Publikation Nr 88/1

ISSN 02809265

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för
Tillämpad termodynamik och strömningslära

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Applied Thermodynamics
and Fluid Mechanics

FLUX2D: A Finite-Volume Computer Program Written in General Non-Orthogonal Co-ordinates for Calculation of Two-Dimensional Turbulent Flow

by

Lars Davidson and Peter Hedberg

Göteborg February 1988

CONTENTS

NOMENCLATURE

Subscript (i,j,k,ℓ) denotes covariant component; superscript (i,j,k,ℓ) denotes contravariant component. In this Nomenclature all quantities are, for convenience, given as covariant components.

a	coefficient in the discretised equations
b	source term
C_μ, $C_{1\epsilon}$, $C_{2\epsilon}$	coefficients in the turbulence model
F, G	geometric arrays (see Chapter 3)
g_{ij}	metric tensor
$\vec{g}_i$	unit base vector
g	determinant of g_{ij}
k	turbulent kinetic energy
$\vec{n}$	normal vector
p	pressure
p''	pressure correction
P_k	produktion term in the turbulence model
T	temperature
v_i	velocity component

v'_i	projected velocity component
v''_i	velocity correction
V	volume
x,y	cartesian co-ordinates
x^i	general contravariant co-ordinate; there exist no equivalent covariant component

Greek Symbols

ϵ	dissipation of turbulent kinetic energy
ϕ	general dependent variable
ρ	density
Γ	exchange coefficient
μ	dynamic viscosity
τ	time

Subscripts

ϕ	general dependent variable
t	turbulent
eff	effective

1. INTRODUCTION

FLUX2D is a general purpose computer program for two-dimensional flow. It is written in general non-orthogonal co-ordinates. FLUX2D uses the covariant velocity components; this choice (rather than contravariant components) makes the pressure-velocity coupling relatively easy to handle at the expense of more complicated expressions for the convective and diffusive fluxes.

The effects due to curvature and/or divergence/convergence of grid lines are usually taken account for via source terms. In FLUX2D a novel procedure of taking account for terms related to non-orthogonality and curvature and/or divergence/convergence of grid lines. The mathematical derivation of this procedure is described in [1]. When a velocity component is solved at a point e (or n), the neighbouring velocities are projected in the direction of the velocity component at the point e. Thus we change the base vectors at the neighbouring points, which means that the direction of the velocity components in the immediate neighbourhood of e is constant. This renders a simpler expression for the covariant derivatives. The procedure of changing the base vectors affects only the convected velocity. The convecting term (dot product of velocity and area) is calculated without any change of the base vectors. The same is true for the operator on the covariant velocity in the diffusion term.

In FLUX2D a novel procedure of taking account for terms related to non-orthogonality and curvature and/or divergence/convergence of grid lines. The mathematical derivation of this procedure is described in [1]. In [1] FLUX2D is used to calculate four laminar flows; the agreement between calculated and analytical or experimental data is good.

The code has also been applied to two turbulent flows [2], with good results.

2 EQUATIONS

2.1 MOMENTUM EQUATIONS

The momentum equations for turbulent flow in general co-ordinates, using covariant components can be written [1]

$$\frac{\partial v_i}{\partial \tau} + (\rho g^{jk} v_i v_j)_{,k} = - \frac{\partial p}{\partial x^i} + (\mu_{eff}\, g^{jk} v_{i,j})_{,k} \qquad (2.1)$$

when the effective viscosity does not vary too much; i.e. the diffusive terms

$$(\mu_{eff}\, g^{jk} v_{j,i})_{,k}$$

have been neglected , which vanish due to continuity when the density and the viscosity are constant. These terms are usually very small. The comma notation is used for denoting covariant derivative. In [1] it was shown that if a local co-ordinate system is used so that the direction of the neighbouring velocities v_{inb} (i=co-ordinate direction, nb=neighbour) are kept the same as that of the velocity (v_{1e} or v_{2n}) being solved for (see Fig. 2.1), Eq. (2.1) can be integrated and rewritten so that

$$\int_V \frac{\partial v_i}{\partial \tau}\, dV + \int_A g^{jk} \left(\rho v_i v_j - \mu_{eff} \frac{\partial v_i}{\partial x^j}\right) n_k dA + \int_A p\, n_i dA = 0 \qquad (2.2)$$

where A denotes the bounding area of the control volume V, and $\vec{n}$ is its normal vector.

It is well known that upwind differencing gives rise to numerical diffusion. For curved grids this becomes especially serious when the flow is across the grid. Even if the magnitude of the velocity component is well approximated by estimating the face value of v_1 (for example) with its node value, the direction of v_1 is not. This was recognised by Galphin et al. [3].

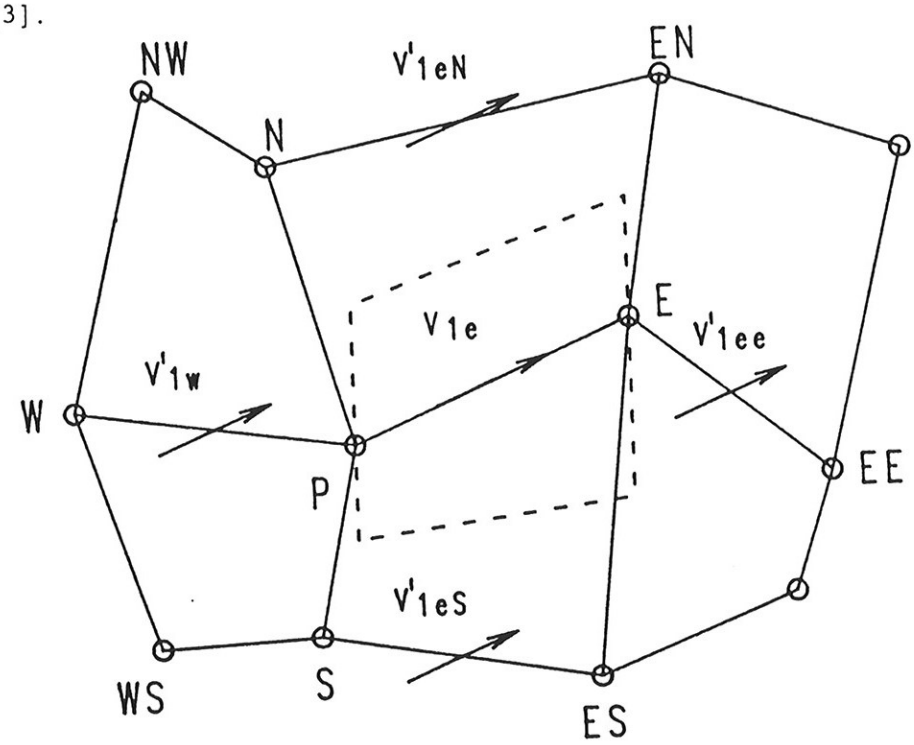


Figure 2.1 v_1-velocity control volume (dashed lines). The directions of the v_{1eN}, v_{1w}, v_{1eS} and v_{1ee} are $\overrightarrow{N(EN)}$, $\overrightarrow{WP}$, $\overrightarrow{S(ES)}$ and $\overrightarrow{E(EE)}$, respectively. The arrows show the velocity vectors projected on $\overrightarrow{PE}$, which are neighbours of v_{1e} [i.e. v'_{1eN}, v'_{1w}, v'_{1eS} and v'_{1ee}].

In the present formulation the velocity components in the immediate neighbourhood of the velocity component being solved for, all have the same direction (see Section 1 and [1]). The velocity vectors, $\vec{v}_{nb}$ (nb=neighbour), is projected in the direction of the velocity component

at the control volume e being solved for. This means that all the neighbours, v'_{1nb}, of v_{1e}, have the same direction. In this way the problem of estimating a face value of v_1 having the in-correct direction is solved. The same is true for the v_2-equation. The procedure of projecting velocities drastically reduces the numerical errors due to upwind differencing associated with curved grids [4].

2.2 SCALAR EQUATIONS

The integrated differential transport equation for a scalar variable ϕ (i.e. T, k or ϵ) can be written

$$\int_V \frac{\partial \phi}{\partial \tau} \, dV + \int_A g^{jk} \left(\rho v_j \phi - \Gamma_{eff} \frac{\partial \phi}{\partial x^j} \right) n_k \, dA + \int_V b_\phi \, dV = 0 \qquad (2.3)$$

where b_ϕ denotes the general source term, which contains terms which are not of transient, convective or diffusive type. In case ϕ=T the source term b_T=0.

2.3 k-ϵ TURBULENCE MODEL

The standard k-ϵ turbulence model is implemented in FLUX2D [5]. The source term b_ϕ for the k and ϵ-equations take the following forms:

$$b_k = \int_V (P_k - \rho\epsilon) \, dV; \quad b_\epsilon = \int_V \frac{\epsilon}{k} (C_{1\epsilon} P_k - C_{2\epsilon} \rho\epsilon) \, dV$$

where the produktion term, P_k, is calculated (without projecting the velocities) as

$$P_k = \mu_t (v_{i,j} + v_{j,i}) g^{jk} v^i{}_{,k}$$

The covariant derivatives of v_i and v^i are defined by [6]

$$v_{i,j} = \frac{\partial v_i}{\partial x^j} - \left\{ {k \atop i\ j} \right\} v_k$$

$$v^i{}_{,j} = \frac{\partial v^i}{\partial x^j} - \left\{ {i \atop k\ j} \right\} v^k$$

The Christoffel symbol can be written [6]

$$\left\{ {\ell \atop i\ j} \right\} = \frac{1}{2} g^{k\ell} \left(\frac{\partial g_{ik}}{\partial x^j} + \frac{\partial g_{jk}}{\partial x^i} - \frac{\partial g_{ij}}{\partial x^k} \right)$$

The covariant components of the metric tensor g_{ij} are defined by [6]

$$g_{11} = g_{22} = 1; \quad g_{12} = \cos\alpha$$

where α is the angle between the base vectors $\vec{g}_1$ and $\vec{g}_2$. The contravariant components g^{ij} are defined in Eq. (3.3)

The turbulent viscosity is calculated as

$$\mu_t = \rho C_\mu k^2/\epsilon$$

The constants in the turbulence model have been assigned their standard values [5]: C_μ=0.09, $C_{1\epsilon}$=1.44, $C_{2\epsilon}$=1.92, σ_k=1.0, σ_ϵ=1.3

3. DETAILS OF THE DISCRETISATION

Equation (2.2) can now be discretised using the control volume formulation described in [7]. For the v_1-equation the discretised equation may be written

$$a_p v_{1e} = \sum a_{nb} v'_{1nb} + b \qquad (3.1)$$

where the prime denotes velocity parallel to the v_{1e}-velocity. The v_2-equation is discretised in the same way. The a_e-coefficient in Eq. (3.1), for instance, contains convective contribution such as $(\rho\vec{v}\cdot\vec{A})_e$ and diffusive contribution such as $(\mu_{eff}\vec{A}\cdot\nabla)_e$ [cf. Eq. (2.1)], where $\vec{A}_e$ is the east face vector area of the control volume. The part of the diffusion terms which contains the cross-derivative due to non-orthogonality has been included in the source term, b.

To make it possible to solve the velocities using the usual TDM-Algorithm, Eq. (3.1) is rewritten so that

$$a_p v_{1e} = \sum a_{nb} v_{1nb} + b + b_{curv}$$

where the source term b_{curv} now contains

$$b_{curv} = \sum a_{nb} [v'_{1nb} - v_{1nb}]$$

The scalar equations [Eq. (2.3)] are discretised in the same manner, except that no curvature terms appear. We can write these equations as

$$a_p\phi_P = \sum a_{nb}\phi_{nb} + b \qquad (3.2)$$

The equations are solved using the SIMPLEC-algorithm [7]. The four main features are staggered grids for the velocities; formulation of the difference equations in implicit, conservative form, using hybrid upwind/central differencing; rewriting of the continuity equation into an equation for the pressure correction; and iterative solving of the equations using TDMA.

3.1 THE GRID

The grid must be generated by the user; the co-ordinates (x_c, y_c) of the corners of each scalar control volume must then be given. The corners are numbered so that ix=0 for the corners at the west (low x) boundary of the computational domain, and ix=nx at the east boundary, see Fig. 3.1. The scalar nodes (x_p, y_p) are taken to be in the centre of their control volumes so that

$$x_p(ix,iy)=0.25[x_c(ix,iy) + x_c(ix-1,iy) + x_c(ix,iy-1) + x_c(ix-1,iy-1)]$$

$$y_p(ix,iy)=0.25[y_c(ix,iy) + y_c(ix-1,iy) + y_c(ix,iy-1) + y_c(ix-1,iy-1)]$$

The straight lines between the scalar nodes define the direction of the unit covariant base vectors $\vec{g}_i$. The scalar nodes outside the computational domain are defined so that the base vectors at the boundaries are

orthogonal. Consider, for example, a control volume at a south boundary, see Fig. 3.2. The node S is placed so that the line $\overrightarrow{SP}$ is orthogonal to the line $\overrightarrow{(es)(en)}$.

