General purpose flow solver applied to flow over hills

Sørensen, Niels N.

Publication date:
1995

Document Version
Publisher's PDF, also known as Version of record

Risø–R–827(EN)

General Purpose Flow Solver Applied to Flow over Hills

Niels N. Sørensen

Risø National Laboratory, Roskilde, Denmark
December 2003

Abstract A 2D and a 3D finite-volume code in general curvilinear coordinates have been developed using the Basis2D/3D platform by Michelsen. The codes are based on the Reynolds averaged incompressible isothermal Navier-Stokes equations and use primitive variables (U, V, W and P). The turbulence is modelled by the high Reynolds number $k-\epsilon$ model.

Cartesian velocity components are used in a non-staggered arrangement following the methodology of Rhie. The equation system is solved using the SIMPLE method of Patankar and Spalding. Solution of the transport equations is obtained by a successive application of a TDMA solver in alternating direction. The solution of the pressure correction equation is accelerated using the multigrid tools from the Basis2D/3D platform. Additionally a three-level grid sequence was implemented in order to minimize the overall solution time.

As turbulent boundary conditions two versions of the logarithmic law-of-the-wall were implemented in order to handle both atmospheric and engineering flows, the wall being assumed to be rough or smooth, respectively.

Higher-order schemes (SUDS and QUICK) were implemented as explicit corrections to a first-order upwind difference scheme.

In both the 2D and the 3D code it is possible to handle multiblock configurations. This feature was added in order to obtain a greater geometric flexibility. The multiblock capability was achieved by use of the domain decomposition tools of the Basis2D/3D environment.

To mesh natural terrain in connection with atmospheric flow over complex terrain, a two- and a three-dimensional hyperbolic mesh generator were constructed. Additionally, a two- and a three-dimensional mesh generator based on a simple version of the transfinite interpolation technique were implemented.

Several two-dimensional test cases were calculated e.g. laminar flow over a circular cylinder, turbulent channel flow, and turbulent flow over a backward facing step, all with satisfying results. In order to illustrate the application of the codes to atmospheric flow two cases was calculated, flow over a cube in a thick turbulent boundary-layer, and the atmospheric flow over the Askervein hill.

For the flow over a cube, the results shows good agreement with the measurements capturing the major structures of the flow, including the large separated region behind the cube, and the horse-shoe vortex in front of the cube. Also, the velocity profiles and pressure distributions on the cube surface are in good agreement with measurements.

For flow over Askervein the speed-up and flow angle in ten meters height over terrain shows an excellent agreement with the measurements, except near the hill summit where the values is found to be to low.

This dissertation is submitted in partial fulfilment of the requirements for the Danish Ph.D. degree in technological sciences at the Technical University of Denmark. The dissertation is based on theoretical and numerical work carried out during the period January 1992 to December 1994 at The Department of Meteorology and Wind Energy, Risø National Laboratory.

ISBN 87-550-2079-8
ISSN 0106-2840

Grafisk Service · Risø · 2003

Contents

Last thing I remember, I was
Running for the door
I had to find the passage back
To the place I was before
'Relax', said the night man,
'We are programmed to receive.
You can check out any time you like,
but you can never leave.'

D. Henley, G. Frey, and D. Felder

1 Introduction

1.1 Background for the present work

Many different subjects with connection to the atmospheric boundary layer are studied at The Department of Meteorology and Wind Energy.

One of these is the study of flow over complex terrain, a topic connected with many practical problems, e.g. wind turbine siting, dispersion in natural terrain, and dispersion within cities. The present work has evolved from the study of flow over complex terrain.

The methods used to study flow over complex terrain range from theoretical considerations, over full-scale measurements and wind tunnel studies to numerical modeling. The present work is dealing with numerical modeling and is a continuation of the numerical study of atmospheric flows using computational fluid dynamics (CFD) started in The Department of Meteorologi and Wind Energy in the late eighties.

The project on numerical modeling was started in cooperation with The Department of Fluid Mechanics at DTU and The Danish Maritime Institute. The aim of the study was to investigate whether 2D/3D CFD codes using rectangular grids could be used to compute flow over complex terrain. The work focussed on the KAMELEON code (a commercial code of the Patankar and Spalding type written in Trondheim by Berge [4]). Two different approaches were tried in order to apply the KAMELEON code to atmospheric flow.

The first approach was to investigate the direct application of the KAMELEON code to nonrectangular geometries. The nonrectangular hill shapes were modeled by blocking out the hill in a rectangular grid. This method resulted in the hill surface being represented as a staircase. As the boundary controls the flow, a very fine resolution was necessary in order to obtain an accurate solution with the staircase method. The results of this approach were unsatisfactory as the model was unable to predict the correct flow behaviour, see de Baas [7].

In the second approach, no attempts were made to model the actual geometry of the hill. It was assumed that the terms arising during the transformation of the curvilinear grid into a rectangular grid were of minor importance for the horizontal flow. Therefore the rectangular grid of the KAMELEON code was retained and instead the hill was modeled by fixing the pressure according to the potential flow solution over the hill. Using the potential pressure, the computing time could be considerable lowered compared to the use of the static pressure. Unfortunately, the use of potential pressure along with neglecting the curvature terms in the flow equations limit the possible range of application to flows over terrain with moderate slopes and without separation. Additionally, this method is unable to represent the vertical velocity perturbations of the hill as the transformation terms are neglected. Despite these obvious limitations the results of the model was encouraging, see [7].

1.2 The present study

It was decided to carry on the work using CFD to compute flow over complex terrain. Based on the experience gained in connection with the KAMELEON code, the new code would have to use general non-orthogonal curvilinear coordinates, and the hydrodynamic pressure. Without the limitation of using the potential pressure, the resulting code should be able to compute separated flow, and no limits on the terrain slope would exist. Using curvilinear coordinates, the hill shape could be modeled accurately with no need of an excessive fine grid therby

limiting the number of necessary computational cells.

It was decided to use the SIMPLE method of Patankar and Spalding [43] to compute the incompressible isothermal flow using Cartesian velocity components in a co-located arrangement. To avoid the well-known pressure decoupling, the Rhie/Chow interpolation technique was used, see Rhie [47].

As turbulence model the $k - \epsilon$ model was chosen, mainly because it is well tested, reasonably accurate and easily implemented. The Reynolds stress model was also considered but abandoned as the improvements of the results are only minor compared to the $k - \epsilon$ model, whereas the implementation is much more complicated.

It was obvious that with a minimum of extra work the flow code could be made applicable to both small-scale atmospheric flows and industrial flows. This extra work was primarily concerned with the turbulence model, where two different versions of the logarithmic law-of-the-wall used as boundary conditions are needed.

Using the SIMPLE algorithm to obtain the pressure velocity coupling, the elliptic pressure correction equations need to be solved within a certain tolerance. As the solution of the pressure correction equations is known to be very time-consuming, multigrid methods were investigated and a simple CG multigrid method was implemented in the developed two-dimensional test code.

The possibility of expanding the geometric flexibility of the code, using the multiblock approach, was investigated. A 2D multiblock code was inertially developed. The final 2D/3D codes were implemented using the basis2D/3D platform by Michelsen [33], providing both the multigrid solver and the domain decomposition tools.

In the first part of the report the steps necessary for developing the model, i.e. transformation of the flow equations, discretization, and related subjects will be addressed. Also the subject of generating general non-orthogonal curvilinear meshes will be addressed. In the second part of the report, the developed codes are validated in several test cases, e.g. laminar flow around a cylinder, turbulent channel flow, turbulent flow over backward facing step. And finally the code is applied to flow over a cube in a thick turbulent boundary layer, and to prediction of the atmospheric boundary layer flow over the Askervein hill, both as a 2D and as a 3D simulation.

2 Governing Equations

2.1 Introduction

The present work is concerned primarily with the development of a general-purpose numerical flow solver for turbulent incompressible isothermal flow. The model is intended to be used both for small scale atmospheric flows[1] and for industrial flows.

The approach chosen is to work with the pressure velocity formulation of the Reynolds averaged Navier-Stokes equations (RANS), using the well-known high Reynolds number $k-\epsilon$ model to describe the turbulence. The $k-\epsilon$ model is chosen because it is well tested, relatively simple (only two extra partial differential equations has to be solved), and it is well suited for steady-state calculations.

2.2 Navier-Stokes equations

The governing equations can be stated in a coordinate free form as

$$\frac{\partial}{\partial t}(\tilde{\rho}) + \nabla \cdot (\tilde{\rho}\tilde{\boldsymbol{v}}) = 0 \ , \tag{1}$$

$$\frac{\partial}{\partial t}(\tilde{\rho}\tilde{\boldsymbol{v}}) + \nabla \cdot (\tilde{\rho}\tilde{\boldsymbol{v}} \otimes \tilde{\boldsymbol{v}} - \boldsymbol{T}) = S_{\tilde{\boldsymbol{v}}} \ , \tag{2}$$

$$\frac{\partial}{\partial t}(\tilde{\rho}\tilde{\varphi}) + \nabla \cdot (\tilde{\rho}\tilde{\boldsymbol{v}}\tilde{\varphi} - \tilde{\boldsymbol{q}}) = S_{\tilde{\varphi}} \ . \tag{3}$$

where

$\tilde{\rho}$ density ,

$\tilde{\boldsymbol{v}}$ velocity vector ,

$\tilde{\varphi}$ scalar quantity ,

$\boldsymbol{T}$ stress tensor ,

$\tilde{\boldsymbol{q}}$ flux vector .

The stress tensor is given as

$$\boldsymbol{T} = -(\tilde{P} + \frac{2}{3}\mu\nabla \cdot \tilde{\boldsymbol{v}})\boldsymbol{I} + 2\mu\boldsymbol{S} \ ,$$

where $\boldsymbol{S}$ is the rate of deformation tensor

$$\boldsymbol{S} = \frac{1}{2}\begin{bmatrix} \frac{\partial\tilde{u}}{\partial x} + \frac{\partial\tilde{u}}{\partial x} & \frac{\partial\tilde{u}}{\partial y} + \frac{\partial\tilde{v}}{\partial x} & \frac{\partial\tilde{u}}{\partial z} + \frac{\partial\tilde{w}}{\partial x} \\ \frac{\partial\tilde{u}}{\partial y} + \frac{\partial\tilde{v}}{\partial x} & \frac{\partial\tilde{v}}{\partial y} + \frac{\partial\tilde{v}}{\partial y} & \frac{\partial\tilde{v}}{\partial z} + \frac{\partial\tilde{w}}{\partial y} \\ \frac{\partial\tilde{u}}{\partial z} + \frac{\partial\tilde{w}}{\partial x} & \frac{\partial\tilde{w}}{\partial y} + \frac{\partial\tilde{v}}{\partial z} & \frac{\partial\tilde{w}}{\partial z} + \frac{\partial\tilde{w}}{\partial z} \end{bmatrix} \ ,$$

$\tilde{P}$ is the mean normal compressive stress, or the static pressure

$$\tilde{P} = -\frac{1}{3}T_{kk} \ ,$$

and $\boldsymbol{I}$ is the unit tensor

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ .$$

[1] In this work small scale atmospheric flows will refer to flows of a typical dimension less than 10 km $\times$ 10 km $\times$ 10 km.

Reynolds averaged Navier-Stokes equations

In principle, the Navier-Stokes (NS) equations suffice to describe turbulent flow. Unfortunately, use of the NS equations in a discrete formulation as a finite volume model, pose some practical problems.

The reason for this is the wide range of length and time scales in a turbulent flow. The largest length scale is equal to the geometry in question, and the smallest length scale is of the size of the dissipative eddies. The smallest time scale of the flow is equal to the turbulent fluctuation time, and the largest time scales are of the size of the slowly changing mean variation of the flow.

If this should be accounted for in a numerical simulation, the computational mesh would have to be fine enough to model the dissipative eddies, and at the same time cover the total domain requiring a large number of cells. In order to resolve the time scales of the flow, small time steps must be used to capture the fast turbulent fluctuations. Also long times series have to be calculated in order to catch the large-scale time variation of the flow, resulting in a very large number of time-steps.

In the present work RANS equations will be used to model the turbulence, using the well-known high Reynolds number $k-\epsilon$ turbulence model.

The time averaged RANS model is well suited for steady-state calculations, and thus the computer time can be minimized for flows that are steady using well established acceleration techniques for steady-state calculations. On the other hand, it can be argued that the model can be used even for simulating transient flows, predicting the transient behaviour of the mean flow when the instationarities of the mean flow are on a time scale which is large compared to the averaging time.

The RANS equation is derived from the standard NS equations by decomposing the variables into mean and fluctuating quantities marked with a $\prime$, followed by a time averaging. The Reynolds decomposition technique yields

$$\begin{aligned}
\tilde{u_i} &= U_i + u_i' \ , \\
\tilde{P} &= P + p' \ , \\
\tilde{\rho} &= \rho + \rho' \ , \\
\tilde{\varphi} &= \varphi + \varphi' \ ,
\end{aligned}$$

etc.

Now writing the governing equations in Cartesian coordinates, expanding the volumetric momentum source into a gravitational force ρg_i plus the Coriolis force $2\epsilon_{ijk}\Omega_j U_k$ and keeping a general part $S_{\boldsymbol{v}}$ for later inclusion of other volume sources. The terms in the expression for the Coriolis force are the permutation tensor ϵ_{ijk} and the angular velocity vector of the earth rotation Ω_j. Inserting the Reynolds decomposed variables and performing time averaging we get

$$\begin{aligned}
&\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x_j}(\rho U_j) = 0 \ , \\
&\frac{\partial}{\partial t}(\rho U_i) + \frac{\partial}{\partial x_j}(\rho U_i U_j) - \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \rho\overline{u_i' u_j'}\right] \\
&\qquad\qquad + \frac{\partial P}{\partial x_i} = \rho g_i - 2\epsilon_{ijk}\Omega_j U_k \ , \\
&\frac{\partial}{\partial t}(\rho\varphi) + \frac{\partial}{\partial x_i}(\rho U_i \varphi) - \frac{\partial}{\partial x_i}\left(\mu\frac{\partial \varphi}{\partial x_i} - \rho\overline{u_i'\varphi'}\right) = S_\varphi \ .
\end{aligned}$$

As a result of the averaging process, extra terms $\rho\overline{u_i' u_j'}$ and $\rho\overline{u_i'\varphi'}$ have appeared. They are called Reynolds stresses and Reynolds fluxes, respectively, and

are grouped together with the laminar stresses or fluxes, even though they come from the convective terms.

In order to make the number of unknowns equal to the number of equations, the Reynolds stresses and fluxes have to be modelled, this problem is the well-known turbulent closure problem. Here the Boussinesq approximation, often called the eddy viscosity technique, will be used to obtain closed expressions for the Reynolds stresses and fluxes. This results in the following approximation for the respective terms

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

$$\rho\overline{u_i'\varphi'} = -\mu_t\left(\frac{\partial \varphi}{\partial x_i}\right) \ .$$

The assumption of incompressible flow implies that no density fluctuations can exist, and the term $\rho' g_i$ must vanish. As we are considering flow in a hydrostatic equilibrium, the gravitational force ρg_i is absorbed into the pressure term

$$\frac{\partial \hat{P}}{\partial x_i} = \frac{\partial P}{\partial x_i} - \rho g_i \ ,$$

in the following the superscript $\hat{}$ will be dropped, and the pressure just denoted P, even though it now represents the perturbation around the hydrostatic pressure. Also the term $2/3\rho k\delta_{ij}$ will be absorbed into the pressure term following standard practice.

The scales of the flows to be considered will be relatively small, a typical length scale will be less than 10 km, so the Coriolis force will be omitted. Inserting this into the RANS equations we end up with the following equations

$$\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x_j}(\rho U_j) = 0 \ , \tag{4}$$

$$\frac{\partial}{\partial t}(\rho U_i) + \frac{\partial}{\partial x_j}(\rho U_i U_j) - \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\right] + \frac{\partial P}{\partial x_i} = S_{\boldsymbol{v}} \ , \tag{5}$$

$$\frac{\partial}{\partial t}(\rho\varphi) + \frac{\partial}{\partial x_i}(\rho U_i\varphi) - \frac{\partial}{\partial x_i}\left[\left(\mu + \frac{\mu_t}{\sigma_\varphi}\right)\frac{\partial \varphi}{\partial x_i}\right] = S_\varphi \ . \tag{6}$$

2.3 $k - \epsilon$ equations

The approximation made for the Reynolds terms in the previous section demands a technique in order to obtain the eddy viscosity. Here the first-order two-equations $k - \epsilon$ model will be used for this purpose. The eddy viscosity is expressed as the ratio between the square of the turbulent kinetic energy and the dissipation of turbulent kinetic energy multiplied by a constant

$$\mu_t = C_\mu \frac{k^2}{\epsilon} \ . \tag{7}$$

In order to use this expression two equations must be derived, one equation governing the turbulent kinetic energy, k, and the second equation governing the dissipation of turbulent kinetic energy, ϵ.

The derivation of these equations will not be given in detail here, as the derivation is rather lengthy. Neither will the approximation used to close the equations be discussed, the reader is instead referred to standard textbooks on the subject.

Starting with the equation for the mean energy of the turbulent fluctuations (turbulent kinetic energy) the following is done. First the Reynolds decomposed Navier-Stokes equations are multiplied by u_i' and time averaged followed by a subtraction of the result of multiplying the RANS by U_i. Finally some of the

terms in the equation have to be modelled before the equation can be used, all this results in the following equation for turbulent kinetic energy

$$\frac{\partial}{\partial t}(\rho k)+\frac{\partial}{\partial x_j}(\rho U_j k)-\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right]=\mu_t\frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)-\rho\epsilon \quad .(8)$$

The equation for dissipation of the turbulent energy can also be derived from the NS equations by first taking the partial derivative $\partial/\partial x_j$ of the momentum equations for the turbulent fluctuations [2], and then multiplying them by $\mu\partial u_i/\partial x_j$ and performing time averaging of the result. Again some of the terms need modelling before the equation can be used, and finally we get

$$\frac{\partial}{\partial t}(\rho\epsilon)+\frac{\partial}{\partial x_j}(\rho U_j\epsilon)-\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial\epsilon}{\partial x_j}\right] = C_{\epsilon 1}\frac{\epsilon}{k}\mu_t\frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)-\rho C_{\epsilon 2}\frac{\epsilon^2}{k} \quad .(9)$$

It is easily seen that both (8) for turbulent kinetic energy and (9) for dissipation of turbulent kinetic energy have the form of the generic scalar transport equation (3). This implies that no special attention is needed for discretization of these equations, except for the treatment of the source terms, which can be done in exactly the same way as for all other scalar transport equations.

Looking at the k and ϵ equation, the lack of the Coriolis force is striking. If we look at the three individual equations for the normal Reynolds stresses instead of looking at the equation for k, we would discover that in these equations the Coriolis force is still present. The explanation of this is that the Coriolis force does not change the energy of the mean normal stresses, it only redistributes energy among the three components.

One important consequence of this is that the $k-\epsilon$ model is unable to properly describe a flow where the redistribution effect of the Coriolis force is essential. In such cases the $k-\epsilon$ model must be abandoned, and another turbulence model that is capable of describing these effects must be used instead, one obvious choice would be the Reynolds stress model.

$k-\epsilon$ model constants

In the past a great deal of work has been put into the determination of the constants of the $k-\epsilon$ model and testing them on a wide variety of flows. The constants have been determined from experimental data, considering basic universal flows, and afterwards these findings have been improved by computer optimization.

The constant C_μ is found by considering equilibrium turbulence near a wall, and this leads to the following equation

$$C_\mu=\left(\frac{U_\tau^2}{k}\right)^2 \quad (10)$$

where U_τ is the friction velocity at the wall.

Again considering equilibrium turbulence near a wall, a relationship between the $C_{\epsilon 1}$, $C_{\epsilon 2}$ and C_μ constants has been determined

$$C_{\epsilon 1}=C_{\epsilon 2}-\frac{\kappa^2}{C_\mu^{\frac{1}{2}}\sigma_\epsilon} \quad , \quad (11)$$

where κ is the von Karman constant equal to 0.40.

The constants $C_{\epsilon 2}$ and σ_ϵ are determined by considering the decay of grid turbulence giving the following values

$$C_{\epsilon 2}\simeq 1.92 \quad (12)$$

[2] The equations for the turbulent fluctuations are the results of subtracting the RANS from the Reynolds decomposed Navier-Stokes equations.

$$\sigma_\epsilon \simeq 1.30 \tag{13}$$

To some extent the ratio of turbulent diffusivity of momentum and turbulent kinetic energy is equal, resulting in the turbulent Prandtl number σ_k having a value of 1.00.

For industrial flows the value of C_μ is well established

$$C_\mu = \left(\frac{U_\tau^2}{k}\right)^2 = 0.09 \ ,$$

resulting in the set of constants listed in table 1 for the $k-\epsilon$ model, originally proposed by the Imperial College Group, see Launder and Spalding [22].

Table 1. $k-\epsilon$ model constants for industrial flow, according to [22].

κ	C_μ	σ_k	σ_ϵ	$C_{\epsilon 1}$	$C_{\epsilon 2}$
0.40	0.09	1.00	1.30	1.42	1.92

For atmospheric flows the typical value of k/U_τ^2 differs from the value found for industrial flows, see Panofsky and Dutton [41], Zeman and Jensen [65] and Raithby, Stubley and Taylor [46]. This results in a different value for C_μ

$$C_\mu = \left(\frac{U_\tau^2}{k}\right)^2 = 0.03 \ ,$$

and as $C_{\epsilon 2}$ and σ_ϵ are unchanged, this difference in C_μ results in a $C_{\epsilon 1}$ value of

$$C_{\epsilon 1} = C_{\epsilon 2} - \frac{\kappa^2}{C_\mu^{\frac{1}{2}}\sigma_\epsilon} = 1.21 \ . \tag{14}$$

The values of the $k-\epsilon$ model constants used as a standard for stable atmospheric boundary layer flows are listed in table 2.

Table 2. $k-\epsilon$ model constants for atmospheric boundary layer flows.

κ	C_μ	σ_k	σ_ϵ	$C_{\epsilon 1}$	$C_{\epsilon 2}$
0.40	0.03	1.00	1.30	1.21	1.92

2.4 Closure

The equations governing the time averaged turbulent flow are based on using Reynolds decomposition of the variables and time averaging. The resulting equations for momentum are identical to the laminar counterpart except that the viscosity has been exchanged by an effective viscosity $\mu_{ef} = \mu + \mu_t$. This means that changing from calculating laminar flow to calculating turbulent flow is simple, just choosing either to use the $k-\epsilon$ model to calculate the eddy viscosity or setting it equal to zero.

The fact that the equations for k and ϵ have the form of the generic equations for scalar transport is convenient as this means that no special considerations of the discretization of these equations are necessary.

It has been argued that even though the model is intended primarily for steady-state calculation, the model can be used for transient calculations when the time variations of the mean flow are slow.

Two sets of constants for the model have been derived, one set for use in industrial flows and the other set for use in the atmospheric boundary layer.

3 Equations in curvilinear form

3.1 Introduction

For the flow in complex domains it is often not possible to use Cartesian or rectangular coordinates. Therefore the flow equations will be transformed into general curvilinear coordinates, allowing the grids to conform to the boundaries even for complex domains.

The transformed equations can be written in at least four different forms, see Viviand [61] and Hindman [13]. The four different forms are called the chain-rule conservation form, weak conservation form, strong conservation form, and non-conservation form.

Even though all of these different forms of the transformed equations are equally good referring to the differential forms of the equations, this is not true for the discrete versions of the equations. Only the strong conservation form will result in discrete equations being conservative, meaning that the accumulation within the domain is equal to the inflow over the domain boundaries minus the outflow over the domain boundaries plus the production within the domain.

The reason why the strong conservation form, in contrast to the other formulations, results in the discrete equation being conservative is that only the strong conservation law form will result in discrete equations where all fluxes are associated with cell faces. All of the remaining formulations will result in discrete formulations where some fluxes are associated with the cell centres.

When all the fluxes are evaluated at cell faces, it is easy to ensure that the flux leaving a cell through a cell face, will enter the neighbouring cell sharing that cell face. This will result in the telescopic collapse of the internal fluxes, leaving only the fluxes at the domain boundary. For the other formulations there is no obvious way of obtaining the telescopic collapse of the internal fluxes as some fluxes is associated with cell centres.

3.2 Transformation relations

Working with a curvilinear coordinate system, the Cartesian flow equations must be transformed into the actual coordinate system. The transformation between the curvilinear coordinate (ξ, η, ζ) and the Cartesian coordinate systems (x, y, z) is defined as

$$\begin{aligned} \xi &= \xi(x, y, z) \ , \\ \eta &= \eta(x, y, z) \ , \\ \zeta &= \zeta(x, y, z) \ , \end{aligned}$$

or

$$\begin{aligned} x &= x(\xi, \eta, \zeta) \ , \\ y &= y(\xi, \eta, \zeta) \ , \\ z &= z(\xi, \eta, \zeta) \ . \end{aligned}$$

using these relations the flow equations can be transformed. The partial differentials can be expressed by the chain rule

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial}{\partial \zeta}\frac{\partial \zeta}{\partial x} = \frac{\partial}{\partial \xi}\xi_x + \frac{\partial}{\partial \eta}\eta_x + \frac{\partial}{\partial \zeta}\zeta_x \ , \tag{15}$$

$$\frac{\partial}{\partial y} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial}{\partial \zeta}\frac{\partial \zeta}{\partial y} = \frac{\partial}{\partial \xi}\xi_y + \frac{\partial}{\partial \eta}\eta_y + \frac{\partial}{\partial \zeta}\zeta_y \ , \tag{16}$$

$$\frac{\partial}{\partial z} = \frac{\partial}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial}{\partial \eta}\frac{\partial \eta}{\partial z} + \frac{\partial}{\partial \zeta}\frac{\partial \zeta}{\partial z} = \frac{\partial}{\partial \xi}\xi_z + \frac{\partial}{\partial \eta}\eta_z + \frac{\partial}{\partial \zeta}\zeta_z \ . \tag{17}$$

Before these relations can be used to transform the flow equations we must derive expressions for the metric terms $\xi_x, \xi_y, \xi_z, \eta_x, \eta_y, \eta_z, \zeta_x, \zeta_y$ and ζ_z. First we write the differential expressions

$$\begin{aligned} d\xi &= \xi_x dx + \xi_y dy + \xi_z dz \ , \\ d\eta &= \eta_x dx + \eta_y dy + \eta_z dz \ , \\ d\zeta &= \zeta_x dx + \zeta_y dy + \zeta_z dz \ . \end{aligned}$$

Or expressed in matrix notation

$$\begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \ . \tag{18}$$

The equivalent can easily be derived for the inverse transformation yielding

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} \ . \tag{19}$$

Comparing (18) and (19) the following relation can be identified

$$\begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}^{-1} \ . \tag{20}$$

We use the complement method to compute the inverse matrix

$$\bar{\bar{A}}^{-1} = \frac{1}{\left|\bar{\bar{A}}\right|} \left(\bar{\bar{K}}_{\bar{\bar{A}}}\right)^T \ , \tag{21}$$

where $\left|\bar{\bar{A}}\right|$ is called the Jacobian J of the transformation and $\bar{\bar{K}}_{\bar{\bar{A}}}$ is the complement matrix resulting when the elements in $\bar{\bar{A}}$ are substituted by there corresponding complements. For the Jacobian we have

$$\begin{aligned} \left|\bar{\bar{A}}\right| &= \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix} \\ &= x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi) \\ &= J \ . \end{aligned} \tag{22}$$

For the complement matrix we get

$$\begin{aligned} \bar{\bar{K}}_{\bar{\bar{A}}} &= \begin{bmatrix} \begin{vmatrix} y_\eta & y_\zeta \\ z_\eta & z_\zeta \end{vmatrix} & -\begin{vmatrix} y_\xi & y_\zeta \\ z_\xi & z_\zeta \end{vmatrix} & \begin{vmatrix} y_\xi & y_\eta \\ z_\xi & z_\eta \end{vmatrix} \\ -\begin{vmatrix} x_\eta & x_\zeta \\ z_\eta & z_\zeta \end{vmatrix} & \begin{vmatrix} x_\xi & x_\zeta \\ z_\xi & z_\zeta \end{vmatrix} & -\begin{vmatrix} x_\xi & x_\zeta \\ z_\xi & z_\zeta \end{vmatrix} \\ \begin{vmatrix} x_\eta & x_\zeta \\ y_\eta & y_\zeta \end{vmatrix} & -\begin{vmatrix} x_\xi & x_\zeta \\ y_\xi & y_\zeta \end{vmatrix} & \begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix} \end{bmatrix} \\ &= \begin{bmatrix} (y_\eta z_\zeta - y_\zeta z_\eta) & (y_\xi z_\zeta - y_\zeta z_\xi) & (y_\xi z_\eta - y_\eta z_\xi) \\ -(x_\eta y_\zeta - x_\zeta y_\eta) & (x_\xi z_\zeta - x_\zeta z_\xi) & -(x_\xi z_\eta - x_\eta z_\xi) \\ (x_\eta y_\zeta - x_\zeta y_\zeta) & -(x_\xi y_\zeta - x_\zeta y_\xi) & (x_\xi y_\eta - x_\eta y_\xi) \end{bmatrix} \ . \end{aligned} \tag{23}$$

Combining (21), (22) and (23) we get the following expressions for the metric terms

$$\begin{aligned}
\xi_x &= \frac{1}{J}(y_\eta z_\zeta - y_\zeta z_\eta) = \frac{1}{J}\alpha_{\xi x} \ , \\
\xi_y &= -\frac{1}{J}(x_\eta z_\zeta - x_\zeta z_\eta) = \frac{1}{J}\alpha_{\xi y} \ , \\
\xi_z &= \frac{1}{J}(x_\eta y_\zeta - x_\zeta y_\eta) = \frac{1}{J}\alpha_{\xi z} \ , \\
\eta_x &= -\frac{1}{J}(y_\xi z_\zeta - y_\zeta z_\xi) = \frac{1}{J}\alpha_{\eta x} \ , \\
\eta_y &= \frac{1}{J}(x_\xi z_\zeta - x_\zeta z_\xi) = \frac{1}{J}\alpha_{\eta y} \ , \\
\eta_z &= -\frac{1}{J}(x_\xi y_\zeta - x_\zeta y_\xi) = \frac{1}{J}\alpha_{\eta z} \ , \\
\zeta_x &= \frac{1}{J}(y_\xi z_\eta - y_\eta z_\xi) = \frac{1}{J}\alpha_{\zeta x} \ , \\
\zeta_y &= -\frac{1}{J}(x_\xi z_\eta - x_\eta z_\xi) = \frac{1}{J}\alpha_{\zeta y} \ , \\
\zeta_z &= \frac{1}{J}(x_\xi y_\eta - x_\eta y_\xi) = \frac{1}{J}\alpha_{\zeta z} \ . \qquad (24)
\end{aligned}$$

Examining the expressions for the α's in (24), taking $\alpha_{\xi x} = (y_\eta z_\zeta - y_\zeta z_\eta)$ as an example it is seen that $|y_\eta z_\zeta - y_\zeta z_\eta|$ is a differential area equal to the projection of the ξ-face on the x-plan of the Cartesian coordinate system.

Inserting the expressions for the metric terms (24) into (15) to (17) we get for the partial differentials

$$\frac{\partial}{\partial x} = \frac{1}{J}\left(\frac{\partial}{\partial \xi}\alpha_{\xi x} + \frac{\partial}{\partial \eta}\alpha_{\eta x} + \frac{\partial}{\partial \zeta}\alpha_{\zeta x}\right) \ , \qquad (25)$$

$$\frac{\partial}{\partial y} = \frac{1}{J}\left(\frac{\partial}{\partial \xi}\alpha_{\xi y} + \frac{\partial}{\partial \eta}\alpha_{\eta y} + \frac{\partial}{\partial \zeta}\alpha_{\zeta y}\right) \ , \qquad (26)$$

$$\frac{\partial}{\partial z} = \frac{1}{J}\left(\frac{\partial}{\partial \xi}\alpha_{\xi z} + \frac{\partial}{\partial \eta}\alpha_{\eta z} + \frac{\partial}{\partial \zeta}\alpha_{\zeta z}\right) \ . \qquad (27)$$

Instead of using the partial differential to transform the flow equations, we will derive an expression for the divergence of a vector $\vec{\nabla} \cdot \vec{F}$, which directly results in the strong conservation form of the flow equations when applied to the Cartesian versions.

The divergence operator $\vec{\nabla} \cdot \vec{F}$ with $\vec{F} = F_1(x,y,z)\vec{i} + F_2(x,y,z)\vec{j} + F_3(x,y,z)\vec{k}$ is defined as

$$\vec{\nabla} \cdot \vec{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z} \ .$$

Using the partial differentials (25), (26) and (27) we get

$$\begin{aligned}
\vec{\nabla} \cdot \vec{F} &= \frac{1}{J}\left(\frac{\partial F_1}{\partial \xi}\alpha_{\xi x} + \frac{\partial F_1}{\partial \eta}\alpha_{\eta x} + \frac{\partial F_1}{\partial \zeta}\alpha_{\zeta x}\right) \\
&+ \frac{1}{J}\left(\frac{\partial F_2}{\partial \xi}\alpha_{\xi y} + \frac{\partial F_2}{\partial \eta}\alpha_{\eta y} + \frac{\partial F_2}{\partial \zeta}\alpha_{\zeta y}\right) \\
&+ \frac{1}{J}\left(\frac{\partial F_3}{\partial \xi}\alpha_{\xi z} + \frac{\partial F_3}{\partial \eta}\alpha_{\eta z} + \frac{\partial F_3}{\partial \zeta}\alpha_{\zeta z}\right) \ .
\end{aligned}$$

Rewriting the terms according to

$$\frac{\partial F_1}{\partial \xi}\alpha_{\xi x} = \frac{\partial F_1 \alpha_{\xi x}}{\partial \xi} - F_1 \frac{\partial \alpha_{\xi x}}{\partial \xi} \ ,$$

and rearranging we get

$$\begin{aligned}
\vec{\nabla}\cdot\vec{F} &= \frac{1}{J}\left[\frac{\partial F_1\alpha_{\xi x}}{\partial\xi} + \frac{\partial F_1\alpha_{\eta x}}{\partial\eta} + \frac{\partial F_1\alpha_{\zeta x}}{\partial\zeta}\right. \\
&\qquad \left. - \left(F_1\frac{\partial\alpha_{\xi x}}{\partial\xi} + F_1\frac{\partial\alpha_{\eta x}}{\partial\eta} + F_1\frac{\partial\alpha_{\zeta x}}{\partial\zeta}\right)\right] \\
&+ \frac{1}{J}\left[\frac{\partial F_2\alpha_{\xi y}}{\partial\xi} + \frac{\partial F_2\alpha_{\eta y}}{\partial\eta} + \frac{\partial F_2\alpha_{\zeta y}}{\partial\zeta}\right. \\
&\qquad \left. - \left(F_2\frac{\partial\alpha_{\xi y}}{\partial\xi} + F_2\frac{\partial\alpha_{\eta y}}{\partial\eta} + F_2\frac{\partial\alpha_{\zeta y}}{\partial\zeta}\right)\right] \\
&+ \frac{1}{J}\left[\frac{\partial F_3\alpha_{\xi z}}{\partial\xi} + \frac{\partial F_3\alpha_{\eta z}}{\partial\eta} + \frac{\partial F_3\alpha_{\zeta z}}{\partial\zeta}\right. \\
&\qquad \left. - \left(F_3\frac{\partial\alpha_{\xi z}}{\partial\xi} + F_3\frac{\partial\alpha_{\eta z}}{\partial\eta} + F_3\frac{\partial\alpha_{\zeta z}}{\partial\zeta}\right)\right] \ .
\end{aligned}$$

Inserting the α's in the negative terms and reducing we get

$$\begin{aligned}
\vec{\nabla}\cdot\vec{F} &= \frac{1}{J}(F_1\alpha_{\xi x} + F_2\alpha_{\xi y} + F_3\alpha_{\xi z})_\xi \\
&+ \frac{1}{J}(F_1\alpha_{\eta x} + F_2\alpha_{\eta y} + F_3\alpha_{\eta z})_\eta \\
&+ \frac{1}{J}(F_1\alpha_{\zeta x} + F_2\alpha_{\zeta y} + F_3\alpha_{\zeta z})_\zeta \ . \qquad (28)
\end{aligned}$$

Now the tools, (25), (26), (27) and (28), necessary to transform the flow equation are ready.

3.3 Transformation of flow equations

To illustrate the transformation of the flow equations, we will go through the transformation of the U-momentum equation. In Cartesian coordinates the U-momentum has the form

$$\begin{aligned}
&\frac{\partial\rho U}{\partial t} + \frac{\partial\rho UU}{\partial x} + \frac{\partial\rho UV}{\partial y} + \frac{\partial\rho UW}{\partial z} = \frac{\partial}{\partial x}\left[2\mu\frac{\partial U}{\partial x}\right] \\
&\quad + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right)\right] - \frac{\partial P}{\partial x} + S_V \ . \qquad (29)
\end{aligned}$$

Time-dependent term

The time-dependent term is invariant under transformation, and needs no further attention.

Convection terms

The convective terms must be transformed, this is done by using the divergence operator.

$$\begin{aligned}
\frac{\partial \rho UU}{\partial x}+\frac{\partial \rho UV}{\partial y}+\frac{\partial \rho UW}{\partial z} &= \frac{1}{J}\left(\rho UU\alpha_{\xi x}+\rho UV\alpha_{\xi y}+\rho UW\alpha_{\xi z}\right)_{\xi} \\
&+ \frac{1}{J}\left(\rho UU\alpha_{\eta x}+\rho UV\alpha_{\eta y}+\rho UW\alpha_{\eta z}\right)_{\eta} \\
&+ \frac{1}{J}\left(\rho UU\alpha_{\zeta x}+\rho UV\alpha_{\zeta y}+\rho UW\alpha_{\zeta z}\right)_{\zeta} \\
&= \frac{1}{J}(C_1U)_{\xi}+\frac{1}{J}(C_2U)_{\eta}+\frac{1}{J}(C_3U)_{\zeta}\ , \qquad (30)
\end{aligned}$$

where the mass fluxes C_1, C_2 and C_3 are given by

$$\begin{aligned}
C_1 &= \rho U\alpha_{\xi x}+\rho V\alpha_{\xi y}+\rho W\alpha_{\xi z}\ , \\
C_2 &= \rho U\alpha_{\eta x}+\rho V\alpha_{\eta y}+\rho W\alpha_{\eta z}\ , \\
C_3 &= \rho U\alpha_{\zeta x}+\rho V\alpha_{\zeta y}+\rho W\alpha_{\zeta z}\ . \qquad (31)
\end{aligned}$$

Diffusion terms

Next we will look at the diffusive terms, again we will use the divergence operator

$$\begin{aligned}
&\frac{\partial}{\partial x}\left[2\mu\frac{\partial U}{\partial x}\right]+\frac{\partial}{\partial y}\left[\mu\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)\right] \\
&=\frac{1}{J}\left(\underbrace{2\mu\frac{\partial U}{\partial x}\alpha_{\xi x}+\mu\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\alpha_{\xi y}+\mu\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)\alpha_{\xi z}}_{a_1}\right)_{\xi} \\
&+\frac{1}{J}\left(\underbrace{2\mu\frac{\partial U}{\partial x}\alpha_{\eta x}+\mu\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\alpha_{\eta y}+\mu\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)\alpha_{\eta z}}_{a_2}\right)_{\eta} \\
&+\frac{1}{J}\left(\underbrace{2\mu\frac{\partial U}{\partial x}\alpha_{\zeta x}+\mu\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\alpha_{\zeta y}+\mu\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)\alpha_{\zeta z}}_{a_3}\right)_{\zeta}\ . \qquad (32)
\end{aligned}$$

Applying the partial differentials (25), (26) and (27) to the three under-braced terms starting with a_1 we get

$$\begin{aligned}
a_1 &= 2\mu\alpha_{\xi x}\frac{1}{J}\left(U_{\xi}\alpha_{\xi x}+U_{\eta}\alpha_{\eta x}+U_{\zeta}\alpha_{\zeta x}\right) \\
&+\mu\alpha_{\xi y}\left[\left(U_{\xi}\alpha_{\xi y}+U_{\eta}\alpha_{\eta y}+U_{\zeta}\alpha_{\zeta y}\right)\frac{1}{J}+\left(V_{\xi}\alpha_{\xi x}+V_{\eta}\alpha_{\eta x}+V_{\zeta}\alpha_{\zeta x}\right)\frac{1}{J}\right] \\
&+\mu\alpha_{\xi z}\left[\left(U_{\xi}\alpha_{\xi z}+U_{\eta}\alpha_{\eta z}+U_{\zeta}\alpha_{\zeta z}\right)\frac{1}{J}+\left(W_{\xi}\alpha_{\xi x}+W_{\eta}\alpha_{\eta x}+W_{\zeta}\alpha_{\zeta x}\right)\frac{1}{J}\right]\ .
\end{aligned}$$

Rearranging the terms we get

$$\begin{aligned}
a_1 &= \frac{\mu}{J}\left[\left(\alpha_{\xi x}\alpha_{\xi x}+\alpha_{\xi y}\alpha_{\xi y}+\alpha_{\xi z}\alpha_{\xi z}\right)U_{\xi}\right. \\
&+ \left.\left(\alpha_{\xi x}\alpha_{\eta x}+\alpha_{\xi y}\alpha_{\eta y}+\alpha_{\xi z}\alpha_{\eta z}\right)U_{\eta}\right.
\end{aligned}$$

$$
\begin{aligned}
&+ \quad (\alpha_{\xi x}\alpha_{\zeta x} + \alpha_{\xi y}\alpha_{\zeta y} + \alpha_{\xi z}\alpha_{\zeta z})\, U_\zeta \\
&+ \quad \alpha_{\xi x}\left(\alpha_{\xi x}U_\xi + \alpha_{\eta x}U_\eta + \alpha_{\zeta x}U_\zeta\right) \\
&+ \quad \alpha_{\xi y}\left(\alpha_{\xi x}V_\xi + \alpha_{\eta x}V_\eta + \alpha_{\zeta x}V_\zeta\right) \\
&+ \quad \alpha_{\xi z}\left(\alpha_{\xi x}W_\xi + \alpha_{\eta x}W_\eta + \alpha_{\zeta x}W_\zeta\right)] \\
&= \quad \frac{\mu}{J}\left(\beta_{11}U_\xi + \beta_{12}U_\eta + \beta_{13}U_\zeta + \alpha_{\xi x}\omega_{11} + \alpha_{\xi y}\omega_{21} + \alpha_{\xi z}\omega_{31}\right) \ , \qquad (33)
\end{aligned}
$$

where

$$
\begin{aligned}
\beta_{11} &= \alpha_{\xi x}\alpha_{\xi x} + \alpha_{\xi y}\alpha_{\xi y} + \alpha_{\xi z}\alpha_{\xi z} \ , \\
\beta_{12} &= \alpha_{\xi x}\alpha_{\eta x} + \alpha_{\xi y}\alpha_{\eta y} + \alpha_{\xi z}\alpha_{\eta z} \ , \\
\beta_{13} &= \alpha_{\xi x}\alpha_{\zeta x} + \alpha_{\xi y}\alpha_{\zeta y} + \alpha_{\xi z}\alpha_{\zeta z} \ , \\
\omega_{11} &= \alpha_{\xi x}U_\xi + \alpha_{\eta x}U_\eta + \alpha_{\zeta x}U_\zeta \ , \\
\omega_{21} &= \alpha_{\xi x}V_\xi + \alpha_{\eta x}V_\eta + \alpha_{\zeta x}V_\zeta \ , \\
\omega_{31} &= \alpha_{\xi x}W_\xi + \alpha_{\eta x}W_\eta + \alpha_{\zeta x}W_\zeta \ .
\end{aligned}
$$

The same procedure can be applied to a_2 and a_3 yielding

$$
a_2 = \frac{\mu}{J}\left(\beta_{21}U_\xi + \beta_{22}U_\eta + \beta_{23}U_\zeta + \alpha_{\eta x}\omega_{11} + \alpha_{\eta y}\omega_{21} + \alpha_{\eta z}\omega_{31}\right) \ , \qquad (34)
$$

where

$$
\begin{aligned}
\beta_{21} &= \alpha_{\eta x}\alpha_{\xi x} + \alpha_{\eta y}\alpha_{\xi y} + \alpha_{\eta z}\alpha_{\xi z} \ , \\
\beta_{22} &= \alpha_{\eta x}\alpha_{\eta x} + \alpha_{\eta y}\alpha_{\eta y} + \alpha_{\eta z}\alpha_{\eta z} \ , \\
\beta_{23} &= \alpha_{\eta x}\alpha_{\zeta x} + \alpha_{\eta y}\alpha_{\zeta y} + \alpha_{\eta z}\alpha_{\zeta z} \ .
\end{aligned}
$$

And

$$
a_3 = \frac{\mu}{J}\left(\beta_{31}U_\xi + \beta_{32}U_\eta + \beta_{33}U_\zeta + \alpha_{\zeta x}\omega_{11} + \alpha_{\zeta y}\omega_{21} + \alpha_{\zeta z}\omega_{31}\right) \ , \qquad (35)
$$

where

$$
\begin{aligned}
\beta_{31} &= \alpha_{\zeta x}\alpha_{\xi x} + \alpha_{\zeta y}\alpha_{\xi y} + \alpha_{\zeta z}\alpha_{\xi z} \ , \\
\beta_{32} &= \alpha_{\zeta x}\alpha_{\eta x} + \alpha_{\zeta y}\alpha_{\eta y} + \alpha_{\zeta z}\alpha_{\eta z} \ , \\
\beta_{33} &= \alpha_{\zeta x}\alpha_{\zeta x} + \alpha_{\zeta y}\alpha_{\zeta y} + \alpha_{\zeta z}\alpha_{\zeta z} \ .
\end{aligned}
$$

Pressure term

The last term to be transformed is the pressure term, this is simply done by using the equation for the partial differential

$$
\begin{aligned}
\frac{\partial P}{\partial x} &= \frac{1}{J}\left(\frac{\partial P}{\partial \xi}\alpha_{\xi x} + \frac{\partial P}{\partial \eta}\alpha_{\eta x} + \frac{\partial P}{\partial \zeta}\alpha_{\zeta x}\right) \\
&= \frac{1}{J}\left(\frac{\partial P\alpha_{\xi x}}{\partial \xi} - P\frac{\partial \alpha_{\xi x}}{\partial \xi} + \frac{\partial P\alpha_{\eta x}}{\partial \eta} - P\frac{\partial \alpha_{\eta x}}{\partial \eta} + \frac{\partial P\alpha_{\zeta x}}{\partial \zeta} - P\frac{\partial \alpha_{\zeta x}}{\partial \zeta}\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{J}\left(\frac{\partial P\alpha_{\xi x}}{\partial \xi}+\frac{\partial P\alpha_{\eta x}}{\partial \eta}+\frac{\partial P\alpha_{\zeta x}}{\partial \zeta}\right)-\frac{P}{J}\left(\frac{\partial \alpha_{\xi x}}{\partial \xi}+\frac{\partial \alpha_{\eta x}}{\partial \eta}+\frac{\partial \alpha_{\zeta x}}{\partial \zeta}\right)\\
&= \frac{1}{J}\left(\frac{\partial P\alpha_{\xi x}}{\partial \xi}+\frac{\partial P\alpha_{\eta x}}{\partial \eta}+\frac{\partial P\alpha_{\zeta x}}{\partial \zeta}\right) . \qquad (36)
\end{aligned}
$$

Volumetric source term

The volumetric source is invariant during the transformation and is left unchanged

Transformed equations

The remaining two momentum equations can be transformed in a similar way. The U-momentum equation is now assembled using (30) to (36) and multiplied by the Jacobian, which is time independent, and we get
U-momentum:

$$
\begin{aligned}
&\frac{\partial \rho JU}{\partial t}+\frac{\partial}{\partial \xi}\left(C_1U\right)+\frac{\partial}{\partial \eta}\left(C_2U\right)+\frac{\partial}{\partial \zeta}\left(C_3U\right)\\
&-\frac{\partial}{\partial \xi}\left[\frac{\mu}{J}\left(\beta_{11}U_\xi\right)\right]-\frac{\partial}{\partial \eta}\left[\frac{\mu}{J}\left(\beta_{22}U_\eta\right)\right]-\frac{\partial}{\partial \zeta}\left[\frac{\mu}{J}\left(\beta_{33}U_\zeta\right)\right]\\
&-\frac{\partial}{\partial \xi}\left[\frac{\mu}{J}\left(\beta_{12}U_\eta+\beta_{13}U_\zeta+\omega_{11}\alpha_{\xi x}+\omega_{21}\alpha_{\xi y}+\omega_{31}\alpha_{\xi z}\right)\right]\\
&-\frac{\partial}{\partial \eta}\left[\frac{\mu}{J}\left(\beta_{21}U_\xi+\beta_{23}U_\zeta+\omega_{11}\alpha_{\eta x}+\omega_{21}\alpha_{\eta y}+\omega_{31}\alpha_{\eta z}\right)\right]\\
&-\frac{\partial}{\partial \zeta}\left[\frac{\mu}{J}\left(\beta_{31}U_\xi+\beta_{32}U_\eta+\omega_{11}\alpha_{\zeta x}+\omega_{21}\alpha_{\zeta y}+\omega_{31}\alpha_{\zeta z}\right)\right]\\
&+\frac{\partial P\alpha_{\xi x}}{\partial \xi}+\frac{\partial P\alpha_{\eta x}}{\partial \eta}+\frac{\partial P\alpha_{\zeta x}}{\partial \zeta}=JS_V \qquad (37)
\end{aligned}
$$

V-momentum:

$$
\begin{aligned}
&\frac{\partial \rho JV}{\partial t}+\frac{\partial}{\partial \xi}\left(C_1V\right)+\frac{\partial}{\partial \eta}\left(C_2V\right)+\frac{\partial}{\partial \zeta}\left(C_3V\right)\\
&-\frac{\partial}{\partial \xi}\left[\frac{\mu}{J}\left(\beta_{11}V_\xi\right)\right]-\frac{\partial}{\partial \eta}\left[\frac{\mu}{J}\left(\beta_{22}V_\eta\right)\right]-\frac{\partial}{\partial \zeta}\left[\frac{\mu}{J}\left(\beta_{33}V_\zeta\right)\right]\\
&-\frac{\partial}{\partial \xi}\left[\frac{\mu}{J}\left(\beta_{12}V_\eta+\beta_{13}V_\zeta+\omega_{12}\alpha_{\xi x}+\omega_{22}\alpha_{\xi y}+\omega_{32}\alpha_{\xi z}\right)\right]\\
&-\frac{\partial}{\partial \eta}\left[\frac{\mu}{J}\left(\beta_{21}V_\xi+\beta_{23}V_\zeta+\omega_{12}\alpha_{\eta x}+\omega_{22}\alpha_{\eta y}+\omega_{32}\alpha_{\eta z}\right)\right]\\
&-\frac{\partial}{\partial \zeta}\left[\frac{\mu}{J}\left(\beta_{31}V_\xi+\beta_{32}V_\eta+\omega_{12}\alpha_{\zeta x}+\omega_{22}\alpha_{\zeta y}+\omega_{32}\alpha_{\zeta z}\right)\right]\\
&+\frac{\partial P\alpha_{\xi y}}{\partial \xi}+\frac{\partial P\alpha_{\eta y}}{\partial \eta}+\frac{\partial P\alpha_{\zeta y}}{\partial \zeta}=JS_V \qquad (38)
\end{aligned}
$$

W-momentum

$$
\begin{aligned}
&\frac{\partial \rho JW}{\partial t}+\frac{\partial}{\partial \xi}\left(C_1W\right)+\frac{\partial}{\partial \eta}\left(C_2W\right)+\frac{\partial}{\partial \zeta}\left(C_3W\right)\\
&-\frac{\partial}{\partial \xi}\left[\frac{\mu}{J}\left(\beta_{11}W_\xi\right)\right]-\frac{\partial}{\partial \eta}\left[\frac{\mu}{J}\left(\beta_{22}W_\eta\right)\right]-\frac{\partial}{\partial \zeta}\left[\frac{\mu}{J}\left(\beta_{33}W_\zeta\right)\right]\\
&-\frac{\partial}{\partial \xi}\left[\frac{\mu}{J}\left(\beta_{12}W_\eta+\beta_{13}W_\zeta+\omega_{13}\alpha_{\xi x}+\omega_{23}\alpha_{\xi y}+\omega_{33}\alpha_{\xi z}\right)\right]\\
&-\frac{\partial}{\partial \eta}\left[\frac{\mu}{J}\left(\beta_{21}W_\xi+\beta_{23}W_\zeta+\omega_{13}\alpha_{\eta x}+\omega_{23}\alpha_{\eta y}+\omega_{33}\alpha_{\eta z}\right)\right]
\end{aligned}
$$

$$-\frac{\partial}{\partial\zeta}\left[\frac{\mu}{J}\left(\beta_{31}W_\xi+\beta_{32}W_\eta+\omega_{13}\alpha_{\zeta x}+\omega_{23}\alpha_{\zeta y}+\omega_{33}\alpha_{\zeta z}\right)\right]$$
$$+\frac{\partial P\alpha_{\xi z}}{\partial\xi}+\frac{\partial P\alpha_{\eta z}}{\partial\eta}+\frac{\partial P\alpha_{\zeta z}}{\partial\zeta}=JS_V \tag{39}$$

where the following definitions have been used

$$\begin{aligned}
\beta_{11} &= \alpha_{\xi x}\alpha_{\xi x}+\alpha_{\xi y}\alpha_{\xi y}+\alpha_{\xi z}\alpha_{\xi z},\\
\beta_{12} &= \alpha_{\xi x}\alpha_{\eta x}+\alpha_{\xi y}\alpha_{\eta y}+\alpha_{\xi z}\alpha_{\eta z},\\
\beta_{13} &= \alpha_{\xi x}\alpha_{\zeta x}+\alpha_{\xi y}\alpha_{\zeta y}+\alpha_{\xi z}\alpha_{\zeta z},\\
\beta_{21} &= \alpha_{\eta x}\alpha_{\xi x}+\alpha_{\eta y}\alpha_{\xi y}+\alpha_{\eta z}\alpha_{\xi z},\\
\beta_{22} &= \alpha_{\eta x}\alpha_{\eta x}+\alpha_{\eta y}\alpha_{\eta y}+\alpha_{\eta z}\alpha_{\eta z},\\
\beta_{23} &= \alpha_{\eta x}\alpha_{\zeta x}+\alpha_{\eta y}\alpha_{\zeta y}+\alpha_{\eta z}\alpha_{\zeta z},\\
\beta_{31} &= \alpha_{\zeta x}\alpha_{\xi x}+\alpha_{\zeta y}\alpha_{\xi y}+\alpha_{\zeta z}\alpha_{\xi z},\\
\beta_{32} &= \alpha_{\zeta x}\alpha_{\eta x}+\alpha_{\zeta y}\alpha_{\eta y}+\alpha_{\zeta z}\alpha_{\eta z},\\
\beta_{33} &= \alpha_{\zeta x}\alpha_{\zeta x}+\alpha_{\zeta y}\alpha_{\zeta y}+\alpha_{\zeta z}\alpha_{\zeta z},\\
\omega_{11} &= \alpha_{\xi x}U_\xi+\alpha_{\eta x}U_\eta+\alpha_{\zeta x}U_\zeta,\\
\omega_{21} &= \alpha_{\xi x}V_\xi+\alpha_{\eta x}V_\eta+\alpha_{\zeta x}V_\zeta,\\
\omega_{31} &= \alpha_{\xi x}W_\xi+\alpha_{\eta x}W_\eta+\alpha_{\zeta x}W_\zeta,\\
\omega_{12} &= \alpha_{\xi y}U_\xi+\alpha_{\eta y}U_\eta+\alpha_{\zeta y}U_\zeta,\\
\omega_{22} &= \alpha_{\xi y}V_\xi+\alpha_{\eta y}V_\eta+\alpha_{\zeta y}V_\zeta,\\
\omega_{32} &= \alpha_{\xi y}W_\xi+\alpha_{\eta y}W_\eta+\alpha_{\zeta y}W_\zeta,\\
\omega_{13} &= \alpha_{\xi z}U_\xi+\alpha_{\eta z}U_\eta+\alpha_{\zeta z}U_\zeta,\\
\omega_{23} &= \alpha_{\xi z}V_\xi+\alpha_{\eta z}V_\eta+\alpha_{\zeta z}V_\zeta,\\
\omega_{33} &= \alpha_{\xi z}W_\xi+\alpha_{\eta z}W_\eta+\alpha_{\zeta z}W_\zeta.
\end{aligned}$$

3.4 Closure

Comparison of the transformed equation (37) with the Cartesian version (29) reveals the transformation has introduced extra terms into the momentum equations. For a Cartesian mesh it is easily seen that the transformed equation (37) reduces to the Cartesian version (29).

As can be seen that the flux terms have become more complicated, the so-called cross diffusion terms have arisen from the diffusion terms, and the pressure term has expanded into three pressure terms. Even though a lot of extra terms have been introduced, the discretization of the individual terms in the transformed equation is not more complicated than the discretization of the individual terms in the Cartesian equations.

The discretization should therefore not cause any special problems and will be given in the following chapter.

4 Discretization of the flow equations

The present derivation of the discrete version of the governing differential equations is based on the finite volume technique. The continuous fields are approximated by the values at discrete points in space and time. The discrete points in space are often referred to as mesh or grid points, and the spacing between the discrete time levels at which the solution exists are called the time steps.

To obtain the difference equation following the finite volume methodology the solution domain is divided into a number of non-overlapping control volumes each holding a single mesh point, and the differential equations are integrated over each of these control volumes and over the finite time step.

The assumption regarding the variation of the variables between the mesh points and the time levels necessary to evaluate the integrals will be discussed along the process of obtaining the integral equation.

Two important aspects of the finite volume method will be mentioned here. First, the method of integrating the differential equation to obtain the discrete equations is easy to understand, facilitating the interpretation of the physical meaning of the resulting terms.

Secondly, and maybe most importantly, when the differential equations are expressed in conservation-law form the discrete equations obtained using the finite volume method will fulfill the conservation principle for each of the finite control volumes. Ensuring that the flux over a cell face is uniquely determined for two cells sharing a common cell face, also the sum over the individual control volumes constituting the total solution domain will fulfill the conservation principle. As a consequence, the solution will always exhibit physical behaviour, irrespective of how coarse a mesh is used. The solution on a coarse mesh may not be very accurate, but as the coarse solution reflects the overall behaviour of the solution, this information can be used to construct a finer mesh for an improved calculation.

4.1 Desired properties of the finite volume equation

Before carrying on with the derivation of the finite volume equation some considerations of the desired properties of the resulting equation will be appropriated.

The resulting equation system will be solved sequentially by the SIMPLE procedure of Patankar and Spalding [43] using standard iterative solvers for linear problems. To allow this, the equations must at least fulfill two conditions. The equations for U, V, W, P etc. must be decoupled, and any nonlinearity must be removed. We will revert to the linearization in connection with the treatment of the nonlinear convective terms. The form of the resulting algebraic equation can be written in the generic way

$$A_P\phi_P + \sum A_{nb}\phi_{nb} = S \,, \tag{40}$$

where A_P is the centre-node coefficient, $\sum A_{nb}\phi_{nb}$ is the influence of the neighbouring nodes on the P-node, and S is a source. Besides being linear the equations must possess the following properties:

- conservativeness
- boundedness
- transportiveness

Conservativeness

Conservativeness means that accumulation of a species in the domain equals the inflow minus the outflow of the domain plus the production within the domain. The fulfilment of this was ensured when transforming the Cartesian flow equations by keeping the differential equation on a strong conservative form. In the discrete analog (40) everything except the volume sources is associated with the cell faces, meaning that the strong conservative form can be retained by ensuring that the fluxes at the cell faces are uniquely represented for two cells sharing a cell face. In this way all internal fluxes will be cancelled in pairs, leaving only the fluxes at the domain boundary and the volume sources.

