Publikation Nr 87/9 ISSN 02809265

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för
Tillämpad termodynamik och strömningslära

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Applied Thermodynamics
and Fluid Mechanics

A GENERAL COMPUTER PROGRAM FOR TRANSIENT, THREE-DIMENSIONAL, TURBULENT, RECIRCULATING FLOWS - SUPPLEMENT 1

by

Lars Davidson and Peter Hedberg

Göteborg December 1987

CONTENTS

1. INTRODUCTION

This report describes the extensions of the computer program by Davidson and Hedberg [1]; in order to get a full description of the program the reader/user must have the original manual [1]. The program in [1] is applicable to three-dimensional cartesian geometries only. The three main extensions that have been included in the current version of the program are:

i) ability to handle three-dimensional cylindrical geometries as well as cartesian geometries;

ii) ability to handle cyclic boundary conditions in the circumferential direction;

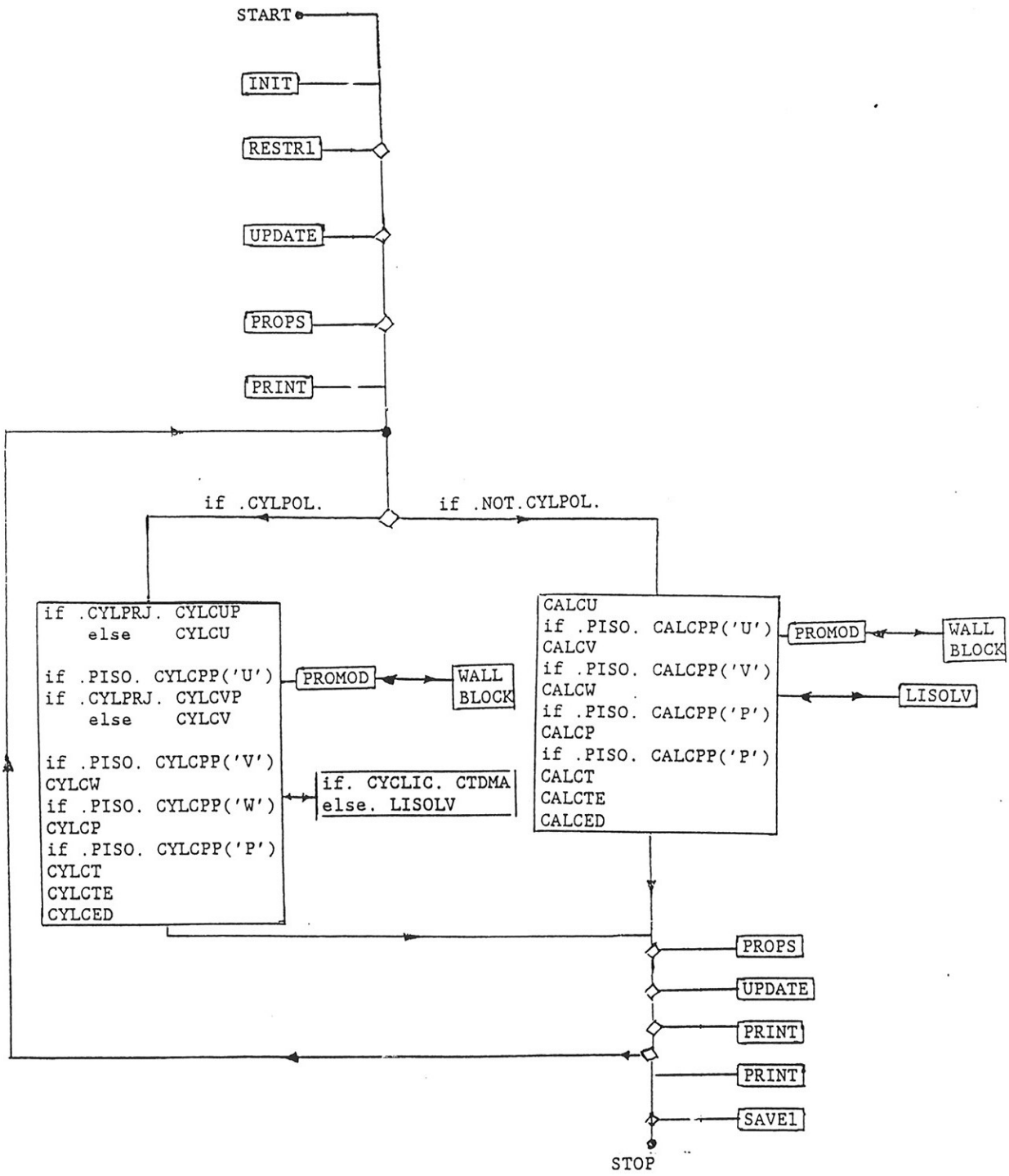
iii) the SIMPLEC [2] and the PISO algorithm [3] has been included as an option to the SIMPLE algorithm.

In [1] the coefficients for each variable ϕ are calculated in CALCϕ; these subroutines have not been changed. For the cylindrical option new subroutines (CYLCϕ) have been included where the coefficients are calculated. The PISO algorithm is coded in CALCPP (CYLCPP).

When gridlines curve and/or diverge, curvature terms appear in the momentum equations. These can be eliminated by projecting the velocity vectors, $\vec{v}_{nb}$ (nb=neighbour), in the direction of the velocity being solved for ($v_{\varphi p}$, or v_{rp}); as a consequence the curvature terms vanish. Using this approach reduces the numerical errors associated with upwind differencing in connection with polar grids (or, generally, curved grids); these errors becomes especially severe near the origo of a polar grid when the flow is across the grid. This has been recognised by Galphin et al. [4].

This procedure has been coded in CYLCUP and CYLCVP, which are called when CYLPRJ=.TRUE.

1.1 Program Flow Chart.



2. EQUATIONS.

2.1 Continuity Equation.

The continuity equation has in cylindrical co-ordinates the form [5]:

$$\frac{\partial \rho}{\partial \tau} + \frac{1}{r}\frac{\partial (r\rho v_r)}{\partial r} + \frac{1}{r}\frac{\partial (\rho v_\varphi)}{\partial \varphi} + \frac{\partial (\rho v_z)}{\partial z} = 0$$

The continuity equation has thus the same form in cylindrical as in cartesian co-ordinates (except that the radius r appears in the former case); this means that the pressure correction equation (P'-equation) also has the same form. Note that the radius r is inside the operator $\partial/\partial r$; this is taken account for in the program through multiplication by the appropriate face areas.

2.2 v_r-Equation.

The v_r-equation (velocity component in the radial direction) has in cylindrical co-ordinates the form [5]:

$$\frac{\partial}{\partial \tau}(\rho v_r) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r^2) + \frac{1}{r}\frac{\partial}{\partial \varphi}(\rho v_\varphi v_r) + \frac{\partial}{\partial z}(\rho v_z v_r) - \underline{\rho v_\varphi^2 / r}$$

$$= -\frac{\partial p}{\partial r} + \frac{1}{r}\frac{\partial}{\partial r}\left(\mu_{eff}\, r \frac{\partial v_r}{\partial r}\right) + \frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(\mu_{eff}\frac{\partial v_r}{\partial \varphi}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial v_r}{\partial z}\right)$$

$$+ \underline{\frac{1}{r}\frac{\partial}{\partial \varphi}\left(\mu_{eff}\left[\frac{\partial v_\varphi}{\partial r} - \frac{v_\varphi}{r}\right]\right) + \frac{\partial}{\partial r}\left(\mu_{eff}\frac{\partial v_r}{\partial r}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial v_z}{\partial r}\right)}$$

$$+ \underline{\frac{2\mu_{eff}}{r^2}\left[\frac{r}{2}\frac{\partial v_r}{\partial r} - \frac{\partial v_\varphi}{\partial \varphi} - v_r\right]}$$

The underlined terms are included in the constant source term S_C. Note that the viscous compressible term has been neglected, i.e. it has been assumed that $\nabla \cdot \vec{v} = 0$ in the diffusion terms.

