Publ. Nr 95/11

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för Termo- och Fluiddynamik

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Thermo and Fluid Dynamics

CALC-BFC

A Finite-Volume Code Employing Collocated Variable Arrangement and Cartesian Velocity Components for Computation of Fluid Flow and Heat Transfer in Complex Three-Dimensional Geometries

by

Lars Davidson and Bijan Farhanieh

Thermo and Fluid Dynamics
Chalmers University of Technology
412 96 Gothenburg, Sweden

Göteborg, November 1995

Summary

CALC-BFC is a finite volume computer code which uses collocated variable arrangement and Cartesian velocity components for prediction of mass, heat, and momentum transfer. The method utilizes a general non-orthogonal boundary fitted coordinate system. The program is written for steady/unsteady, three-dimensional, turbulent/laminar recirculating flows.

Acknowledgements

This report is a 2nd edition of Ref. [1].

Contents

1 Introduction

CALC-BFC (Boundary Fitted Coordinates) is a computer code for computation of two or three- dimensional steady/unsteady, turbulent/laminar recirculating flows. The development of the code was started in 1989 [2]. The computation is based upon the solution of the partial differential equations governing the flow. The equations are written in a non-orthogonal coordinate system. The finite-volume method is applied to transform the partial differential equations to algebraic relations which link the values of the dependent variables at the nodes of the computational grid. The TDMA (Tri-Diagonal

Matrix Algorithm) is then adapted to solve the obtained algebraic relations. Three different differencing schemes are available to approximate the convective fluxes: Hybrid central/upwind differencing, the QUICK scheme, or the Van Leer scheme. SIMPLEC [3] handles the linkage between velocities and pressure.

The Cartesian velocity components are used in the program. This method have been used by Shyy *et al.*[4] and Braaten and Shyy [5]. In contrast to most of the finite volume codes which use the conventional method of staggered grids for the velocity components, the present code utilizes the collocated variable arrangement in which all variables are stored at the same control volume. This arrangement has various advantages over the staggered grid, e.g. the control volumes for all variables coincide with the boundaries of the solution domain, facilitating the specification of boundary conditions, and geometrical data need to be calculated only for one set of control volumes (when staggered arrangement is used the geometrical data must be calculated for four sets of control volume).

The code can be applied to axi-symmetric and plane flows. The boundaries in a flow can be a wall, a symmetry plane or a boundary along which the values of the dependent variables are prescribed.

In computation of turbulent flows, the effect of turbulence is represented by $k - \varepsilon$ model. This model does not account for the viscous effect near the walls. The wall functions [6] are employed to bridge the viscous sub-layer near wall region.

1.1 Why use non-staggered grid and Cartesian velocity components

The use of staggered grid in SIMPLEC algorithm, requires the solution of momentum equations for covariant or contravariant velocity components on the faces of the control volumes. However, the geometrical data needed to describe a full non- orthogonal grid is large and the storage of such data is expensive, and if staggered velocity grids are used this data must be calculated for four sets of grids components compared with only one in non-staggered arrangement. Moreover, the momentum equations for co- or contravariant velocity components require the computation of covariant derivatives and thereby the Christoffel symbol (the Christoffel symbols can be avoided even in staggered grid by using local coordinate system, see [7, 8]). As the momentum equation contain second derivatives, the second covariant derivatives must be calculated. This results in 81 extra geometrical quantities. These quantities should either be stored or recomputed.

These problems can be avoided through the solution of the momentum equations for velocity components in a fixed Cartesian coordinate system on a non-staggered grid. That is to say, all variables (U, V, W, P etc) are stored at the center of the control volume. In this case when the momentum equations are transferred from physical coordinates x_i to the computational coordinates ξ_i , the velocity components can be calculated as scalars and the need for the Christoffel symbols in the momentum equations vanishes, thus drastically reducing the required amount of geometrical data. Also since we have

only one grid and not four, the programming becomes simpler and the required amount of geometrical data is further reduced.

2 Solution Methodology

In order to extend the capabilities of the finite difference method to deal with complex geometries, a boundary fitted coordinate method is used.

The basic idea in this method is to map the complex flow domain in the physical space to a simple rectangular domain in the computational space by using a curvilinear coordinate transformation. In other words, the Cartesian coordinate system x_i in the physical domain is replaced by a general non-orthogonal system ξ_i.

The momentum equations are solved for the velocity components U, V, W in the fixed Cartesian directions on a non-staggered grid. This means that all the variables are stored at the center of the control volume. This method was suggested and worked out by Rhie and Chow [10] and later used by Burns and Wilkes [11], Majumdar [12], Peric *et al.*[13] and Miller and Schmidt [14]. Majumdar [15] later discussed the importance of underrelaxation in momentum interpolation when non-staggered grids are used.

The steady transport equation for a general dependent variable Φ in the Cartesian coordinates can be written as

$$\frac{\partial}{\partial t}(\rho\Phi) + \frac{\partial}{\partial x_i}(\rho U_i \Phi) = \frac{\partial}{\partial x_i}\left(\Gamma_\Phi \frac{\partial \Phi}{\partial x_i}\right) + S \tag{1}$$

where Γ_Φ is the exchange coefficient and is equal to viscosity in the momentum equations, and is equal to viscosity/Pr in the temperature equation. The dependent variable Φ can be equal U, V, W, T, k, ε etc.

The total flux, convective and diffusive fluxes, is defined as

$$I_i = \rho U_i \Phi - \Gamma_\Phi \frac{\partial \Phi}{\partial x_i} \tag{2}$$

It is now convenient to write eq. (2) in the equivalent form

$$\frac{\partial I_i}{\partial x_i} = S_\phi \tag{3}$$

or, in vector notation

$$\nabla \cdot \vec{I} = S_\phi \tag{4}$$

Integration of eq. (4) over a control volume in the physical space, using Gauss' law, gives

$$\int_A \vec{I} \cdot d\vec{A} = \int_V S_\phi dV \tag{5}$$

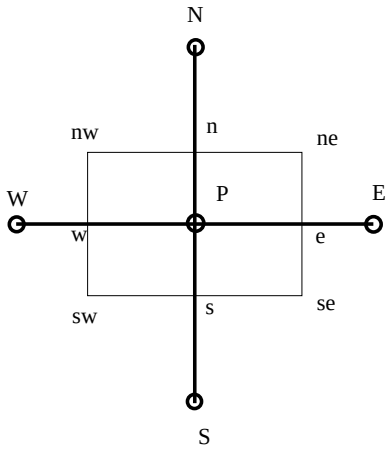

Figure 1: Control volume

Equations (4) and (5) are used for performing the transformation to the computational space coordinates (general non-orthogonal coordinates) ξ_i.

The scalar advection-diffusion equation 5 is discretized. The integration of this gives

$$\left(\vec{I}\cdot\vec{A}\right)_e + \left(\vec{I}\cdot\vec{A}\right)_w + \left(\vec{I}\cdot\vec{A}\right)_n + \left(\vec{I}\cdot\vec{A}\right)_s + \left(\vec{I}\cdot\vec{A}\right)_h + \left(\vec{I}\cdot\vec{A}\right)_\ell = S\delta V \tag{6}$$

where e, w, n, s, h and l refer to the faces of the control volume, see $Fig.1$. The discretized equation is rearranged in the standard form

$$a_P\Phi_P = \sum a_{NB}\Phi_{NB} + S_C \tag{7}$$

where

$$a_P = \sum a_{NB} - S_P \tag{8}$$

The coefficients a_{NB} contains the contribution due to convection and diffusion and the source terms S_P and S_C contains the remaining terms.

2.1 Convection

For the sake of conciseness and simplicity we restrict ourself in this and the following sub-sections only to the east face of the control volume for explanation of the numerical procedure. The total flux $\vec{I}$ contains convective and diffusive fluxes. The first term in the right-hand side of eq. (2) is the convective term. The mass flow rate through the east face can be expressed as the scalar product of the velocity and area vectors multiplied by the density. Thus we have

$$\dot{m}_e = \rho_e\vec{U}_e\cdot\vec{A}_e = \rho_e\left(U_eA_{ex} + V_eA_{ey} + W_eA_{ez}\right) \tag{9}$$

where the Cartesian areas are calculated by

$$A_{ex} = |\vec{A}|_e\vec{n}\cdot e_x; \quad A_{ey} = |\vec{A}|_e\vec{n}\cdot e_y; \quad A_{ez} = |\vec{A}|_e\vec{n}\cdot e_z \tag{10}$$

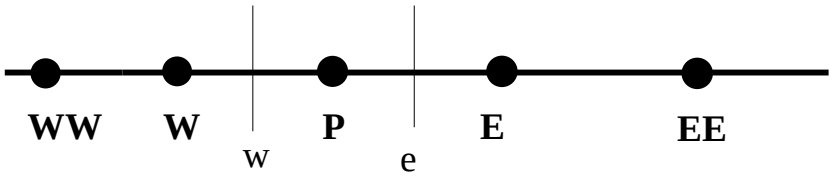

Figure 2: Grid nomenclature

where $|\vec{A}|_e$ is the total area of the east face, $\vec{n}$ its normal vector and $\vec{e}$ the Cartesian base vectors. In order to obtain the velocity components on the control volume faces from those on the control volume centers, the Rhie-Chow [10] interpolation method is used. In this method the weighted linear interpolation in physical space, $U_e = f_x U_E + (1 - f_x) U_P$, is not used in order to avoid non-physical oscillations in pressure and velocity. The method can be described as follows: Consider the interpolation of U to the east face for a control volume centered P, see $Fig.2$. The pressure gradient is subtracted From the velocity field stored at the center of the control volumes we subtract the pressure gradient and obtain a new velocity field $\hat{U}$, i.e.

$$\hat{U}_P = U_P - \frac{-\left(P_e - P_w\right)\delta V}{|\vec{we}|\left(a_P\right)_P} \qquad \text{and} \qquad \hat{U}_E = U_E - \frac{-\left(P_{ee} - P_e\right)\delta V}{|\vec{e(ee)}|\left(a_P\right)_E} \tag{11}$$

where $|\vec{we}|$ denotes the distance between faces e (east) and w (west). The velocity components on the east face is now calculated as

$$U_e = f_x \hat{U}_E + \left(1 - f_x\right)\hat{U}_P + \text{Pressure gradient} \tag{12}$$

$$U_e = f_x \hat{U}_E + \left(1 - f_x\right)\hat{U}_P - \frac{-\left(P_E - P_P\right)\delta V}{|\vec{PE}|\left(a_P\right)_e} \tag{13}$$

where f_x is the interpolation factor and calculated by

$$f_x = \frac{|\vec{Pe}|}{|\vec{Pe}| + |\vec{eE}|} \tag{14}$$

P_e, P_w, P_{ee} in eq. (11) are calculated by linear interpolation. As seen from eq. (13) the pressure gradient is now calculated using the adjacent nodes of the east face. This avoids any non-physical oscillations in the pressure field. V_e and W_e are calculated in a similar manner.

CALC-BFC can calculate the convective fluxes by three different methods. These methods are briefly presented below.

2.1.1 Hybrid Upwind/Central Differencing Scheme

This scheme approximates the convective terms using a central differencing scheme if the local grid Reynolds number is below two, and otherwise upwind differencing, see $Fig.2$.

$$\Phi_e = \Phi_P, \quad \text{if} \quad |Re_{\delta x}| > 2 \quad \text{and} \quad U_e > 0 \tag{15}$$

$$\Phi_e = \Phi_E \quad \text{if} \quad |Re_{\delta x}| > 2 \quad \text{and} \quad U_e < 0 \tag{16}$$

$$\Phi_e = f_x \Phi_E + (1 - f_x)\, \Phi_P \quad \text{if} \quad |Re_{\delta x}| < 2 \tag{17}$$

where f_x is an interpolation factor equal to 0.5 if the face e lies midway between P and E.