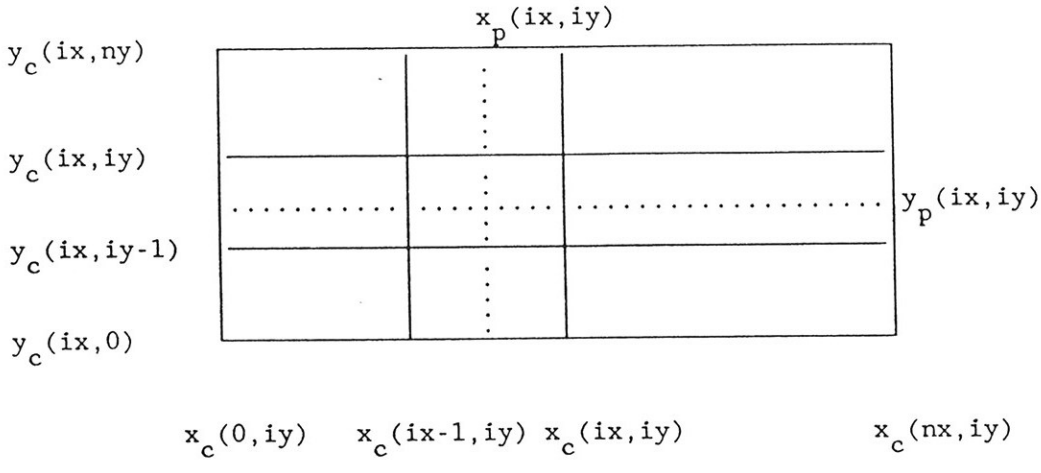


Figure 3.1 The grid. Solid lines: x_c and y_c constant; dotted lines: x_p and y_p constant.

This definition of the location of the scalar node S has the following advantages:

i) when a flux across the boundary is to be prescribed or calculated, only the derivative $\partial/\partial x^2$ is needed (i.e. the derivative $\partial/\partial x^1$ is not needed as for the general case);

ii) the boundary condition for the v_2-velocity becomes very simple: $v_2 = v^2 = 0$.

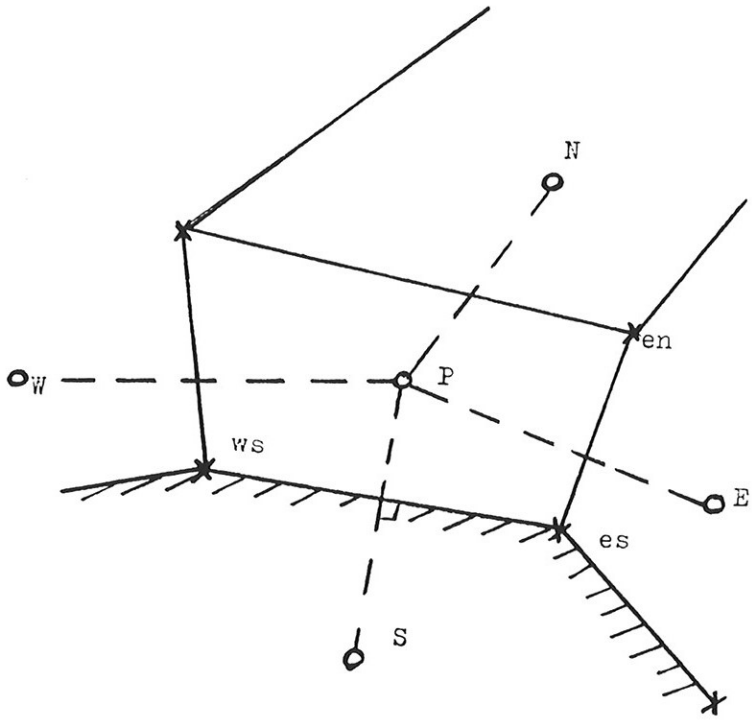


Figure 3.2 Control volume at a south boundary of the calculation domain. Circles denote scalar nodes; crosses denote corners of control volume. See Fig. 3.3 for grid nomenclature.

3.1.1 Nomenclature for the grid

The grid is shown in Fig. 3.3. Single capital letters define scalar nodes [E(ast), S(outh), etc.], and single small letters define faces of scalar control volumes. When a location can not be referred to by a single character, combinations of letters are used. The north-east corner of the scalar control volume and the v_1-control volume are referred to as 'en' and 'En', respectively. The order in which the characters appear are: first east-west and then north-south.

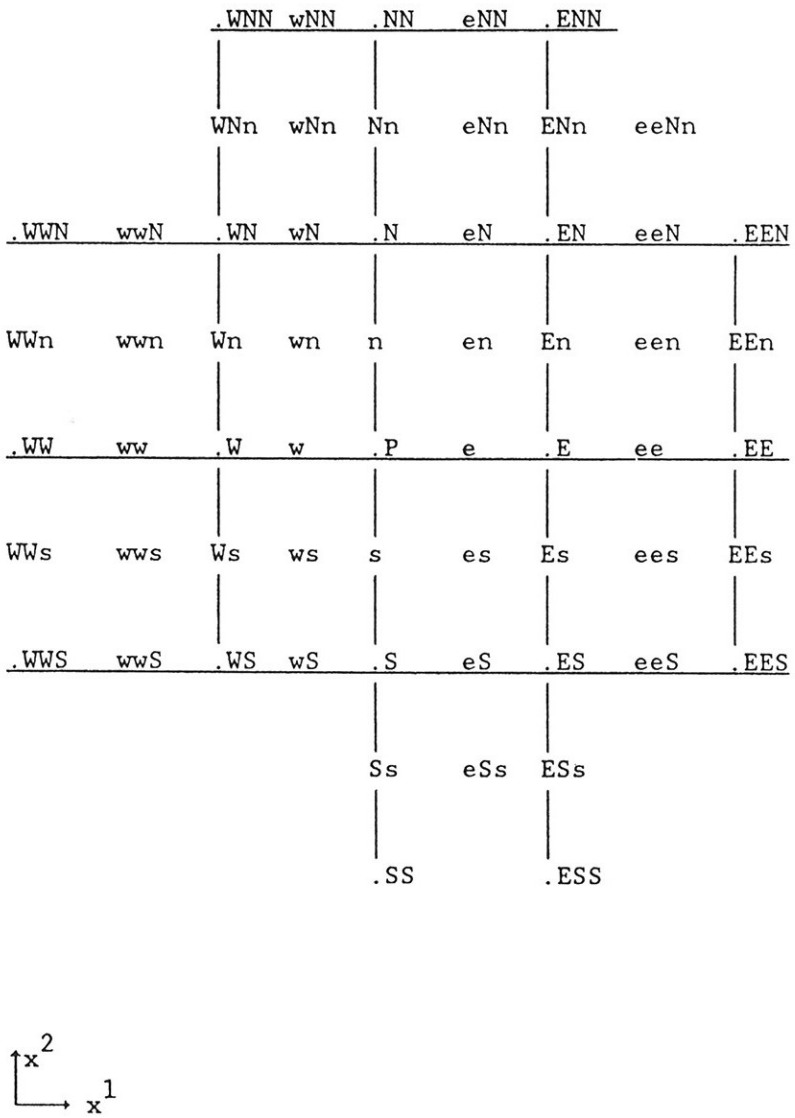


Figure 3.3. Grid nomenclature. The grid drawn orthogonal for clarity. Dots denote scalar nodes.

3.2 CONVECTION

The convection across a face of a scalar control volume is calculated as [1]

$$\rho\vec{v} \cdot \vec{A} = \rho\vec{v} \cdot \vec{n}\, A = \rho v^j\, \vec{g}_j \cdot \vec{n}\, A = \rho v_i g^{ij}\, \vec{g}_j \cdot \vec{n}\, A \qquad (3.2)$$

where A denotes the face area. The scalar product of the unit vectors $\vec{g}_j$ and $\vec{n}$ is simply $\cos\alpha$, where α is the angle between the two vectors. The contravariant components of g^{ij} are calculated as [1,6]

$$g^{11} = 1/g;\ g^{12} = -\cos\alpha/g;\ g^{21} = g^{12};\ g^{22} = 1/g;\ g = 1-\cos^2\alpha \qquad (3.3)$$

where α denotes the angle between the two appropriate vectors [$\vec{g}_j$ and $\vec{n}$ in the case of Eq. (3.2)], and g is the determinant of the covariant metric tensor g_{ij}.

3.2.1 East Face

From Eq. (3.2) we obtain

$$\rho_{up}(v_i g^{ij}\, \vec{g}_j \cdot \vec{n}\, A)_e = \rho_{up}[(v_1 g^{11}\, \vec{g}_1 + v_2 g^{21}\, \vec{g}_1 + v_1 g^{12}\, \vec{g}_2 + v_2 g^{22}\, \vec{g}_2) \cdot \vec{n}\, A]_e$$

The upwind density , ρ_{up}, is used, which enables FLUX2D to handle transonic flows [8]. The v_2- velocity is needed at face e; we take v_{2e} to be the arithmetic mean value of the four neighbours closest to face e, i.e.

$$v_{2e} = 0.25(v_{2En} + v_{2Es} + v_{2s} + v_{2n})$$

In order to calculate the g^{ij}-components in Eq. (3.3) two relevant directions (one x^1 and one x^2-direction) must be defined. We use $\overrightarrow{PE}$ and $\overrightarrow{(es)(en)}$; the latter is taken as

$$\overrightarrow{(es)(en)} = 0.25[\overrightarrow{E(EN)} + \overrightarrow{(ES)E} + \overrightarrow{SP} + \overrightarrow{PN}]$$

The product $(\vec{v} \cdot \vec{n})_e$ can be written as

$$F_u v_{1e} + G_u v_{2e}$$

where F_u and G_u contain all geometrical quantities, which FLUX2D calculates once and for all.

The convective flux through the east faces of the v_1 and v_2-control volumes is taken as the arithmetic mean value of the fluxes at face e and ee, and at face e and eN, respectively.

3.2.2 North Face

From Eq. (3.2) we obtain

$$\rho_{up}(v_i g^{ij}\,\vec{g}_j \cdot \vec{n}\,A)_n = \rho_{up}[(v_1 g^{11}\,\vec{g}_1 + v_2 g^{21}\,\vec{g}_1 + v_1 g^{12}\,\vec{g}_2 +$$

$$+ v_2 g^{22}\,\vec{g}_2) \cdot \vec{n}\,A]_n$$

Again the upwind density , ρ_{up}, is used. The v_1- velocity is needed at face n; we take v_{1n} to be the arithmetic mean value of the four neighbours closest to face n, i.e.

$$v_{1n} = 0.25(v_{1eN} + v_{1e} + v_{1w} + v_{1wN})$$

In order to calculate the g^{ij}-components in Eq. (3.3) two relevant directions (one x^1 and one x^2-direction) must be defined. We use $\overrightarrow{PN}$ and $\overrightarrow{(wn)(en)}$; the latter is taken as

$$\overrightarrow{(wn)(en)} = 0.25[\overrightarrow{N(EN)} + \overrightarrow{PE} + \overrightarrow{WP} + \overrightarrow{(WN)N}]$$

The product $(\vec{v} \cdot \vec{n})_n$ can be written as

$$F_v v_{1n} + G_v v_{2n}$$

where F_v and G_v contain all geometrical quantities, which FLUX2D calculates once and for all.

The convective flux through the north faces of the v_1 and v_2-control volumes is taken as the arithmetic mean value of the fluxes at face n and En, and at face n and nn, respectively.

3.3 DIFFUSION

The diffusive flux across a face of a scalar control volume is calculated as

$$\Gamma_{eff}\,\vec{A}\cdot\nabla = \Gamma_{eff}\,\vec{A}\cdot\vec{g}^{j}\,\frac{\partial}{\partial x^{j}} = \Gamma_{eff}\,A\,\vec{n}\cdot\vec{g}_{i}\,g^{ij}\,\frac{\partial}{\partial x^{j}} \qquad (3.4)$$

3.3.1 East Face

The diffusive flux through the face consist of two parts: the flux due to $\partial/\partial x^{1}$, and the flux due to $\partial/\partial x^{2}$ (cross-derivative term; this term vanishes for orthogonal grids). The first part can be written [see Eq. (3.4)]

$$[(\Gamma_{eff}\,A\,\vec{n})\cdot(\vec{g}_{1}g^{11} + \vec{g}_{2}g^{21})\,\frac{\partial}{\partial x^{1}}]_{e}$$

This part is included in the a_{e}-coefficient [see Eqs. (3.1-2)]. The second part is given by

$$[(\Gamma_{eff}\,A\,\vec{n})\cdot(\vec{g}_{1}g^{12} + \vec{g}_{2}g^{22})\,\frac{\partial}{\partial x^{2}}]_{e}$$

This part is included in the source term.

The total diffusion can be written

$$[A\,\Gamma_{eff}(F_{u}\,\frac{\partial}{\partial x^{1}} + G_{u}\,\frac{\partial}{\partial x^{2}})]_{e}$$

Note that the same geometrical factors are used for the convective and for the diffusive fluxes; this is because v_1& v_2 as well as $\partial/\partial x^1$& $\partial/\partial x^2$ are covariant components.

The diffusive flux through the east faces of the v_1 and v_2-control volumes is taken as the arithmetic mean value of the fluxes at face e and ee, and at face e and eN, respectively.