2.3 v_φ-Equation.

The v_φ-equation (velocity component in the circumferential direction) has in cylindrical co-ordinates the form [5]:

$$\frac{\partial}{\partial \tau}(\rho v_\varphi) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r v_\varphi) + \frac{1}{r}\frac{\partial}{\partial \varphi}(\rho v_\varphi^2) + \frac{\partial}{\partial z}(\rho v_z v_\varphi) + \underline{\rho v_\varphi v_r / r}$$

$$= \underline{-\frac{1}{r}\frac{\partial p}{\partial \varphi}} + \frac{1}{r}\frac{\partial}{\partial r}\left(\mu_{eff}\, r \frac{\partial v_\varphi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(\mu_{eff}\frac{\partial v_\varphi}{\partial \varphi}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial v_\varphi}{\partial z}\right)$$

$$\underline{+ \frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(\mu_{eff}\left[\frac{\partial v_\varphi}{\partial \varphi} + 2v_r\right]\right) + \frac{1}{r}\frac{\partial}{\partial r}\left(\mu_{eff}\left[\frac{\partial v_r}{\partial \varphi} - v_\varphi\right]\right)}$$

$$\underline{+ \frac{1}{r}\frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial v_z}{\partial \varphi}\right) + \frac{\mu_{eff}}{r^2}\left[r\frac{\partial v_\varphi}{\partial r} + \frac{\partial v_r}{\partial \varphi} - v_\varphi\right]}$$

The underlined terms are included in the constant source term S_C; the Coriolis term is linearized using both S_P and S_C. The viscous compressible term has been neglected.

2.4 v_z-Equation.

The v_z-equation (velocity component in the axial direction) has in cylindrical co-ordinates the form [5]:

$$\frac{\partial}{\partial \tau}(\rho v_z) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r v_z) + \frac{1}{r}\frac{\partial}{\partial \varphi}(\rho v_\varphi v_z) + \frac{\partial}{\partial z}(\rho v_z^2)$$

$$= \underline{-\frac{\partial p}{\partial z}} + \frac{1}{r}\frac{\partial}{\partial r}\left(\mu_{eff}\, r \frac{\partial v_z}{\partial r}\right) + \frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(\mu_{eff}\frac{\partial v_z}{\partial \varphi}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial v_z}{\partial z}\right)$$

$$+ \underline{\frac{1}{r}\frac{\partial}{\partial\varphi}\left(\mu_{eff}\frac{\partial v_\varphi}{\partial z}\right) + \frac{1}{r}\frac{\partial}{\partial r}\left(r\,\mu_{eff}\frac{\partial v_r}{\partial z}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial v_z}{\partial z}\right)}$$

The underlined terms are included in the constant source term S_C. Again the viscous compressible term has been neglected.

2.5 Projection of the Neighbour Velocities

It is well known that upwind differencing gives rise to numerical diffusion. For polar co-ordinates (or, generally 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_φ (for example) with its node value, the direction of v_φ is not. This was recognised by Galphin et al. [4], who suggested to introduce a correction velocity, weighted with a factor ω ($0<\omega<1$); the value of ω was zero if the flow was aligned with the grid, and $\omega=1$ if the flow was across the polar grid.

In the present work this problem is, in our opinion, solved in a more general and straight forward way. The velocity vectors, $\vec{v}_{nb}$ (nb=neighbour), is projected in the direction of the velocity component at the control volume p being solved for. This means that all the neighbours, $v'_{\varphi nb}$ (prime denotes projected velocity), of $v_{\varphi p}$, have the same direction. In this way the problem of estimating a face value of v_φ having the in-correct direction is solved. The same is true for the v_r-equation.

Since all the v_φ-velocities in the immediate surrounding (i.e. its four neighbours, $v'_{\varphi e}$, $v'_{\varphi w}$, $v'_{\varphi s}$, $v'_{\varphi n}$) have the same direction as $v_{\varphi p}$ all curvature terms in the v_φ-equation vanish; the same is, of course, true for the v_r-equation.

The procedure of projecting velocities and in this way getting rid of the curvature terms, has previously succesfully been used by the authors [6,7].

The effective (turbulent plus laminar) shear stress in a cartesian co-ordinate system can be expressed as

$$\tau_{i,j,eff} = \mu_{eff}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right)$$

The second term is, for simplicity, neglected in the present formulation; this term is usually very small, and it is identically zero in the momentum equations (when the density and the effective viscosity are constant) due to continuity.

2.5.1 v_φ-equation

The momentum equation for v_φ, when the second term in the formula for the shear stress is neglected, has the form [5]

$$\frac{\partial}{\partial \tau}(\rho v_\varphi) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r v_\varphi) + \frac{1}{r}\frac{\partial}{\partial \varphi}(\rho v_\varphi^2) + \frac{\partial}{\partial z}(\rho v_z v_\varphi) + \underline{\rho v_\varphi v_r / r}$$

$$= -\frac{1}{r}\frac{\partial p}{\partial \varphi} + \frac{1}{r}\frac{\partial}{\partial r}\left(\mu_{eff}\, r \frac{\partial v_\varphi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial}{\partial \varphi}\left(\mu_{eff}\frac{\partial v_\varphi}{\partial \varphi}\right) + \frac{\partial}{\partial z}\left(\mu_{eff}\frac{\partial v_\varphi}{\partial z}\right)$$

$$+ \underline{\frac{2}{r^2}\mu_{eff}\left(\frac{\partial v_\varphi}{\partial \varphi} - v_\varphi\right)}$$

The underlined terms (the curvature terms) vanish when the velocities are projected.