2.1.2 Quadratic Upstream-Weighted Interpolation

This scheme of Leonard [16] utilizes a polynomial of the second order fitted to three nodes, two located upstream of the face, and one node located downstream; the scheme is thus a form of upwind scheme. It combines the accuracy of central differencing (second order accuracy) with the inherent stability of upwind differencing (due to its upwind character). For a uniform grid the Φ_e is approximated as (see $Fig.2$):

$$\Phi_e = -0.125\Phi_W + 0.75\Phi_P + 0.375\Phi_E \quad \text{if} \quad U_e > 0 \tag{18}$$

$$\Phi_e = 0.375\Phi_P + 0.75\Phi_E - 0.125\Phi_{EE} \quad \text{if} \quad U_e < 0 \tag{19}$$

This scheme is unbounded which means that it can give rise to unphysical oscillations. The resulting form of the coefficients for a non-uniform grid is given in [9].

2.1.3 The Van Leer Scheme

This scheme of Van Leer [17] is of second order accuracy, except at local minima or maxima where its accuracy is of first order. One advantage for this scheme is that it is bounded.

For face east it can be written, see $Fig.2$

$$\begin{aligned} U_e &> 0 \\ \text{if } |\Phi_E - 2\Phi_P + \Phi_W| &\geq |\Phi_E - \Phi_W| \Longrightarrow \Phi_e = \Phi_P \\ \text{else } \Phi_e &= \Phi_P + \frac{(\Phi_E - \Phi_P)\,(\Phi_P - \Phi_W)}{\Phi_E - \Phi_W} \\ U_e &< 0 \\ \text{if } |\Phi_P - 2\Phi_E + \Phi_{EE}| &< |\Phi_P - \Phi_{EE}| \Longrightarrow \Phi_e = \Phi_E \\ \text{else } \Phi_e &= \Phi_E + \frac{(\Phi_P - \Phi_E)\,(\Phi_E - \Phi_{EE})}{\Phi_P - \Phi_{EE}} \end{aligned}$$

This scheme is thus a first-order upwind with a correction term which gives it second order accuracy.

2.2 Diffusion

The second term in the total flux $\vec{I}$ presented in eq. (2), is the diffusion term. Through an area $\vec{A}$ we have

$$\left(\vec{I}\cdot\vec{A}\right)_{diff} = -\Gamma\vec{A}\cdot\nabla\Phi \tag{20}$$

$\vec{A}\cdot\nabla\Phi$ in eq. (20) can for the east face be rewritten in Cartesian coordinates as

$$\left(-\vec{A}\cdot\nabla\Phi\right)_e = -\left(A_{ex}\frac{\partial\Phi}{\partial x} + A_{ey}\frac{\partial\Phi}{\partial y} + A_{ez}\frac{\partial\Phi}{\partial z}\right)_e \tag{21}$$

and in general non-orthogonal coordinates

$$\left(-\vec{A}\cdot\nabla\Phi\right)_e = -\left(\vec{A}\cdot\vec{g}_i g^{ij}\frac{\partial\Phi}{\partial\xi_j}\right)_e = -\left(|\vec{A}|\vec{n}\cdot\vec{g}_i g^{ij}\frac{\partial\Phi}{\partial\xi_j}\right)_e \tag{22}$$

where $\vec{g}_i$ is the covariant (tangent) unit base vector. The appearance of the metric tensor, g^{ij} in eq. (22), is due to the fact that the components of the product $\vec{A}\cdot\vec{g}_i$ and the derivative $\frac{\partial\Phi}{\partial\xi_j}$ are both covariant and the product of their contravariant base vectors is not zero for $i \neq j$ since they are non-orthogonal to each other. The components of g^{ij} can be calculated as shown in e.g. [18].

The normal vector $\vec{n}$ $(= \vec{g}^3)$ in eq. (22) is equal to the cross product of $\vec{g}_2$ and $\vec{g}_3$ which implies that $\vec{n}\cdot\vec{g}_2 \equiv \vec{n}\cdot\vec{g}_3 = 0$, since by definition a co- and contravariant vector are orthogonal to each other $(\vec{g}^i\vec{g}_j = \delta^i_j)$. Eq. (22) can now be written as

$$\left(-\vec{A}\cdot\nabla\Phi\right)_e = -\left(|\vec{A}|\vec{n}\cdot\vec{g}_1 g^{1j}\frac{\partial\Phi}{\partial\xi_j}\right)_e \tag{23}$$

The diffusion can be written

$$\left(-\vec{A}\cdot\nabla\Phi\right)_e = D_{e\xi}\left(\Phi_E - \Phi_P\right) + D_{e\eta}\left(\Phi_{en} - \Phi_{es}\right) + D_{e\zeta}\left(\Phi_{eh} - \Phi_{el}\right) \tag{24}$$

where the geometrical arrays $D_{e\xi}$, $D_{e\eta}$ and $D_{e\zeta}$ are calculated once and stored.

2.3 Pressure correction equation

The pressure correction equation is obtained by applying the SIMPLEC algorithm [3] on the non-staggered grid. The mass flux $\dot{m}$ is divided into one old value, $\dot{m}^*$, and another correction value, $\dot{m}^{'}$,. The mass flux correction at the east face can be calculated by

$$\dot{m}'_e = \left(\rho\vec{A}\cdot\vec{U}'\right)_e = \rho_e\left(A_{ex}U'_e + A_{ey}V'_e + A_{ez}W'_e\right) = \left(\rho\vec{A}\cdot\vec{g}^j U'_j\right)_e \tag{25}$$

where $U_j^{'}$ is the covariant correction velocity. The covariant velocity components are related to the pressure gradient [8]

$$U_j^{'} = -\frac{\delta v}{a_P}\frac{\partial p^{'}}{\partial x_j} \tag{26}$$

By introducing eq. (25) into eq. (26) we obtain

$$\dot{m}^{'} = \left[\rho\vec{A}\cdot\left(-\frac{\delta v}{a_P}\frac{\partial p^{'}}{\partial x_j}\vec{g}^{j}\right)\right]_e = -\left[\frac{\delta v\rho}{a_P}\vec{A}\cdot\nabla p^{'}\right] \tag{27}$$

Consider, for simplicity, the continuity equation in one dimension

$$\dot{m}_e - \dot{m}_w = 0 \tag{28}$$

If $\dot{m} = \dot{m}^* + \dot{m}^{'}$ and eq. (27) are substituted into eq. (28) we obtain

$$\left[\frac{\delta v\rho}{a_P}\vec{A}\cdot\nabla p^{'}\right]_w - \left[\frac{\delta v\rho}{a_P}\vec{A}\cdot\nabla p^{'}\right]_e + \dot{m}_e^* - \dot{m}_w^* = 0 \tag{29}$$

This is a diffusion equation for the pressure correction $p^{'}$. $\vec{A}\cdot\nabla p^{'}$ can be calculated with eq. (23) by replacing Φ by $p^{'}$.

2.4 The k-ε turbulence model

Adapting the famous Reynolds method, the instantaneous values are decomposed into time-average (U_i, Θ) and fluctuating (u_i, θ) components. The time average momentum and energy equations can be written

$$\frac{\partial}{\partial t}(\rho U_i) + \frac{\partial}{\partial x_j}(\rho U_i U_j) = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right] + \frac{\partial}{\partial x_j}(-\rho\overline{u_i u_j}) \tag{30}$$

$$\frac{\partial}{\partial t}(\rho\Theta) + \frac{\partial}{\partial x_j}(\rho U_j\Theta) = \frac{\partial}{\partial x_j}\left[\frac{\mu}{\sigma_\Theta}\frac{\partial\Theta}{\partial x_j} + \frac{\partial}{\partial x_j}\left(-\rho\overline{u_j\theta}\right)\right] \tag{31}$$

The above equations contain unknown Reynolds stresses $\rho\overline{u_i u_j}$ and scalar fluxes $\rho\overline{u_j\theta}$. These turbulent diffusional fluxes play a significant role in determining the flow behavior as the effects of turbulence are expressed through them. These unknown should be modelled in order to close the equation system.

In the eddy viscosity turbulent model $(k-\varepsilon)$, the unknown turbulent diffusional fluxes are expressed by means of gradient transport hypothesis in which the fluxes are assumed proportional to the gradient of mean flow properties. For example the Reynolds stress tensor is expressed using the Boussinesq eddy viscosity concept

$$\rho\overline{u_i u_j} = -\mu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) + \frac{2}{3}\delta_{ij}\rho k \tag{32}$$

$$\rho\overline{u_j\theta} = -\frac{\mu_t}{\sigma_t}\frac{\partial\Theta}{\partial x_j} \tag{33}$$

σ_t in eq. (33) is often assumed as constant and from dimensional analysis μ_t is found to be a function of the turbulent energy k and its dissipation rate ε i.e.

. $\qquad \mu_t \sim \rho v l; \qquad l \sim \frac{k^{1.5}}{\varepsilon}; \qquad \Rightarrow \mu_t = C_\mu \rho \frac{k^2}{\varepsilon}$

Two partial differential equations for the two new unknowns, k and ε can be derived

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_j}(\rho U_j k) = \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] + P_k - \rho\varepsilon \tag{34}$$

$$\frac{\partial}{\partial t}(\rho \varepsilon) + \frac{\partial}{\partial x_j}(\rho U_j \varepsilon) = \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial \varepsilon}{\partial x_j}\right] + \frac{\varepsilon}{k}(C_{\varepsilon 1} P_k - C_{\varepsilon 2}\rho\varepsilon) \tag{35}$$

where the production term P_k and the constants have the form

$$P_k = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\frac{\partial U_i}{\partial x_j} \tag{36}$$

where $C_\mu = 0.09$, $C_{\varepsilon 1} = 1.44$, $C_{\varepsilon 2} = 1.92$, $\sigma_k = 1.0$ and $\sigma_\varepsilon = 1.31$.

3 Boundary Conditions

3.1 Inlet

3.1.1 Velocities

All flow properties must be known and prescribed along this boundary. The distribution of the inlet velocity (e.g. at the west boundary), U can either be interpolated from experimental data or otherwise evaluated, for example, from an assumed fully developed profile. The transverse velocity components V and W are usually set to zero at the inlet of the computational domain, in consistence with a parallel flow assumption. They can be specified from experimental data when these are available. In case of turbulent flows the fully developed velocity profile can be prescribed from the $\frac{1}{7}$-law.

Since the velocities across the inlet boundary are prescribed, they are never corrected or perturbated, and hence the streamwise gradient of the pressure correction P' is zero at the inlet. This is implemented in the computations by setting the coefficient a_W in the pressure-correction equation equal to zero for the column of control volume adjacent to the inlet.

3.1.2 Turbulent Kinetic Energy

The profile of k at the inlet is either given from experimental measurements, or may be estimated from the following two relationships

I) For parallel flow

$$k_{in} = (\gamma U_{in})^2 \tag{37}$$

where γ is the turbulent intensity and can take a value between 0.01 to 0.1 (it is usually set equal to 0.1)

II) For a flow with prescribed profile

$$k_{in} = C_\mu^{-0.5} l_m^2 \left(\frac{\partial U}{\partial y}\right)^2 \tag{38}$$

where l_m is the Prandtl's mixing length and is given by

$$l_m = \max(\kappa y, \lambda \delta) \tag{39}$$

where κ is the von Karman constant ($= 0.41$), and the constant $\lambda = 0.09$ for channel flows and 0.13 for pipe flows, y being the distance from the wall.

Equation (38) stems from the assumption of turbulence-energy equilibrium, i.e. production of turbulent kinetic energy is balanced by its dissipation, $P = \varepsilon$.