3.3.2 North Face

The diffusive flux through the face consist of two parts: the flux due to $\partial/\partial x^2$, and the flux due to $\partial/\partial x^1$ (cross-derivative term; this term vanishes for orthogonal grids). The first part can be written [see Eq. (3.4)]

$$[(\Gamma_{eff}\, A\, \vec{n}) \cdot (\vec{g}_1 g^{12} + \vec{g}_2 g^{22})\, \frac{\partial}{\partial x^2}]_n$$

This part is included in the a_n-coefficient [see Eqs. (3.1-2)]. The second part is given by

$$[(\Gamma_{eff}\, A\, \vec{n}) \cdot (\vec{g}_1 g^{11} + \vec{g}_2 g^{21})\, \frac{\partial}{\partial x^1}]_n$$

This part is included in the source term.

The total diffusion can be written

$$[A\, \Gamma_{eff} (F_v \frac{\partial}{\partial x^1} + G_v \frac{\partial}{\partial x^2})]_n$$

Note that the same geometrical factors are used for the convective and for the diffusive fluxes; this is because v_1& v_2 as well as $\partial/\partial x^1$& $\partial/\partial x^2$ are covariant components.

The diffusive flux through the north faces of the v_1 and v_2-control volumes is taken as the arithmetic mean value of the fluxes at face n and En, and at face n and nn, respectively.

3.4 CALCULATION OF THE NEIGHBOURS OF v_{1e} AND v_{2n}

The neighbour velocities, v'_{inb}, are obtained by calculating the velocity vectors, $\vec{v}_{nb}$, and then projecting them in the direction of the velocity being solved for (v_{1e} or v_{2n}). We can write this procedure as

$$v'_{1nb} = \vec{v}_{nb} \cdot \vec{g}_{1e} = (v^j \vec{g}_j)_{nb} \cdot \vec{g}_{1e} = (g^{jk} v_k \vec{g}_j)_{nb} \cdot \vec{g}_{1e}$$

$$= [(g^{11} v_1 + g^{12} v_2) \vec{g}_1 + (g^{21} v_1 + g^{22} v_2) \vec{g}_2]_{nb} \cdot \vec{g}_{1e} \qquad (3.5a)$$

$$v'_{1nb} = \vec{v}_{nb} \cdot \vec{g}_{2n} = (v^j \vec{g}_j)_{nb} \cdot \vec{g}_{2n} = (g^{jk} v_k \vec{g}_j)_{nb} \cdot \vec{g}_{2n}$$

$$= [(g^{11} v_1 + g^{12} v_2) \vec{g}_1 + (g^{21} v_1 + g^{22} v_2) \vec{g}_2]_{nb} \cdot \vec{g}_{2n} \qquad (3.5b)$$

The scalar products of the unit base vectors are defined by:

$$\vec{g}_1 \cdot \vec{g}_2 = \cos\alpha; \quad \vec{g}_1 \cdot \vec{g}_1 = 1; \quad \vec{g}_2 \cdot \vec{g}_2 = 1$$

where α is the angle between $\vec{g}_1$ and $\vec{g}_2$.

3.4.1 West Neighbour of v_{1e}

In order to calculate v'_{1w} using Eq. (3.5a) we must first calculate g^{ij} and v_{2w}. We proceed as in Section 3.2.1; we calculate v_{2w} as

$$v_{2w} = 0.25(v_{2n} + v_{2s} + v_{2Ws} + v_{2Wn})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at w: $\overrightarrow{WP}$ and $\overrightarrow{(ws)(wn)}$ are taken. The latter is calculated as

$$\overrightarrow{(ws)(wn)} = 0.25[\overrightarrow{PN} + \overrightarrow{SP} + \overrightarrow{(WS)W} + \overrightarrow{W(WN)}] = \vec{g}_{2w}$$

The neighbour velocity v'_{1w} can be written

$$v'_{1w} = F_{uw}v_{1w} + G_{uw}v_{2w}$$

where F_{uw} and G_{uw} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.4.2 East Neighbour of v_{1e}

In order to calculate v'_{1ee} using Eq. (3.5a) we must first calculate g^{ij} and v_{2ee}. We proceed as in Section 3.2.1; we calculate v_{2ee} as

$$v_{2ee} = 0.25(v_{2EEn} + v_{2EEs} + v_{2Es} + v_{2En})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at ee: $\overrightarrow{E(EE)}$ and $\overrightarrow{(ees)(een)}$ are taken. The latter is calculated as

$$\overrightarrow{(ees)(een)} = 0.25[\overrightarrow{(EE)(EEN)} + \overrightarrow{(ESS)(EE)} + \overrightarrow{(ES)E} + \overrightarrow{E(EN)}] = \vec{g}_{2ee}$$

The neighbour velocity v'_{1ee} can be written

$$v'_{1ee} = F_{ue}v_{1ee} + G_{ue}v_{2ee}$$

where F_{ue} and G_{ue} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.4.3 South Neighbour of v_{1e}

In order to calculate v'_{1eS} using Eq. (3.5a) we must first calculate g^{ij} and v_{2eS}. We proceed as in Section 3.2.1; we calculate v_{2eS} as

$$v_{2eS} = 0.25(v_{2Es} + v_{2ESs} + v_{2Ss} + v_{2s})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at eS: $\overrightarrow{S(ES)}$ and $\overrightarrow{(eSs)(es)}$ are taken. The latter is calculated as

$$\overrightarrow{(eSs)(es)} = 0.25[\overrightarrow{(ES)E} + \overrightarrow{(ESS)(ES)} + \overrightarrow{(SS)S} + \overrightarrow{SP}] = \vec{g}_{2eS}$$

The neighbour velocity v'_{1eS} can be written

$$v'_{1eS} = F_{us} v_{1eS} + G_{us} v_{2eS}$$

where F_{us} and G_{us} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.4.4 North Neighbour of v_{1e}

In order to calculate v'_{1eN} using Eq. (3.5a) we must first calculate g^{ij} and v_{2eN}. We proceed as in Section 3.2.1; we calculate v_{2eN} as

$$v_{2eN} = 0.25(v_{2ENn} + v_{2En} + v_{2n} + v_{2Nn})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at eN: $\overrightarrow{N(EN)}$ and $\overrightarrow{(en)(eNn)}$ are taken. The latter is calculated as

$$\overrightarrow{(en)(eNn)} = 0.25[\overrightarrow{(EN)(ENN)} + \overrightarrow{E(EN)} + \overrightarrow{PN} + \overrightarrow{N(NN)}] = \vec{g}_{2eN}$$

The neighbour velocity v'_{1eN} can be written

$$v'_{1eN} = F_{un} v_{1eN} + G_{un} v_{2eN}$$

where F_{un} and G_{un} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.4.5 West Neighbour of v_{2n}

In order to calculate v'_{2Wn} using Eq. (3.5b) we must first calculate g^{ij} and v_{1Wn}. We proceed as in Section 3.2.2; we calculate v_{1Wn} as

$$v_{1Wn} = 0.25(v_{1wN} + v_{1w} + v_{1ww} + v_{1wwN})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at Wn: $\overrightarrow{W(WN)}$ and $\overrightarrow{(wwn)(wn)}$ are taken. The latter is calculated as

$$\overrightarrow{(wwn)(wn)} = 0.25[\overrightarrow{(WN)N} + \overrightarrow{WP} + \overrightarrow{(WW)W} + \overrightarrow{(WWN)(WN)}] = \vec{g}_{1Wn}$$

The neighbour velocity v'_{2Wn} can be written

$$v'_{2Wn} = F_{vw}v_{1Wn} + G_{vw}v_{2Wn}$$

where F_{vw} and G_{vw} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.4.6 East Neighbour of v_{2n}

In order to calculate v'_{2En} using Eq. (3.5b) we must first calculate g^{ij} and v_{1En}. We proceed as in Section 3.2.2; we calculate v_{1En} as

$$v_{1En} = 0.25(v_{1eeN} + v_{1ee} + v_{1e} + v_{1eN})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at En: $\overrightarrow{E(EN)}$ and $\overrightarrow{(en)(een)}$ are taken. The latter is calculated as

$$\overrightarrow{(en)(een)} = 0.25[\overrightarrow{(EN)(EEN)} + \overrightarrow{E(EE)} + \overrightarrow{PE} + \overrightarrow{N(EN)}] = \vec{g}_{1En}$$

The neighbour velocity v'_{2En} can be written

$$v'_{2En} = F_{ve} v_{1En} + G_{ve} v_{2En}$$

where F_{ve} and G_{ve} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.4.7 South Neighbour of v_{2n}

In order to calculate v'_{2s} using Eq. (3.5b) we must first calculate g^{ij} and v_{1s}. We proceed as in Section 3.2.2; we calculate v_{1s} as

$$v_{1s} = 0.25(v_{1e} + v_{1eS} + v_{1wS} + v_{1w})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at s: $\overrightarrow{SP}$ and $\overrightarrow{(ws)(es)}$ are taken. The latter is calculated as

$$\overrightarrow{(ws)(es)} = 0.25[\overrightarrow{PE} + \overrightarrow{S(ES)} + \overrightarrow{(WS)S} + \overrightarrow{WP}] = \vec{g}_{1s}$$

The neighbour velocity v'_{2s} can be written

$$v'_{2s} = F_{vs} v_{1s} + G_{vs} v_{2s}$$

where F_{vs} and G_{vs} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.4.8 North Neighbour of v_{2n}

In order to calculate v'_{2Nn} using Eq. (3.5b) we must first calculate g^{ij} and v_{1Nn}. We proceed as in Section 3.2.2; we calculate v_{1Nn} as

$$v_{1Nn} = 0.25(v_{1eNN} + v_{1eN} + v_{1wN} + v_{1wNN})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at Nn: $\overrightarrow{N(NN)}$ and $\overrightarrow{(wNn)(eNn)}$ are taken. The latter is calculated as

$$\overrightarrow{(wNn)(eNn)} = 0.25[\overrightarrow{(NN)(ENN)} + \overrightarrow{N(EN)} + \overrightarrow{(WN)N} + \overrightarrow{(WNN)(NN)}] = \vec{g}_{1Nn}$$

The neighbour velocity v'_{2Nn} can be written

$$v'_{2Nn} = F_{vn} v_{1Nn} + G_{vn} v_{2Nn}$$

where F_{vn} and G_{vn} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5 DIFFUSION DUE TO CROSS-DERIVATIVES

The diffusion flux of a scalar variable, ϕ, across the east face of its control volume is given by (see Section 3.3.1)

$$[(\Gamma_{eff} \, A \, \vec{n}) \cdot (\vec{g}_1 g^{12} + \vec{g}_2 g^{22}) \frac{\partial \phi}{\partial x^2}]_e$$

The values of ϕ at the east corners of the control volume, ϕ_{en} and ϕ_{es}, are needed in order to calculate the derivative $\partial\phi/\partial x^2$. The value ϕ_{en} is taken as

$$\phi_{en} = 0.25(\phi_{EN} + \phi_E + \phi_P + \phi_N)$$

The remaining corner values, ϕ_{es}, ϕ_{ws} and ϕ_{wn}, are calculating in the same way.

The diffusion flux of momentum across the east face of the control volume due to the cross-derivatives in the v_1-equation, for example, is given by (see Section 3.3.1)

$$[(\mu_{eff} \, A \, \vec{n}) \cdot (\vec{g}_1 g^{12} + \vec{g}_2 g^{22}) \frac{\partial v_1}{\partial x^2}]_E$$

In order to calculate the derivative $(\partial v_1/\partial x^2)_E$ we need the v_1'-velocities at the east corners of the v_1-control volume, i.e. v_{1En} and v_{1Es}. When the derivative at the west face, $(\partial v_1/\partial x^2)_P$, is to be calculated we need v_{1n}' and v_{1s}'. The four projected velocities (v_{1nb}') at the corners are also

used when the diffusive flux of momentum across the south and north faces of the v_1-control volume, due to the cross-derivatives, are calculated.

The projected velocities v'_{2eN}, v'_e, v'_{2w} and v'_{2wN} are needed in order to calculate the corresponding fluxes of momentum in the v_2-equation.