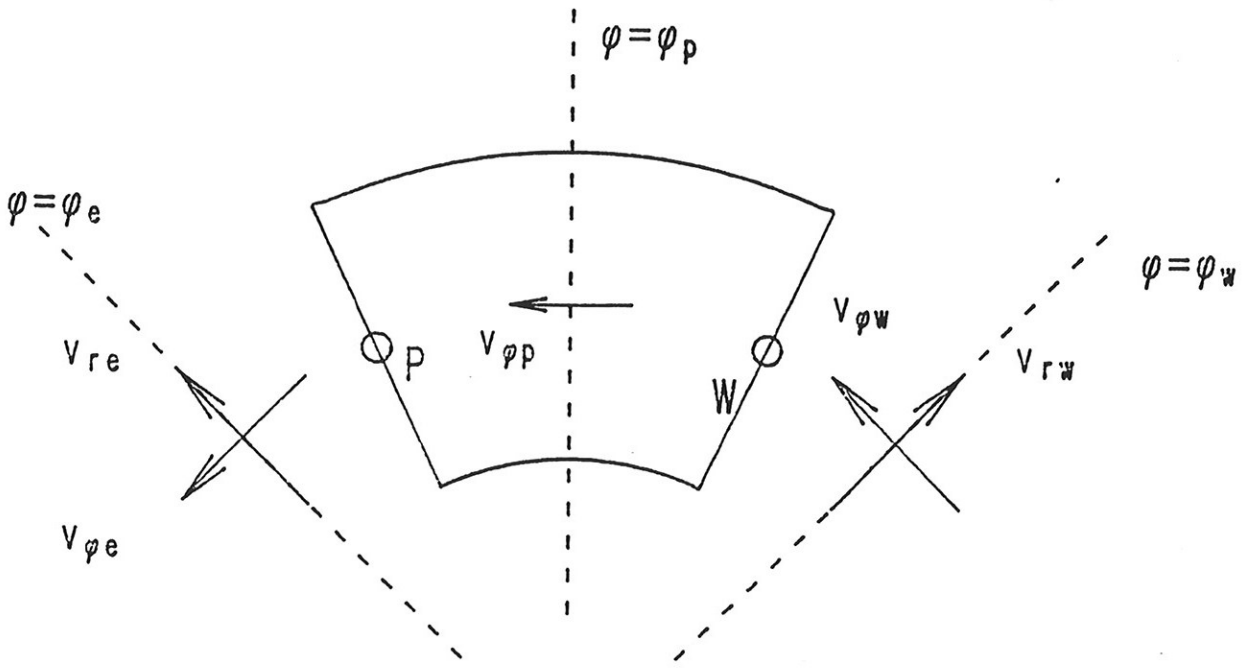


Figure 1. v_φ-control volume.

The neighbours of $v_{\varphi p}$, see Fig. 1, are projected in the direction of $v_{\varphi p}$, whose direction is $\vec{\varphi}_p \equiv \overrightarrow{WP}$. The south and north neighbours ($v_{\varphi s}$ and $v_{\varphi n}$) need not to be changed since their direction coincide with that of $v_{\varphi p}$. The west and east neighbours are calculated as [6,7]

$$v'_{\varphi e} = \vec{v}_e \cdot \vec{\varphi}_p = v_{\varphi e} \cos(\varphi_e - \varphi_p) + v_{re} \sin(\varphi_e - \varphi_p)$$

$$v'_{\varphi w} = \vec{v}_w \cdot \vec{\varphi}_p = v_{\varphi w} \cos(\varphi_p - \varphi_w) - v_{rw} \sin(\varphi_p - \varphi_w)$$

The discretised equation for $v_{\varphi p}$ can be written

$$a_p v_{\varphi p} = a_e v'_{\varphi e} + a_w v'_{\varphi w} + a_n v_{\varphi n} + a_s v_{\varphi s} + b$$

In order to be able to use TDMA, this equation is rewritten so that

$$a_p v_{\varphi p} = a_e v_{\varphi e} + a_w v_{\varphi w} + a_n v_{\varphi n} + a_s v_{\varphi s} + b + b_{curve}$$

where the last term contains the curvature effects, i.e.

$$b_{curve} = a_e(v'_{\varphi e} - v_{\varphi e}) + a_w(v'_{\varphi w} - v_{\varphi w})$$

2.5.2 v_r-equation

The momentum equation for v_r, when the second term in the formula for the shear stress is neglected, has the form [5]

$$\frac{\partial}{\partial \tau}(\rho v_r) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r^2) + \frac{1}{r}\frac{\partial}{\partial \varphi}(\rho v_\varphi v_r) + \frac{\partial}{\partial z}(\rho v_z v_r) - \underline{\rho v_\varphi^2 / r}$$

$$= -\frac{\partial p}{\partial r} + \frac{1}{r}\frac{\partial}{\partial r}(\mu_{eff}\, r \frac{\partial v_r}{\partial r}) + \frac{1}{r^2}\frac{\partial}{\partial \varphi}(\mu_{eff}\frac{\partial v_r}{\partial \varphi}) + \frac{\partial}{\partial z}(\mu_{eff}\frac{\partial v_r}{\partial z})$$

$$\underline{- \frac{2}{r^2}\mu_{eff}(\frac{\partial v_\varphi}{\partial \varphi} + v_r)}$$

The underlined terms (the curvature terms) vanish when the velocities are projected.

The neighbours of v_{rp}, see Fig. 2, are projected in the direction of v_{rp}, whose direction is $\vec{r}_p \equiv \overrightarrow{SP}$. The south and north neighbours (v_{rs} and v_{rn}) need not to be changed since their direction coincide with that of v_{rp}. The west and east neighbours are calculated as [6,7]

$$v'_{re} = \vec{v}_e \cdot \vec{r}_p = v_{re} \cos(\varphi_E - \varphi_P) - v_{\varphi e} \sin(\varphi_E - \varphi_P)$$

$$v'_{rw} = \vec{v}_w \cdot \vec{r}_p = v_{rw} \cos(\varphi_P - \varphi_W) + v_{\varphi w} \sin(\varphi_P - \varphi_W)$$

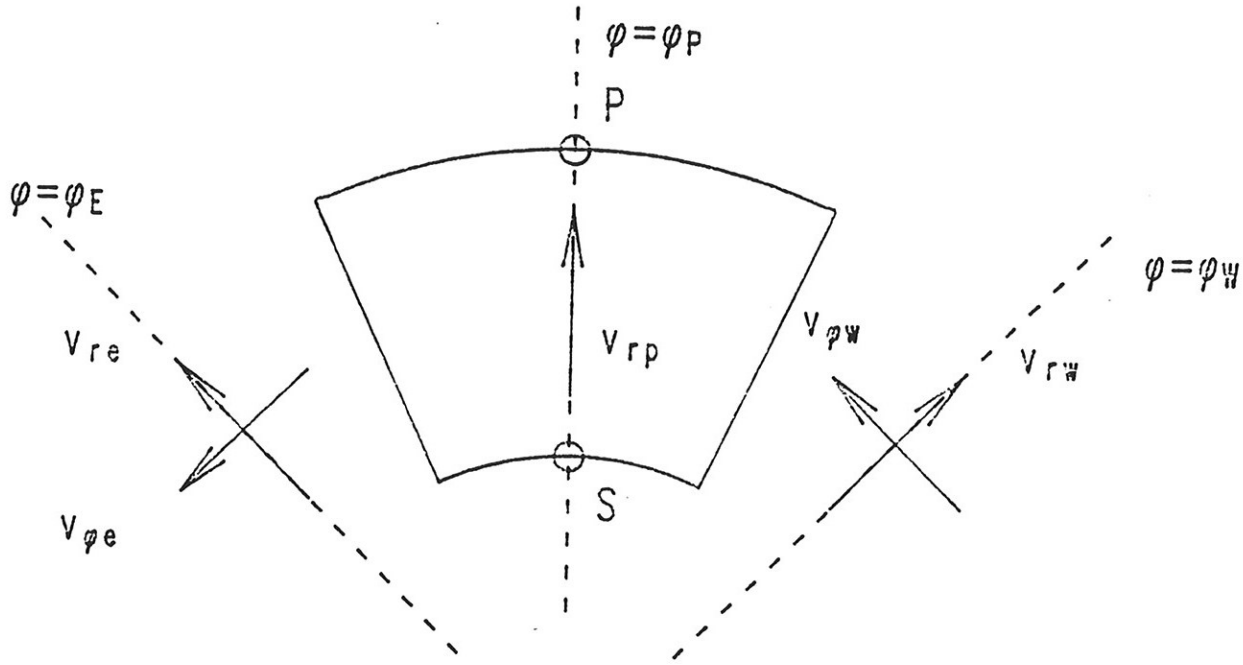


Figure 2. v_r-control volume.