3.1.3 Energy Dissipation Rate

The dissipation ε can be expressed in terms of the turbulence energy and turbulence length scale, via:

$$\varepsilon = \frac{k^{1.5}}{L} \tag{40}$$

where L is related to l_m via

$$L = C_\mu^{-0.75} l_m \tag{41}$$

This relationship can be used to estimate the dissipation rate at inlet

$$\varepsilon_{in} = \frac{k^{1.5}}{C_\mu^{-0.75} l_m} \tag{42}$$

3.2 Outlet

A fluid exit boundary is normally encountered in confined cases, where the fluid leaves the calculation domain. In such a case, the flow is generally very close to be purely

parabolic in nature. When there is no area change in the outlet region, and if this region is sufficiently long and far downstream, the flow may safely be assumed as fully developed, which implies negligible streamwise gradients of all variables

$$\frac{\partial \Phi}{\partial x} = 0 \qquad \Phi = U, V, W, P^{'}, k, \varepsilon \tag{43}$$

When the flow area changes in the outlet region, the condition $\partial U/\partial x = 0$ is clearly inappropriate because mass conservation is not thereby ensured. In this case, the appropriate condition is

$$\frac{\partial (UA)}{\partial x} = 0 \tag{44}$$

where A is the cross sectional area of the outlet domain. The implementation of this condition will be discussed in MODIFY.

3.3 Axis or Plane of Symmetry

At symmetry axis or plane, there is no flux of any kind normal to the boundary, either convective or diffusive. Thus, the normal velocity component, as well as the normal gradients of the remaining dependent variable, are zero.

3.4 Wall

3.4.1 Momentum Equations

Due to the viscous influences near wall, the local Reynolds number becomes very small, thus the turbulent model which is designed for high Reynolds number become inadequate. Both this fact and the steep variation of properties near wall necessitate special treatment for nodes close to the wall. One way of handling this problem is to use turbulence models in which modifications have been introduced to take the viscous effects into account. The use of low-Reynolds-number models is one way. However, these models, when used near wall regions where the dependent variables and their gradients vary steeply, require a very fine mesh for adequate numerical solution and may lead to very high computational costs. The approach adapted in this program is called "Wall Function" as suggested by Launder and Spalding [21]. This treatment is based on the fact that the logarithmic law of the wall applies to the velocity component parallel to the wall in the region close to the wall, corresponding to a $y^+ = \frac{y u^*}{\nu}$ value in the region $30 \leq y^+ \leq 200$.

In the following explanation of the treatment of turbulence quantities near the wall, it is assumed that the region near the wall consists of two layers. The layer nearest the wall is designated the "viscous sublayer" in which the turbulent viscosity is much smaller than molecular viscosity, i.e. the turbulent shear stress is negligible. Ignoring the buffert layer, the second layer is designated the "inertial sublayer" in which the turbulent viscosity is

much greater than molecular viscosity, making it a fully turbulent region. These two layers are the wall dominated regions and it is assumed that the total shear stress is constant, an assumption that is supported by experimental data. The point $y^+ = 11.63$ is defined to dispose the buffer (transition) layer, and it correspond to the intersection point between the log-law and the near-wall linear law. Above this point the flow is assumed fully turbulent and below this point the flow is assumed purely viscous.

Since the streamwise pressure gradient is assumed to be negligible, i.e $\frac{\tau_w}{|dP/dx|} \gg y^+$, the right-hand side of the momentum equation reduces to a particularly simple one-dimensional form

$$\tau = (\mu + \mu_t) \frac{\partial U}{\partial y} \tag{45}$$

The wall function implemented in the code can be summarized as follows.

a) For $y^+ \geq 11.63$ where $\frac{\mu_t}{\mu} \gg 1$, $\tau \simeq \tau_w$

1. The wall shear stress is obtained by calculating the viscosity at the node adjacent to the wall from the log-law. The viscosity used in momentum equations is prescribed at the nodes adjacent to the wall (index P) as follows. Eq. 45 gives

$$\tau_w = \mu_t \frac{\partial U}{\partial \eta} \approx \mu_t \frac{U_P}{\eta} \tag{46}$$

Using the definition of the friction velocity U_*

$$\tau_w = \rho U_*^2 \tag{47}$$

we obtain

$$\mu_t \frac{U_P}{\eta} = \rho U_*^2 \rightarrow \mu_t = \frac{U_*}{U_P} \rho U_* \eta \tag{48}$$

Substituting U_*/U_P with the log-law

$$\frac{U_P}{U_*} = \frac{1}{\kappa} ln(E\eta^+) \tag{49}$$

we finally can write

$$\mu_t = \frac{\rho U_* \eta \kappa}{ln(E\eta^+)} \tag{50}$$

where η denotes the normal distance to the wall, and $\eta^+ = U_* \eta / \nu$.

2. The turbulent kinetic energy is set as

$$k_P = C_\mu^{-0.5} U_*^2 \tag{51}$$

3. The energy dissipation rate is set as

$$\varepsilon_P = \frac{U_*^3}{\kappa y} \tag{52}$$

4. The shear stress is obtained by

$$\tau_w = \rho U_*^2 \tag{53}$$

b) For $y^+ \leq 11.63$ where $\frac{\mu_t}{\mu} \ll 1$, $\tau \simeq \tau_w$

1. calculate U_* as follow

$$\frac{U_P}{U_*} = \frac{U_* y}{\nu} \tag{54}$$

2. follow the procedure 2 - 4 as explained above

where $E = 9$ and von Karman constant, $\kappa = 0.41$

3.4.2 Scalar Transport

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 momentum transport.

The right-hand side of the temperature equation reduces to

$$\dot{q} = (\Gamma + \Gamma_t)\, C_p \frac{\partial T}{\partial y} \tag{55}$$

It should be mentioned that the heat flux across the viscous sub-layer is assumed to be constant. As in momentum transport treatment, the point $y^+ = 11.63$ is also defined here for disposing buffer layer. For $y^+ \leq 11.63$ the transport is assumed to be due to molecular activity, and the expression for the heat flux parameter T^+ is a simple one.

For $y \geq 11.63$, the transport is assumed to be due entirely to turbulence. The heat flux parameter T^+ becomes a logarithmic function of y^+.

The solution procedure implemented in **CALC-BFC** is summarized as follows

a) For $y^+ \geq 11.63$ where $\frac{\Gamma_t}{\Gamma} \gg 1$, $q \simeq q_w$ calculate q_w from the following relation

$$T^+ = \frac{(T_w - T_P)\, \rho C_p U^*}{q_w} = \sigma_t \left[U^+ + P\left(\frac{\sigma}{\sigma_t}\right) \right] \tag{56}$$

$$\frac{q_w}{C_p} = \frac{\rho U^* (T_w - T_P)}{\sigma_t \left[U^+ + P\left(\frac{\sigma}{\sigma_t}\right) \right]} \tag{57}$$

where, according to Jayatillika [20], the P-function is defined as

$$P\left(\frac{\sigma}{\sigma_t}\right) = 9.24\left[\left(\frac{\sigma}{\sigma_t}\right)^{0.75} - 1\right]\left[1 + 0.28e^{\left(-0.007/\frac{\sigma}{\sigma_t}\right)}\right] \tag{58}$$

There are many forms of the P-function, but the particular form employed is due to Jayatillaka [20] and is valid only for impermeable, 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.

b) For $y^+ \leq 11.63$ where $\frac{\Gamma_t}{\Gamma} \ll 1$, $q \simeq q_w$

calculate q_w from the following relation

$$T^+ = \frac{\rho U^* y}{\frac{\mu}{\sigma}} = \frac{(T_w - T_P)\,\rho C_p U^*}{q_w} \tag{59}$$

$$\frac{q_w}{C_p} = \frac{\mu}{\sigma y}(T_w - T_P) \tag{60}$$

3.5 Free Boundary

This type of boundary is typically used for computation of free jet and plumes. The procedure for implementing free boundary is as follows.

1. The boundary condition for continuity equation is given by calculating the convection at the boundary from local continuity in MODCON.

2. Values are given at the boundary to all dependent variables (usually zero for the velocity components, k and ε, and free stream temperature for the temperature equation).

3. In MODP the continuity equation for control volume adjacent to the boundary is suppressed by setting

SP(I,J,K) = - GREAT
SU(I,J,K) = 0.0
PHI(I,J,K,PP) = 0.0

This is due to the fact that the continuity equation has already been solved from local continuity (see point 1).

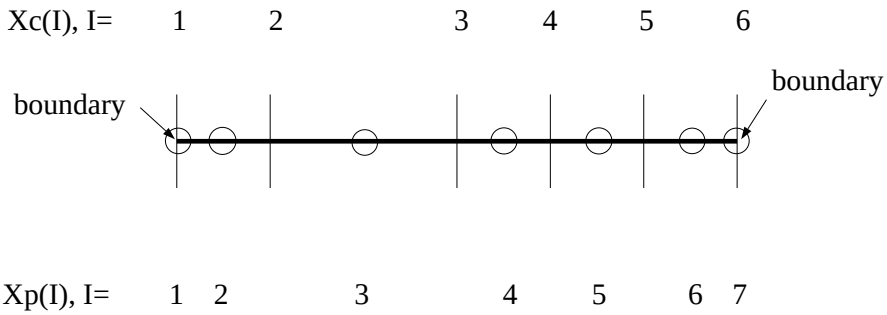

Figure 3: Location of nodes and control volume faces

4 Geometrical Details of the Grid

4.1 Grid

The coordinates of the corners (X_C, Y_C, Z_C) of each control volume should be specified by the user, i.e. the grid must be generated by the user. The nodes of the control volume (X_P, Y_P, Z_P) are placed at the center of their control volumes. The control volume adjacent to the boundaries have two nodes, one in the center and one at the boundary, see *Fig.* 3. In any coordinate direction, lets say ξ, there are NI nodes, NI-1 control volume faces, and NI-2 control volumes. The nodes are numbered (from low ξ to high ξ) from 1 to NI , the control volume faces are numbered from 1 to NI-1, and the control volumes from 2 to NI-1. This is the same for η and ζ directions. Note that (ξ, η, ζ) must form a right-hand coordinate system.

CALC-BFC employs curvilinear grids but the code can also be used with Cartesian as well as cylindrical coordinate systems.

4.1.1 Nomenclature for the Grid

A schematic control volume grid is shown in *Fig.* 1. Single capital letters define nodes [E(ast), S(outh), etc.], and single small letters define faces of the control volumes. When a location can not be referred to by a single character, combinations of letters are used. The order in which the characters appear is: first east-west, then north-south, and finally high-low.

4.1.2 Area Calculation of Control Volume Faces

The area of the control volume faces are calculated as the sum of two triangles. The x-coordinates of the corners of the east face are, for example, X_C(I,J,K), X_C(I,J-1,K), X_C(I,J,K-1) and X_C(I,J-1,K-1); the Y and Z- coordinates are Y_C and Z_C with the same

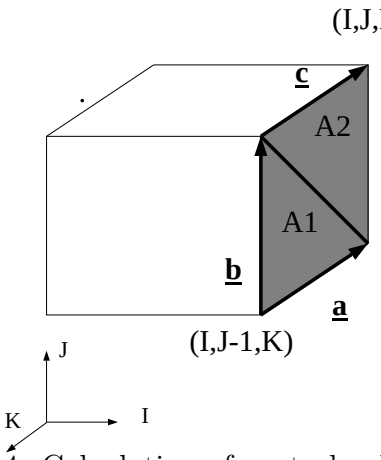

Figure 4: Calculation of control volume faces

indices, see *Fig.* 4. The area of the two triangles, $A1$, $A2$, is calculated as the cross product

$$A1 = \frac{1}{2}|\vec{a} \times \vec{b}|; \qquad A2 = \frac{1}{2}|-\vec{b} \times \vec{c}| \tag{61}$$

Above is has been assumed that the fourth edge [from (I,J-1,K-1) to (I,J,K-1)] of the east face (shaded area in *Fig.* 4) is approximately equal to $\vec{b}$. The vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ for faces in *Fig.*4 are set in a manner that the normal vectors $\vec{n1}$ and $\vec{n2}$ are pointed outwards. For the east face, for example, they are defined as

$\vec{a}$: from corner (I,J-1,K) to (I,J-1,K-1)
$\vec{b}$: from corner (I,J-1,K) to (I,J,K)
$\vec{c}$: from corner (I,J,K) to (I,J,K-1)

The Cartesian components of $\vec{a}$ are thus

$$a_x = X(I, J-1, K-1) - X(I, J-1, K) \tag{62}$$
$$a_y = Y(I, J-1, K-1) - Y(I, J-1, K) \tag{63}$$
$$a_z = Z(I, J-1, K-1) - Z(I, J-1, K) \tag{64}$$

The total area for the east face is obtained as

$$|\vec{A}|_e = |\vec{A1}|_e + |\vec{A2}|_e \tag{65}$$

The normal vector of the vector area is obtained as the average of the normal vectors of the two triangles

$$\vec{n} = \frac{1}{2}\left(\vec{a} \times \vec{b} - \vec{b} \times \vec{c}\right) \tag{66}$$

The Cartesian areas are calculated as

$$\vec{A}_{ex} = |\vec{A}|_e \vec{n} \cdot \vec{e}_x \tag{67}$$
$$\vec{A}_{ey} = |\vec{A}|_e \vec{n} \cdot \vec{e}_y \tag{68}$$
$$\vec{A}_{ez} = |\vec{A}|_e \vec{n} \cdot \vec{e}_z \tag{69}$$

where $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$ are the Cartesian unit base vector

The areas and the normal vectors of the north and high control volumes faces are calculated in exactly the same way. For the north face the vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ are defined as

$\vec{a}$: from corner (I-1,J,K) to (I,J,K)
$\vec{b}$: from corner (I-1,J,K) to (I-1,J,K-1)
$\vec{c}$: from corner (I-1,J,K-1) to (I,J,K-1)

For the high faces the vectors a, b and c are defined as

$\vec{a}$: from corner (I-1,J,K) to (I-1,J-1,K)
$\vec{b}$: from corner (I-1,J,K) to (I,J,K)
$\vec{c}$: from corner (I,J,K) to (I,J-1,K)

In **CALC-BFC** the Cartesian areas for each face (east, north and high) are calculated only once and stored in three dimensional arrays.

