL WALL('U','LOW',2,NIM1,2,NJM1,2,2,0.)
      CALL WALL('U','HIGH',2,NIM1,2,NJM1,NKM1,NKM1,0.)
C INLET
      DO 200 K=KINLET,NKM1
      DO 200 J=2,JINLET
  200 U(2,J,K)=UIN
C OUTLET
      DATA AREOUT/0.08/
      UOUT=FLOWIN/AREOUT/DEN(3,3,3)
      DO 201 K=2,KOUTLET
      DO 201 J=2,NJM1
  201 U(NI,J,K)=UOUT
      CALL WALL('U','EAST',NIM1,NIM1,2,NJM1,2,KOUTLET,-101.)
C INLET
                  RETURN
CHAPTER  3  3  3  3  3  3  3  3  V MOMENTUM  3  3  3  3  3  3  3  3  3
C
      ENTRY MODV
      CALL WALL('V','WEST',2,2,JINLET+1,NJM1,2,NKM1,0.)
      CALL WALL('V','WEST',2,2,2,JINLET,2,KINLET-1,0.)
      CALL WALL('V','EAST',NIM1,NIM1,2,NJM1,KOUTLET+1,NKM1,0.)
C
C SYMMETRY PLANE
      CALL WALL('V','SOUTH',2,NIM1,2,2,2,NKM1,-101.)
C
      CALL WALL('V','LOW ',2,NIM1,2,NJM1,2,2,0.)
      CALL WALL('V ','HIGH ',2,NIM1,2,NJM1,NKM1,NKM1,0.)
C OUTLET
      CALL WALL('V','EAST',NIM1,NIM1,2,NJM1,2,KOUTLET,-101.)
      RETURN
CHAPTER  4  4  4  4  4  4  4  4  W MOMENTUM  4  4  4  4  4  4  4  4  4
C
      ENTRY MODW
      CALL WALL('W','WEST',2,2,JINLET+1,NJM1,2,NKM1,0.)
      CALL WALL('W','WEST',2,2,2,JINLET,2,KINLET-1,0.)
      CALL WALL('W','EAST ',NIM1,NIM1,2,NJM1,KOUTLET+1,NKM1,0.)
C
C SYMMETRY PLANE
      CALL WALL('W','SOUTH',2,NIM1,2,2,2,NKM1,-101.)
C
```

```
      CALL WALL('W','NORTH',2,NIM1,NJM1,NJM1,2,NKM1,0.)
C
C OUTLET
      CALL WALL('W','EAST',NIM1,NIM1,2,NJM1,2,KOUTLET,-101.)
      RETURN
C
CHAPTER  5  5  5  5  5  5  PRESSURE CORRECTION  5  5  5  5  5  5  5  5
C
      ENTRY MODP
      RETURN
C
CHAPTER  6  6  6  6  6  6  6  TEMPERATURE  6  6  6  6  6  6  6  6  6  6
C
      ENTRY MODT
      RETURN
C
CHAPTER  7  7  7  7  7  TURBULENT KINETIC ENERGY  7  7  7  7  7  7  7
C
      ENTRY MODTE
      CALL WALL('TE','WEST',2,2,JINLET+1,NJM1,2,NKM1,0.)
      CALL WALL('TE','WEST',2,2,2,JINLET,2,KINLET-1,0.)
      CALL WALL('TE','EAST',NIM1,NIM1,2,NJM1,KOUTLET+1,NKM1,0.)
C
C SYMMETRY PLANE
      CALL WALL('TE','SOUTH',2,NIM1,2,2,2,NKM1,-101.)
C
      CALL WALL('TE','NORTH',2,NIM1,NJM1,NJM1,2,NKM1,0.)
C
      CALL WALL('TE','LOW',2,NIM1,2,NJM1,2,2,0.)
      CALL WALL('TE','HIGH',2,NIM1,2,NJM1,NKM1,NKM1,0.)
C INLET
      DO 700 K=KINLET,NKM1
      DO 700 J=2,JINLET
 700  TE(1,J,K)=TEIN
C OUTLET
      CALL WALL('TE','EAST',NIM1,NIM1,2,NJM1,2,KOUTLET,-101.)
      RETURN
C
CHAPTER  8  8  8  8  8  8   DISSIPATION  8  8  8  8  8  8  8  8  8  8
C
      ENTRY MODED
      CALL WALL('ED','WEST',2,2,JINLET+1,NJM1,2,NKM1,0.)
      CALL WALL('ED','WEST',2,2,2,JINLET,2,KINLET-1,0.)
      CALL WALL('ED','EAST',NIM1,NIM1,2,NJM1,KOUTLET+1,NKM1,0.)
C
C SYMMETRY PLANE
      CALL WALL('ED','SOUTH',2,NIM1,2,2,2,NKM1,-101.)
C
      CALL WALL('ED','NORTH',2,NIM1,NJM1,NJM1,2,NKM1,0.)
C
      CALL WALL('ED','LOW',2,NIM1,2,NJM1,2,2,0.)
      CALL WALL('ED','HIGH',2,NIM1,2,NJM1,NKM1,NKM1,0.)
C INLET
      DO 800 K=KINLET,NKM1
```

```
      DO 800 J=2,JINLET
 800  ED(1,J,K)=EDIN
C OUTLET
      CALL WALL('ED','EAST',NIM1,NIM1,2,NJM1,2,KOUTLET,-101.)
      RETURN
      END

C
      PROGRAM KASECOM
C
      COMMON/KASE/UIN,TEIN,EDIN,KINLET,KOUTLET,JINLET,FLOWIN,
     1       CL,FX(IT),FY(JT),FZ(KT),XL(IT,JT,KT),YL(IT,JT,KT),
     2       ZL(IT,JT,KT),XYZL(IT,JT,KT),ITURB,NIHALF,NJHALF,
     3       NKHALF,NITER
```