Boundedness

Boundedness means that in the absence of volume sources, the values in the interior of the domain should be bounded by the values at the boundary of the domain. In this connection the instationary term can be viewed as a volume source. It can be shown that this requirement will be fulfilled [26] if,

$$\mid A_P \mid > \sum \mid A_{nb} \mid \quad . \tag{41}$$

The inequality can be relaxed to an equality in all points except one. As $A_P = \sum -A_{nb}$ for all inner points, it is clear that all neighbouring coefficients must possess the same sign to fulfill the relaxed requirement[3]. When boundary conditions are applied, the unrelaxed requirement will be fulfilled at all boundary points with Dirichlet conditions.

This criterion is identical to that of Scarborough, stating that a sufficient condition for convergence of the Gauss-Seidel method for an iterative solution of linear algebraic equation systems is the fulfilment of (41). This means that when ensuring the solution to be bounded we also ensure convergence of at least one iterative method.

Transportiveness

Transportiveness means that the discrete equation must reflect the parabolic nature of the fluid flow. For a stagnant fluid with constant diffusivity a point source will spread equally in all directions. If a velocity is added, the spreading will be shifted into direction of the velocity. Using the Peclet number, $Pe = \rho U L/\Gamma$ where L is a characteristic length and Γ the diffusivity, expressing the ratio of convection to diffusion, the transportiveness can be illustrated as a series of ellipses, see Fig. 1. For $Pe = 0$ the spreading occurs equally in all directions. As the Peclet number is raised the spreading is shifted into the downstream direction. In the limit of an infinite Peclet number the ellipse will collapse into a streak line.

The technique of making the resulting finite volume equation reflect the transportiveness of the flow will be addressed in a later section of this chapter dealing with differencing schemes.

4.2 Geometric layout

To discretize the flow equation in 3D, the physical domain is subdivided into finite volumes. Here the subdivision will be restricted to structured grids composed of hexahedrons (quadrangles in 2D). The flow variables will be stored in the centre of these computational cells, see Fig. 2, where the cell centre is defined as the

[3] Here all neighbour coefficients are chosen to be negative.

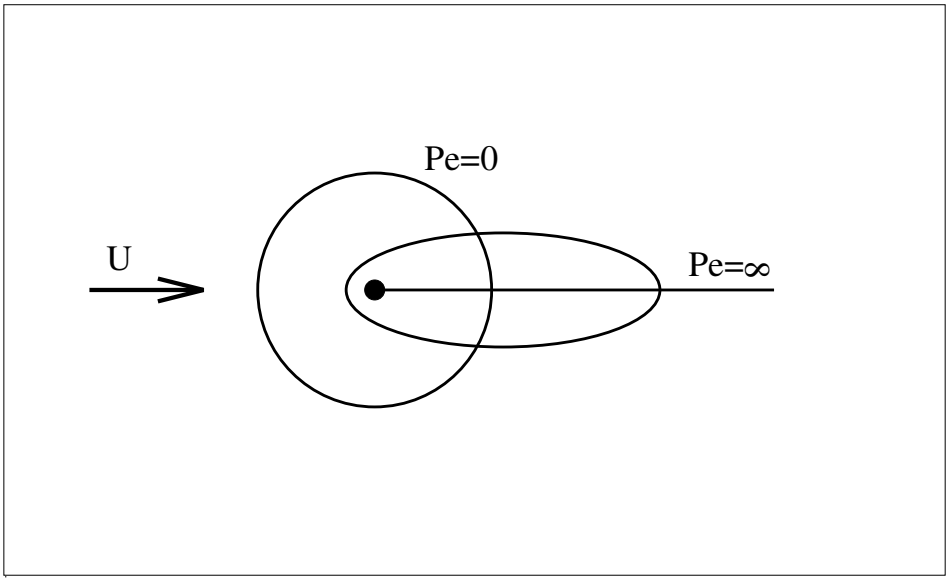

Figure 1. Transportiveness of the fluid illustrated by the influence of the Peclet number on the spreading of a point source. For $Pe = 0$ the spreading occurs equally in all directions. As the Peclet number is raised the spreading is shifted into the downstream direction. In the limit of an infinite Peclet number the ellipse will collapse into a streak line.

point where the coordinates equal 1/8 of the sum of the coordinates of the cell vertices. We will use capital letter for the cell centres, lower-case letters for the cell vertices, the midpoints of cell faces and cell edges.

Here the phrase structured grid will cover a grid where the individual cells have well defined neighbours, 26 in 3D (8 in 2D). Using extended compass notation in three dimensions the neighbours are, 6 face neighbours N, S, E, W, T and B, 12 edge neighbours NW, NE, SE, S, TW, TE, BE, BW, TS, TN, BN, BS, and 8 vertex neighbours TNW, TNE, TSE, TS, BNW, BNE, BSE, BS, see Fig. 3.

4.3 Cell-face values

In the process of obtaining the finite-volume equations we will need to evaluate gradients and values at the cell faces.

As the fluid properties and variables are stored at the cell centres, the most obvious way of obtaining cell-face values is to use linear interpolation between cell centres. For an east cell-face we have

$$\phi_e = \frac{1}{2}\left(\phi_P + \phi_E\right) \ ,$$

using one half for the linear interpolation factor, see Fig. 10. The viscosity, density, and the pressure at the cell face will be calculated in this way, and the value will be assumed to be constant over the cell face.

For the convected velocities in the convection terms this treatment will sometimes result in erroneous results, and an alternative treatment must be devised. The discussion of this problem will be left to be treated in connection with the evaluation of the the convection terms.

For the gradient evaluation we will distinguish between two types of gradients, normal gradients and cross gradients. The reason for this is purely computational, and is dictated by the storing of the variables.

The normal gradients are the $\partial/\partial\xi$ at ξ-faces, $\partial/\partial\eta$ at η-faces, and $\partial/\partial\zeta$ at ζ-faces where a ξ-face is a face with the normal pointing in the ξ-direction in

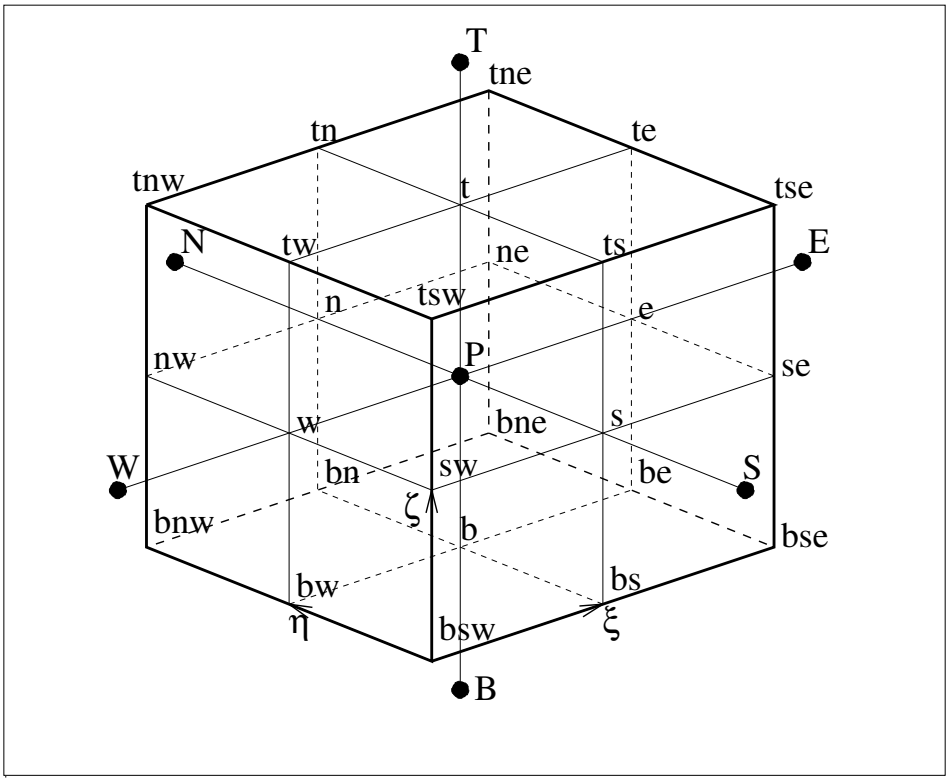

Figure 2. A single finite-volume cell around point P. The point P is located in the geometric centre of the cell. The cell centres of the normal neighbours are indicated in capital letters, the cell face positions are indicated in lower-case letters, using an extended compass notation.

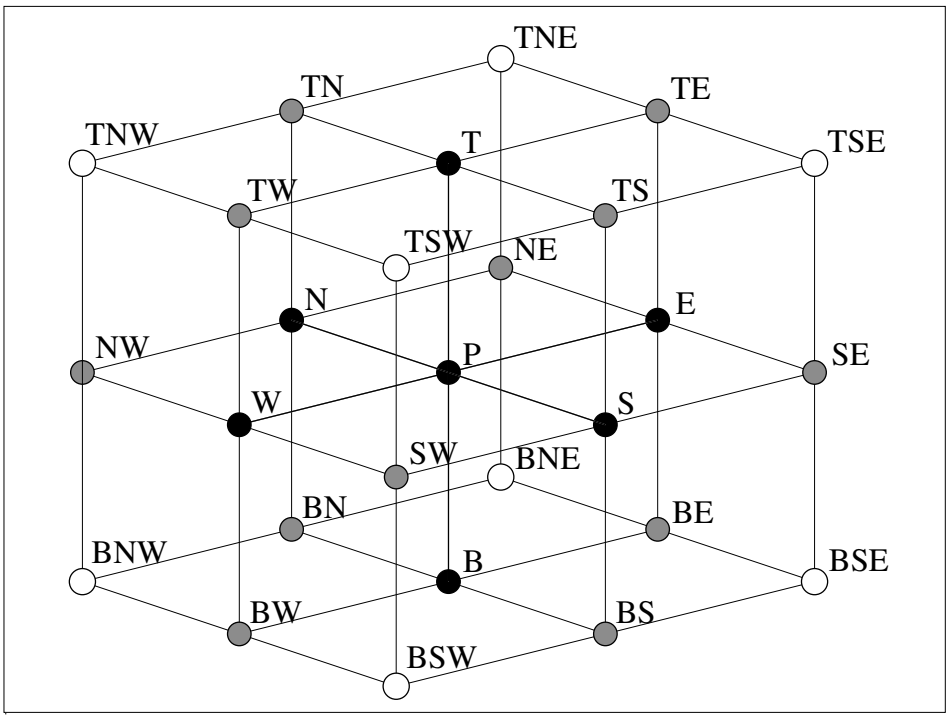

Figure 3. Cell centres of finite-volume cells in a structured grid, P is the centre of the present cell, and the neighbours are named using an extended compass notation.

computational space. All normal gradients can be computed using the second-order accurate central differences between cell centres, using an east face (ξ-face) as an example

$$\left(\frac{\partial\phi}{\partial\xi}\right)_e = \frac{\phi_E - \phi_P}{\Delta\xi} = \phi_E - \phi_P \ .$$

For the cross-term gradients, these are $\partial/\partial\eta$, $\partial/\partial\zeta$ at ξ-faces, $\partial/\partial\xi$, $\partial/\partial\zeta$ at η-faces, and $\partial/\partial\xi$, $\partial/\partial\eta$ at ζ-faces the direct application of the central differences between cell centres is not possible. Instead the gradients are computed as interpolation between two central differences, again using the east cell face as an example

$$\begin{aligned}\left(\frac{\partial\phi}{\partial\eta}\right)_e &= \frac{1}{2}\left[\left(\frac{\partial\phi}{\partial\eta}\right)_P + \left(\frac{\partial\phi}{\partial\eta}\right)_E\right] \\ &= \frac{1}{2}\left[\frac{\phi_N - \phi_S}{2\Delta\eta} + \frac{\phi_{NE} - \phi_{SE}}{2\Delta\eta}\right] \\ &= \frac{1}{4}\left[(\phi_N - \phi_S) + (\phi_{NE} - \phi_{SE})\right] \ , \end{aligned} \tag{42}$$

$$\begin{aligned}\left(\frac{\partial\phi}{\partial\zeta}\right)_e &= \frac{1}{2}\left[\left(\frac{\partial\phi}{\partial\zeta}\right)_P + \left(\frac{\partial\phi}{\partial\zeta}\right)_E\right] \\ &= \frac{1}{4}\left[(\phi_T - \phi_B) + (\phi_{TE} - \phi_{BE})\right] \ . \end{aligned} \tag{43}$$

The necessary geometric information, the metrics or cofactors at cell faces, the interpolation factors, and the calculation of the cell volumes are described in Chapter 8 dealing with geometric quantities.

4.4 Integration of flow equations

Having decided the geometric layout and how to evaluate the cell-face values and gradients, we are now able to integrate the flow equation over a finite-volume cell. The U-momentum equation will be used as an example of the integration process. In the following the integration over space and time will be indicated as

$$\int d\mathrm{Vol}\, dt \equiv \int_t^{t+\Delta t}\int_b^t\int_s^n\int_w^e d\xi\, d\eta\, d\zeta\, dt \ .$$

The integration of the U-momentum equation (37) can now be written as

$$\begin{aligned}&\int \frac{\partial \rho J U}{\partial t} d\mathrm{Vol}\, dt \\ &+ \int \frac{\partial}{\partial\xi}\left(C_1 U\right) d\mathrm{Vol}\, dt - \int \frac{\partial}{\partial\xi}\left[\frac{\mu}{J}\left(\beta_{11}U_\xi\right)\right] d\mathrm{Vol}\, dt \\ &+ \int \frac{\partial}{\partial\eta}\left(C_2 U\right) d\mathrm{Vol}\, dt - \int \frac{\partial}{\partial\eta}\left[\frac{\mu}{J}\left(\beta_{22}U_\eta\right)\right] d\mathrm{Vol}\, dt \\ &+ \int \frac{\partial}{\partial\zeta}\left(C_3 U\right) d\mathrm{Vol}\, dt - \int \frac{\partial}{\partial\zeta}\left[\frac{\mu}{J}\left(\beta_{33}U_\zeta\right)\right] d\mathrm{Vol}\, dt \\ &- \int \frac{\partial}{\partial\xi}\left[\frac{\mu}{J}\left(\beta_{12}U_\eta + \beta_{13}U_\zeta + \omega_{11}\alpha_{\xi x} + \omega_{21}\alpha_{\xi y} + \omega_{31}\alpha_{\xi z}\right)\right] d\mathrm{Vol}\, dt \\ &- \int \frac{\partial}{\partial\eta}\left[\frac{\mu}{J}\left(\beta_{21}U_\xi + \beta_{23}U_\zeta + \omega_{11}\alpha_{\eta x} + \omega_{21}\alpha_{\eta y} + \omega_{31}\alpha_{\eta z}\right)\right] d\mathrm{Vol}\, dt \\ &- \int \frac{\partial}{\partial\zeta}\left[\frac{\mu}{J}\left(\beta_{31}U_\xi + \beta_{32}U_\eta + \omega_{11}\alpha_{\zeta x} + \omega_{21}\alpha_{\zeta y} + \omega_{31}\alpha_{\zeta z}\right)\right] d\mathrm{Vol}\, dt\end{aligned}$$

$$+\int \frac{\partial P\alpha_{\xi x}}{\partial \xi} d\mathrm{Vol}\, dt + \int \frac{\partial P\alpha_{\eta x}}{\partial \eta} d\mathrm{Vol}\, dt + \int \frac{\partial P\alpha_{\zeta x}}{\partial \zeta} d\mathrm{Vol}\, dt$$
$$= \int S_V d\mathrm{Vol}\, dt \ . \tag{44}$$

The individual terms of the U-momentum equation will now be treated, starting with the instationary term.

Instationary term

The instationary term ρJU is assumed to be constant over the cell volume so the integration simply yields

$$\begin{aligned}\int \frac{\partial \rho JU}{\partial t} d\mathrm{Vol}\, dt & \\ &= (\rho JU \Delta\xi\Delta\eta\Delta\zeta)_P^{t+\Delta t} - (\rho JU \Delta\xi\Delta\eta\Delta\zeta)_P^t \\ &= \rho_P J_P U_P^{t+\Delta t} - \rho_P J_P U_P^t \ , \end{aligned} \tag{45}$$

where J_P is the volume of the computational cell, and $\Delta\xi$, $\Delta\eta$ and $\Delta\zeta = 1$ for a computational cell. No interpolation is necessary as everything is evaluated in the cell centre.

Convective terms

Looking at the convective terms, these being the second, fourth and sixth term in (44) we get

$$\begin{aligned}\int \frac{\partial}{\partial \xi}(C_1 U)\, d\mathrm{Vol}\, dt &= \Delta t \left[\left(C_1 U^{t+\Delta t}\right)_e - \left(C_1 U^{t+\Delta t}\right)_w \right] \\ &= (I_e^c - I_w^c)\, \Delta t \ , \end{aligned} \tag{46}$$

where the $^{t+\Delta t}$ indicates that the values have been evaluated at the end of the time step treating these fully implicitly.

An analog treatment of the two remaining convective terms in (44) results in

$$\begin{aligned}\int \frac{\partial}{\partial \eta}(C_2 U)\, d\mathrm{Vol}\, dt &= \Delta t \left[\left(C_2 U^{t+\Delta t}\right)_n - \left(C_2 U^{t+\Delta t}\right)_s \right] \\ &= (I_n^c - I_s^c)\, \Delta t \ , \end{aligned} \tag{47}$$

and

$$\begin{aligned}\int \frac{\partial}{\partial \eta}(C_3 U)\, d\mathrm{Vol}\, dt &= \Delta t \left[\left(C_3 U^{t+\Delta t}\right)_t - \left(C_3 U^{t+\Delta t}\right)_b \right] \\ &= (I_t^c - I_b^c)\, \Delta t \ . \end{aligned} \tag{48}$$

Taking the convective momentum fluxes (I^c), we see that these terms are non-linear caused by the velocity entering in second power. To get a linear algebraic equation, these terms must be linearized. This is accomplished by taking the velocities going into the mass flux terms (C_1, C_2 and C_3) from the previous time step.

Failing to evaluate all quantities at the end of the time step, we violate the fully implicit time step assumption, and this is one of the reasons why in the beginning of the chapter we stated that the time marching scheme was semi-implicit.

We now have, using the convective flux at the east face as an example

$$C_{1e} = \rho_e U_e^t (\alpha_{\xi x})_e + \rho_e V_e^t (\alpha_{\xi y})_e + \rho_e W_e^t (\alpha_{\xi z})_e \ ,$$

where the ^{t} indicates that these velocities are evaluated at the start of the time step.

The calculation of the α's is described in section 8.3, the densities and velocities going into the mass flux could be linearly interpolated between cell centres as described earlier in this chapter.

As the mass flux is calculated in connection with enforcement of the continuity, this value will be used instead of performing the interpolation just described. Further details are left until the treatment of the pressure correction equation.

Making the solution bounded and fulfilling the transportive nature of the flow, care must be taken when evaluating the convected quantity at the cell face. This is discussed in the section dealing with difference schemes.

Diffusive fluxes

Taking the diffusive coefficients we will again violate the fully implicit time stepping by treating the cross-diffusive part of the flux explicit, using old values.

This explicit term will be treated as a source term and to distinguish it from volume sources it will be called a false source term.

The explicit treatment of the cross diffusive part of the diffusive terms can be allowed because in most cases these terms are small. When the mesh is orthogonal the cross terms are identically zero, and for small departures from orthogonality the terms will be small. When care is taken to minimize the degree of nonorthogonality when generating meshes, the explicit implementation of cross terms has been shown to work quite well both for steady-state and transient calculations [47], [45], and [30].

Normal diffusion terms

Taking the normal diffusive term, i.e. the third, fifth and seventh term in (44), still using the fully implicit evaluation we get

$$\begin{aligned}
&-\int \frac{\partial}{\partial \xi}\left[\frac{\mu}{J}\left(\beta_{11} U_\xi\right)\right] d\mathrm{Vol}\, dt \\
&\quad = -\Delta t \left[\left(\frac{\mu}{J}\beta_{11} U_\xi^{t+\Delta t}\right)_e - \left(\frac{\mu}{J}\beta_{11} U_\xi^{t+\Delta t}\right)_w\right] \\
&\quad = \left(I_e^{dn} - I_w^{dn}\right)\Delta t \ , \qquad (49) \\
&-\int \frac{\partial}{\partial \eta}\left[\frac{\mu}{J}\left(\beta_{22} U_\eta\right)\right] d\mathrm{Vol}\, dt \\
&\quad = -\Delta t \left[\left(\frac{\mu}{J}\beta_{22} U_\eta^{t+\Delta t}\right)_n - \left(\frac{\mu}{J}\beta_{22} U_\eta^{t+\Delta t}\right)_s\right] \\
&\quad = \left(I_n^{dn} - I_s^{dn}\right)\Delta t \ , \qquad (50)
\end{aligned}$$

and

$$\begin{aligned}
&-\int \frac{\partial}{\partial \zeta}\left[\frac{\mu}{J}\left(\beta_{33} U_\xi\right)\right] d\mathrm{Vol}\, dt \\
&\quad = -\Delta t \left[\left(\frac{\mu}{J}\beta_{33} U_\zeta^{t+\Delta t}\right)_t - \left(\frac{\mu}{J}\beta_{33} U_\zeta^{t+\Delta t}\right)_b\right] \\
&\quad = \left(I_t^{dn} - I_b^{dn}\right)\Delta t \ . \qquad (51)
\end{aligned}$$

For the normal diffusive terms (I^{dn}), using the east face term as an example, we have

$$I_e^{dn} = \left(\frac{-\mu}{J} \beta_{11} U_\xi^{t+\Delta t} \right)_e .$$

The α's going into the β_{11} are all evaluated according to the description in Chapter 8, and the β can be assembled from these as

$$(\beta_{11})_e = (\alpha_{\xi x}\alpha_{\xi x})_e + (\alpha_{\xi y}\alpha_{\xi y})_e + (\alpha_{\xi z}\alpha_{\xi z})_e \ .$$

Using central differentiation for the velocity gradient as earlier described and linear interpolation to obtain J_e and μ_e we have

$$I_e^{dn} = \frac{\frac{1}{2}(\mu_P + \mu_E)}{\frac{1}{2}(J_P + J_E)} (\beta_{11})_e \left(U_E^{t+\Delta t} - U_P^{t+\Delta t} \right)$$

$$I_e^{dn} = A_E^{dn} \left(U_E^{t+\Delta t} - U_P^{t+\Delta t} \right) \ ,$$

where

$$A_E^{dn} = -\frac{\mu_e}{J_e}(\beta_{11})_e \ .$$

$$A_W^{dn} = -\frac{\mu_w}{J_w}(\beta_{11})_w \ .$$

$$A_N^{dn} = -\frac{\mu_n}{J_n}(\beta_{21})_n \ .$$

$$A_S^{dn} = -\frac{\mu_s}{J_s}(\beta_{21})_s \ .$$

$$A_T^{dn} = -\frac{\mu_t}{J_t}(\beta_{31})_t \ .$$

$$A_B^{dn} = -\frac{\mu_b}{J_b}(\beta_{31})_b \ .$$

$$A_P^{dn} = -A_W^{dn} - A_E^{dn} - A_S^{dn} - A_N^{dn} - A_B^{dn} - A_T^{dn} \ .$$

As a result of our explicit treatment of the cross diffusion terms, the part of the influence coefficient resulting from diffusion will always be negative, and thereby fulfill the boundedness requirement.

Cross-diffusion terms

For the cross-diffusive terms, the eight, ninth, and tenth terms in (44), the fully implicit treatment will be violated, and the term will simply be evaluated at the start of the time step.

$$\begin{aligned}
&-\int \frac{\partial}{\partial \xi} \left[\frac{\mu}{J} \left(\beta_{12} U_\eta + \beta_{13} U_\zeta + \omega_{11}\alpha_{\xi x} + \omega_{21}\alpha_{\xi y} + \omega_{31}\alpha_{\xi z} \right) \right] d\mathrm{Vol}\, dt \\
&\quad = -\Delta t \left[\left(\frac{\mu}{J} \left(\beta_{12} U_\eta^t + \beta_{13} U_\zeta^t + \omega_{11}^t \alpha_{\xi x} + \omega_{21}^t \alpha_{\xi y} + \omega_{31}^t \alpha_{\xi z} \right) \right)_e \right. \\
&\qquad \left. + \left(\frac{\mu}{J} \left(\beta_{12} U_\eta^t + \beta_{13} U_\zeta^t + \omega_{11}^t \alpha_{\xi x} + \omega_{21}^t \alpha_{\xi y} + \omega_{31}^t \alpha_{\xi z} \right) \right)_w \right] \\
&\quad = (I_e^{dc} - I_w^{dc}) \Delta t \ , \qquad (52)
\end{aligned}$$

$$\begin{aligned}
&-\int \frac{\partial}{\partial \eta} \left[\frac{\mu}{J} \left(\beta_{21} U_\xi + \beta_{23} U_\zeta + \omega_{11}\alpha_{\eta x} + \omega_{21}\alpha_{\eta y} + \omega_{31}\alpha_{\eta z} \right) \right] d\mathrm{Vol}\, dt \\
&\quad = -\Delta t \left[\left(\frac{\mu}{J} \left(\beta_{21} U_\xi^t + \beta_{23} U_\zeta^t + \omega_{11}^t \alpha_{\eta x} + \omega_{21}^t \alpha_{\eta y} + \omega_{31}^t \alpha_{\eta z} \right) \right)_n \right. \\
&\qquad \left. + \left(\frac{\mu}{J} \left(\beta_{21} U_\xi^t + \beta_{23} U_\zeta^t + \omega_{11}^t \alpha_{\eta x} + \omega_{21}^t \alpha_{\eta y} + \omega_{31}^t \alpha_{\eta z} \right) \right)_s \right] \\
&\quad = (I_n^{dc} - I_s^{dc}) \Delta t \ , \qquad (53)
\end{aligned}$$

and

$$
\begin{aligned}
&-\int \frac{\partial}{\partial\zeta}\left[\frac{\mu}{J}\left(\beta_{31}U_\xi+\beta_{32}U_\eta+\omega_{11}\alpha_{\zeta x}+\omega_{21}\alpha_{\zeta y}+\omega_{31}\alpha_{\zeta z}\right)\right] d\mathrm{Vol}\,dt \\
&\quad=-\Delta t\left[\left(\frac{\mu}{J}\left(\beta_{31}U^t_\xi+\beta_{32}U^t_\eta+\omega^t_{11}\alpha_{\zeta x}+\omega^t_{21}\alpha_{\zeta y}+\omega^t_{31}\alpha_{\zeta z}\right)\right)_t\right. \\
&\qquad\left.+\left(\frac{\mu}{J}\left(\beta_{31}U^t_\xi+\beta_{32}U^t_\eta+\omega^t_{11}\alpha_{\zeta x}+\omega^t_{21}\alpha_{\zeta y}+\omega^t_{31}\alpha_{\zeta z}\right)\right)_b\right] \\
&\quad=(I^{dc}_t-I^{dc}_b)\Delta t\ . \qquad (54)
\end{aligned}
$$

Again taking the east face as an example, we have

$$
I^{dc}_e=\left(\frac{\mu}{J}\left(\beta_{12}U^t_\eta+\beta_{13}U^t_\zeta+\omega^t_{11}\alpha_{\xi x}+\omega^t_{21}\alpha_{\xi y}+\omega^t_{31}\alpha_{\xi z}\right)\right)_e\ ,
$$

where the viscosity, cell volume, and the β's can be evaluated in the same way as the normal diffusive terms. The normal derivatives of the velocity will be derived according to the practice earlier described.

The expressions for the α's going into the the ω terms are calculated according to the practice described in Chapter 8, and the method earlier described will be used to obtain the cross derivatives of the velocity. For the ω's at the east face this results in the following expressions

$$
\begin{aligned}
(\omega_{11})_e&=(\alpha_{\xi x})_e\left(U^t_E-U^t_P\right)+(\alpha_{\eta x})_e\frac{1}{4}\left[\left(U^t_N-U^t_S\right)+\left(U^t_{NE}-U^t_{SE}\right)\right] \\
&\quad+(\alpha_{\zeta x})_e\frac{1}{4}\left[\left(U^t_T-U^t_B\right)+\left(U^t_{TE}-U^t_{BE}\right)\right]\ , \\
(\omega_{21})_e&=(\alpha_{\xi x})_e\left(V^t_E-V^t_P\right)+(\alpha_{\eta x})_e\frac{1}{4}\left[\left(V^t_N-V^t_S\right)+\left(V^t_{NE}-V^t_{SE}\right)\right] \\
&\quad+(\alpha_{\zeta x})_e\frac{1}{4}\left[\left(V^t_T-V^t_B\right)+\left(V^t_{TE}-V^t_{BE}\right)\right]\ ,
\end{aligned}
$$

and

$$
\begin{aligned}
(\omega_{31})_e&=(\alpha_{\xi x})_e\left(W^t_E-W^t_P\right)+(\alpha_{\eta x})_e\frac{1}{4}\left[\left(W^t_N-W^t_S\right)+\left(W^t_{NE}-W^t_{SE}\right)\right] \\
&\quad+(\alpha_{\zeta x})_e\frac{1}{4}\left[\left(W^t_T-W^t_B\right)+\left(W^t_{TE}-W^t_{BE}\right)\right]\ .
\end{aligned}
$$

As the cross-diffusive terms are treated explicitly, there is no reason for splitting them into influence coefficients, instead the six cross-diffusive fluxes are moved to the right-hand side of the equation and assembled into a false source term S_F, the resulting false source term gets the form

$$
S_F=I^{dc}_w-I^{dc}_e+I^{dc}_s-I^{dc}_n+I^{dc}_b-I^{dc}_t\ .
$$

Pressure terms

The pressure terms, the eleventh, twelfth and thirteenth term in (40) will also be treated explicitly, in order to make the momentum equation and pressure equation decoupled

$$
\begin{aligned}
\int\frac{\partial P\alpha_{\xi x}}{\partial\xi}d\mathrm{Vol}\,dt&=\Delta t\left[\left(P^t\alpha_{\xi x}\right)_e-\left(P^t\alpha_{\xi x}\right)_w\right] \\
&=\left(I^p_e-I^p_w\right)\Delta t\ , \qquad (55)
\end{aligned}
$$

$$
\begin{aligned}
\int\frac{\partial P\alpha_{\eta x}}{\partial\eta}d\mathrm{Vol}\,dt&=\Delta t\left[\left(P^t\alpha_{\eta x}\right)_n-\left(P^t\alpha_{\eta x}\right)_s\right] \\
&=\left(I^p_n-I^p_s\right)\Delta t\ , \qquad (56)
\end{aligned}
$$

$$\int \frac{\partial P \alpha_{\zeta x}}{\partial \zeta} d\mathrm{Vol}\, dt = \Delta t \left[\left(P^t \alpha_{\zeta x} \right)_t - \left(P^t \alpha_{\zeta x} \right)_b \right]$$
$$= \left(I_t^p - I_b^p \right) \Delta t \ . \tag{57}$$

Taking the east face as an example to show how the quantities are evaluated we get

$$I_e^p = \left(P^t \alpha_{\xi x} \right)_e \ .$$

The α's are evaluated at the cell face where they are used, see Chapter 8 and no interpolation is needed. For the pressure at the cell faces linear interpolation is used resulting in the following expression for the pressure force at the east wall

$$I_e^p = \frac{1}{2} \left(P_P^t + P_E^t \right) \left(\alpha_{\xi x} \right)_e \ .$$

As the pressure terms are treated explicitly there is no reason for splitting them into influence coefficients, instead the pressure terms are moved to the right-hand side of the equation and are assembled into a pressure source

$$S_P = I_w^p - I_e^p + I_s^p - I_n^p + I_b^p - I_t^p \ .$$

Volumetric source term

Finally, the volumetric source will be treated, this is the last term in (40), and this term will also be treated explicitly by evaluating it a the start of the time step.

$$\int J \, S_V \, d\mathrm{Vol}\, dt = J_P S_V^t \Delta t \ . \tag{58}$$

As was the case with the instationary term, no difficulties are encountered as everything is located in the cell centre just where it is needed.

4.5 Differencing schemes

Before the final algebraic equation can be assembled, the evaluation of the convected velocities in the convective terms has to be addressed. Here four different schemes for treating these velocities will be discussed, the second-order accurate Central Differencing Scheme, the first-order accurate Upwind Differencing Scheme, the Second-order accurate Upwind Scheme, and the third-order accurate QUICK scheme.

A technique making the higher-order schemes more stable, recasting them into a upwind formulation plus a source term will be described, see Yeo et al. [64].

Central Differencing Scheme (CDS)

A simple way of performing the interpolation of the quantity to the cell face is by linear interpolation between the adjoining cell centres, resulting in the Central Differencing Scheme. In terms of Taylor series expansions, the CDS is second-order accurate. For an east cell face we have

$$\phi_e = \frac{1}{2} \left(\phi_E + \phi_P \right) \ .$$

Similar expressions can be derived for the remaining cell faces. Inserting these interpolated cell-face values into (46-48) and rearranging we get

$$
\begin{aligned}
A_W^c &= -\frac{1}{2}C_w \ , \\
A_E^c &= \frac{1}{2}C_e \ , \\
A_S^c &= -\frac{1}{2}C_s \ , \\
A_N^c &= \frac{1}{2}C_n \ , \\
A_B^c &= -\frac{1}{2}C_b \ , \\
A_T^c &= \frac{1}{2}C_t \ , \\
A_P^c &= -A_W^c - A_E^c - A_S^c - A_N^c - A_B^c - A_T^c \ . \qquad (59)
\end{aligned}
$$

Some of these coefficients are always positive, and when the flow is convectively dominated $\mid A^c \mid > \mid A^{dn} \mid$, this will result in positive influence coefficients $A = A^c + A^{dn} > 0$ leading to unbounded solutions. As downstream and upstream information have the same weight in the CDS, the scheme is unable to reflect the transportiveness of the flow as $Pe \to \infty$, resulting in unbounded solutions and even divergence using an iterative solver.

Upwind Differencing Scheme (UDS)

The UDS do not have the boundedness problem of the CDS, this is achieved by assuming the cell-face value to be equal to the value at the cell centre in the upstream grid direction. In terms of Taylor series expansions the UDS is only first-order accurate. For an east cell face we have

$$
\phi_e = \begin{cases} \phi_P & \text{if } C_e \geq 0 \\ \phi_E & \text{if } C_e \leq 0 \end{cases} \ .
$$

Similar expressions can be derived for the remaining cell faces. Inserting into (46-48) and rearranging we get

$$
\begin{aligned}
A_W^c &= -\max\{0, \quad C_w\} \ , \\
A_E^c &= -\max\{0, -C_e\} \ , \\
A_S^c &= -\max\{0, \quad C_s\} \ , \\
A_N^c &= -\max\{0, -C_n\} \ , \\
A_B^c &= -\max\{0, \quad C_b\} \ , \\
A_T^c &= -\max\{0, -C_t\} \ , \\
A_P^c &= -A_W^c - A_E^c - A_S^c - A_N^c - A_B^c - A_T^c \ . \qquad (60)
\end{aligned}
$$

As no positive coefficients arise, the scheme has no boundedness problems.

The price paid for getting rid of the boundedness problems is the introduction of false (or numerical) diffusion.

False diffusion is a multidimensional phenomenon. False diffusion occurs when the flow is oblique to the grid lines and where there is a nonzero gradient of the dependent variable in the direction normal to the flow [42].

The reason of the false diffusion is the practice of interpolating in the grid direction instead of the flow direction, de Vahl and Mallison [8] give the following approximate expression for the false diffusion

$$
\Gamma_{false} = \frac{\rho U \, \Delta x \Delta y \sin 2\theta}{4(\Delta y \sin^3 \theta + \Delta x \cos 3\theta)} \ ,
$$

where θ is the angel between the U-velocity and the x-direction, Δx and Δy are the grid spacings. It is seen that the false diffusion is most serious when $\theta = 45°$ and absent when $\theta = 0°$.

Second-order Upwind Scheme (SUDS)

A more accurate upwind scheme can be constructed if two upstream points are used instead of one to determine the cell face value. The resulting scheme is second order accurate in terms of Taylor series expansions.

A straightforward implementation of this scheme would yield two major problems, first the computational molecule would expand from a seven-point molecule to a thirteen point-molecule in three dimensions, and secondly positive coefficients would arise violating the boundedness criterion.

In order to circumvent these problems, the scheme will be put into an upwind form, treating it as a pure upwind scheme plus a source term following [64]. In this case we only have a seven point molecule, and no positive influence coefficients arise.

Using two upstream points we get the following expression for the east cell value

$$\phi_e = \begin{cases} \phi_P + \frac{1}{2}(\phi_P - \phi_W) & \text{if } C_e \geq 0 \\ \phi_E + \frac{1}{2}(\phi_E - \phi_{EE}) & \text{if } C_e \leq 0 \end{cases} ,$$

The flux at the east face can now be written as

$$\begin{aligned} C_e\phi_e &= \max\{0, C_e\}\phi_P - \max\{0, -C_e\}\phi_E \\ &+ \frac{1}{2}\left[\max\{0, C_e\}(\phi_P - \phi_W) - \max\{0, -C_e\}(\phi_E - \phi_{EE})\right] , \end{aligned}$$

where the last term will be treated explicitly and moved to the right-hand side of the equation in order to get a upwind formulation. Similar expressions can be derived for the remaining faces.

Inserting these flux expressions into (46-48) and rearranging we get

$$\begin{aligned} A_W^c &= -\max\{0, \ C_w\} , \\ A_E^c &= -\max\{0, -C_e\} , \\ A_S^c &= -\max\{0, \ C_s\} , \\ A_N^c &= -\max\{0, -C_n\} , \\ A_B^c &= -\max\{0, \ C_b\} , \\ A_T^c &= -\max\{0, -C_t\} , \\ A_P^c &= -A_W^c - A_E^c - A_S^c - A_N^c - A_B^c - A_T^c , \\ S_{SUDS} &= \frac{1}{2}(\max\{0, C_w\}(\phi_W - \phi_{WW}) - \max\{0, -C_w\}(\phi_P - \phi_E)) \\ &\quad -\frac{1}{2}(\max\{0, C_e\}(\phi_P - \phi_W) - \max\{0, -C_e\}(\phi_E - \phi_{EE})) \\ &\quad +\frac{1}{2}(\max\{0, C_s\}(\phi_S - \phi_{SS}) - \max\{0, -C_s\}(\phi_P - \phi_N)) \\ &\quad -\frac{1}{2}(\max\{0, C_n\}(\phi_P - \phi_S) - \max\{0, -C_n\}(\phi_N - \phi_{NN})) \\ &\quad +\frac{1}{2}(\max\{0, C_b\}(\phi_B - \phi_{BB}) - \max\{0, -C_b\}(\phi_P - \phi_T)) \\ &\quad -\frac{1}{2}(\max\{0, C_t\}(\phi_P - \phi_B) - \max\{0, -C_t\}(\phi_T - \phi_{TT})) , \quad (61) \end{aligned}$$

where S_{SUDS} is a term added to the source term on the right-hand side of (40). The same practice can be applied to the CDS, so the scheme would become bounded. But even though the scheme becomes bounded, the scheme will still lack the transportiveness of the upwind schemes.

Quadratic Upstream Interpolation for Convection Kinematics (QUICK)

The third-order accurate QUICK scheme of [23], is based on an upstream shifted quadratic interpolation. This means performing the interpolation of the face value, using a parabola fitted through the two nearest upstream points and the nearest downstream point.

For an east face value this practice result in the following expressions

$$\phi_e = \begin{cases} \frac{1}{2}(\phi_P + \phi_E) - \frac{1}{8}(\phi_W + \phi_E - 2\phi_P) & \text{if } C_e \geq 0 \\ \frac{1}{2}(\phi_E + \phi_P) - \frac{1}{8}(\phi_P + \phi_{EE} - 2\phi_E) & \text{if } C_e \leq 0 \end{cases} .$$

Looking at the expressions, the scheme can be seen as a CDS plus a stabilizing term proportional to the curvature of the fitted parabola. The scheme includes downstream information. Accordingly, it does not fulfill the requirement of transportiveness, as do the UDS and SUDS. The influense of the downstream point is weaker than that of the CDS, as the scheme possesses a clear upstream weighting.

The direct application of the QUICK scheme will pose the same problems as were discussed in connection with the SUDS scheme. The resulting computational molecule includes thirteen points, and the scheme will generate positive influence coefficients. Again the cure for these problems is the same as in the case of SUDS, the scheme will be put into an upwind formulation resulting in the following expression for the east face flux

$$\begin{aligned} C_e\phi_e &= \max\{0, C_e\}\,\phi_P - \max\{0, -C_e\}\,\phi_E \\ &+ \frac{1}{8}\left[\max\{0, C_e\}\,(3\phi_E - 2\phi_P - \phi_W)\right. \\ &- \left.\max\{0, -C_e\}\,(3\phi_P - 2\phi_E - \phi_{EE})\right] , \end{aligned}$$

where the last term will be treated explicitly and put on the right-hand side of the equation in order to obtain the upwind formulation. Similar expressions can be derived for the remaining faces. Inserting these flux expressions into (46-48) and rearranging we get

$$\begin{aligned} A_W^c &= -\max\{0, \quad C_w\} , \\ A_E^c &= -\max\{0, -C_e\} , \\ A_S^c &= -\max\{0, \quad C_s\} , \\ A_N^c &= -\max\{0, -C_n\} , \\ A_B^c &= -\max\{0, \quad C_b\} , \\ A_T^c &= -\max\{0, -C_t\} , \\ A_P^c &= -A_W^c - A_E^c - A_S^c - A_N^c - A_B^c - A_T^c , \\ S_{QUICK} &= \frac{1}{8}\left[\max\{0, \quad C_w\}\,(3\phi_P - 2\phi_W - \phi_{WW})\right. \\ &\quad \left. - \max\{0, -C_w\}\,(3\phi_W - 2\phi_P - \phi_E)\right] \\ &\quad -\frac{1}{8}\left[\max\{0, \quad C_e\}\,(3\phi_E - 2\phi_P - \phi_W)\right. \\ &\quad \left. + \max\{0, -C_e\}\,(3\phi_P - 2\phi_E - \phi_{EE})\right] \end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{8}\left[\max\left\{0, \quad C_s\right\}\left(3\phi_P - 2\phi_S - \phi_{SS}\right)\right. \\
&\quad \left. - \max\left\{0, -C_s\right\}\left(3\phi_S - 2\phi_P - \phi_N\right)\right] \\
&-\frac{1}{8}\left[\max\left\{0, \quad C_n\right\}\left(3\phi_N - 2\phi_P - \phi_S\right)\right. \\
&\quad \left. + \max\left\{0, -C_n\right\}\left(3\phi_P - 2\phi_N - \phi_{NN}\right)\right] \\
&+\frac{1}{8}\left[\max\left\{0, \quad C_b\right\}\left(3\phi_P - 2\phi_B - \phi_{BB}\right)\right. \\
&\quad \left. - \max\left\{0, -C_b\right\}\left(3\phi_B - 2\phi_P - \phi_T\right)\right] \\
&-\frac{1}{8}\left[\max\left\{0, \quad C_t\right\}\left(3\phi_T - 2\phi_P - \phi_B\right)\right. \\
&\quad \left. + \max\left\{0, -C_t\right\}\left(3\phi_P - 2\phi_T - \phi_{TT}\right)\right] \ . \qquad (62)
\end{aligned}
$$

4.6 Finite volume equation

Now the components can be assembled to give the finite-volume equation (40). For the U-momentum equation, this results in

$$
\begin{aligned}
&A_P U_P^{t+\Delta t} + A_W U_W^{t+\Delta t} + A_E U_E^{t+\Delta t} + A_S U_S^{t+\Delta t} + A_N U_N^{t+\Delta t} \\
&\quad + A_B U_B^{t+\Delta t} + A_T U_T^{t+\Delta t} = S_T + S_V + S_F + S_P + S_C \ , \qquad (63)
\end{aligned}
$$

where the following definitions have been used

$$
\begin{aligned}
A_W &= A_W^{dn} + A_W^c \ , \\
A_E &= A_E^{dn} + A_E^c \ , \\
A_S &= A_S^{dn} + A_S^c \ , \\
A_N &= A_N^{dn} + A_N^c \ , \\
A_B &= A_B^{dn} + A_B^c \ , \\
A_T &= A_T^{dn} + A_T^c \ , \\
A_P &= -A_W - A_E - A_S - A_N - A_B - A_T + \frac{\rho_P J_P}{\Delta t} \ , \\
S_T &= \frac{\rho_P J_P U_P^t}{\Delta t}, \\
S_V &= 0 \ , \\
S_F &= -I_e^{dc} + I_w^{dc} - I_n^{dc} + I_s^{dc} - I_t^{dc} + I_b^{dc} \ , \\
S_P &= -I_e^P + I_w^P - I_n^P + I_s^P - I_t^P + I_b^P \ , \\
S_C &= \text{explicitly treated terms for convection.} \qquad (64)
\end{aligned}
$$

The generalization of the remaining momentum equations and the transport equations for k and ϵ is straightforward and will not be given here. The resulting equations can be found in Appendix A.

4.7 Closure

Looking at the discretized U-momentum equation we see that it fulfills all of the requirements stated in the beginning of the chapter. The equation is decoupled from the remaining equations, the equation is linear and fulfills the requirement of conservativeness, boundedness and transportiveness.

Starting with the conservative property of the equation, this was ensured by use of the strong conservation law form of the transformed equation as a basis for the finite-volume derivation of the discrete equation.

The decoupling of the equation from the remaining equation was obtained by evaluating all variables except the U-velocity at the start of the time step. This was done for the pressure source S_P and the V and W velocities entering the mass flux.

The explicit evaluation of the convective terms ensured not only the decoupling of the momentum equations, but it also ensured the resulting equation to be linear, which is essential as the equations will be solved using techniques from linear algebra.

The explicit treatment of the cross-diffusional terms and the S_C term resulting when the various difference schemes were cast into upwind form ensured the equation to be bounded. As already mentioned this criterion often referred to as the Scarborough criterion is also a sufficient condition to guarantee that at least the Gauss-Seidel iteration will converge.

Apart from the CDS scheme all the schemes possesses clear upwind weighting, meaning that except when using the CDS, the resulting equation will reflect the transportiveness of the flow.

As mentioned the generalization of this procedure to the remaining equations is straightforward. For the momentum equations for V and W all terms are identical with the U-momentum equation terms except for the explicit treated source terms, S_T, S_V, S_F, S_P, and S_C. In the rest of this report the five parts of the source term will gathered in a single term called $S_{U-\mathrm{mom}}$, $S_{V-\mathrm{mom}}$, and $S_{W-\mathrm{mom}}$ respectivley for the three momentum equations. For points adjacent to boundaries also the central coefficient A_P will differ for the three momentum equations, when boundary conditions are included. In fact this can be used when making the computer code, meaning that we have to calculate and store only a single set of the neighbouring coefficients A_W, A_E, A_S, A_N, A_B, A_T and use these for all three momentum equations.

5 Pressure equation

5.1 Introduction

Working with incompressible flow, no equation of state exists for the pressure. The standard practice for incompressible codes is to derive an equation for the pressure by combining the continuity equation with the momentum equations.

The momentum equations are used as a predictor to advance the solution in time. In general the resulting flow field will not fulfill the continuity equation, and the rewritten continuity equation working on the pressure is used as a corrector making the predicted flow field satisfy the continuity constraint.

5.2 Cell-face velocities

To introduce the pressure into the discretized continuity equation, the cell-face velocities will be expressed by the momentum equations just as in the standard staggered approach. For the cell centre the velocities are given by

$$\begin{aligned} U_P &= \frac{S_{U-\mathrm{mom}} - \sum A_{nb} U_{nb}}{A_{P,U}} , \\ V_P &= \frac{S_{V-\mathrm{mom}} - \sum A_{nb} V_{nb}}{A_{P,V}} , \\ W_P &= \frac{S_{W-\mathrm{mom}} - \sum A_{nb} W_{nb}}{A_{P,W}} . \end{aligned} \tag{65}$$

Where the three source terms, $S_{U-\mathrm{mom}}$, $S_{V-\mathrm{mom}}$, and $S_{W-\mathrm{mom}}$ includes all the explicitly treated terms.

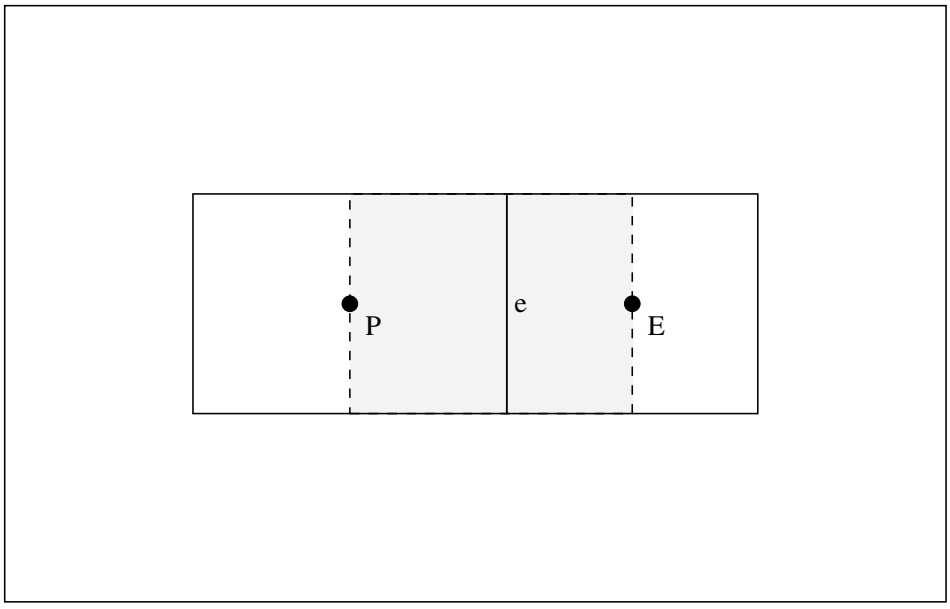

Figure 4. The figure shows the staggered cell used for calculating the cell-face velocities, the staggered cell is moved half a cell length in order to be centred around the cell face.

For a staggered cell as shown in Fig. 4, here the east face velocities will be used as an example, the cell face velocities U_e, V_e and W_e for the non-staggered cell can

be expressed by the momentum equations for the staggered cells. The coefficients and variables referring to the staggered cell are indicated with a ^{S}.

$$
\begin{aligned}
U_e &= \frac{S^S_{U-\mathrm{mom}} - \sum A^S_{nb} U^S_{nb}}{A^S_{P,U}} \ , \\
V_e &= \frac{S^S_{V-\mathrm{mom}} - \sum A^S_{nb} V^S_{nb}}{A^S_{P,V}} \ , \\
W_e &= \frac{S^S_{W-\mathrm{mom}} - \sum A^S_{nb} W^S_{nb}}{A^S_{P,W}} \ . \qquad (66)
\end{aligned}
$$

As the coefficients and variables for the staggered cell momentum equation do not exist, a practice for obtaining them must be devised. The most obvious would be to use linear interpolation between the momentum equations for the neighbouring cells. For an east face this would be an interpolation between the present cell and the eastern cell neighbour. This practice would lead to the well-known odd even pressure decoupling, as only every second pressure node figures in the resulting equation see Patankar [42].

A remedy to overcome this problem is the Rhie/Chow interpolation, Rhie [47], using linear interpolation for all terms except for the pressure source. Instead the pressure source is centred around the cell face, giving the following expressions for the cell border velocities

$$
\begin{aligned}
U_e &= \overline{\left(\frac{S^-_{U-\mathrm{mom}} - \sum A_{nb} U_{nb}}{A_{P,U}} \right)}_e \\
&+ \overline{\left(\frac{1}{A_{P,U}} \right)}_e \left[(\alpha_{\xi x})_e (P_E - P_P) \right. \\
&+ \left. (\alpha_{\eta x})_e (P_{ne} - P_{se}) + (\alpha_{\zeta x})_e (P_{te} - P_{be}) \right] \ , \\
V_e &= \overline{\left(\frac{S^-_{V-\mathrm{mom}} - \sum A_{nb} V_{nb}}{A_{P,V}} \right)}_e \\
&+ \overline{\left(\frac{1}{A_{P,V}} \right)}_e \left[(\alpha_{\xi y})_e (P_E - P_P) \right. \\
&+ \left. (\alpha_{\eta y})_e (P_{ne} - P_{se}) + (\alpha_{\zeta y})_e (P_{te} - P_{be}) \right] \ , \\
W_e &= \overline{\left(\frac{S^-_{W-\mathrm{mom}} - \sum A_{nb} W_{nb}}{A_{P,W}} \right)}_e \\
&+ \overline{\left(\frac{1}{A_{P,W}} \right)}_e \left[(\alpha_{\xi z})_e (P_E - P_P) \right. \\
&+ \left. (\alpha_{\eta z})_e (P_{ne} - P_{se}) + (\alpha_{\zeta z})_e (P_{te} - P_{be}) \right] \ , \qquad (67)
\end{aligned}
$$

where the overlined terms are calculated by linear interpolation between the cell centres on each side of the cell border. For the east cell face we interpolated between the influence coefficients for the present cell P and the east cell E. The source terms, $S^-_{U-\mathrm{mom}}$, $S^-_{V-\mathrm{mom}}$, and $S^-_{w-\mathrm{mom}}$ includes all explicit treated terms except the pressure sources. For the remaining cell faces the generalization of (67) is straightforward.

5.3 SIMPLE algorithm

The momentum and pressure equations will be used in a predictor corrector fashion.

Predictor step

The pressure at the new time step (or the new iteration in case of a steady-state calculation) is found by assuming it to be unchanged from the previous time step, i.e. $P^{'} = P$. In the following the predicted variables will be indicated by a $'$. Solution of the momentum equations using the pressure P' gives the predicted velocity field at the cell centre

$$\begin{aligned} U_P^{'} &= \frac{S_{U-\mathrm{mom}} - \sum A_{nb} U_{nb}^{'}}{A_{P,U}} \ , \\ V_P^{'} &= \frac{S_{V-\mathrm{mom}} - \sum A_{nb} V_{nb}^{'}}{A_{P,V}} \ , \\ W_P^{'} &= \frac{S_{W-\mathrm{mom}} - \sum A_{nb} W_{nb}^{'}}{A_{P,W}} \ . \end{aligned} \tag{68}$$

The predicted velocity at the cell face is given by (67), using an east face as an example

$$\begin{aligned} U_e^{'} &= \overline{\left(\frac{S_{U-\mathrm{mom}}^{-} - \sum A_{nb} U_{nb}^{'}}{A_{P,U}} \right)}_e \\ &+ \overline{\left(\frac{1}{A_{P,U}} \right)}_e \Big[(\alpha_{\xi x})_e \left(P_E^{'} - P_P^{'} \right) \\ &+ (\alpha_{\eta x})_e \left(P_{ne}^{'} - P_{se}^{'} \right) + (\alpha_{\zeta x})_e \left(P_{te}^{'} - P_{be}^{'} \right) \Big] \ , \\ V_e^{'} &= \overline{\left(\frac{S_{V-\mathrm{mom}}^{-} - \sum A_{nb} V_{nb}^{'}}{A_{P,V}} \right)}_e \\ &+ \overline{\left(\frac{1}{A_{P,V}} \right)}_e \Big[(\alpha_{\xi y})_e \left(P_E^{'} - P_P^{'} \right) \\ &+ (\alpha_{\eta y})_e \left(P_{ne}^{'} - P_{se}^{'} \right) + (\alpha_{\zeta y})_e \left(P_{te}^{'} - P_{be}^{'} \right) \Big] \ , \\ W_e^{'} &= \overline{\left(\frac{S_{W-\mathrm{mom}}^{-} - \sum A_{nb} W_{nb}^{'}}{A_{P,W}} \right)}_e \\ &+ \overline{\left(\frac{1}{A_{P,W}} \right)}_e \Big[(\alpha_{\xi z})_e \left(P_E^{'} - P_P^{'} \right) \\ &+ (\alpha_{\eta z})_e \left(P_{ne}^{'} - P_{se}^{'} \right) + (\alpha_{\zeta z})_e \left(P_{te}^{'} - P_{be}^{'} \right) \Big] \ . \end{aligned} \tag{69}$$

Inserting the cell-face velocities on the form of (69) into the equation for the east cell face flux having the form of (31), we get

$$C_e^{'} = \rho_e \Bigg[(\alpha_{\xi x})_e \Bigg\{ \overline{\left(\frac{S_{U-\mathrm{mom}}^{-} - \sum A_{nb} U_{nb}^{'}}{A_{P,U}} \right)}_e$$

$$
\begin{aligned}
&+ \quad \overline{\left(\frac{1}{A_{P,U}}\right)}_e \Big((\alpha_{\xi x})_e (P'_E - P'_P) \\
&+ \quad (\alpha_{\eta x})_e \frac{1}{4}((P'_N - P'_S) + (P'_{NE} - P'_{SE})) \\
&+ \quad (\alpha_{\zeta x})_e \frac{1}{4}((P'_T - P'_B) + (P'_{TE} - P'_{BE})) \Big) \Big\} \\
&+ \quad (\alpha_{\xi y})_e \Bigg\{ \overline{\left(\frac{S^-_{V-\mathrm{mom}} - \sum A_{nb} V'_{nb}}{A_{P,V}}\right)}_e \\
&+ \quad \overline{\left(\frac{1}{A_{P,V}}\right)}_e \Big((\alpha_{\xi y})_e (P'_E - P'_P) \\
&+ \quad (\alpha_{\eta y})_e \frac{1}{4}((P'_N - P'_S) + (P'_{NE} - P'_{SE})) \\
&+ \quad (\alpha_{\zeta y})_e \frac{1}{4}((P'_T - P'_B) + (P'_{TE} - P'_{BE})) \Big) \Big\} \\
&+ \quad (\alpha_{\xi z})_e \Bigg\{ \overline{\left(\frac{S^-_{W-\mathrm{mom}} - \sum A_{nb} W'_{nb}}{A_{P,W}}\right)}_e \\
&+ \quad \overline{\left(\frac{1}{A_{P,W}}\right)}_e \Big((\alpha_{\xi z})_e (P'_E - P'_P) \\
&+ \quad (\alpha_{\eta z})_e \frac{1}{4}((P'_N - P'_S) + (P'_{NE} - P'_{SE})) \\
&+ \quad (\alpha_{\zeta z})_e \frac{1}{4}((P'_T - P'_B) + (P'_{TE} - P'_{BE})) \Big) \Big\} \Big] \ .
\end{aligned}
$$

The cross-term pressure gradients in the previous derivation have been approximated by expressions of the following kind

$$
P_{ne} - P_{se} = \frac{1}{4}((P_N - P_S) + (P_{NE} - P_{SE})) \ , \tag{70}
$$

and linear interpolation was used to calculate ρ_e.

Generalization of the expression for the cell-face flux (70) to the remaining cell faces is straightforward. Inserting the cell-face fluxes into the discrete continuity equation the mass deficit, S_{mass}, can be calculated, namely

$$
S_{\mathrm{mass}} = C'_e - C'_w + C'_n - C'_s + C'_t - C'_b \ . \tag{71}
$$

As indicated the predicted velocities will in general not fulfill the continuity equation.