4.1.3 Volume Calculation of Control Volume

The volume is calculated using Gauss' law, see Burns and Wilkes [11]. Gauss' law for a vector field $\vec{B}$ reads

$$\int_V \nabla \cdot \vec{B} dV = \int_A \vec{B} \cdot d\vec{A} \tag{70}$$

setting $\vec{B} = \vec{x}$ gives

$$\delta V = \int_V dV = \frac{1}{3} \int_A \vec{x} \cdot d\vec{A} \tag{71}$$

In **CALC-BFC** the volume is calculated once only and stored in a three dimensional array.

4.1.4 Interpolation

The nodes where all variables are stored are situated in the center of the control volume. When a variable is needed at a control volume face, linear interpolation is used. The value of the variable ϕ at the east face is

$$\phi_e = f_x \phi_E + (1 - f_x)\phi_P \tag{72}$$

where

$$f_x = \frac{|\vec{Pe}|}{|\vec{Pe}| + |e\vec{E}|} \tag{73}$$

where $|\vec{Pe}|$ is the distance from P (the node) to e (the east face). In **CALC-BFC** the interpolation factors (f_x, f_y, f_z) are calculated once only and stored in three-dimensional arrays.

4.2 Gradient

Since terms which contain derivatives of ϕ ($\partial\phi/\partial x_i$) are frequently calculated in **CALC-BFC**, the geometrical quantities which appear in these expressions are calculated once only.

The transformation between the (x, y, z) and (ξ, η, ζ)-coordinate system can be written

$$\frac{\partial\phi}{\partial\xi} = \frac{\partial\phi}{\partial x}\frac{\partial x}{\partial\xi} + \frac{\partial\phi}{\partial y}\frac{\partial y}{\partial\xi} + \frac{\partial\phi}{\partial z}\frac{\partial z}{\partial\xi} \tag{74}$$

$$\frac{\partial\phi}{\partial\eta} = \frac{\partial\phi}{\partial x}\frac{\partial x}{\partial\eta} + \frac{\partial\phi}{\partial y}\frac{\partial y}{\partial\eta} + \frac{\partial\phi}{\partial z}\frac{\partial z}{\partial\eta} \tag{75}$$

$$\frac{\partial\phi}{\partial\zeta} = \frac{\partial\phi}{\partial x}\frac{\partial x}{\partial\zeta} + \frac{\partial\phi}{\partial y}\frac{\partial y}{\partial\zeta} + \frac{\partial\phi}{\partial z}\frac{\partial z}{\partial\zeta} \tag{76}$$

This can be cast in matrix form as

$$\left[\frac{\partial\phi}{\partial\xi_i}\right] = [A]\left[\frac{\partial\phi}{\partial x_i}\right] \tag{77}$$

where $\left[\frac{\partial\phi}{\partial\xi_i}\right]$ and $\left[\frac{\partial\phi}{\partial x_i}\right]$ are columns vectors. Inversion of the A-matrix gives

$$\left[\frac{\partial\phi}{\partial x_i}\right] = \frac{[B]}{J}\left[\frac{\partial\phi}{\partial\xi_i}\right] \tag{78}$$

where J (Jacobian) is the determinant of $[A]$, which is equal to $\sqrt{g}$, see Irgens [18] and Farhanieh [19]. The components of $[B]$ have the form

$$B_{11} = \frac{\partial y}{\partial\eta}\frac{\partial z}{\partial\zeta} - \frac{\partial y}{\partial\zeta}\frac{\partial z}{\partial\eta} \tag{79}$$

$$-B_{12} = \frac{\partial y}{\partial\xi}\frac{\partial z}{\partial\zeta} - \frac{\partial y}{\partial\zeta}\frac{\partial z}{\partial\xi} \tag{80}$$

$$B_{13} = \frac{\partial y}{\partial\xi}\frac{\partial z}{\partial\eta} - \frac{\partial y}{\partial\eta}\frac{\partial z}{\partial\xi} \tag{81}$$

$$\tag{82}$$

$$-B_{21} = \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \eta} \quad (83)$$

$$B_{22} = \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial z}{\partial \xi} \quad (84)$$

$$-B_{23} = \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \xi} \quad (85)$$

(86)

$$B_{31} = \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta} \quad (87)$$

$$-B_{32} = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \xi} \quad (88)$$

$$B_{33} = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \quad (89)$$

The Cartesian derivatives can be written

$$\frac{\partial \phi}{\partial x} = X_\xi (\phi_e - \phi_w) + X_\eta (\phi_n - \phi_s) + X_\zeta (\phi_h - \phi_\ell) \quad (90)$$

$$\frac{\partial \phi}{\partial y} = Y_\xi (\phi_e - \phi_w) + Y_\eta (\phi_n - \phi_s) + Y_\zeta (\phi_h - \phi_\ell) \quad (91)$$

$$\frac{\partial \phi}{\partial z} = Z_\xi (\phi_e - \phi_w) + Z_\eta (\phi_n - \phi_s) + Z_\zeta (\phi_h - \phi_\ell) \quad (92)$$

The geometrical quantities (X_ξ, X_η ....) are calculated only once and stored in three-dimensional arrays.

5 Solution Algorithm

5.1 The Iterative Solution of the Discretized Equations

The iterative solution of the discretized equations proceeds in a segregated manner with momentum equations solved first, pressure correction next and turbulence quantities last.

The solution of each set of equations is achieved by application of the TDMA. The For each iteration, the solution proceeds first along north- south grid lines sweeping the whole domain, then along east-west grid lines, and finally along low-high grid lines sweeping the whole computational domain.

During the iterative sequence, convergence is assessed at the end of each iteration on the basis of the residual sources criterion which compares the sum of the absolute residual source over all the control volumes in the computational field, for each finite-volume equation, with some reference value (typically the inlet flux of the relevant variable fed into the domain of calculation). The residual is defined as

$$R_\phi = \sum_{\text{allcells}} |a_P\phi_P - (a_N\phi_N + a_S\phi_S + a_E\phi_E + a_W\phi_W + a_H\phi_H + a_L\phi_L + S_U)| \quad (93)$$

5.2 Under-relaxation

During the solution procedure, under-relaxation may be used to promote stability, and this involves depressing the predicted changes in the calculated variables below the levels which would ordinarily be returned by the difference equations. Using an explicit formulation, under-relaxation for a general variable ϕ at a location P may be expressed by

$$\phi_P = \alpha \Phi_P^{new} + (1-\alpha)\Phi_P^{old} \tag{94}$$

where ϕ^{old} is the ϕ-value obtained from the previous iteration, ϕ^{new} is the value obtained directly through the solution of the discretized equations and α is the under-relaxation factor α $(0 < \alpha < 1)$. Under-relaxation can also be implemented implicitly (as in **CALC-BFC**) by introducing the above equation into equation (7) for ϕ_P

$$\frac{a_P}{\alpha} = \sum a_{NB}\phi_{NB} + S_U + \frac{(1-\alpha)}{\alpha} a_P \Phi_P^{old} \tag{95}$$

This arrangement is more favorable since it allows the under-relaxation to be applied without the need of the ϕ^{new}-field.

5.3 Iterative Sequence

Given the boundary conditions for the flow to be predicted, the solution proceeds as follows

1. Assign initial values (usually 10^{-10}) to the variable fields U^*, V^*, W^*, P^* and turbulence quantities k and ε.

2. Solve the U-momentum equation by first calculating the coefficients and sources, then imposing the U-velocity boundary conditions followed by application of the TDMA.

3. Point 2 is repeated for V and W.

4. Solve the pressure-correction equation by first calculating the coefficients and sources, then imposing the pressure-correction boundary conditions followed by application of the TDMA.

5. Correct the velocity fields U^*, V^*, W^* and mass fluxes (see Eq. 25) $\dot{m}_e^*$, $\dot{m}_n^*$, $\dot{m}_h^*$ with $U^{'}$, $V^{'}$, $W^{'}$.

6. Correct the pressure field P^* with $P^{'}$ to give the correct pressure field P.

7. Solve additional equations such as k, ε, T etc.

8. Go to step 2 and repeat step 2 to 7 until convergence.

6 Structure and Operation of CALC-BFC

CALC-BFC is a general program for calculating steady/transient, two- or three- dimensional recirculating laminar/turbulent flow. The flow can be of variable properties.

CALC-BFC embodies subroutines for solving the equations U, V, W, P, k, ε and any number of additional equations Φ. The operation of the code is such that the calculation of k and ε can be suppressed to allow calculation of laminar flow.

The programming language is FORTRAN 77, and the program may be run on various computers such as VAX, CONVEX, ALLIANT, CRAY and UNIX Work stations and PC machines. The program is teaching oriented and it is written in a very simple and straightforward form which is readily amenable. For computation of different flow, the case is set up in only two subroutines.

The program starts from initial specification of all field values of the dependent variables and the parameters of the calculation, including the grid parameters and those used to control and monitor the progress of the calculation. Grid parameters are either specified within the code or read from a file produced by a grid- generation program. Then the geometrical parameters are calculated by the program and stored in three-dimensional arrays. The field values and the remaining parameters of the calculation can either be specified by the user or, in the case of the continuation of a previous calculation, read from a file, containing the results of the previous calculation. The code then solves for the dependent variables in an iterative manner. Within each iteration, the individual variables U, V, W, P, k and ε are solved for.

During the calculation, the field residuals are monitored. The iteration scheme is continued until the results converges, i.e. until the normalized field residuals have fallen below a prescribed upper limit; then the grid coordinates and the dependent variables are stored in a file called SAVRES and are also printed out, for monitoring, on standard output (unit 6). In this file the variation of residuals from iteration to iteration is also included.

The structure of the **CALC-BFC** consists of two major parts: a user area which is dependent of the flow type considered and a general area independent of the flow type. The user area consists of two subroutines, SETUP and MODIFY, where most of the calculation parameters, such as control and grid parameters, together with the boundary conditions are set. Detailed description of these two subroutines is given later. The general area consists of subroutines CALCU, CALCV, CALCW, CALCP, CALCTE, CALCED, CALCPH for calculating the U, V, W, P', k and ε and Φ fields, respectively; CONV where the convection is calculated using the Rhie-Chow interpolation, COEFF where one of the three differencing schemes are used, ECHO1 which prints out the in-data given in SETUP, INIT which computes the grid and geometrical parameters and sets the initial dependent variable fields; the TDMA solver SOLVER; the printing routine PRINT; the subroutine for calculating turbulent viscosity VIST and RESTR1 which reads the results of a previous calculation from SAVRES, SAVE1 which writes the results on the file SAVRES, WALL which calculates the wall functions, BLOCK which is used to block part of the calculation domain, FIXVAL which is used for fixing a velocity component at interior nodes.