The discretised equation for v_{rp} can be written

$$a_p v_{rp} = a_e v'_{re} + a_w v'_{rw} + a_n v_{rn} + a_s v_{rs} + b$$

In order to be able to use TDMA, this equation is rewritten so that

$$a_p v_{rp} = a_e v_{re} + a_w v_{rw} + a_n v_{rn} + a_s v_{rs} + b + b_{curve}$$

where the last term contains the curvature effects, i.e.

$$b_{curve} = a_e(v'_{re} - v_{re}) + a_w(v'_{rw} - v_{rw})$$

2.6 Scalar Equations

The variables P', P'', T, k and ϵ are here referred to as scalar variables. These may all be written in the form:

$$\frac{\partial}{\partial\tau}(\rho\phi) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r\phi) + \frac{1}{r}\frac{\partial}{\partial\varphi}(\rho v_\varphi\phi) + \frac{\partial}{\partial z}(\rho v_z\phi)$$

$$= \frac{1}{r}\frac{\partial}{\partial r}(\mu_{eff}\, r\,\frac{\partial\phi}{\partial r}) + \frac{1}{r^2}\frac{\partial}{\partial\varphi}(\mu_{eff}\,\frac{\partial\phi}{\partial\varphi}) + \frac{\partial}{\partial z}(\mu_{eff}\,\frac{\partial\phi}{\partial z}) + S^\phi$$

Note that the radial co-ordinate r is inside the operator $\partial/\partial r$; this is taken account for in the program through multiplication by the appropriate face areas. The scalar equations have thus the same form in cylindrical co-ordinates as in cartesian co-ordinates (except that the radius r appears in the former case).

2.7 Generation Term in the k-ϵ Model

The viscous dissipation term ϕ in the energy equation (with the second viscosity λ=0) is very similar to the generation term in the k-equation. They have the form [1,5]

$$\phi = \sigma_{ij}\,\partial\tilde{U}_i/\partial x_j; \qquad G = \sigma^t_{ij}\,\partial U_i/\partial x_j$$

where σ_{ij} and σ^t_{ij} denote the viscous and the turbulent stress, respectively, and $\tilde{U}$ and U denote the instantaneous and the time average

velocity, respectively. σ_{ij} contains the molecular viscosity and σ^t_{ij} the turbulent. We obtain (with λ=0) [5]:

$$G = \mu_t[\ 2\ [(\frac{\partial v_r}{\partial r})^2 + (\frac{1}{r}\frac{\partial v_\varphi}{\partial \varphi} + \frac{v_r}{r})^2 + (\frac{\partial v_z}{\partial z})^2] + (\frac{1}{r}\frac{\partial v_z}{\partial \varphi} + \frac{\partial v_\varphi}{\partial z})^2$$

$$+ (\frac{\partial v_r}{\partial z} + \frac{\partial v_z}{\partial r})^2 + (\frac{1}{r}\frac{\partial v_r}{\partial \varphi} + \frac{\partial v_\varphi}{\partial r} - \frac{v_\varphi}{r})^2]$$

The production term in the k-equation is G; in the ϵ-equation it is $C_1 G\frac{\epsilon}{k}$.

2.8 SIMPLE, SIMPLEC and PISO

The following derivations are, for simplicity, carried out in one dimension. Let the superscripts *, ** and *** denote intermediate field values during the splitting process. Which of the three algorithms the program uses is determined as:

if PISO=.TRUE. => PISO-algorithm is used.
if SIMPLE=.TRUE. => SIMPLE-algorithm is used.
if SIMPLE=.FALSE. and PISO=.FALSE. => SIMPLEC-algorithm is used.

SIMPLE

In SIMPLE [8] the pressure p^* is guessed or taken as the value calculated at the last iteration. The u^* field is then obtained from the momentum equation. The corrected velocity, u^{**}, and pressure, p^{**}, are expressed as old values (u^*, p^*) plus corrections (u', p')

$$u^{**} = u^* + u' \qquad (1a)$$

$$p^{**} = p^{*} + p' \qquad (1b)$$

A pressure correction equation can be derived by the use of the momentum equation and the continuity equation [8]

$$a_p^p = \sum a_{nb}^p p'_{nb} + S^p \qquad (2a)$$

where superscript p denotes pressure correction equation; subscript nb denotes neighbours; the source S^p contains the continuity error; and

$$a_W^p = (\rho A)_w D_w; \quad D_u = A_w / a_p^u, \text{ etc.} \qquad (2b)$$

where subscript w denotes west face and A denotes area, and superscript u denotes u-equation. The velocity correction is given by

$$u'_w = D_u \left(p'_W - p'_P \right)$$

and the corrected fields u^{**} and p^{**} can thus be obtained from Eq. (1). SIMPLE is a one stage correction algorithm.

SIMPLEC

SIMPLEC [2] is identical to SIMPLE, except that the D_u in Eq. (2b) is calculated as

$$D_u = A_w / (a_p^u - \sum a_{nb}^u) \qquad (2c)$$

At a first sight it may seem that the denominator is zero in the equation above since

$$a_p^u = \sum a_{nb}^u - S_p^u$$

and S_p^u usually is zero. The a_p^u-coefficient has, however, been under-relaxated before it is used in Eq. (2c).

No special subroutines are written for SIMPLEC, but it is coded in the same subroutines as for SIMPLE.

PISO

PISO is a two stage correction algorithm [3]. The first correction is carried out as in SIMPLE; after that a second pressure correction equation is solved, and this second pressure correction, p'', is used to correct the velocity and pressure once more to obtain

$$u^{***} = u^{**} + u'' \tag{3a}$$

$$p^{***} = p^{**} + p'' \tag{3b}$$

The discretized u-momentum equation for u^{**} and u^{***} may be written as

$$a_p^u u_w^{***} = \sum a_{nb}^u u_{nb}^{**} + (p_W^{***} - p_P^{***})\, A_w + S^u$$

$$a_p^u u_w^{**} = \sum a_{nb}^u u_{nb}^{*} + (p_W^{**} - p_P^{**})\, A_w + S^u$$

When the latter is subtracted from the former we obtain

$$u_w'' = \frac{1}{a_p^u}\left[\sum a_{nb}^u u_{nb}' + (p_W'' - p_P'')\, A_w\right] \tag{4}$$

An equation for the pressure correction p'' can be derived in the same way as for SIMPLE [8]. By using the continuity equation and Eq. (3) we get

$$[\rho u^{***}]_e - [\rho u^{***}]_w = [\rho(u^{**} + u'')]_e - [\rho(u^{**} + u'')]_w$$

$$= [\rho u'']_e - [\rho u'']_w + [\rho u^{**}]_e - [\rho u^{**}]_w \tag{5}$$

Equation (4) in Eq. (5) gives

$$a_P^P \, p_P'' = \sum a_{nb}^P p_{nb}'' + S_{cont} + S_2 \qquad (6)$$

where

$$S_{cont} = [\rho u^{**}]_w - [\rho u^{**}]_e$$

$$S_2 = \left[\frac{1}{a_p^u} \sum a_{nb}^u u_{nb}' \right]_w - \left[\frac{1}{a_p^u} \sum a_{nb}^u u_{nb}' \right]_e$$

Note that the coefficients in Eq. (2) and Eq. (6) are identical, which means that the coefficients do not have to be calculated in CALCPP (CYLCPP); they have already been calculated in CALCP (CYLCP) when p' was solved. The source terms S_{cont} and S_2, have, however, to be calculated in CALCPP (CYLCPP).