Corrector step

Now continuity is enforced by adjusting the pressure to get a correct cell-face mass flux. The cell-face velocity fulfilling the continuity equation is given by

$$
\begin{aligned}
U''_e \quad &= \quad \overline{\left(\frac{S^-_{U-\mathrm{mom}} - \sum A_{nb} U''_{nb}}{A_{P,U}}\right)}_e \\
&+ \quad \overline{\left(\frac{1}{A_{P,U}}\right)}_e \left[(\alpha_{\xi x})_e \left(P''_E - P''_P \right) \right.
\end{aligned}
$$

$$
\begin{aligned}
&+ \quad (\alpha_{\eta x})_e \left(P''_{ne} - P''_{se}\right) + (\alpha_{\zeta x})_e \left(P''_{te} - P''_{be}\right)\Big] \ , \\
V''_e &= \overline{\left(\frac{S^-_{V-\mathrm{mom}} - \sum A_{nb} V''_{nb}}{A_{P,V}}\right)_e} \\
&+ \overline{\left(\frac{1}{A_{P,V}}\right)_e} \Big[(\alpha_{\xi y})_e \left(P''_E - P''_P\right) \\
&+ (\alpha_{\eta y})_e \left(P''_{ne} - P''_{se}\right) + (\alpha_{\zeta y})_e \left(P''_{te} - P''_{be}\right)\Big] \ , \\
W''_e &= \overline{\left(\frac{S^-_{W-\mathrm{mom}} - \sum A_{nb} W''_{nb}}{A_{P,W}}\right)_e} \\
&+ \overline{\left(\frac{1}{A_{P,W}}\right)_e} \Big[(\alpha_{\xi z})_e \left(P''_E - P''_P\right) \\
&+ (\alpha_{\eta z})_e \left(P''_{ne} - P''_{se}\right) + (\alpha_{\zeta z})_e \left(P''_{te} - P''_{be}\right)\Big] \ . \qquad (72)
\end{aligned}
$$

where the velocities and corresponding pressure fulfilling the continuity constraint are indicated by $''$. The following relationship exists between the predicted ($'$) and the corrected ($''$) variables

$$
\begin{aligned}
U'' &= U' + U^c \ , \\
V'' &= V' + V^c \ , \\
W'' &= W' + W^c \ , \\
P'' &= P' + P^c \ . \qquad (73)
\end{aligned}
$$

Subtracting the equation for the predicted cell-face velocities (69) from the equation for the corrected cell-face velocities (72) and using (73) yields

$$
\begin{aligned}
U^c_e &= \overline{\left(\frac{-\sum A_{nb} U^c_{nb}}{A_{P,U}}\right)_e} + \overline{\left(\frac{1}{A_{P,U}}\right)_e} \Big[(\alpha_{\xi x})_e \left(P^c_E - P^c_P\right) \\
&+ (\alpha_{\eta x})_e \left(P^c_{ne} - P^c_{se}\right) + (\alpha_{\zeta x})_e \left(P^c_{te} - P^c_{be}\right)\Big] \ , \\
V^c_e &= \overline{\left(\frac{-\sum A_{nb} V^c_{nb}}{A_{P,V}}\right)_e} + \overline{\left(\frac{1}{A_{P,V}}\right)_e} \Big[(\alpha_{\xi y})_e \left(P^c_E - P^c_P\right) \\
&+ (\alpha_{\eta y})_e \left(P^c_{ne} - P^c_{se}\right) + (\alpha_{\zeta y})_e \left(P^c_{te} - P^c_{be}\right)\Big] \ , \\
W^c_e &= \overline{\left(\frac{-\sum A_{nb} W^c_{nb}}{A_{P,W}}\right)_e} + \overline{\left(\frac{1}{A_{P,W}}\right)_e} \Big[(\alpha_{\xi z})_e \left(P^c_E - P^c_P\right) \\
&+ (\alpha_{\eta z})_e \left(P^c_{ne} - P^c_{se}\right) + (\alpha_{\zeta z})_e \left(P^c_{te} - P^c_{be}\right)\Big] \ .
\end{aligned}
$$

The first term on the right-hand sides of the former equations $\overline{(-\sum A_{nb} U^c_{nb}/A_P)}$ etc. will be dropped, this is done in order to allow a decoupled solution method to be used for the flow equations. Keeping the terms $\overline{(-\sum A_{nb} U^c_{nb}/A_P)}$ etc. in the equations, a direct coupled solution of the problem would result, see [42] for more details. The approximated equations for the cell-face correction velocities are now given as

$$
\begin{aligned}
U^c_e &= \overline{\left(\frac{1}{A_{P,U}}\right)_e} \Big[(\alpha_{\xi x})_e \left(P^c_E - P^c_P\right) \\
&+ (\alpha_{\eta x})_e \left(P^c_{ne} - P^c_{se}\right) + (\alpha_{\zeta x})_e \left(P^c_{te} - P^c_{be}\right)\Big] \ ,
\end{aligned}
$$

$$
\begin{aligned}
V_e^c &= \overline{\left(\frac{1}{A_{P,V}}\right)}_e \left[(\alpha_{\xi y})_e (P_E^c - P_P^c)\right. \\
&+ \left.(\alpha_{\eta y})_e (P_{ne}^c - P_{se}^c) + (\alpha_{\zeta y})_e (P_{te}^c - P_{be}^c)\right] \ , \\
W_e^c &= \overline{\left(\frac{1}{A_{P,W}}\right)}_e \left[(\alpha_{\xi z})_e (P_E^c - P_P^c)\right. \\
&+ \left.(\alpha_{\eta z})_e (P_{ne}^c - P_{se}^c) + (\alpha_{\zeta z})_e (P_{te}^c - P_{be}^c)\right] \ . \qquad (74)
\end{aligned}
$$

The same procedure can be applied to the cell centre velocities to get a correction equation for these

$$
\begin{aligned}
U_P^c &= \frac{1}{A_{P,U}} \left[(\alpha_{\xi x})_e P_e^c - (\alpha_{\xi x})_w P_w^c\right. \\
&+ \left.(\alpha_{\eta x})_n P_n^c - (\alpha_{\eta x})_s P_s^c + (\alpha_{\zeta x})_t P_t^c - (\alpha_{\zeta x})_b P_b^c\right] \ , \\
V_P^c &= \frac{1}{A_{P,V}} \left[(\alpha_{\xi x})_e P_e^c - (\alpha_{\xi x})_w P_w^c\right. \\
&+ \left.(\alpha_{\eta x})_n P_n^c - (\alpha_{\eta x})_s P_s^c + (\alpha_{\zeta x})_t P_t^c - (\alpha_{\zeta x})_b P_b^c\right] \ , \\
W_P^c &= \frac{1}{A_{P,W}} \left[(\alpha_{\xi x})_e P_e^c - (\alpha_{\xi x})_w P_w^c\right. \\
&+ \left.(\alpha_{\eta x})_n P_n^c - (\alpha_{\eta x})_s P_s^c + (\alpha_{\zeta x})_t P_t^c - (\alpha_{\zeta x})_b P_b^c\right] \ . \qquad (75)
\end{aligned}
$$

The pressure equation, which in practice will be a pressure correction equation, can now be derived from the continuity equation. Expressing the cell-face fluxes by the correct cell-face velocities, indicated by $''$, inserting these into the discrete continuity equation and using (73), we get

$$
C_e^c + C_e' - C_w^c - C_w' + C_n^c + C_n' - C_s^c - C_s' + C_t^c + C_t' - C_b^c - C_b' = 0 \ ,
$$

or after rearranging and using (71)

$$
C_e^c - C_w^c + C_n^c - C_s^c + C_t^c - C_b^c = S_{\text{mass}} \ . \qquad (76)
$$

The right-hand side of the equation is the mass deficit, S_{mass}, generated when the momentum equations are solved. When performing the pressure correction, the S_{mass} term is therefore already known, and if no mass deficit exists no pressure correction will result.

The cell face correction flux for the east cell face can be expressed using (74) and (31), namely

$$
\begin{aligned}
C_e^c &= \rho_e \left[(\alpha_{\xi x})_e \overline{\left(\frac{1}{A_{P,U}}\right)}_e \left((\alpha_{\xi x})_e (P_E^c - P_P^c)\right.\right. \\
&+ (\alpha_{\eta x})_e \frac{1}{4}((P_N^c - P_S^c) + (P_{NE}^c - P_{SE}^c)) \\
&+ \left.(\alpha_{\zeta x})_e \frac{1}{4}((P_T^c - P_B^c) + (P_{TE}^c - P_{BE}^c))\right) \\
&+ (\alpha_{\xi y})_e \overline{\left(\frac{1}{A_{P,V}}\right)}_e \left((\alpha_{\xi y})_e (P_E^c - P_P^c)\right. \\
&+ (\alpha_{\eta y})_e \frac{1}{4}((P_N^c - P_S^c) + (P_{NE}^c - P_{SE}^c)) \\
&+ \left.\left.(\alpha_{\zeta y})_e \frac{1}{4}((P_T^c - P_B^c) + (P_{TE}^c - P_{BE}^c))\right)\right.
\end{aligned}
$$

$$
\begin{aligned}
&+ \quad (\alpha_{\xi z})_e \overline{\left(\frac{1}{A_{P,W}}\right)}_e \Big((\alpha_{\xi z})_e (P_E^c - P_P^c) \\
&+ \quad (\alpha_{\eta z})_e \frac{1}{4}((P_N^c - P_S^c) + (P_{NE}^c - P_{SE}^c)) \\
&+ \quad (\alpha_{\zeta z})_e \frac{1}{4}((P_T^c - P_B^c) + (P_{TE}^c - P_{BE}^c)) \Big) \Big] \ .
\end{aligned}
$$

Similar expressions can be derived for the remaining cell face fluxes and after rearranging we end up with

$$
\begin{aligned}
C_e^c &= H1_e P_E^c - H1_e P_P^c \\
&+ M1_e P_N^c - M1_e P_S^c + M1_e P_{NE}^c - M1_e P_{SE}^c \\
&+ M2_e P_T^c - M2_e P_B^c + M2_e P_{TE}^c - M2_e P_{BE}^c \ , \\
C_w^c &= H1_w P_P^c - H1_w P_W^c \\
&+ M1_w P_N^c - M1_w P_S^c + M1_w P_{NW}^c - M1_w P_{SW}^c \\
&+ M2_w P_T^c - M2_w P_B^c + M2_w P_{TW}^c - M2_w P_{BW}^c \ , \\
C_n^c &= H1_n P_N^c - H1_n P_P^c \\
&+ M1_n P_E^c - M1_n P_W^c + M1_n P_{NE}^c - M1_n P_{NW}^c \\
&+ M2_n P_T^c - M2_n P_B^c + M2_n P_{TN}^c - M2_n P_{BN}^c \ , \\
C_s^c &= H1_s P_P^c - H1_s P_S^c \\
&+ M1_s P_E^c - M1_s P_W^c + M1_s P_{SE}^c - M1_s P_{SW}^c \\
&+ M2_s P_T^c - M2_s P_B^c + M2_s P_{TS}^c - M2_s P_{BS}^c \ , \\
C_t^c &= H1_t P_T^c - H1_t P_P^c \\
&+ M1_t P_E^c - M1_t P_W^c + M1_t P_{TE}^c - M1_t P_{TW}^c \\
&+ M2_t P_N^c - M2_t P_S^c + M2_t P_{TN}^c - M2_t P_{TS}^c \ , \\
C_b^c &= H1_b P_P^c - H1_b P_B^c \\
&+ M1_b P_E^c - M1_b P_W^c + M1_b P_{BE}^c - M1_b P_{BW}^c \\
&+ M2_b P_N^c - M2_b P_S^c + M2_b P_{BN}^c - M2_b P_{BS}^c \ .
\end{aligned}
$$

The constants H1, M1 and M2 are given by

$$
\begin{aligned}
H1_e &= \rho_e \left(\alpha_{\xi x}^2 \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}^2 \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}^2 \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_e \ , \\
H1_w &= \rho_w \left(\alpha_{\xi x}^2 \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}^2 \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}^2 \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_w \ , \\
H1_n &= \rho_n \left(\alpha_{\eta x}^2 \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\eta y}^2 \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\eta z}^2 \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_n \ , \\
H1_s &= \rho_s \left(\alpha_{\eta x}^2 \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\eta y}^2 \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\eta z}^2 \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_s \ ,
\end{aligned}
$$

$$
\begin{aligned}
H1_t &= \rho_t \left(\alpha_{\zeta x}^2 \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\zeta y}^2 \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\zeta z}^2 \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_t \ , \\
H1_b &= \rho_b \left(\alpha_{\zeta x}^2 \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\zeta y}^2 \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\zeta z}^2 \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_b \ , \\
M1_e &= \frac{1}{4}\rho_e \left(\alpha_{\xi x}\alpha_{\eta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\eta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\eta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_e \ , \\
M1_w &= \frac{1}{4}\rho_w \left(\alpha_{\xi x}\alpha_{\eta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\eta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\eta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_w \ , \\
M1_n &= \frac{1}{4}\rho_n \left(\alpha_{\xi x}\alpha_{\eta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\eta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\eta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_n \ , \\
M1_s &= \frac{1}{4}\rho_s \left(\alpha_{\xi x}\alpha_{\eta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\eta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\eta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_s \ , \\
M1_t &= \frac{1}{4}\rho_t \left(\alpha_{\xi x}\alpha_{\eta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\eta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\eta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_t \ , \\
M1_b &= \frac{1}{4}\rho_b \left(\alpha_{\xi x}\alpha_{\eta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\eta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\eta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_b \ , \\
M2_e &= \frac{1}{4}\rho_e \left(\alpha_{\xi x}\alpha_{\zeta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\zeta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\zeta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_e \ , \\
M2_w &= \frac{1}{4}\rho_w \left(\alpha_{\xi x}\alpha_{\zeta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\zeta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\zeta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_w \ , \\
M2_n &= \frac{1}{4}\rho_n \left(\alpha_{\xi x}\alpha_{\zeta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\zeta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\zeta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_n \ , \\
M2_s &= \frac{1}{4}\rho_s \left(\alpha_{\xi x}\alpha_{\zeta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\zeta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\zeta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_s \ , \\
M2_t &= \frac{1}{4}\rho_t \left(\alpha_{\xi x}\alpha_{\zeta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\zeta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\zeta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_t \ , \\
M2_b &= \frac{1}{4}\rho_b \left(\alpha_{\xi x}\alpha_{\zeta x} \overline{\left(\frac{1}{A_{P,U}}\right)} + \alpha_{\xi y}\alpha_{\zeta y} \overline{\left(\frac{1}{A_{P,V}}\right)} + \alpha_{\xi z}\alpha_{\zeta z} \overline{\left(\frac{1}{A_{P,W}}\right)} \right)_b \ .
\end{aligned}
$$

Inserting the correction cell face fluxes into the continuity correction equation (76) and rearranging to get an equation of the form

$$A_P P_P^c + \sum A_{nb} P_{nb}^c = S_{\text{mass}} \ , \tag{77}$$

we get the following expressions for the influence coefficients.

$$
\begin{aligned}
A_E &= H1_e + M1_n - M1_s + M1_t - M1_b \ , \\
A_W &= H1_w - M1_n + M1_s - M1_t + M1_b \ , \\
A_N &= H1_n + M1_e - M1_w + M2_t - M2_b \ , \\
A_S &= H1_s - M1_e + M1_w - M2_t + M2_b \ , \\
A_T &= H1_t + M2_e - M2_w + M2_n - M2_s \ ,
\end{aligned}
$$

$$
\begin{aligned}
A_B &= H1_b - M2_e + M2_w - M2_n + M2_s \ , \\
A_{NE} &= M1_e + M1_n \ , \\
A_{SE} &= -M1_e - M1_s \ , \\
A_{TE} &= M2_e + M1_t \ , \\
A_{BE} &= -M2_e - M1_b \ , \\
A_{NW} &= -M1_w - M1_n \ , \\
A_{SW} &= M1_w + M1_s \ , \\
A_{TW} &= -M2_w - M1_t \ , \\
A_{BW} &= M2_w + M1_b \ , \\
A_{TN} &= M2_n + M2_t \ , \\
A_{BN} &= -M2_n - M2_b \ , \\
A_{TS} &= -M2_s - M2_t \ , \\
A_{BS} &= M2_s + M2_b \ , \\
A_P &= -\sum A_{nb} \ . \qquad (78)
\end{aligned}
$$

5.4 Solution of the pressure correction equation

As the solution of the pressure correction equation takes most of the total solution time, it will need special attention. As seen from (77) and (78) the computational molecule for the pressure is a nineteen-point molecule. This implies that the matrix system to be solved is a nineteen-band diagonal matrix.

As either a line solver will be used in alternating lines, or a plane solver on alternating planes, a technique must be chosen that reduces the computational molecule.

Choosing an explicit treatment for all of the M1 and M2 terms (cross-terms), as in the case of the momentum equations, could be one solution. One major difference between the momentum equations and the pressure equation, is that the pressure equation has to be solved within a certain tolerance in order to make the algorithm converge. The explicitly treated source term would then have to be updated within the inner iteration of the pressure equation, making it rather expensive in terms of computer time. Instead, an alternative practice is used, shown to be working well in previous studies [45], [47] and [30].

In this practice the cross terms are simply dropped, giving a seven-diagonal system without the need for updating the source term within the inner iteration.

The reason why this approximation is working can be seen from the following facts. When the mesh is moderately nonorthogonal the cross-terms (M1 and M2 in (78)) will be small and even zero for an orthogonal mesh. Secondly as the solution converges either within the inner iteration of a single time-step or for the outer iteration of a steady-state calculation, the mass deficit will become smaller and thereby reducing the inaccuracies introduced by neglecting the cross-terms. When the solution has converged, no correction is needed, and the reduced pressure correction equation is equal to the full pressure correction equation. The reduced equation has the following form

$$A_P P_P^c + \sum A_{nb} P_{nb}^c = S_{\text{mass}} \ , \qquad (79)$$

The coefficients of the reduced equation are given by

$$A_E = H1_e \ ,$$

$$
\begin{aligned}
A_W &= H1_w \ , \\
A_N &= H1_n \ , \\
A_S &= H1_s \ , \\
A_T &= H1_t \ , \\
A_B &= H1_b \ , \\
A_P &= -\sum A_{nb} \ , \\
S_{\text{mass}} &= \text{ is given by (71).}
\end{aligned} \tag{80}
$$

After solving the pressure equation (79), the cell centre pressure can be corrected using (73). In this process it is necessary to use underrelaxation [42], and (73) is changed to

$$
P^{''} = P^{'} + \alpha_{relax} P^c \ , \text{ with } \alpha_{relax} < 1 \ . \tag{81}
$$

The cell-centre velocities will be corrected using (73) and (75). Finally the cell-face fluxes will be corrected according to the reduced pressure equation, using the following expression

$$
\begin{aligned}
C_e^{''} &= C_e^{'} + A_E \left(P_E^c - P_P^c \right) \ , \\
C_w^{''} &= C_w^{'} + A_W \left(P_P^c - P_W^c \right) \ , \\
C_n^{''} &= C_n^{'} + A_N \left(P_N^c - P_P^c \right) \ , \\
C_s^{''} &= C_s^{'} + A_S \left(P_P^c - P_S^c \right) \ , \\
C_t^{''} &= C_t^{'} + A_T \left(P_T^c - P_P^c \right) \ , \\
C_b^{''} &= C_b^{'} + A_B \left(P_P^c - P_B^c \right) \ .
\end{aligned} \tag{82}
$$

After this correction, the cell centre velocities will in general not fulfill the momentum equations. To make the field satisfy both the momentum equations and the continuity equation simultaneously, the process is started all over again. The time dependent part of the source terms $S_{U-\text{mom}}$, $S_{V-\text{mom}}$ and $S_{W-\text{mom}}$, along with the part of the central coefficients dependent on time ($\frac{\rho J}{\Delta t}$) are not changed during this process. When both the momentum equations and continuity equation are fulfilled, a new time-step can be taken. The final value of the cell face fluxes will be stored and used to calculated the convective terms in the transport equations.

5.5 Closure

The derivation of the pressure correction equation based on the discrete continuity equation using Rhie/Chow interpolation, was described. It was argued that the 19 band diagonal matrix (in 3D) resulting from this derivation could be approximated by a seven band diagonal diagonal matrix. The expressions to correct the cell centre velocities and the cell face fluxes according to the pressure correction were also given.

The convergens criterium used for the pressure correction equation in the present work was a reduction by a factor 100 of the mass deficit.

6 Boundary conditions

6.1 Introduction

Before the solution of the derived discrete equations can be attempted, the specification of boundary conditions at the domain boundaries must be addressed. Here physical boundary conditions as wall boundaries, inlet and outlet as well as boundary conditions of more computational nature as periodic boundaries and symmetry conditions will be discussed.

As the program is able to handle both steady-state and transient computations, a connection between the underrelaxation used in steady-state calculations and the time step used for transient computations will be shown. This simple relation between underrelaxation and time step, will then be used in the transient calculations to derive optimum time steps at each time level during the calculation.

6.2 Generalized boundary conditions

The main part of the physical boundary conditions used in the present study is either a prescription of a value at the cell face, or a prescription of the gradient at a cell face. Exceptions from this are cyclic boundary conditions, the treatment of the pressure, and the treatment of the turbulent wall boundary conditions. The different types of boundary conditions will be dealt with in this chapter, except the turbulent wall boundary conditions which will be dealt with in the following chapter.

Specifying boundary conditions we will use the same practice for the boundary conditions as for the inner part of the domain, meaning that we will use linear interpolation between cell centres to obtain values at cell faces, and central difference between cell centres to obtain gradients at cell faces.

Dirichlet condition

Starting with the prescribed value at the cell face (Dirichlet condition) taking an east cell face as an example, and using linear interpolation with an interpolation factor equal to one half, we get

$$\begin{aligned} \varphi_e &= \frac{\varphi_E + \varphi_P}{2} = \text{constant} \\ \varphi_E &= 2\varphi_e - \varphi_P \ . \end{aligned} \tag{83}$$

Adjusting the discrete finite-volume equation in order to reflect the boundary conditions can now be done by inserting the expressions for φ_E into the finite-volume equations and rearranging. For a prescription of the east cell-face value (83) results in

$$\begin{aligned} &A_P\varphi_P + A_E\left(2\varphi_e - \varphi_P\right) + A_W\varphi_W + A_N\varphi_N + A_S\varphi_S + A_T\varphi_T + A_B\varphi_B = S_\varphi \\ &A'_P\varphi_P + A'_E\varphi_E + A'_W\varphi_W + A'_N\varphi_N + A'_S\varphi_S + A'_T\varphi_T + A'_B\varphi_B = S'_\varphi \ . \end{aligned}$$

where

$$\begin{aligned} A'_P &= A_P - A_E \ , \\ S'_\varphi &= S_\varphi - 2A_E\varphi_e \ , \\ A'_E &= 0 \ , \end{aligned} \tag{84}$$

and the unmentioned coefficients have been left unchanged.

Neumann condition

For the second type of boundary condition, the prescribed gradient (von Neumann condition) using central differences we get, again at an east face

$$\left(\frac{\partial\varphi}{\partial n}\right)_e = \varphi_E - \varphi_P = \text{constant}$$

$$\varphi_E = \left(\frac{\partial\varphi}{\partial n}\right)_e + \varphi_P \ . \tag{85}$$

Adjusting the discrete finite-volume equation in order to reflect the boundary conditions, can now be done by inserting the expressions for φ_E into the finite volume equations and rearranging. For a prescribed gradient at a cell face (85), we get

$$A_P\varphi_P + A_E\left[\left(\frac{\partial\varphi}{\partial n}\right)_e + \varphi_P\right] + A_W\varphi_W + A_N\varphi_N + A_S\varphi_S + A_T\varphi_T + A_B\varphi_B = S_\varphi$$

$$A'_P\varphi_P + A'_E\varphi_E + A'_W\varphi_W + A'_N\varphi_N + A'_S\varphi_S + A'_T\varphi_T + A'_B\varphi_B = S'_\varphi \ .$$

where

$$\begin{aligned} A'_P &= A_P + A_E \ , \\ S'_\varphi &= S_\varphi - A_E\left(\frac{\partial\varphi}{\partial n}\right)_e \ , \\ A'_E &= 0 \ , \end{aligned} \tag{86}$$

and the unmentioned coefficients have been left unchanged.

6.3 Inlet

At an inlet, the values of the variables must be known, and for a laminar calculation the three velocity components must be known. In addition, the value of turbulent kinetic energy and dissipation of turbulent kinetic energy must be known when turbulent calculations using the $k-\epsilon$ model are performed.

The inlet values can be known either from measurements or from theoretical considerations. For a turbulent calculation sufficient data for k and ϵ will often be unavailable, and in lack of better information the theoretical equilibrium profile will often have to be used instead.

For an inlet at an east wall we thus have an expression equal to (83) with $\varphi_e = \varphi_{\text{inlet}}$ which results in the following changes to the discrete equations according to (84)

$$\begin{aligned} A'_P &= A_P - A_E \ , \\ S'_\varphi &= S_\varphi - 2A_E\varphi_{\text{inlet}} \ , \\ A'_E &= 0 \ . \end{aligned} \tag{87}$$

6.4 Outlet

At an outlet the assumption of a fully developed flow will be used. For high Reynolds number flows this assumption will be in good agreement with the physics, but at low Reynolds numbers the parabolic nature of the flow is weakened making the assumption less accurate. However, by placing the outlet far downstream of the region of interest, the errors introduced by imposing the slightly incorrect

boundary condition can be minimized. The assumption of a fully developed flow corresponds to zero normal-gradient at the outlet.

For an east wall the outlet condition is equal to (85) with $(\partial\varphi/\partial n)_e = 0$ and results in changes of the discrete equations according to (86)

$$\begin{aligned} A'_P &= A_P + A_E \ , \\ S'_\varphi &= S_\varphi \ , \\ A'_E &= 0 \ . \end{aligned} \tag{88}$$

In order to guarantee convergence of the pressure correction algorithm, global continuity must be ensured. As the fluxes are fixed at inlets, walls, symmetry planes etc., we will scale the mass fluxes at the outlet to ensure the fulfilment of the requirement of global continuity.

After applying the fully developed assumption, velocities and density are obtained at the outlet boundary. Calculating the outlet mass flux from these values, the sum of the outlet mass fluxes will in general not equal the sum of the mass fluxes going into the domain. To ensure this the velocities at the outlet are scaled by the ratio between the inlet and the outlet mass fluxes.

6.5 Symmetry plane

At a symmetry plane the normal gradient and the flux through the plane are zero. For a vector, taking the velocity as an example, the symmetry condition implies the following bindings on the three Cartesian velocity components. The normal velocity at the symmetry plane must be zero, and in addition the normal gradient of each of the two tangential components must be zero.

$$\begin{aligned} V_n &= 0 \ , \\ \frac{\partial V_t}{\partial n} &= 0 \ . \end{aligned} \tag{89}$$

For a scalar variable the condition is less complex and simply states that the normal gradient of the variable must be equal to zero at the symmetry plane.

$$\frac{\partial \varphi}{\partial n} = 0 \ . \tag{90}$$

The symmetry condition could be implemented for arbitrary planes, but in most practical cases the symmetry planes are coplanar with the Cartesian planes. As the restriction to Cartesian symmetry planes offers a simplification of the boundary condition for vector quantities and only minor restrictions, we will choose to allow Cartesian symmetry planes only.

For a vector variable at an east wall parallel to the Cartesian x-plane, the symmetry condition, here choosing the velocity as the variable, will result in the following three conditions on the three Cartesian velocity components

$$\begin{aligned} U_e &= 0 \ , \\ \left(\frac{\partial V}{\partial n}\right)_e &= 0 \ , \\ \left(\frac{\partial W}{\partial n}\right)_e &= 0 \ . \end{aligned} \tag{91}$$

The condition on the U-momentum equation is equal to (83) with $\varphi_e = \mathrm{U}_e = 0$ and results in a change of the discrete equation according to (84)

$$A'_P = A_P - A_E \ ,$$

$$S'_{U-\mathrm{mom}} = S_{U-\mathrm{mom}} \ ,$$
$$A'_E = 0 \ . \tag{92}$$

The conditions on V and W-momentum equations are both equal to (85) with $(\partial/\partial n)_e = 0$ and result in changes of the discrete equations according to (86)

$$A'_P = A_P + A_E \ ,$$
$$S'_{V-\mathrm{mom}} = S_{V-\mathrm{mom}} \ ,$$
$$A'_E = 0 \ , \tag{93}$$

and

$$A'_P = A_P + A_E \ ,$$
$$S'_{W-\mathrm{mom}} = S_{W-\mathrm{mom}} \ ,$$
$$A'_E = 0 \ . \tag{94}$$

For a scalar variable at an east wall parallel to the Cartesian x-plane, the symmetry condition will result in the following condition

$$\left(\frac{\partial \varphi}{\partial n}\right)_e = 0 \ . \tag{95}$$

This condition is equal to (85) with $(\partial\varphi/\partial n)_e = 0$ and results in change of the discrete equation according to (86)

$$A'_P = A_P + A_E \ ,$$
$$S'_\varphi = S_\varphi \ ,$$
$$A'_E = 0 \ . \tag{96}$$

6.6 Wall boundary

At a wall, different conditions exist for different variables. Here we will concentrate on the boundary condition on the velocity, as the pressure, pressure correction, and turbulent quantities are dealt with else where.

The physical condition on the velocity at a wall, is a no-slip condition ensuring the velocity to be zero at the wall. At an east wall this results in the following expressions for three Cartesian components of the velocity

$$U_e = 0 \ ,$$
$$V_e = 0 \ ,$$
$$W_e = 0 \ ,$$

all of the Dirichlet type (83), with $\varphi_e = 0$, corresponding to the following adjustment of the discrete equation according to (84).

U-momentum

$$A'_P = A_P - A_E \ ,$$
$$S'_{U-\mathrm{mom}} = S_{U-\mathrm{mom}} \ ,$$
$$A'_E = 0 \ , \tag{97}$$

V-momentum

$$A'_P = A_P - A_E \ ,$$
$$S'_{V-\mathrm{mom}} = S_{V-\mathrm{mom}} \ ,$$
$$A'_E = 0 \ , \tag{98}$$

and W-momentum

$$A'_P = A_P - A_E \ ,$$
$$S'_{W-\mathrm{mom}} = S_{W-\mathrm{mom}} \ ,$$
$$A'_E = 0 \ . \tag{99}$$

6.7 Pressure and pressure correction

Boundary conditions for the pressure and pressure correction must be given at all external boundaries.

Pressure correction

The boundary condition for the pressure correction is of the von Neumann type (85). This can be deduced in the following way.

At the time of pressure correction the fluxes at all external boundaries must be known, for inlets, walls, symmetry boundaries and outlets. As the fluxes thus are known already, no flux correction is needed at the domain boundaries. Looking at the expression for the flux correction (82) the requirement that the flux must be left unchanged at the boundaries implies that the gradient of the pressure correction is zero.

Taking an east face as an example, this results in adjustment of the following coefficient in the discrete equation in accordance with (86), where $(\partial/\partial n)_e$ has been used

$$\begin{aligned} A'_P &= A_P + A_E \ , \\ S' &= S \\ A'_E &= 0 \ . \end{aligned} \tag{100}$$

Pressure

Even though the boundary condition for the pressure correction is a specification of zero pressure correction, it can be seen from a simple example that the same condition cannot be applied to the pressure. In order to get the correct pressure source in the momentum equations, the following practices will be used for the pressure.

We take a fully developed flow between two infinite plates, with the flow going from west to east, with walls at the top and bottom boundaries and extending infinitely in the north and south direction. Looking at the outlet boundary at the east face, the pressure gradient here cannot be zero as it is the driving force for the flux over the outlet. In fact the condition in this special case is a zero second derivative of the pressure or constant gradient over the outlet.

As the pressure has no equation of its own, the boundary condition will be on the variable instead of on the coefficients. In the general case, the pressure at the boundaries will be calculated using second-order extrapolation from the known values in the inner part of the domain. At an east face this results in

$$P_e = \frac{15}{8}P_P - \frac{10}{8}P_W + \frac{3}{8}P_{WW} \ . \tag{101}$$

For the fully developed flow between infinite plates, the practices mentioned above will result in the correct boundary condition.

6.8 Periodic boundaries

The cyclic or periodic boundary condition is often used for fully developed flows, as a mean to reduce the computational domain. Typical areas of application are complex heat exchangers, or flow in complex channels such as the flow shown in Fig. 5.

For the flow in a channel with transversal ribs as the one shown in Fig. 5, the flow will be fully developed after passing of the initial developing zone. For

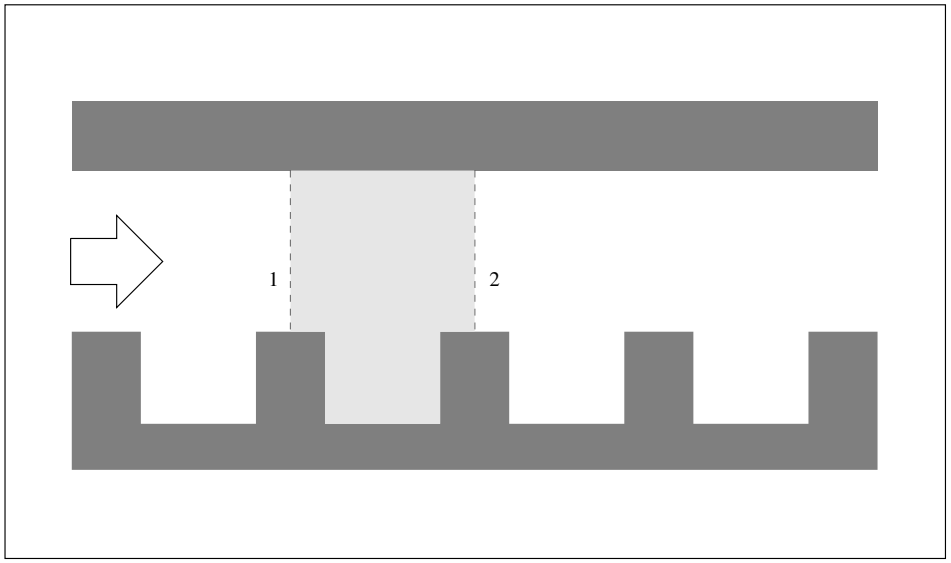

Figure 5. Channel with transversal ribs on one wall. When the initial developing region is passed and the flow has become fully developed, a periodic element can be identified, where all variables except the pressure has the same value at boundary 1 and 2.

the fully developed region a periodic element can be identified for which the flow pattern is repeated, thus the boundary condition at position 1 in Fig. 5 is given by the state of the variables at position 2 and vice versa.

This coupling can be programmed directly into the equation system, which results in an implicit application of the cyclic boundary condition. This approach gives a fast convergence, but unfortunately it destroys the band structure of the equation matrix, and in addition this approach is difficult to program in a general fashion. For the last two reasons, this approach will be abandoned and an alternative practice will be used, even though it is known that this alternative practice will not converge as fast as the implicit implementation.

Instead of the implicit method we will use an explicit implementation, this method will not change the structure of the equation matrix, and it can be implemented in a general fashion. This technique relies on the iterative solution strategy by explicitly moving all of the variables (except the pressure and pressure correction, which are specified in the standard way) from position 2 to position 1 and vice versa after every iteration until convergence. For a transient calculation this procedure may prove inconvenient, but for a steady-state calculation it should be applicable.

6.9 Time step and underrelaxation

Using the decoupled SIMPLE strategy or other algorithms of the same family (PISO, SIMPLEC etc.) underrelaxation of the discrete equations becomes necessary. We will show the well-known connection between the underrelaxation factor used for steady-state calculations and the time step entering the time dependent terms in the transient equations.

For a transient calculation the U-momentum has the following form, see (63-64)

$$A_P U_P^{t+\Delta t} + \sum A_{nb} = S_{U-\mathrm{mom}} \ ,$$

where

$$A_P = \sum -A_{nb} + \frac{\rho_P J_P}{\Delta t} \ ,$$

$$S_{U-\mathrm{mom}} = S'_{U-\mathrm{mom}} + \frac{\rho_P J_P}{\Delta t} U_P^t \ . \tag{102}$$

and $S'_{U-\mathrm{mom}}$ represents the part of the source term that does not depend on time.

For a steady-state calculation, following Patankar [42], we have

$$A_P U_P + \sum A_{nb} U_{nb} = S_{U-\mathrm{mom}} \ ,$$

where

$$A_P = \sum -A_{nb} \frac{1}{\mathrm{relax}} \ ,$$

$$S_{U-\mathrm{mom}} = S'_{U-\mathrm{mom}} + \sum -A_{nb} \frac{\mathrm{relax}}{1-\mathrm{relax}} \ , \tag{103}$$

and a typical value of relax around 0.8.

From this it is easily seen that the following relation exists between the time step used in a transient calculation and the underrelaxation used in a steady-state calculation

$$\Delta t = \frac{\rho_P J_P}{-\sum A_{nb}} \frac{relax}{1-relax} \ . \tag{104}$$

For a transient calculation the cell holding the smallest time step will be found, and this time step will then be used for all cells, as a result all variables will be evaluated at the same time level.

For a steady-state calculation where the time evolution of the flow is of no interest, one can use the maximum allowable time step in every cell, thereby losing the true transient behaviour, but instead gain is obtained by reducing the number of iterations or 'distorted time step' necessary to reach the steady-state situation.

6.10 Closure

The necessary boundary conditions for inlet, outlet, symmetry, walls, and periodic boundaries have been described. Nearly all the boundary conditions treated here are of either the Dirichlet type or of the von Neumann type, the exceptions are the pressure boundary condition and the periodic boundary condition.

We discussed how to determine the time step in case of transient calculations, and in this connection the well-known steady-state iteration method was mentioned.

The final boundary condition to be discussed before all remedies necessary for a functional finite volume code has been addressed, is the turbulent wall boundary conditions for the $k-\epsilon$ model which is the subject of the following chapter.

7 Turbulent wall treatment

7.1 Introduction

The treatment of the wall boundary condition in connection with turbulent flows demands some special considerations. The treatment of laminar wall boundary conditions is found in the previous chapter.

The two aspects of turbulent flow near a wall dictating a special treatment are the strong gradients of the variables typical found near the walls and the fact that the local Reynolds number will become low as the wall is approached.

The problem of low local Reynolds number is a consequence of the high Reynolds number assumption used to neglect terms from the k and ϵ equations during derivation. Special so-called low Reynolds number versions of the $k-\epsilon$ models exist, which are applicable near walls.

The problem of large gradients of the variables near the walls is primarily a problem of the calculation time for the model going up, as the mesh is refined near walls in order to resolve the gradients.

Thus, the problem could be solved using a low Reynolds number version of the $k-\epsilon$ model and a very fine mesh near the walls. Even though this approach has been shown to work for industrial flows with smooth walls, the wish to make predictions in the atmospheric boundary-layer over rough walls seems to be impossible choosing this technique. The reason for this is the need to resolve the laminar sublayer of the individual roughness elements. For a flow over a hill the domain is typical of the size of 10 km, while the roughness elements have sizes of about 5-10 cm, resulting in the number of cells exceeding the storage capacity of modern computers.

Instead we will choose a technique where the calculation of the flow near the walls is abandoned, thus both circumventing the problem of the low Reynolds numbers and the need for fine meshes to resolve the strong gradients.

This alternative approach is to use the standard high Reynolds number $k-\epsilon$ model together with a model for the behaviour of the flow near the walls, known as the logarithmic wall-law. In the logarithmic wall-law method the steep gradient region near the walls is excluded from the computations, and instead the near wall flow is modelled by assuming one-dimensional Couette flow. By modelling the near-wall flow, the fine mesh needed here can be avoided, cutting down the execution time, and at the same time avoiding the low Reynolds number region allowing the use of the standard $k-\epsilon$ model.

7.2 Logarithmic law-of-the-wall

The logarithmic law-of-the-wall can be derived for a boundary-layer flow, where the variation in the flow direction is negligible, following Tennekes and Lumley [57]. In this special case we have $\partial/\partial x = 0$ except for the favourable pressure gradient driving the flow, which in fact is a one-dimensional Couette flow.

The three scales entering the general rough wall flow are the viscous length expressed as the ratio between the kinematic viscosity and the wall friction velocity ν/U_τ, the roughness height z_o and the height of the boundary layer δ.

Performing asymptotic matching of the expression valid for the surface layer and the outer layer (or core region for internal flow), we get the logarithmic wall-law

$$\frac{U}{U_\tau} = \frac{1}{\kappa}\log_e\left(y^+\right) + f(R_{z_o}) \ , \tag{105}$$

or

$$\frac{U}{U_\tau} = \frac{1}{\kappa}\log_e\left(\frac{y}{z_o}\right) + g(R_{z_o}) \ , \tag{106}$$

where

$$U_\tau = \sqrt{\frac{\tau_w}{\rho}}$$
$$R_{z_o} = \frac{z_o U_\tau}{\nu}$$
$$y^+ = \frac{y U_\tau}{\nu}. \tag{107}$$

Now we have a model describing the mean velocity profile in the vicinity of the wall for the general case of rough wall flow. To be able to use the $k-\epsilon$ model we must have corresponding models for the equation of turbulent kinetic energy, and the equation of dissipation of turbulent kinetic energy.

k-equation

Using the assumption of a steady flow with negligible development in the flow direction, the equation for turbulent kinetic energy reduces to

$$\begin{aligned} \nu_t \left(\frac{\partial U}{\partial y}\right)^2 &= \epsilon \\ \tau_w \frac{\partial U}{\partial y} &= \epsilon \\ \text{Prod} &= \epsilon \ , \end{aligned} \tag{108}$$

stating that in these conditions an equilibrium exists between the production (Prod) and dissipation (ϵ) of turbulent kinetic energy. Approximating the shear stress in the vicinity of the wall to be constant and equal to the wall stress $\tau_{xy} \approx \tau_w = \rho \nu_t \partial U/\partial y$, equation (108) gives

$$\text{Prod} = \nu_t \left(\frac{\tau_w}{\rho \nu_t}\right)^2 \ . \tag{109}$$

Alternatively we can use the definition of the eddy diffusivity, ν_t, and (108) to get

$$\begin{aligned} \nu_t \left(\frac{\tau_w}{\rho \nu_t}\right)^2 &= \epsilon \\ \tau_w &= \epsilon \rho^2 \nu_t \\ \tau_w &= \rho c_\mu^{\frac{1}{2}} k \ . \end{aligned} \tag{110}$$

Also the dissipation near the wall can be estimated from (108), using the knowledge of the logarithmic profile (105 -106) to calculate the gradient $\partial U/\partial y = U_\tau/\kappa y$, this yields

$$\begin{aligned} \epsilon &= \nu_t \left(\frac{\partial U}{\partial y}\right)^2 \\ \epsilon &= c_\mu \frac{k^2}{\epsilon} \left(\frac{U_\tau}{\kappa y}\right)^2 \\ \epsilon &= \frac{c_\mu^{\frac{3}{4}} k^{\frac{3}{2}}}{\kappa y} \ . \end{aligned} \tag{111}$$

If the wall stress τ_w is known, (110) may serve as a boundary condition for the turbulent kinetic energy equation. This condition can be relaxed by using an assumption of zero diffusion of kinetic energy to the wall, and instead using (109) and (111) to estimate the production and dissipation near the wall. Using the last approach, the value near the wall is not fixed but only the production and dissipation of turbulent kinetic energy are specified to be in balance.

ϵ-equation

In a numerical calculation of turbulent flow the ϵ-equation is often neglected at the point near a wall, and the value here is instead fixed by the value given by (111). This is done because the ϵ-equation is known to be in error near the wall, where the assumption of high local Reynolds number used during the derivation is often not fulfilled.

Smooth wall

In the case of flow over a smooth wall the roughness length is equal to zero, thereby reducing the number of scales entering the problem to two. Looking at (105) and specifying that $f(R_{z_0}) \to 5.5$ when the roughness Reynolds number goes to zero, the expression reduces to the well-known logarithmic wall-law for flow over smooth surfaces

$$\frac{U}{U_\tau} = \frac{1}{\kappa}\log_e(y^+) + 5.5 \ , \tag{112}$$

or

$$\frac{U}{U_\tau} = \frac{1}{\kappa}\log_e(Ey^+) \ , \ \mathrm{E} = 9.0 \ . \tag{113}$$

Rough wall

For $R_{z_o} \to \infty$, $g(R_{z_o})$ is independent of R_{z_o} and becomes a constant. Often, the position of the wall $y = 0$ is not known accurately enough to bother with the additive constant; instead, it is absorbed in the definition of z_o according to [57]. The logarithmic law of the wall for rough wall flow then reads

$$\frac{U}{U_\tau} = \frac{1}{\kappa}\log_e\left(\frac{y}{z_o}\right). \tag{114}$$

This profile is often used all the way down to $y/z_o = 1$ (where $U = 0$ when the additive constant is ignored), even though the profile is derived assuming that $y/z_o \to \infty$.

7.3 Coding of the logarithmic wall law

In the following we will discuss the coding of the logarithmic wall law for the case of smooth wall flow. First the boundary conditions for the momentum equations, then the boundary conditions the for kinetic energy equation and finally the treatment of the dissipation. In the end the analog results for the rough wall case will be listed.

Momentum equations

As we do not want to abandon the momentum equation in the near wall cell and fix the velocity according to the logarithmic law of the wall, we rewrite (113) using that $\tau_w = \rho U_\tau^2$

$$\begin{aligned}
\frac{U}{U_\tau}\frac{U_\tau}{U_\tau} &= \frac{1}{\kappa}\log_e(Ey^+) \\
\frac{U}{\frac{\tau_w}{\rho}}U_\tau &= \frac{1}{\kappa}\log_e(Ey^+) \\
\tau_w &= \frac{\rho\kappa U_\tau U}{\log_e(Ey^+)}
\end{aligned}$$

$$\tau_w = \frac{\rho\kappa c_\mu^{\frac{1}{4}} k^{\frac{1}{2}} U}{\log_e(Ey^+)}$$

$$\tau_w = \lambda U \quad , \text{where } \lambda = \frac{\rho\kappa c_\mu^{\frac{1}{4}} k^{\frac{1}{2}}}{\log_e(Ey^+)} \ . \tag{115}$$

Normally this expression is used in the fully turbulent region ($y^+ > 11.6$) whereas the expression used for the laminar sublayer ($y^+ < 11.6$) is given by

$$\tau_w = \lambda U \quad , \text{where } \lambda = \frac{\mu}{y} \ . \tag{116}$$

Integration of the wall stress over the cell face gives the force, T_w, that must replace the diffusive terms from the ordinary procedure (I^d)

$$T_w = \int \tau_w dA = \lambda U \text{Area}$$

In the general three-dimensional case the velocity U is the tangential velocity $\vec{V_t}$ in the near-wall point P, and the friction force is assumed to be directed in the opposite direction of this tangential velocity. In order to include the force in the momentum equations it must be decomposed in the Cartesian directions. Decomposing the friction force $\vec{T_w}$ is done by decomposing the tangential velocity $\vec{V_t}$ into Cartesian components, using

$$\vec{V_t} = \vec{V_{tot}} - \vec{V_n}.$$

Here $\vec{V_{tot}}$ is the velocity vector and $\vec{V_n}$ is the normal component of the velocity vector $\vec{V_n} = \vec{n}(\vec{n} \cdot \vec{V_{tot}})$ giving that

$$\vec{V_t} = \vec{V_{tot}} - \vec{n}(\vec{n} \cdot \vec{V_{tot}}).$$

In Cartesian components we have

$$\vec{V_{t1}} = (1 - n_1^2)U - n_1 n_2 V - n_1 n_3 W,$$
$$\vec{V_{t2}} = -n_1 n_2 U + (1 - n_2^2)V - n_2 n_3 W,$$
$$\vec{V_{t3}} = -n_1 n_3 U - n_2 n_3 V + (1 - n_3^2)W.$$

When this is inserted in the expression for the friction force we get

$$T_{w1} = \lambda \text{Area}((1 - n_1^2)U - n_1 n_2 V - n_1 n_3 W),$$
$$T_{w2} = \lambda \text{Area}(-n_1 n_2 U + (1 - n_2^2)V - n_2 n_3 W),$$
$$T_{w3} = \lambda \text{Area}(-n_1 n_3 U - n_2 n_3 V + (1 - n_3^2)W).$$

In the code, the boundary condition is then specified by setting the diffusive fluxes $I^d = I^{dn} + I^{dc}$ equal to zero, the convective flux is already zero as there is no flow through solid walls, and replacing these by the friction force, $\vec{T_w}$, found from the logarithmic conditions. Following standard linearization practice we get

U-momentum equation:

$$A'_P = A_P + \lambda \text{Area}(1 - n_1^2) \ ,$$
$$S'_{U-\text{mom}} = S_{U-\text{mom}} + \lambda \text{Area}(-n_1 n_2 V - n_1 n_3 W) \ .$$

V-momentum equation:

$$A'_P = A_P + \lambda \text{Area}(1 - n_2^2) \ ,$$
$$S'_{V-\text{mom}} = S_{V-\text{mom}} + \lambda \text{Area}(-n_1 n_2 U - n_1 n_3 W) \ .$$

W-momentum equation:

$$A'_P = A_P + \lambda \text{Area}(1 - n_3^2) \ ,$$
$$S'_{W-\text{mom}} = S_{W-\text{mom}} + \lambda \text{Area}(-n_1 n_3 U - n_1 n_2 V) \ . \tag{117}$$

k-equation

For the turbulent kinetic energy equation we will retain maximum flexibility by using the assumption of zero diffusion at the wall, and specifying the production and dissipation of kinetic energy according to (109) and (111).

For the mean production in the near wall cell we have from (108) that

$$\overline{\text{Prod}} = \frac{1}{y_n - y_o} \int_{y_o}^{y_n} \tau_w \frac{\partial U}{\partial y} dy.$$

Taking the integration from the lower limit of the logarithmic layer, meaning $y_o = \nu / U_\tau E$ for the smooth wall, to the top of the near wall cell y_n. Assuming the stress to be constant in the near-wall cell, we have

$$\overline{\text{Prod}} = \frac{\tau_w}{y_n - y_o}(U(y_n) - U(y_o)).$$

Using that $y_n = 2y_P$ and $y_P >> y_o$ we get

$$\overline{\text{Prod}} = \frac{\alpha}{2y_P} \tau_w U(y_P) \ . \tag{118}$$

Here we used that

$$\frac{U(y_n)}{U(y_P)} = \frac{log_e(E2y_p^+)}{log_e(Ey_p^+)} = \frac{log_e(Ey_p^+) + log_e(2)}{log_e(Ey_p^+)} = 1 + \frac{log_e(2)}{log_e(Ey_p^+)} = \alpha \ . \tag{119}$$

For the mean dissipation in the near-wall cell we have (111)

$$\bar{\epsilon} = \frac{1}{y_n - y_o} \int_{y_o}^{y_n} \frac{c_\mu^{\frac{3}{4}} k^{\frac{3}{2}}}{\kappa y} dy$$

Again we take the integration from the lower limit of the logarithmic layer, and assuming k to be constant over the near-wall cell we get

$$\bar{\epsilon} = \frac{1}{y_n - y_o} \frac{c_\mu^{\frac{3}{4}} k^{\frac{3}{2}}}{\kappa} (log_e(y_n) - log_e(y_o)).$$

Again using $y_p >> y_o$, and the approximations in (119) we get

$$\bar{\epsilon} = \frac{\alpha}{2y_P} c_\mu^{\frac{3}{4}} k^{\frac{3}{2}} U^+ \qquad \text{, where } U^+ = \frac{U}{U_\tau}. \tag{120}$$

In the computer code this is implemented by setting the diffusion flux $I^d = I_{dn} + I_{dc}$ equal to zero at the wall, and replacing the generation term in the kinetic energy equation and the dissipation term by (118) and (120), respectively.

As the boundary conditions state only that the production equals the dissipation of turbulent kinetic energy, the constant $\alpha/2$ can be dropped from the expressions for production and dissipation.

ϵ-equation

The equation for the dissipation of kinetic energy is abandoned in the near-wall cell as earlier described. This is implemented in the computer code by a simple variation of the standard way of fixing the value in the inner of a domain, see [42],

$$S = A_P \frac{c_\mu^{\frac{3}{4}} k^{\frac{3}{2}}}{\kappa y} \ ,$$

$$A_{nb} = 0 \ . \tag{121}$$

The advantage of using this approach instead of the usual multiplication with a large value is primarily seen for unsteady calculations, where the use of the standard approach would result in an erroneous time stepping for the dissipation equation in the wall cell.

Rough wall

The derivation carried out for the smooth wall flow can easily be carried out for the rough wall flow. The derivation for the rough wall flow will not be given here as there are only minor differences to the smooth wall derivation.

The difference between the smooth wall and rough wall expressions is mainly caused by the different expressions for the wall stress, which in the rough wall case are given by

$$\tau_w = \lambda U \qquad \text{, where } \lambda = \frac{\rho \kappa U_\tau}{\log_e\left(\frac{y}{z_0}\right)} \ . \tag{122}$$

Here no special treatment of the laminar sublayer is necessary, as the roughness elements penetrate the laminar sublayer reaching out in the fully turbulent region.

Using the expression for λ from (122), the boundary conditions for the rough wall case is identical to the expressions (117), (118), (120), and (121).

7.4 Closure

The turbulent boundary conditions for the high Reynolds number k-ϵ model were derived based on the assumption of a one dimensional Couette flow. The boundary conditions were stated both for flow over smooth and rough walls found in atmospheric flows.

It was found that the difference in between the two cases was minor concerning the changes that have to be made in the computer code, actually the only difference is the way of evaluating λ and the inclusion of the laminar sublayer for the case of flow over smooth walls.

The calculations performed have proved that the present implementation of the turbulent boundary conditions is very robust and gives reasonably good results.

The reasonably good results obtained for separating flows are quite surprising as the Couette flow assumption used for deriving the logarithmic wall law is not fulfilled. If one would like to have the ability to compute strongly separating flows more accurately, the use of the low Reynolds number version of the $k - \epsilon$ model could be a possibility for smooth wall flows. For flow over rough walls, as earlier discussed, this choice is not possible with the present computer capacity and a different path must be sought.

8 Geometric quantities

8.1 Introduction

When the differential equations are discretized, information about several geometric quantities is needed, cell-volumes, cell-face areas etc.

For the formulation used in the present study, all the necessary geometric informations can be derived from the cell vertices coordinates. This means that the user must supply the code with the coordinates of the cell vertices in form of a mesh file, the remaining geometric information necessary is then derived from these coordinates by the code.

The calculations of the different geometric quantities needed, will be given in this chapter. The calculation of the following geometric information used within the code is described :

- Cell volumes
- Cofactors or cell-face areas
- Normal vectors
- Normal distance between cell face and cell centre
- Linear interpolation factors

8.2 Volume calculation

The three-dimensional computational cells have the form of hexahedrons, each composed of eight corners joined by straight lines. The discrete volumes

$$\int J \, d\xi d\eta d\zeta = \text{Vol} \ ,$$

can be computed by a method suggested by Kordulla and Vinokur [20].

The practice given by [20] is to decompose the computational cell into six non-overlapping tetrahedras, for which a simple expression for the calculation of the individual volumes exist, and thereafter to sum the volumes of these six tetrahedras to get the volume of the computational cell.

Decomposing a cell is done by cutting it with three planes, where each of the planes goes through four of the cell vertices. To ensure for a general double curved surface that the cell volumes are non-overlapping, a cell face shared by two cells must be split into triangles in exactly the same way for both cells. A simple way to obtain this, is to use the same three planes for all cells, the planes used here will be a plane through the vertices a, c, h and f, a plane through the vertices a, d, e, and h, and a plane through the vertices a, b, g, and h, see Fig 6 and 7.

The volume of a tetrahedrons is computed by the expression:

$$\text{Vol} = \frac{1}{6}\vec{a} \cdot \left(\vec{b} \times \vec{c}\right)$$

where the three vectors $\vec{a}, \vec{b}$ and $\vec{c}$ distend the tetrahedra and form a right-hand system. Looking at Fig. 7, the volume of the computational cell can be computed as:

$$\begin{aligned} Vol &= \frac{1}{6}\left(\left(\vec{ac} \times \vec{ad}\right) + \left(\vec{af} \times \vec{ae}\right) + \left(\vec{ab} \times \vec{af}\right)\right. \\ &\quad \left. + (\vec{ae} \times \vec{ag}) + \left(\vec{ad} \times \vec{ab}\right) + (\vec{ag} \times \vec{ac})\right) \cdot \vec{ah} \end{aligned}$$

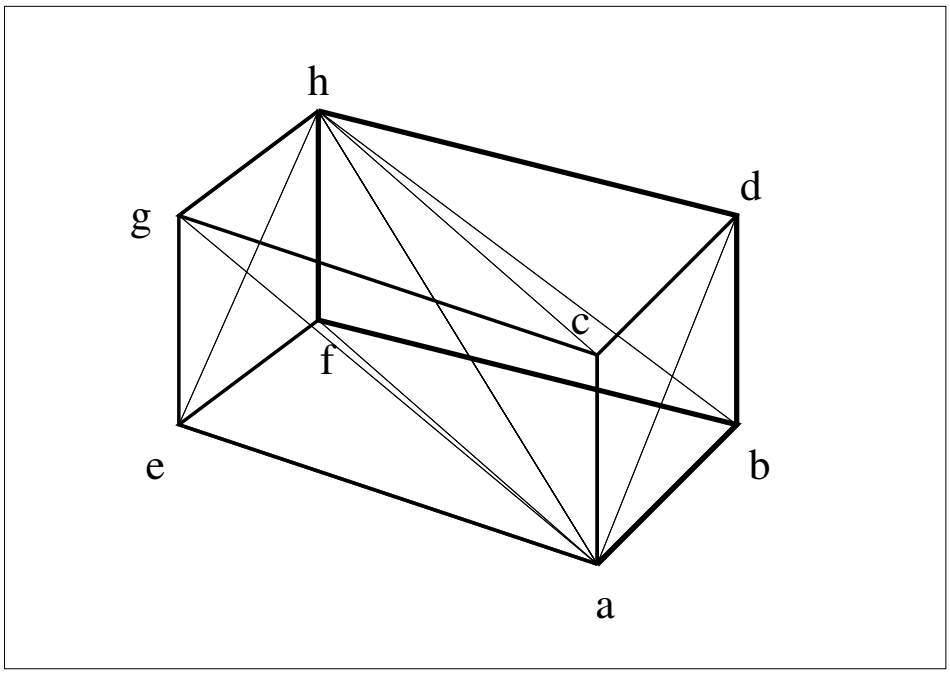

Figure 6. Three-dimensional computational cell, with lines drawn for decomposing it into six tetrahedras.