The overall structure and interconnection among the various subroutines of **CALC-BFC** is illustrated in *Fig.*5.

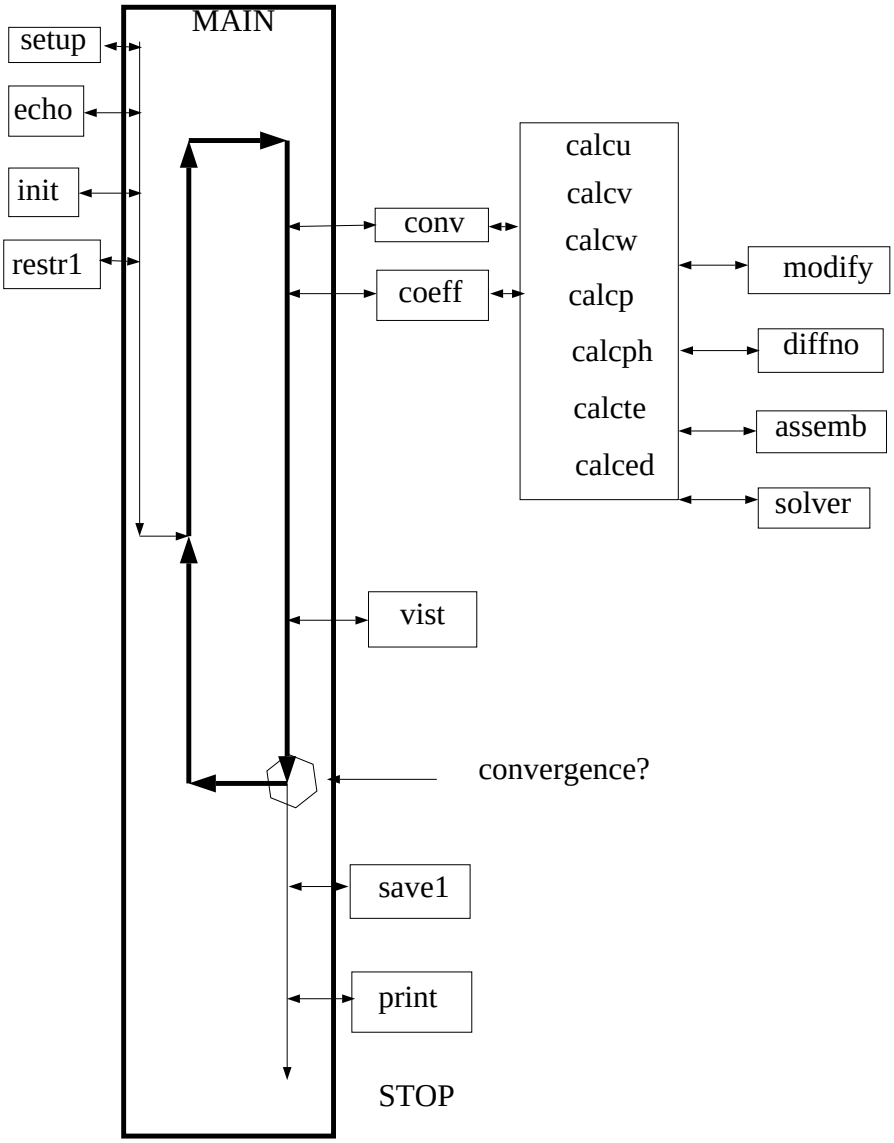

Figure 5: Flow chart of **CALC-BFC**

7 Descriptions of the Subroutines

This section is devoted to description of the various subroutines in **CALC-BFC**.

MAIN This is the main subroutine which has the overall control. It performs the initial and final operations, and also controls the iteration. MAIN is divided into three parts. In the first part the subroutines SETUP, INIT and if RESTRT = .TRUE., RESTR1 are called. In the second part of MAIN, the iteration procedure starts and the solution of the problem is controlled and monitored. The residuals are printed out in each iteration. In the third part PRINT and if SAVE = .TRUE., SAVE1 is called.

SETUP This subroutine is in the user's area. The user specifies the problem by means of this subroutine. More on SETUP will be given in next section.

ECHO1 This subroutine is called to echo (if ECHO = .TRUE.) the given data in SETUP.

INIT The grid coordinates which are specified in SETUP are used in the first part of this subroutine to calculate all geometrical quantities. The calculated quantities are then stored in three-dimensional arrays. In the second part of INIT the initial values of the dependent variables are specified.

RESTR1 Upon the user's request (RESTRT = .TRUE.) this subroutine reads the results of a previous calculation.

CONV The convections through the faces of the control volumes are calculated using Rhie- Chow interpolation method.

COEFF The coefficient(a_E , a_W ....) in the discretized equation are calculated for one of the three available schemes: hybrid central/upwind differencing, QUICK scheme or the Van Leer scheme.

CALCΦ ($\Phi = U$, V, W, T, k, ε, Φ) All these subroutines have the same general structure. If non-orthogonal coordinates are used (ORTOGO = .FALSE.), the subroutine DIFFNO is called for calculation of diffusion due to non-orthogonality. The sources are calculated. The boundary conditions are incorporated by a call to MODΦ and the coefficients are assembled by a call to the subroutine ASSEMB. Finally the equations are solved in SOLVER.

DIFFNO The diffusion due to non-orthogonality is calculated and used in CALCΦ.

ASSEMB The coefficients of combined convective/diffusive flux are assembled, the residuals are calculated, and under-relaxations are applied.

MODIFY The boundary conditions are set by the user through specification of boundary values and modification of the coefficients and source terms. More on this subroutine will be presented in the next chapter.

CALCP In this subroutine, the coefficients and sources (=mass imbalances) of the pressure-correction are first calculated. The coefficients are then assembled and residual sources calculated by summing the absolute mass sources. Next, the values of the pressure corrections are obtained by application of TDMA in SOLVER. Finally, corrections are applied

to the pressure, mass fluxes and velocity fields before a return to MAIN.

SOLVER This subroutine performs the TDMA solution procedure of the discretized equations.

VIST The turbulent viscosity in the k-ε turbulent model is calculated.

PRINT This subroutine performs the printing of dependent variables, in tabular form, in any specified plane together with a heading which contains the name of the variable printed.

PRIN3D The user can print out the three-dimensional arrays e.g., the geometrical quantities, with this subroutine.

WALL This subroutine is called in MODIFY by the user. It computes the wall functions used in turbulent flow calculations.

BLOCK This subroutine is also called in MODIFY by the user for blocking part of the flow domain.

FIXVAL With this subroutine the user can fix a velocity component at a chosen point to a fix value.

8 Users Area – SETUP and MODIFY

The user can specify the problem considered and control the code by means of subroutines SETUP and MODIFY.

8.1 SETUP

This subroutine is divided into 18 sections. A brief description of the way in which different sections has been structured is given below.

(1) The number of dependent variables are specified.

(2) The default values of the turbulent Prandtl number PRT(NPHI), number of application of line iterations NSWEEP(NPHI), residual scaling parameters, and relaxation factors are set.

(3) In this section the new relaxation factors can be specified.

(4) The maximum number of iterations is specified.

(5) The convergence criteria for termination of the computation is specified.

(6) The fluid properties such as the density, dynamic viscosity and the Prandtl number are specified.

(7) The number of applications of TDMA solver for dependent variables are specified here.

(8) In this section, by setting the logical parameters equal to .TRUE., the code reads the

results of the previous calculations (RESTRT), saves the results of current run (SAVE), applies cyclic boundary conditions (CYCLIC), neglects non-orthogonal diffusion (ORTOGO), applies steady flow condition (STEADY) and prints out the in-data (ECHO). Those events which are not required can be suppressed by putting the considered logical parameter equal to .FALSE.

(9) The required differencing scheme should be specified here.

(10) The cyclic pressure gradient, for the case of cyclic boundary conditions is specified.

(11) The required variables to be solved is selected in this section.

(12) The coordinates of the control volumes corners are specified here by the user. The coordinates can be either read from a file or specified directly in this section. In case of unsteady calculation, the time steps are given here.

(13) The residual scaling parameters for normalization of the residuals are specified here.

(14) All variables are during the iteration monitored at a specified node and with a specified interval. The node and the interval are specified in this section.

(15) The plane at which the variables should be printed out is specified.

(16) The turbulence constants are specified in this section.

(17) The Prandtl numbers for turbulent kinetic energy and dissipation rate are calculated in this section.

(18) The pressure is fixed to zero at a specified node.

```
c
            SUBROUTINE SETUP
c
             INCLUDE '../common.f'

c- section 1 ——number of variables ————————————————-
             NPHMAX = 8

c- section 2 ——set default values for all vectors ——————————
             DO 100 NPHI = 1 , NPHMAX
                  PRT(NPHI) = 0.72
                  NSWEEP(NPHI) = 1
                  REREF(NPHI) = 1.
                  DTFALS(NPHI) = GREAT
                  URF(NPHI) = 0.5
100          CONTINUE
             URFVIS = 0.5

c- section 3 ——relaxation factors ————————————————

c- section 4 ———max number of iteration ————————————————-
```

```
              MAXIT = 500

c- section 5 ——convergence criteria ————————————————
              SORMAX = 0.01

c- section 6 ——fluid properties ——————————————————
              DENSIT = 1.189
              PRANDL = 0.72
              VISCOS = 18.1E-6

c- section 7 ——number of sweeps to be performed by the tdma-solver
              NSWEEP(PP) = 2

c- section 8 ——logical parameters ————————————————-
              RESTRT = .TRUE.
              SAVE = .TRUE.
              CYCL = .FALSE.
              STEADY = .TRUE.
              ORTOGO = .TRUE.
              ECHO=.TRUE.

c- section 9 ————————choice of differencing scheme ————
c (hybrid='h', van leer='v' or quick='q')
              SCHEME = 'h'
              SCHTUR = 'h'

c- section 10 —-cyclic pressure gradient ————————————-
              BETAP = 0.0

c- section 11 ——dependent variable selection ——————————
              SOLVE(U) = .TRUE.
              SOLVE(V) = .TRUE.
              SOLVE(W) = .FALSE.
              SOLVE(PP) = .TRUE.
              SOLVE(TE) = .TRUE.
              SOLVE(ED) = .TRUE.
              SOLVE(T) = .FALSE.

c- section 12 —-grid specification ————————————————-

c- section 13 ——residual scaling parameters ——————————
c inlet: area = 0.1, Uin = 0.5
              FLOW = DENSIT*0.1*0.5
```

```
            REREF(PP) = FLOW
            REREF(U) = FLOW*0.5
            REREF(V) = FLOW*0.5
            REREF(TE)= FLOW*0.5**2
            REREF(ED) = FLOW*0.5**3/0.1

c- section 14 ——all variables are printed during the iteration at node
c (imon,jmon,kmon) every indmon iteration ————————
            IMON = 2
            JMON = 2
            KMON = 2
            INDMON = 1

c- section 15 ——print-out parameters ————————————————-
            PLANE = 'XY'

c- section 16 ——turbulence constants ————————————————
            CMU = 0.09
            CD = 1.0
            CMUCD = CMU*CD
            C1 = 1.44
            C2 = 1.92
            CAPPA = 0.435
            ELOG = 9.0

c- section 17 ——Prandtl numbers ————————————————————-
            PRT(ED) = CAPPA*CAPPA/(C2-C1/(CMU**0.5)
            PRT(TE) = 1.
            PFUN = PRANDL/PRT(T)
            PFUN =9.24*(PFUN**0.75 - 1.0)*(1.0 + 0.28*EXP(-0.007*PFUN))

c- section 18 ——pressure fixed to zero at node ————————————-
            IPREF = 2
            JPREF = 2
            KPREF = 2

            RETURN
            END
```

8.2 MODIFY

The second subroutine which also lies in the users area is MODIFY. This subroutine is mainly concerned with the modification of the boundary conditions, via changing the coefficients and sources in the discretized equations, to suit the individual problems. A brief description of the different sections of the subroutine is given below.