Experience has shown that it is better <u>not</u> to correct the u^{**} velocity field, but to use u'' for correcting p^{**} only.

The PISO is coded in CALCPP (CYLCPP); this subroutine is called for when the logical variable PISO=.TRUE. The subroutine CALCPP (CYLCPP) is called for after CALCU (CYLCU), CALCV (CYLCV) and CALCW (CYLCW) in order to store the coefficients for these variables in local arrays; the subroutine is finally called for after CALCP (CYLCP) in order to calculate p'' and to correct pressure.

It should be noted that when using the PISO-algorithm, considerable extra storage is required in terms of local arrays in CALCPP (CYLCPP), namely 30 ITxJTxKT arrays (these are dimensioned in the COMMON-file PISCOM.FOR). The subroutines CALCPP & CYLCPP have therefore been made inactive by placing a C (comment line) in the first position of each line, in order to reduce storage requirements. Users who want to use the PISO algorithm must therefore delete the C's in the first position of all lines, <u>and</u> recompile the code. The user must, of course, also set PISO=.TRUE.

2.9 Cyclic Boundary Conditions

In many applications the flow is cyclic in the circumferential direction, i.e., it repeats itself every nth degree. By using the cyclic option (for cylindrical co-ordinates) this periodicity is taken advantage of in the program, by solving for n degrees in the x-direction only. The special solution algorithm is called CTDMA (Cyclic Tri Diagonal Matrix Algorithm).

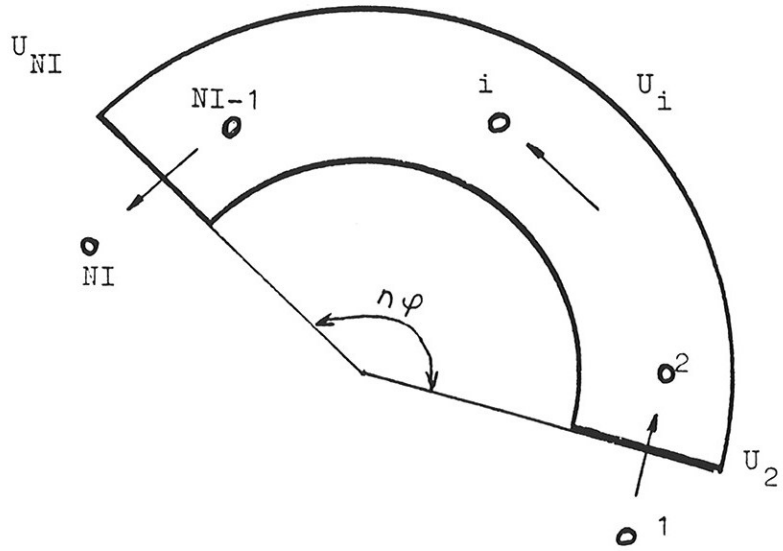


Figure 3. Configuration where cyclic boundary conditions are used.

For a configuration with cyclic boundary conditions in the circumferential direction (see Fig. 3), we obtain a discretised equation system as follows:

$$b_2\phi_2 - c_2\phi_3 \qquad\qquad -a_2\phi_{NI-1} = d_2$$

$$-a_i\phi_{i-1} + b_i\phi_i - c_i\phi_{i+1} \qquad\qquad = d_i$$

$$-c_{NI-1}\phi_2 \qquad -a_{NI-1}\phi_{NI-2} + b_{NI-1}\phi_{NI-1} = d_{NI-1}$$

Note that the ~~east~~ west (i.e. right) neighbour of ϕ_2 is ϕ_{NI-1}, and that the ~~west~~ east (i.e. left) neighbour of ϕ_{NI-1} is ϕ_2. The U-velocity is normally solved for from I=3 to I=NI-1 [1]. When cyclic boundary conditions are used U is solved for from I=2 to I=NI-1 since the velocity <u>at</u> the boundary must be calculated; note that $U_2 = U_{NI}$.

On matrix form we get

$$\begin{vmatrix} b_2 & -c_2 & & & & a_2 \\ -a_3 & b_3 & -c_3 & & & \\ & -a_i & b_i & -c_i & & \\ & & & -a_{NI-2} & b_{NI-2} & -c_{NI-2} \\ -c_{NI-1} & & & & -a_{NI-1} & b_{NI-1} \end{vmatrix} \begin{vmatrix} \phi_2 \\ \phi_3 \\ \phi_i \\ \phi_{NI-2} \\ \phi_{NI-1} \end{vmatrix} = \begin{vmatrix} d_2 \\ d_3 \\ d_i \\ d_{NI-2} \\ d_{NI-1} \end{vmatrix}$$

Due to the appearance of the corner elements a_2 and c_{NI-1} we can not use the standard TDM algorithm as in [1].

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

$$\begin{vmatrix} \beta_2 & -c_2 & & & & -\alpha_2 \\ & \beta_3 & -c_3 & & & -\alpha_3 \\ & & & \beta_i & -c_i & -\alpha_i \\ & & & & & \\ & & & & \beta_{NI-2} & -S_{NI-2} \\ -c_{NI-1} & & & & & \beta_{NI-1} \end{vmatrix} \begin{vmatrix} \phi_2 \\ \phi_3 \\ \phi_i \\ \\ \phi_{NI-2} \\ \phi_{NI-1} \end{vmatrix} = \begin{vmatrix} \gamma_2 \\ \gamma_3 \\ \gamma_i \\ \\ \gamma_{NI-2} \\ \gamma_{NI-1} \end{vmatrix}$$

where

$$\beta_2 = b_2 \tag{7a}$$

$$\beta_i = b_i - \frac{c_{i-1} a_i}{\beta_{i-1}} \tag{7b}$$

$$\alpha_2 = a_2 \tag{8a}$$

$$\alpha_i = \frac{\alpha_{i-1}\, a_i}{\beta_{i-1}} \tag{8b}$$

$$\beta_{NI-2} = b_{NI-2} - \frac{c_{NI-3}\, a_{NI-2}}{\beta_{NI-3}} \tag{9}$$

$$S_{NI-2} = c_{NI-2} + \frac{\alpha_{NI-3}\, a_{NI-2}}{\beta_{NI-3}} \tag{10}$$

$$\beta_{NI-1} = b_{NI-1} - \frac{S_{NI-2} a_{NI-1}}{\beta_{NI-2}} \tag{11}$$

c_{NI-1} does not change

$$\gamma_2 = d_2 \tag{12a}$$

$$\gamma_i = d_i + \frac{\gamma_{i-1} a_i}{\beta_{i-1}} \tag{12b}$$

Note that the coefficients in Eqs. (7), (8) and (12) are the same as for the TDM algorithm. Elimination of the c's from our matrix system gives

$$\begin{vmatrix} \beta_2 & & & -S_2 \\ & \beta_3 & & -S_3 \\ & & \beta_i & -S_i \\ & & \beta_{NI-2} & -S_{NI-2} \\ -c_{NI-1} & & & \beta_{NI-1} \end{vmatrix} \begin{vmatrix} \phi_2 \\ \phi_3 \\ \phi_i \\ \phi_{NI-2} \\ \phi_{NI-1} \end{vmatrix} = \begin{vmatrix} G_2 \\ G_3 \\ G_i \\ G_{NI-2} \\ G_{NI-1} \end{vmatrix}$$

where

$$S_i = \alpha_i + \frac{S_{i+1}\, c_i}{\beta_{i+1}} \tag{13}$$

$$G_i = \gamma_i + \frac{G_{i+1}\, c_i}{\beta_{i+1}} \tag{14a}$$

$$G_{NI-2} = \gamma_{NI-2};\ G_{NI-1} = \gamma_{NI-1} \tag{14b}$$

From the matrix we can write

$$\phi_i = \frac{1}{\beta_i}(S_i \phi_{NI-1} + G_i),\ i=2,3,\ldots,NI-2 \tag{15a}$$