3.5.1 North-east Neighbour of v_{1e}

In order to calculate v'_{1En} using Eq. (3.5a) we must first calculate g^{ij} and v_{1En}. We proceed as in Section 3.2.1; we calculate v_{1En} as

$$v_{1En} = 0.25(v_{1eeN} + v_{1ee} + v_{1e} + v_{1eN})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at En: $\overrightarrow{E(EN)}$ and $\overrightarrow{(en)(een)}$ are taken. The latter is calculated as

$$\overrightarrow{(en)(een)} = 0.25[\overrightarrow{(EN)(EEN)} + \overrightarrow{E(EE)} + \overrightarrow{PE} + \overrightarrow{N(EN)}] = \vec{g}_{1En}$$

The neighbour velocity v'_{1En} can be written

$$v'_{1En} = F_{uen} v_{1en} + G_{uen} v_{2en}$$

where F_{uen} and G_{uen} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5.2 South-east Neighbour of v_{1e}

In order to calculate v'_{1Es} using Eq. (3.5a) we must first calculate g^{ij} and v_{1Es}. We proceed as in Section 3.2.1; we calculate v_{1Es} as

$$v_{1Es} = 0.25(v_{1ee} + v_{1eeS} + v_{1eS} + v_{1e})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at Es: $\overrightarrow{(ES)E}$ and $\overrightarrow{(es)(ees)}$ are taken. The latter is calculated as

$$\overrightarrow{(es)(ees)} = 0.25[\overrightarrow{E(EE)} + \overrightarrow{(ES)(EES)} + \overrightarrow{S(ES)} + \overrightarrow{PE}] = \vec{g}_{1Es}$$

The neighbour velocity v'_{1Es} can be written

$$v'_{1Es} = F_{ues} v_{1Es} + G_{ues} v_{2Es}$$

where F_{ues} and G_{ues} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5.3 South-west Neighbour of v_{1e}

In order to calculate v'_{1s} using Eq. (3.5a) we must first calculate g^{ij} and v_{1s}. We proceed as in Section 3.2.1; we calculate v_{1s} as

$$v_{1s} = 0.25(v_{1e} + v_{1eS} + v_{1wS} + v_{1w})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at s: $\overrightarrow{SP}$ and $\overrightarrow{(ws)(es)}$ are taken. The latter is calculated as

$$\overrightarrow{(ws)(es)} = 0.25[\overrightarrow{PE} + \overrightarrow{S(ES)} + \overrightarrow{(WS)S} + \overrightarrow{WP}] = \vec{g}_{1s}$$

The neighbour velocity v'_{1s} can be written

$$v'_{1s} = F_{uws} v_{1s} + G_{uws} v_{2s}$$

where F_{uws} and G_{uws} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5.4 North-west Neighbour of v_{1e}

In order to calculate v'_{1n} using Eq. (3.5a) we must first calculate g^{ij} and v_{1n}. We proceed as in Section 3.2.1; we calculate v_{1n} as

$$v_{1n} = 0.25(v_{1eN} + v_{1e} + v_{1w} + v_{1wN})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at n: $\overrightarrow{PN}$ and $\overrightarrow{(wn)(en)}$ are taken. The latter is calculated as

$$\overrightarrow{(wn)(en)} = 0.25[\overrightarrow{N(EN)} + \overrightarrow{PE} + \overrightarrow{WP} + \overrightarrow{(WN)N}] = \vec{g}_{1n}$$

The neighbour velocity v'_{1n} can be written

$$v'_{1n} = F_{uwn} v_{1n} + G_{uwn} v_{2n}$$

where F_{uwn} and G_{uwn} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5.5 North-east Neighbour of v_{2n}

In order to calculate v'_{2eN} using Eq. (3.5b) we must first calculate g^{ij} and v_{2eN}. We proceed as in Section 3.2.2; we calculate v_{2eN} as

$$v_{2eN} = 0.25(v_{2ENn} + v_{2En} + v_{2n} + v_{2Nn})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at eN: $\overrightarrow{N(EN)}$ and $\overrightarrow{(en)(eNn)}$ are taken. The latter is calculated as

$$\overrightarrow{(en)(eNn)} = 0.25[\overrightarrow{(EN)(ENN)} + \overrightarrow{E(EN)} + \overrightarrow{PN} + \overrightarrow{N(NN)}] = \vec{g}_{2eN}$$

The neighbour velocity v'_{2eN} can be written

$$v'_{2eN} = F_{ven} v_{1eN} + G_{ven} v_{2eN}$$

where F_{ven} and G_{ven} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5.6 South-east Neighbour of v_{2n}

In order to calculate v'_{2e} using Eq. (3.5b) we must first calculate g^{ij} and v_{2e}. We proceed as in Section 3.2.2; we calculate v_{2e} as

$$v_{2e} = 0.25(v_{2En} + v_{2Es} + v_{2s} + v_{2n})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at e: $\overrightarrow{PE}$ and $\overrightarrow{(es)(en)}$ are taken. The latter is calculated as

$$\overrightarrow{(es)(en)} = 0.25[\overrightarrow{E(EN)} + \overrightarrow{(ES)E} + \overrightarrow{SP} + \overrightarrow{PN}] = \vec{g}_{2e}$$

The neighbour velocity v'_{2e} can be written

$$v'_{2e} = F_{ves} v_{1e} + G_{ves} v_{2e}$$

where F_{ves} and G_{ves} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5.7 South-west Neighbour of v_{2n}

In order to calculate v'_{2w} using Eq. (3.5b) we must first calculate g^{ij} and v_{2w}. We proceed as in Section 3.2.2; we calculate v_{2w} as

$$v_{2w} = 0.25(v_{2n} + v_{2s} + v_{2Ws} + v_{2Wn})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at w: $\overrightarrow{WP}$ and $\overrightarrow{(ws)(wn)}$ are taken. The latter is calculated as

$$\overrightarrow{(ws)(wn)} = 0.25[\overrightarrow{PN} + \overrightarrow{SP} + \overrightarrow{(WS)W} + \overrightarrow{W(WN)}] = \vec{g}_{2w}$$

The neighbour velocity v'_{2w} can be written

$$v'_{2w} = F_{vws}v_{1w} + G_{vws}v_{2w}$$

where F_{vws} and G_{vws} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.5.8 North-west Neighbour of v_{2n}

In order to calculate v'_{2wN} using Eq. (3.5b) we must first calculate g^{ij} and v_{2wN}. We proceed as in Section 3.2.2; we calculate v_{2wN} as

$$v_{2wN} = 0.25(v_{2Nn} + v_{2n} + v_{2Wn} + v_{2WNn})$$

When calculating g^{ij} using Eq. (3.3) two directions (x^1 and x^2) are needed at wN: $\overrightarrow{(WN)N}$ and $\overrightarrow{(wn)(wNn)}$ are taken. The latter is calculated as

$$\overrightarrow{(wn)(wNn)} = 0.25[\overrightarrow{N(NN)} + \overrightarrow{PN} + \overrightarrow{W(WN)} + \overrightarrow{(WN)(WNN)}] = \vec{g}_{2wN}$$

The neighbour velocity v'_{2wN} can be written

$$v'_{2wN} = F_{vwn} v_{1wN} + G_{vwn} v_{2wN}$$

where F_{vwn} and G_{vwn} contain all geometrical quantities and which FLUX2D calculates once and for all.

3.6 THE PRESSURE CORRECTION EQUATION

The SIMPLEC algorithm [9] is implemented in FLUX2D. SIMPLEC (as well as SIMPLE [7]) is a one stage correction algorithm. The continuity equation is turned into an equation for the pressure correction. Velocities and pressure are expressed as old values (*) plus a correction ("), i.e.

$$v_1 = v_1^* + v_1''; \; v_2 = v_2^* + v_2''; \; p = p^* + p''$$

and the velocity corrections and the pressure correction are related as

$$v_1'' = \text{const.} \frac{\partial p''}{\partial x^1} \; ; \; v_2'' = \text{const.} \frac{\partial p''}{\partial x^2}$$

The momentum equations are first solved using old pressure (p^*); the pressure correction equation is then solved, and the obtained pressure correction is used to correct velocities and pressure.

The discretisation of the pressure correction equation is now described. The mass flux through the east face of the control volume is given by (see Section 3.2.1)

$$m_e = m_e^* + m_e'' = \rho_{up} A_e [F_u (v_{1e}^* + v_{1e}'') + G_u (v_{2e}^* + v_{2e}'')]$$

$$= m_e^* + \rho_{up} A_e (F_u v_{1e}'' + G_u v_{2e}'')$$

The relation between v_{1e}'', v_{2e}'' and the pressure correction, p", is given by [7,9]

$$v_{1e}'' = \frac{A_e}{a_e^{v_1} - \sum a_{nb}^{v_1}} (p_P'' - p_E'') = D_{ue} (p_P'' - p_E'') \qquad (3.6a)$$

$$v_{2e}'' = \frac{A_e}{a_e^{v_2} - \sum a_{nb}^{v_2}} (p_{es}'' - p_{en}'') = D_{ve} (p_{es}'' - p_{en}'') \qquad (3.6b)$$

The superscripts v_1 and v_2 refer to which momentum equation the coefficients a_e and a_{nb} belong. It may seem that the denominators in Eq. (3.6) are zero, since, generally

$$a_P^\phi = \sum a_{nb}^\phi - S_P^\phi$$

and S_P^ϕ normally is zero for the momentum equations. The denominators in Eq. (3.6) are, however, not zero because the coefficient $a_e^{v_1}$ and $a_e^{v_2}$ have been under-relaxated before they are used in Eq. (3.6).

We obtain now the following expression for m_e:

$$m_e = m_e^* + \rho_{up} A_e [F_u D_{ue} (p''_P - p''_E) + G_u D_{ve} (p''_{es} - p''_{en})]$$

In the same way the relation between v''_{1n}, v''_{2n} and p" is given by

$$v''_{1n} = \frac{A_n}{a_n^{v_1} - \sum a_{nb}^{v_1}} (p''_{wn} - p''_{en}) = D_{un} (p''_{wn} - p''_{en})$$

$$v''_{2n} = \frac{A_n}{a_n^{v_2} - \sum a_{nb}^{v_2}} (p''_P - p''_N) = D_{vn} (p''_P - p''_N)$$

so that

$$m_n = m_n^* + \rho_{up} A_n [F_v D_{un} (p''_{wn} - p''_{en}) + G_v D_{vn} (p''_P - p''_N)]$$

The correction expressions for m_w and m_s are obtained in the same way. The pressure equation can now be derived from the continuity equation as

$$a_P p''_P = a_W p''_W + a_E p''_E + a_S p''_S + a_N p''_N + b + b_{NO}$$

where

$a_E = \rho_{up} A_e F_u D_{ue}$; $a_W = \rho_{up} (A_e F_u D_{ue})_{ix-1}$;

$a_N = \rho_{up} A_n G_v D_{vn}$; $a_S = \rho_{up} (A_n G_v D_{vn})_{iy-1}$;

$a_P = a_E + a_W + a_N + a_S$; $b = m_w^* - m_e^* + m_s^* - m_n^* + (\rho_P^o - \rho_P)\delta V/\delta t$;

b_{NO}= [NO=Non-Orthogonality]=

$$\rho_{up}(A_e G_u D_{ve})_{ix-1}(p''_{ws} - p''_{wn}) - \rho_{up} A_e G_u D_{ve}(p''_{es} - p''_{en})$$

$$+ \rho_{up}(A_n F_v D_{un})_{iy-1}(p''_{ws} - p''_{es}) - \rho_{up} A_n F_v D_{un}(p''_{wn} - p''_{en})$$

In the current version of FLUX2D the source term b_{NO} is included in the code but it is made inactive, since it has been found that it does not increase the convergence rate; in some cases even the reverse.

3.7 CYCLIC BOUNDARY CONDITIONS

In many applications the flow is cyclic in one direction, i.e. it repeats itself every Lth meter, where L is the length of the calculation domain. By using the cyclic option this periodicity is taken advantage of in FLUX2D, by solving for L meters in the x-direction only. The special solution algorithm is called CTDMA (Cyclic Tri Diagonal Matrix Algorithm).

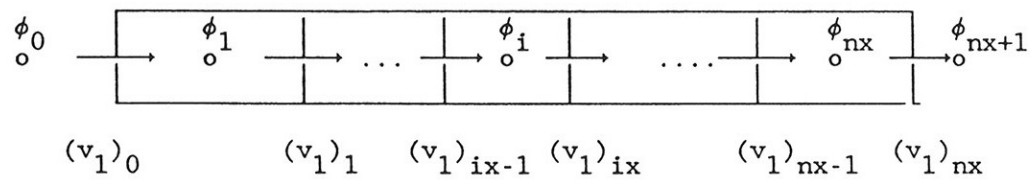


Figure 3.4. Configuration where cyclic boundary conditions are used.

For a configuration with cyclic boundary conditions in the x- direction (see Fig. 3.4), we obtain a discretised equation system as follows:

$$
\begin{aligned}
& b_1\phi_1 - c_1\phi_2 && -a_1\phi_{nx} && = d_1 \\
& -a_i\phi_{i-1} + b_i\phi_i - c_i\phi_{i+1} && && = d_i \\
& -c_{nx}\phi_1 && -a_{nx}\phi_{nx-1} + b_{nx}\phi_{nx} && = d_{nx}
\end{aligned}
\qquad (3.6)
$$

Note that the west neighbour of ϕ_1 is ϕ_{nx}, and that the east neighbour of ϕ_{nx} is ϕ_1. The v_1-velocity is normally solved for from ix=1 to ix=nx-1. When cyclic boundary conditions are used v_1 is solved for from ix=1 to ix=nx since the velocity <u>at</u> the boundary must be calculated; note that $(v_1)_0 = (v_1)_{nx}$.

With a matrix formulation we get

$$
\begin{bmatrix}
b_1 & -c_1 & & & & & -a_1 \\
-a_2 & b_2 & -c_2 & & & & \\
 & -a_i & b_i & -c_i & & & \\
 & & & & -a_{nx-1} & b_{nx-1} & -c_{nx-1} \\
-c_{nx} & & & & & -a_{nx} & b_{nx}
\end{bmatrix}
\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_i \\ \phi_{nx-1} \\ \phi_{nx} \end{bmatrix}
=
\begin{bmatrix} d_1 \\ d_2 \\ d_i \\ d_{nx-1} \\ d_{nx} \end{bmatrix}
$$

Due to the appearance of the corner elements a_1 and c_{nx} we can not use the standard TDM algorithm.