(0) The initial fields can be given here.

(1) The turbulent viscosity and density are modified in this section.

(2) The convection is modified here.

(3) The modifications are made here to the U - momentum discretized equations.

(4) The modifications are made here to the V - momentum discretized equations.

(5) The modifications are made here to the W - momentum discretized equations.

(6) The modification to pressure correction field is applied here.

(7) In this entry, modifications are made to the turbulence kinetic energy.

(8) In this section the modification is introduced to the turbulence energy dissipation rate.

(9) The modifications to the Φ-equations are introduced in this entry.

```
      SUBROUTINE MODIFY
      INCLUDE '../common.f'

c- section 0 ————————initialization————————————————
      ENTRY MODINI
      RETURN

c- section 1 ————————- properties ——————————————————-
      ENTRY MODPRO
      RETURN

c- section2 ———————— convection ——————————————————
      ENTRY MODCON
      RETURN

c- section 3 ————————- U - momentum ————————————————
      ENTRY MODU
      RETURN

c- section 4 ————————- V - momentum ————————————————
      ENTRY MODV
      RETURN
```

```
c- section 5 ————— W - momentum ————————————————-
             ENTRY MODW
             RETURN

c- section 6 ————— Pressure ————————————————————-
             ENTRY MODP
             RETURN

c- section 7 ———— turbulent kinetic energy ————————————-
             ENTRY MODTE
             RETURN

c- section 8 ———— energy dissipation rate ————————————
             ENTRY MODED
             RETURN

c- section 9 ———— auxiliary equation (Φ) ——————————
             ENTRY MODPHI(NPHI)
             RETURN
             END
```

9 How to Use CALC-BFC

The easiest way of learning how to use **CALC-BFC** is to start with a laminar two-dimensional problem which someone else has already done, then make small changes to the input and observe the consequences. Two test problems will be given here for users' practice. The first one is a laminar heat and mass transfer in a square duct and the second one is a two-dimensional turbulent flow between parallel plates. The users are required to run these two problems and compare the results with available analytical solution in the literature.

9.1 Example One

A three dimensional heat and mass transfer in a square duct is a problem which has a well established numerical solution. We will show here how the user should make the adequate changes to SETUP and MODIFY. Due to the symmetric nature of the flow, the calculation domain is confined to right hand side upper (north-high) quarter of the cross-section.

9.1.1 Required Changes in SETUP

In section (2) the appropriate under-relaxation factors are 0.5. The user can investigate the effect of changing the value of these factors on the convergence.

URF(NPHI) = 0.5

In sections (4) and (5) the maximum number of iteration and convergence criteria are chosen to be 300 and 0.001, respectively.

MAXIT = 300
SORMAX = 0.001

The fluid properties in section (6) is set as air properties.

DENSIT = 1.189
VISCOS = 18.1E-6
PRANDL = 0.72

Please note that VISCOS is the *dynamic* viscosity μ. Number of sweeps to be performed by the TDMA solver are chosen in section (7) to be 2 for PP. Change the number of sweeps and see the effect on convergence.

NSWEEP(PP)= 2

In section (8) the logical parameters are set to .FALSE. for RESTRT, CYCL and STEADY, but SAVE and ECHO are put to .TRUE. in order to save the results in an unformatted file called SAVRES and print out the in-data. Run the program by using these results as initial values and observe the effect on convergence. Since using Cartesian coordinates are used, ORTOGO is set to .TRUE., so that the non-orthogonal diffusion terms are not unnecessarily calculated.

RESTRT = .FALSE.
SAVE = .TRUE.
CYCL = .FALSE.
STEADY = .FALSE.
ORTOGO = .TRUE.
ECHO = .TRUE.

Since the flow is laminar, there is no need for calculation of ε and k. Therefore, in section (10) SOLVE(TE) and SOLVE(ED) are set equal to .FALSE.

SOLVE(U) = .TRUE.

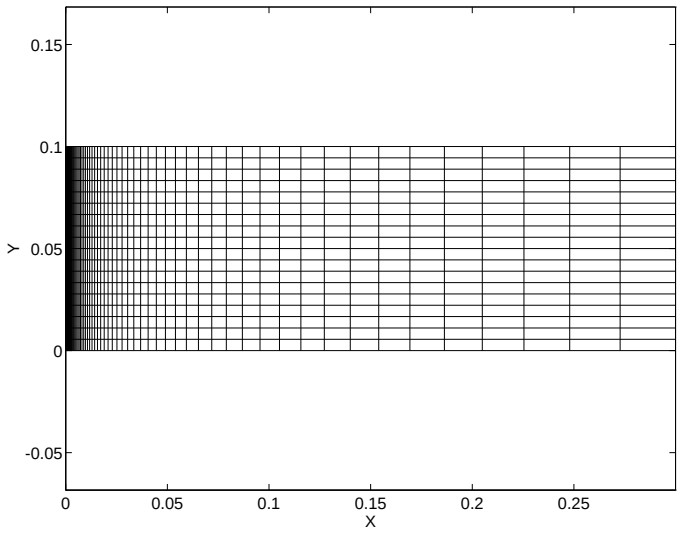

Figure 6: Grid.

```
SOLVE(V) = .TRUE.
SOLVE(W) = .TRUE.
SOLVE(PP)= .TRUE.
SOLVE(TE)= .FALSE.
SOLVE(ED)= .FALSE.
SOLVE(T) = .TRUE.
```

Grid specification is given in section (12). Note that an expansion parameter is introduced to obtain finer mesh in the entrance region of the duct in the flow direction, see *Fig.*6. Note that DX is arbitrarily chosen.

```
      NI=70
      NJ=20
      NK=20
      NIM1=NI-1
      NJM1=NJ-1
      NKM1=NK-1
c
c x-direction
      XL=0.3
      F=1.1 !Expansion factor
      DX=1
      DO I=2,NIM1
            XC(I,1,1)=XC(I-1,1,1)+DX
            DX=F*DX
      END DO
```

```
      XMAX=XC(NI-1,1,1)
c
C make the total length = XL
      DO I=2,NIM1
         XC(I,1,1)=XC(I,1,1)*XL/XMAX
      END DO

      H = 0.1
      B = 0.05
      DO 20 K = 1, NKM1
      DO 20 J = 1, NJM1
      DO 20 I = 1, NIM1
         XC(I,J,K) = XC(I,1,1)
         YC(I,J,K) = H*FLOAT(J-1)/FLOAT(NJ-2)
         ZC(I,J,K) = B*FLOAT(K-1)/FLOAT(NK-2)
20       CONTINUE
```

where XL, H and B are length, half the height and half the width of the duct, respectively.

In section (13) the residuals of the momentum equations are normalized by inlet flux, whereas the pressure correction is normalized by the mass flux and the temperature with heat in-flux.

```
      TIN = 30.
      UIN = 0.5
      FLOW = DENSIT*UIN*H*B
      REREF(PP) = FLOW
      REREF(U) = FLOW*UIN
      REREF(V) = FLOW*UIN
      REREF(W) = FLOW*UIN
      REREF(T) = FLOW*TIN
```

9.1.2 Required Changes in MODIFY

In section (2) the convection through the outlet is calculated by means of continuity. This is the boundary condition for the continuity equation.

```
      ENTRY MODCON
c—-inlet
      FLOWIN = 0.0
      DO 20 K = 2, NKM1
      DO 20 J = 2, NJM1
```

```
c compute the total mass influx
                  FLOWIN = FLOWIN + CONVE(1,J,K)
20              CONTINUE

c—-outlet
              ASUM = 0.0
              FLOWUT = 0.0
              DO 30 K = 2, NKM1
              DO 30 J = 2, NJM1
                 ARES =SQRT(AREAEX(NIM1,J,K)**2+AREAEY(NIM1,J,K)**2
                       +AREAEZ(NIM1,J,K)**2)
                 ASUM = ASUM + ARES
c compute the total mass outflux through the last INTERIOR face
                 FLOWUT = FLOWUT + CONVE(NI-2,J,K)
30            CONTINUE

c compare total in and out mass flux ...
              UINC = (FLOWIN - FLOWUT)/ASUM/DENSIT

              DO 40 K = 2, NKM1
              DO 40 J = 2, NJM1
                 ARES =SQRT(AREAEX(NIM1,J,K)**2+AREAEY(NIM1,J,K)**2
                       +AREAEZ(NIM1,J,K)**2)
c and use it to correct the outgoing mass flux so that
c global mass conservation is satisfied
                 CONVE(NIM1,J,K) = CONVE(NI-2,J,K) + UINC*DENSIT*ARES
40            CONTINUE
              RETURN
```

The U - velocity boundary conditions is given in section (3). In this case a uniform velocity profile is imposed at the inlet. The subroutine WALL is used for specification of symmetry boundary conditions. The call to the subroutine WALL is made by WALL(Φ,'face', I-start, I-end, J-start, J-end, K-start, K-end, value)
where $\Phi = U$, V, W, T, ε, k, 'face' indicates at which face (East, West, North ....) the boundary condition is to be applied, and (I-, J-, K-start) and (I-, J-, K-end) indicating the start and end of the wall or symmetry line. They should not be less than 2 and not greater than NI-1, NJ-1 and NK-1, respectively. If value is set equal to a value less than -100., e.g. -200., it indicates symmetry plane. In case of temperature (i.e. $\Phi = T$) 'value' is the wall temperature. For all other variables, 'value' has no meaning except indicating symmetry or wall.

```
              ENTRY MODU
c—-inlet
```

```
            DO 100 K = 2, NKM1
            DO 100 J = 2, NJM1
                  PHI(1,J,K,U) = UIN
100            CONTINUE

c---outlet
            DO 110 K = 2, NKM1
            DO 110 J = 2, NJM1
                  PHI(NI,J,K,U) = PHI(NIM1,J,K,U)
110            CONTINUE

c---symmetry planes
            CALL WALL(U,'SOUTH',2,NIM1,2,2,2,NKM1,-200.)
            CALL WALL(U,'LOW',2,NIM1,2,NJM1,2,2,-200.)
            RETURN

            ENTRY MODV
c---outlet
            DO 210 K = 2, NKM1
            DO 210 J = 2, NJM1
                  PHI(NI,I,J,K,V) = PHI(NIM1,J,K,V)
210            CONTINUE
c---symmetry plane
            CALL WALL(V,'LOW',2,NIM1,2,NJM1,2,2,-200.)
            RETURN
```

Note that the Fortran array PHI has four indices, where the last index is and INTEGER specifying the variable. As default we have eleven pre-defined indices: U, V, W, P, PP, TE, ED, T, F1, F2 and F3 which are defined as INTEGERS in the common-file, and they are assigned the values from 1 to 11 in the beginning of MAIN. In section (5) the boundary conditions for W - velocity are given.

```
            ENTRY MODW
c---outlet
            DO 310 K = 2, NKM1
            DO 310 J = 2, NJM1
                  PHI(NI,I,J,K,W) = PHI(NIM1,J,K,W)
310            CONTINUE

c---symmetry planes
            CALL WALL(W,'SOUTH',2,NIM1,2,2,2,NKM1,-200.)
            RETURN
```

In section (9) the modifications to energy equation are imposed via MODPHI. When MODPHI is called NPHI=T. The wall temperature is specified by calling subroutine WALL.

```
            ENTRY MODPHI(NPHI)
c---inlet
            DO 400 K = 2, NKM1
            DO 400 J = 2, NJM1
                  PHI(1,J,K,T) = TIN
400             CONTINUE

c---outlet
            DO 410 K = 2, NKM1
            DO 410 J = 2, NJM1
                  PHI(NI,J,K,T) = PHI(NIM1,J,K,T)
410             CONTINUE

c---the walls
            CALL WALL(T,'NORTH',2,NIM1,NJM1,NJM1,2,NKM1,TWALL)
            CALL WALL(T,'HIGH',2,NIM1,2,NJM1,NKM1,NKM1,TWALL)
c--symmetry planes
            CALL WALL(T,'SOUTH',2,NIM1,2,2,2,NKM1,-200.)
            CALL WALL(T,'LOW',2,NIM1,2,NJM1,2,2,-200.)
            RETURN
```

9.2 Example Two

The second test problem is a two-dimensional turbulent flow between parallel plates. As in the first test problem the required modifications to SETUP and MODIFY are presented in this sub-section.