From Eq. (15) we get an expression for ϕ_2 and from the matrix we obtain an expression for ϕ_{NI-1} so that

$$\phi_{NI-1} = \frac{G_2 c_{NI-1} + G_{NI-1}\beta_2}{\beta_{NI-1}\beta_2 - S_2 c_{NI-1}} \tag{15b}$$

The solution procedure can now be summarized as:

a) Calculate the matrix elements α, β and S_{NI-2} from Eqs. (7-11).

b) Calculate the matrix elements γ, S and G from Eqs. (12-14).

c) Calculate ϕ_{NI-1} from Eq. (15b).

d) Calculate the rest of the ϕ's from Eq. (15a)

When NI is smaller than 5 the algorithm described above has to be modified; rather than modifying the algorithm, this is not applied when NI is smaller than 5. The cyclic boundary condtions are automatically taken account for when the CTDMA-subroutine sweeps in the two other directions.

3. TEST CASES

3.1 Test Case 1: Abrupt Expansion in a Circular Pipe.

The axi-symmetric flow after an abrupt expansion in a circular pipe has been calculated, see Fig. 3. A recirculating region appears after the expansion, and the flow subsequently re-attaches to the wall. The flow is calculated for two different Reynolds numbers: Re=100 (laminar) and Re=4000 (turbulent).

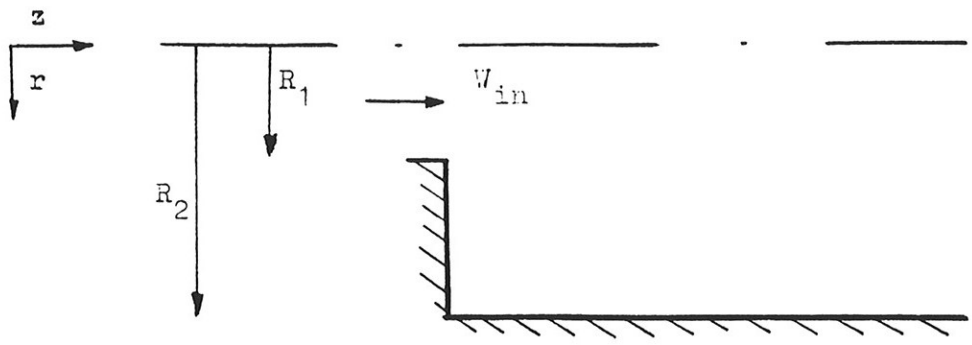


Figure 3. Configuration.

The inlet velocity profile is set as a parabolic profile for the laminar case, and according to the 1/7-th-power law for the turbulent case. The subroutine WALL has been used to impose standard wall-functions at the walls, and to impose zero gradient boundary conditions at the symmetry planes (VALUE.LE.-100.) At the outlet the exit velocity was calculated from mass balance, and zero streamwise gradient was imposed on the rest of the variables.

There exist experimental data on the location of the re-attachment point, z_{re} [9]. For the laminar case the calculations yield $z_{re}/H=10.7$ ($H=R_2-R_1$), which should be compared with the experimental value of 10 ± 0.15. The corresponding figures for the turbulent case are 9.25 and 9.5 ± 0.3. The

agreement between calculations and experiments for the laminar case is not very good; it should be mentioned that the calculations are rather sensitive to how the inlet velocity profile is prescribed. If a profile according to the 1/7-th-power law is prescribed for the laminar case also it gives $z_{re}/H=7$.

The laminar case was also calculated using the SIMPLEC and the PISO-algorithms. The following relaxation parameters were used for SIMPLE and PISO (they may not be optimal):

SIMPLE: 0.5 on the velocities and 0.8 on the pressure;

SIMPLEC: 0.8 on the velocities and 1.0 on the pressure.

PISO: 0.7 on the velocities and 1.0 on the pressure.

One of the advantages with SIMPLEC and PISO is that these algorithms are more stable than SIMPLE, so that less under-relaxation can be used [2,3] compared to SIMPLE; the disadvantage for PISO is that more CPU time is required per iteration, and that PISO needs very much extra core storage (see Section 2.8). SIMPLE converged on 123 iterations, SIMPLEC on 64 and PISO converged on 70 iterations; the corresponding required CPU time was: 90, 90, and 47 seconds. The SIMPLEC proves thus, for this particular case, to be the best algorithm.

A third case (laminar) has been calculated in which a swirl velocity was precribed at the inlet. The W-velocity was set to constant velocity (corresponding to Re=100), and U was set as

$$U = 0.1\ r/R_1 W_{in}$$

Since the case still is axi-symmetric, the case is two-dimensional; the difference from the two former cases is that in this case $U \neq 0$. The cyclic option thus has to be used.

No experimental data exist for this case.

3.2 Test Case 2: Rotating Couette Flow

The flow is driven by the outer cylinder (R_2=2), which rotates steadily with ω_2=1., while the inner cylinder (R_1=1) is stationary, see Fig. 5.

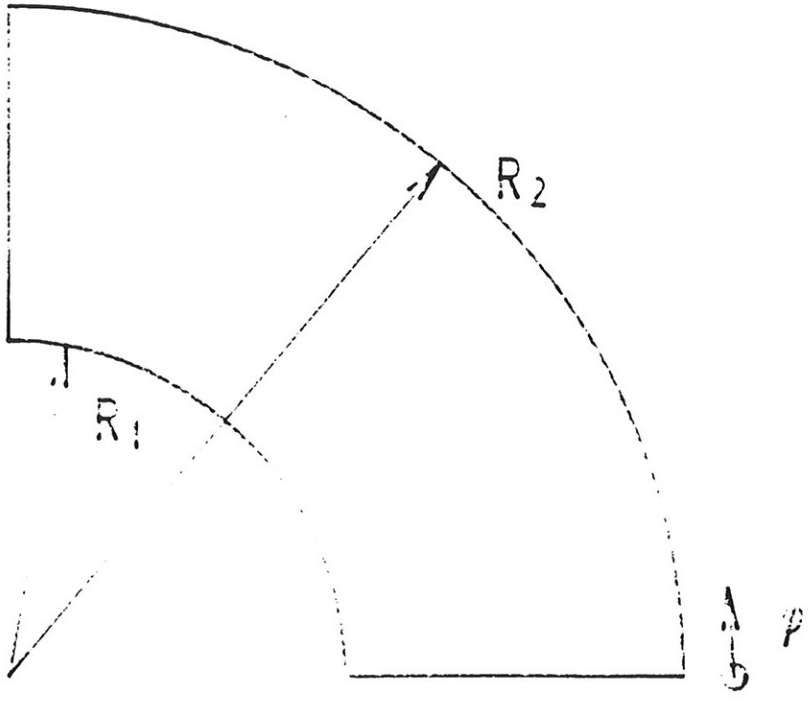


Figure 5. Configuration.