We start by eliminating the a's. Using Gauss elimination we obtain

$$\begin{bmatrix} \beta_1 & -c_1 & & & -\alpha_1 \\ & \beta_2 & -c_2 & & -\alpha_2 \\ & & & & \\ & & \beta_i & -c_i & -\alpha_i \\ & & & & \\ & & & \beta_{nx-1} & -S_{nx-1} \\ -c_{nx} & & & & \beta_{nx} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \\ \phi_i \\ \\ \phi_{nx-1} \\ \phi_{nx} \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \\ \gamma_i \\ \\ \gamma_{nx-1} \\ \gamma_{nx} \end{bmatrix}$$

where

$$\beta_1 = b_1 \tag{3.7a}$$

$$\beta_i = b_i - \frac{c_{i-1} a_i}{\beta_{i-1}} \tag{3.7b}$$

$$\alpha_1 = a_1 \tag{3.8a}$$

$$\alpha_i = \frac{\alpha_{i-1}\, a_i}{\beta_{i-1}} \tag{3.8b}$$

$$\beta_{nx-1} = b_{nx-1} - \frac{c_{nx-2}\, a_{nx-1}}{\beta_{nx-2}} \tag{3.9}$$

$$S_{nx-1} = c_{nx-1} + \frac{\alpha_{nx-1}\, a_{nx-1}}{\beta_{nx-2}} \tag{3.10}$$

$$\beta_{nx} = b_{nx} - \frac{S_{nx-1} a_{nx}}{\beta_{nx-1}} \tag{3.11}$$

c_{nx} does not change

$$\gamma_1 = d_1 \tag{3.12a}$$

$$\gamma_i = d_i + \frac{\gamma_{i-1} a_i}{\beta_{i-1}} \tag{3.12b}$$

Note that the coefficients in Eqs. (3.7), (3.8) and (3.12) are the same as for the TDM algorithm. Elimination of the c's from our matrix system gives

$$\begin{bmatrix} \beta_1 & & & & -S_1 \\ & \beta_2 & & & -S_2 \\ & & \beta_i & & -S_i \\ & & & \beta_{nx-1} & -S_{nx-1} \\ -c_{nx} & & & & \beta_{nx} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_i \\ \phi_{nx-1} \\ \phi_{nx} \end{bmatrix} = \begin{bmatrix} G_1 \\ G_2 \\ G_i \\ G_{nx-1} \\ G_{nx} \end{bmatrix}$$

where

$$S_i = \alpha_i + \frac{S_{i+1}\, c_i}{\beta_{i+1}} \tag{3.13}$$

$$G_i = \gamma_i + \frac{G_{i+1}\, c_i}{\beta_{i+1}} \tag{3.14a}$$

$$G_{nx-1} = \gamma_{nx-1};\ G_{nx} = \gamma_{nx} \tag{3.14b}$$

From the matrix we can write

$$\phi_i = \frac{1}{\beta_i}(S_i \phi_{nx} + G_i),\ i=1,2,\ldots,nx-1 \tag{3.15a}$$

From Eq. (3.15a) we get an expression for ϕ_1 and from the matrix we obtain an expression for ϕ_{nx} so that

$$\phi_{nx} = \frac{G_1 c_{nx} + G_{nx}\beta_1}{\beta_{nx}\beta_1 - S_1 c_{nx}} \tag{3.15b}$$

The solution procedure can now be summarized as:

a) Calculate the matrix elements α, β and S_{nx-1} from Eqs. (3.7-11).

b) Calculate the matrix elements γ, S and G from Eqs. (3.12-14).

c) Calculate ϕ_{nx} from Eq. (3.15b).

d) Calculate the rest of the ϕ's from Eq. (3.15a)

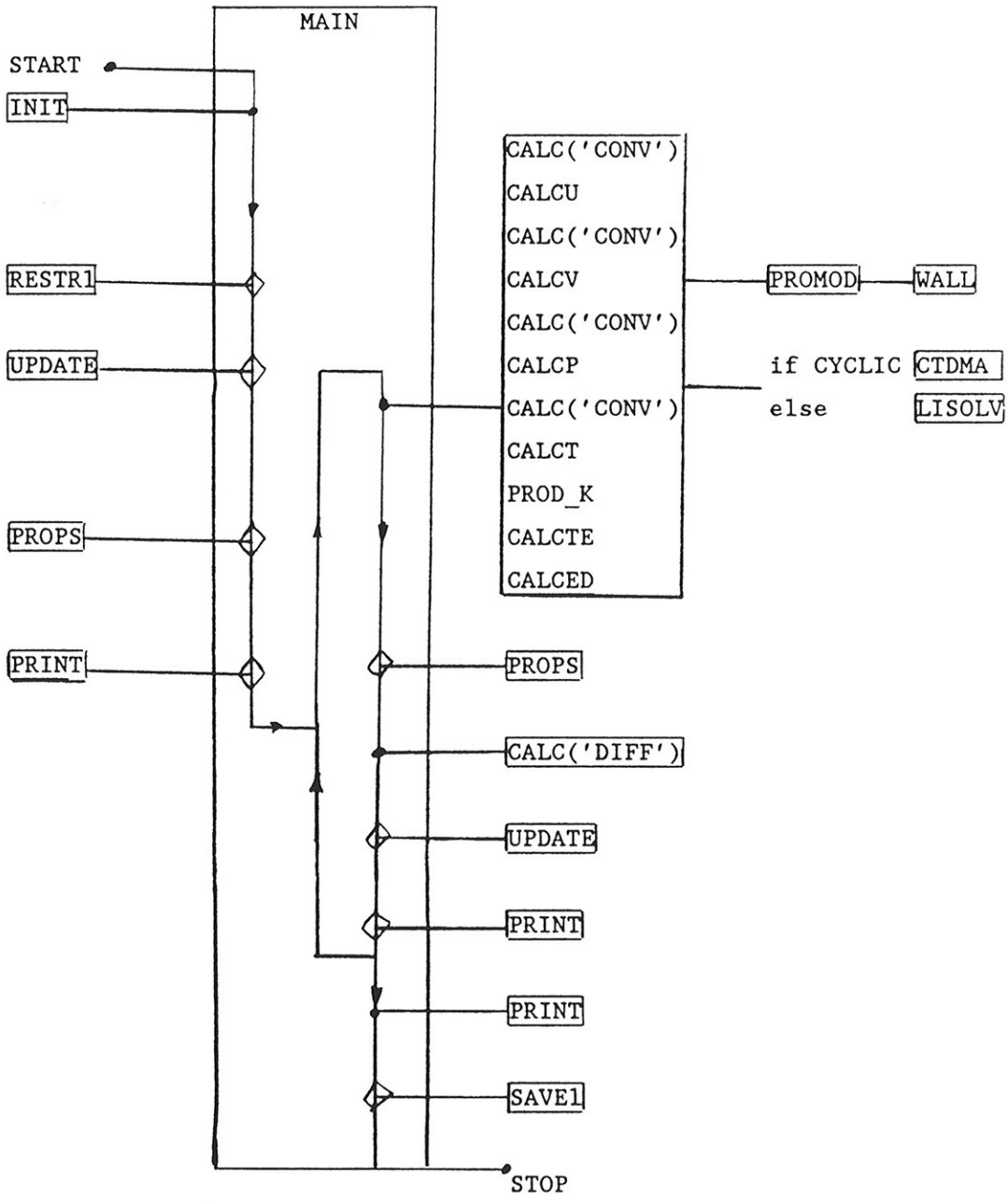
When nx is smaller than 3 the algorithm described above has to be modified; rather than modifying the algorithm, this is not applied when nx is smaller than 3. The cyclic boundary conditions are automatically taken account for when the CTDMA-subroutine sweeps in the other direction.

In subroutine CTDMA ϕ_0 is set to ϕ_{nx}, and $\phi_{nx+1} = \phi_1$. In subroutine CALC the convection through the west boundary is set equal to that through the east boundary, i.e. $m_{w,ix=1} = m_{e,ix=nx}$; the same is done for the diffusion.

Special care must be taken by the user to ensure that the grid lines between ix=0 and ix=1 have the same direction as those between ix=nx and ix=nx+1. For some types of grids this is automatically fulfilled, for others not. Remember that the scalar nodes outside the calculation domain (e.g. ix=0 and i=nx+1) are placed so that the base vectors are orthogonal at the boundaries (see Section 3.1).

4. STRUCTURE OF FLUX2D

4.1 PROGRAM FLOW CHART



4.2 DESCRIPTION OF THE SUBROUTINES IN FLUX2D

MAIN.

The problem is set up. The grid is specified, relaxation parameters, number of sweeps, constants of the turbulence model, fluid properties, etc., are given.

INIT.

All geometric quantities are calculated and stored in 2D-arrays.

RESTR1.

Fields from a previous calculation is read from file INDATA.DAT, which are used as initial fields in the current execution.

UPDATE.

This subroutine is called (in transient executions) at the end of each time step when dependent variables are to be updated (i.e. UO=U, VO=V, etc.)

ITERA.

This subroutine is not shown in the flow chart. It contains the steady (or at each time step) iteration loop, from where all CALCϕ-subroutines, PROPS, PROD_K, UPDATE, and PRINT are called

CALC.

The convection and diffusion for the east and north faces of the scalar control volumes are calculated and stored in 2D-arrays. MODCON and MODDIF are called so that the user can modify the convection and diffusion.

CALCϕ.

The coefficients (AE, AW, etc.) are calculated, MODϕ is called, relaxation is introduced and, finally, LISOLV (or CTDMA) is called. In CALCP velocities and pressure are corrected.

PROD_K.

The produktion term, P_k (see Section 2.3), is calculated.

PROMOD.

This subroutine contains nine entries (ENTRY MODϕ, MODCON and MODDIF), where boundary conditions are set by the user.

LISOLV.

This subroutine is used if CYCLIC=.FALSE. The linearised field equations are solved using TDMA sweeping in both x_1 and x_2-direction.

CTDMA.

This subroutine is used if CYCLIC=.TRUE. The linearised field equations are solved using CTDMA sweeping in x_1 and TDMA in x_2-direction.

PROPS.

Fluid properties such as turbulent viscosity are calculated; if, for instance, the density depends on any variable (T for example), the user should introduce appropriate coding in this subroutine (or in MODPRO).

WALL.

Boundary conditions, according to conventional wall-functions [2,10] for the variables v_1, v_2, T, k and ϵ, are set in this subroutine. WALL should normally be called from PROMOD. It is also possible to prescribe zero gradient boundary conditions with this subroutine. The arguments in the CALL WALL statement are:

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

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

IXSTRT, IXEND, IYSTRT, IYEND=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 IXSTRT, IXEND, etc. are in the region.

VALUE: If VALUE.LE.-100. zero gradient boundary condition is applied (not for normal velocity component at outlets), otherwise boundary conditions according to the wall functions. For the variable T VALUE=wall temperature; if VALUE.LE.-100. adiabatic 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.

5. TEST CASES

5.1 ROTATING COUETTE FLOW

This case is easily solved using a conventional polar co-ordinate system. We wish, however, to show that the formulation presented in this paper may be used for such calculations. The outer cylinder rotates ($\omega=\omega_2$) and the inner cylinder is stationary (see Fig. 5.1). This is a one-dimensional problem for which there exists an analytical solution [11] (using $R_1=\omega_2=1$, $R_2=2$; see Fig. 5.1)

$$v_r \equiv 0; \quad p=\frac{8}{9}\rho[\, r^2 - 1/r^2 - 4\ln(r)]; \qquad (5.1)$$

$$v_\varphi = \frac{4}{3}(r-1/r)$$

where r and φ denote the radial and the circumferential co-ordinate, respectively. The analytical solution is used when prescribing boundary conditions at the boundaries.

The flow is, despite being one-dimensional ($\partial/\partial\varphi=0$), solved two-dimensionally, in order to check how well the calculated flow field satisfies the condition $\partial/\partial\varphi=0$.

The calculated v_φ-velocity and the pressure are compared with the analytical solution in Fig. 5.2, for three different angles, namely $\varphi=10^o$, 45^o and 80^o. The agreement is very good.

A 28 x 28-node polar grid was used and the required CPU time for obtaining a converged solution was three minutes on a VAX 785 machine with an FPA.

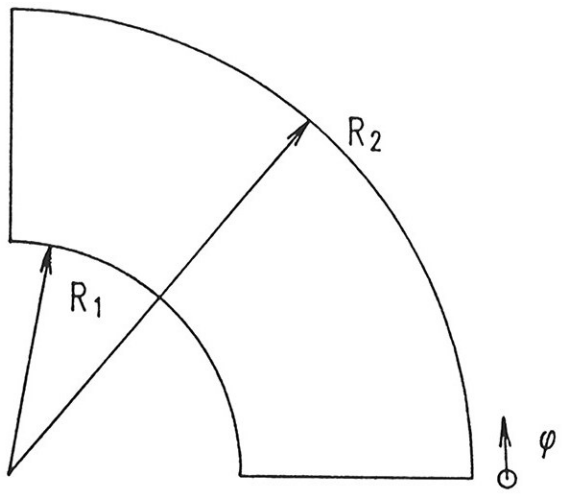


Figure 5.1. Configuration.