9.2.1 Required Changes in SETUP

SETUP in this case is mainly similar to the previous test example, the only difference is the changes which are required in section (10). Since the flow is turbulent, the k and ε equations must be solved. Also, the flow is two-dimensional, therefore the W component of the velocity is not required.

The relaxation factor in section (2) is set equal to 0.5.

```
            SOLVE(U) = .TRUE.
            SOLVE(V) = .TRUE.
```

```
SOLVE(W) = .FALSE.
SOLVE(PP)= .TRUE.
SOLVE(TE)= .TRUE.
SOLVE(ED)= .TRUE.
SOLVE(T) = .FALSE.
```

The grid specification is given in section (12). Since the flow is two-dimensional, **we must put NK = 3** and the Z-coordinate of the control volume corners should be specified as follows:

```
      DO 10 I = 1, NIM1
      DO 10 J = 1, NJM1
            ZC(I,J,1) = 0.0
            ZC(I,J,2) = 1.0
10        CONTINUE
```

The grid can be specified as follows. In this case too, a stretched grid can be used, both in X and Y directions. However, for simplicity constant grid spacing is used.

```
      XL = 0.5
      H = 0.05
      DO 20 I = 1, NIM1
      DO 20 J = 1, NJM1
      DO 20 K = 1, NKM1
            XC(I,J,K)=XL*FLOAT(I-1)/FLOAT(NI-2)
            YC(I,J,K)=H*FLOAT(J-1)/FLOAT(NJ-2)
20        CONTINUE
```

9.2.2 Required Changes in MODIFY

Due to the symmetry in the flow field, the computational domain in this case can be confined to the half width of the channel, e.g. between the south wall and the symmetry line running through the middle of the channel.

The flow is two-dimensional, therefore in all the MODΦ, for concerned dependent variables, the coefficients AH and AL are put equal to zero for <u>all</u> variables in the whole domain.

```
AH(I,J,K) = 0.0
AL(I,J,K) = 0.0
```

which can be done with a call to WALL in MODU, MODV, MODTE, MODED and MODPHI.

```
CALL WALL(Φ,'HIGH',2,NIM1,2,NJM1,NKM1,NKM1,-200.)
CALL WALL(Φ,'LOW',2,NIM1,2,NJM1,2,2,-200.)
```

Since the flow is turbulent the wall function is applied at the wall by calling the subroutine WALL. The turbulent viscosity at wall (at the node adjacent to the wall) is calculated by log-law and wall shear stress equations.

```
ENTRY MODPRO
CALL WALL(U,'SOUTH',2,NIM1,2,2,2,2,0.0)
RETURN
```

When the viscosity has been set, no call to WALL is needed for the velocities (see section 3.4.1).

At the symmetry line

```
ENTRY MODU
CALL WALL(U,'NORTH',2,NIM1,NJM1,NJM1,2,2,-200.)
RETURN
```

The subroutine WALL is called in MODTE and MODED.

```
ENTRY MODTE
CALL WALL(TE,'SOUTH',2,NIM1,2,2,2,2,0.)
CALL WALL(TE,'NORTH',2,NIM1,NJM1,NJM1,2,2,-200.)
RETURN

ENTRY MODED
CALL WALL(ED,'SOUTH',2,NIM1,2,2,2,2,0.)
CALL WALL(ED,'NORTH',2,NIM1,NJM1,NJM1,2,2,-200.)
RETURN
```

9.3 Blockage

CALC-BFC is often required to simulate the flow in a space of which parts are inaccessible to the fluid, because of a partial or total blockage by solid material. This inaccessibility can be represented by special treatment of the source terms, which allows the extent of blockage of each cell face and volume to be numerically expressed.

Although the extensive use of blockage can make the use of body-fitted coordinates entirely unnecessary, it is more common for blockage to be used in combination with body-fitted-coordinate grid. Such combinations are, for example, used for simulations of the flow around motor vehicles.

To block the fluid flow to an area the following treatment is employed in the every MODΦ and MODCON by calling the subroutine BLOCK. The following three operations are taken place in the subroutine BLOCK

1. The dependent variables are put to zero by

SP(I,J,K) = - GREAT
SU(I,J,K) = SMALL
PHI(I,J,KΦ) = SMALL

where GREAT = 10^{20} and SMALL = 10^{-20}.

To explain the reason behind this treatment, the discretized equations are rewritten in the following form

$$\Phi_P = \frac{\sum a_{NB}\Phi_{NB} + SU}{\sum a_{NB} - SP} \tag{96}$$

by putting SU = SMALL and SP = - GREAT, the dependent variable Φ_P becomes equal to SMALL.

2. The subroutine BLOCK manipulates the interpolation factors in a way that the nodes in the blocked region are not used in the calculations.

3. When BLOCK is called from MODCON (with $\Phi = 0$) the convections are set to zero.

4. The subsroutime WALL is called for faces applying wall functions or symmetry boundary conditions according the 'value'.

BLOCK(Φ,ISTART,IEND,JSTART,JEND,KSTART,KEND,VALUE)

where $\Phi = 0$, U, V, W, PP, T, ε, k, and (I-, J-, K-start) and (I-, J-, K-end) indicating the start and end of the blocked region; $\Phi = 0$ is used when BLOCK is called from MODCON. The start and end values of I, J and K should not be less than 2 and not greater than NI-1, NJ-1 and NK-1, respectively. 'value' has the same meaning as for WALL.

Note that if a region with zero velocities is prescribed using BLOCK, the subroutine BLOCK must also be called for the convection in MODCON with $\Phi = 0$ and for the pressure correction equation, PP.

10 General Advice in Using CALC-BFC

CALC-BFC can be used successfully by users with varied experience and background. However, the probability of success is greatest if the user is already familiar with the general features of computer simulation.

In this section, we give some advice in running the program and combating eventual divergence.

10.1 Computing a New Problem

Start by determining the salient features of the predicted flow by setting NI, NJ and NK to values as low as 20 so that the required information can be obtained speedily and at low cost. If the flow is unsteady, the time steps should be as large as possible, and the number of time steps correspondingly small, for the same reason.

The coarse grid solution are not naturally reliable as indicators of how fine grid will behave; but they are likely to show data-input errors.

The progress of the calculation is better to be followed in the beginning. It is useful to set MAXIT to a small value, e.g. 100, and to proceed by way of a succession of restarts. Even if this is not done, it is wise to ensure that MAXIT is set to some not-excessive value.

The above mentioned advises should not be by-passed; for it is only highly experienced users who can set up **CALC-BFC** for a problem of any complexity without having to modify their initial formulation.

10.2 Encountering Divergence

Because the solution procedures are essentially iterative, it is inevitable that some adjustment shall be carried out before the solution has converged sufficiently for the computation to be terminated. Divergence is caused by the tendency of the residuals in one or more equations to increase rather than decrease as the iterations proceed and it is usually accompanied by the appearance of unphysically large values in some of the dependent variables. It should be mentioned here that the divergence usually occurs due to the error in the input data. When divergence occurs, it is necessary to establish the cause; this is usually to be found in the strength of the linkages between two or more sets of equations. For example if a problem of convection heat transfer is being solved, the-two way interaction between the temperature field and the velocity field, whereby each influences the other, is a possible source of divergence. Whether it is the source in a particular case is easily established by freezing the temperature field before the divergence has progressed too far, and then observing if the divergence continues. If it does not, the velocity - temperature link can be regarded as the source of divergence; otherwise, the cause must be searched in some other linkages.

To apply freezing, the simplest way is to under-relax heavily. In this way, one can investigate the contributions to divergence of linkages between individual velocity components or between turbulence energy and its dissipation rate in a turbulent flow.

If freezing by very severe under-relaxation restores convergent behavior, it is obviously possible that modest under-relaxation will have the same qualitative tendency, which

allows the solution to proceed so that all residuals do finally diminish.

The use of under-relaxation is a common mean of securing convergence in practice. If under-relaxation is employed indiscriminately, it can lead to waste of computer time. However, when it is applied to only those equations that have been identified as potential cause of divergence, and with the amount that is necessary to procure convergence, it is a good second-recourse remedy to apply. The first-recourse is to check the in-data.

10.3 Turbulent Flow

The k and ε - equations are especially sensitive for initial values due to their strong non-linear sources. One useful method is to calculate the flow without solving for k and ε, but using a laminar viscosity which is 100 to 1000 times larger than the physical one. Then restart can be activated. The initial values for k and ε should, in the first calculation, be set to $k = 0.1U_{in}^2$ and $\varepsilon = \rho C_\mu \frac{k^2}{\mu}$.

References

[1] L. Davidson and B. Farhanieh *CALC-BFC: A Finite-Volume Code Employing Collocated Variable Arrangement and Cartesian Velocity Components for Computation of Fluid Flow and Heat Transfer in Complex Three-Dimensional Geometries*, Rept. 92/4, Thermo and Fluid Dynamics, Chalmers University of Technology, Göteborg, 1992.

[2] L. Davidson, *A Note on CALC-BFC: A Finite-Volume Code for Complex Three-Dimensional Geometries Using Collocated Variables and Cartesian Velocity Components*, Dept. of Applied Thermodynamics and Fluid Mechanics, Chalmers University of Technology, Göteborg, Sweden, 1989.

[3] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow* McGraw-Hill, Washington, 1980.

[4] W. Shyy, S. S. Tong and S. M. Corre, *Numerical Recirculating Flow Calculation Using Body-Fitted Coordinate System*, Numer. Heat Transfer, Vol.8, pp. 99-113, 1985.

[5] M. Braaten and W. Shyy, *Study of Recirculating Flow Computation using Body-Fitted Coordinates: Consistency Aspects and Mesh Skewness*, Numer. Heat Transfer, Vol. 9, pp. 559-574, 1986.

[6] W. Rodi, *Turbulence Models and Their Application in Hydraulics*, International association of Hydraulic Research, Monograph Delft, The Netherlands, 1980.

[7] K. C. Karki, *A Pressure Based Calculation Procedure for Viscous Flows at All Speeds in Arbitrary Configurations*, AIAA paper 88-0058, Reno, 1988.

[8] L. Davidson and P. Hedberg, *Mathematical Derivation of a Finite-Volume Formulation for Laminar Flow in Complex Geometries*, *Int. J. Numer. Meth. Fluids*, Vol. 9, pp. 531-540, 1989.

[9] L. Davidson, *Numerical Simulation of turbulent Flow in Ventilated Rooms*, PhD thesis. Dept. of Applied Thermodynamics and Fluid Mechanics, Chalmers University of Technology, Göteborg, Sweden, 1989.

[10] C. M. Rhie and W. L. Chow, *Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation*, AIAA J., Vol. 2, pp. 1525-1532, 1983.

[11] A. D. Burns and N. S. Wilkes, *A Finite Difference Method for the Computation of Fluid Flow in Complex Three-Dimensional Geometries*, AERE R 12342, Harwell Laboratory, U.K., 1986.

[12] S. Majumdar *Developing of a Finite-Volume Procedure for Prediction of Fluid Flow Problems with Complex Irregular Boundaries* SFB 210/T/29, University of Karlsruhe, 1986.

[13] M. Peric. R. Kessler and G. Scheuerer, *Comparison of Finite-Volume Numerical Methods with Staggered and Collocated grids*, Comput. Fluids, Vol. 16, pp. 389-403, 1988.

[14] T. F. Miller and F. W. Schmidt, *Use of a Pressure-Weighted Interpolation Method for the Solution of the Incompressible Navier-Stokes Equations on a Non-Staggered System*, Numer. Heat Transfer, Vol. 14, pp. 213-233, 1988.