The flow is laminar and axi-symmetric. There exists an analytical solution [10]

$$U = \frac{4}{3}(r-1/r);\ V = 0;\ p = \frac{8}{9}\rho[\ r^2 - 1/r^2 - 4\ \ln(r)]$$

We have used this case to test the cyclic option in the program. The boundary conditions at the outer cylinder are U=2, V=0 (se above), and at the inner cylinder both velocities are zero. The boundary conditions at the inlet and the outlet are automatically handled by the cyclic boundary conditions.

The calculated results are compared with the analytical solution (printed out in the OUTDAT-file), and the agreement is very good (within one percent).

3.3 Test Case 3: Uniform Flow Across a Polar Grid

This test case has been chosen in order to estimate how much better results we obtain if the curvature effects are included by projecting the velocities (see Section 2.5), rather than using curvature source terms.

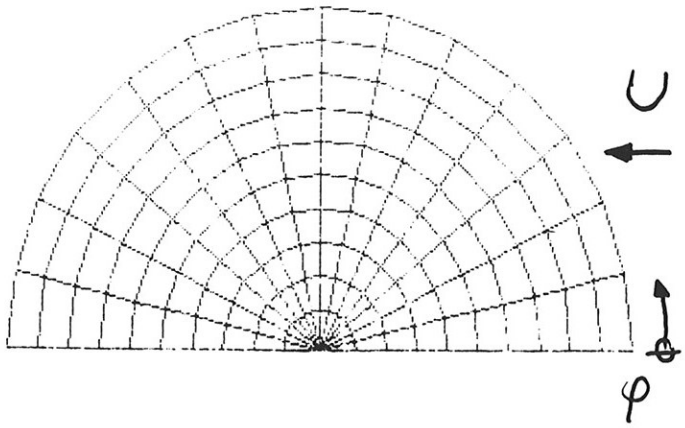


Figure 6. Configuration with grid.

The configuration with the grid is shown in Fig. 6. Since we want to estimate the errors for upwind differencing, the viscosity is set to zero. The boundary conditions are:

$$v_r = U \cos\alpha, \quad v_\varphi = -U \sin\alpha$$

which also, together with p=0, is the exact solution.

The flow has been calculated both using the standard treatment of curvature terms (CYLPRJ=.FALSE.; Case 3a), and using the procedure of projecting velocities (CYLPRJ=.TRUE.; Case 3b).

The maximal error in the velocities (se data-listings in OUTDAT.DAT) are 8.5% and 0.5% of U for Case 3a and 3b, respectively. In Fig. 7 the contours of the pressure are presented. The results for Case 3b is again much better that for Case 3a.

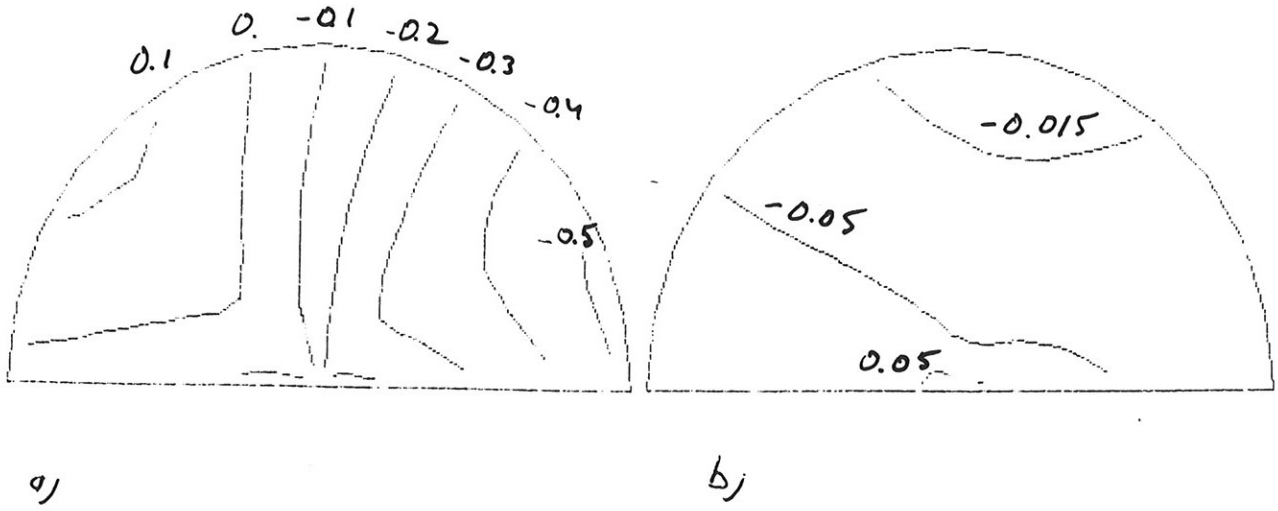


Figure 7. Contours of isobars. Numbers denote pressure scaled with dynamic pressure, $\rho U^2/2$. a) Case 3a. b) Case 3b.

4. FORTRAN SYMBOLS

A(I)	coefficient of recurrence formula
ALFA(I)	coefficient of recurrence formula
AE, AH, AL, AN, AS AW(I,J,K)	coefficients of convective/diffusive flux through east, high, low, north, south and west wall of control volume
AP(I,J,K)	sum of coefficients AE, AW, AN, AS, AH, AL and APO and source SP
APO(I,J,K)	coefficient for old time step
AREAEW, AREAHL, AREAN, AREAS	area of control volume faces east & west, high & low, north, and south
B, C(I)	coefficients of recurrence formula
C1, C2	constants of turbulence model (=1.44 and 1.92)
CAPPA	von Karman's constant (=0.435)
CD	constant of turbulence model (=1.0)
CE, CH,CL	coefficient of convective flux through east, high and low wall of control volume

CMU	constant of turbulence model (=0.09)
CMUCD	constant of turbulence model (=CMU*CD)
CN	coefficient of convective flux through north wall of control volume
CP	maximum of zero and net outflow (SMP) from control volume
CPO	=CP
CS, CW	coefficient of convective flux through south and west wall of control volume
CYCLIC	logical variable to control whether cyclic boundary conditions are to be applied or not
CYLPOL	logical variable to control whether cylindrical co-ordinates are to be used or not
CYLPRJ	logical variable which to control whether the procedure of
including	curvature effects by projecting velocity vectors are to be used or not (see Section 2.5)
D(I)	coefficient of recurrence formula
DEN, DENO(I,J,K)	density of fluid at current and old timestep
DENSIT	constant density of fluid set in INIT
DENU, DENV, DENW	density for U, V, and W control volume

DFE, DFH,
DFL, DFN,
DFS, DFW — coefficient of diffusive flux through east, high, low, north, south and west wall of control volume

DU(I,J,K) — coefficient of velocity-correction term for U velocity

DUDX, DUDY
DUDZ — $\partial U/\partial x$, $\partial U/\partial y$ and $\partial U/\partial z$ at control volume

DUDXM, DUDYM
DUDZM — $\partial U/\partial x$, $\partial U/\partial y$ and $\partial U/\partial z$ at negative (i.e. west, south or low) control volume face

DUDXP, DUDYP
DUDZP — $\partial U/\partial x$, $\partial U/\partial y$ and $\partial U/\partial z$ at positive (i.e. east, north or high) control volume face

DV(I,J,K) — coefficient of velocity-correction term for V velocity

DVDX, DVDY
DVDZ — $\partial V/\partial x$, $\partial V/\partial y$ and $\partial V/\partial z$ at control volume