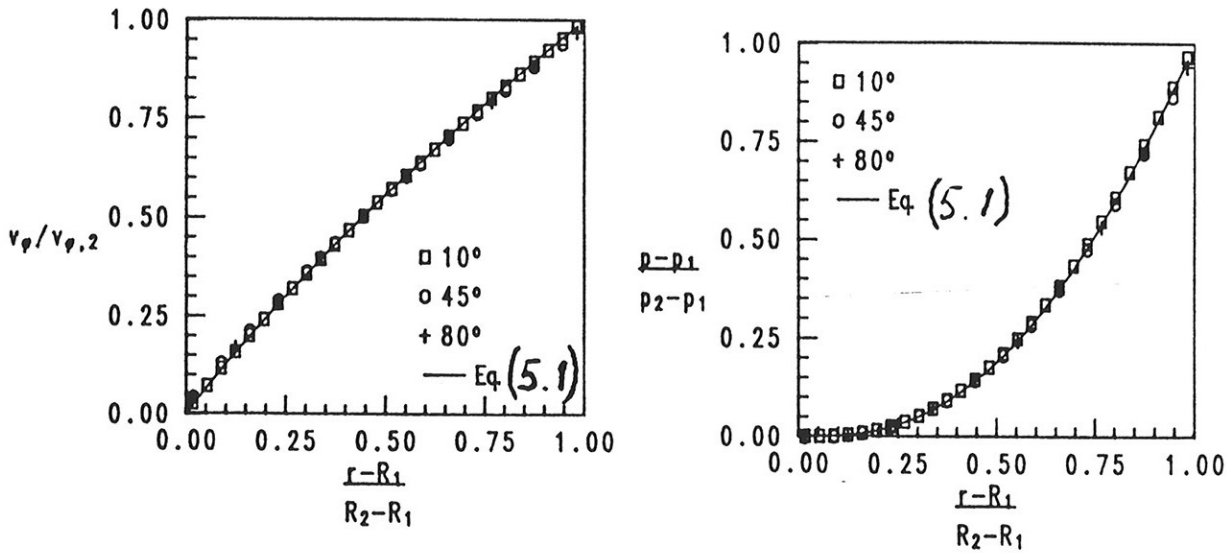


Figure 5.2. Comparison of calculated results and the analytical solutions [Eq. (5.1)]. a) v_φ-velocity scaled with the velocity of the outer cylinder. b) Pressure scaled with the pressure difference between the outer and the inner cylinder.

5.2 FLOW THROUGH TUBES WITH CONSTRICTIONS

The laminar flow in constricted tubes has an important application in arterial stenoses in the human body. Young and Tsai [12] have carried out experimental investigation on this type of flow, where they determined the positions of the separation and re-attachment points after the constriction, see Fig. 5.3.

The configuration is axi-symmetric and is thus two-dimensional. Since the code is written for two-dimensional plane configurations, special care must be taken to account for the divergence of the grid lines in the r-φ plane. This procedure is explained in Appendix B.

The outer radial boundary (see Fig. 5.3) is given by

$$\frac{R}{R_o} = 1 - \frac{\delta}{2R_o}\left[1 + \cos\left(\frac{\pi x}{x_o}\right)\right] \quad \text{for } |x| \leq x_o$$

$$\frac{R}{R_o} = 1 \quad \text{for } |x| > x_o$$

where $x_o = 4R_o$ and $\delta = 2R_o/3$.

At the inlet, the cartesian velocity components are set as

$$u = 2\bar{u}\left(1 - \frac{r^2}{R_o^2}\right)$$

$$v = 0$$

where $\bar{u}$ is the mean velocity. At the wall v_1 and v_2 are set to zero. Zero streamwise gradient is imposed for both v_1 and v_2 at the outlet.

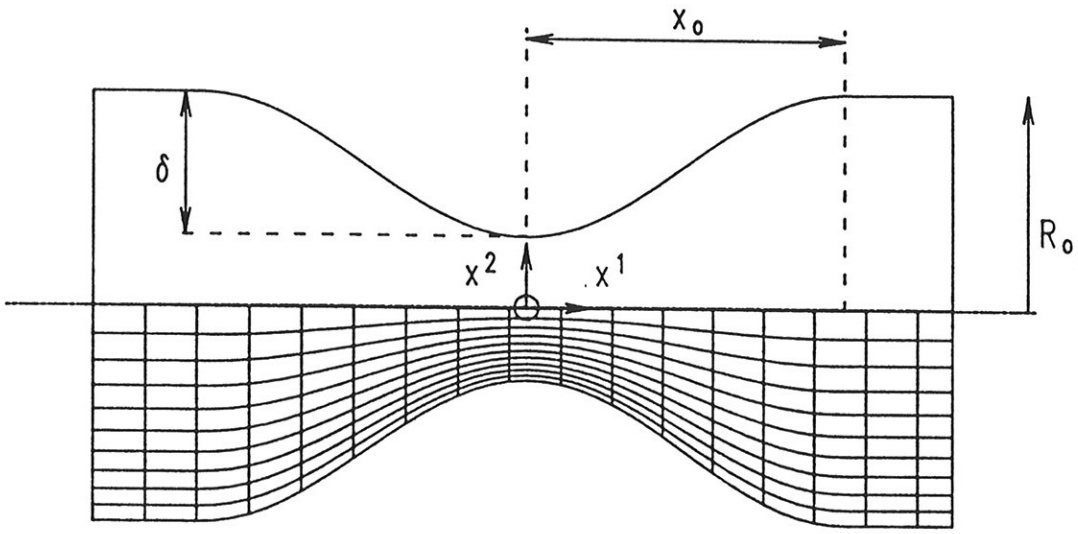


Figure 5.3. Configuration (not to scale) and the grid (schematically drawn).

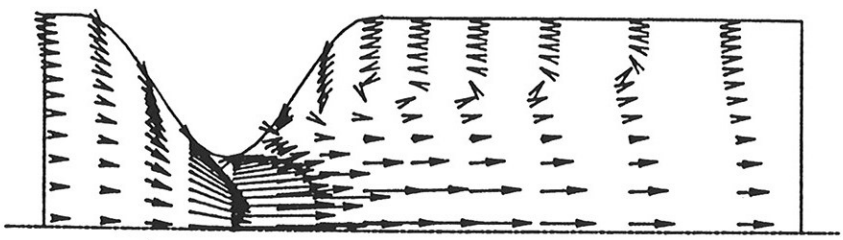


Figure 5.4. Calculated velocity vectors. Re=100. The configuration is drawn not to scale.

The flow was calculated for two different Reynolds numbers. The calculated velocity vectors for Re=100 are shown in Fig. 5.4. The results are compared with experimental data in Table 1; the calculated results by Karki [8], Deshpande et al. [13] and Rastogi [14] are also shown.

Table 1. Point of Separation and Re-attachment.

Source of data	Re=50 x_S/x_o, x_R/x_o	Re=100 x_S/x_o, x_R/x_o
Karki [8] (calculations)	0.35, 2.1	0.35, 4.0
Deshpande et al. [13] (calculations)	0.35, 2.1	0.35, 4.2
Rastogi [14] (calculations)	-	0.35, 3.8
Present calculations	0.35, 2.07	0.31, 3.75
Experiments [12]	0.37, 2.2	0.37, 4.0

The predicted x-coordinate of the re-attachment point is, as can be seen from Table 1, somewhat too low. The re-attachment point was, however, difficult to visualize, since the thickness of the separated region became exceedingly thin near the re-attachment point [12]. A 72 x 38-node grid (see Fig. 5.3) was used with denser spacing in the

y-direction near the wall. The location of the exit was varied between $6.5x_o$ to $14x_o$ for Re=100, and between $5x_o$ and $10x_o$ for Re=50; this variation did not, however, influence the calculated results.

The CPU-time required for a converged solution was approximately a half an hour on a VAX-785 machine with an FPA.

5.3 FLOWS BETWEEN NON-PARALLEL PLANE WALLS

The Jeffery-Hamel flow is another laminar case where the exact solution is known [15].

In our test we have studied the diverging channel flow, Fig. 5.5, where the total angle, 2α, between the walls is 60 degrees. We have studied the effect of grid size and the effect of varying Reynolds number.

The flow is purely radial, so the radial velocity can be written as

$$v_r = \frac{\nu F(\theta)}{r} ,$$

where $F(\theta)$ is the dimensionless velocity profile. It can be calculated from:

$$F(\theta) = Re_0 - 6(m^2 k^2 sn^2(m\theta,k))$$

where

$Re_0 = F(0) = v_{r0} r/\nu$ (v_{r0} is the velocity along the axial stream line, $\theta = 0$);

the operator, sn, stands for the Jacobian elliptic function,

$m=[(1+Re_0/2)/(1+k^2)]^{1/2}$;

the elliptic-modulus, k, can be determined from the trancendental equation

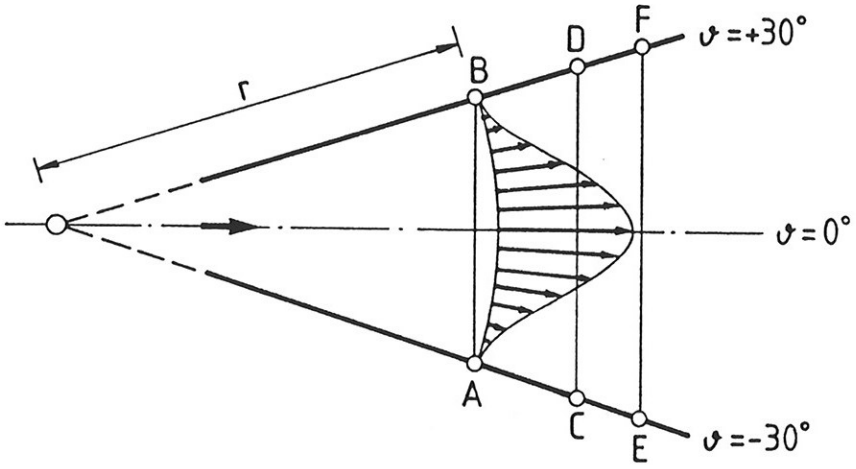
$$\frac{1+k^2}{3k^2[1+2/Re_0]} = sn^2(m\alpha,k)$$

The trancendental equation was solved by a regula-falsi method, evaluating the Jacobian elliptic function by employing the subroutine of Umstaetter [16].

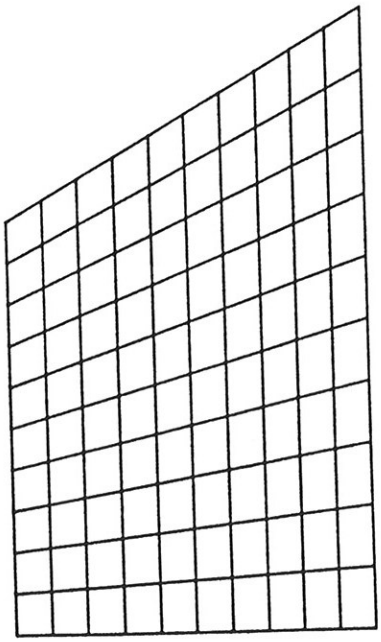
The velocities at the inlet (A-B in Fig. 5.5a) and the outlet (E-F in Fig. 5.5a) were prescribed according to the exact solution. At the wall v_1 and v_2 were set to zero.

The flow field was calculated for three Reynolds numbers, 20 ,37.6 and 150. These numbers were chosen because of the difference in the character of the flow. For each Reynolds number two different meshes were used, 10 x 10 and 20 x 20. At Re_0 = 20 the velocities are always positive (zero at the wall). At Re_0=37.6 the derivative at the wall is close to zero, and at Re_0=150 there is an inflow in the channel. The corresponding values of the elliptic-modulus k are 0.710646, 0.683006 and 0.903324 respectively. The same test was reported by Parameswaran [17] but for other Reynolds numbers and angles.

The results are shown in Figs. 5.6-5.8. The conclusion is that a 10 x 10 grid is enough to resolve the flow field for these cases. None of these cases required more than seven CPU-minutes, on a VAX 750 with an FPA, for a converged solution.



a)



b)

Figure 5.5. a) Configuration. The lines A-B, C-D and E-D intersect the symmetry line at r = 1, r = 1.3 and r = 1.5, respectively.
b) Grid 10 x 10 (one side of the symmetry line).

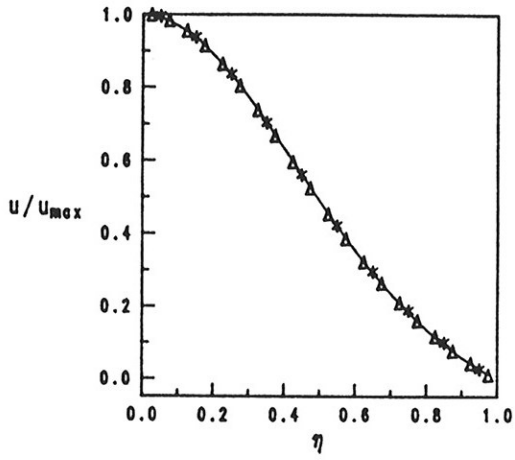


Figure 5.6. Velocity distribution for Re = 20 . η is the non-dimensional coordinate, perpendicular from the symmetry line to point D in Fig. 5.5a. Markers: * 10 x 10 , Δ 20 x 20 .