[15] S. Majumdar, *Role of Underrelaxation in Momentum Interpolation for Calculation of Flow with Non-staggered grids*, Numer. Heat Transfer Vol. 13, pp. 125-132, 1988.

[16] B. P. Leonard, *A Stable and Accurate Convective Modeling Based on Quadratic Upstream Interpolation*, Comp. Math. Appl. Mech. Engng., Vol. 19, pp. 59, 1979.

[17] B. Van Leer, *Towards the Ultimate Conservative Difference Scheme. II. Monotonicity and Conservation Combined in a Second-Order Scheme*, J. Comp. Phys. Vol. 14, pp. 361-370, 1974.

[18] F. Irgens, *Tensoranalyse og Kontinumsmekanik. del III*, Institutt for Mekanikk, Norge Tekniska Hogskole, Trondhiem, 1966. (in Norwegian)

[19] B. Farhanieh, *Introduction to Tensor Calculus and General Coordinate System*, Publ. PB-91/13-SE, Dept of Thermo and Fluid Dynamics, Chalmers university of Technology, Göteborg, Sweden, 1991.

[20] C. L. V. Jayatillaka, Progr. in Heat Mass Transfer. Vol. 1, pp. 193, 1969.

[21] B.E. Launder and D.B. Spalding , The Numerical Computation of Turbulent Flows, *Computer Methods in Applied Mech. and Eng.*, Vol. 3, pp. 269-289, 1974.

11 Common Block

```
parameter(it=52,jt=32,kt=32,nphit=11)
common
1/pres/ipref,jpref,kpref
1/prin1/plane
1/prin2/indmon,imon,jmon,kmon
1/vecti/nsweep(nphit)
1/vectl/solve(nphit)
1/vectc/head(nphit),
1/vectr/phi(it,jt,kt,nphit),phio(it,jt,kt,nphit),prt(nphit),
2       resor(nphit),reref(nphit),urf(nphit),
3       deksi(it,jt,kt),deeta(it,jt,kt),dezeta(it,jt,kt),
4       dnksi(it,jt,kt),dneta(it,jt,kt),dnzeta(it,jt,kt),
5       dhksi(it,jt,kt),dheta(it,jt,kt),dhzeta(it,jt,kt),
6       conve(it,jt,kt),convn(it,jt,kt),convh(it,jt,kt),
7       smp(it,jt,kt),dtfals(nphit)
1/alli/ni,nj,nk,nim1,njm1,nkm1,iter,maxit,nphmax,itstep,ntstep
1/allr/great,small,sormax,betap,time,dt(10000)
1/allc/scheme,schtur
1/alll/save,restrt,cycl,ortogo,steady,echo
common
1/geom/xc(it,jt,kt),yc(it,jt,kt),zc(it,jt,kt),vol(it,jt,kt),
2       areaex(it,jt,kt),areaey(it,jt,kt),areaez(it,jt,kt),
3       areanx(it,jt,kt),areany(it,jt,kt),areanz(it,jt,kt),
4       areahx(it,jt,kt),areahy(it,jt,kt),areahz(it,jt,kt),
5       fx(it,jt,kt),fy(it,jt,kt),fz(it,jt,kt)
1/flupr/urfvis,viscos,densit,den(it,jt,kt),vis(it,jt,kt)
1/turb/cmucd,cmu,cd,c1,c2,cappa,elog,pfun,prandl
1/coef/ap(it,jt,kt),an(it,jt,kt),as(it,jt,kt),ae(it,jt,kt),
2       aw(it,jt,kt),ah(it,jt,kt),al(it,jt,kt),
3       su(it,jt,kt),sp(it,jt,kt)
1/coefge/xksi(it,jt,kt),xeta(it,jt,kt),xzeta(it,jt,kt),
2       yksi(it,jt,kt),yeta(it,jt,kt),yzeta(it,jt,kt),
3       zksi(it,jt,kt),zeta(it,jt,kt),zzeta(it,jt,kt)
1/point/u,v,w,p,pp,te,ed,t,f1,f2,f3
integer u,v,w,p,pp,te,ed,t,f1,f2,f3
logical solve,save,restrt,cycl,steady,ortogo,echo
character plane*2,head*24,scheme*2,schtur*2
```

12 Glossary of FORTRAN Symbols for CALC

A(I)	coefficient of recurrence formula
AE(I,J,K)	coefficients of convective/diffusive flux through east wall of control volume
AH(I,J,K)	coefficients of convective/diffusive flux through high wall of control volume
AL(I,J,K)	coefficients of convective/diffusive flux through low wall of control volume
AN(I,J,K)	coefficients of convective/diffusive flux through north wall of control volume
AP(I,J,K)	sum of coefficients AE, AW, AN, AS, AH, AL and APO and source SP
APO	coefficient for old time step
AREAST	area of east wall of scalar control volume projected on the plane normal to PE
AREAEX(I,J,K)	A_{ex}
AREAEY(I,J,K)	A_{ey}
AREAEZ(I,J,K)	A_{ez}
AREANX(I,J,K)	A_{nx}
AREANY(I,J,K)	A_{ny}
AREANZ(I,J,K)	A_{nz}
AREAHX(I,J,K)	A_{hx}
AREAHY(I,J,K)	A_{hy}
AREAHZ(I,J,K)	A_{hz}
ARHIGH	area of east wall of scalar control volume projected on the plane normal to PH
ARNORT	area of north wall of scalar control volume projected on the plane normal to PN
AS(I,J,K)	coefficients of convective/diffusive flux through south wall of control volume
AW(I,J,K)	coefficients of convective/diffusive flux through west wall of control volume
B(I)	coefficients of recurrence formula
BETAP	cyclic pressure gradient
C(I)	coefficients of recurrence formula
C1	constants of turbulence model (=1.44)
C2	constants of turbulence model (=1.92)
CAPPA	von Karman's constant (=0.41)
CD	constant of turbulence model (=1.0)
CMU	constant of turbulence model (=0.09)
CMUCD	constant of turbulence model (=CMU*CD)
CONVE(I,J,K)	convection flux through the east wall of scalar control volume
CONVH(I,J,K)	convection flux through the high wall of scalar control volume
CONVN(I,J,K)	convection flux through the north wall of scalar control volume
CP	maximum of zero and net outflow (SMP) from control volume

CYCL	logical variable to control whether cyclic boundary conditions are to be applied or not
DEETA	
DEKSI	
DEZETA	
DHETA	
DHKSI	
DHZETA	
DNETA	
DNKSI	
DNZETA	coefficient for calculation of diffusion [see, eq. (24)]
DEN(I,J,K)	density of fluid at current time step
DENSIT	constant density of fluid set in SETUP
DT(ITSTEP)	time step
DU	coefficient of velocity-correction term for U velocity
DV	coefficient of velocity-correction term for V velocity
DW	coefficient of velocity-correction term for W velocity
ECHO	logical for activating subroutine ECHO1 in order to echo the given in-data in SETUP
ELOG	constant used in the log-law of the wall (= 9.0)
FX(I,J,K)	
FY(I,J,K)	
FZ(I,J,K)	interpolation factors
G11, G12, G13	
G21, G22, G23	
G31, G32, G33	the covariant components of the metric tensor ($g_{11}, g_{12}, g_{13}, g_{21}, g_{22}, g_{23}, g_{31}, g_{32}, g_{33}$)
G1X, G1Y, G1Z	the Cartesian components of the covariant base vector, $\vec{g}_1$
G2X, G2Y, G2Z	the Cartesian components of the covariant base vector, $\vec{g}_2$
G2X, G2Y, G3Z	the Cartesian components of the covariant base vector, $\vec{g}_3$
GEN	generation of turbulence by shear from mean flow
GREAT	a very large value (i.e. 10^{20})
HEAD(ED)	heading 'Energy Dissipation'
HEAD(TE)	heading 'Turbulent Energy'
HEAD(P)	heading 'Pressure'
HEAD(T)	heading 'Temperature'
HEAD(U)	heading 'U-Velocity'
HEAD(V)	heading 'V-Velocity'
HEAD(W)	heading 'W-Velocity'
IMON	I-index of monitoring location
INDMON	monitoring output each INDMON iteration
IPREF	I-index of location where pressure is fixed if IPREF > 0; if IPREF < 0 the pressure is not fixed
IT	I-index of maximum dimension of dependent variable

ITER	number of iterations completed
ITSTEP	time step index
JMON	J-index of monitoring location
JPREF	J-index of location where pressure is fixed if JPREF > 0; if JPREF < 0 the pressure is not 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 if KPREF > 0; if KPREF < 0 the pressure is not fixed
MAXIT	maximum number of iterations to be completed in the current run if iteration is not stopped by test on value of SORMAX
NI	maximum value of i-index for the calculation domain
NIM1	=NI-1
NJ	maximum value of j-index for the calculation domain
NJM1	=NJ-1
NSWEEP(ED)	number of application of line iteration for ε-equation
NSWEEP(TE)	number of application of line iteration for k-equation
NSWEEP(PP)	number of application of line iteration for PP-equation
NSWEEP(T)	number of application of line iteration for T-equation
NSWEEP(U)	number of application of line iteration for U-equation
NSWEEP(V)	number of application of line iteration for V-equation
NSWEEP(W)	number of application of line iteration for W-equation
NTSTEP	last time step
ORTOGO	logical for de-activating subroutine DIFFNO where diffusion due to non-orthogonality is calculated
PFUN	constant of P-function for heat transfer at walls
PHI(I,J,K,ED)	energy dissipation rate, ε, at current time step
PHI(I,J,K,P)	pressure
PHI(I,J,K,PP)	pressure-correction, p'
PHI(I,J,K,TE)	turbulent kinetic energy k at current time step
PHI(I,J,K,U)	U-velocity component at current time step
PHI(I,J,K,V)	V-velocity component at current time step
PHI(I,J,K,W)	W-velocity component at current time step
PHI(I,J,K,Φ)	general representation for all dependent variables
PHIO(I,J,K,Φ)	Φ-value at old time step
PRT(T)	turbulent Prandtl number for temperature
PRT(ED)	turbulent Prandtl number for energy dissipation rate
PRT(TE)	turbulent Prandtl number for turbulent kinetic energy
PRT(Φ)	turbulent Prandtl number for variable Φ
REREF(ED)	
REREF(TE)	
REREF(PP)	
REREF(T)	
REREF(U)	
REREF(V)	
REREF(W)	the residuals for Φ are normalised with REREFΦ

RESOR	residual source for individual control volume
RESTRT	logical variable which controls whether the initial field are to be read from the file SAVRES or not
SAVE	logical variable which controls whether the fields are to be saved on the file SAVRES or not
SCHEME	activating a difference scheme for the mean flow equations ('h' = hybrid, 'q' = quick, 'v' = van leer)
SCHTUR	activating a difference scheme for the turbulent equations ('h' = hybrid, 'q' = quick, 'v' = van leer)
SMP(I,J,K)	continuity error
SOLVE(NPHI)	activating the solution procedure for variable Φ
SMALL	a very small value (i.e. 10^{-20})
SP(I,J,K)	coefficient b of linearized source treatment
SU(I,J,K)	coefficient c of linearized source treatment
STEADY	logical variable which controls whether the calculation is a steady one or not
TIME	real time in unsteady calculation
TMULT	coefficient of wall shear expression
UPLUS	U^*/U
URF(NPHI)	under-relaxation factor for ε, k, P', T, U, V, W-equation
URFVIS	under-relaxation factor for turbulent viscosity μ_t
VELPAR	velocity parallel to the wall
VIS(I,J,K)	effective viscosity $(\mu + \mu_t)$
VISCOS	laminar viscosity (μ)
VISOLD	value of effective viscosity before under-relaxation
XETA	
XKSI	
XZETA	
YETA	
YKSI	
YZETA	
ZETA	
ZKSI	
ZZETA	coefficient for calculation of gradients [see, eq. (90)]
XC(I,J,K)	x co-ordinate of north-east corner of scalar control volume
YC(I,J,K)	y co-ordinate of north-east corner of scalar control volume
ZC(I,J,K)	z co-ordinate of north-east corner of scalar control volume