DVDXM, DVDYM
DVDZM — $\partial V/\partial x$, $\partial V/\partial y$ and $\partial V/\partial z$ at negative (i.e. west, south or low) control volume face

DVDXP, DVDYP
DVDZP — $\partial V/\partial x$, $\partial V/\partial y$ and $\partial V/\partial z$ at positive (i.e. east, north or high) control volume face

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

DWDX, DWDY

DWDZ — $\partial W/\partial x$, $\partial W/\partial y$ and $\partial W/\partial z$ at control volume

DWDXM, DWDYM
DWDZM — $\partial W/\partial x$, $\partial W/\partial y$ and $\partial W/\partial z$ at negative (i.e. west, south or low) control volume face

DWDXP, DWDYP
DWDZP — $\partial W/\partial x$, $\partial W/\partial y$ and $\partial W/\partial z$ at positive (i.e. east, north or high) control volume face

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

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

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

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

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

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

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

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

DZHP(K) =Z(K+1)-Z(K)

DZHPW(K) =ZW(K+1)-ZW(K)

DZPL(K) =Z(K)-Z(K-1)

DZPLW(K) =ZW(K)-ZW(K-1)

ED, EDO(I,J,K)
: energy dissipation rate, ϵ, at current and old timestep

ELOG
: constant used in the log-low of the wall

FITER
: first iteration

GAMM, GAMP
: viscosity at negative (i.e. west, south or low) and positive (i.e. east, north or high) control volume face

GAME, GAMHX, GAML, GAMN, GAMS, GAMW
: coefficients of diffusion for scalar variables at east, high, low, north, south and west wall

GAMH(I,J,K)
: coefficient of diffusion for temperature

GE,GH,GL GN,GS,GW
: mass flux through east, high, low, north, south and west wall of cell

GEN(I,J,K)
: generation of turbulence by shear from mean flow

GP
: mass flux at location of velocity

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

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'

HEDW heading 'W-Velocity'

IMON I-index of monitoring location

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

INDMON monitoring output each INDMON iteration

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)

IPREF I-index of location where pressure is fixed if IPREF>0; if IPREF<0 the pressure is not fixed

INPRO logical variable for updating of fluid properties

IT I-index of maximum dimension of dependent variable

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	J-index of location where pressure is fixed if KPREF>0; if KPREF<0 the pressure is not fixed
KT	K-index of maximum dimension of dependent variable
MAXIT	maximum number of iterations to be completed in the current run if iteration is not stopped by test on value of SORCE
NI	maximum value of I-index for the calculation domain
NIM1	=NI-1
NITER	number of iterations completed
NJ	maximum value of J-index for the calculation domain
NJM1	=NJ-1
NK	maximum value of K-index for the calculation domain
NKM1	=NK-1
NSWPD,NSWPK	

NSWPP,NSWPT
NSWPU,NSWPV
NSWPW — number of application of line iteration for ϵ, k, P', T, U, V and W-equation

P(I,J,K) — pressure

PFUN — constant of P-function for heat transfer at walls

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

PISO — logical variable to control whether the PISO-algorithm is to be used or not (see Section 2.8)

PLANE — 2-character variable which controls in which plane the fields are printed out; the fields can be printed in all three planes, and PLANE can be set to XY, XZ, or YZ. Default: XY

PP(I,J,K) — pressure-correction, P' and P''

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

RESOR — residual source for individual control volume

RESORE,RESORK
RESORM,RESORT
RESORU,RESORV
RESORW — sum of residual sources within calculation domain for ϵ, k, P', T, U, V and W-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 stored on the file UTDATA or not

SEW(I) 0.5*[DXEP(I)+DXPW(I)]

SEWU(I) 0.5*[DXEPU(I)+DXPWU(I)]

SMP net outflow from control volume

SHL(K) 0.5*(DZHP(K)+DZPL(K)]

SHLW(K) 0.5*[DZHPW(K)+DZPLW(K)]

SIMPLE logical variable to control if the SIMPLE-algorithm is to be used or not (see Section 2.8)

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

SNS(J) 0.5*[DYNP(J)+DYPS(J)]

SNSV(J) 0.5*[DYNPV(J)+DYPSV(J)]

SORCE maximum of RESORM, RESORU, RESORV and RESORW

SORMAX maximum acceptable value of SORCE for converged solution

SP,SU(I,J,K) coefficient b and C of linearized source treatment

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

T,
TO(I,J,K) temperature at current and old time step

TE,
TEO(I,J,K) turbulent kinetic energy at current and old time step

TMULT coefficient of wall shear expression

U,
UO(I,J,K) velocity in the x-direction at current and old time step

UM U-velocity at a negative (i.e. west, south or low) control volume face

UP U-velocity at a positive (i.e. east, north or high) control volume face

UPLUS $u_*/U_{parallel}$

UR, URM,
URP v_φ/r in the control volume; at the negative (i.e. west, south or low) control volume face; and at the positive (i.e. east, north or high) control volume face

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

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

V,
VO(I,J,K) velocity in the y-direction at current and old time step

VELPAR	velocity parallel to the wall
VIS(I,J,K)	effective viscosity ($\mu+\mu_t$)
VISCOS	laminar viscosity (μ)
VISE,VISH VISL,VISN VISS,VISW	effective viscosity at midpoint of east, high, low, north, south and west wall of cell
VISOLD	value of effective viscosity before under-relaxation
VM	V-velocity at a negative (i.e. west, south or low) control volume face
VP	V-velocity at a positive (i.e. east, north or high) control volume face
VOL	volume of control volume
VRM	ρv_r
W, WO(I,J,K)	the velocity in the z-direction at the current and the old time step
X(I)	x co-ordinate of main cells; for the cylindrical option x denotes the circumferential direction and x is then measured in radians
XMOMIN	momentum of fluid at inlet of flow domain

XP	x-distance [=XP*(radius) for the cylindrical option] between wall and adjacent grid node
XU(I)	x co-ordinate at storage location of U
Y(J)	y co-ordinate of main cells; for the cylindrical option y denotes the radial co-ordinate
YP	y-distance between wall and adjacent grid node
YV(J)	y co-ordinate at storage location of V
Z(K)	z co-ordinate of main cells; for the cylindrical option z denote the axial co-ordinate
ZP	z-distance between wall and adjacent grid node
ZW(K)	z co-ordinate at storage location of W

5. REFERENCES

1. 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).

2. 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).

3. R.I. Issa, "Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting", J. Comp. Physics, Vol. 62, pp. 40-65 (1986).

4. 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)

5. W.F. Hughes and E.W. Gaylord, Basic Equations of Engineering Science, Shaums's Outline Series, McGraw-Hill, New York (1964).

6. L. Davidson and P. Hedberg, "Mathematical Derivation of a Finite-Volume Formulation for Laminar Flow in Complex Geometries", in print (1988).

7. L. Davidson and P. Hedberg, "FLUX2D: A Finite-Volume Computer Program Written in General Non-Orthogonal Co-ordinates for Calculation of Two-Dimensional Turbulent Flow", in print (1988)

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

9. L.H. Back and E.J. Roschke, "Shear-Layer Flow Regions and Wave Instabilities and Reattachment Lengths Downstream of an Abrupt Circular Channel Expansion", J. of Appl. Mech., pp. 677-681 (1972)

10. H. Schlichting, Boundary-Layer Theory, 7th ed., McGraw-Hill (1979).