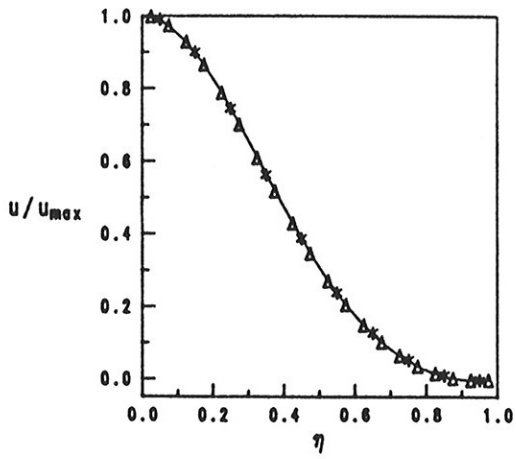


Figure 5.7. Velocity distribution for Re = 37.6 . η is the non-dimensional coordinate, perpendicular from the symmetry line to point D in Fig. 5.5a. Markers: * 10 x 10 , Δ 20 x 20 .

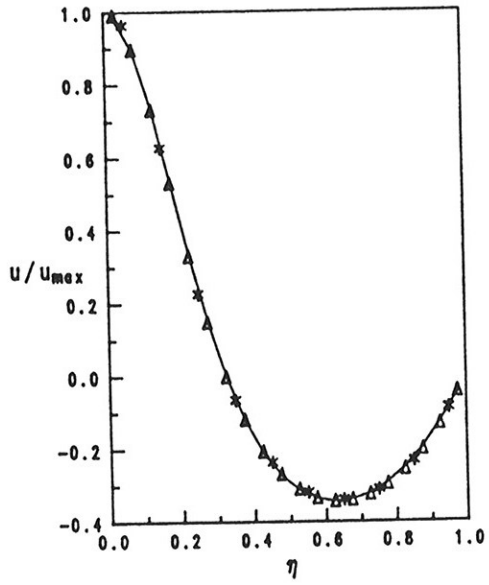


Figure 5.8. Velocity distribution for Re = 150 . η is the non-dimensional coordinate, perpendicular from the symmetry line to point D in Fig. 5.5a. Markers: * 10 x 10 , Δ 20 x 20 .

6. REFERENCES

1. L. Davidson and P. Hedberg, "Mathematical Derivation of a Finite-Volume Formulation for Laminar Flow in Complex Geometries", Int. J. Numer. Methods Fluids, Vol. 9, pp. 531-540 (1989).

2. L. Davidson, P. Hedberg and E. Olsson, "A Finite-Volume Computer Program for Turbulent Flow in Complex Geometries", Proc. 7th International Conference on Finite Element Methods in Fluid Problems, Huntsville, Alabama, pp. 390-396 (1989).

3. P.F. Galpin, G.D. Raithby and J.P. Van Doormaal, "Discussion of Upstream-weighted Advection Approximat,ons for Curved Grids", Num. Heat Transfer, Technical Note, Vol. 9, pp. 241-246 (1986)

4. L. Davidson, and P. Hedberg, "A General Computer Program for Transient, Three-Dimensional, Turbulent, Recirculating Flows - Supplement 1", Rept. 87/9, Dept. of Applied Thermodynamics and Fluid Mechanics, Chalmers Univ. of Tech., Göteborg (1987).

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

6. W. Flugge, Tensor Analysis and Continuum Mechanics, Springer-Verlag, Berlin (1972).

7. S.V. Patankar, Numerical Heat Transfer and Fluid Flow, McGraw-Hill, Washington (1980).

8. K.C. Karki, "A Calculation Procedure for Viscous Flows at All Speeds in Complex Geometries", PhD thesis, University of Minnesota (1986).

9. J.P. Van Doormaal and G.D. Raithby, "Enhancements of the SIMPLE Method for Predicting Incompressible Fluid Flows", Num. Heat Transfer, Vol. 7, pp. 147-163 (1984).

10. L. Davidson, and P. Hedberg, "A General Computer Program for Transient, Three-Dimensional, Turbulent, Recirculating Flows", Rept. 86/13, Dept. of Applied Thermodynamics and Fluid Mechanics, Chalmers Univ. of Tech., Göteborg (1986).

11. H. Schlichting, Boundary-Layer Theory, 6th ed., McGraw-Hill (1968)

12. D.F. Young and F.Y. Tsai, "Flow Characteristics in Models of Arterial Stenoses - I Steady Flow", J. Biomechanics, Vol. 6, pp. 395-410 (1973).

13. M.D. Deshpande, D.P. Giddens, and R.F. Mabon, "Steady Laminar Flow Through Modelled Vascular Stenoses", J. Biomechanics, Vol. 9, pp. 165-174 (1976)

14. A.K. Rastogi, "Hydrodynamics in Tubes Perturbed by Curvilinear Obstructions", J. Fluid Engng., Vol. 106, pp. 262-269 (1984)

15. K. Millsaps and K. Pohlhausen, "Thermal Distribution in Jeffrey-Hamel Flows between Parallel Plane Walls", J. of Aeronautical Science, Vol. 20, pp. 187-196 (1953)

16. H.H. Umstaetter, "Jacobian Elliptic Functions sn, cn and dn", Cern Computer Centre Program Library, C 318 (1980).

17. S. Parameswaran, "Finite Volume Equations for Fluid Flow Based on Non-Orthogonal Velocity Projections", PhD thesis, University of London (1986).

APPENDIX A: BOUNDARY CONDITIONS

A.1 WALL BOUNDARY CONDITIONS

If the mesh is arbitrarily chosen at the boundary, the expressions for the variables become very complicated. These expressions can be greatly simplified if the direction of the base vectors are orthogonal at the boundary. One direction is along the boundary, i.e. along the control-volume surface. The other direction is the direction between the two scalar points located on each side of the boundary (see Fig. 3.2). If the point outside the boundary is the mirror image of the point inside the boundary, then we have a right angle between these base vectors. This is taken care of in subroutine INIT where the the coordinates of the scalar points are calculated from the cell corner coordinates. Note that this does not mean that the mesh has to be orthogonal at the boundary.

A.1.1 WALL BOUNDARY CONDITIONS FOR THE MOMENTUM EQUATIONS

The boundary condition is introduced by giving an estimate of the term,

$$\int_A g^{jk}(\rho v_i v_j - \mu_{eff}\frac{\partial v_i}{\partial x^j})\, n_k dA \qquad (A.1)$$

which appears in Eq. (2.2). Here A is the surface of the control volume which coincides with the boundary. If we have no convection through the wall the, (A.1) is reduced to,

$$-\int_A g^{jk}\, \mu_{eff}\, \frac{\partial v_i}{\partial x^j}\, n_k dA\ . \qquad (A.2)$$

A.1.1.1 LAMINAR CASE

We now take an example of what this term will amount to at a "south" boundary when the flow is laminar. Equation (A.2) can be interpreted as

$$-A\,\mu\,\vec{n}\cdot\vec{\nabla}\,v_1 = -A\,\mu\,(n^1\,\vec{g}_1 + n^2\,\vec{g}_2)\cdot\left(\frac{\partial}{\partial x^1}\,\vec{g}^1 + \frac{\partial}{\partial x^2}\,\vec{g}^2\right) v_1 =$$

$\{n^1 = 0\ ,\ n^2 = 1$ in this case$\}$

$$= -A\,\mu\,\left(\vec{g}_2\cdot\vec{g}^1\,\frac{\partial}{\partial x^1} + \vec{g}_2\cdot\vec{g}^2\,\frac{\partial}{\partial x^2}\right) v_1$$

The final result, after observing that

$$\vec{g}_2\cdot\vec{g}^1 = 0;\ \vec{g}_2\cdot\vec{g}^2 = g^{22} = 1$$

is

$$-A\,\mu\,\frac{\partial v_1}{\partial x^2}\ . \qquad \text{(A.3)}$$

The easiest way to calculate this term in the discretised equation is to project the velocity of the wall in the v_1 direction. Thus we avoid to calculate the Christoffel symbol. The derivative is then the difference between v_1 and the projected velocity of the wall divided by the perpendicular distance from the wall to the v_1 location. If the wall is not moving the derivative is simply v_1 divided by the distance to the wall. For a symmetry condition, which is the same as a slip condition, the term drops out all together i.e. AS(IX,1) should be set to zero.

A.1.1.2 TURBULENT CASE USING LAW OF THE WALL FOR THE VELOCITIES

The term $-A\ \mu\ \partial v_1/\partial x^2$ is the product of the boundary area and the shear stress on the surface. In a turbulent case where we do not resolve the flow field down to the viscous sublayer, but we introduce a turbulent shear stress. Assuming that the log-law applies we can write the shear stress as,

$$\tau = \rho U_*^2$$

where ρ is the density and U_* is the friction velocity which can be calculated from the expression:

$$U_* = (C_\mu C_D)^{1/4} k^{1/2}$$

where k is the turbulent kinetic energy, and C_μ and C_D are constants.

This is programmed in subroutine WALL and can be activated by inserting a call to this subroutine in subroutine PROMOD under the entries MODU or MODV.

A.1.2 BOUNDARY CONDITIONS FOR SCALARS

The boundary condition is introduced by specifying the value of φ, or by giving an estimate of the term,

$$\int_A g^{jk}(\rho v_j \varphi - \Gamma_{eff} \frac{\partial \varphi}{\partial x^j})\ n_k dA$$

This term can be estimated as desribed in A.1.1.1 if v_1 is replaced by φ.

A prescribed value of the variable can be set by using source terms (SP(IX,IY)=-10^{10} and SU(IX,IY)=10^{10}*value), or simply by setting the variable itself if it is located outside the computational domain (IX,IY=0 or IX=NX+1,IY=NY+1).

A.1.2.1 TURBULENT CASE USING THE LAW OF THE WALL FOR k & ϵ

Assuming that the law of the wall applies, k and ϵ can be calculated from the expressions:

$$k = U_*^2/(C_\mu C_D)^{1/2}$$

$$\epsilon = (C_\mu C_D)^{3/4} k^{3/2}/(\kappa n)$$

where κ is the von Karman constant and n is the distans to the wall from the scalar point where k and ϵ is stored. U_* is determined in a iterative manner from the expression:

$$U_* = \kappa|v|/\ln(\rho E U_* n/\mu)$$

where E is a constant which is related to the wall roughness.

This is programmed in subroutine WALL, where k and ϵ are specified by using source terms, and can be activated by inserting a call to this subroutine in subroutine PROMOD under the entries MODK and MODED respectively.

A.2 INFLOW BOUNDARY CONDITIONS

The inlet boundary condition is the easiest to specify. This is done by giving a value to the variables in subroutine PROMOD. This is true for variables that are stored outside the computational domain such as U(0,IY), V(0,IY), TE(0,IY), U(NX,IY), ED(NX+1,IY) and ED(IX,0) etc. If an inlet is to be specified in the middle of the computational domain, this has to be done by introducing source terms (SP(IX,IY)=-10^{10} and SU(IX,IY)=10^{10}*value).

It might be worth noting that because the scalar point outside the boundary is the mirror image of the scalar point inside the boundary, the velocity components are parallel and orthogonal to the boundary respectively. Therefore cartesian components can be used at the boundaries.

A.3 OUTFLOW BOUNDARY CONDITIONS

The values of the variables at the outlet boundary are in general not known. If they are known, the variables on and outside the boundary can be set directly in PROMOD. However, assuming that no diffusion is present at the outlet means that no values have to be specified for scalar variables. This is due to the upwind treatment of variables.

The convection has to be specified at the outlet. The boundary condition is needed for the pressure-correction equation (see section 3.6) only.

The usual boundary condition for Cartesian grids is that the gradient of the exit velocity should vanish. If the grid is orthogonal at the outlet this boundary conditions can be used. For the general case,

however, the coresponding boundary condition is that the gradient of $\vec{v} \cdot \vec{A}$ should be zero.

Here is described how the convection at the outlet (east boundary) can be specified. All the coding should be done in PROMOD under ENTRY MODCON.

1. Sum up the total specified mass inflow, FLOWIN.
2. Calculate the total convection, as described in section 3.2, of a cross section upstream the outlet, FLOWOUT.
3. The global mass imbalance is then: FLOWDIFF=FLOWIN-FLOWOUT. (The difference should be zero when global continuity is satisfied.)
4. Next step is to distribute FLOWDIFF over the cells at the outlet. This distribution can be done proportionally to the cells surface facing the outlet, FLOWDIFF(IY).
5. Calculate the convection at the outlet as: CONVE(NX,IY)= CONVE(NX-1,IY)+FLOWDIFF(IY)

When the solution has converged, global continuity is satisfied and FLOWDIFF is zero. This leads to that we have no change of convection in the outflow direction. Thus we have no acceleration of flow, which is our boundary condition.

APPENDIX B: POLAR COORDINATE SYSTEM

The program is written for two-dimensional plane configurations. The configuration in Section 5.2 is axi-symmetric; special treatment is then required due to the divergence of the grid lines in the r-φ plane. Co-ordinate directions 1 and 2 are defined in Fig. 5.3. For a straight pipe these directions are denoted by x and r, respectively.