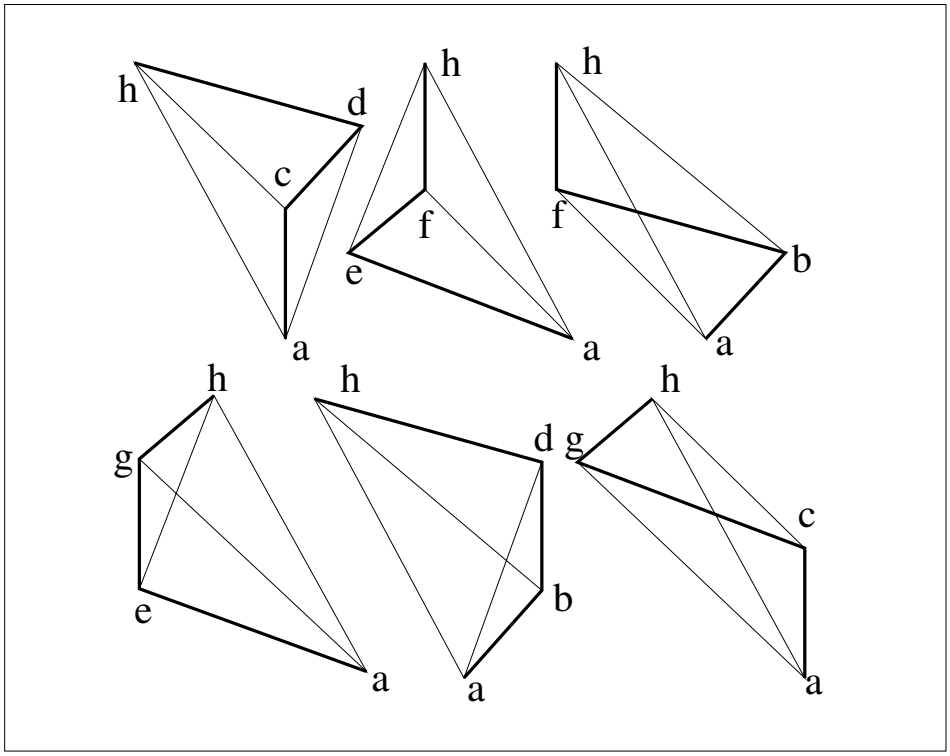

Figure 7. The six tetrahedras resulting from the decomposition of the computational cell.

8.3 Cofactors

The discrete version of the cofactors, α, can be identified with the projected area, except for the sign.

For a finite cell face, the discrete version of the expression for the cofactors (24)

will, in general, not fulfill this requirement, and another practice must be used.

Instead of using the discrete expressions for the cofactors we will calculate the projected areas using vector algebra, and afterwards choose the sign to get the right value.

To calculate the area of the cell face on Fig. 8, we will split the cell face into two triangles acd and abd. Using the same splitting as used for the cell-volume computation, it is ensured that the area of a cell face shared by two neighbouring cells will be equal, even for a general double-curved surface.

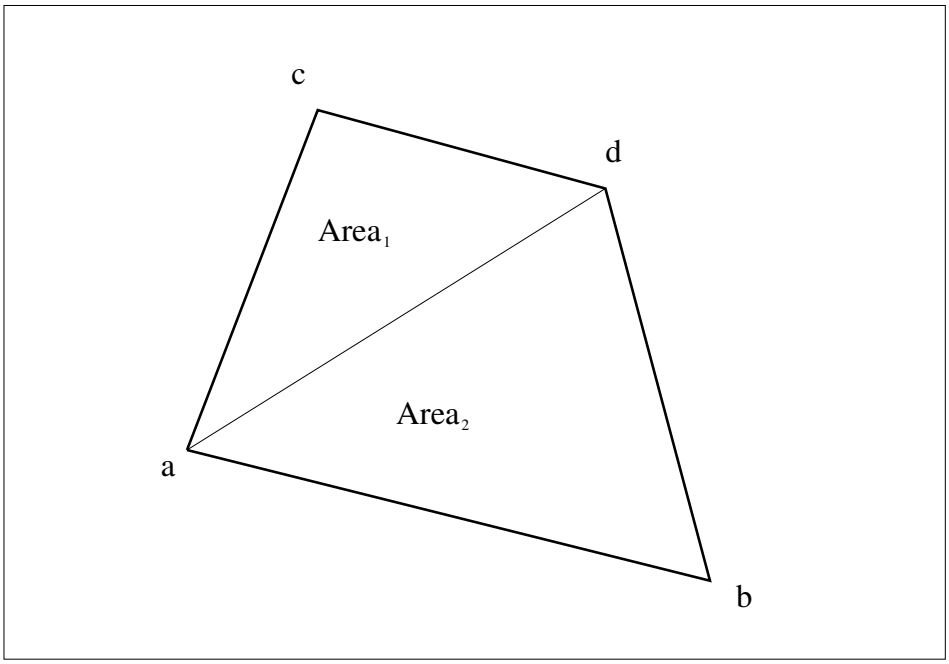

Figure 8. A cell face split into two triangles for area calculation.

The areas of these triangles can be computed as

$$\text{Area}_1 = \frac{1}{2} \mid \vec{ad} \times \vec{ac} \mid \ ,$$

$$\text{Area}_2 = \frac{1}{2} \mid \vec{ab} \times \vec{ad} \mid \ .$$

The area of the cell face is then

$$\text{Area} = \frac{1}{2} \left(\mid \vec{ad} \times \vec{ac} \mid + \mid \vec{ab} \times \vec{ad} \mid \right)$$

Being aware that both $\vec{ad}, \vec{ac}, \vec{ad} \times \vec{ac}$ and $\vec{ab}, \vec{ad}, \vec{ab} \times \vec{ad}$ are right hand systems, we have

$$\begin{aligned} \mid \vec{ad} \times \vec{ac} \mid + \mid \vec{ab} \times \vec{ad} \mid &= \mid \vec{ad} \times \vec{ac} \ + \vec{ab} \times \vec{ad} \mid \\ &= \mid \vec{ad} \times \vec{ac} \ - \vec{ad} \times \vec{ab} \mid \\ &= \mid \vec{ad} \times \left(\vec{ac} - \vec{ab} \right) \mid \\ &= \mid \vec{ad} \times \vec{cb} \mid \end{aligned}$$

$$\text{Area} = \frac{1}{2} \mid \vec{ad} \times \vec{cb} \mid$$

Choosing the vectors $\vec{ad}$ and $\vec{cb}$ such that the resulting vector is pointing into the cell, the following sign convention is used. If we are looking at a south, west or bottom wall of a cell, the cofactors are positive if the normal vector of the projected area is pointing in the direction of the corresponding Cartesian axis, and negative in the opposite case. To ensure that the cofactors shared by two cells have the same sign, the opposite convention is used for north, east and top walls.

Using the following relations between cofactors and projected areas with the above-mentioned sign convention the expressions for the cofactors are at hand, realizing that projecting on the x-plane the y and z components of the vectors are used, projecting on the y-plane the x and z component of the vectors are used, and projecting on the z-plane the x and y components of the vectors are used.

$$
\begin{aligned}
\alpha_{\xi x} &= \quad \xi\text{-face projected on } x\text{-plane} \\
\alpha_{\xi y} &= -\xi\text{-face projected on } y\text{-plane} \\
\alpha_{\xi z} &= \quad \xi\text{-face projected on } z\text{-plane} \\
\alpha_{\eta x} &= \quad \eta\text{-face projected on } x\text{-plane} \\
\alpha_{\eta y} &= -\eta\text{-face projected on } y\text{-plane} \\
\alpha_{\eta z} &= \quad \eta\text{-face projected on } z\text{-plane} \\
\alpha_{\zeta x} &= \quad \zeta\text{-face projected on } x\text{-plane} \\
\alpha_{\zeta y} &= -\zeta\text{-face projected on } y\text{-plane} \\
\alpha_{\zeta z} &= \quad \zeta\text{-face projected on } z\text{-plane}
\end{aligned}
$$

To save space in the code, only the discrete analog of the primary cofactors will be stored, $\alpha_{\xi x}$, $\alpha_{\xi y}$ and $\alpha_{\xi z}$ at ξ-faces, $\alpha_{\eta x}$, $\alpha_{\eta y}$, and $\alpha_{\eta z}$ at η-faces and $\alpha_{\zeta x}$, $\alpha_{\zeta y}$ and $\alpha_{\zeta z}$ at ζ-faces.

The remaining six discrete cofactors at each cell face will then be interpolated between the primary discrete cofactors when needed. For the east ξ-face we have, see Fig. 9

$$(\alpha_{\eta i})_e = \frac{1}{4}\left((\alpha_{\eta i})_s + (\alpha_{\eta i})_n + (\alpha_{\eta i})_{ne\,e} + (\alpha_{\eta i})_{se\,e}\right),$$

where i can be x, y or z.

The equivalent expressions for the remaining cell faces are straightforward.

8.4 Normal vector

In general the normal vector will be varying from point to point, but in most cases this variation will be small over a cell face. As a consequence a mean normal vector can be calculated as the vector product of the two cell face diagonal vectors $\vec{ad} \times \vec{bc}$, see Fig 8.

$$\vec{N} = \vec{ad} \times \vec{bc}\,,$$

$$\vec{n} = \frac{\vec{N}}{|\,\vec{N}\,|}\,.$$

When the unit mean normal vector $\vec{n}$ is calculated, the normal distance from the wall to the cell centre of the first inner cell can easily be calculated as the scalar product of the unit mean normal vector and the vector pointing from the cell face midpoint B to the cell centre P

$$\delta n = \vec{n} \cdot \vec{\mathrm{BP}}\,.$$

The cell-face midpoint is calculated as one fourth of the coordinate values of the four vertices enclosing the cell-face, and the cell-centre is calculated as one eighth of the coordinate values of the eight vertices enclosing the computational cell.

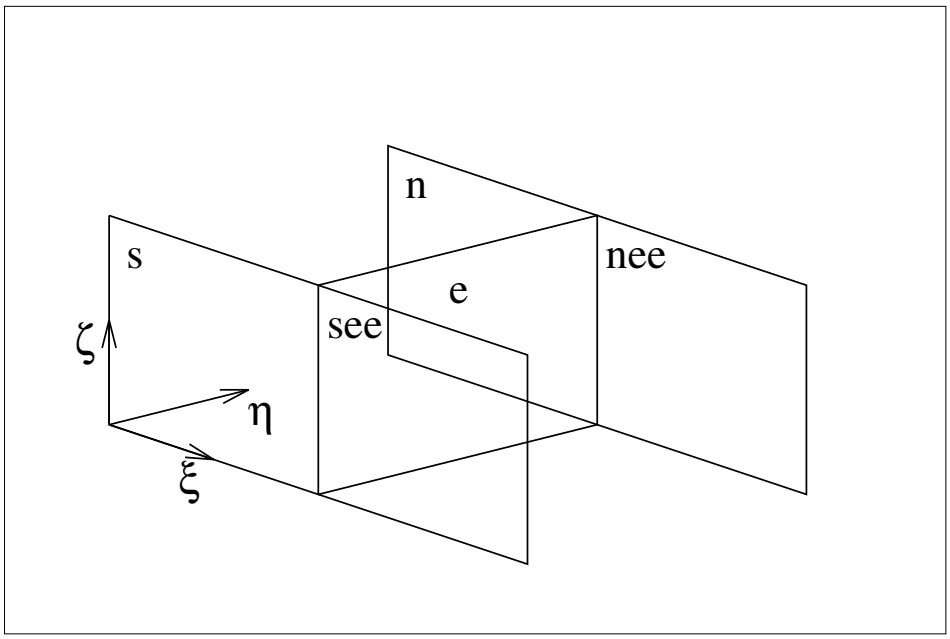

Figure 9. Interpolation of $(\alpha_{\eta i})_e$ from the surrounding four $\alpha_{\eta i}$, where i can be x, y or z.

8.5 Interpolation factors

Whenever a value of a variable is needed at the centre of a cell face, interpolation between cell-centre values will be used. In the present implementation linear interpolation is used for this purpose with an interpolation factor of one half.

Taking a two-dimensional example Fig. 10, the error introduced by using a interpolation factor of one half instead of the proper interpolation factor will be examined.

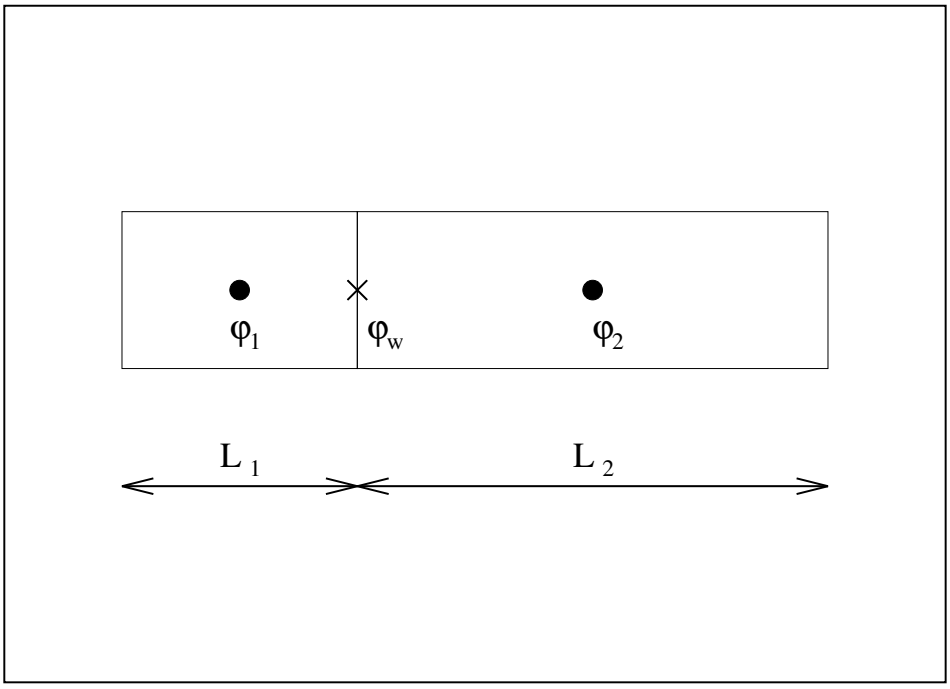

Figure 10. Interpolation between cell centres for a two-dimensional non skewed mesh.

Using linear interpolation the following expression for the cell face value, φ_w, can be derived

$$\varphi_w^{lin} = \left(1 - \frac{L_1}{L_1 + L_2}\right)\varphi_1 + \frac{L_1}{L_1 + L_2}\varphi_2 \ . \tag{123}$$

Using the alternative procedure, by taking the cell face value to be equal to the mean of the two neighbouring cell centre values, we get

$$\varphi_w^{half} = \frac{1}{2}\left(\varphi_1 + \varphi_2\right) \ . \tag{124}$$

Assuming the linear interpolation to give the correct answer, we get the following expression for the relative error introduced by using the mean value

$$\frac{\varphi_w^{lin} - \varphi_w^{half}}{\varphi_w^{lin}} = \frac{\frac{L_1}{L_2} + \frac{\varphi_2}{\varphi_1} - \frac{1}{2}\left(1 + \frac{L_1}{L_2}\right)\left(1 + \frac{\varphi_2}{\varphi_1}\right)}{\frac{L_1}{L_2} + \frac{\varphi_2}{\varphi_1}} \ , \tag{125}$$

where L_2/L_1 is the cell expansion ratio, which typical is kept within the range of [0.9,1.10] in regions of strong gradients. And φ_2/φ_1 is the ratio between the values at the cell centres, typical much smaller that one hundred for the main part of the computational domain.

Looking at Fig. 11, it is seen that for the range of parameters just mentioned, the introduced error is at most a few procent.

When the mesh is skewed even the linear interpolation does not give the correct answer, and as most meshes used in practical computation are skewed, the gain of using the more complicated 'correct' linear interpolation is believed to be minor.

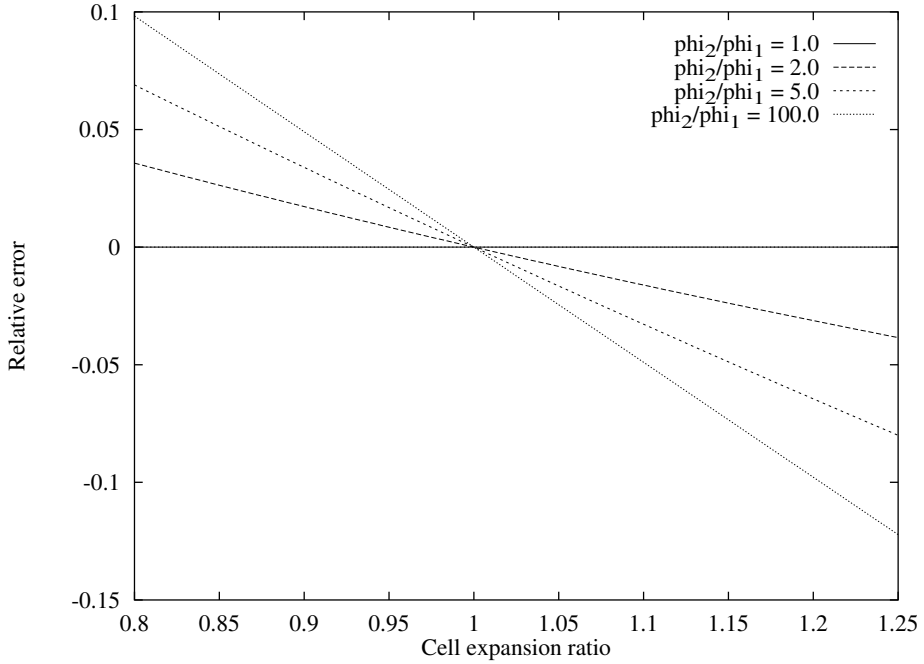

Figure 11. The relative error $(\varphi_w^{lin} - \varphi_w^{half})/\varphi_w^{lin}$ introduced by using mean values instead of linear interpolation shown as a function of the cell expansion ratio L_2/L_1 for four different values of φ_2/φ_1.

8.6 Closure

The cell-volume and the primary cofactors will be computed only once, namely at the start of the calculation, and then stored for use during the remaining computation. This is done to achieve a good performance of the program, as these

quantities are needed several times when the transport equations are assembled, and knowing that the evaluation of these quantities is very time consuming.

For the remaining geometric information, the secondary cofactors, the normal vectors, and distances between cell-face centres and cell centres, the main consideration will be saving space instead of attaining maximum speed. Therefore these will be computed whenever needed during computation, increasing the computational time at the expense of saving space.

The linear interpolation will be replaced by the mean value, as it was found that the error introduced by this practice was of minor importance.

Different techniques to obtain the meshes necessary for the present method, used to describe the geometry and obtain the geometric quantities, will be discussed in the following chapter.

9 Multiblock concept

9.1 Introduction

When working with a standard structured mesh, only simple geometries represented by a 'box' in computational space (ξ, η, ζ) can be handled. Consequently, the geometrical flexibility is rather limited, and mesh generation for moderate complex domains can be a very difficult task.

To avoid this well-known limitation of structured meshes, we choose to work with block structured meshes. The implications of this choice will be explained in the present chapter.

Even though a separate two-dimensional multiblock code was initially developed in order to become acquainted with the multiblock philosophy, the present two- and three-dimensional codes are based on the Basis2D/3D platform developed at the Department of Fluid Mechanics at DTU by Michelsen [33].

The following sections will discuss the details of the Basis2D/3D platform, relevant to the development of finite volume codes.

9.2 Multiblock meshes

In order to circumvent the fact that the computational domain must be hexahedral, the multiblock technique makes use of one, two or several of these individual computational domains together with a communication technique to exchange information between the individual domains.

The use of the block-structured approach can be illustrated most easily by an example. Considering the flow over a backward facing step, see Fig. 12, the procedure is as follows. First, the physical domain is partitioned into three subdomains, the inlet part and the upper and lower outlet part. The individual subdomains are subsequently meshed separately. As a consequence of the partitioning of the flow domain, simple rectangular meshes can be constructed in the individual subdomains in contrast to the curvilinear mesh that would have been used for a single block domain.

In order to fulfill the Basis2D/3D standards, the block splitting must obey certain rules. The first rule is that all blocks must be cubic in the computational space (quadratic in two dimensions), meaning that the number of cells in the ξ, η and ζ directions must be identical, and secondly that the the individual blocks must be of the same size. At block/block interface the mesh-lines must be c_0-continuous.

The requirement that the cell vertices must coincide at block/block interfaces is not a limitation for a cell centered finite-volume method as used in the present approach. Instead, it may be seen as a way of ensuring that the conservative property of the method is retained.

Even though the demand for equally sized cubic blocks in the computational space is not a limitation from a theoretical point of view, it may result in a large amount of small blocks degrading the efficiency of the method in extremely complex cases.

9.3 Block communication

Having subdivided the domain into subdomains, the block to block communication becomes a crucial part of the method.

Following the requirements of the Basis2D/3D standard for the domain subdivision, the block communication becomes very robust and transparent.

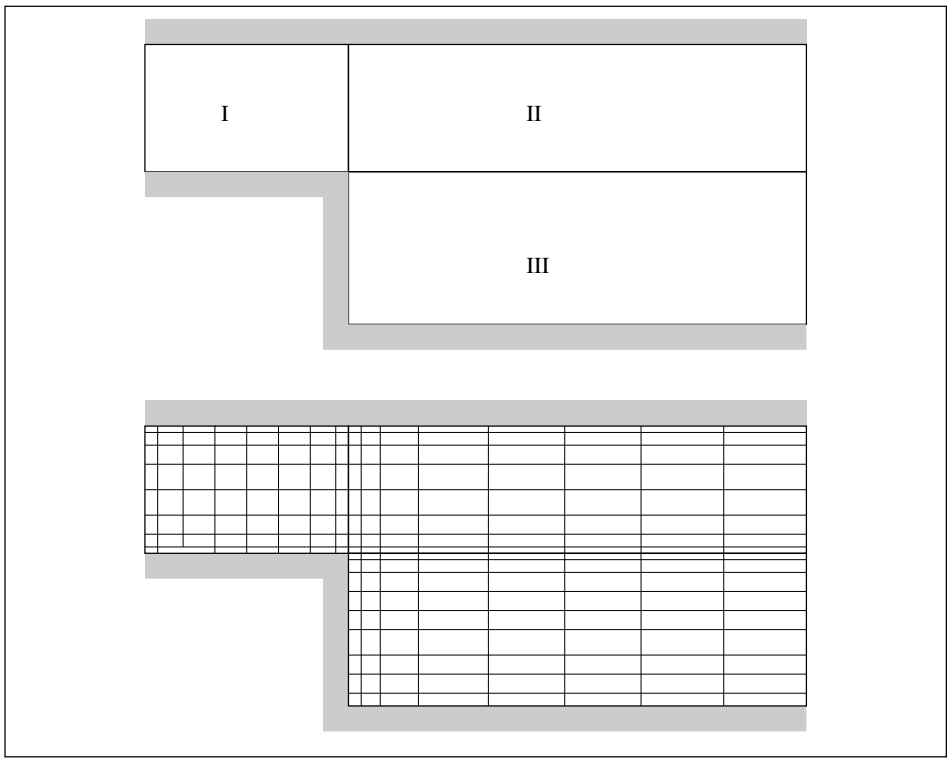

Figure 12. Backward facing step, in the upper half the geometry is shown together with the division into subblocks (I, II and III), and in the lower part the finite volume mesh is shown.

The communication between blocks is based on two elements, a single layer of cells around the blocks, the so-called ghost cells, see Fig. 13, and a communication table taking care that the correct values are installed in the ghost cells.

We will distinguish between three types of ghost cells, normal ghost cells, edge ghost cells, and corner ghost cells, see Fig. 13.

Corner ghost cells

The difference schemes used in the present three-dimensional code do not use information from the corner ghost cells. We will use this knowledge and exclude the corner ghost cells from the discussion to follow, as the corner ghost cells influence only the post-processing of the present code.

It appears from the construction of a two-dimensional case from a three-dimensional one by taking a slice for constant ζ that only normal and edge ghost cells exist for the two-dimensional case. As the corner ghost cells do not influence the three-dimensional case, the communication principle can be illustrated by the more simple two-dimensional case without loss of generality.

Normal ghost cells

The communication for normal ghost cells turns out to be simple. We only have to ensure that the values from the first inner column of the neighbouring block will be installed in the normal ghost cells, see Fig. 14. In the final code this updating of the normal ghost cells is accomplished by a single subroutine call.

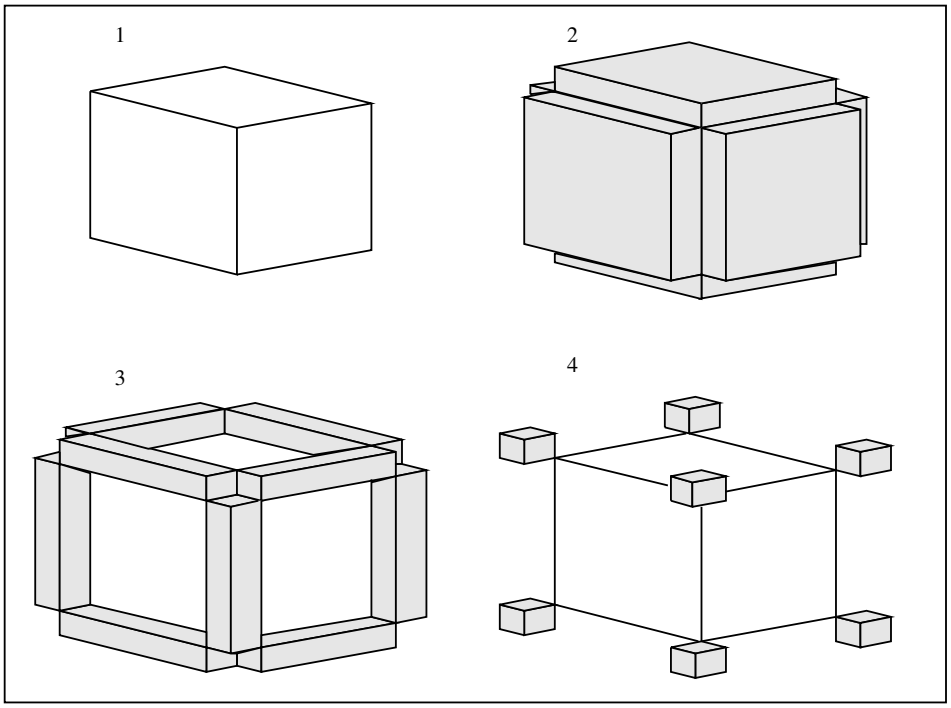

Figure 13. The figure shows the ghost cells. The block without ghost cells is shown at 1), the normal ghost cells are shown at 2), the edge ghost cells are shown at 3), and finally the corner ghost cells are shown at 4).

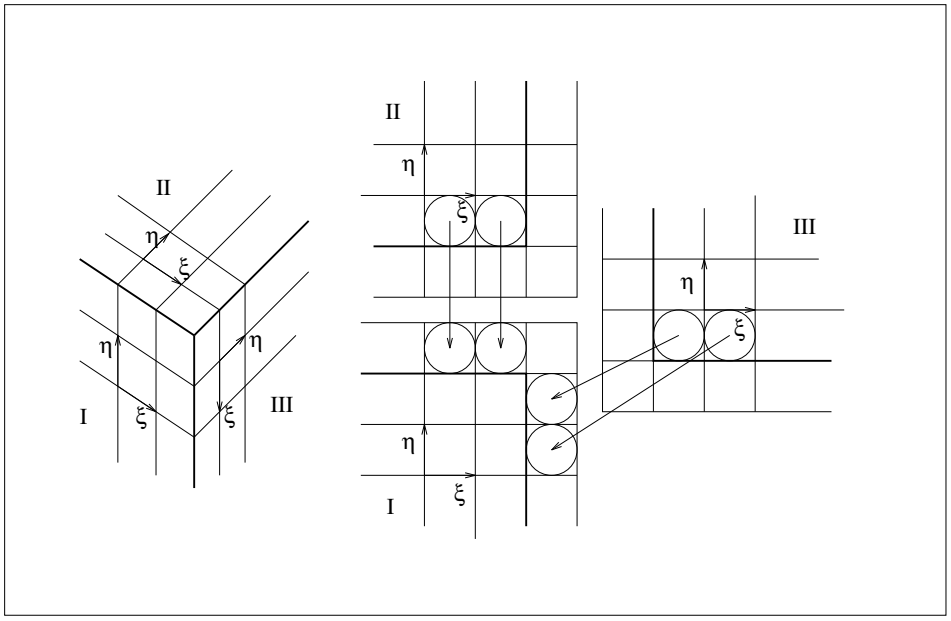

Figure 14. Movements necessary to install the right values in the normal ghost cells for block I.

Edge ghost cells

The treatment of edge ghost cells is more difficult than the normal ghost cells. This problem is related to the calculation of cross-term gradients in connection with complex topologies.

Again using the above three-block configuration, see Fig. 15, the calculation of

the cross-term gradient $\partial/\partial\eta$ at the east face of the north-east cell of block I, and the cross gradient $\partial/\partial\xi$ at the north face of the north-east cell in block I will be examined.

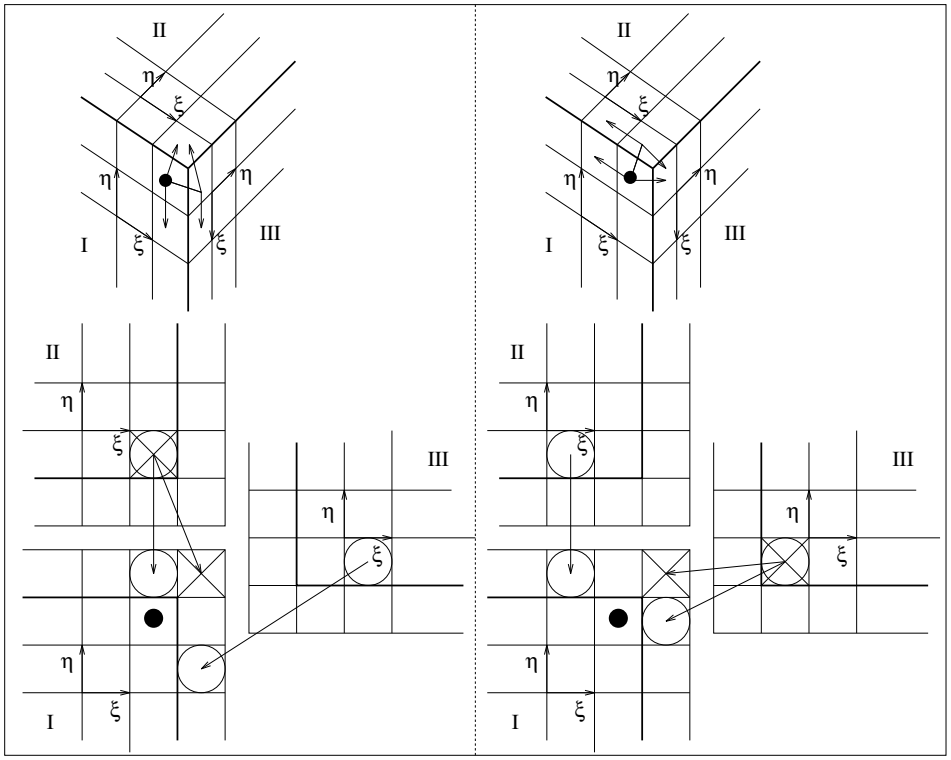

Figure 15. Calculation of cross-term gradients.The left-hand side of the figure shows the necessary movements to calculate the cross-term gradients $\partial/\partial\eta$ at the eastern face of the north-eastern cell of block I. The right hand-side of the figure shows the necessary movements to calculate the cross-term gradients $\partial/\partial\xi$ at the northern face of the north-eastern cell of block I.

From Fig. 15 it is seen that the calculations of the two cross-term gradients demand that the edge ghost cell holds two different values. It may be noted that this problem is non-existent for the normal ghost cells.

Directional splitting of the cross-term calculation

In order to save communication the following observation is utilized. The $\partial/\partial\eta$ cross-terms are needed only at the ξ-faces, whereas the $\partial/\partial\xi$ cross-terms are needed only at the η-faces. By use of this knowledge the cross-term calculations can be split by directions thereby minimizing communication. One communication call will ensure the installation of the appropriated value in the ghost cells in order to compute the $\partial/\partial\eta$ cross-terms at the ξ-faces, followed by the calculation of the $\partial/\partial\eta$ cross-terms. A second communication call will ensure the installation of the appropriated values in the ghost cells in order to compute the $\partial/\partial\xi$ cross-terms at the η-faces, followed by the calculation of the $\partial/\partial\xi$ cross-terms.

In the three-dimensional case this practice is easily extended. It is noted that even though the $\partial/\partial\eta$ and $\partial/\partial\zeta$ cross-terms are needed at the ξ-faces, the edge ghost cells holding the values for the $\partial/\partial\eta$ and $\partial/\partial\zeta$ cross-terms are different. The communication routines can therefore still be split by directions, i.e. one call to make the cross-term calculation correct for the ξ-faces, one call for the η-faces, and one call for the ζ-faces.

The values of the edge ghost cells are needed only in connection with the assembling of the cross-terms for the various equations and in connection with the Rhie/Chow interpolation. Treating the cross-terms explicitly, the communication necessary to compute them can be limited to three calls for each transport equation, plus three calls in connection with the Rhie/Chow interpolation.

9.4 Communication tables

The communication tables necessary to provide the information for the four different communication calls (three in two dimensions) must be constructed before the communication can be performed.

The simplest way of constructing this table is to code the communication table by hand for each special case. This approach was chosen in connection with the original two-dimensional multiblock test code. A more advanced but similar approach is found in Rizzi et al. [48], where the block connectivity is user specified.

The need to install the right elements into the edge ghost cells, can be very complex even for two-dimensional cases. It requires a good understanding of the actual mesh configuration and communication technique by the user. For three-dimensional cases the communication gets even more complex, and the risk of errors becomes more pronounced.

For a test code this approach may be adequate and economical, but if the goal is a production code, the need for fast reliable answers makes this strategy undesirable. In this case it may prove worthwhile to program a preprocessor capable of constructing the communication tables solely from the information about the coordinates of the cell vertices.

Based on the 'relatively simple' configurations (equally sized cubic blocks, with c_0-continuous mesh lines over block boundaries) used within the Basis2D/3D framework, it has been possible to construct a robust preprocessor. The Basis2D/3D preprocessor use information about the coordinates of cell vertices only, whereby the mesh file is the only information needed in order to calculate the communication tables. On the basis of the coordinates of the cell vertices the preprocessor checks that the mesh fulfills the Basis2D/3D standard, and if no errors are found in the mesh, the block connectivity is computed and a communication table is written to a file, see [33]. Together with the file holding the communication table the mesh file is then used by the application program when the Navier-Stokes computations are performed.

9.5 Boundary conditions

Another aspect of the present codes, both the test code developed and the final two and three-dimensional codes using the Basis2D/3D platform, is the use of attributes in order to specify boundary conditions.

For a cell face at the block boundary, a table holds the attribute value telling what kind of physical condition is needed at this face location.

The physical boundary condition inlet, outlet, wall, etc., already programmed in the code is then chosen for a face position according to the value of the attribute. In this way a boundary condition once programmed will never need to be reprogrammed.

For the two-dimensional test code the attribute table was simply programmed for every single case, following the approach of the communication table. In the Basis2D/3D environment the attributes for every single vertex is given together with the coordinate information in the mesh file [33]. The Basis2D/3D platform then interprets this information and constructs the attribute tables for the cell faces.

Using the approach of storing the information about the boundary condition together with the coordinates of the vertices in the mesh file, all the essential information for a specific problem is in fact gathered in a single file. In a test situation one can easily run a series of different test cases without changing a single line of the code, simply by use of the mesh files and the communication files generated by the preprocessor. Actually it has proven very efficient to store a series of test cases in the form of the mesh file together with the computed results. Having made changes or small adjustments to the code, one simply runs through the batch of test cases checking the result with the values earlier obtained.

9.6 Mesh singularities

In some cases the use of mesh singularities can be advantageous, allowing a face of a mesh block to collapse into a point or line. As an example one can think of a polar mesh in two dimensions where a polar singularity exists at the centre of the polar mesh, see Fig. 16.

Using a cell centred finite-volume method, the appearance of mesh singularities do not pose problems, again taking the two-dimensional polar mesh as an example, see Fig. 16. The mesh singularity is located at the western block face where the block face has degenerated into a point. As the cell-face 'area' at the western face thereby has become equal to zero, the influence coefficient at the western face is already equal to zero, and no special treatment of the boundary condition is needed.

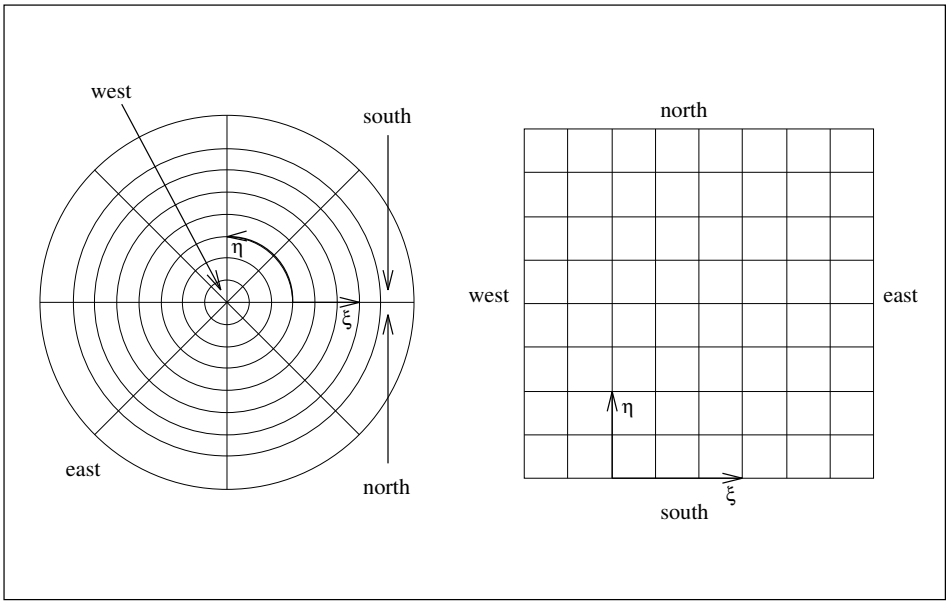

Figure 16. Polar mesh with mesh singularity, the left part of the figure shows the physical mesh and the right part shows the mesh in transformed space. The mesh singularity is located at the western block face, where the block face has degenerated to a point. As the western cell face thereby vanish, no special treatment of the boundary conditions is needed.

9.7 Solution strategy

The solution of the transport equations in a three-dimensional multiblock configuration will be obtained using a TDMA solver successively applied in alternating

directions, see also the section concerning the Schwarz Alternating Method in the chapter about multigrid. Using a TDMA successively applied in alternating directions in three dimensions, the possibility of twelve different sweep directions exists. Analyzing the possible sweep directions, one discovers that only six of these are essentially different.

By combining one TDMA line block with two sweep directions, the following six combinations have been chosen for the three-dimensional solver

$$\xi - \text{line-block} : \begin{cases} \text{bottom-west to top-east} \\ \text{top-east to bottom-west} \end{cases},$$

$$\eta - \text{line-block} : \begin{cases} \text{bottom-south to top-north} \\ \text{top-north to bottom-south} \end{cases},$$

$$\zeta - \text{line-block} : \begin{cases} \text{south-west to north-east} \\ \text{north-east to south-west} \end{cases}.$$

In order to solve a multiblock configuration the six sweep-directions will be used in the following way. Taking the sweep bottom-west to top-east for a ξ-line-block as an example, a two-step procedure is applied to all blocks. First a single sweep is made for all planes within a block, then a communication among all the blocks is performed using the standard communication call in order to ensure that the latest information is installed in all ghost cells. This communication call concludes the two-step procedure and the remaining block is treated using the same two-step procedure. When all blocks have been treated, the remaining five sweep-directions are applied in the same fashion.

As all cross-terms are treated explicitly, only the normal ghost cells need to be correctly installed during the solution procedure, limiting the communication to a single call when a block have been sweept.

The procedure may not be fully optimal, as the blocks always are solved in all directions irrespective of the coupling strength. But as the procedure has proven to be robust and computationally cheap, the gain of adding some logic in order to choose between the solution directions does not seem worthwhile.

9.8 Closure

The few components necessary to alter a single block code to a multiblock code have been described. Of these components only the ghost-cell layer and the communication table are essential. The attribute table used to specify the boundary conditions and the treatment of mesh singularities could be used in a single block code as well.

Even though the steps necessary to transform a single block code into a multiblock code are few, the changes may be difficult if the code were not originally developed for later implementation of a multiblock scheme. The two final codes were developed as single block codes, bearing in mind that later on they would be changed into multiblock codes, in fact only a few communication calls had to be added in order to use the code as a multiblock code.

As stated earlier, the multiblock capacity was originally added to the code in order to obtain geometric flexibility, but as a consequence of storing the information on the boundary conditions together with the coordinates of the cell vertices, the Basis2D/3D strategy has proven very powerful for debugging and developing codes as it allows many different cases to be calculated without changing one single line of the code.

10 Multigrid and Grid Sequence

10.1 Introduction

As mentioned earlier the solution of the pressure correction takes up a large portion of the solution time, even when the mass defects are only reduced one order of magnitude. To diminish this time and to allow mass defect reduction of two or three orders, a multigrid method was implemented in the two-dimensional multiblock test code.

In this chapter a CG multigrid method using a fixed V-cycle and a TDMA solver as basic smoother will be described. The components of the multigrid method will be addressed. Difficulties associated with combination of the multigrid method and the multiblock concept will be discussed. In this connection the Schwarz Alternating Method (SAM) will be introduced.

Inspired by the achievements of the multigrid method for the solution of the pressure correction equation, the overall method or outer loop was accelerated using a grid sequence. This was done only for the final codes implemented in the Basis2D/3D environment. Most of the tasks necessary for the grid sequence could be taken care of by standard Basis2D/3D subroutine calls.

The coupling between the Schwarz Alternating Method, used for multiblock configurations, and the multigrid technique is also discussed.

Finally, the grid sequence technique is described. This method uses the same tools as the multigrid method and is easily implemented when the multigrid routines are available.

10.2 Multigrid

In the following the correction grid (CG) multigrid method used for the solution of the linear pressure correction equation will be described.

It is well-known that standard smoothers as TDMA, point Gauss-Seidel etc. reduce the residual fluctuations on the scale of the mesh faster than the longer wavelengths. As a consequence the convergence stalls after the initial period when most of the shortwave fluctuation on the mesh scale has been removed. In a multigrid method the original fine mesh is used together with a series of coarser grids or meshes. By transferring the solution between these meshes of different coarseness, long wavelengths are smoothed on coarse meshes, while the shorter wavelengths are smoothed on the fine meshes.

The CG multigrid method developed for the two-dimensional multiblock test code consists of the following parts, a relaxation, a restriction, and a prolongation operator, together with a criterion to decide when to shift between the meshes.

The coarser meshes are constructed by standard coarsening, removing every second cell in all directions. In two dimensions this results in the coarse mesh cell being assembled of four fine mesh cells, see Fig. 17.

When descending the mesh levels i.e. restricting, it is necessary to transfer the mass source or right-hand side together with the coefficients only, as the equation solved is a correction equation.

When ascending the mesh levels i.e. prolongating, we need information on the pressure correction obtained on the coarse mesh only, and the prolongation operator thus only needs to operate on the pressure correction.

Restriction operators

Restriction of the mass source :

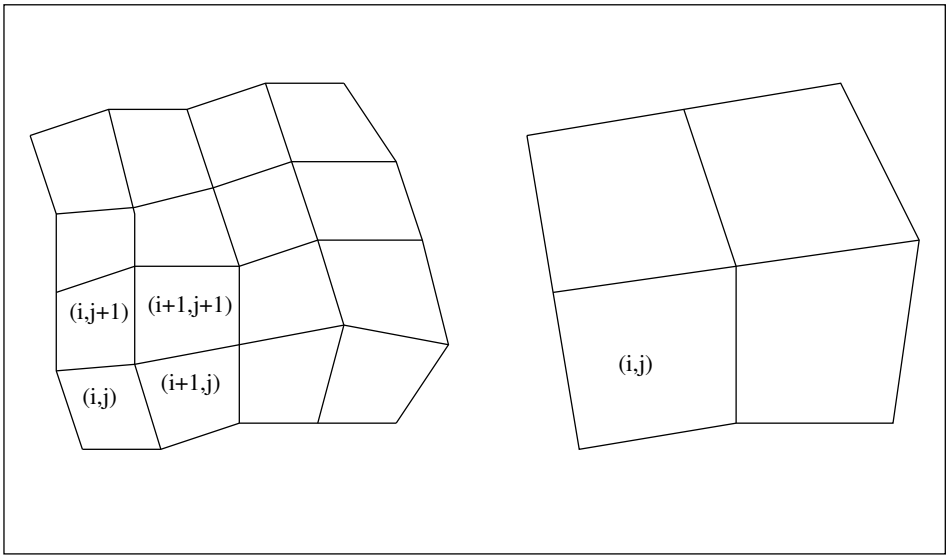

Figure 17. Construction of the coarser meshes by standard coarsening removing every second cell.

The restriction of the mass source from the fine to the coarse mesh is simply obtained by a summation of the mass sources from the four mesh cells on the fine mesh constituting the coarse mesh cell,

$$S_M^C(i,j) = S_M^F(i,j) + S_M^F(i+1,j) + S_M^F(i,j+1) + S_M^F(i+1,j+1) \ .$$

Restriction of the coefficients :

Looking at Fig. 18 it is easily seen that the following relation must be true for the eastern face

$$C_e^C = C_{e1}^F + C_{e2}^F \ ,$$

stating that the flux through the coarse mesh face is equal to the sum of the fluxes through the fine mesh faces constituting the coarse cell face. Now we express the mass fluxes by the pressure differences over the face, as proposed by Michelsen [32]

$$C_{e1}^F = A_{e1}^F(P_{E1}^F - P_{P1}^F) = A_{e1}^F \Delta P_{e1}^F \ ,$$
$$C_{e2}^F = A_{e2}^F(P_{E2}^F - P_{P2}^F) = A_{e2}^F \Delta P_{e2}^F \ ,$$
$$C_e^C = A_e^C(P_E^C - P_P^C) = A_e^C \Delta P_e^C \ .$$

Using the approximation that $\Delta P_{e1}^F = \Delta P_{e2}^F$, and that the spacing is doubled in the coarse mesh with respect to the fine mesh, we get

$$\Delta P_e^C \simeq 2\Delta P_{e1}^F \simeq 2\Delta P_{e2}^F \ .$$

Inserting this in the expression for the cell face fluxes we get

$$A_e^C = \frac{1}{2}\left(A_{e1}^F + A_{e2}^F\right) \ .$$

The generalization of this expression to the remaining faces is straightforward and will not be given here.

A prolongation operator

The prolongation of the coarse mesh results to the fine mesh needed when ascending the mesh levels, is in fact an interpolation between the coarse mesh values. The interpolation will be performed in transformed space, where bilinear interpolation is sufficient, see Fig. 19, and expressions of the following form can be derived

$$\phi^f(i,j) = \frac{9\phi^c(i,j) + 3\phi^c(i,j-1) + 3\phi^c(i-1,j) + \phi^c(i-1,j-1)}{16} \ .$$

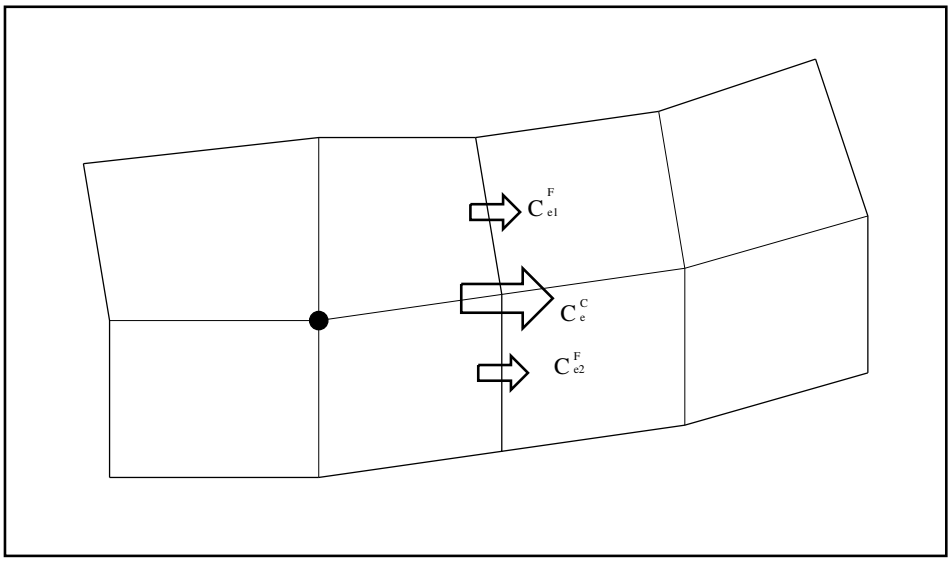

Figure 18. The coarse cell flux (C_e^C) is equal to the sum of the fluxes over the two fine cell faces C_{e1}^F and C_{e2}^F constituting the eastern face of the coarse cell.

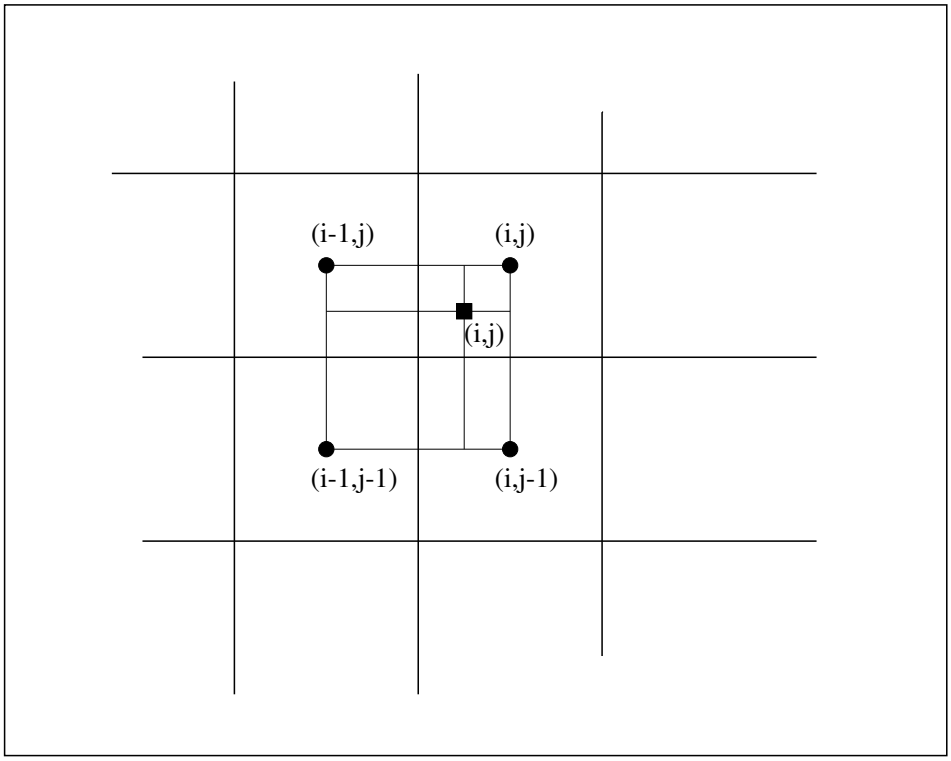

Figure 19. When prolongating coarse mesh quantities to the fine mesh, the prolongation is performed by a simple bilinear interpolation between the coarse mesh values. The fine cell value is indicated by the square and the coarse cell values are indicated by the four dots.

A relaxation operator

The relaxation operator or basic smoother used in the present multigrid method is the TDMA solver successively applied in alternating direction. The efficiency of the TDMA applied successively in alternating direction is reduced when the cell aspect ratio becomes large. As the present multigrid method was intended

primarily for testing purposes, this was not considered to be a major problem. If an efficient solver is needed even on meshes with large aspect ratios, a more advanced solver than the TDMA could be implemented as were done by Michelsen [34] in connection with the Basis2D/3D platforms.

Multigrid cycle

The multigrid cycle, i.e. the pattern in which the mesh levels are visited during solution, has to be decided. Two main options exist, using either a fixed cycle, or alternatively, making the cycle dependent on the behaviour of the convergence. In the present implementation a fixed V-cycle was chosen, following the discussion of Michelsen [31]. The number of smoothing sweeps performed on each level when descending and ascending the mesh can be seen in Fig. 20.

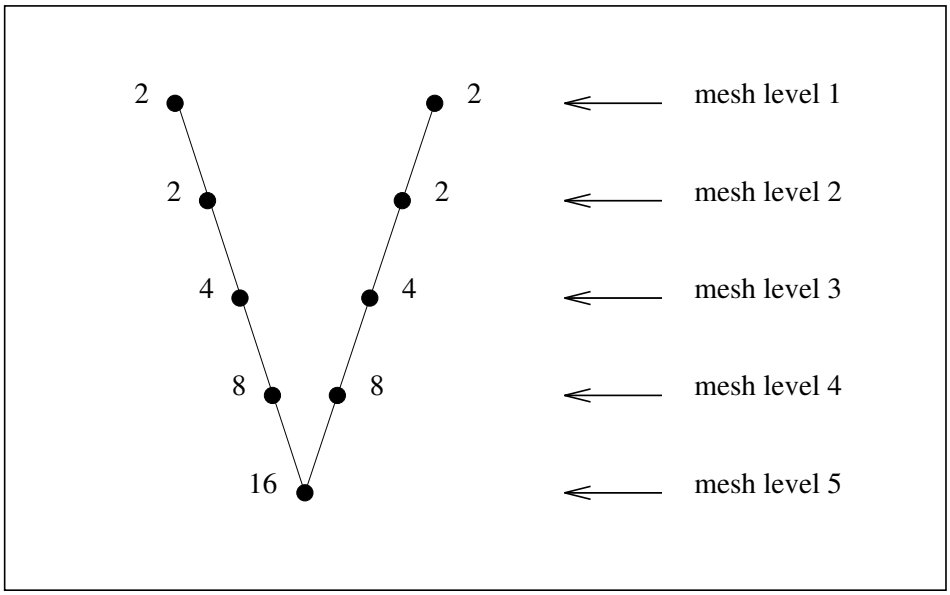

Figure 20. The fixed V-cycle for the multigrid solver, the number of relaxation sweeps performed on a given mesh level are indicated.

10.3 Schwarz Alternating Method

The solution of the multiblock domain is obtained using the Schwarz Alternating Method (SAM), see Schwarz [51]. SAM is a technique for solving a large domain as a coupled problem of several small domains, where the small domains may overlap. In the present work the domain decomposition technique was used to overcome the limited geometric flexibility of the standard structured mesh. Another frequent reason for applying domain decomposition is the wish to use parallel computers.

SAM can be applied in two different ways, i.e. multiplicative or additive SAM respectively. In the multiplicative SAM the individual domains (including the overlap) are solved in a sequential fashion, meaning that the individual domain always uses the latest information in the overlap region. In the additive SAM, all blocks are solved using the old information for the overlap region followed by an update of the overlap region when all blocks have been solved. The fact that all blocks use the old information, allows the blocks to be solved simultaneously, making this technique very well suited for parallel processing.

Combining the multigrid method with the Schwarz method, one will naturally arrive at either a simultaneous or a sequential Multigrid within Multiblock strat-

egy. If we relax the demand of the Schwarz method so that the individual blocks are not solved but only relaxed during each visit, we can arrive at two additional techniques, a simultaneous or a sequential Multiblock within Multigrid strategy. A schematic representation of the four different strategies is shown in Fig. 21, where the treatment of a domain decomposed into two subdomains and using three mesh levels in the multigrid method is shown.

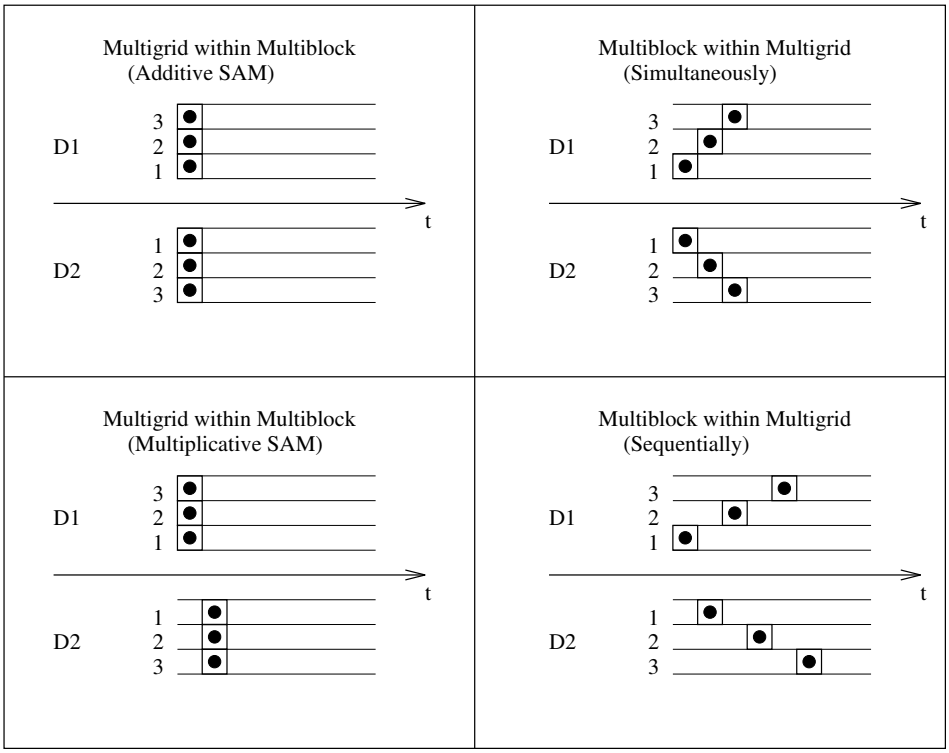

Figure 21. Four different ways of solving multigrid in connection with a multiblock configuration. The figure shows the solution of a two-domain configuration having three multigrid levels. Above the time axis domain 1 (D1) is shown, and below domain 2 (D2) is shown. The vertical columns indicate events where information is exchanged only within the individual domains, whereas information is exchanged between the domains when shifting to a new time column.

Using the multigrid within multiblock strategy, the individual blocks are solved simultaneously or sequentially. The simultaneous approach is identical to the additive SAM, whereas the sequential approach is identical to the multiplicative SAM. For the additive SAM it is said by Wei [63] that the use of the simultaneous Multiblock within Multigrid gives a better convergence than the standard additive SAM. In the same way the use of the sequentially Multiblock within Multigrid an improvement is found in convergence compared to the multiplicative SAM by Michelsen [34].

In the present work the sequential Multiblock within Multigrid domain decomposition technique was implemented. For the momentum equation an equivalent technique was used on a single grid, where the sweep directions of the TDMA take the place of the mesh levels in the multigrid method.

Domain overlap

Another important aspect of the Domain Decomposition Technique is the overlap between the individual blocks. In connection with the treatment of the momen-

tum equations (and additional transport equations), an overlap of a single cell is sufficient, using the TDMA solver.

A single cell overlap definitely proves ineffective when solving the pressure correction equation using the multigrid solver described in the present chapter. Instead a large overlap between the blocks is used, and relying on the results of Wei [63], an overlap of half a block is used.

10.4 Grid sequence

In the final two- and three-dimensional codes, where the multigrid solver and necessary tools for the SAM were supplied by the Basis2D/3D platforms, a grid sequence was implemented for the flow solver using the standard multigrid tools.

A grid sequence can be seen as an easy way to get a good start guess. In the present codes all information on boundary conditions and initial flow fields are given for the finest mesh level. The grid sequence is started by restricting this information to the coarsest mesh [4]. The problem is then solved on the coarse mesh, typically using a convergence criterion that is one hundred times larger than the convergence criterion on the fine mesh. When the convergence criterion is fulfilled, the coarse mesh solution is interpolated to the next mesh where the solution is continued. This procedure is continued until the fine mesh is reached, and the original convergence criterion is fulfilled.

Using the grid sequence technique, there is no need for a very accurate start guess on the fine mesh, a constant field will often suffice. The start guess obtained by the grid sequence typically reflects a lot of features of the final solution, for example recirculating regions and qualitatively correct velocity profiles.

As the standard coarsening in three dimensions reduces the number of mesh cells by a factor of eight (a factor of four in two dimensions), the work needed for solving the coarse mesh levels is only a fraction of the fine mesh solution. A 50% reduction in overall computing time has been recorded in 3D, using the three level grid sequence.

10.5 Closure

The multigrid solver implemented in the two-dimensional multiblock test code was described. The basic description of the restriction and prolongation tools is adequate for the final two- and three-dimensional code implemented in the Basis2D/3D environment as well. The acceleration obtained by the multigrid method is primarily a result of the coarse mesh levels permitting smoothing of longer wavelengths of the residual fluctuations than does a single grid solver.

The Schwarz Alternating Method was briefly described, and the problem of degradation of the multigrid convergence when used in a multiblock environment without overlap was mentioned.

Finally, the grid sequence technique was described. This technique removes the need of a good starting guess in order to obtain a fast solution, as the solutions of the coarse mesh levels provide this at very low computational expense.

[4] For the present code it is possible to use up to three mesh levels, depending on the number of cells in the finest mesh.

11 Mesh generation

11.1 Introduction

The meshes necessary to obtain flow solutions can be generated in many different ways, only the few methods used in the present work will be discussed here.

For geometrical simple configurations, such as flow around a sphere and flow in a driven cavity, a simple analytical prescription suffice to generate a mesh.

For more general geometries, such as flow over natural terrain and flow over surface mounted obstacles, the analytical prescription technique is not adequate. For this type of flow cases many advanced methods exist for mesh generation, e.g. elliptic mesh generation (Thompson et al. [59]), hyperbolic mesh generation (Steger and Chaussee [53]), (Steger and Sorenson [54]), conformal mapping [59], and variations of transfinite interpolation [59].

Three of the mentioned techniques have been used for mesh generation in the present work. For simple geometries analytical mesh generations have been used. To generate meshes over smoothly varying terrain, where the hyperbolic mesh generation is fast and produces meshes of a good quality, this method has been used. For flow around surface mounted obstacles and for the staggered tube bank in two dimensions, a simple variant of the transfinite interpolation technique has been used with good results.

In the following the coordinate variation in the transformed space will be given by $\xi \in$ [1,ni], $\eta \in$ [1,nj], and $\zeta \in$ [1,nk].

The generation of meshes by an analytical prescription will not be addressed here as the method is straightforward to apply, instead the effort will be concentrated on the two remaining techniques.

11.2 Hyperbolic mesh generation

Hyperbolic mesh generation is based on a hyperbolic equation system, which is a system of equations that can be solved in a marching fashion.

The hyperbolic technique is well suited for geometries where the mesh quality near the surface is of primary concern, and where the exact location of the far-field mesh is unimportant.

To illustrate the method, mesh generation over a natural terrain is described. For this problem the mesh must be fine enough near the surface to represent the steep variation of the variables located here. Far away from the surface the gradients of the solution are very weak, and the cells can be larger. The actual location of the mesh points in the far-field is unimportant as long as the domain boundary is so far away that it will not disturb the flow near the wall.

First, mesh points are distributed over the terrain surface, $x(\xi,\eta,\zeta)$, $y(\xi,\eta,\zeta)$, $z(\xi,\eta,\zeta)$ for $\zeta = 1$. Then the hyperbolic equations are solved to obtain $x(\xi,\eta,\zeta)$, $y(\xi,\eta,\zeta)$, $z(\xi,\eta,\zeta)$ for $\zeta = 2$ with the appropriated boundary conditions at the horizontal boundaries. This practice is then repeated for $\zeta =$ [3,nk] where nk is the number of planes in the ζ-direction.

Hyperbolic equations

To obtain the three equations defining the hyperbolic equation system the grid lines are assumed to be orthogonal in two directions, and the cell volumes are defined by some given function.

$$\xi \perp \zeta ,$$

$$\eta \perp \zeta ,$$
$$\text{Vol} = f(\xi,\eta,\zeta) ,$$

or

$$x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta = 0 ,$$
$$x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta = 0 ,$$
$$x_\xi y_\eta z_\zeta - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta + x_\eta y_\zeta z_\xi + x_\zeta y_\xi z_\eta - x_\zeta y_\eta z_\xi = \text{Vol}(\xi,\eta,\zeta) .$$

These equations are linearized around the present plane, and in matrix notation the following equations result

$$\bar{\bar{C}}\bar{r}_\xi + \bar{\bar{B}}\bar{r}_\eta + \bar{\bar{A}}\bar{r}_\zeta = \bar{h} , \tag{126}$$

where

$$\bar{\bar{A}} = \begin{bmatrix} x^o_\xi & y^o_\xi & z^o_\xi \\ x^o_\eta & y^o_\eta & z^o_\eta \\ y^o_\xi z^o_\eta - y^o_\eta z^o_\xi & x^o_\eta z^o_\xi - x^o_\xi z^o_\eta & x^o_\xi y^o_\eta - x^o_\eta y^o_\xi \end{bmatrix} ,$$

$$\bar{\bar{B}} = \begin{bmatrix} 0 & 0 & 0 \\ x^o_\zeta & y^o_\zeta & z^o_\zeta \\ y^o_\zeta z^o_\xi - y^o_\xi z^o_\zeta & x^o_\xi z^o_\zeta - x^o_\zeta z^o_\xi & x^o_\zeta y^o_\xi - x^o_\xi y^o_\zeta \end{bmatrix} ,$$

$$\bar{\bar{C}} = \begin{bmatrix} x^o_\zeta & y^o_\zeta & z^o_\zeta \\ 0 & 0 & 0 \\ y^o_\eta z^o_\zeta - y^o_\zeta z^o_\eta & x^o_\zeta z^o_\eta - x^o_\eta z^o_\zeta & x^o_\eta y^o_\zeta - x^o_\zeta y^o_\eta \end{bmatrix} ,$$

$$\bar{h} = \begin{bmatrix} x^o_\xi x^o_\zeta + y^o_\xi y^o_\zeta + z^o_\xi z^o_\zeta \\ x^o_\eta x^o_\zeta + y^o_\eta y^o_\zeta + z^o_\eta z^o_\zeta \\ \text{Vol} + 2\text{Vol}^o \end{bmatrix} ,$$

$$\bar{r}_\xi = \begin{bmatrix} x_\xi \\ y_\xi \\ z_\xi \end{bmatrix} ,$$

$$\bar{r}_\eta = \begin{bmatrix} x_\eta \\ y_\eta \\ z_\eta \end{bmatrix} ,$$

$$\bar{r}_\zeta = \begin{bmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{bmatrix} . \tag{127}$$

The r_ξ and r_η vectors are evaluated using central differences in the plane, and the r_ζ vector is evaluated using forward difference between the present plane and the new plane.

The terms x^0_ξ, y^0_ξ, z^0_ξ and x^0_η, y^0_η and z^0_η are evaluated using central differences in the present plane, where the ^{0} indicates that the evaluation is performed in the present plane (by known values). Knowing the volumes in the present plane, the three governing equations can be solved simultaneously to give the values of x^0_ζ, y^0_ζ and z^0_ζ.

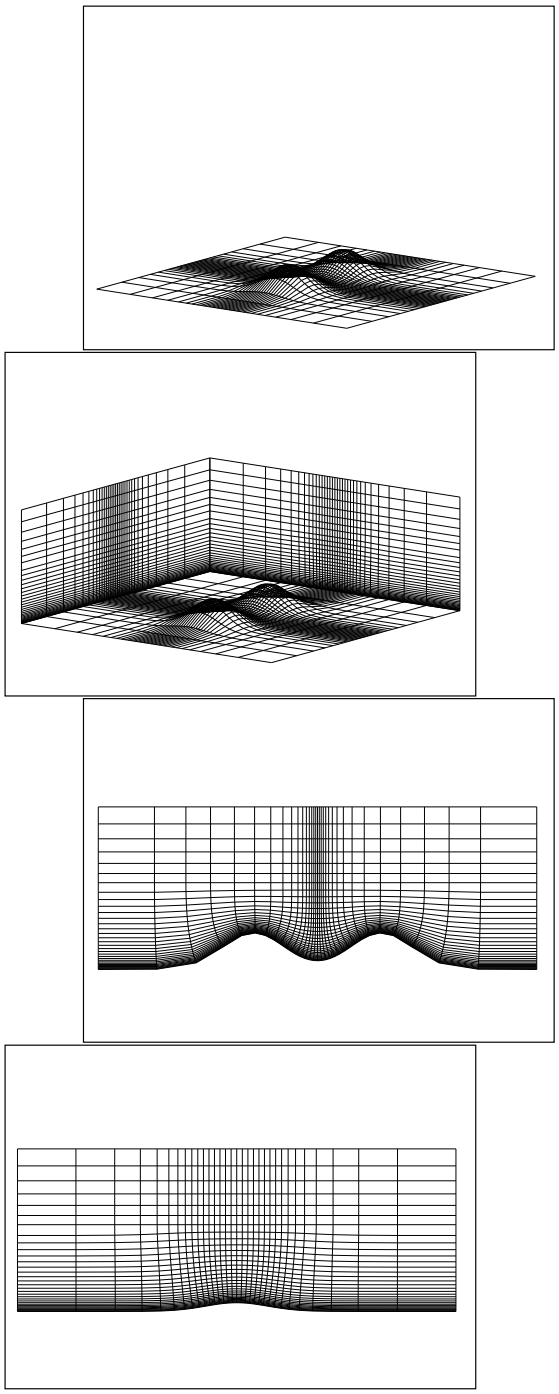

Figure 22. Hyperbolic mesh of two hills in tandem. Upper left picture shows the initial point distribution at the surface (ζ-plane). Upper right picture shows the northern and eastern vertical boundaries resulting from the hyperbolic mesh generation. Lower left picture shows the vertical mesh slice over the hill peaks (η-plane). Lower right picture shows the vertical slice between the hills (ξ-plane).

Solution of the equation system

The equation system (126) can be solved inverting a 3×3 block penta diagonal system. This approach will not be used here, instead an approximated equation

system will be used to obtain a first estimate for the coordinates of the new layer, followed by a correction using a point Gauss-Seidel solver applied successively in alternating directions.

The system approximated by using values from the present layer for $\bar{r}_\xi$ and $\bar{r}_\eta$

$$\bar{\bar{A}}\bar{r}_\zeta = \bar{h} - \bar{\bar{C}}\bar{r}^0_\xi - \bar{\bar{B}}\bar{r}^0_\eta \ .$$

This system can be solved using Cramers rule to obtain a first estimate of the coordinates for the new layer. Eventually, some smoothing of the new layer can be performed, using a simple weighting of the neighbouring coordinates.

Now $\bar{r}_\xi$ and $\bar{r}_\eta$ are recalculated using information partly from the present layer and partly from the new layer, followed by use of the Gauss-Seidel solver. Recalculation of $\bar{r}_\xi$ and $\bar{r}_\eta$ and the application of the point Gauss-Seidel sweeps are continued until equations (126) have converged. When the new layer has converged the procedure can be repeated again for the next layer and so on.

Boundary conditions and volume function

We still need to state the boundary conditions for the edges of the slice and the volume function.

When generating meshes over natural terrain, von Neumann conditions are used for the vertical coordinate, and the horizontal coordinates are allowed to slide along prescribed lines.

A good volume function is essential to generate a good mesh, actually most of the grid control is connected with this function, although some control can be achieved by the point distribution at the bottom surface.

The cell volume function for generating meshes over natural terrain is constructed using the following ideas. In the far-field the grid cells should be of the same size, and the grid slices should be horizontal. To obtain equal sized cells in the far-field requires, a weighting between the actual cell size and the mean cell size of the present plane. In order to make the mesh planes horizontal a forcing function is used, making the cells grow fastest where the plane elevation is lowest.

$$f(i,j,k) = \left(\alpha \mathrm{Vol}_{cell} + (1-\alpha)\sum \mathrm{Vol}_{cell}\right) g(i,j,k) \ ,$$

where the forcing function $g(i,j,k)$ is given by

$$g(i,j,k) = \frac{z_{max} - z}{hmax^2_{cell} + z_{max} - z_{min}} \ ,$$

and z_{max} and z_{min} are the maximum and minimum z values of the previous plane, and $hmax_{cell}$ is the maximum height of the cells in the previous plane. The square of the maximum cell height entering the denominator makes the forcing function become weaker when the difference in elevation of the plane is small compared to the maximum cell height.

If the grid must expand away from the surface, the volume can be multiplied by an exponential factor

$$expand(k) = (1+\varepsilon)^k \ ,$$

where ε is a small number.

The volume function used to generate an expanding grid is

$$f(i,j,k) = \left(\alpha \mathrm{Vol}_{cell} + (1-\alpha)\sum \mathrm{Vol}_{cell}\right) g(i,j,k) expand(k) \ .$$

The hyperbolic technique has some attractive features. It is fast using only a fraction of the time needed for elliptic techniques, and it is able to make meshes with good orthogonality and smoothness.

The drawback of the method is the lack of control of the outer boundary, but concerning flow over natural terrain and other external flows this is no major drawback.

11.3 Mesh generation by transfinite interpolation

For domains where the lack of outer boundary control cannot be accepted, other practices must be advised.

A method giving reasonably good grids in many cases is a simple version of transfinite interpolation. Basically this method is a two-step procedure. First the coordinates are distributed on the surfaces of the mesh block, thereafter an interpolation is performed to obtain the values of the coordinates in the inner part of the domain.

The distribution of the mesh points at the block surface can either be done by analytical methods, or the two of the three coordinates can be distributed by analytical means followed by a bilinear interpolation of the third coordinate.

When the coordinates of the cell vertices have been specified at the six surfaces of the domain, the coordinates of the interior points are found from interpolation between the surface points, using the following formulas

$$\begin{aligned} x(i,j,k) &= x(1,j,k) + f^1_{i,j,k}\left[x(ni,j,k) - x(1,j,k)\right] \ , \\ y(i,j,k) &= y(i,1,k) + f^2_{i,j,k}\left[y(i,nj,k) - y(i,1,k)\right] \ , \\ z(i,j,k) &= z(i,j,1) + f^3_{i,j,k}\left[z(i,j,nk) - z(i,j,1)\right] \ . \end{aligned} \tag{128}$$

The interpolation functions used in the expressions are given by

$$\begin{aligned} f^1_{i,j,k} &= \frac{(nj-j)f^1_{i,1,k} + (j-1)f^1_{i,nj,k}}{nj-1} \ , \\ f^2_{i,j,k} &= \frac{(nk-k)f^2_{i,j,1} + (k-1)f^2_{i,j,nk}}{nk-1} \ , \\ f^3_{i,j,k} &= \frac{(ni-i)f^3_{1,j,k} + (i-1)f^3_{ni,j,k}}{ni-1} \ . \end{aligned} \tag{129}$$

The interpolation functions at the six surfaces $f^1_{i,1,k}$ etc., are simply calculated by interpolation between the prescribed coordinates

$$\begin{aligned} f^1_{i,1,k} &= \frac{x(i,1,k) - x(1,1,k)}{x(ni,1,k) - x(1,1,k)} \ , \\ f^1_{i,nj,k} &= \frac{x(i,nj,k) - x(1,nj,k)}{x(ni,nj,k) - x(1,nj,k)} \ , \\ f^2_{i,j,1} &= \frac{y(i,j,1) - y(i,1,1)}{y(i,nj,1) - y(i,1,1)} \ , \\ f^2_{i,j,nk} &= \frac{y(i,j,nk) - y(i,1,nk)}{y(i,nj,nk) - y(i,1,nk)} \ , \\ f^3_{1,j,k} &= \frac{z(1,j,k) - z(1,j,1)}{z(1,j,nk) - z(1,j,1)} \ , \\ f^3_{ni,j,k} &= \frac{z(ni,j,k) - z(ni,n,1)}{z(ni,j,nk) - z(ni,j,1)} \ . \end{aligned} \tag{130}$$

All mesh control is achieved through the point distribution on the domain boundaries. Attraction of mesh lines toward solid surfaces is easily obtained by the surface point distribution.

The method is very fast, as no equation system has to be solved, the internal coordinates can be obtained simply by evaluating explicit expressions. A drawback of the method is the lack of orthogonality in all but very simple cases as in the case of quadratic and polar meshes. As the present method does not need the mesh to be orthogonal, this is no serious drawback.

11.4 Closure

Three methods have been used to obtain meshes in the present work: analytical prescription, hyperbolic and transfinite interpolation techniques.

The analytical method is mentioned only as the application is straightforward. The analytical technique has been used to generate mesh around half a cylinder, see Chapter 12. The two remaining techniques were discussed is some detail, and the advantages and main limitations were stressed. For the hyperbolic technique the main limitation is the lack of control of the location of the outer boundary, and for the transfinite interpolation technique the main limitation was the lack of orthogonality.

Other transfinite interpolations techniques exist, where a good degree of orthogonality can be obtained, but as the mesh generation was not the main topic a further investigation of these techniques did not take place.

In connection with the use of multiblock codes, it is interesting that the transfinite interpolation can easily be used for generating multiblock meshes. As all coordinates are specified at the block surfaces, the domain can be divided into subdomains where the transfinite technique can be used to obtain meshes for the individual domains. As the coordinates are specified at the block surfaces, it is easily ensured that the coordinate lines are c_0-continuous over block/block interfaces, even though the higher derivatives may be discontinuous.

12 2D test cases

12.1 Introduction

To verify the correctness of the code, some standard test cases have been calculated. These represent a wide range of flows. The laminar and the turbulent part of the code were tested. The ability to work with Cartesian, general orthogonal, and general non-orthogonal grids was tested as well. The implemented domain decomposition method was also tested. The test cases were:

- Laminar steady-state Driven Square Cavity flow at Reynolds number 100 in a Cartesian grid.
- Laminar steady-state separation over a circular cylinder at Reynolds number 40 in orthogonal grid (polar coordinates).
- Turbulent steady-state channel flow at Reynolds number 61000 in a rectangular grid.
- Turbulent steady-state flow trough Staggered tube banks at Reynolds number 140000 in a general non-orthogonal grid.
- Turbulent steady-state flow over Backward-facing step at Reynolds number 140000 in a 3 domain rectangular grid.

For every test-case a table is given, here IP and JP refers to the number of points in ξ- and η-direction in the individual grid blocks, NB is the number of grid blocks. *Re* refers to the Reynolds number. Tran/Sted is whenever the calculation is transient or steady-state. DS is referring to the kind of differencing used, where CDS is central differencing scheme and UDS is upwind differencing scheme. It gives the number of iterations, and CPU(s) is the CPU time in seconds.

All calculations, except the driven cavity, were stopped when the maximal normalized residual was less than 10^{-4}, for the driven cavity the computation was stopped when all residuals had been reduced by a factor 10^{-4}.

The calculations were performed on an IBM 320h RISC 6000 work-station, with a peak Mflop rate of 50.

12.2 Driven Square Cavity

The laminar Driven Square Cavity was selected as the first major test case. This case does only use the laminar Cartesian parts of the code. The very basic elements, the Rhie-Chow interpolation, boundary conditions, and multigrid scheme can therefore be tested whitout any disturbance from the curvilinear parts.

Table 3. Computational parameters for Driven Square Cavity flow.

IP	JP	NB	*Re*	Tran/Sted	DS	It	CPU(s)
64	64	1	100	Sted	CDS	493	306

As the flow is expected to separate in the bottom corners of the cavity, the ability to calculate separation will be tested.

The Reynolds number is based on the lid velocity and the cavity height

$$Re = \frac{\rho U_{\text{lid}} H}{\mu} .$$

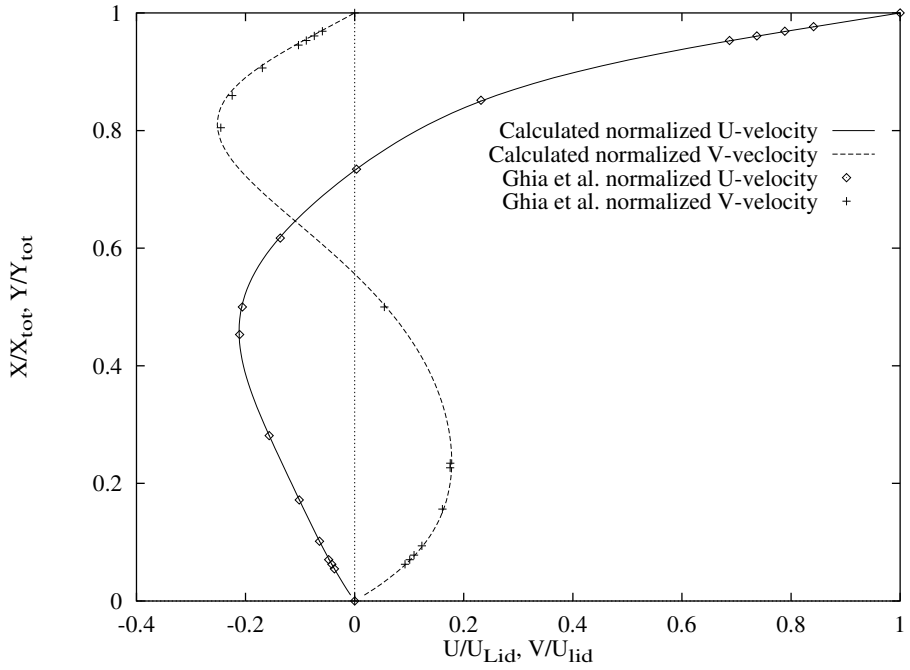

Figure 23. Driven Square Cavity, Re=100. U-velocity at x/x_{tot}=0.5, V-velocity at y/y_{tot}=0.5.

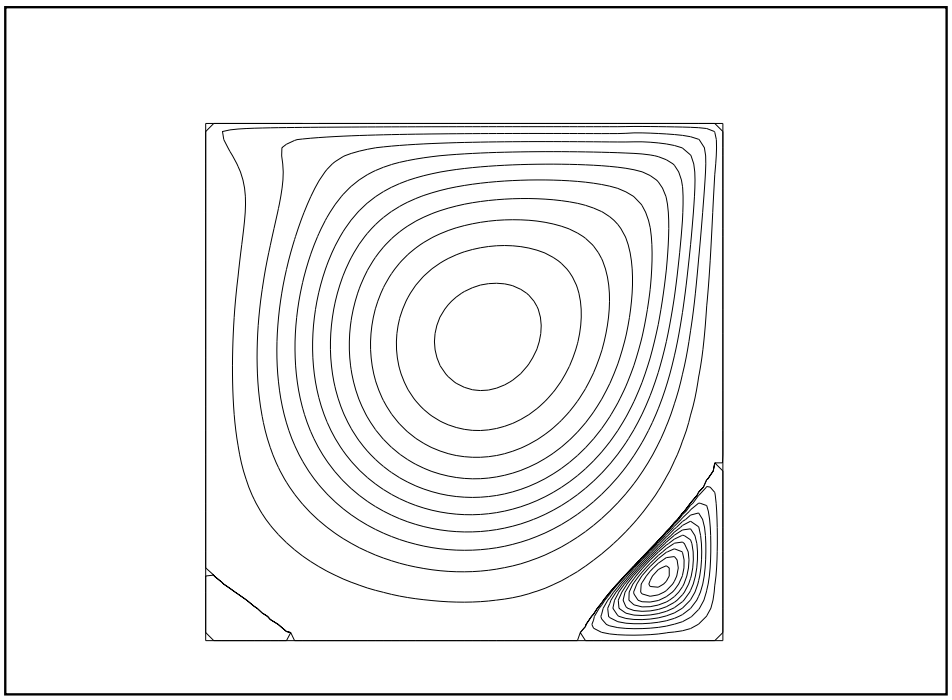

Figure 24. Stream lines for Driven Square Cavity at Re=100. The separated regions in the bottom corners are clearly seen.

The south, east and west boundaries were specified to be wall boundaries, the north boundary was specified to be a wall moving with velocity U_{lid}.

The results were compared to the calculation of Ghia [11], and according to Fig. 23, the agreement is quite good. The basic elements of the code are found to be working, and the code is able to calculate separation, see Fig. 24.

12.3 Flow over a circular cylinder

The next part of the code to be tested was the general orthogonal laminar part. To do this, the separated flow over a circular cylinder in polar coordinates was choosen.

The Reynolds number is based on the diameter of the cylinder and the free stream velocity

$$Re = \frac{\rho U_\infty D}{\mu} .$$

To save computing time the flow was assumed to be symmetric, and only half of the domain was calculated.

Table 4. Computational parameters for flow over cylinder.

IP	JP	NB	Re	Tran/Sted	DS	It	CPU(s)
64	64	1	40	Sted	UDS	269	597

To minimize the influence of the free stream boundary condition, the outer boundary was placed 30 diameters from the cylinder. The inflow boundary condition was specified over the upstream 90 degrees of the outer boundary, meaning that the velocity was set to free stream velocity here. Over the downstream 90 degrees of the outer boundary, outlet condition was specified, assuming fully developed flow.

On the cylinder surface, the no-slip boundary condition was specified, and on the horizontal pieces of the boundary, symmetry conditions were applied.

From the calculation, the pressure drag coefficient and the friction drag coefficient were calculated by the following formulas

$$C_p = \frac{\int P \sin\theta ds}{\rho U_\infty^2 R} ,$$

$$C_f = \frac{\int \tau_0 \cos\theta ds}{\rho U_\infty^2 R} .$$

The nondimensionalized surface pressure was also extracted from the calculation, by the formula

$$P(\theta) = \frac{P_0 - P_\infty}{\frac{1}{2}\rho U_\infty^2} .$$

The separation angle was determined, as the point on the cylinder where the streamline $\psi = 0$ originates.

Table 5.

	C_p	C_f	$\frac{L_{\text{recirc}}}{R}$	θ_{separ}
present computation	1.06	0.55	4.07	53.0
Dennis and Chang	0.99	0.52	4.69	53.8

The results were compared to the calculation of Dennis and Chang [9] and for the drag coefficients and separation angle the error was less than 5%; for actual values see Table 5. The nondimensionalized recirculation length L_{recirc}/R was found to be about 15% too short, the numerical diffusion of the upwind scheme

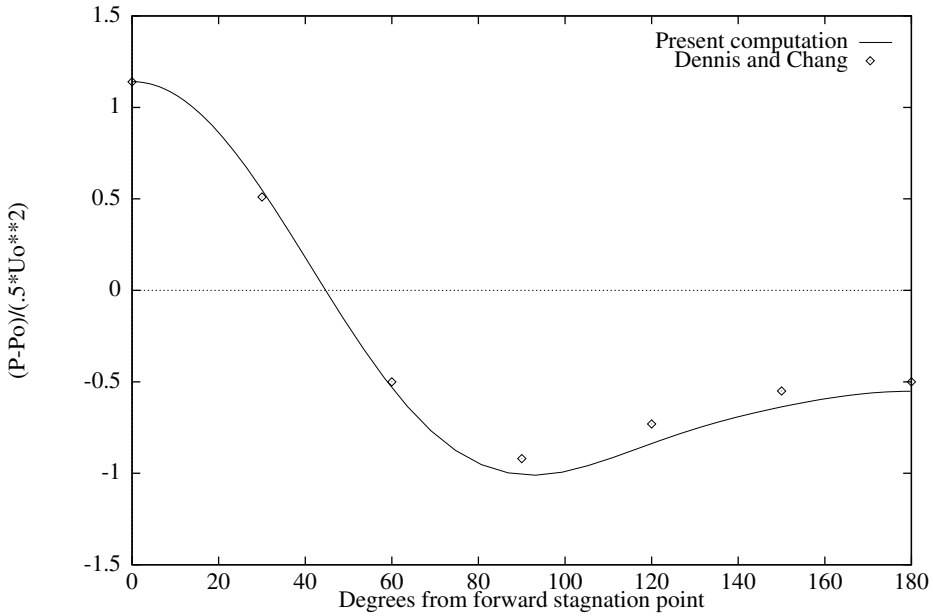

Figure 25. Surface pressure for flow over cylinder, Re=40.

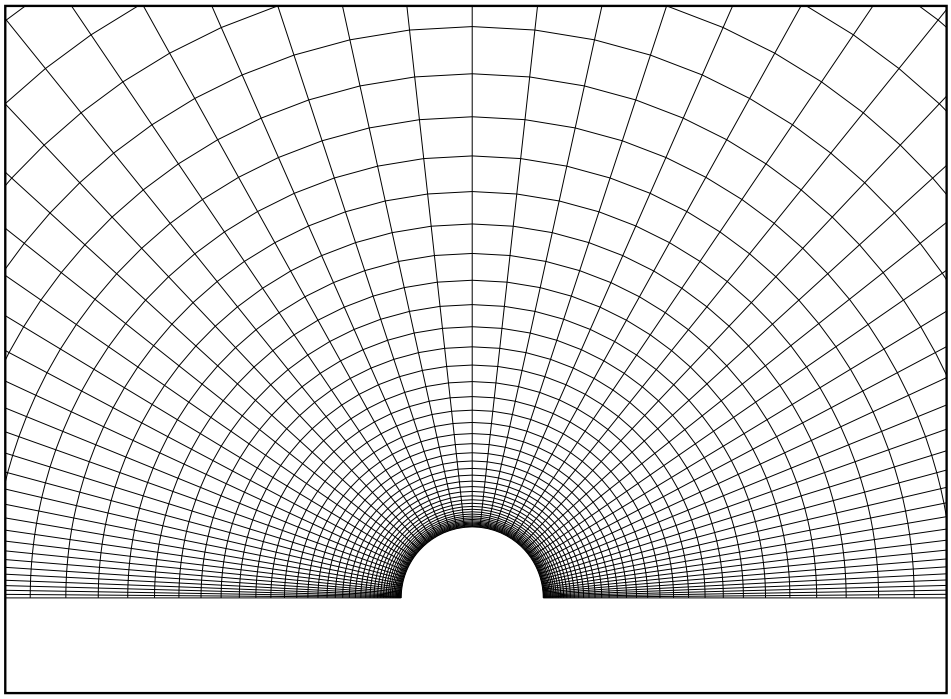

Figure 26. Flow over cylinder, detail of grid.

in regions where the grid is not aligned with the flow may be the cause of this. In the separated region the flow travels through cells under an angle of about 45 degrees, giving rise to strong numerical diffusion.

From Fig. 25 it is seen that similar agreement is found for the surface pressure distribution.

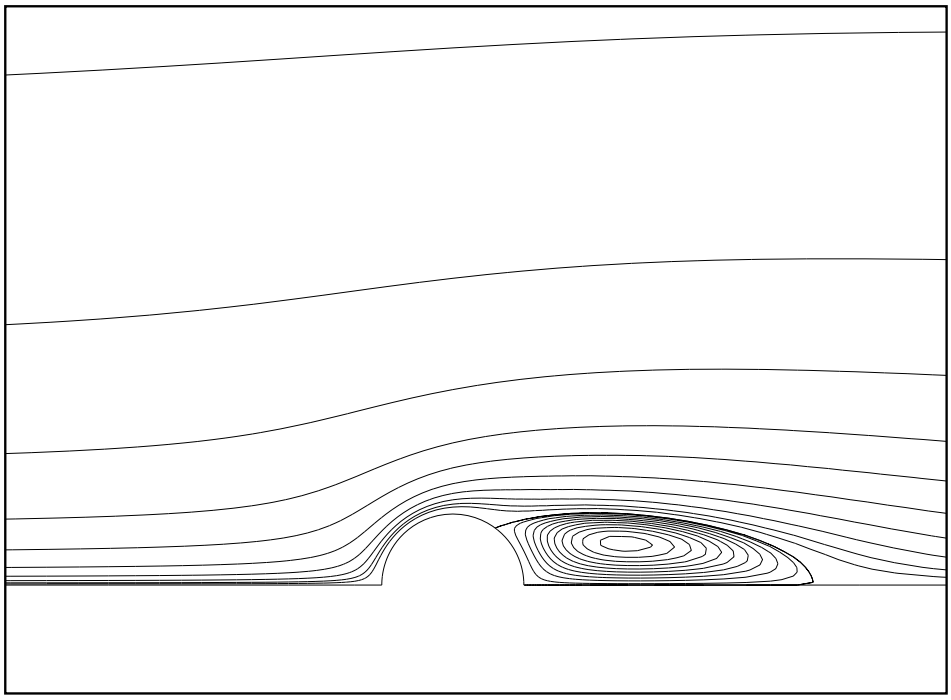

Figure 27. Flow over cylinder at Re=40, streamlines near cylinder.

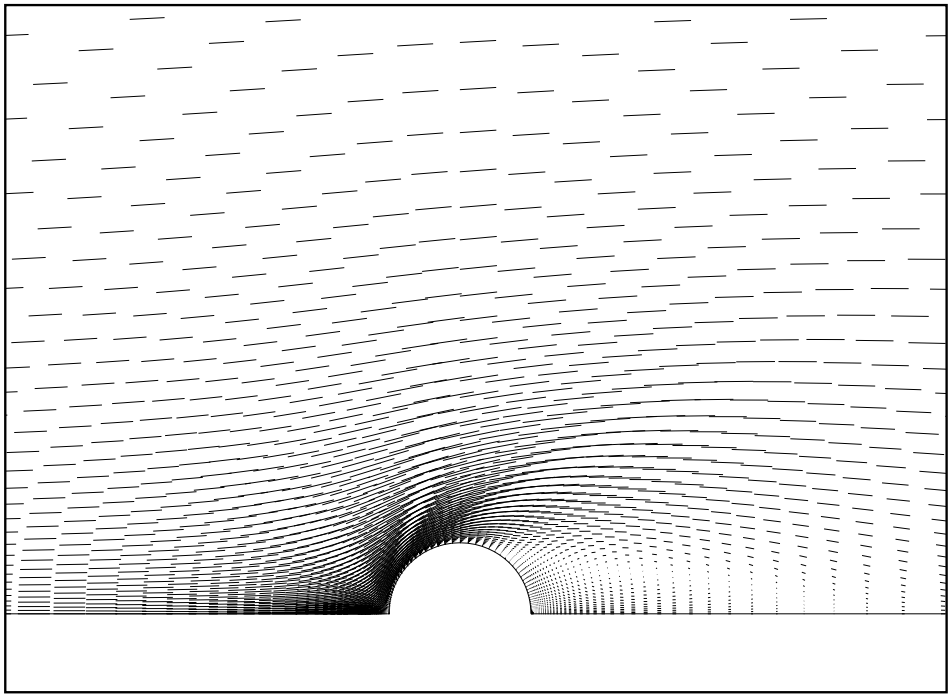

Figure 28. Flow over cylinder at Re=40, velocity vectors near cylinder.

12.4 Turbulent channel flow

To test the basics of the implemented $k - \epsilon$ model, the orthogonal part of the turbulence equations and the implemented wall law, fully developed turbulent flow in a straight channel was computed, in at Cartesian grid.

The Reynolds number is based on the half width of the channel and the maximum flow velocity

$$Re = \frac{\rho U_{\text{center}} D}{\mu} \ .$$

Table 6. Computational parameters for turbulent channel flow.

IP	JP	NB	*Re*	Tran/Sted	DS	It	CPU(s)
16	16	1	61000	Sted	UDS	242	35

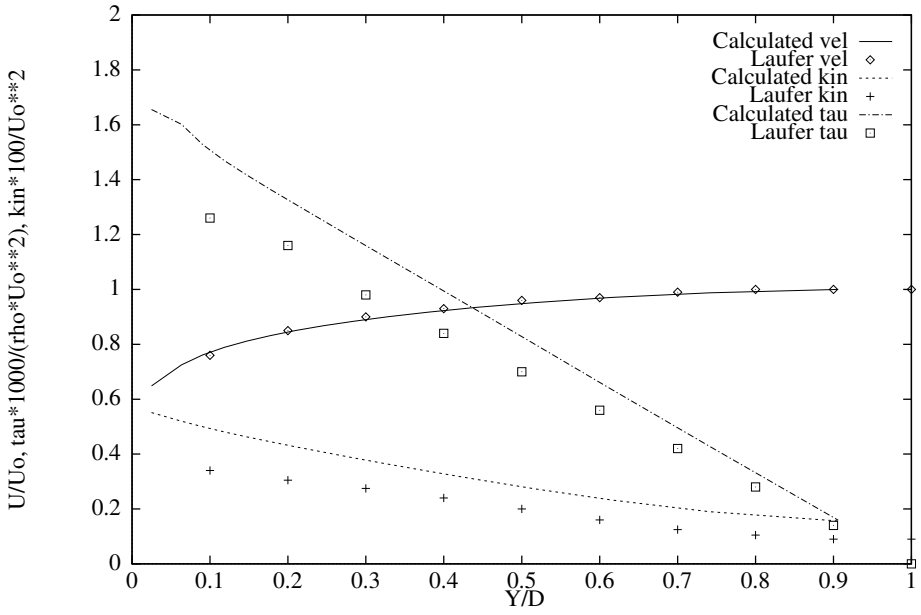

Figure 29. Turbulent fully developed channel flow, Re=61000.

The calculation was started with constant value profiles for velocities, and turbulent quantities.

$U = U_{in}$,

$V = 0$,

$k = \text{turbin} \times U_{in}^2$, turbin $= 0.01$,

$\epsilon = \frac{C_\mu^{\frac{3}{4}} k^{\frac{3}{2}}}{\kappa l}$, $l = 0.25$.

To get a fully developed flow, the outlet profile from the previous iteration was used as inlet profile for the next iteration. The inlet was moved for the first time after 10 iterations, and then after every iteration until convergence was reached. The outlet boundary condition was specified as fully developed flow, and the logarithmic law was used at the walls.

The calculation was compared to the experiment of Laufer [21], and good agreement was found for the mean velocity. The agreement for shear stress and turbulent kinetic energy is not as good, the computed shear stress is about 20% higher than the one calculated from the measured mean velocity distribution.

The fully developed profiles are shown together with the experimental values on Fig. 29.

12.5 Staggered tube bank

Turbulent flow over staggered tubes was choosen to test the general nonorthogonal part of the $k-\epsilon$ model. The more general form of the turbulent boundary conditions could be tested as well, because turbulent boundary conditions had to be applied on surfaces that were not parallel to either the x- or y-axis.

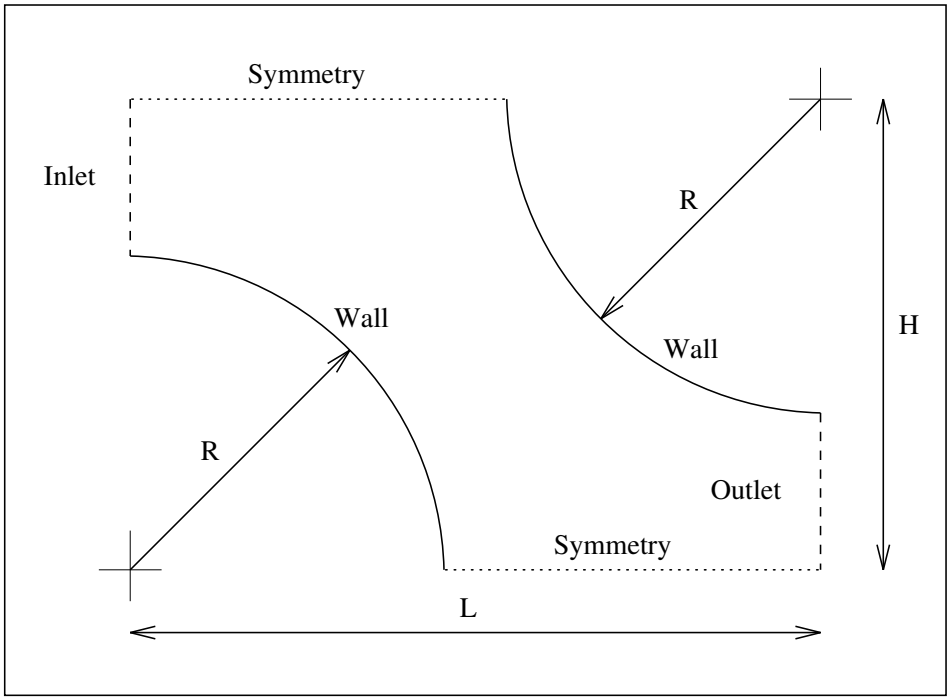

Figure 30. Anti-symmetric unit of staggered tube bank

The Reynolds number of the flow is based on the diameter of the tubes and the maximum velocity in the minimum gab between the tubes

$$Re = \frac{\rho U_{max} D}{\mu} \ .$$

Table 7. Computational parameters for staggered tube bank.

IP	JP	NB	*Re*	Tran/Sted	DS	It	CPU(s)
64	32	1	140000	Sted	UDS	2314	2660

The calculation domain is a basic anti-symmetric unit of the staggered tube bank, see Fig. 30.

The condition for using the anti-symmetric unit instead of the whole tube bank is that the flow is fully developed. This can be achieved experimentally by using a bank of many tubes assuring that the entrance region does not influence the domain in question.

Inlet boundary conditions were specified as constant value profiles for all variables

$$U = U_{in} \ ,$$

$$V = 0 \ ,$$

$k = \text{turbin} \times U_{in}^2 \qquad , \text{turbin} = 0.1 \ ,$

$\epsilon = \dfrac{C_\mu^{\frac{3}{4}} k^{\frac{3}{2}}}{\kappa l} \qquad , l = 0.01 \ .$

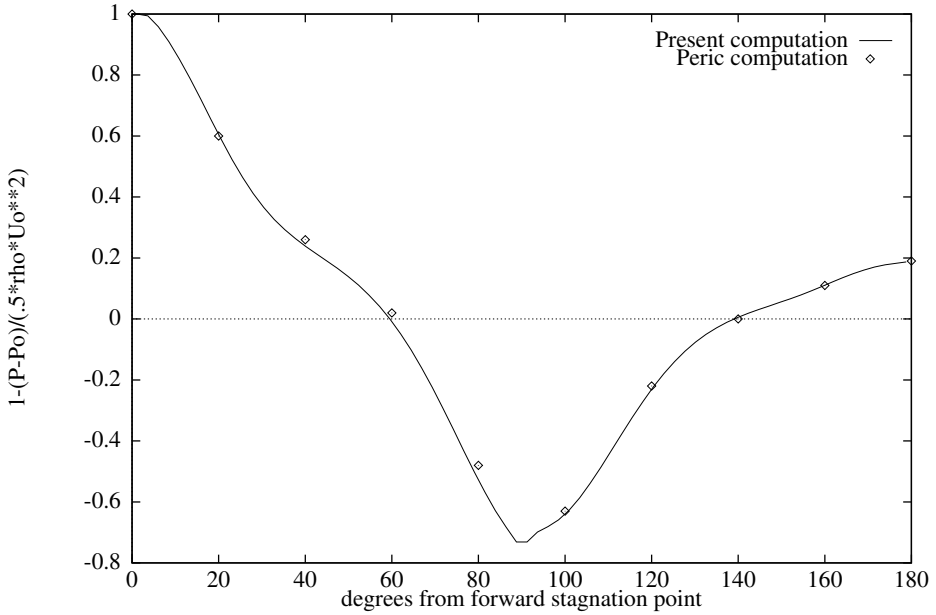

Figure 31. Pressure distribution at tube surface for staggered tube bank, Re=140000

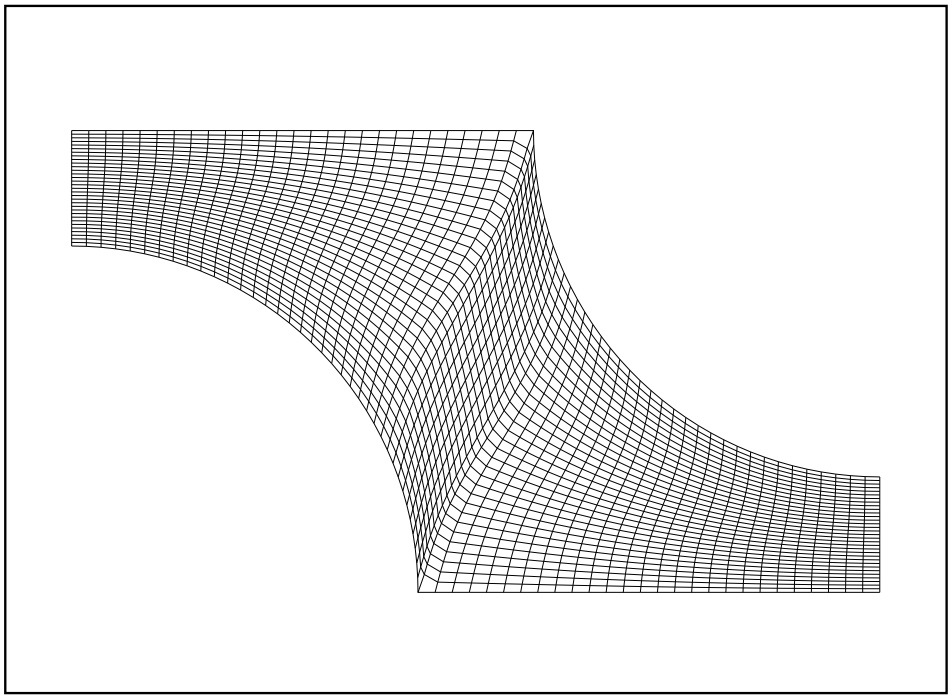

Figure 32. Staggered tube bank, grid.

To reach the fully developed flow condition the outlet conditions were moved to the inlet, as described in the case of turbulent channel flow. The only difference

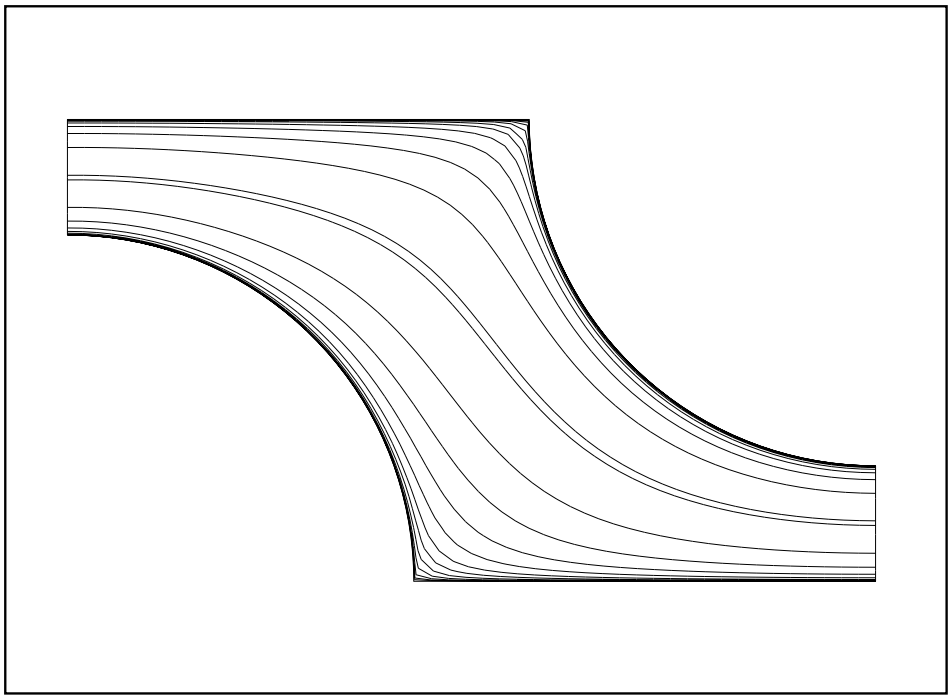

Figure 33. Staggered tube bank, streamlines, Re=140000

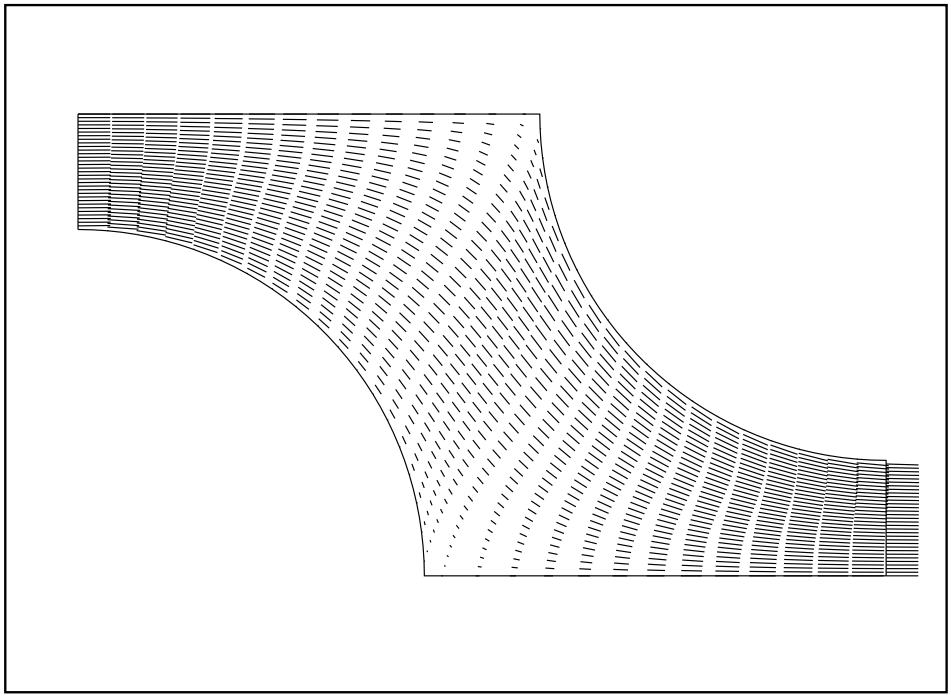

Figure 34. Staggered tube bank, velocity vectors, Re=140000

was the anti-symmetric periodic nature of the computation domain, demanding the profile coordinate to be reversed.

Outlet conditions were specified as fully developed, counting on the parabolic nature of the flow.

At the horizontal lines, symmetry conditions were applied, and on the tube

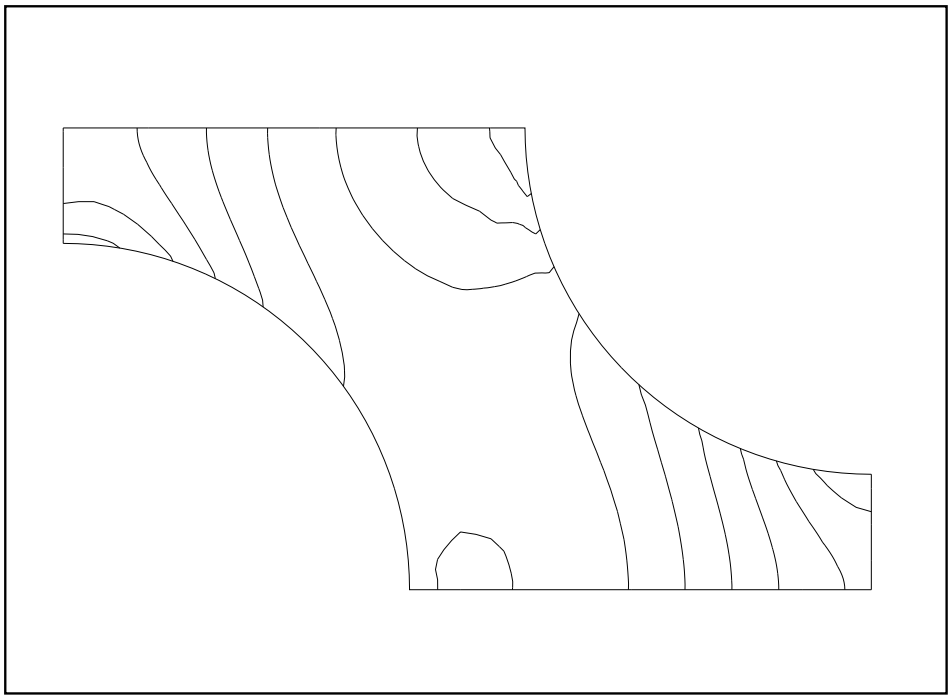

Figure 35. Staggered tube bank, pressure contours, Re=140000

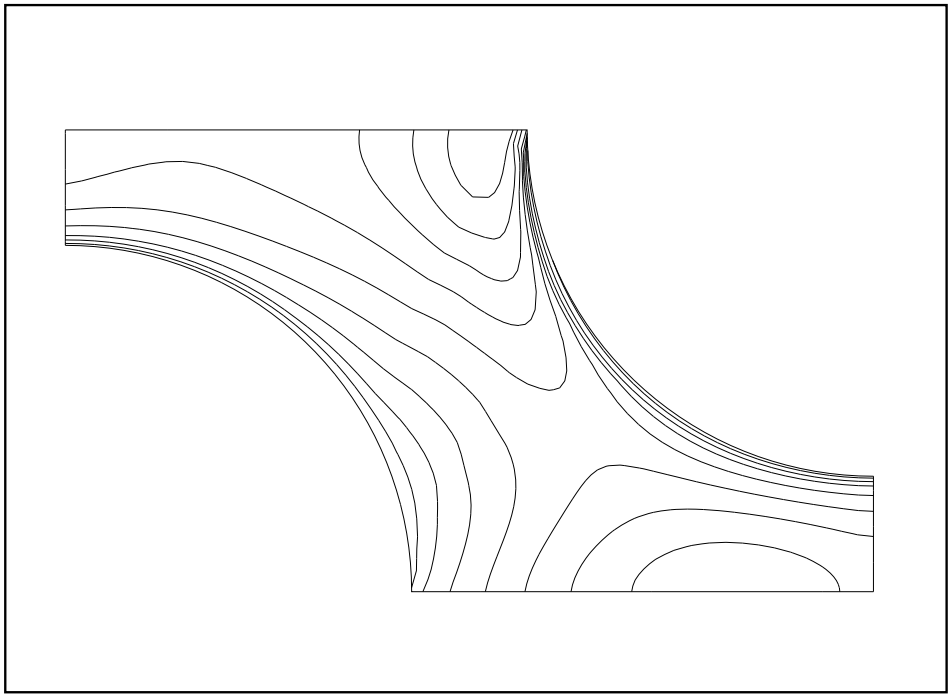

Figure 36. Staggered tube bank, kinetic energy, Re=140000

walls turbulent boundary conditions were used. The computational mesh used for the calculation was not constructed with the transfinite method described in the chapter dealing with mesh generation, instead the mesh was generated by a more advanced transfinite method allowing some control of the gradients at the surfaces, the mesh generation code using this approach was supplied by Hvid [17].

The results were compared to the calculation of Peric [45]. On Fig. 31 the obtained surface pressure distribution is shown together with the results of [45]. The agreement is quite good, only the non smooth behaviour near $\theta = 90$ degrees is not so good. This discontinuity is caused by the pressure distribution being assembled from the wall-pressure from the two tubes connected to each other at $\theta = 90$.

The three isolines plots on Figs. 33, 35 and 36 are in good agreement with the results of [45], not shown here. The most obvious difference is the lack of separation in the present computation. This can be explained by numerical diffusion in the upwind scheme suppressing the separation, and the log-law being in error in this low velocity region.

12.6 Backward facing step

Finally, the multi-domain facility of the code had to be tested. The turbulent backward facing step is well suited for this purpose.

The test case of Kim [19] was selected, unfortunately the original reference was not available but sufficient data could be taken from Hackmans calculation of the same test case [12].

Table 8. Computational parameters for backward facing step.

IP	JP	NB	*Re*	Tran/Sted	DS	It	CPU(s)
32	32	3	140000	Sted	UDS	649	1290

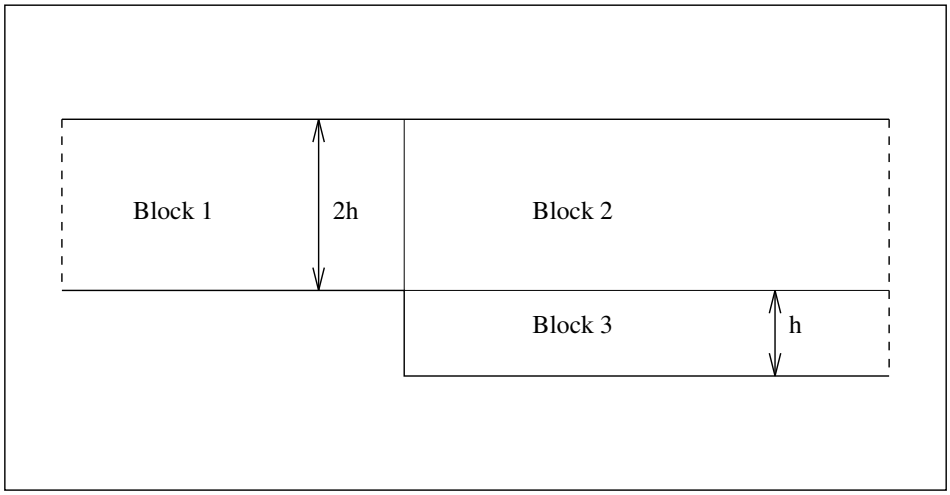

Figure 37. Geometry and block configuration for backward facing step

The geometry is shown on Fig. 37, along with the block configuration. The Reynolds number is based on twice the step height and the mean inlet velocity

$$Re = \frac{\rho U_{in} 2h}{\mu} \ .$$

The inlet profiles upstream from the step are not available in [12], instead the calculation was started far upstream from the step, about fourteen step heights, with constant value profiles for all variables.

$$U = U_{in} \ ,$$

$V = 0$,

$k = \text{turbin} \times U_{in}^2$,where turbin = 0.0001 ,

$\epsilon = \dfrac{C_\mu^{\frac{3}{4}} k^{\frac{3}{2}}}{\kappa l}$, where $l = 10 \times h$.

The turbulence intensity was then adjusted to get the profile correct at the step corner, see Fig. 39. This procedure resulted in an inlet turbulence intensity equal to 0.0001.

Turbulent boundary conditions were applied on the walls, and the outlet condition was assumed to be fully developed flow. Placing the outlet about fourteen step heights from the step, the parabolic nature of the flow assures that the outlet condition should not influence the upstream flow.

To use the multi-domain feature of the code, a three domain rectangular grid was generated instead of a single domain curvilinear, grid see Fig. 37 and Fig. 40.

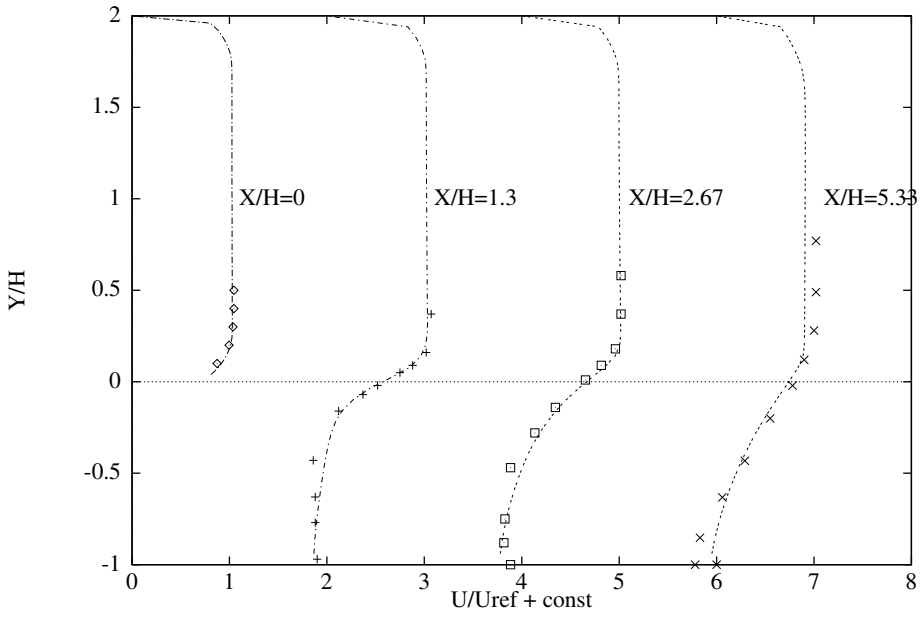

Figure 38. Velocity profiles behind the step at different X/H, Re=140000. The constants added are, const=0. for x/H=0., const=2. for x/H=1.3, const=4. for x/H=2.67, const=6. for x/H=5.33,

The results are compared to the data of [19]. The profiles one step height behind the step, are in good agreement with the measurements, but further downstream of the step the agreement gets worse, see Fig. 38. The reasons for this are partially the former mentioned numerical diffusion, and partially the turbulence model giving wrong spreading of turbulent kinetic energy in the recirculation bubble.