The continuity equation in the two-dimensional plane co-ordinate system and the axi-symmetric co-ordinate system have, respectively, the forms

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \qquad \text{(B1a)}$$

$$\frac{\partial}{\partial x}(ru) + \frac{\partial}{\partial r}(rv_r) = 0 \qquad \text{(B1b)}$$

The v-momentum equation for the two co-ordinate systems can be written

$$\frac{\partial}{\partial x}(uv) + \frac{\partial}{\partial y}(vv) = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \qquad \text{(B2a)}$$

$$\frac{\partial}{\partial x}(ruv_r) + \frac{\partial}{\partial r}(rv_r v_r)$$

$$= -\frac{r}{\rho}\frac{\partial p}{\partial r} + \nu\left\{\frac{\partial}{\partial x}\left(r\frac{\partial v_r}{\partial x}\right) + \frac{\partial}{\partial r}\left(r\frac{\partial v_r}{\partial r}\right) - \frac{v_r}{r}\right\} \qquad \text{(B2b)}$$

Equations (B1b, B2b) are obtained from Eqs. (B1a, B2a) by an appropriate multiplication of the latter by the radial co-ordinate r, and by adding the diffusive source term

$$- \nu \frac{v_r}{r}$$

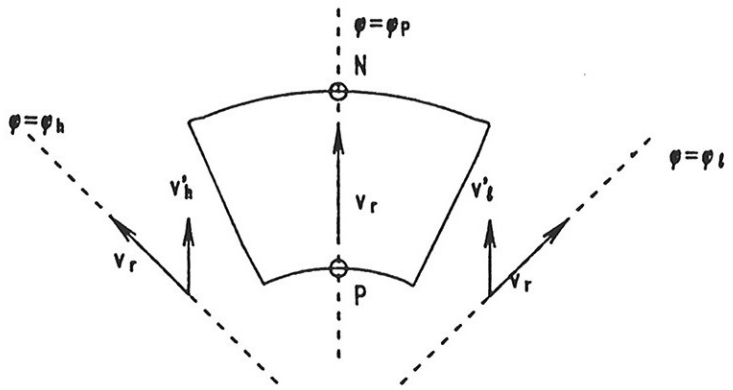


Figure B1. A control volume in the r-φ plane.

In axi-symmetric configurations the φ-lines (lines of constant φ, see Fig. B1) diverge, which gives rise to the additional diffusive source term in Eq. (B2b). In our formulation this is taken care of as is described in Section 2. The neighbouring velocity in the third co-ordinate direction, $(v_2')_\ell$, of $(v_2)_p$ $[(v_2)_p \equiv (v_2)_h \equiv (v_2)_\ell \equiv v_r]$ is taken as (see [1])

$$(v_2')_\ell = \vec{v}_\ell \cdot (\vec{g}_r)_p = (v^j \vec{g}_j)_\ell \cdot (\vec{g}_2)_p = (v^2 \vec{g}_2)_\ell \cdot (\vec{g}_2)_p = (v_r \vec{r})_\ell \cdot \vec{r}_p =$$

$$v_r \cos(\Delta\varphi) = (v_2')_h \qquad \text{(B3)}$$

where $\Delta\varphi$ denotes the angle between two adjacent φ-lines (see Fig. B1). The third expression on the RHS was obtained because

$$(\vec{g}_1)_\ell \cdot (\vec{g}_2)_p = \vec{x}_\rho \cdot \vec{r}_p = 0; \; v^3 = 0$$

The fourth expression on the RHS was given since $\vec{g}_2 = \vec{r}$, and because the contravariant component v^2 and the physical component v_r are equal in a cylindrical co-ordinate system.

The diffusion contribution which is obtained using $(v_r')_\ell$ and $(v_r')_h$ in Eq. (B3) is indeed identical to the diffusion source term in Eq. (B2b). Note that for a conventional cylindrical co-ordinate system [as in Eq. (B2b)] the diffusion contribution in φ-direction is zero because $(v_r)_p = (v_r)_\ell = (v_r)_h$; in our formulation the diffusion is non-zero since $(v_2)_p \neq (v_2')_\ell = (v_2')_h$.

In Eqs. (B1)-(B3) and the discussion above, it has been assumed that the x-lines are straight (i.e. an axi-symmetric configuration). When they curve, as in the case in Section 5.2, additional terms appear; they are derived as explained in [1] and above.

APPENDIX C: FORTRAN SYMBOLS

A(ix) coefficient of recurrence formula

ALFA(ix) coefficient of recurrence formula

AE, AN, AS
AW(ix,iy) coefficients of convective/diffusive flux through east, north, south and west wall of control volume

AP(ix,iy) sum of coefficients AE, AW, AN, AS and APO and source SP

APO(ix,iy) coefficient for old time step

AREAE(ix,iy) area of east wall of scalar control volume projected on the plane normal to $\overrightarrow{PE}$

AREAN(ix,iy) area of north wall of scalar control volume projected on the plane normal to $\overrightarrow{PN}$

B, C(ix) 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 coefficient of convective flux through east 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
CONVE(ix,iy)	convection flux through the east wall of scalar control volume
CONVN(ix,iy)	convection flux through the north wall of scalar control volume
CP	maximum of zero and net outflow (SMP) from control volume
CS, CW	coefficient of convective flux through south and west wall of control volume
CYCLIC	logical variable to control whether cyclic boundary conditions are to be applied or not
D(ix)	coefficient of recurrence formula
DEN, DENO(ix,iy)	density of fluid at current and old time step
DENSIT	constant density of fluid set in INIT
DENU, DENV	density for U, and V control volume
DFE1, DFN2, DFS2, DFW1	coefficient of orthogonal diffusive flux through east, north, south and west wall of control volume
DFE2, DFN1,	

DFS1, DFW2 coefficient of non-orthogonal diffusive flux through east, north, south and west wall of control volume

DIFFE1(ix,iy)
orthogonal diffusion flux through the east wall of scalar control volume

DIFFE2(ix,iy)
non-orthogonal diffusion flux through the east wall of scalar control volume

DIFFN1(ix,iy)
non-orthogonal diffusion flux through the north wall of scalar control volume

DIFFN2(ix,iy)
orthogonal diffusion flux through the north wall of scalar control volume

DPE(ix,iy) distance between scalar node (ix,iy) and (ix+1,iy)

DPN(ix,iy) distance between scalar node (ix,iy) and (ix,iy+1)

DU(ix,iy) coefficient of velocity-correction term for U velocity

DV(ix,iy) coefficient of velocity-correction term for V velocity

ED, EDO(ix,iy)
energy dissipation rate, ϵ, at current and old time step

EDCEN, EDCES,
EDCWN, EDCWS ϵ at north-east, south-east, north-west and south-west corner of its control volume

ELOG — constant used in the log-law of the wall

FU, FUS, FUN,
FUW, FUE,
FV, FVS, FVN,
FVW, FVE,
FUEN, FUES,
FUWN, FUWS
FVEN, FVES,
FVWN, FVWS
— geometrical arrays (see Chapter 3)

FITER — first iteration

G11, G12,
G21, G22 — the covariant components of the metric tensor (g_{11}, g_{12}, g_{21} and g_{22})

G1X, G1Y — the cartesian components of the covariant base vector, $\vec{g}_1$

G2X, G2Y — the cartesian components of the covariant base vector, $\vec{g}_2$

G2EX, G2EY — the cartesian components of the covariant base vector, $\vec{g}_2$ at the east face of a scalar control volume

G1NX, G1NY — the cartesian components of the covariant base vector, $\vec{g}_1$ at the north face of a scalar control volume

GAM111, GAM122,
GAM211, GAM222(ix,iy)
— the components of the Christoffel symbol $\{{}^{1}_{1\,1}\}$, $\{{}^{1}_{2\,2}\}$, $\{{}^{2}_{1\,1}\}$, $\{{}^{2}_{2\,2}\}$

GAME, GAMN,
GAMS, GAMW coefficients of effective diffusion for scalar variables at east, north, south and west wall

GAMH(ix,iy) coefficient of effective diffusion for temperature

GCON11, GCON12,GCON21,
GCON22(ix,iy)
the contravariant components of the metric tensor (g^{11}, g^{12}, g^{21}, g^{22})

GEN(ix,iy) generation of turbulence by shear from mean flow

GREAT a very large value (i.e. 10^{20})

GU, GUS, GUN,
GUW, GUE,
GV, GVS, GVN,
GVW, GVE,
GUEN, GUES,
GUWN, GUWS
GVEN, GVES,
GVWN, GVWS
geometrical arrays (see Chapter 3)

HEDD heading 'Energy Dissipation'

HEDK heading 'Turbulent Energy'

HEDM heading 'Viscosity'

HEDP heading 'Pressure'

HEDR heading 'Density'

HEDT heading 'Temperature'

HEDU heading 'U-Velocity'

HEDV heading 'V-Velocity'

IXMON ix-index of monitoring location

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

INDMON monitoring output each INDMON iteration

INFIL character variable (default: INDATA) which contains the name of the restart file

INDPRI intermediate output of variable fields each INDPRI iteration

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

IXPREF ix-index of location where pressure is fixed if IXPREF>0; if IXPREF<0 the pressure is not fixed

INPRO logical variable for updating of fluid properties

IXT ix-index of maximum dimension of dependent variable

ITSTEP time step index

IYMON	iy-index of monitoring location
IYPREF	iy-index of location where pressure is fixed if IYPREF>0; if IYPREF<0 the pressure is not fixed
IYT	iy-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
NUPR1	logical unit to which the OUTDAT.DAT-file is connected
NX	maximum value of ix-index for the calculation domain
NXM1	=NX-1
NITER	number of iterations completed
NY	maximum value of iy-index for the calculation domain
NYM1	=NY-1
NSWPD,NSWPK NSWPP,NSWPT NSWPU,NSWPV	number of application of line iteration for ϵ, k, P', T, U, V-equation
P(ix,iy)	pressure
PFUN	constant of P-function for heat transfer at walls
PHI(ix,iy)	general representation for all dependent variables

POROE(ix,iy) porosity for the east wall of scalar control volume; the value 0 means that the wall is impermeable

PORON(ix,iy) porosity for the north wall of scalar control volume; the value 0 means that the wall is impermeable

POROV(ix,iy) porosity for the volume of scalar control volume; the value 0 means that the volume is zero

PP(ix,iy) pressure-correction, p"

PPCEN, PPCES,
PPCWN, PPCWS p" at north-east, south-east, north-west and south-west corner of its control volume

PRANDT,PRED
PRTE turbulent Prandtl number for temperature, turbulent dissipation and turbulent kinetic energy

REREFU, REREFV,
REREFM the residuals for U, V and PP are normalised with REREFU, REREFV and REREFM, respectively

RESOR residual source for individual control volume

RESORE,RESORK
RESORM,RESORT
RESORU,RESORV
sum of residual sources within calculation domain for ϵ, k, P', T, U, V-equation

RESTRT logical variable which controls whether the initial field are to be read from the file INDATA or not

SAVEM — logical variable which controls whether the fields are to be saved on the file UTDATA or not

SMALL — a very small value (i.e. 10^{20})

SORCE — maximum of RESORM, RESORU and RESORV

SORMAX — maximum acceptable value of SORCE for converged solution

SP,SU(ix,iy) — coefficient b and C of linearized source treatment

STEADY — logical variable which controls whether the calculation is a steady one or not

T,
TO(ix,iy) — temperature at current and old time step

TCEN, TCES,
TCWN, TCWS — T at north-east, south-east, north-west and south-west corner of its control volume

TE,
TEO(ix,iy) — turbulent kinetic energy at current and old time step

TECEN, TECES,
TECWN, TECWS — k at north-east, south-east, north-west and south-west corner of its control volume

TMULT — coefficient of wall shear expression

U,
UO(ix,iy) — v_1-velocity in the x-direction at current and old time step

UCON(ix,iy) the contravariant component, v^1, of the velocity vector in the x^1-direction

UCRT cartesian velocity component in the x-direction

UE, UN, US, UW east, north, south and west v_1'-velocity

UCEN, UCES, UCWN, UCWS v_1'-velocity at north-east, south-east, north-west and south-west corner of its control volume

UPLUS $u_*/U_{parallel}$

URFE,URFK URFP,URFT URFU,URFV under-relaxation factor for ϵ, k, P', T, U, V-equation

UTFIL character variable (default: UTDATA) which contains the name of the file where the fields are stored (when SAVEM=.TRUE.)

V, VO(ix,iy) v_2-velocity in the y-direction at current and old time step

VCON(ix,iy) the contravariant component, v^2, of the velocity vector in the x^2-direction

VCRT cartesian velocity component in the y-direction

VE, VN, VS, VW east, north, south and west v_2'-velocity

VCEN, VCES,
VCWN, VCWS v_2'-velocity at north-east, south-east, north-west and south-west corner of its control volume

VELPAR velocity parallel to the wall

VIS(ix,iy) effective viscosity ($\mu+\mu_t$)

VISCOS laminar viscosity (μ)

VISOLD value of effective viscosity before under-relaxation

VOL(ix,iy) volume of scalar control volume

XCRNR(ix,iy) x co-ordinate of north-east corner of scalar control volume

XPOINT(ix,iy)
x co-ordinate of scalar control grid node

YCRNR(ix,iy) y co-ordinate of north-east corner of scalar control volume

YPOINT(ix,iy)
y co-ordinate of scalar control grid node