On Fig. 41 and Fig. 42 the streamlines and velocities are shown, the domain boundaries are drawn to show that no discontinuity arises at the block interfaces.

The nondimensionalized recirculation length 5.96 is close to the findings of [12] for grids of the same density.

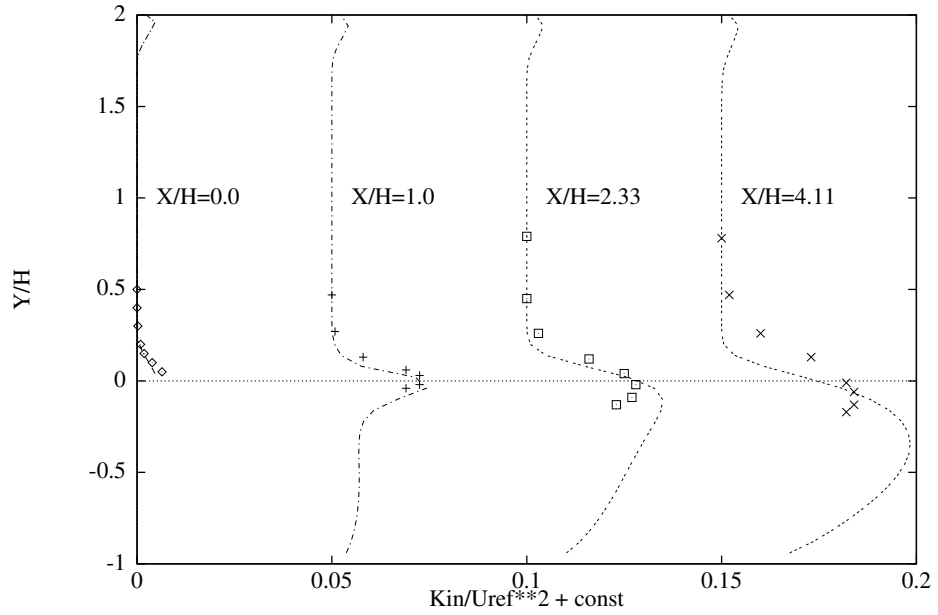

Figure 39. Turbulent kinetic energy profiles behind the step at different x/H, Re=140000. The constants added are, const=0. for x/H=0., const=.05 for x/H=1.3, const=.1 for x/H=2.67, const=.15 for x/H=5.33.

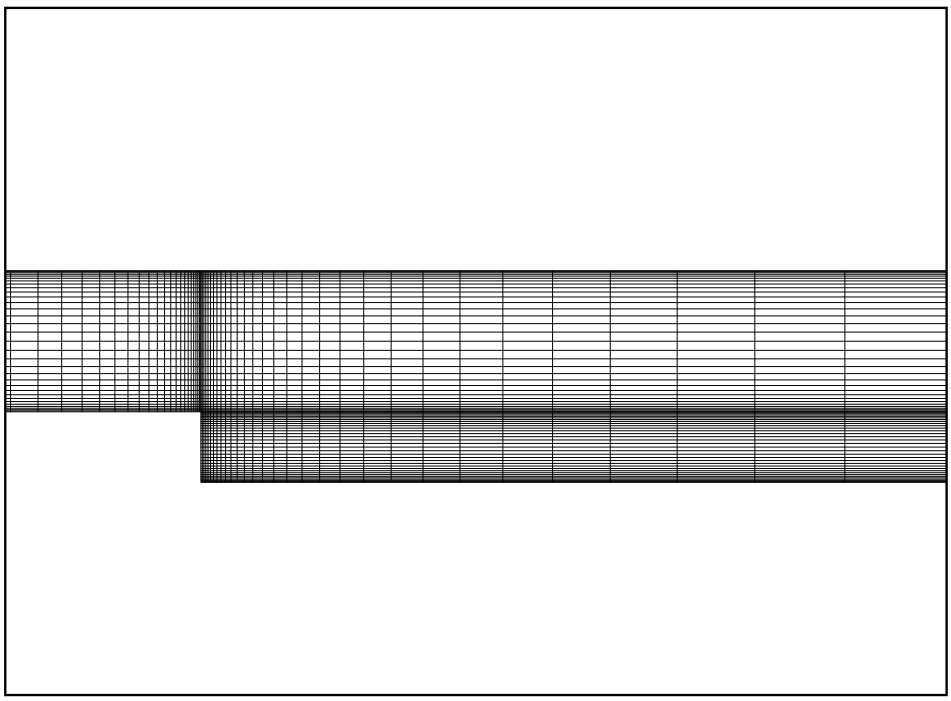

Figure 40. Backward facing step, grid detail.

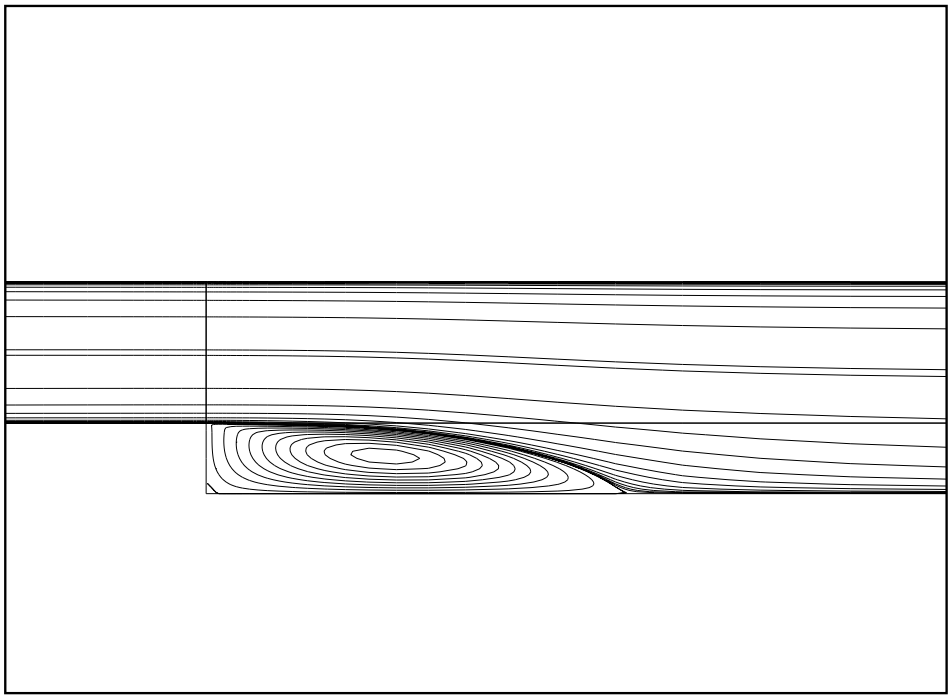

Figure 41. Backward facing step streamlines, Re=140000.

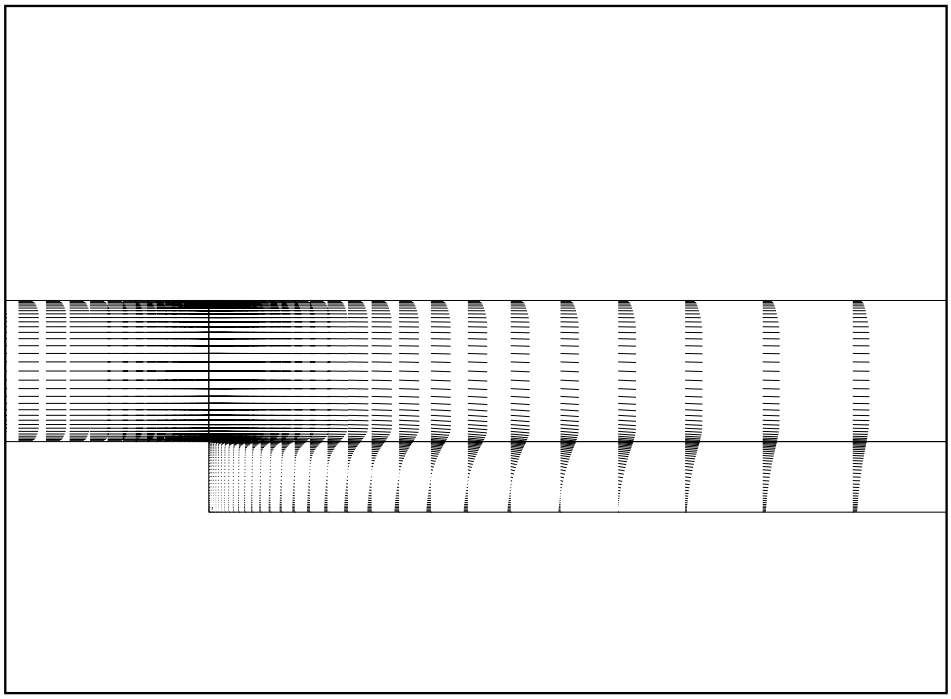

Figure 42. Backward facing step, velocity vectors, Re=140000.

12.7 Closure

As the test cases considered in the present chapter span a wide range of flow situations, from laminar flows in Cartesian meshes, to highly turbulent flows in general non orthogonal meshes and as the results are in good agreement with the literature, it may be concluded that the code is working well and able to calculate a wide variety of flows.

Even though the code was found to be working fine, one major problem connected to the accuracy of the code has to be addressed. This problem is connected to the use of the UDS scheme when the cell Peclet number exceeds 2, giving rise to large numerical diffusion when the flow is inclined with the mesh lines. As a consequence of this problem, the second order accurate SUDS scheme, and the third order accurate QUICK scheme was implemented as described in Chapter 4. As the main goal of the test cases was to prove that the code was working, and the accuracy was of minor interest, the test cases will not be recalculate in order to show the better accuracy of the higher order schemes.

13 Flow over a surface mounted cube

13.1 Introduction

The thick boundary-layer flow around a cube was selected as a three-dimensional test case. This flow allows the codes ability to compute three-dimensional massive separation to be tested in a situation that resembles atmospheric conditions. The flow is highly complex, and several recirculation regions exist as well as a horse-shoe vortex that wraps around the cube.

The fact that the separations in the present case are fixed by the sharp edges of the cube makes it simple compared to flow over streamlined bodies.

A large number of investigations of the flow around surface mounted objects have been performed in the past, ranging from purely theoretical considerations, see Ferziger [10], Hunt et al. [16], Hunt et al. [15], Hosker [14], over numerical calculations, see Murakami et. al. [37], Murakami and Mochida [38], Murakami et. al. [39], Baetke et. al. [2], Paterson and Apelt [44], Stathopoulos and Baskaran [52], Panneer Selvam [40], Zhang [66], Mikkelsen and Livesey [35], to wind channel and full scale atmospheric measurements, see Robins and Castro [49], Castro and Robins [6], Martinuzzi and Tropea [27], Schofield and Logan [50], Levitan and Mehta [24], Levitan and Mehta [25].

13.2 Problem description

The onset flow is perpendicular to the cube front face see Fig. 43, making the problem symmetric around the xz-plane going through the centre of the cube.

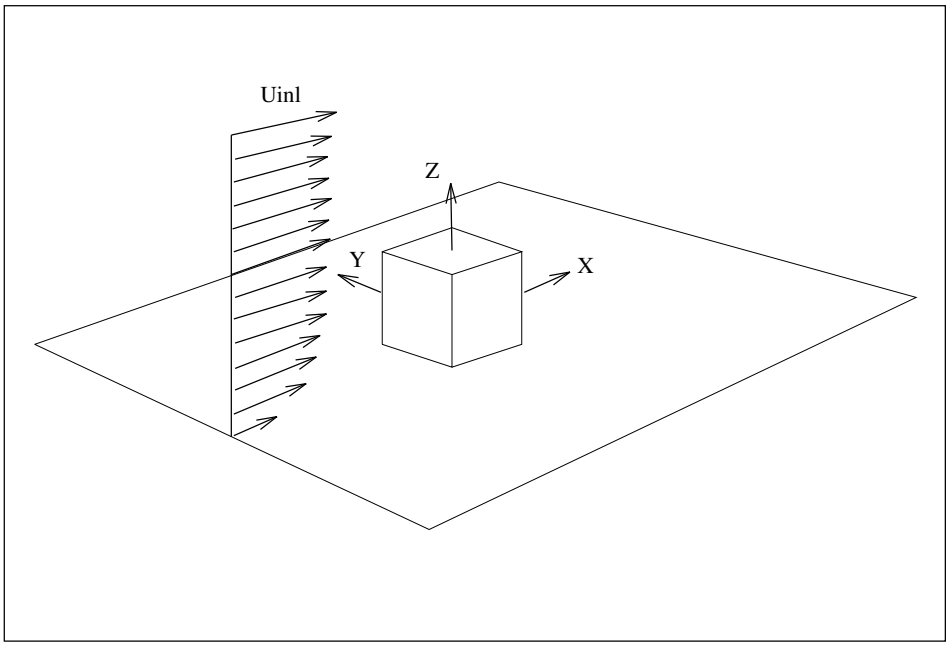

Figure 43. Problem setup showing the position and orientation of the coordinate system and the onset flow.

In the present computation the symmetry of the problem will be used, considering the flow around the half of cube only, using a symmetry condition at the

centre plan. For inclined flows this would have been impossible and a complete cube must have been used.

We will specify the variables according to an equilibrium profile at all free surfaces of the computational domain except for the boundary downstream of the cube.

The practice of using equilibrium profiles at all free surfaces is not correct as the presence of the cube will affect the surrounding flow. However, if the boundaries are moved far away from the cube, the approximation will become better and the introduced error will be small.

In the present calculation, the effect of different locations of the outer boundary was briefly investigated, using two computational domains of different sizes. Baetke et al. [2] also investigated this aspect and found minimal difference between their largest and next largest domain. Both of the domains used in the present investigation are greater than the largest domain used by [2]. The smallest of the present domains, mesh A covers $-9.5 < x < 9.5$, $-8.5 < y < 0$, and $0 < z < 9$. The slightly larger domain in mesh B covers $-12.5 < x < 12.5$, $-12.5 < y < 0.0$, and $0 < z < 12.0$. In mesh A, as well as in mesh B, the cube centre is placed in (x, y, z)=(0,0,0).

At the downstream boundary the flow is assumed to be fully developed, and zero normal-gradient is specified. This may not be physically true, but for large Reynolds numbers the influence from the downstream boundary on the upstream flow will be small.

The Reynolds number was based on the onset velocity at cube height, the cube height, and the molecular viscosity

$$Re = \frac{\rho U_H H}{\mu} \ .$$

At all solid walls, the bottom surface and the cube walls, the rough-wall version of the logarithmic law of the wall is used.

For the velocities the use of equilibrium boundary layer profile at the air/air interfaces implies use of the logarithmic profile

$$U(z) = \frac{U_*}{\kappa} \ln\left(\frac{z}{z_0}\right) \ . \tag{131}$$

Combining the equation for the Reynolds number and the equation for the velocity profile gives the following equation for the friction velocity

$$U_* = \frac{Re\mu\kappa}{\rho H \ln\left(\frac{H}{z_0}\right)} \ . \tag{132}$$

The equilibrium profiles of turbulent kinetic energy and dissipation of turbulent kinetic energy are given by

$$k = \frac{U_*^2}{\sqrt{C_\mu}} \ , \tag{133}$$

and

$$\epsilon = \frac{{C_\mu}^{3/4} k^{3/2}}{\kappa z} \ .$$

13.3 Inlet profiles

After the general discussion of the appropriated inlet profiles, the values used in the actual computations will be addressed.

Unfortunately, the agreement between the theoretical profile and the measured profile is not very good in the lower part of the boundary layer, see Fig. 44. In

order to eliminate the differences between the measured and calculated velocity fields, the inlet velocity will be specified instead according to

$$U(z) = \alpha(z/\delta_0)^{0.28}, \tag{134}$$

where α is a proportionality constant and δ_0 is the boundary-layer height. As seen from Fig. 44 this profile is in good agreement with the actual measurements of Castro and Robins [6].

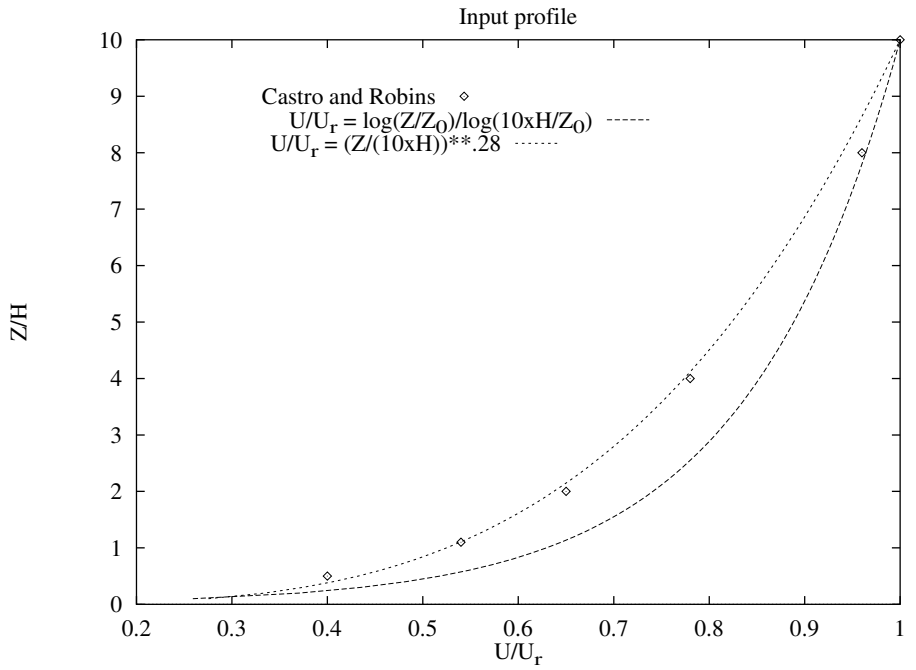

Figure 44. Inlet profiles, the discrepancy between the logarithmic profile and the measured profile is obvious. The profile used in the simulation is shown for comparison.

As the derivation of the profiles for the turbulent quantities (k, ϵ, ν_t) is based on equilibrium assumptions, the logarithmic velocity profile will be used in the derivation of these, even though this is not consistent with the velocity profile used.

In the experimental work [6] all of the pressure measurements were performed for $Re > 10^5$, and it was stated that above this value no Reynolds number dependency was experienced. For the measurements in the wake, the flow was found to be independent of the Reynolds number for $Re > 3.10^4$. In the present simulation a Reynolds number of $Re = 10^5$ was used in order to allow use of both the pressure and wake measurements.

The roughness length of the floor in [6] was equal to $z_0 = 0.02 \times H$, but no information on the cube-face roughness was given. In the computation, the roughness of the cube will be set equal to $z_0 = 0.0002 \times H$.

Using (132) the Reynolds number and the roughness length we get a friction velocity equal to $U_* = 0.16$.

The turbulence intensity in the experiment was $\sqrt{k}/U_H = 0.27$. Using this value and (133), we get the following value for C_μ

$$C_\mu = \left(\frac{U_\tau^2}{k}\right)^2 = \left(\frac{U_*^2}{(0.27U_H)^2}\right)^2 = 0.03.$$

When C_μ is changed into this value, the $C_{\epsilon 1}$ constant of the $k-\epsilon$ model must also be changed, see Chapter 2. The dependency of $C_{\epsilon 1}$ on changes of C_μ can be

found from a simple assumption concerning boundary layer flow, see Arpaci and Larsen [1], giving the following expression

$$C_{\epsilon 1} = C_{\epsilon 2} - \frac{\kappa^2}{C_\mu^{1/2}\sigma_\epsilon} ,$$

using standard values for $C_{\epsilon 2}$ and σ_ϵ, a value of $C_{\epsilon 1} = 1.19$ is found. The resulting constants of the $k-\epsilon$ model are listed in table 9.

Table 9. Model constants in the $k-\epsilon$ model

κ	C_μ	$C_{\epsilon 1}$	$C_{\epsilon 2}$	σ_k	σ_ϵ
0.4	0.03	1.19	1.90	1.00	1.30

Table 10. Inlet profiles for the different variables for the model run.

Variable	Inlet profile	Constants
U	$U(z) = \alpha(z/\delta_0)^{0.28}$	$\alpha = 2.99$, $\delta_0 = 10 \times H$
V	0.0	
W	0.0	
k	$k(z) = U_*^2/\sqrt{C_\mu}$	$U_* = 0.16$
ϵ	$\epsilon(z) = C_\mu^{3/4} k^{3/2}/(\kappa z)$	
ν_t	$\nu_t(z) = \kappa U_* z$	

13.4 Computational mesh

The simple geometry of the problem is well suited for a rectangular grid, stretched from the solid surfaces.

Using the multiblock facility of the code the domain is divided into 11 blocks of $16 \times 16 \times 16$ cells. This configuration has a length of three blocks in the x-direction, a width of two blocks in the y-direction, and a height of two blocks in the z-direction. This would add up to twelve blocks, but as no mesh block is necessary where the cube is located, the number is reduced to eleven blocks.

A perspective view of the mesh near the cube is shown in the top left part of Fig. 45, and different slices of the mesh can be seen in the other parts of Fig. 45. From this figure the stretching of the mesh towards the solid surfaces and the size of the total domain can also be seen.

The physical dimensions of the domain have already been discussed in connection with the location of the outer boundary, and only the blockage ratio will be discussed. The blockage ratio is defined as the ratio between the frontal area of the cube and the cross-sectional area of the flow domain or wind tunnel. In the experiment of [6] the ratio was 0.7%. In the present computations the ratio was 0.65% for mesh A, and 0.36% for mesh B, indicating that the blockage ratio should have less influence in this computation than in the measurements.

13.5 Results

In order to evaluate the quality of the simulation, both qualitative and quantitative results will be addressed. To give an idea of the complexity of the flow, particle traces around the cube is shown in Fig. 46.

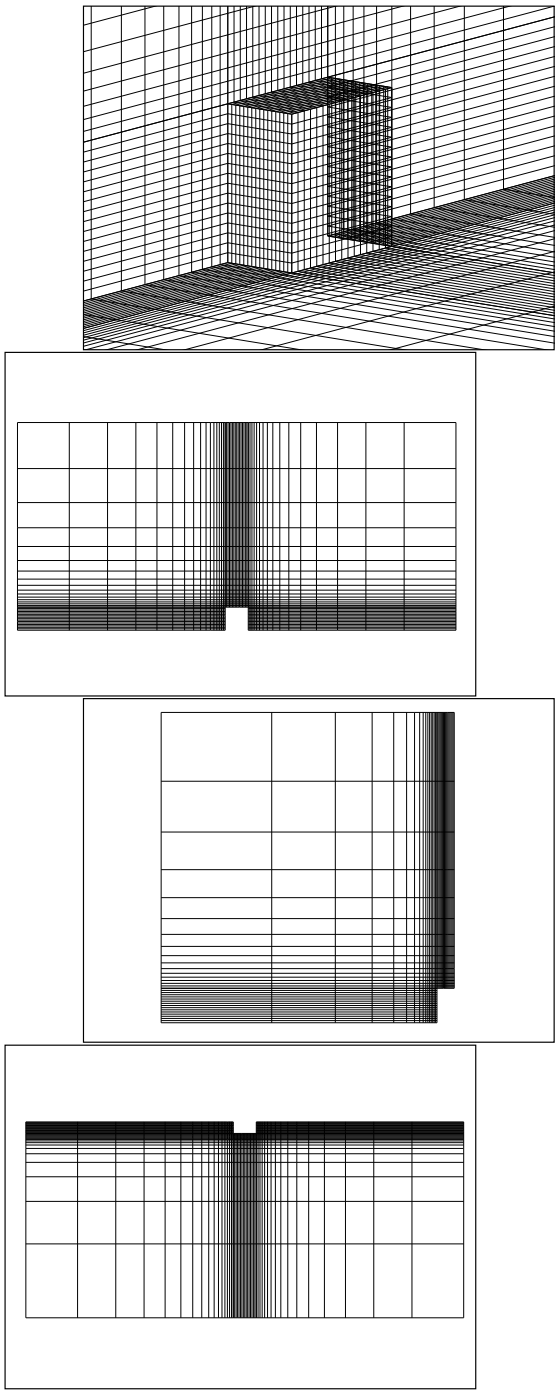

Figure 45. Computational mesh, top left showing a close view of the mesh near the cube, top right showing a xz-plan through the centre of the cube, bottom left showing a yz-plan through the centre of the cube, and bottom right showing a xy-plan through the centre of the cube.

In the following discussion the pressure coefficient C_p will be defined by

$$C_p = \frac{P - P_\infty(H)}{\frac{1}{2}\rho U(H)} \ .$$

The reference velocity used to normalized the velocity and the turbulent kinetic

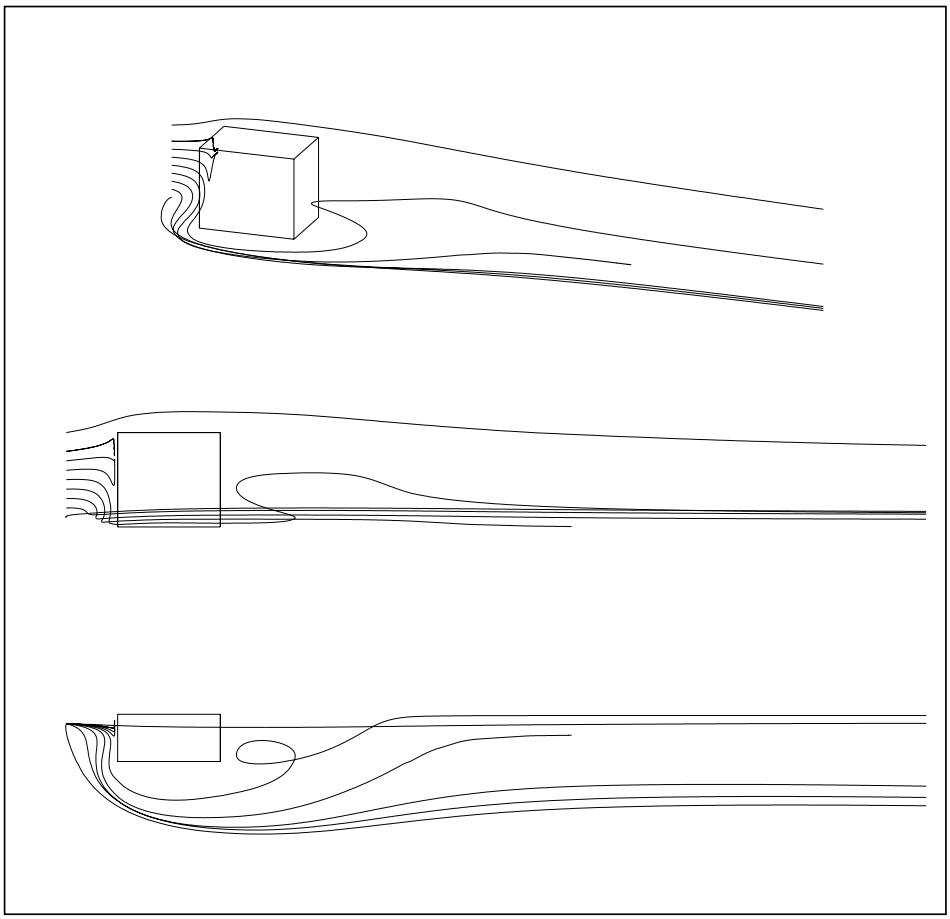

Figure 46. Particle traces around the cube. The particles are released at (x,y)= (-1,0.1) for z in the range [0.1,1.0] with increments of 0.1. At the top of the figure the traces are shown in a perspective view. In the middle the traces are seen in sideview along the y-axis. The traces are shown viewed from above the cube in the bottom of the figure.

energy, is the upstream velocity at height $10 \times H$

$$U_{ref} = U_{\infty}(10 \times H) \ .$$

From the plots of the pressure coefficients on the cube surface Figs. 47 and 48, only a minimal difference is found between the simulation on mesh A and mesh B. As the difference between the results from mesh A and B is small compared to the differences between the simulated and the measured values, only the results from mesh A will be shown in the following.

Surface pressure

The overall form of the pressure distribution resembles the measured distribution quite well, but some differences exist. From Fig. 47 it is seen that on the top of the cube along line B the calculated pressure restitutes much faster than the measured pressure, indicating that in the simulation no recirculation is found on the top of the cube.

Another aspect is the distinct underprediction of the pressure coefficient on the cube beyond $x = 0.0$. One explanation for this could be the assumption behind the logarithmic wall law being in error, because no favourable pressure gradient exists in this region.

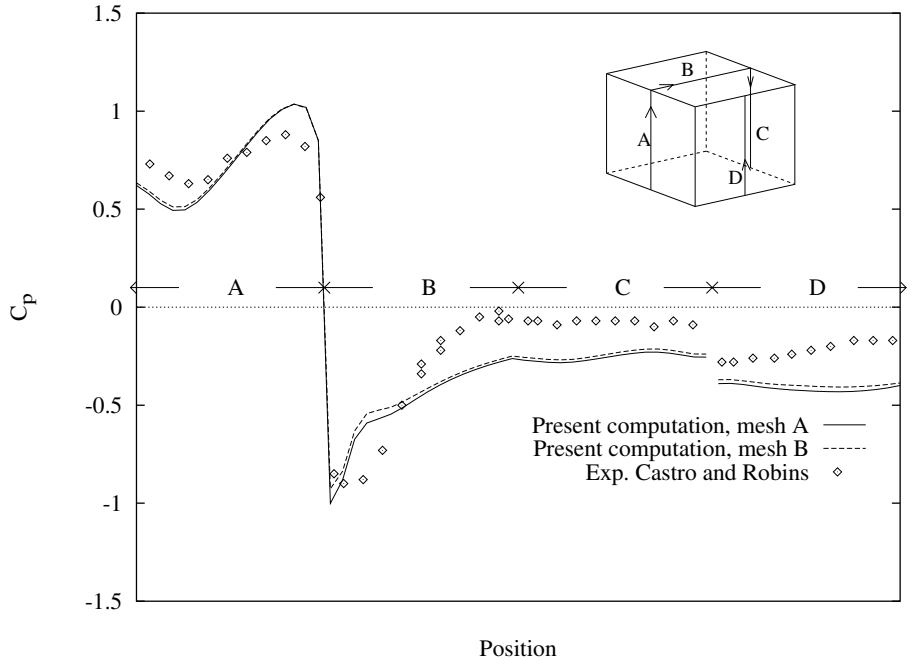

Figure 47. Pressure distribution. The positions along the four lines A, B, C, and D are indicated on the cube in the top right corner. Where $C_p = (P - P_\infty(H))/(1/2\rho U^2(H))$.

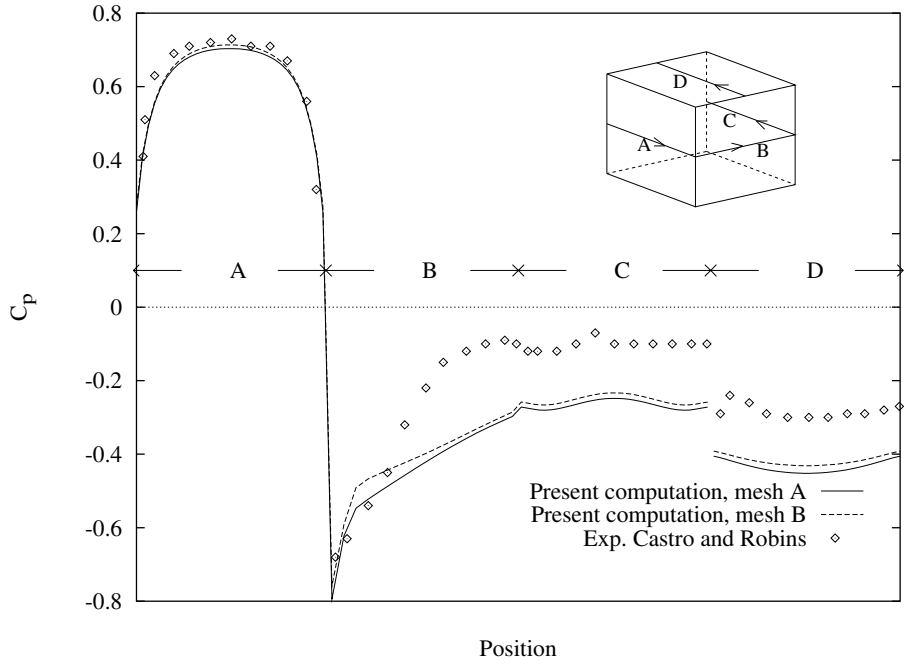

Figure 48. Pressure distribution. The positions along the four lines A, B, C, and D are indicated on the cube in the top right corner. Where $C_p = (P - P_\infty(H))/(1/2\rho U^2(H))$.

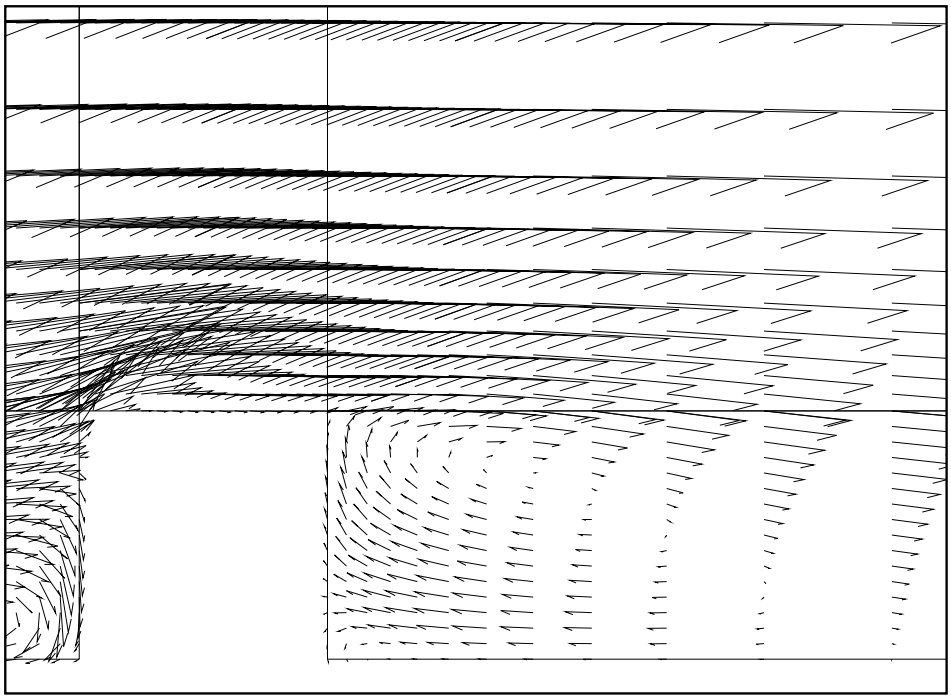

Figure 49. Slice in the xz-plane for $y/H = 0.0$, showing the large separated region behind the cube. Also the lack of separation on top of the cube can be seen.

Recirculating regions

As indicated by the pressure distribution along line B at Fig. 47, the small recirculation bubble separating from the front of the roof is not found in the present computation, see also Fig. 49. The main reason for the lack of separation on the roof is believed to be the poor resolution of the computation grid near the walls. The measured recirculation bubble is about $0.3 \times H$ long and the height of the reversed region is $0.05 \times H$ only. With the present mesh, only one cell will be located within the reversed region.

The simulation, lacking the separation on top of the cube, correctly predicts the boundary layer to separate from the back edge of the roof. However, the length of the recirculating region is too long compared with measurements. As the wake development is affected by the upstream turbulence, the specification of the inlet turbulence become rather important. The assumption that the inlet turbulence is an equilibrium profile, may have been too rude an approximation. Experiments with the level of kinetic energy in the inlet were performed, and they showed that it actually was possible to influence the wake development.

Another aspect influencing the wake development is the blockage ratio, where an increased blockage ratio results in shorter wakes. A direct comparison between the blockage ratios in the experiment and in the calculation is not possible, because in contrast to the experiment no boundary layer is formed at the outer boundaries in the simulation.

As the blockage ratio for both mesh A and B is less than that of the measurements, and as no boundary layer is formed at the outer boundaries, the separation bubble will be expected to be longer than what is found in the experiment.

At the sides of the cube the simulation predicts recirculation bubbles, separating from the front of the cube, see Fig. 52. Near the bottom of the cube the bubbles do not reattach to the side-wall, but continue into the separated region behind the cube. At a larger distance from the bottom surface the bubbles start reattaching to the side-walls, and for heights larger than about $z/H = 0.75$ the separation

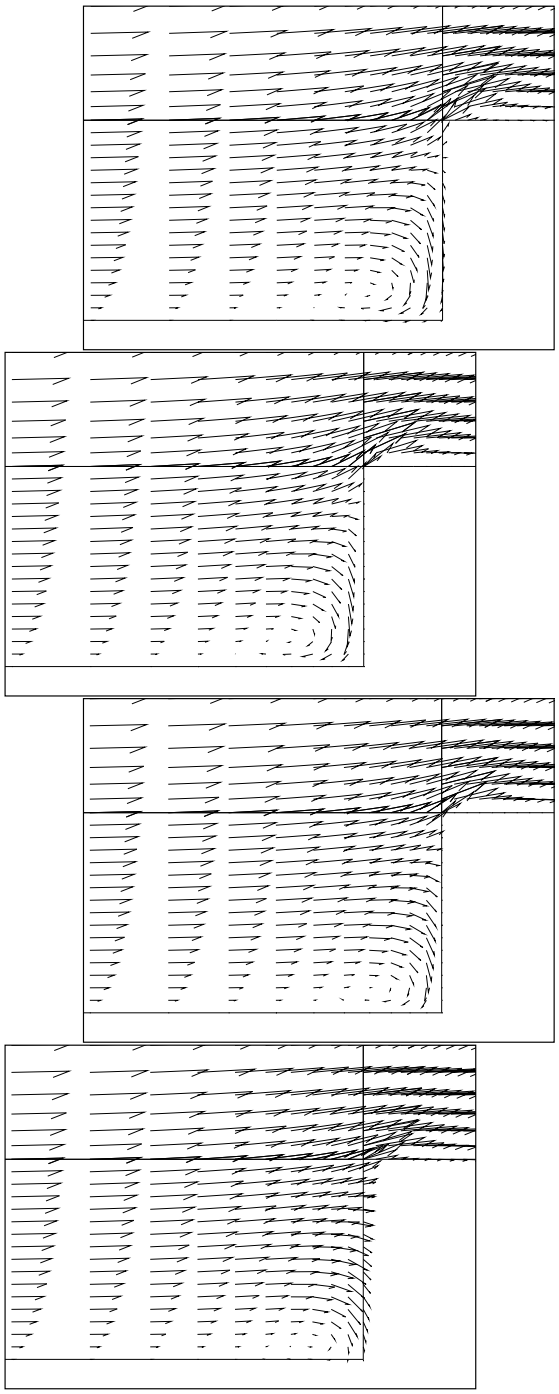

Figure 50. Slices in the xz-plane. The position of the slices are from top left to bottom right picture, $y/H = 0.00$, $y/H = -0.13$, $y/H = -0.38$, $y/H = -0.50$. It is seen that the horse-shoe vortex in front of the cube is losing its strength as the side of the cube is approached.

bubbles disappear. No measurements are available for this flow region, and in [6] it is stated only that the region is similar to the separated region on top of the cube.

In front of the cube a recirculating region or a horse-shoe vortex is generated. The strength of the vortex is largest at the centre of the cube and reduces as

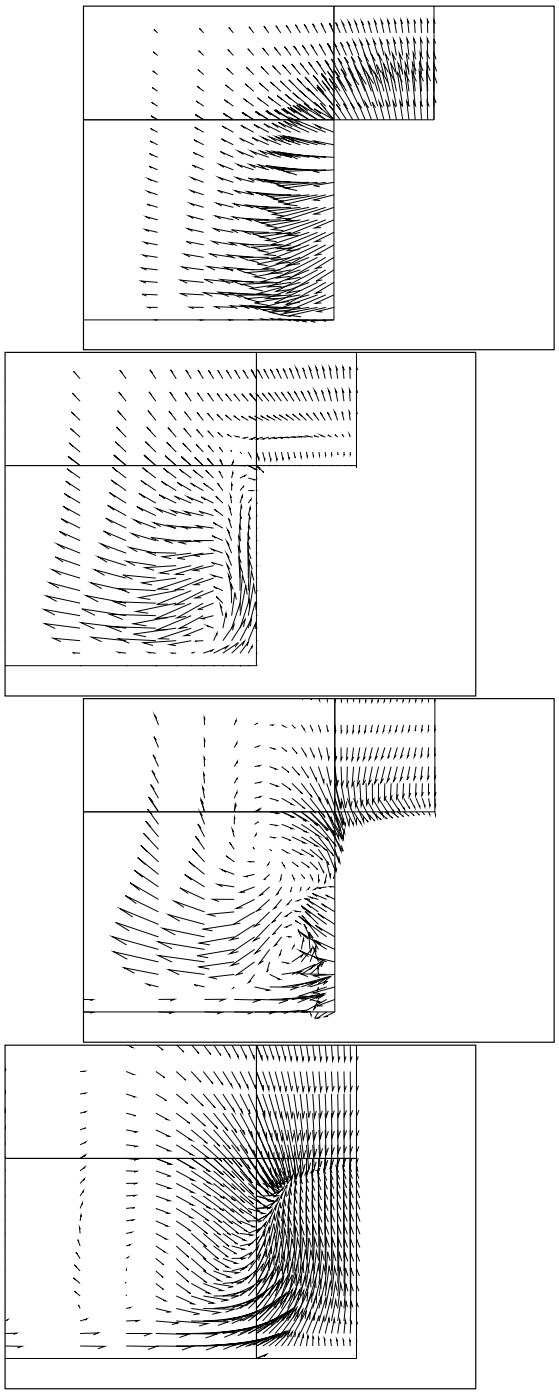

Figure 51. Slices in the yz-plane, the position of the slices are from top left to the bottom right picture, $x/H = -0.50$, $x/H = 0.00$, $x/H = 0.50$, $x/H = 0.99$. The pictures show the secondary motion within the recirculation bubbles on the side and behind the cube.

the sidewalls of the cube are approached, see Fig. 50. In Fig. 51 the cube is seen from behind along the x-axis, showing the secondary motion. In this figure the horse-shoe vortex is not visible, but a vortex rotating in the opposite direction is seen to be generated near the bottom corner of the cube.

As no measurements are reported for the flow development at the sidewalls of the

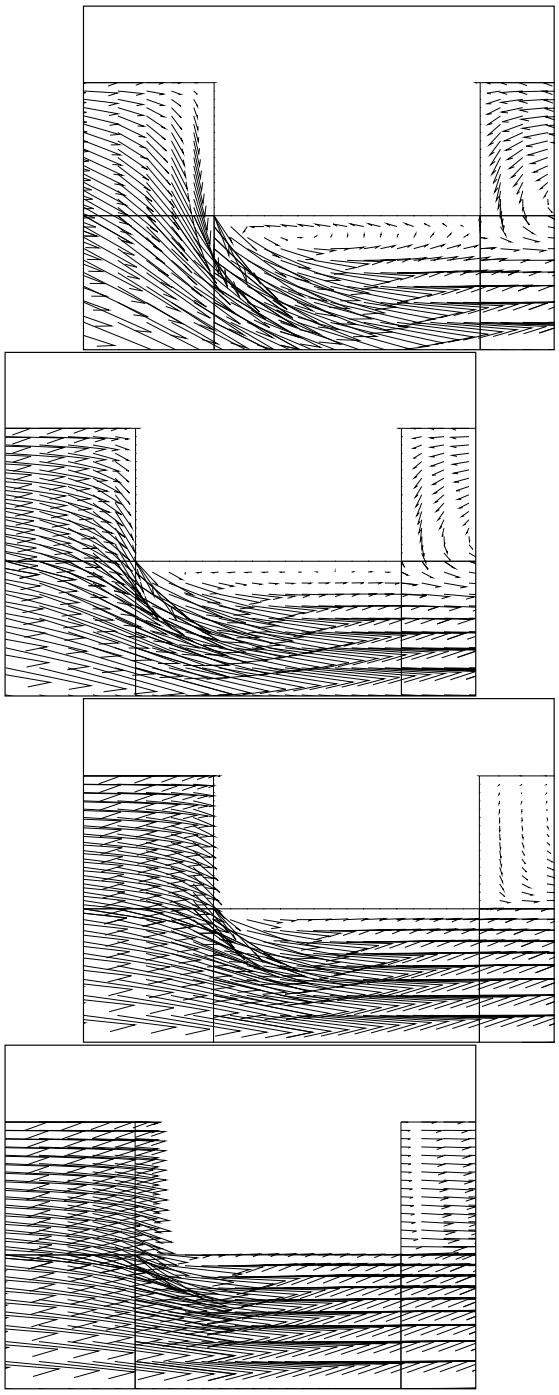

Figure 52. A slice in the xy-plane. The position of the slices is from top left to the bottom right picture, $z/H = 0.25$, $z/H = 0.50$, $z/H = 0.75$, $z/H = 1.00$. The pictures show the recirculation bubble at the side of the cube.

cube, the correctness of the secondary motion cannot be verified. The secondary motion is very weak, the vertical component is less than 5% of the onset flow in the same height, which could make it difficult to measure or visualize.

Looking at Fig. 53, a very good agreement is found between the computed results and the measurements over the centre of the cube. The agreement is still reasonable at $x/H = 1.0$ in the recirculation region near the cube, further downstream in the

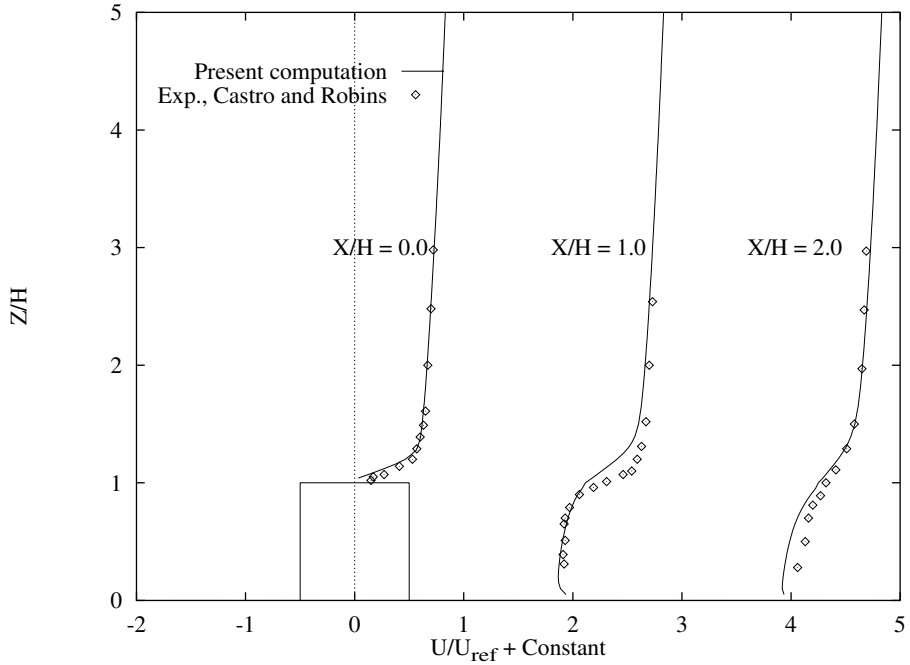

Figure 53. A comparison of the computed velocity profile at different values of x/H at the slice $y/H = 0.0$ with measurements from [6]. The constants added is 0.0 for $x/H = 0.0$, 2.0 for $x/H = 1.0$, and 4.0 for $x/H = 2.0$. Where $U_{ref} = U_{\infty}(10 \times H)$ has been used for normalization.

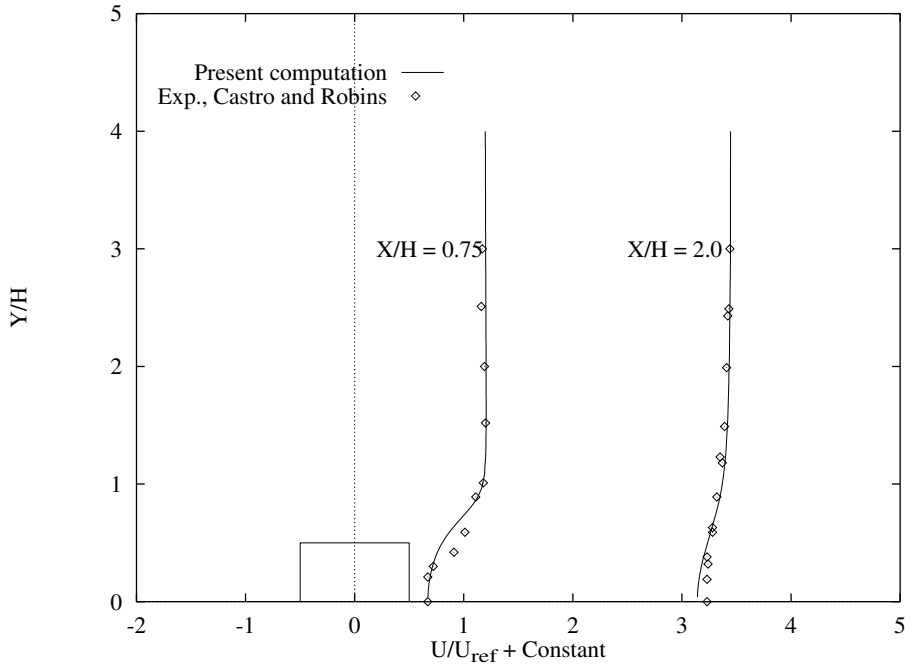

Figure 54. A comparison of the computed velocity profiles at different x/H at the slice $z/H = 0.5$ with measurements from [6]. The constants added is .75 for $x/H = .75$, and 3.0 for $x/H = 3.0$. Where $U_{ref} = U_{\infty}(10 \times H)$ has been used for normalization.

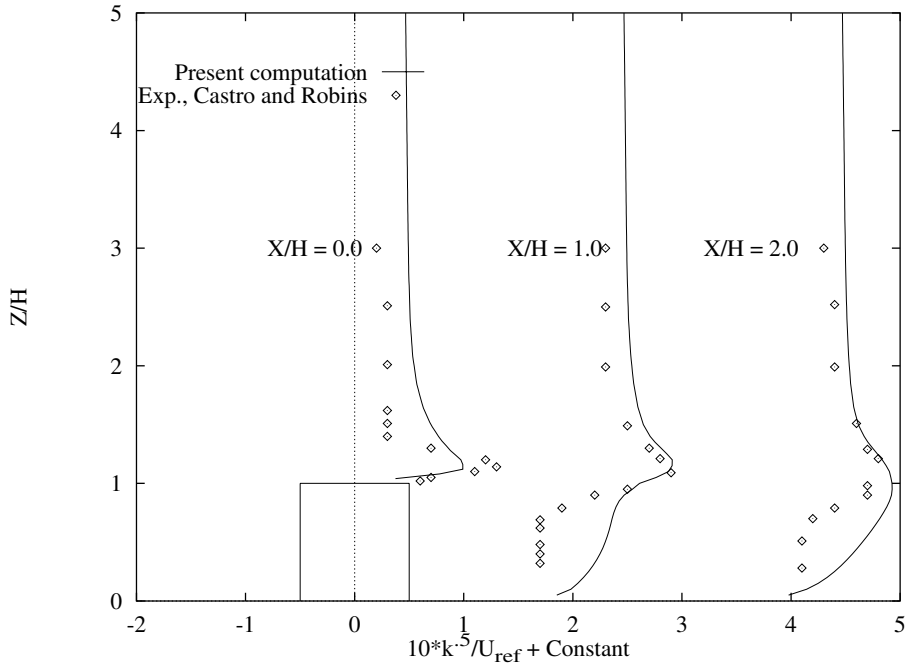

Figure 55. A comparison of the computed turbulence intensity profiles at different x/H at the slice $y/H = 0.0$ with measurements from [6]. The constants added is -1.0 for $x/H = 0.0$, 1.0 for $x/H = 1.0$, and 3.0 for $x/H = 2.0$. Where $U_{ref} = U_\infty(10 \times H)$ has been used for normalization.

wake the agreement is less satisfactory.

The reason for the too slow wake development has been discussed at an earlier stage and is assumed to be related to the level of turbulent kinetic energy in the inlet. The behaviour found in the present case is opposite to the behaviour normally found using the $k - \epsilon$ turbulence model, where it is known to give too short recirculation regions.

The coarse meshes, used in the present computation, gives rise to numerical diffusion to the solution, and thereby lowering the effective Reynolds number, would also result in a too short recirculation bubble in contrast to the actual findings.

Turbulent kinetic energy

As the individual Reynolds stresses do not appear in the $k-\epsilon$ model, the turbulence intensity will be approximated by $\sqrt{k}/U_{ref}$ using an assumption of isotropy ($\overline{u} = \overline{v} = \overline{w}$). According to the measurements of [6], this approximation is rather poor, but as no additional information is available it will be used anyhow.

Comparing the overall slope of the computed turbulence intensity profiles Fig. 55 with the measurements a reasonably good agreement is found. The main difference is the high turbulence intensity computed in the recirculation region behind the cube. This phenomenon is also seen in the two-dimensional computations of backward facing steps and is believed to be a consequence of the $k - \epsilon$ model.

The location of the peak value of turbulent kinetic energy is well predicted by the model, even though the peaks are not quite as narrow as shown by the measurements. This smearing out of the peaks may be caused by a too coarse mesh which is unable to resolve the strong gradients near the peak.

13.6 Closure

From the results of the simulations it can be concluded that computation of the flow around a surface mounted obstacle in a boundary layer similar to that in the atmosphere is possible.

This statement is based on the fact that even though the simulation fails to produce details, such as the separation on top of the cube, it captures all the major structures of the flow. Also the quantitative agreement is reasonably good, at least in the proximity of the cube.

To get a better result, it is believed that a finer mesh is necessary. Thus, use of 24^3 or 32^3 blocks would allow a better resolution of the region near the cube surfaces. Unfortunately, the computer resources at hand do not allow larger mesh blocks than 16^3.

The computer time to obtain a convergent solution is another aspect. For the present problem, the CPU time was about twelve hours on a IBM RS6000 220h workstation. In case of a finer mesh, the execution time would increase making the simulation rather time consuming.

14 A two-dimensional Hill

14.1 Introduction

The estimation of wind resources at a given place is of great interest in connection with siting of wind turbines, airports, and many other applications strongly depend on the local wind climate.

The techniques available to obtain this information are full scale measurements, wind tunnel measurements, and numerical modelling.

The first two techniques are expensive and will not be used in general in the search for an optimal place for a wind turbine or an airport. Instead, these techniques can be used for a refined study when a potential site is already found. The final possibility, the numerical modelling in form of a linearized model is the standard approach dealing with the estimation of wind resources, often in the form of creating a wind atlas, see Troen and Petersen [60].

The use of linearized models is both computationally cheap (can run on a standard PC) and accurate. The main limitation is the lack of ability to compute separation caused by using the linearized equations and the restriction to gentle terrain.

As a consequence, the use of nonlinear models, like the one developed in the present study, may be of interest as a supplement to the linear models. The model can be used in the special cases where the linear models are known to give wrong results, but it can also be used when trying to improve linear models to incorporate nonlinear separation phenomena.

The flows over Blashval, and that of Askervein, have become standard test cases for linear flow models. The Blashval hill is a bell shaped 109 m high hill located at North Uist in Scotland, see the description by Mason and King [28]. Askervein hill is a 126 m high hill located at the Hebrides, the hill has a elliptical plane form with a 1 km minor axis and a 2 km major axis, see Taylor and Teunissen [55]. The fact that Askervein hill has an elliptical plane form allows us to approximate the flow over the central part of the hill to be two-dimensional when the flow is directed along the minor axis. The Askervein hill will therefore be preferred in the present study, as it in contrast to the Blashval hill allows us to use the assumption of two-dimensional flow.

Askervein hill has been studied both by full-scale measurements, see Taylor and Teunissen [55] and [56]; by wind tunnel measurements, see Teunissen and Shokr [58], and Bowen and Teunissen [5]; by linear models, see Walmsley and Salmon [62], Beljaars et al. [3] , Troen and Petersen [60], Zeman and Jensen [65], and by nonlinear models, Raithby et al. [46].

14.2 Problem description

The topography of Askervein hill is described in [55], the contour lines of the hill has been digitized from the map of [55], and the hill profile along line A was taken out. A contour plot of the Askervein hill is shown in Fig. 56, and the hill profile along line A is shown in Fig. 57.

Inlet profiles

In accordance with the model calculations of [46], a flow case corresponding to the measurements named TU03-B of [55] was chosen. This run is a case with the wind direction equal to 210 degrees or approximately along line A going from southwest to northeast. The velocity profile at the reference site (RS) upstream of the hill

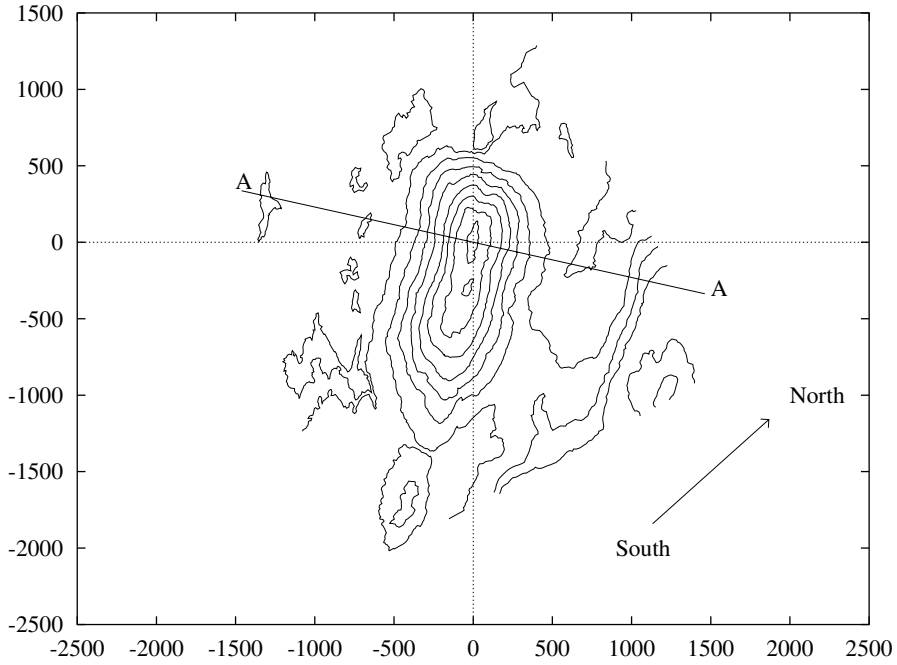

Figure 56. Contour plot of Askervein hill having the x-axis pointing in the flow direction (210 degrees) with the hill top at origo. The location of line A is indicated.

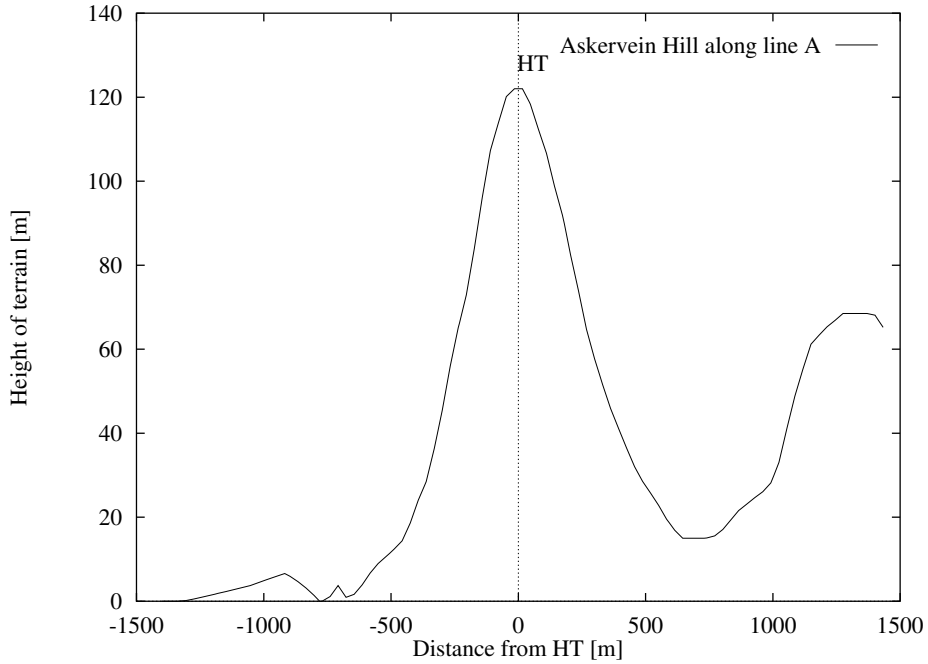

Figure 57. Profile of Askervein hill along line A, the hill top is placed at x = 0.0.

was a logarithmic profile with a velocity of 8.9 m/s at a height of 10 m, and the estimated roughness length (z_0) was equal to 0.03 m. The velocity profile is shown in Fig. 58.

$$U(z) = \frac{U_*}{\kappa} \log(\frac{z}{z_0}) \ .$$

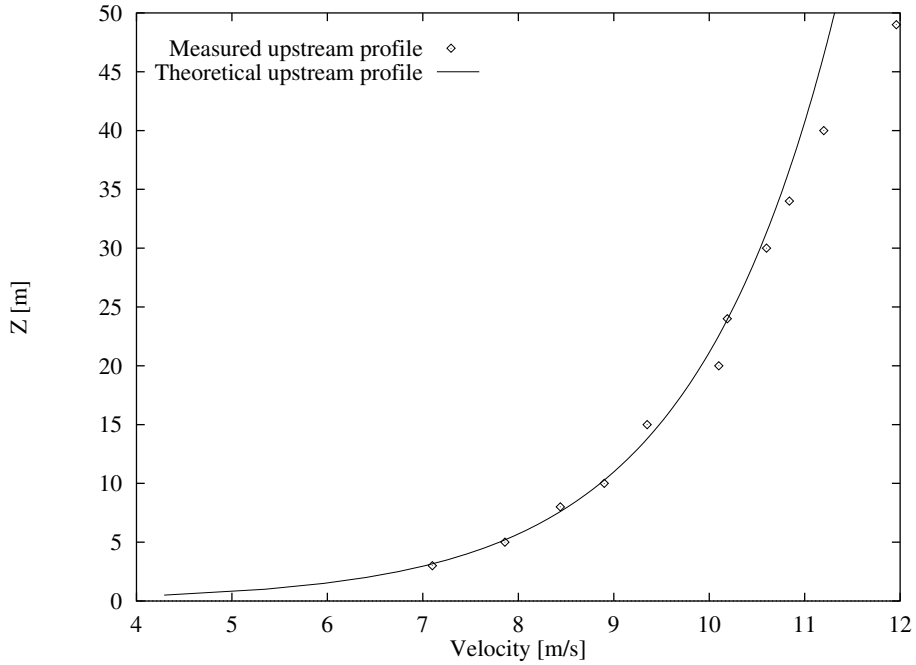

Figure 58. Comparison of measured upstream profile at RS and the theoretical logarithmic profile with $U_* = 0.61$, and $z_0 = 0.03$.

The turbulence intensity ($\sqrt{k}/U_{10\text{ m}}$) at RS was equal to 0.12. The turbulent kinetic energy was specified according to a equilibrium profile, namely

$$k(z) = \frac{U_*^2}{\sqrt{C_\mu}} ,$$

and the constants C_μ and $C_{\epsilon 1}$ were adjusted to obtain the correct turbulence intensity, see Chapter 2, resulting in the values shown in Table 11 for the $k-\epsilon$ model.

Also, the dissipation of turbulent kinetic energy was specified according to a equilibrium profile resulting in the following profile

$$\epsilon(z) = \frac{C_\mu^{3/4} k^{3/2}}{\kappa z} .$$

Table 11. $k-\epsilon$ model constants for flow over Askervein hill.

κ	C_μ	σ_k	σ_ϵ	$C_{\epsilon 1}$	$C_{\epsilon 2}$
0.40	0.11	1.00	1.30	1.54	1.92

Boundary conditions

The inlet was specified according to the profiles just given and the outlet was assumed to be fully developed. The hill surface was modelled according to the rough wall version of the logarithmic wall law, see Chapter 7. At the far-field or top boundary a symmetry condition was used. At the top boundary a specification according to the free stream profiles was also tried resulting in minor differences for the case of the 1900 m heigh domain.

All computation were performed using the third-order accurate QUICK scheme by Leonard [23].

14.3 Domain and mesh investigations

In order to determine the necessary height of the computational domain and the number of mesh points needed to resolve the flow, a series of computations were performed. In order to compare the model runs against each other the calculated speed-up in 10 m height above the terrain and the speed-up along a vertical line located at the hill top were compared in the different cases. The speed-up is defined as the difference between the actual velocity and the undisturbed velocity normalized by the undisturbed velocity

$$\text{Speed-up} = \frac{U(z') - U(z')_{inlet}}{U(z')_{inlet}} \ ,$$

where z' is the local height over terrain.

Table 12. Naming convention for mesh independency test.

Name	Number of cells	Domain height [m]	iterations	CPU time [s]
case 1	32 x 32	1900	186	56
case 2	64 x 64	1900	333	310
case 3	128 x 128	2000	955	3610

Table 13. Naming convention for domain height independency test, case 2b is identical to case 2.

Name	Number of cells	Domain height [m]	iterations	CPU time [s]
case 2a	64 x 64	1200	374	353
case 2b	64 x 64	1900	333	310
case 2c	64 x 64	3000	426	386

First three computations were performed in a domain covering the region -1500 m $< x <$ 1500 m and 0 m $< y <$ 1900 m with the hill top located at $x = 0.0$. In order to estimate the necessary number of computational cells, the number of cells was varied over the following range 32×32 cells, 64×64 cells, and 128×128 cells, see Table 12.

Looking at Figs. 60 and 61 the results are seen to be approximately mesh independent when the mesh has more than about 32×32 cells. In the lee of the hill the results are slightly better for the 128×128 cell mesh (case 3) than for cases 1 and 2. As a three-dimensional prediction is planned, we will focus on the speed-up near the hill top where a mesh of only 32×32 cells is sufficient to make the results mesh independent, because this will allow the planned three-dimensional computation to run on the computer available.

The influence of the height of the computational domain was investigated by varying the height of the domain keeping the number of cells constant. Using the results from the previous investigation, the number of cells was fixed to 64×64 cells in order to ensure that the results will be grid independent even for the highest domain. The following three domain heights were investigated, $h = 1200$ m, $h = 1900$ m, and $h = 3000$ m, see Table 13.

From Figs. 62 and 63 it is seen that the results are unaffected by the height of the domain when the domain is higher than about 1200 m. From these two tests we conclude that a 1900 m high domain with 32×32 cells will be adequate in order to make the solution independent of the domain height and number of cells.

The results computed from case 1 will thus be the ones compared to the measured values.

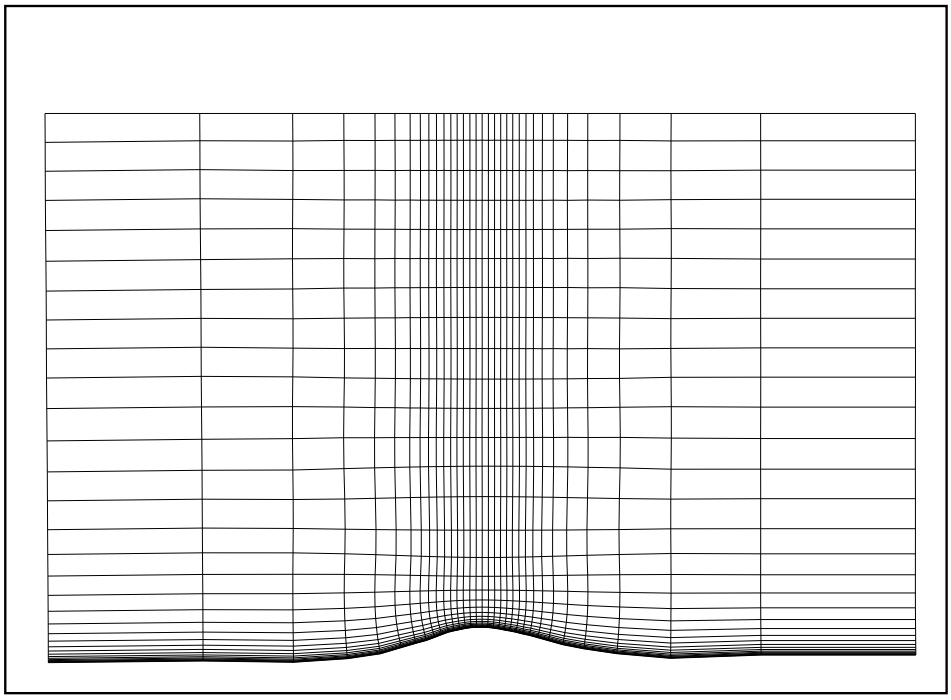

Figure 59. Mesh from case 1 with 32×32 *computational cells and a domain height of 1900 m.*

14.4 Comparison of measurements and computations

One important difference must be made clear before the actual comparison of the measured and computed results. The measurements are fully three-dimensional, allowing the flow both to move vertically and horizontally around the hill. In contrast the model runs are two-dimensional, allowing only the flow to move vertically over the hill, or in other words the two-dimensional hill is in fact a infinite fence perpendicular to the flow direction.

From this difference one may expect that the speed-up at the hill summit will be larger than that found for the three-dimensional measurements, and the tendency towards separation will be greater in the lee of the hill.

Speed-up

The computed value of the speed-up for case 1 is about 1% too low compared to the measured value, the computed value being equal to 0.87 in contrast to the measured value of 0.88. The speed-up of the two-dimensional calculation is thus equal to the measured value in contrast to expectations. The most obvious explanation –besides the model computing wrong results– is that the high speed-up at the hill top is caused by some local terrain effects (roughness change) not represented in the model.

The speed-up upwind of the summit is quite good, and even for the coarse mesh used in case 1 the agreement is good. In the lee of the hill the speed-up is predicted to be too high, see Fig. 60 and also the velocity vectors in Fig. 64.

Looking at the speed-up along a vertical line at the summit of the hill, the computed speed-up is predicted to be correct in the range 10 to 30 m over the terrain. The lower part being erroneous can again be explained by local terrain effects near the summit of the hill not resolved by the model, and the outer part may be explained by the two-dimensionality of the computed flow.

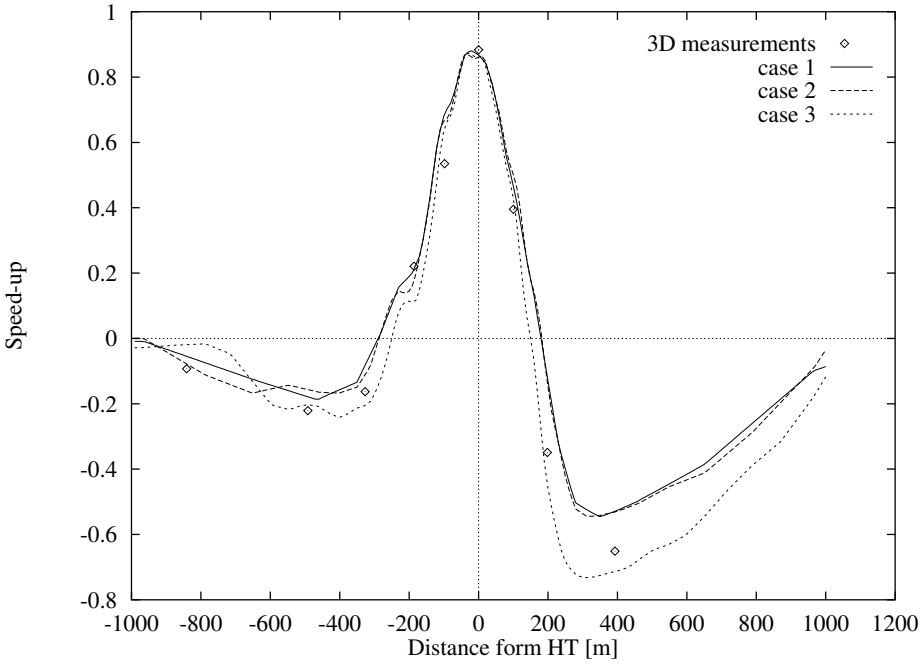

Figure 60. Mesh independency test showing the speed-up at 10 m height over terrain along line A for three different cases all with a constant domain height equal to 1900 m.

Turbulence intensity

As for the computed speed-up, the computed turbulence intensity shows a good agreement up to 100 m downstream of the hill top. After this point the turbulence intensity is strongly underpredicted, i.e. about 30% to low at $x = 400$ m.

The logarithmic law derived for a favourable pressure gradient may be one reason why the flow prediction is inaccurate in the lee of the hill where the pressure gradient is non favourable.

14.5 Closure

For the speed-up near the summit, we found that the solution was mesh independent already when the mesh had 32×32 cells. A domain height of 1900 m was sufficient to make the calculation nearly independent of the domain height.

The computed speed-up at the hill top in 10 m height was found to be within 1% of the measured value. As the computed case was two-dimensional in contrast to the three-dimensional measurements, we may anticipate that a three-dimensional computation will result in an underprediction of the speed-up, as the flow in this case also will be able to pass horizontally around the hill.

In other words, even though the results at the hill top show a good agreement with the measurements, the agreement is actually not very good as the speed-up should have been larger than that found for the full scale three-dimensional measurements.

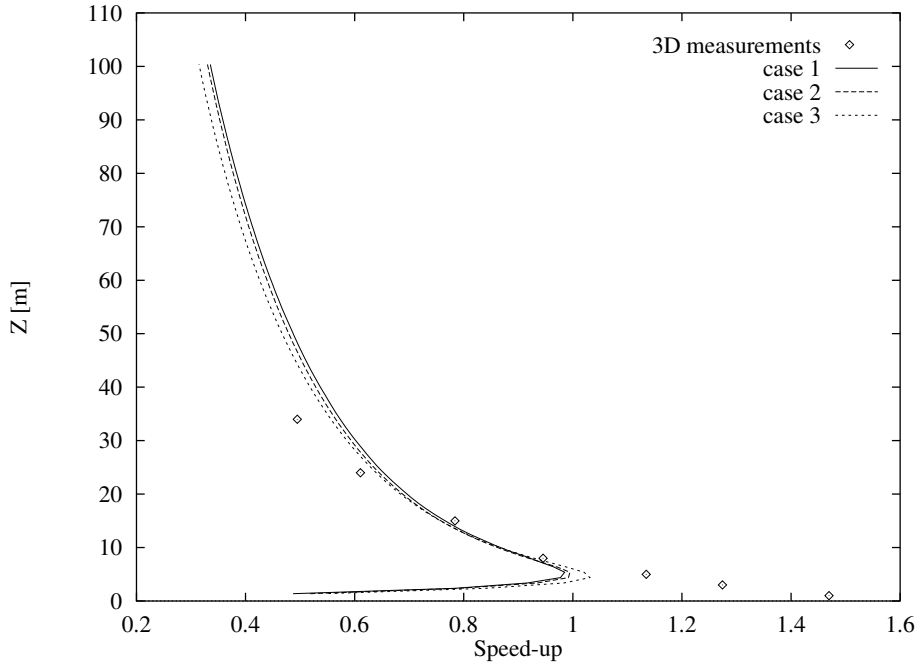

Figure 61. Mesh independency test showing the speed-up at along a vertical line at the hill top (HT) for three different cases all with a constant domain height equal to 1900 m.

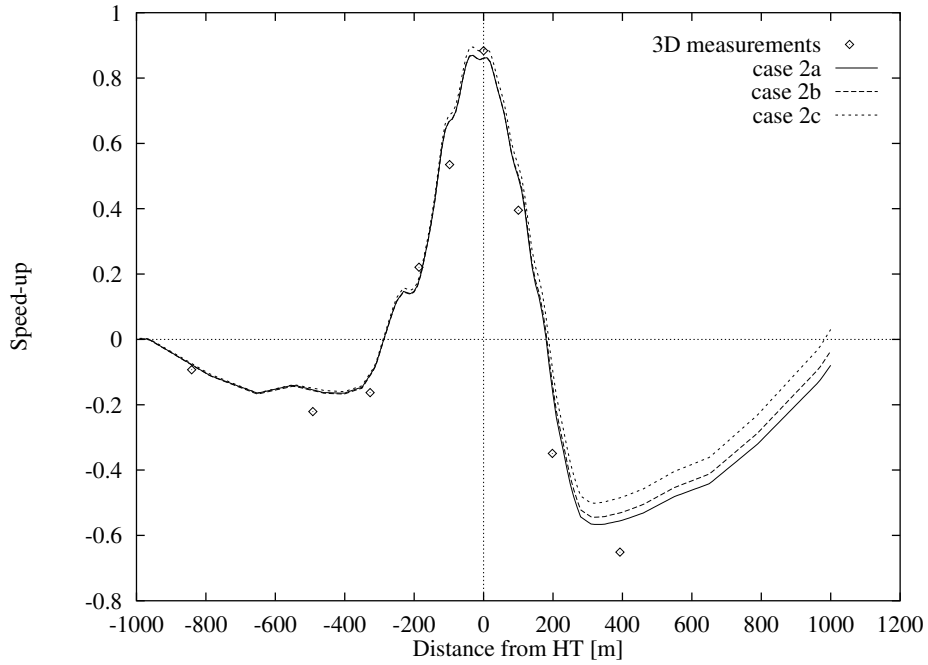

Figure 62. Domain height independency test showing the speed-up at 10 m height over terrain along line A for three different cases all with 32×32 *cells.*

In the next chapter it will be shown what happens when the flow over the hill is computed as fully three-dimensional.

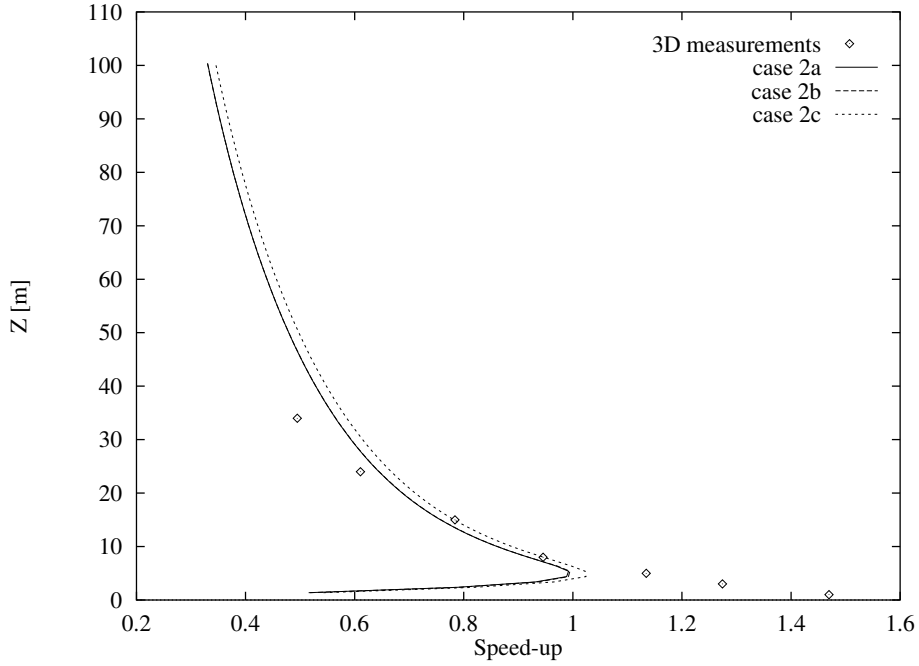

Figure 63. Domain height independency test showing the speed-up along a vertical line at the hill top (HT), for three different cases all with 32×32 *cells.*

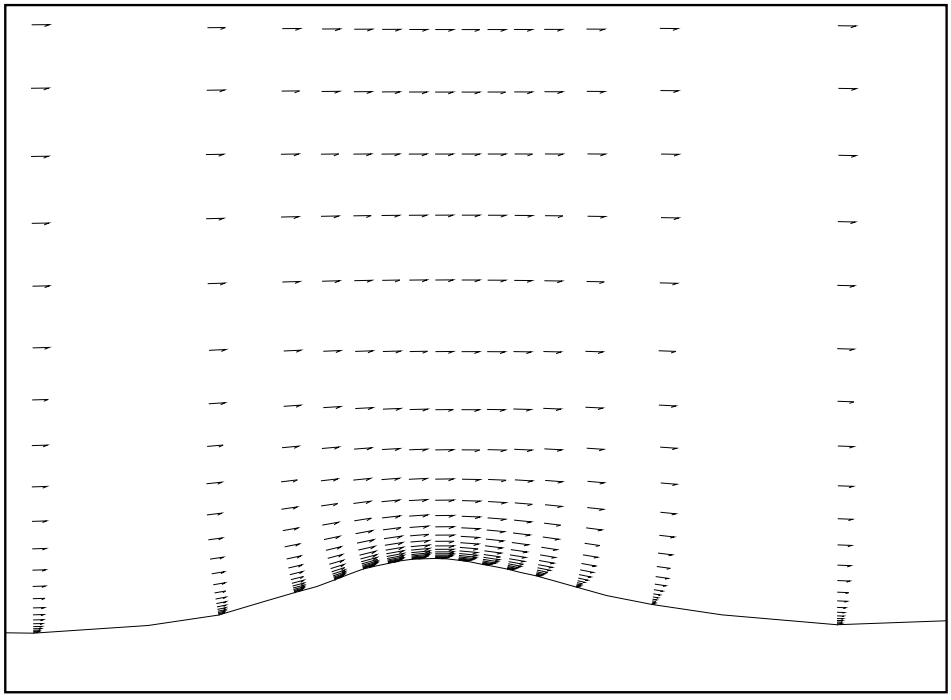

Figure 64. Details of the computed velocity field near the hill top, only every second vector is shown in the horizontal direction.

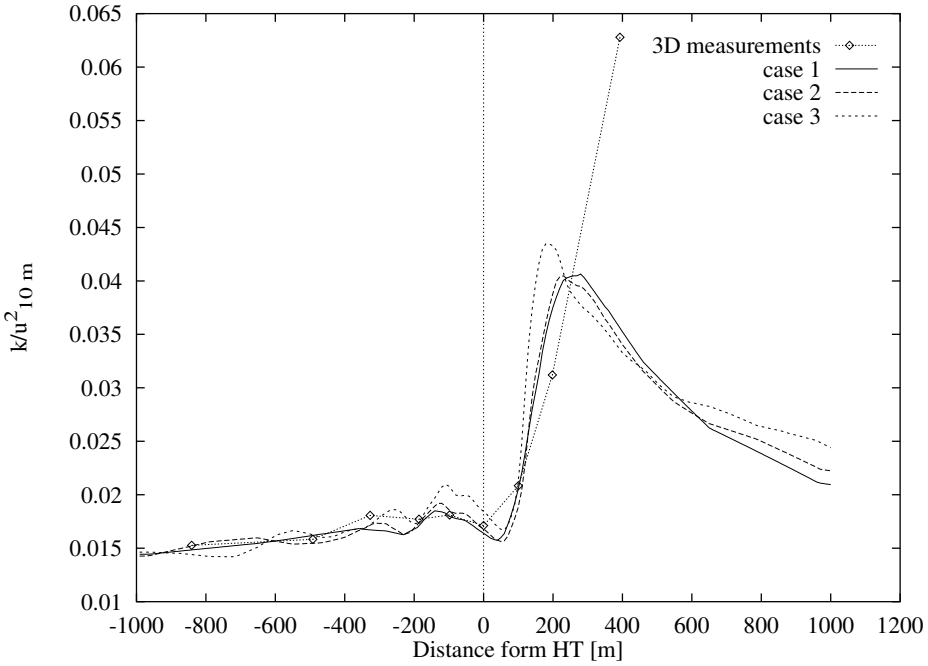

Figure 65. Comparison of the computed turbulence intensity for cases 1 to 3 and the full-scale three-dimensional measurements along line A at 10 m height above terrain.

15 A three-dimensional hill

15.1 Introduction

After having calculated Askervein hill using a two-dimensional approximation of the actual flow, the hill will now be calculated for fully three-dimensional flow.

In the fully three-dimensional simulation a horizontal passage of the flow around the hill is allowed, thereby removing the limitation inherent in the two-dimensional simulation.

In the three-dimensional simulation it is possible also to investigate the change of the wind direction when passing the hill, a possibility not present in the two-dimensional simulation.

For information on previous computations of Askervein hill, see Chapter 14 dealing with the two-dimensional hill calculation.

15.2 Problem description

The topography of Askervein hill is described in [55], the contour lines of the hill have been digitized from the map of [55], and thereafter grided using WA^{S}P, see Mortensen et al. [36].

The x-axis of the mesh was aligned with the mean flow direction (210 degrees) in order to minimize numerical diffusion, see Figs.66 and 56.

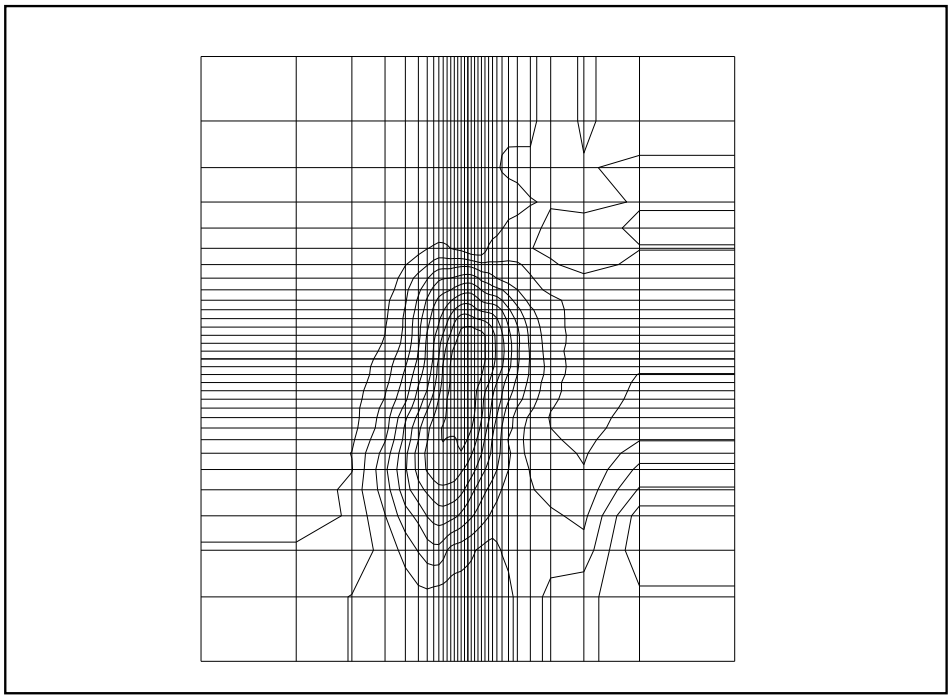

Figure 66. Contour plot of Askervein hill shown together with the mesh on the bottom surface. The x-axis of the mesh was aligned with the mean flow direction (210 degrees).

Inlet profiles

In accordance with the model calculations of [46], a flow case corresponding to the measurements named TU03-B of [55] was chosen. This run is a case with the

wind direction equal to 210 degrees, or approximately along line A going from southwest to northeast, see Fig. 56. The velocity profile at the reference site (RS) upstream of the hill was a logarithmic profile with a velocity of 8.9 m/s at a height of 10 m, and the estimated roughness length (z_0) was equal to 0.03 m.

$$U(z) = \frac{U_*}{\kappa} \log(\frac{z}{z_0}) \ .$$

The turbulence intensity ($\sqrt{k}/U_{10\ \mathrm{m}}$) at RS was equal to 0.12. The turbulent kinetic energy was specified according to a equilibrium profile, namely

$$k(z) = \frac{U_*^2}{\sqrt{C_\mu}} \ ,$$

and the constants C_μ and $C_{\epsilon 1}$ were adjusted to obtain the correct turbulence intensity, see Chapter 2, resulting in the values shown in Table 11 for the $k-\epsilon$ model.

Also the dissipation of turbulent kinetic energy was specified according to a equilibrium profile resulting in the following profile

$$\epsilon(z) = \frac{C_\mu^{3/4} k^{3/2}}{\kappa z} \ .$$

Boundary conditions

The inlet was specified according to the profiles just given, the outlet was assumed to be fully developed, and the far-field or top boundary condition was set according to the equilibrium profile. The hill surface was modelled according to the rough wall version of the logarithmic wall law, see Chapter 7, and symmetry conditions were used at the two vertical planes parallel to the flow direction.

The computation was performed using the third-order accurate QUICK scheme by Leonard [23].

Mesh and domain size

Having made a mesh and domain size independency test in the two-dimensional case, we will use these results directly for the three-dimensional case according to the following arguments. For the three-dimensional case, where both a horizontal and vertical passage of the flow is allowed around the hill, the gradients in the mean flow direction will be lower than for the two-dimensional case, thereby making the needed mesh resolution smaller in the three-dimensional case. As the hill is less steep in the cross-stream direction, we assume without any test that the resolution used in the mean flow direction is sufficient also in the cross stream direction.

The same argumentation holds for the domain-size considerations, and we will again use a domain with a height of 1900 m. The horizontal area covered is the region -1500 m $< x <$ 1500 m, -1700 m $< y <$ 1700 m, with the hill top placed at (x, y)=(0,0).

The mesh used in the present investigation was generated by the three dimensional hyperbolic mesh generator described in Chapter 11.

15.3 Results

The computation of Askervein hill was performed on the $32 \times 32 \times 32$ mesh, using the grid sequence technique. The solution was initiated on the $8 \times 8 \times 8$ mesh, where the residuals were reduced by a factor of 100. Then the solution was prolongated to the $16 \times 16 \times 16$ mesh and again the residuals were reduced by a factor of 100 with respect to the starting residual on the coarsest mesh. Finally, the solution

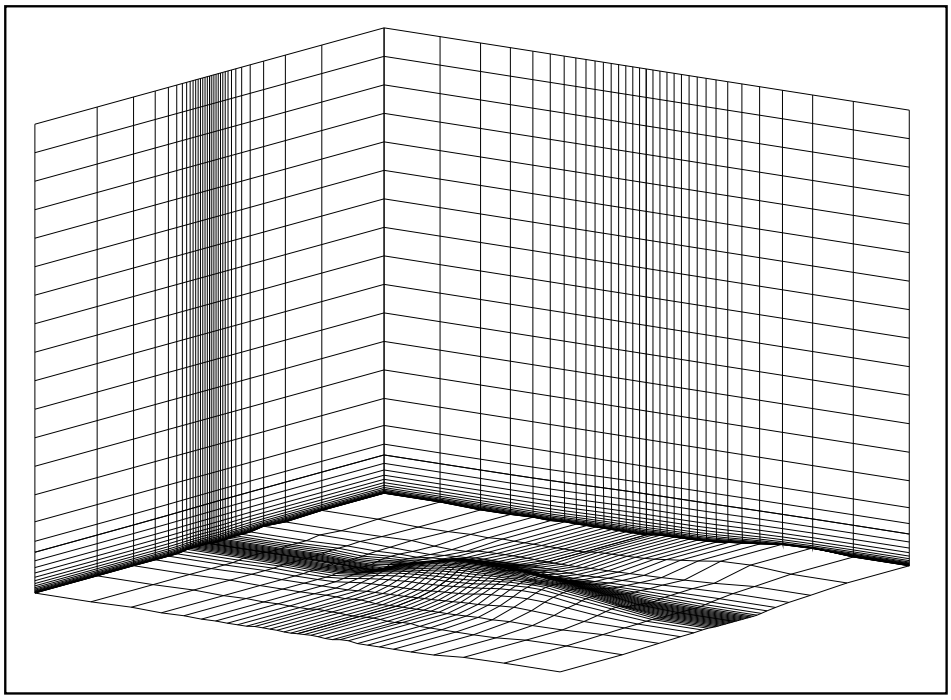

Figure 67. Perspective view of the finite volume mesh $32 \times 32 \times 32$ *cells, showing the hill surface, the northern wall, and the eastern wall i or outlet boundary.*

was prolongated to the original mesh ($32 \times 32 \times 32$ cells) where the residuals were reduced by a factor of 10000 with respect to the original residual on the coarsest mesh. A plot of the convergence history is show in Fig. 68, the characteristic jumps in the residuals are clearly seen when the solution is prolongated to the finer meshes. The number of iterations needed to get a convergent solution was equal to 164, and the CPU time used on a IBM RS6000 220h work-station was 3697 seconds.

To get an overview of the flow, two mesh slices through the three-dimensional field are shown. A slice for constant ξ indices (this plane approximates the x-plane having y equal to zero) which should resemble the two-dimensional simulation from the previous chapter, see Fig. 69. The speed-up near the hill top is easily seen as well as the deceleration of the flow in the lee of the hill.

The second mesh slice is a plane having the ζ index constant in computational space (this plane approximates the velocities at 10 m height above terrain), see Fig. 70. From this figure one can see that the flow direction changes as the flow passes the hill, and secondly, a separated region is visible in the lee of the hill near its centre axis.

Speed-up

Looking at the speed-up at the hill top in the three-dimensional computation compared to that of the two-dimensional computation, the speed-up is clearly reduced in agreement with the expectations. The value of the speed-up at the summit 10 m above the terrain is 14% too low in the three-dimensional computation, i.e. a value of 0.76 in contrast to the measured value of 0.88. As for the two-dimensional case the agreement is good upstream of the hill top, and in addition the agreement downstream of the summit is found to be good also, see Fig. 71.

Taking the vertical profile of speed-up at the hill top, we observe that far away from the surface ($z > 30$ m) the agreement seems quite good, in contrast to the

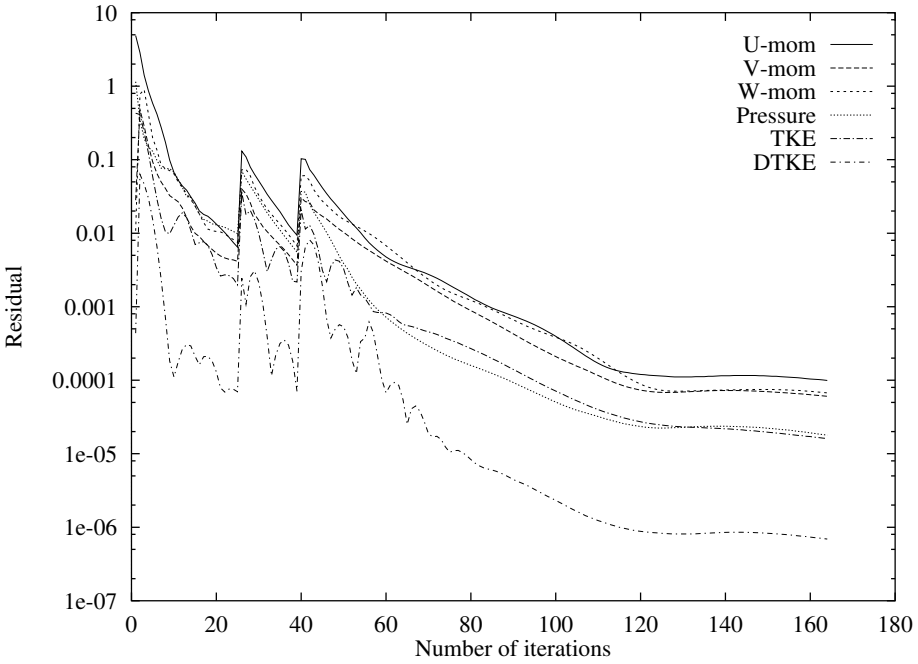

Figure 68. Convergence history for the Askervein hill calculation, the residual jumps connected to the mesh shifts in the grid sequence are clearly visible around it. = 26 and it. = 40.

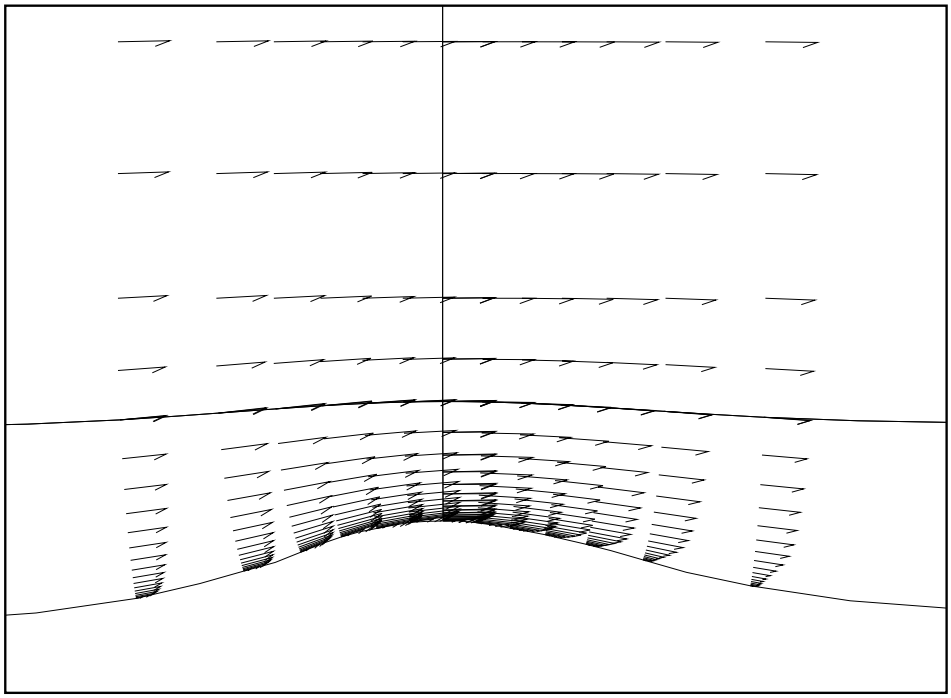

Figure 69. Velocity vectors for flow over Askervein hill near the summit for a plane having the ξ index in the computational space constant and going through HT ((x,y) = (0,0)). Only every second vector in the flow direction is shown.

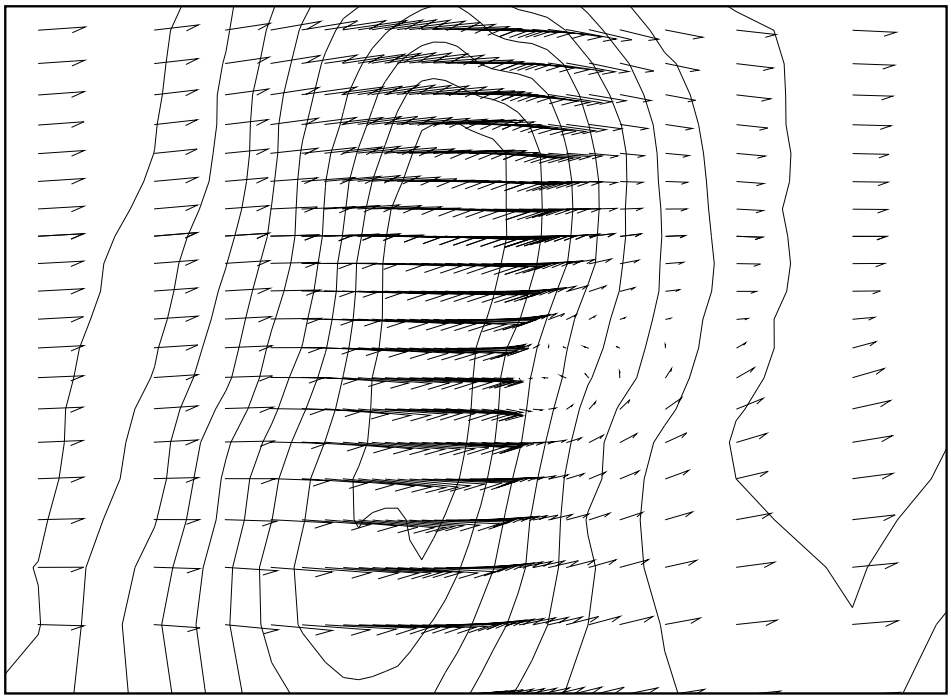

Figure 70. Velocity vectors for the flow over Askervein hill shown together with the contour lines of the hill. The plane shown, having the ζ index in the computational space constant, approximates the velocities in 10 m height above terrain.

overprediction found in the two-dimensional case, see Fig. 72.

In connection with the two-dimensional simulation, the perturbation of the far-field velocity was assumed to be caused by the two-dimensionality of the flow, resulting in too high speed-up in this region. The fact that this phenomenon is not present in the three-dimensional simulation supports the theory that the missing possibility for the flow to pass around the hill in the two-dimensional simulation is the reason for the error.

Flow angle

The change in the flow direction as the flow passes the hill, shown by the flow angle, is in good agreement with the measurements at the upwind side of the hill top. In Taylor and Teunissen [56] the hill top measurement is said to be suspect. This may allow us to exclude the hill top measurement and thereby obtain good agreement over the total range of line A. In the lee of the hill a variation is observed on a scale smaller than that resolvable by the measurements. Whether this actually is a physical feature of the flow or an error in the computation can not be judged from the present measurements. One possibility is that the flow is channeled by the main hill and the small hill downstream of the main hill in easterly direction, making a perturbation of the flow in the lee of the hill, see Fig. 70.

Turbulent kinetic energy

The behaviour of the computed turbulent kinetic energy is similar to that found in the two-dimensional computation. A good agreement is found until about 250 m downstream of the hill top, followed by a strong underprediction downstream of this point. The underprediction is slightly more pronounced than was the case in the two-dimensional computation, which is in good agreement with the fact that

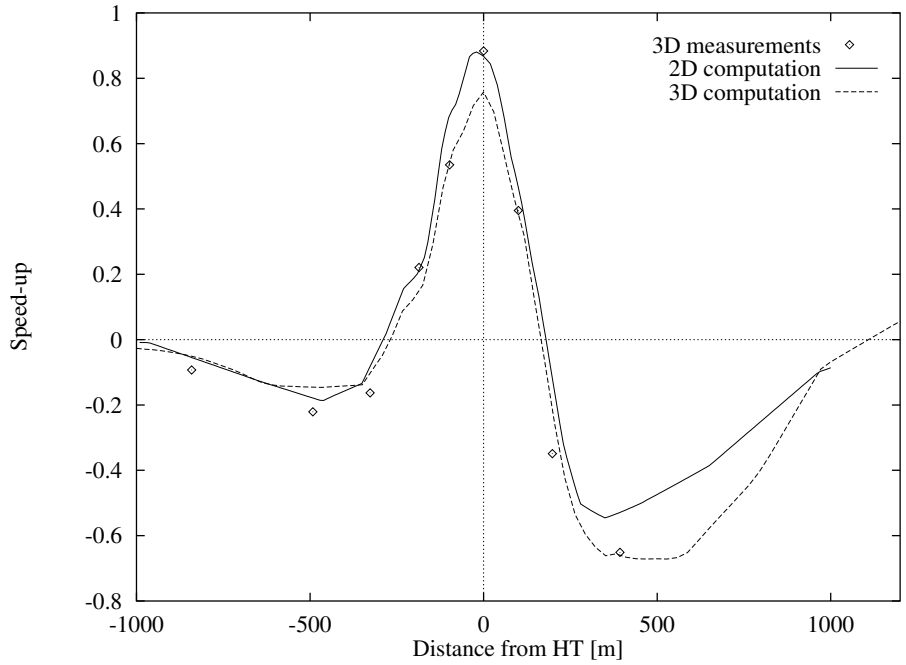

Figure 71. Speed-up over Askervein hill along line A, the two-dimensional results shown are taken from case 1 in the previous chapter.

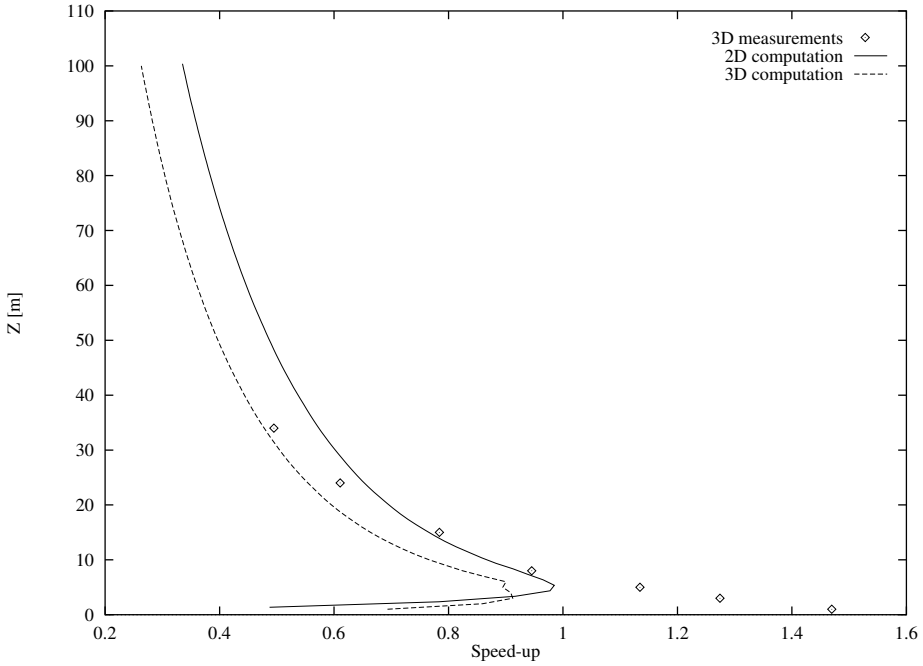

Figure 72. Speed-up along a vertical line over the hill top of the Askervein hill, the two-dimensional results shown are taken from case 1 in the previous chapter.

the speed-up is lower in the three-dimensional case leading to a lower production of turbulent kinetic energy, see Fig. 74.

15.4 Closure

Based on the information gathered from the two-dimensional simulation, a finite volume mesh for the three-dimensional flow over Askervein hill was generated

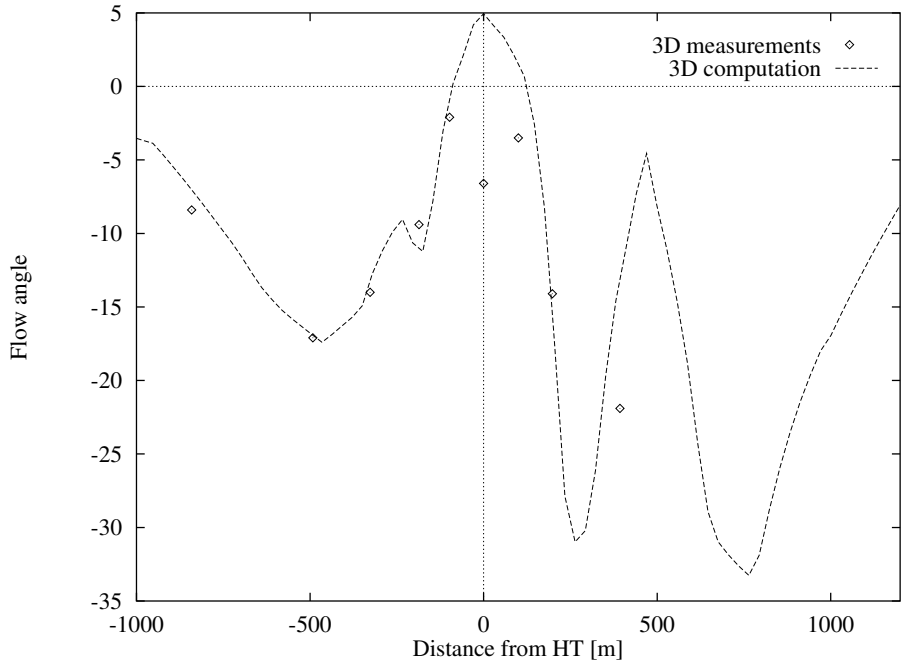

Figure 73. Change of the wind direction over Askervein hill along line A shown as a variation around the 210 degree direction, positive in the counter-clockwise direction.

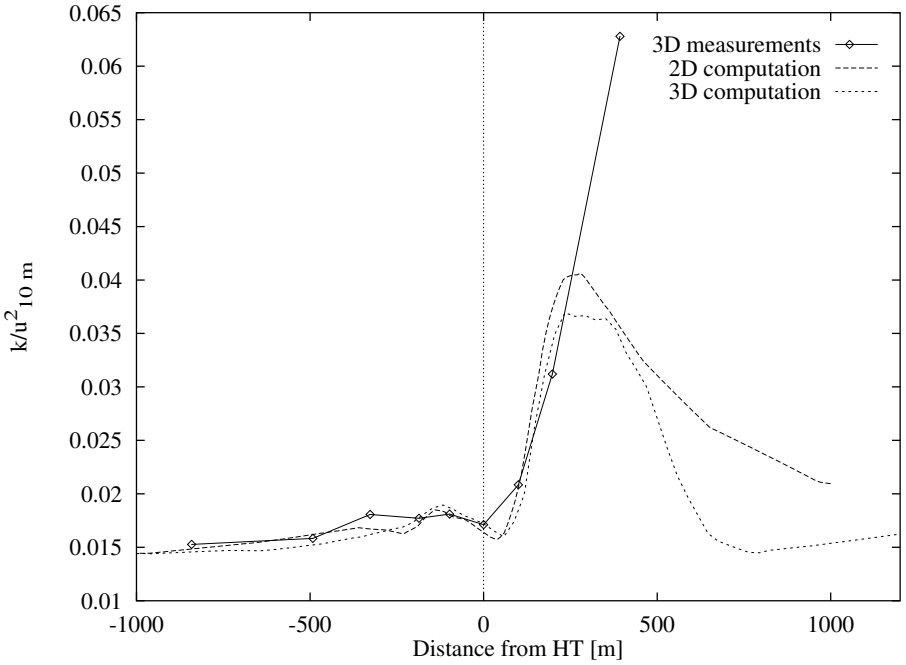

Figure 74. The turbulence intensity over Askervein hill along line A, the two-dimensional results shown are from case 1 in the previous chapter.

using the three-dimensional hyperbolic mesh generator.

The x-axis of the computational mesh was aligned with the mean flow direction, in order to minimize the numerical diffusion.

As expected from the two-dimensional simulation, the three-dimensional simulation shows a too low speed-up at the hill top. Upstream of the hill summit the agreement was found to be quite good, and additionally the speed-up was

found to be well predicted in the lee of the hill, in contrast to the findings in the two-dimensional simulation.

The shift of the wind direction as the flow passes the hill is very well predicted, if one neglects the measurement at the hill top (said to be suspect in [56]). A interesting feature is the variation of the flow angle in the lee of the hill seen on a smaller scale than that reproduced by the measurements. This could in fact be caused by the channeling between the main hill and the smaller hill downstream of the main hill in the easterly direction.

The turbulent kinetic energy is again well predicted until about 250 m downstream of the hill top. Downstream of this point a considerable underprediction is found.

The conclusion must be that the code is capable of computing flow over natural terrain, because the simulation reproduces many of the features found in the measured flow. It is unsatisfactory that the speed-up is underpredicted at the hill top. However, this will have to be investigated later.

16 Conclusion

16.1 The developed code

A two- and a three-dimensional finite-volume code in general curvilinear coordinates, have been developed. The codes are based on the Reynolds averaged incompressible isothermal Navier-Stokes equations and use primitive variables (U, V, W and P) in a non-staggered arrangement.

The codes use Rhie/Chow interpolation [47] and the standard SIMPLE approach. In the inertial phase, a simple multigrid method was implemented in order to obtain fast convergence of the pressure correction equation. Also a domain decomposition technique was implemented, allowing greater geometric flexibility than with a standard single block structured mesh. In the final codes, the Basis2D/3D platforms have been used, providing both the multigrid method used for solving the pressure correction equation, and the domain decomposition technique.

All boundary conditions were implemented using an attribute technique, where a value assigned to the cell face tells the code which physical condition is needed at this location. All types of boundary conditions are programmed in advance, allowing the user to change the boundary condition simply by changing the attribute value. The attribute technique is very user friendly and fully integrated with the Basis2D/3D platform.

Higher-order schemes (SUDS and QUICK) have been implemented as explicit corrections to a first-order upwind difference scheme. The explicit implementation makes the schemes more stable than would the natural implicit implementation.

In order to exploit the possibilities of the codes, it was necessary to develop mesh generators capable of generating general curvilinear meshes. Two different methods were implemented, a hyperbolic method and a variation of the transfinite interpolation technique. The marching nature of the hyperbolic method is well suited for generating meshes over natural terrain, where the actual position of the far-field boundary is unimportant. In contrast to this, the transfinite interpolation allows absolute control of the location of all block faces, and can be used where this property is needed.

A simple point localization and interpolation scheme for use in general curvilinear coordinates, was developed in order to facilitate the interpretion of the computed results. The method is based on dividing the computational cell into six tetrahedrons for which the point localization is straight forward. The method has been used to extract profiles from computed three-dimensional data fields, and additionally it has been used to compute particle traces for flow around a surface mounted cube.

16.2 Computed test cases

A series of test cases were calculated in order to test the different parts of the developed code. Five of these test cases are described in the present report, namely

- Laminar steady-state Driven Square Cavity flow at Reynolds number 100 in a Cartesian grid.
- Laminar steady-state separation over a circular cylinder at Reynolds number 40, in orthogonal grid (polar coordinates).
- Turbulent steady-state channel flow at Reynolds number 61000 in a rectangular grid.

- Turbulent steady-state flow trough Staggered tube banks at Reynolds number 140000 in a general non-orthogonal grid.
- Turbulent steady-state flow over Backward-facing step at Reynolds number 140000 in a 3 domain rectangular grid.

The test cases were designed to test different parts of the code, and as the implementation of the two- and three-dimensional model is identical, only the two-dimensional results are reported. From the successful calculation of these test cases it was concluded that the codes were well functioning. In connection with the test cases it was found that the accuracy of the implemented upwind difference scheme was insufficient, and as a consequence higher-order schemes were implemented. For the straight channel it was found that the $k-\epsilon$ model computed wall shear stresses about 20 % to high.

When the correctness of the codes had been verified, two large simulations were performed to illustrate the application of the code to atmospheric conditions.

First, the flow over a surface mounted cube in a thick turbulent boundary-layer was calculated and compared with the results of Castro and Robins [6]. The simulations captured the major structures of the flow, e.g. the large separated region behind the cube and the horse-shoe vortex in front of the cube. The pressure distribution on the cube was relatively well predicted, even though the lack of separation on top of the cube results in a too low value at this location. An interesting secondary flow pattern was found on the side of the cube, unfortunately this could not be verified as no measurements were reported for this location.

Secondly, flow over Askervein hill was investigated for a case corresponding to the measurements of Taylor and Teunissen [55] referred to as TU03-B. First a series of two-dimensional simulations were performed in order to find the height of the domain and the number of mesh cells necessary to make the calculation mesh and domain size independent. This resulted in a 1900 m high domain, with 32×32 cells. In the three-dimensional calculation the previous result was used, and a mesh with $32 \times 32 \times 32$ cells and 1900 m high was generated.

Looking at the speed-up prediction from the three-dimensional calculation at a height of ten meters over terrain, an obvious problem exists near the hill summit where the computed value was too low. Upstream and in the lee of the hill the predicted speed-up was in good agreement with the measurements. The calculated variation of the flow angle shows a good agreement with the measurements, except near the hill top where the measurements are said to be suspect. The turbulent kinetic energy was well predicted until 250 m downstream of the hill top, whereafter it was strongly under-predicted.

16.3 Future plans

Even though good results have already been obtained with the model, some of the features implemented in the code have yet to be documented. The most important part still needing verification is the transient module of the codes. In this connection it may prove worthwhile to implement a second-order accurate time-stepping scheme instead of the first-order scheme implemented at the moment.

As indicated by the calculated test cases, a turbulence model with greater accuracy than that of the $k-\epsilon$ model is desirable. As an alternative a nonlinear $k-\epsilon$ model could be a possibility for both atmospheric and industrial flows. Additionally, a low Reynolds number $k-\epsilon$ model could be interesting for separated flow in connection with industrial flows.

To expand the range of application of the model, additional equations may be implemented. For meteorological flows, an implementation of the energy equation would allow the calculation of stratified flow. Also, an equation for transport of

a passive scalar could be of interest in connection with pollution dispersion. As the code is constructed in a modular way, the addition of extra equation can to a large extent be accomplished using standard modules.

Acknowledgement

The present work was funded by the Danish Research Academy. The work has been carried out under guidance of professor Poul Scheel Larsen (The Department of Fluid Mechanics, Technical University of Denmark), Niels Otto Jensen (The Department of Meteorology and Wind Energy, Risø National Laboratory), and Jess Andreas Michelsen (The Department of Fluid Mechanics, Technical University of Denmark). I would like to express deep gratitude for their continuous support and interest in this project. Also I would like to thank Anne F. de Baas for the guideance during the inertial face of my study, while she was still an employe of The Department of Meteorology and Wind Energy.

A special thanks to Jess Andreas Michelsen for providing the Basis2D/3D platform, and invaluable advice on specific numerical problems during development and debugging of the codes.

Furthermore, I wish to thank Erik Andresen (International Research Centre for Computational Hydrodynamics) for letting me use the HPGLVIEW graphic display program, and Søren Lovmand Hvid (Danish Maritime Institute) for providing the mesh generator used for the staggered tube bank calculation. I also wish to thank my friends at the Department of Fluid Mechanics, along with my colleagues at the Department of Meteorology and Wind Energy.

Thanks to Birthe Skrumsager (The Department of Meteorology and Wind Energy) for careful proofreading, and to all the others who helped me prepare the manuscript.

Finally, a special thanks to Lone Bølge for her patience, and her pretending to be interested when I talked about my work.

Dansk sammendrag

Introduktion

Mange forskellige emner med tilknytning til emnet grænselagsmeteorologi studeres i Afdelingen for Meteorologi og Vindenergi.

Et af disse emner er strømning i komplekst terræn, et emne med forbindelse til mange praktiske problemer, feks. placering af vindmøller, forureningsspredning i naturligt terræn og i bymiljø.

I disse studier benyttes mange forskellige metoder, fra teoretiske studier, over fuldskala målinger og vindkanal forsøg til numeriske strømningsmodeller.

I grænselagsmeteorologi er der en lang tradition for at benytte lineariserede strømningsmodeller, se Jackson and Hunt [18], Mason and Sykes [29].I slutningen af 1980'erne påbegyndtes et studie af anvendelsen af ikkelineariserede strømningsmodeller (CFD) i forbindelse med strømning over komplekst terræn.

Projektet foregik i samarbejde med Afdelingen for Fluid Mekanik ved DTU og Skibsteknisk Laboratorium. Ved Afdelingen for Meteorologi og Vindenergi var hovedformålet med studiet at undersøge muligheden for at anvende 2D/3D CFD koder i retangulære koordinater til beregning af strømning over bakker. I dette studie blev den kommercielle CFD kode KAMELEON kode benyttet (KAMELEON koden er udviklet i Trondheim af Berge [4] og er af Patankar/Spalding typen). Der blev anvendt to forskellige metoder.

Først blev det forsøgt direkte at anvende KAMELEON til at beregne ikke rektangulære geometrier. Bakkens ikkerektangulære form blev modelleret ved udblokning i det rektangulære net, hvorved bakkens overflade blev repræsenteret ved en trappeformet kurve. Da det hovedsageligt er bakkens overflade, der styrer strømningen, kræver trappemetoden et meget fint net for at give gode resultater. Resultaterne af denne undersøgelse var da også utilfredsstillende.

Herefter blev en metode forsøgt, hvor geometrien af bakken ikke direkte blev modelleret. Ved transformation fra et kurveliniært til et rektangulært koordinatsystem opstår der ekstra led i ligningerne. Hvis bakken er lille og krumningen derfor også er lille, kan disse ekstra led i ligningerne negligeres. I stedet blev bakken modelleret ved at fastholde trykket svarende til potential strømningsløsningen over bakken. Metoden er ikke i stand til at repræsentere den vertikale strømning over bakken og er ude af stand til at beregne separeret strømning. På trods af disse begrænsninger var resultaterne lovende, se de Baas [7].

Emne for studiet

Det blev besluttet at fortsætte studiet af anvendelsen af CFD til beregning af atmosfærisk strømning. Ud fra erfaringerne med KAMELEON modellen var det klart, at modellen skulle benytte generelle kurveliniære koordinater og det hydrodynamiske tryk.

Modellen skulle være i stand til at beregne småskala[5] neutralt stratifiseret atmosfærisk strømning med separation. Den nævnte begrænsning på geometriens støerrelse tillader udeladelse af Corriolis effekter, og antagelsen om isoterm strømning gør desuden implementering af energiligningen unødvendig.

I afsnittet om resultater findes beskrivelser af anvendelse af modellen på atmosfæriske strømninger, foruden beregning af en serie testtilfælde.

[5] I forbindelse med dette studie har småskala områder en udstrækning på mindre end 10 km × 10 km.

De udviklede koder

Det blev ud fra ovenstående krav besluttet at udvikle en 2D og en 3D inkompressibel isoterm Navier-Stokes kode i generelle kurveliniære koordinater. Koderne skulle benytte en cellecentreret formulering uden forskydning af de variable (co-located arrangement), og primitive variable (U, V, W, og P) i et blokstruktureret net. Tryk/hastigheds koblingen blev opnået ved at benytte SIMPLE metoden i henhold til Patankar og Spalding [43], koblet med Rhie/Chow interpolation, se Rhie [47].

Turbulensen blev modelleret ved hjælp af en standard høj Reynolds tal $k-\epsilon$ model, med standard logaritmelovrandbetingelser. Den relativ simple $k-\epsilon$ model blev valgt, idet denne er velafprøvet, rimelig nøjagtig, og simpel at implementere. Den mere sofistikerede Reynoldsspændingsmodel blev også overvejet, men opgivet på grund af den større kompleksitet ved implementeringen og det faktum at resultaterne kun er marginalt bedre (i det generelle tilfælde) end hvad der kan opnås med en $k-\epsilon$ model.

Ved at implementere to forskellige versioner af de tubulente randbetingelser (glat og ru væg), var det muligt at gøre modellen anvendelig både til atmosfærisk og standard CFD anvendelser. Dette har den klare fordel, at de mange veldokumenterede test tilfælde fra standard CFD anvendelser kan benyttes til verificering af modellen.

For at opnå et større anvendelsesområde for modellen end en simpel enkelt blok struktureret kode tillader, blev en domæne dekompositionsmetode studeret og implementeret i en 2D test kode. Herved opnåes, at geometrier, der ikke kan transformeres til en kubus i det transformerede rum, alligevel kan behandles, feks. strømning over bygninger. I forbindelse med 2D test koden blev multigrid acceleration af trykløsningen ligeledes studeret. Der blev således implementeret en simpel Correction Grid multigrid metode med en TDMA løser som basis glatter og en fast V-cycle med fem net niveauer. I de endelige 2D/3D koder blev disse teknikker implementeret ved hjælp af Basis2D/3D platformene, udviklet af Michelsen [33].

En yderligere gevinst ved at benytte Basis2D/3D platformene var deres brug af attributter, en teknik der også var blevet afprøvet tidligere i forbindelse med den to-dimensionale testkode. Attributterne er værdier knyttet til hjørnerne af beregnings cellerne (og lagret sammen med koordinaterne i net filen), der fortæller hvilken fysisk randbetinglse, der er behov for.

Attribut systemet tillader en fast implementering af alle randbetingelser, hvilket begrænser muligheden for at introducere fejl i forbindelse med nye strømningstilfælde. I det enkelte strømnings tilfælde vælges blandt de fast implementerede randbetingelserne ifølge de givne attribut værdier. Denne teknik muligør kørsel af en hel serie test tilfælde uden at skulle ændre en eneste linie kode, hvilket er meget attraktivt i forbindelse med fejlfinding, og ligeledes i forbindelse med produktions kørsler.

For at kunne udnytte modellernes mulighed for at arbejde i generelle ikke ortogonale kurvelineære koordinater, blev der udviklet to typer netgeneratorer. Til strømning i naturligt terræn, hvor den nøjagtige beliggenheden af top randen er uden betydning, blev der udviklet en 2D og en 3D hyperbolsk netgenerator. Til anvendelse hvor der var behov for kontrol af alle blok flader, blev der implementeret en simpel variant af transfinit interpolation.

For at kunne fortolke de beregnede resultater blev en interpolationsmetode for generelle skæve hexaedere udviklet. Metoden blev blandt andet brugt til at udrage profiler fra 3D beregninger og til at beregne partiklerbaner i forbindelse med grænselagsstrømningen omkring en kubus.

Resultater

De opnåede resultater kan deles i to grupper, testtilfælde og anvendelse. Først blev en serie testtilfælde beregnet for at verificerer koden, alle de her dokumenterede er 2D tilfælde. Generelt var der en god overensstemmelse mellem testdata og de beregnede data, dog blev det klart, at det simple førsteordens upwind skema ikke var tilstrækkelig nøjagtigt. Som følge af dette blev der implementeret et andenordens upwind skema (SUDS) og ligeledes et tredieordens QUICK skema. De beregnede test tilfælde vil blot blive nævnt her, men ikke yderligere kommenteret. De beregnede tilfælde var:

- Laminar stationær strømning i kvadratisk dreven kavitet (Driven cavity) ved Reynoldstal 100, i kartesisk net.
- Laminar stationær strømning omkring en cirkulær cylinder ved Reynoldstal 40, i polære koordinater.
- Turbulent stationær strømning i ret kanal ved Reynoldstal 61000, i rektangulært net.
- Turbulent stationær strømning i en forskudt rør varmeveksler ved Reynoldstal 140000, i generelle ikkeortogonale kurveliniære koordinater.
- Turbulent stationær strømning over backward-facing step ved Reynolds tal 140000, i et 3 domæne rektangulært net.

Anvendelsen har i det nærværende arbejde knyttet sig til atmosfærisk strømning, dette er et bevidst valg og ikke en begrænsning i modellen. To tilfælde blev beregnet, strømning omkring en kubus i et tykt turbulent grænselag [6], og atmosfærisk strømning over Askervein [56].

For strømningen over kuben var resultaterne gode. Således reproducerede beregningen mange af de større flow strukturer omkring kuben, feks. det store seperarede område bag kuben, og hestesko hvirvlen foran kuben. Trykfordelingen på kubens overflade var generelt i god overensstemmelse med målingerne, dog var den underestimeret på toppen af kuben. Årsagen til den lave værdi af trykket oven på kuben skyldes formodenlig, at simuleringen ikke fanger det lille seperarede område her.

Strømningen over Askervein var ligeledes vellykket, idet den beregnede speed-up værdi 10 m over terræn stemmer overens med målingerne, undtagen nær toppen af bakken hvor værdien var underestimeret. Den beregnede værdi af vindens drejning ved passage af Askervein, udviser ligeledes god overensstemmelse med målinger, nær toppen hvor overensstemmelsen er dårlig betvivler Taylor og Teunnissen [56] rigtigheden af målingen. For den turbulente kinetiske energi er resultaterne gode indtil cirka 250 m nedstrøms for bakketoppen, hvorefter der optræder en markant underestimering.

Fremtidsplaner

Selv om gode resultater således allerede er blevet opnået med modellerne, er der stadig nogle af de implementerede muligheder, der mangler at blive verificeret. Den væsentligste af disse er muligheden for transiente beregninger. I forbindelse med dette kan det vise sig at være interessant at implementere et anden ordens nøjagtigt tidsskema i stedet for det nuværende førsteordens skema.

De beregnede tilfælde peger desuden på, at $k-\epsilon$ modellen ikke er tilstrækkelig i visse tilfælde. Mere avancerede modeller, eksempelvis en ikkelineær $k-\epsilon$ model kunne være en mulighed, både for atmosfærisk strømning og standard CFD anvendelser. Desuden kunne en lav Reynolds tal $k-\epsilon$ model være en mulig forbedring i forbindelse med standard CFD anvendelser.

For at udvide modellernes anvendelsesområde, kunne yderligere ligninger implementeres. For meteorologiske strømninger kunne implementering af energiligningen være interessant i forhold til stratifiseret strømning. Ligeledes ville en skalar transportligning være interessant i forbindelse med forureningsspredning. Implementeringen af yderligere ligninger kan i stor udstrækning foregå ved hjælp af eksisterende subrutiner, idet koden er opbygget i en modulær struktur.

References

[1] V.S. Arpaci and P.S. Larsen. *Convection Heat Transfer*. Prentice-Hall, Inc., Englewood Cliffs, 1984.

[2] F. Baetke, H. Werner, and H. Wengle. Numerical simulation of turbulent flow over surface-mounted obstacles with sharp edges and corners. *J. Wind Eng. Ind. Aerodyn.*, 35:129–147, 1990.

[3] A.C.M. Beljaars, J.L. Walmsley, and P.A. Taylor. A Mixed Spectral Finite-Difference Model for Neutrally Stratified Boundary-Layer Flow over Roughness Changes and Topography. *Boundary-Layer Meteorol.*, 38:273–303, 1987.

[4] G. Berge. *Numerisk programsystem baseret på fleirdimensjonale ortogonale koordinatar for simulering av forbernning, masse og varmetransport.* PhD thesis, University of Trondheim, Juli 1982.

[5] A.J. Bowen and H.W. Teunissen. Wind-Tunnel Simulation at Length Scale 1:2500 Part 1: Tests at the Atmosperic Enviroment Service Volume I + II. Research Report 86-7, Atmos. Environ. Service, Downsview, Ontario, 1986.

[6] I.P. Castro and A.G. Robins. The flow around a surface-mounted cube in uniform and turbulent streams. *J. Fluid Mech.*, 79(2):307–335, 1977.

[7] A.F. de Baas. Final report of the project >>Numerical Modelling<<. Technical report, Risø National Laboratory, DK-4000 Roskilde, Denmark, September 1990.

[8] G. de Vahl Davis and G.D. Mallinson. False Diffusion in Numerical Fluid Mechanics. Rept 1972/fmt/1, Univ. of New South Wales, School of Mech. and Ind. Eng., 1972.

[9] S.C.R. Dennis and Gau-Zu Chang. Numerical solution for steady flow past a circular cylinder at Reynolds numbers up to 100. *J. Fluid Mech.*, 42(3):471–489, 1970.

[10] J.H. Ferziger. Approaches to turbulent flow computation: applications to flow over obstacles. *J. Wind Eng. Ind. Aerodyn.*, 35:1–19, 1990.

[11] U. Ghia, K.N. Ghia, and G.T. Shin. Heigh-Re solutions for incompressible Flow using the Navier-Stokes equations and a multigrid methode. *J. Computational Phys.*, 48:387–411, 1982.

[12] L.P. Hackman. *A numerical study of the turbulent flow over a backward facing step using a two equation turbulence model.* PhD thesis, University of Waterloo, 1982.

[13] R.G. Hindman. Generalized coordinate form of governing fluid equations and associated geometrically indeuced errors. *AIAA J.*, 20:1359–1367, 1982.

[14] R.P. Hosker. Flow and Diffusion Near Obstacles. In *Atmospheric Science and Power Production*, pages 241–326, Springfield, VA (US), 1984. The Technical Information Center.

[15] J.C.R. Hunt, C.J. Abell, J.A. Peterka, and H. Woo. Kinematical studies of the flows around free or surface-mounted obstacles: applying topology to flow visualization. *J. Fluid Mech.*, 86:179–200, 1978.

[16] J.C.R. Hunt, R.J. Perkins, H. Kawai, and S.R. Ramsey. A review of velocity and pressure fluctuations in turbulent flows around bluff bodies. *J. Wind Eng. Ind. Aerodyn.*, 35:49–85, 1990.

[17] S.L. Hvid. *Three Dimensional Algebraic Grid Generation.* PhD thesis, Technical University of Denmark, Lyngby, August 1994.

[18] P.S. Jackson and J.C.R. Hunt. Turbulent Wind Flow Over a Low Hill. *Quart. J. R. Met. Soc.*, 101:929–955, 1975.

[19] J.J. Kim. *Investigation of Separation and Reattachment of a Turbulent Shear Layer: Flow over Backward Facing Step.* PhD thesis, Stanford University, 1978.

[20] W. Kordulla and M. Vinokur. Efficient computation of the volume in flow predictions. *AIAA*, 21:917–918, 1983.

[21] J. Laufer. Investigation of Turbulent Flow in a Two-Dimensional Channel. Technical Report 1053, NASA, 1951.

[22] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Comput. Meths. Appl. Mech. Eng.*, 3:269–289, 1974.

[23] B.P. Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Comput. Meths. Appl. Mech. Eng.*, 19:59–98, 1979.

[24] M.L. Levitan and K.C. Mehta. Texas Tech field experiments for wind loads part 1: building and pressure measuring system. *J. Wind Eng. Ind. Aerodyn.*, 41-44:1565–1576, 1992.

[25] M.L. Levitan and K.C. Mehta. Texas Tech field experiments for wind loads part II: meteorological instrumentation and terrain parameters. *J. Wind Eng. Ind. Aerodyn.*, 41-44:1577–1588, 1992.

[26] J.N. Lillington. An assessment of vector upstream differencing scheme predictions for the transport of a scalar (eg. temperature) in unidirectional recirculating flow regimes. *AEWW-M 1833*, 1981.

[27] R. Martinuzzi and C. Tropea. The Flow Around Surface-Mounted, Prismatic Obstacles Placed in a Fully Developed Channel Flow. *J. Fluids Eng.*, 115:85–92, March 1993.

[28] P.J. Mason and J.C. King. Measurements and predicitions of flow and turbulence over an isolated hill of moderate slope. *Quart. J. R. Met. Soc.*, 111:617–640, 1985.

[29] P.J. Mason and R.I. Sykes. Flow over an Isolated Hill of Moderate Slope. *Quart. J. R. Met. Soc.*, 105:383–395, 1979.

[30] J.A. Michelsen. Solution of the Navier-Stokes equations on general non-orthogonal meshes. Technical Report AFM 89-08, Technical University of Denmark, Lyngby, 1989.

[31] J.A. Michelsen. Multigrid Acceleration of Velocity-Pressure Coupled Navier-Stokes Solver. Technical Report AFM 90-13, Technical University of Denmark, Lyngby, September 1990.

[32] J.A. Michelsen. Mesh-Adaptive Solution of the Navier-Stokes Equations. *Int. Series of Num. Math.*, 98, 1991. Birkhäuser.

[33] J.A. Michelsen. Basis3D - a Platform for Development of Multiblock PDE Solvers. Technical Report AFM 92-05, Technical University of Denmark, Lyngby, 1992.

[34] J.A. Michelsen. Block structured Multigrid solution of 2D and 3D elliptic PDE's. Technical Report AFM 94-06, Technical University of Denmark, Lyngby, 1994.

[35] A.C. Mikkelsen and F.M. Livesey. Evaluation of the Use of the Numerical k-ϵ Model Kameleon II, for Predicting Wind Pressure on Building Surfaces. In *IAWE 1st European and African Regional Conference, Guernsey, Channel Island*, September 1993.

[36] N.G. Mortensen, L. Landberg, I. Troen, and E.L. Petersen. *Wind Atlas Analysis and Application Program (WA^{s}P) Vol. 2: User's Guide.* Risoe National Laboratory, Roskilde, Denmark, 1993. Risoe-I-666(EN)(v.2).

[37] S. Murakami and A. Mochida. 3-D numerical simulation of airflow around a cubic model by means of the $k-\epsilon$ model. *J. Wind Eng. Ind. Aerodyn.*, 31:283–303, 1988.

[38] S. Murakami, A. Mochida, and Y. Hayashi. Examining the $k-\epsilon$ model by means of a wind tunnel test and large-eddy simulation of the turbulence structure around a cube. *J. Wind Eng. Ind. Aerodyn.*, 35:87–100, 1990.

[39] S. Murakami, A. Mochida, Y. Hayashi, and S. Sakamoto. Numerical study on velocity-pressure field and wind forces for bluff bodies by $k-\epsilon$, ASM and LES. *J. Wind Eng. Ind. Aerodyn.*, 41-44:2841–2852, 1992.

[40] R. Panneer Selvam. Computation of Pressure on Texas Tech Building. *J. Wind Eng. Ind. Aerodyn.*, 41-44:1619–1627, 1992.

[41] H.A. Panofsky and J.A. Dutton. *Atmospheric Turbulence Models and Methods for Engineering Applications.* John Wiley & Sons, New York, 1984.

[42] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow.* Hemisphere, New York, 1980.

[43] S.V. Patankar and D.B. Spalding. A Calculation Prodedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows. *Int. J. Heat Mass Transfer*, 15:1787, 1972.

[44] D.A. Paterson and C.J. Apelt. Simulation of flow past a cube in a turbulent boundary layer. *J. Wind Eng. Ind. Aerodyn.*, 35:149–176, 1990.

[45] M. Peric. *A finit volume method for prediction of three-dimensional fluid flow in complex ducts.* PhD thesis, Univ. of London, 1985.

[46] G.D. Raithby, G.D. Stubley, and P.A. Taylor. The Askervein Hill Project: A Finite Control Volume Prediction of Three-Dimensional Flows Over the Hill. *Boundary-Layer Meteorol.*, 39:247–267, 1987.

[47] C.M. Rhie. *A numerical study of the flow past an isolated airfoil with separation.* PhD thesis, Univ. of Illinois, Urbane-Champaign, 1981.

[48] A. Rizzi, P. Eliasson, I. Lindblad, C. Hirsch, C. Lacor, and J. Haeuser. The Engineering of Multiblock/Multigrid Software for Navier-Stokes Flows on Structured Meshes. *Computers Fluids*, 22(2/3):341–367, 1993.

[49] A.G. Robins and I.P. Castro. A Wind Tunnel Investigation of Plume Dispersion in the Vincinity of a Surface Mounted Cube-I. The Flow Field. *Atmospheric Enviroment*, 11:291–297, 1977.

[50] W.H. Schofield and E. Logan. Turbulent Shear Flow Over Surface Mounted Obstacles. *J. Fluids Eng.*, 112:376–385, 1990.

[51] H.A. Schwarz. Über einige abbildungsaufgaben. *Journal reine angew. Mathematik*, 70:105–120, 1869.

[52] T. Statopoulos and A. Baskaran. Boundary treatment for the computation of three-dimensional wind flow conditions around a building. *J. Wind Eng. Ind. Aerodyn.*, 35:177–200, 1990.

[53] J.L. Steger and D.S. Chaussee. Generation of Body-Fitted Coordinates using Hyperbolic Partial Differential Equations. *SIAM J. Sci. Stat. Comput.*, 1(4):431–437, December 1980.

[54] J.L. Steger and R.L. Sorenson. Use of Hyperbolic Partial Differential Equations to Generate Body Fitted Coordinates. In *Numerical Grid Generation Techniques*. Nasa, 1980. NASA Conference Publication 2166.

[55] P.A. Taylor and H.W. Teunissen. ASKERVEIN '82: Report on the September/October 1982 Experiment to Study Boundary-Layer Flow over Askervein, South Uist. Technical Report MSRB-83-8, Atmos. Environ. Service, Downsview, Ontario, 1983.

[56] P.A. Taylor and H.W. Teunissen. The Askervein Hill Project: Report on the Sept./Oct. 1983, Main Field Experiment. Technical Report MSRB-84-6, Atmos. Environ. Service, Downsview, Ontario, 1984.

[57] H. Tennekes and J.L. Lumley. *A First Course in Turbulence*. MIT Press, Cambridge, MA (US), 3 edition, 1974.

[58] H.W. Teunissen and M.E. Shokr. The Askervein Hill Project: Wind-Tunnel Simulation (Smooth Model) at Length Scale 1:1200 Volume I + II. Research Report MSRB-85-1, Atmos. Environ. Service, Downsview, Ontario, 1985.

[59] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin. *Numerical Grid Generation Foundations and Applications*. North Holland, New York, 1985.

[60] I. Troen and E.L. Petersen. *European Wind Atlas*. Risø National Lab., Roskilde, Denmark, 1989.

[61] H. Viviand. Formes conservatives des equations de la dynamique de gas. *La Recherche Aerospatiale*, 1, 1974.

[62] J.L. Walmsley and J.R. Salmon. A Boundary-Layer Model for Wind Flow over Hills: Comparison of Model Results with Askervein 1983 Data. In *European Wind Energy Conference*, Hamburg, Germany, 1984.

[63] T. Wei-Pai. *Schwarz Splitting and Template Operators*. PhD thesis, Center of Large Scale Scientific Computation, Stanford Univ., 1987.

[64] R.W. Yeo, P.E. Wood, and A.N. Hrymak. A numerical study of laminar 90-degree bend duct flow with different discretization schemes. *J. Fluids Eng.*, 113:563–568, 1991.

[65] O. Zeman and N.O. Jensen. Modification of turbulence characteristics in flow over hills. *Quart. J. R. Met. Soc.*, 113:55–80, 1987.

[66] C.X. Zhang. Numerical predictions of turbulent recirculating flows with a $k - \epsilon$ model. *J. Wind Eng. Ind. Aerodyn.*, 51:177–201, 1994.

A Scalar transport equations

A.1 Introduction

The partial differential equation governing the transport of a scalar quantity will be treated in this appendix. Many physical phenomena can be described by a scalar transport equation, an example is the energy equation. The turbulence model used in the present study is another example of a system governed by scalar partial differential equations, the finite-volume equivalent of these two equations will be given as special cases of the generic scalar transport equation in the present appendix.

A.2 Scalar equation

The steps necessary to obtain the finite-volume equivalent of the generic scalar transport equation will be given including the transformation and the discritization.

Cartesian equation

The generic scalar transport equation can be written in Cartesian coordinates in the following way

$$\frac{\partial \rho\phi}{\partial t} + \frac{\partial \rho U\phi}{\partial x} + \frac{\partial \rho V\phi}{\partial y} + \frac{\partial \rho W\phi}{\partial z} - \frac{\partial}{\partial x}\left[\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\frac{\partial \phi}{\partial x}\right] - \frac{\partial}{\partial y}\left[\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\frac{\partial \phi}{\partial y}\right] - \frac{\partial}{\partial x}\left[\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\frac{\partial \phi}{\partial z}\right] = S\phi_V \ ,$$

where $S\phi_V$ represents any volume source that may be present.

Transformed equation

Using the transformation relations given in Chapter 3, the Cartesian equation can be transformed to a curvilinear coordinate system, resulting in the following equation

$$\begin{aligned}
&\frac{\partial \rho J\phi}{\partial t} + \frac{\partial C_1\phi}{\partial \xi} + \frac{\partial C_2\phi}{\partial \eta} + \frac{\partial C_3\phi}{\partial \zeta} - \frac{\partial}{\partial \xi}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\beta_{\xi x}\frac{\partial \phi}{\partial \xi}\right] \\
&\quad - \frac{\partial}{\partial \eta}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\beta_{\eta y}\frac{\partial \phi}{\partial \eta}\right] - \frac{\partial}{\partial \zeta}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\beta_{\zeta z}\frac{\partial \phi}{\partial \zeta}\right] \\
&\quad - \frac{\partial}{\partial \xi}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\xi y}\frac{\partial \phi}{\partial \eta} + \beta_{\xi z}\frac{\partial \phi}{\partial \zeta}\right)\right] \\
&\quad - \frac{\partial}{\partial \eta}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\eta x}\frac{\partial \phi}{\partial \xi} + \beta_{\eta z}\frac{\partial \phi}{\partial \zeta}\right)\right] \\
&\quad - \frac{\partial}{\partial \zeta}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\zeta x}\frac{\partial \phi}{\partial \xi} + \beta_{\zeta y}\frac{\partial \phi}{\partial \eta}\right)\right] = JS\phi_V \ .
\end{aligned}$$

The following definitions have been used

$$\begin{aligned}
C_1 &= \rho U\alpha_{\xi x} + \rho V\alpha_{\xi y} + \rho W\alpha_{\xi z} \ , \\
C_2 &= \rho U\alpha_{\eta x} + \rho V\alpha_{\eta y} + \rho W\alpha_{\eta z} \ , \\
C_3 &= \rho U\alpha_{\zeta x} + \rho V\alpha_{\zeta y} + \rho W\alpha_{\zeta z} \ ,
\end{aligned}$$

$$
\begin{aligned}
\beta_{11} &= \alpha_{\xi x}\alpha_{\xi x} + \alpha_{\xi y}\alpha_{\xi y} + \alpha_{\xi z}\alpha_{\xi z}, \\
\beta_{12} &= \alpha_{\xi x}\alpha_{\eta x} + \alpha_{\xi y}\alpha_{\eta y} + \alpha_{\xi z}\alpha_{\eta z}, \\
\beta_{13} &= \alpha_{\xi x}\alpha_{\zeta x} + \alpha_{\xi y}\alpha_{\zeta y} + \alpha_{\xi z}\alpha_{\zeta z}, \\
\beta_{21} &= \alpha_{\eta x}\alpha_{\xi x} + \alpha_{\eta y}\alpha_{\xi y} + \alpha_{\eta z}\alpha_{\xi z}, \\
\beta_{22} &= \alpha_{\eta x}\alpha_{\eta x} + \alpha_{\eta y}\alpha_{\eta y} + \alpha_{\eta z}\alpha_{\eta z}, \\
\beta_{23} &= \alpha_{\eta x}\alpha_{\zeta x} + \alpha_{\eta y}\alpha_{\zeta y} + \alpha_{\eta z}\alpha_{\zeta z}, \\
\beta_{31} &= \alpha_{\zeta x}\alpha_{\xi x} + \alpha_{\zeta y}\alpha_{\xi y} + \alpha_{\zeta z}\alpha_{\xi z}, \\
\beta_{32} &= \alpha_{\zeta x}\alpha_{\eta x} + \alpha_{\zeta y}\alpha_{\eta y} + \alpha_{\zeta z}\alpha_{\eta z}, \\
\beta_{33} &= \alpha_{\zeta x}\alpha_{\zeta x} + \alpha_{\zeta y}\alpha_{\zeta y} + \alpha_{\zeta z}\alpha_{\zeta z} \ .
\end{aligned}
$$

Discretized equation

Finally, the transformed equation can be discretized using the same procedure as applied in Chapter 4. The resulting finite volume equation is

$$
\begin{aligned}
&A_P\phi_P^{t+\Delta t} + A_W\phi_W^{t+\Delta t} + A_E\phi_E^{t+\Delta t} + A_S\phi_S^{t+\Delta t} + A_N\phi_N^{t+\Delta t} \\
&\quad + A_B\phi_B^{t+\Delta t} + A_T\phi_T^{t+\Delta t} = JS\phi_T + JS\phi_V + JS\phi_F + JS\phi_C \ ,
\end{aligned}
$$

where

$$
\begin{aligned}
A_W &= A_W^{dn} + A_W^c \ , \\
A_E &= A_E^{dn} + A_E^c \ , \\
A_S &= A_S^{dn} + A_S^c \ , \\
A_N &= A_N^{dn} + A_N^c \ , \\
A_B &= A_B^{dn} + A_B^c \ , \\
A_T &= A_T^{dn} + A_T^c \ , \\
A_P &= -A_W - A_E - A_S - A_N - A_B - A_T + \frac{\rho_P J_P}{\Delta t} \ , \\
JS\phi_T &= \frac{\rho_P J_P \phi_P^t}{\Delta t}, \\
JS\phi_V &= 0 \ , \\
JS\phi_F &= -I\phi_e^{dc} + I\phi_w^{dc} - I\phi_n^{dc} + I\phi_s^{dc} - I\phi_t^{dc} + I\phi_b^{dc} \ , \\
JS\phi_C &= 0 \ .
\end{aligned}
$$

The expressions for the normal diffusive terms A^{dn} can be found in Chapter 4 along with the expressions for the convective terms A^c. The expressions for the cross-diffusive terms are different from the ones given for the momentum equations and will be given here

$$
\begin{aligned}
I\phi_e^{dc} &= \left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\xi y}\frac{\partial\phi}{\partial\eta} + \beta_{\xi z}\frac{\partial\phi}{\partial\zeta}\right)\right]_e \ , \\
I\phi_w^{dc} &= \left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\xi y}\frac{\partial\phi}{\partial\eta} + \beta_{\xi z}\frac{\partial\phi}{\partial\zeta}\right)\right]_w \ , \\
I\phi_n^{dc} &= \left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\eta x}\frac{\partial\phi}{\partial\xi} + \beta_{\eta z}\frac{\partial\phi}{\partial\zeta}\right)\right]_n \ ,
\end{aligned}
$$

$$
\begin{aligned}
I\phi_s^{dc} &= \left[\frac{1}{J}\left(\mu+\frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\eta x}\frac{\partial\phi}{\partial\xi}+\beta_{\eta z}\frac{\partial\phi}{\partial\zeta}\right)\right]_s , \\
I\phi_t^{dc} &= \left[\frac{1}{J}\left(\mu+\frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\zeta x}\frac{\partial\phi}{\partial\xi}+\beta_{\zeta y}\frac{\partial\phi}{\partial\eta}\right)\right]_t , \\
I\phi_b^{dc} &= \left[\frac{1}{J}\left(\mu+\frac{\mu_t}{\sigma_\phi}\right)\left(\beta_{\zeta x}\frac{\partial\phi}{\partial\xi}+\beta_{\zeta y}\frac{\partial\phi}{\partial\eta}\right)\right]_b .
\end{aligned}
$$

A.3 k-equation

The transport equation for turbulent kinetic energy is a scalar partial differential equation and can thus be discretized using the procedure just described for scalar equations. The only terms needing special attention is the source terms. The partial differential equation for turbulent kinetic energy can be stated in Cartesian coordinates, namely

$$
\begin{aligned}
&\frac{\partial\rho k}{\partial t}+\frac{\partial\rho U k}{\partial x}+\frac{\partial\rho V k}{\partial y}+\frac{\partial\rho W k}{\partial z}-\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x}\right] \\
&\qquad -\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial y}\right]-\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial z}\right]=Sk_P-\rho\epsilon .
\end{aligned}
$$

Production term

In Cartesian coordinates the production term Sk_P is given

$$
\begin{aligned}
Sk_P &= \mu\left[2\left(\frac{\partial U}{\partial x}\right)^2+\left(\frac{\partial U}{\partial y}\right)^2+\left(\frac{\partial U}{\partial z}\right)^2\right] \\
&+\mu\left[\left(\frac{\partial V}{\partial x}\right)^2+2\left(\frac{\partial V}{\partial y}\right)^2+\left(\frac{\partial V}{\partial z}\right)^2\right] \\
&+\mu\left[\left(\frac{\partial W}{\partial x}\right)^2+\left(\frac{\partial W}{\partial y}\right)^2+2\left(\frac{\partial W}{\partial z}\right)^2\right] \\
&+2\mu\left[\frac{\partial U}{\partial y}\frac{\partial V}{\partial x}+\frac{\partial U}{\partial z}\frac{\partial W}{\partial x}+\frac{\partial V}{\partial z}\frac{\partial W}{\partial y}\right]
\end{aligned}
$$

As the production term will be treated explicitly, we will simply compute the individual velocity gradients $\partial U/\partial x$ etc., and then insert these expressions in the equation for the production term.

$$
\begin{aligned}
\frac{\partial U}{\partial x} &= \frac{1}{J}\left(\frac{\partial U}{\partial\xi}\alpha_{\xi x}+\frac{\partial U}{\partial\eta}\alpha_{\eta x}+\frac{\partial U}{\partial\zeta}\alpha_{\zeta x}\right) , \\
\frac{\partial U}{\partial y} &= \frac{1}{J}\left(\frac{\partial U}{\partial\xi}\alpha_{\xi y}+\frac{\partial U}{\partial\eta}\alpha_{\eta y}+\frac{\partial U}{\partial\zeta}\alpha_{\zeta y}\right) , \\
\frac{\partial U}{\partial z} &= \frac{1}{J}\left(\frac{\partial U}{\partial\xi}\alpha_{\xi z}+\frac{\partial U}{\partial\eta}\alpha_{\eta z}+\frac{\partial U}{\partial\zeta}\alpha_{\zeta z}\right) , \\
\frac{\partial V}{\partial x} &= \frac{1}{J}\left(\frac{\partial V}{\partial\xi}\alpha_{\xi x}+\frac{\partial V}{\partial\eta}\alpha_{\eta x}+\frac{\partial V}{\partial\zeta}\alpha_{\zeta x}\right) , \\
\frac{\partial V}{\partial y} &= \frac{1}{J}\left(\frac{\partial V}{\partial\xi}\alpha_{\xi y}+\frac{\partial V}{\partial\eta}\alpha_{\eta y}+\frac{\partial V}{\partial\zeta}\alpha_{\zeta y}\right) , \\
\frac{\partial V}{\partial z} &= \frac{1}{J}\left(\frac{\partial V}{\partial\xi}\alpha_{\xi z}+\frac{\partial V}{\partial\eta}\alpha_{\eta z}+\frac{\partial V}{\partial\zeta}\alpha_{\zeta z}\right) ,
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial W}{\partial x} &= \frac{1}{J}\left(\frac{\partial W}{\partial \xi}\alpha_{\xi x} + \frac{\partial W}{\partial \eta}\alpha_{\eta x} + \frac{\partial W}{\partial \zeta}\alpha_{\zeta x}\right) , \\
\frac{\partial W}{\partial y} &= \frac{1}{J}\left(\frac{\partial W}{\partial \xi}\alpha_{\xi y} + \frac{\partial W}{\partial \eta}\alpha_{\eta y} + \frac{\partial W}{\partial \zeta}\alpha_{\zeta y}\right) , \\
\frac{\partial W}{\partial z} &= \frac{1}{J}\left(\frac{\partial W}{\partial \xi}\alpha_{\xi z} + \frac{\partial W}{\partial \eta}\alpha_{\eta z} + \frac{\partial W}{\partial \zeta}\alpha_{\zeta z}\right) .
\end{aligned}
$$

All the gradients are evaluated in the centre of the cells by old values.

Dissipation term

In order to obtain a more stable equation, the dissipation term will be rewritten, namely

$$\rho\epsilon = \rho^2 C_\mu \frac{k^2}{\mu_t} ,$$

and linearized

$$\rho\epsilon = \rho^2 C_\mu \frac{k^t}{\mu_t} k^{t+\Delta t} .$$

The discrete finite volume equation for the turbulent kinetic energy can thus be expressed as

$$
\begin{aligned}
&A_P k_P^{t+\Delta t} + A_W k_W^{t+\Delta t} + A_E k_E^{t+\Delta t} + A_S k_S^{t+\Delta t} + A_N k_N^{t+\Delta t} \\
&\quad + A_B k_B^{t+\Delta t} + A_T k_T^{t+\Delta t} = JSk_T + JSk_V + JSk_F + JSk_C + JSk_P ,
\end{aligned}
$$

where

$$
\begin{aligned}
A_W &= A_W^{dn} + A_W^c , \\
A_E &= A_E^{dn} + A_E^c , \\
A_S &= A_S^{dn} + A_S^c , \\
A_N &= A_N^{dn} + A_N^c , \\
A_B &= A_B^{dn} + A_B^c , \\
A_T &= A_T^{dn} + A_T^c , \\
A_P &= -A_W - A_E - A_S - A_N - A_B - A_T + \frac{\rho_P J_P}{\Delta t} + \rho_P^2 J_P C_\mu \frac{k_P^t}{\mu_t} , \\
JSk_T &= \frac{\rho_P J_P k_P^t}{\Delta t}, \\
JSk_V &= 0 , \\
JSk_F &= -Ik_e^{dc} + Ik_w^{dc} - Ik_n^{dc} + Ik_s^{dc} - Ik_t^{dc} + Ik_b^{dc} , \\
JSk_C &= 0 , \\
JSk_P &= \text{ : This term is given above.}
\end{aligned}
$$

A.4 ϵ-equation

The transport equation for dissipation of turbulent kinetic energy is a scalar partial differential equation, and can thus be discretized using the procedure described in the first part of this appendix. The only terms needing special attention are the

source terms.The partial differential equation for dissipation of turbulent kinetic energy can be stated in Cartesian coordinates, namely

$$\frac{\partial \rho\epsilon}{\partial t} + \frac{\partial \rho U\epsilon}{\partial x} + \frac{\partial \rho V\epsilon}{\partial y} + \frac{\partial \rho W\epsilon}{\partial z} - \frac{\partial}{\partial x}\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x}\right] - \frac{\partial}{\partial y}\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial y}\right] - \frac{\partial}{\partial x}\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial z}\right] = S\epsilon_P - \rho C_{\epsilon 2}\frac{\epsilon^2}{k} \ .$$

Production term

The production term of dissipation of turbulent kinetic energy is modelled as

$$S\epsilon_P = C_{\epsilon 1}\frac{\epsilon}{k} S k_P \ .$$

The term representing the loss of isotropic dissipation will be linearized to obtain a more stable discrete equation

$$\rho C_{\epsilon 2}\frac{\epsilon^2}{k} = \rho C_{\epsilon 2}\frac{\epsilon^t}{k^t}\epsilon^{t+\Delta t} \ .$$

The discrete finite volume equation for the dissipation of turbulent kinetic energy can thus be expressed as

$$A_P\epsilon_P^{t+\Delta t} + A_W\epsilon_W^{t+\Delta t} + A_E\epsilon_E^{t+\Delta t} + A_S\epsilon_S^{t+\Delta t} + A_N\epsilon_N^{t+\Delta t} + A_B\epsilon_B^{t+\Delta t} + A_T\epsilon_T^{t+\Delta t} = S\epsilon_T + S\epsilon_V + S\epsilon_F + S\epsilon_C + S\epsilon_P \ ,$$

where

$$\begin{aligned}
A_W &= A_W^{dn} + A_W^c \ , \\
A_E &= A_E^{dn} + A_E^c \ , \\
A_S &= A_S^{dn} + A_S^c \ , \\
A_N &= A_N^{dn} + A_N^c \ , \\
A_B &= A_B^{dn} + A_B^c \ , \\
A_T &= A_T^{dn} + A_T^c \ , \\
A_P &= -A_W - A_E - A_S - A_N - A_B - A_T + \frac{\rho_P J_P}{\Delta t} + \rho_P J_P C_{\epsilon 2}\frac{\epsilon^t}{k^t} \ , \\
S\epsilon_T &= \frac{\rho_P J_P \epsilon_P^t}{\Delta t}, \\
S\epsilon_V &= 0 \ , \\
S\epsilon_F &= -I\epsilon_e^{dc} + I\epsilon_w^{dc} - I\epsilon_n^{dc} + I\epsilon_s^{dc} - I\epsilon_t^{dc} + I\epsilon_b^{dc} \ , \\
S\epsilon_C &= 0 \ , \\
S\epsilon_P &= C_{\epsilon 1}\frac{\epsilon}{k} S k_P \ .
\end{aligned}$$

B Point localization

B.1 Introduction

After having computed a three-dimensional flow field, a technique is necessary to reduce the flow field information in order to comprehend the information. Several techniques exist for this purpose.

One can look at slices of the solution for constant values of one of the three computational indices, or one can look at profiles along lines with two constant computational indices.

In the general case, computational planes will be general curved surfaces, and the lines will be curved, making the interpretation of the information more complex. Often the position of the measurements do not coincide with either mesh planes or lines, making the comparison of computed and measured values impossible.

One solution to this problem is an interpolation tool able to provide values between the computed values. In the present case this implies a tool capable of localizing a point in a block structured mesh composed of general skewed hexahedrons, and performing interpolation in a single hexahedron.

Such a interpolation tool can be used to provide profiles along arbitrary curves, or alternatively tracking particles in order to visualize complex flow fields. The particle tracking possibility of the method was used in the present work in connection with visualizing the flow around a surface mounted cube.

In the following the development of a localization and interpolation method working in physical space will be described.

B.2 Point localization method

The general computational cell is a hexahedron with nonplanar cell faces, implying that the face cannot be represented uniquely by a single normal vector.

Inspired by the volume calculation for the hexahedron, we will decompose the hexahedron into six non-overlapping tetrahedrons. These six tetrahedrons have planar faces, and can be represented uniquely by a single face normal. Now we just have to investigate if the coordinate triple (x_p, y_p, z_p) of the point in question is located within one of these six tetrahedrons, and if this is the case interpolate within the actual tetrahedron.

Point in tetrahedron

To investigate if the coordinate triple is located within a given tetrahedron, the following approach will be used. For each of the faces of the tetrahedron we calculate the dot product of the face normal pointing into the tetrahedron and the vector connecting one of the three face corner points with the point in question. If the result is positive, the point is located on the positive side of the plane defined by the direction of the face normal. Repeating this procedure for all four faces of the tetrahedron, we know that the point is located within the tetrahedron if all four dot products are positive, and the point is outside the tetrahedron if one or more of the dot products are negative.

B.3 Interpolation tool

Taking a close look at the way the point is localized within the tetrahedron, we will see that in fact we compute the volumes of the four subtetrahedrons when

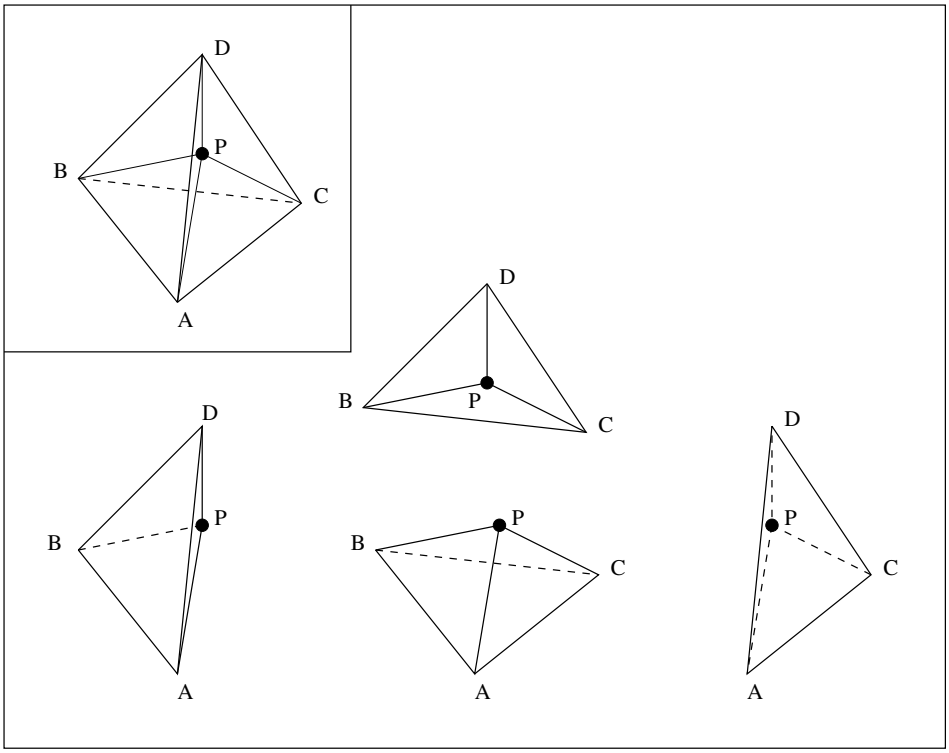

Figure 75. Subtetrahedras used when checking if the point is within the present tetrahedron.

performing the investigation.

These volumes can be used as weights when interpolating to point P. Looking at Fig. 75, we get the following expression for the value at point P

$$\varphi_P = \varphi_A \mathrm{Vol}_{BCDP} + \varphi_B \mathrm{Vol}_{ACDP} + \varphi_C \mathrm{Vol}_{ABDP} + \varphi_D \mathrm{Vol}_{ABCP} \ .$$

This means that no additional work is necessary to calculate the interpolation weights as these are already at hand when the point is located.

B.4 Profiles and particle traces

The general point localization and interpolation method can easily be combined with other routines in order to obtain profiles along lines and curves or to track particles transported by the field for both transient and steady-state solutions.

B.5 Program listing

The actual subroutines used for point localization and interpolation are listed below.

```
      subroutine incell(cp,c1,c2,c3,c4,c5,c6,c7,c8
     &                 ,tindex,weight,in_cell)
c-----------------------------------------------------------------------
c     Routine for checking if point (cp(1),cp(2),cp(3)) is located   c
c     within the cell bounded by the eight vertices 'c1' to 'c8'.    c
c     The vertices are given in accordance with the following figure.c
c                                                                     c
```

```
c               c7--------c8                                          c
c              /|        /|                                          c
c             / |       / |                                          c
c            /  |      /  |                                          c
c           c5--------c6  |                                          c
c           |   c3----|-- c4                                         c
c           |  /      |  /                                           c
c         k ^ /j      | /                                            c
c           |/        |/                                             c
c           c1->------c2                                             c
c               i                                                    c
c                                                                    c
c     or                                                             c
c                                                                    c
c     corner 1 = bsw                                                 c
c     corner 2 = bse                                                 c
c     corner 3 = bnw                                                 c
c     corner 4 = bne                                                 c
c     corner 5 = tsw                                                 c
c     corner 6 = tse                                                 c
c     corner 7 = tnw                                                 c
c     corner 8 = tne                                                 c
c                                                                    c
c      If the point is located within a cell, the logical variable   c
c    'in_cell' is set 'true', and the tetrahedra indices is          c
c    returned in 'tindex', and the necessary wieghts are returned    c
c    in 'weight'. These two arrays can then be used together with    c
c    the cell indices 'ip', 'jp', 'kp', and 'np' to performe the     c
c    interpolation.                                                  c
c      The interpolation is performed in the following way           c
c                                                                    c
c      var=phi(ip+tindex(1,1),jp+tindex(1,2),kp+tindex(1,3))         c
c     &   *weight(1)                                                 c
c     &   +phi(ip+tindex(2,1),jp+tindex(2,2),kp+tindex(2,3))         c
c     &   *weight(2)                                                 c
c     &   +phi(ip+tindex(3,1),jp+tindex(3,2),kp+tindex(3,3))         c
c     &   *weight(3)                                                 c
c     &   +phi(ip+tindex(4,1),jp+tindex(4,2),kp+tindex(4,3))         c
c     &   *weight(4)                                                 c
c                                                                    c
c--------------------------------------------------------------------
c    variables                                                       c
c    cp(1) .. cp(3)    : coordinats for point in question            c
c    c1(1) .. c1(3)    : coordinats of the cell vertices             c
c    c2(1) .. c2(3)    :         -- || --                            c
c    c3(1) .. c3(3)    :         -- || --                            c
c    c4(1) .. c4(3)    :         -- || --                            c
c    c5(1) .. c5(3)    :         -- || --                            c
c    c6(1) .. c6(3)    :         -- || --                            c
c    c7(1) .. c7(3)    :         -- || --                            c
c    c8(1) .. c8(3)    :         -- || --                            c
c    tindex             : tetrahedron indices                       c
c    weight             : the four weights to be used in            c
c                         interpolation                              c
```

```
c     in_cell           : logical variable, 'true' if point is found c
c                         within the cell                           c
c                                                                   c
c     Author            : Niels N. S\{o}rensen                     c
c     Last revision     :                                         c
c-------------------------------------------------------------------c
      implicit none
      real*8 cp(3),c1(3),c2(3),c3(3),c4(3),c5(3),c6(3),c7(3),c8(3)
      integer tindex(4,3)
      real*8 weight(4)
      logical in_cell
      logical in_tetra

c-----initialize in_cell to false
      in_cell=.false.

c-----check if the point is within any of the six tetrahedras forming
c-----the cube
      tindex(1,1)=0
      tindex(1,2)=1
      tindex(1,3)=0
      call tetrahedra(cp,c3,c1,c2,c6,weight,in_tetra)
      if(in_tetra)then
        tindex(2,1)=0
        tindex(2,2)=0
        tindex(2,3)=0
        tindex(3,1)=1
        tindex(3,2)=0
        tindex(3,3)=0
        tindex(4,1)=1
        tindex(4,2)=0
        tindex(4,3)=1
        in_cell=.true.
        return
      endif
      call tetrahedra(cp,c3,c1,c6,c5,weight,in_tetra)
      if(in_tetra)then
        tindex(2,1)=0
        tindex(2,2)=0
        tindex(2,3)=0
        tindex(3,1)=1
        tindex(3,2)=0
        tindex(3,3)=1
        tindex(4,1)=0
        tindex(4,2)=0
        tindex(4,3)=1
        in_cell=.true.
        return
      endif
      call tetrahedra(cp,c3,c5,c6,c7,weight,in_tetra)
      if(in_tetra)then
        tindex(2,1)=0
        tindex(2,2)=0
        tindex(2,3)=1
```

```
      tindex(3,1)=1
      tindex(3,2)=0
      tindex(3,3)=1
      tindex(4,1)=0
      tindex(4,2)=1
      tindex(4,3)=1
      in_cell=.true.
      return
    endif
    call tetrahedra(cp,c3,c6,c8,c7,weight,in_tetra)
    if(in_tetra)then
      tindex(2,1)=1
      tindex(2,2)=0
      tindex(2,3)=1
      tindex(3,1)=1
      tindex(3,2)=1
      tindex(3,3)=1
      tindex(4,1)=0
      tindex(4,2)=1
      tindex(4,3)=1
      in_cell=.true.
      return
    endif
    call tetrahedra(cp,c3,c6,c4,c8,weight,in_tetra)
    if(in_tetra)then
      tindex(2,1)=1
      tindex(2,2)=0
      tindex(2,3)=1
      tindex(3,1)=1
      tindex(3,2)=1
      tindex(3,3)=0
      tindex(4,1)=1
      tindex(4,2)=1
      tindex(4,3)=1
      in_cell=.true.
      return
    endif
    call tetrahedra(cp,c3,c2,c4,c6,weight,in_tetra)
    if(in_tetra)then
      tindex(2,1)=1
      tindex(2,2)=0
      tindex(2,3)=0
      tindex(3,1)=1
      tindex(3,2)=1
      tindex(3,3)=0
      tindex(4,1)=1
      tindex(4,2)=0
      tindex(4,3)=1
      in_cell=.true.
      return
    endif
    return
    end
```

```
      subroutine tetrahedra(cp,c1,c2,c3,c4,weight,in_tetra)
c-----------------------------------------------------------------------c
c     Routine for checking if the point cp is within the tetrahedra  c
c     distended by vector c1-c2,c1-c3,c1-c4 (that forms a right hand c
c     system). In case the point is within the tetrahedra the        c
c     variable 'in_tetra' is set true, and the weights necessary to  c
c     performe interpolation in the tetrahedra is returned.          c
c                                                                    c
c     The algorithme used to check if the point is located within    c
c     the tetrahedron is based on the following. For each of the     c
c     four faces of the tetrahedron the vector product of the two    c
c     vectors distenting the face is calculated to give the normal   c
c     vector of the plane pointing into the tetrahedra. Then the     c
c     inner product between the normal vector and the vector         c
c     connecting the origin of the two face vectors with the point   c
c     in question is performed, if the result is positive for all    c
c     four faces the point is located within the cube, if one or     c
c     more of the inner products is negative the point is located    c
c     outside the tetrahedron.                                       c
c-----------------------------------------------------------------------c
c     variables                                                      c
c     cp(1) .. cp(3)    : coordinats for point in question           c
c     c1(1) .. c1(3)    : corrdinats of vertice of tetrahedra        c
c     c2(1) .. c2(3)    :        -- || --                            c
c     c3(1) .. c3(3)    :        -- || --                            c
c     c4(1) .. c4(3)    :        -- || --                            c
c     weight(1) to (4)  : weights for interplation                   c
c     in_tetra          : logical set true if point is in tetrahedra c
c-----------------------------------------------------------------------c
      implicit none
      real*8 cp(3),c1(3),c2(3),c3(3),c4(3)
      real*8 weight(4)
      logical in_tetra
      real*8 vp1(3),vp2(3),v1(3),v2(3),v3(3),v4(3),v5(3)
      real*8 vres(3),norm,res

c-----set in_tetra to false
      in_tetra=.false.
c-----vector from point 1 to point p
      vp1(1)=cp(1)-c1(1)
      vp1(2)=cp(2)-c1(2)
      vp1(3)=cp(3)-c1(3)
c-----vector from point 3 to point p
      vp2(1)=cp(1)-c3(1)
      vp2(2)=cp(2)-c3(2)
      vp2(3)=cp(3)-c3(3)
c-----calculate the three vectors distenting the tetrahedra
      v1(1)=c2(1)-c1(1)
      v1(2)=c2(2)-c1(2)
      v1(3)=c2(3)-c1(3)
      v2(1)=c3(1)-c1(1)
      v2(2)=c3(2)-c1(2)
      v2(3)=c3(3)-c1(3)
      v3(1)=c4(1)-c1(1)
```

```
      v3(2)=c4(2)-c1(2)
      v3(3)=c4(3)-c1(3)
c-----calculate the two vectors distenting face four
      v4(1)=c2(1)-c3(1)
      v4(2)=c2(2)-c3(2)
      v4(3)=c2(3)-c3(3)
      v5(1)=c4(1)-c3(1)
      v5(2)=c4(2)-c3(2)
      v5(3)=c4(3)-c3(3)

c-----check if point is within tetrahedra
c-----face 1 (distended by v1 and v2)
      call cross_product(v1,v2,vres)
      call dot_product(vres,vp1,res)
      weight(4)=res
c-----face 2 (distended by v3 and v1)
      call cross_product(v3,v1,vres)
      call dot_product(vres,vp1,res)
      weight(3)=res
c-----face 3 (distended by v2 and v3)
      call cross_product(v2,v3,vres)
      call dot_product(vres,vp1,res)
      weight(2)=res
c-----face 4 (distended by v4 and v5)
      call cross_product(v4,v5,vres)
      call dot_product(vres,vp2,res)
      weight(1)=res
c-----check if point is within tetrahedra and in case this is true
c-----set 'in_tetra' to 'true'
      if(weight(1).ge.0.d0.and.weight(2).ge.0.d0.and
     &  .weight(3).ge.0.d0.and.weight(4).ge.0.d0)then
         in_tetra=.true.
c--------normalize weights
         norm=weight(1)+weight(2)+weight(3)+weight(4)
         weight(1)=weight(1)/norm
         weight(2)=weight(2)/norm
         weight(3)=weight(3)/norm
         weight(4)=weight(4)/norm
      endif
      return
      end

      subroutine cross_product(v1,v2,vres)
c-----------------------------------------------------------------------c
c     Routine calculating the cross product of vector 'v1' and      c
c     'v2', and returning the result in vector 'vres'.              c
c                                                                   c
c-----------------------------------------------------------------------c
c     variables                                                     c
c     v1        : vector 1.                                         c
c     v2        : vector 2.                                         c
c     vres      : result of cross product (a vector)                c
c     Author            : Niels N. S\{o}rensen                      c
c     Last revision     : 9 oct. 94                                 c
```

```
c--------------------------------------------------------------------c
      implicit none
      real*8 v1(3),v2(3),vres(3)

c-----cross product
      vres(1)=v1(2)*v2(3)-v1(3)*v2(2)
      vres(2)=v1(3)*v2(1)-v1(1)*v2(3)
      vres(3)=v1(1)*v2(2)-v1(2)*v2(1)
      return
      end


      subroutine dot_product(v1,v2,res)
c--------------------------------------------------------------------c
c     Routine calculating the dot product of vector 'v1' and 'v2'    c
c     'v2', and returning the result in variable 'res'.              c
c--------------------------------------------------------------------c
c     variables                                                      c
c     v1        : vector 1.                                          c
c     v2        : vector 2.                                          c
c     res       : result of dot product                              c
c                                                                    c
c     Author            : Niels N. S\{o}rensen                       c
c     Last revision     : 9 oct. 94                                  c
c--------------------------------------------------------------------c
      implicit none
      real*8 v1(3),v2(3),res

c-----dot product
      res=v1(1)*v2(1)+v1(2)*v2(2)+v1(3)*v2(3)
      return
      end
```

Bibliographic Data Sheet **Risø–R–827(EN)**

Title and author(s)

General purpose flow solver applied to flow over hills

Niels N. Sørensen

Dept. or group

Meteorology and Wind Energy

Date

Groups own reg. number(s) Project/contract No.

Pages	Tables	Illustrations	References
154	13	75	66

Abstract (Max. 2000 char.)

The present report describes the development a 2D and a 3D finite-volume code in general curvilinear coordinates using the Basis2D/3D platform by Michelsen. The codes are based on the Reynolds averaged incompressible isothermal Navier-Stokes equations and use primitive variables (U, V, W and P). The turbulence is modelled by the high Reynolds number $k-\epsilon$ model.

Cartesian velocity components are used in a non-staggered arrangement following the methodology of Rhie. The equation system is solved using the SIMPLE method of Patankar and Spalding. Solution of the transport equations is obtained by a successive application of a TDMA solver in alternating direction. The solution of the pressure correction equation is accelerated using the multigrid tools from the Basis2D/3D platform. Additionally a three-level grid sequence is implemented in order to minimize the overall solution time.

Higher-order schemes (SUDS and QUICK) are implemented as explicit corrections to a first-order upwind difference scheme.

In both the 2D and the 3D code it is possible to handle multiblock configurations. This feature is added in order to obtain a greater geometric flexibility.

To mesh natural terrain in connection with atmospheric flow over complex terrain, a two- and a three-dimensional hyperbolic mesh generator are constructed. Additionally, a two- and a three-dimensional mesh generator based on a simple version of the transfinite interpolation technique are implemented.

Several two-dimensional test cases are calculated e.g. laminar flow over a circular cylinder, turbulent channel flow, and turbulent flow over a backward facing step, all with satisfying results. In order to illustrate the application of the codes to atmospheric flow two cases are calculated, flow over a cube in a thick turbulent boundary-layer, and the atmospheric flow over the Askervein hill.

Descriptors INIS/EDB

BOUNDARY LAYERS; COMPLEX TERRAIN; COMPUTERIZED SIMULATION; CURVILINEAR COORDINATES; FLOW MODELS; FLUID FLOW; LAMINAR FLOW; NAVIER-STOKES EQUATIONS; THREE-DIMENSIONAL CALCULATIONS; TURBULENCE; TURBULENT FLOW; TWO-DIMENSIONAL CALCULATIONS

Available on request from:
Information Service Department, Risø National Laboratory
P.O. Box 49, DK–4000 Roskilde, Denmark
Phone (+45) 46 77 46 77, ext. 4004/4005 · Telex 43 116 · Fax (+45) 46 75 56 27