Experiments with the improved finite volume discretization of the incompressible Navier-Stokes equations in non-smooth boundary-fitted coordinates in two dimensions

G. Segal, K. Kassels, P. Wesseling

ISSN 0922-5641
Reports of the Faculty of Technical Mathematics and Informatics no. 96-107
Delft 1996
1 Introduction

In Segal et al (1992) [11] the numerical solution of the incompressible Navier-Stokes equations in complex geometries has been studied. Starting point of the discretization is a region consisting of subregions, each of which may be mapped onto a rectangular domain. In [11] we have limited ourselves to a so-called one block decomposition, implying that the region is mapped onto one rectangular domain. In [15], [14], [16] and [13] an improvement of the discretization with special emphasis to non-smooth mappings has been proposed. The extension to multi-block has been the subject of a number of papers, see [1], [4], [2] and [3].

In this report we apply this improved discretization with some minor modifications to some specific examples. These examples show that the performance of the "non-smooth" scheme is considerably better than that of the classical scheme derived in [11], even for relatively simple examples.

An important aspect of both approaches is that all computations take place in the computational domain, which is the rectangle onto which the region is mapped. We use a structured grid, hence the grid in the physical domain is mapped onto a rectangular grid in the computational domain. It is assumed that this mapping is generated numerically, giving the mapping in a discrete set of points. The mapping is extended to all of the domain by bilinear interpolation.

Because all discretizations are carried out in computational space, it is necessary to transform the differential equations to this space. Such a transformation may be easily carried out by means of tensor analysis. This makes the differential equations more complicated, however, the discretization is relatively simple.

In [11] it has been shown that, in order to get an acceptable discretization, it is necessary to use mass fluxes as unknowns. The reason is, that only in that case, the transformation of a constant vector field from physical space to computational domain and backwards, results in a constant vector field. The discretizations derived in [11], have proven to be able to solve a number of practical problems, provided the grid is relatively smooth. However, in practice, it is very hard to create a smooth grid and one is often faced with discontinuities in the mapping from physical domain to computational domain. For example a sudden jump in the grid size may result in very inaccurate pressures, even for a simple 2D channel flow. For this reason the improved discretization described here, has been developed.

Other publications describing discretization on staggered grids in boundary
fitted coordinates are [12], [8], [6], [10], [5], [9] and [17].
In the next section we recall the discretization used in [11]. Thereafter we
will derive the improved discretization.

2 Discretization of the incompressible Navier-Stokes equations on a curvilinear grid.

In [11] we have used the Navier-Stokes equations in tensor formulation to
derive our discretization:

\[ U_\alpha^\alpha = 0, \quad (2.1) \]

\[ \frac{\partial}{\partial t} (\rho U_\alpha) + (\rho U_\alpha U_\beta)_\beta + (g^{\alpha\beta} p)_\beta - \tau^{\alpha\beta}_\beta = \rho f_\alpha, \quad (2.2) \]

where \( \tau^{\alpha\beta} \) represents the deviatoric stress tensor

\[ \tau^{\alpha\beta} = \mu (g^{\alpha\gamma} U_\gamma^\beta + g^{\beta\gamma} U_\gamma^\alpha), \quad (2.3) \]

with \( \mu \) the (laminar or turbulent) viscosity, \( p \) the pressure, \( U_\alpha \) the con-
travariant velocity component and \( \rho \) the density of the fluid.

Equations (2.1) to (2.3) must be solved on a curvilinear grid which is mapped
onto a rectangular grid. Figure 2.1 shows a typical example of such a map-
ing. The transformation \( T \) is defined by the relation \( x = x(\xi) \). The

covariant base vectors \( a_\alpha(\xi) \) are defined as tangent vectors to the coordinate
lines \( \xi^\alpha = \) constant, \( \beta \neq \alpha \)

\[ a_\alpha(\xi) = \frac{\partial x}{\partial \xi^\alpha}, \quad (2.4) \]

and the contravariant base vectors \( a^\alpha \) as

\[ a^\alpha(\xi) = \nabla \xi^\alpha = \left( \frac{\partial \xi^\alpha}{\partial x^1}, \frac{\partial \xi^\alpha}{\partial x^2} \right)_T, \quad (2.5) \]

so that \( a^\alpha(\xi) \) is normal to the coordinate lines \( \xi^\alpha = \) constant.

The covariant and contravariant metric tensors \( g_{\alpha\beta} \) and \( g^{\alpha\beta} \) are defined by

\[ g_{\alpha\beta} = a_\alpha(\xi) \cdot a_\beta(\xi); \quad g^{\alpha\beta} = a^\alpha(\xi) \cdot a^\beta(\xi). \quad (2.6) \]

Let \( g = \text{det}(g_{\alpha\beta}) \), then \( \sqrt{g} \) equals the Jacobian of the transformation, \( J = \text{det}(\frac{\partial x}{\partial \xi}) \).

If we define \( a \otimes b \) as

\[ a \otimes b = a_1 b_2 - a_2 b_1, \quad (2.7) \]
(hence \( \mathbf{a} \otimes \mathbf{b} \) is the signed length of \( \mathbf{a} \times \mathbf{b} \)), then \( \sqrt{g} \) can be written as

\[
\sqrt{g} = \mathbf{a}_{(1)} \otimes \mathbf{a}_{(2)}.
\]  

(2.8)

The covariant derivative is a tensor which reduces to a partial derivative of a vector field in Cartesian coordinates. For an absolute scalar, the covariant derivative is the same as the partial derivative, and is denoted by

\[
\mu_{,\alpha} = \frac{\partial \mu}{\partial \xi^\alpha}.
\]  

(2.9)

The covariant derivative of a contravariant tensor of rank one \( U^\alpha \) is given by

\[
U_{\beta}^{\alpha} = \frac{\partial U^\alpha}{\partial \xi^\beta} + \{ \alpha_{\gamma \beta} \} U^\gamma,
\]  

(2.10)

where \( \{ \alpha_{\gamma \beta} \} \) is the so-called Christoffel symbol of the second kind, defined by

\[
\{ \alpha_{\gamma \beta} \} = \mathbf{a}^{(\alpha)} \cdot \frac{\partial \mathbf{a}^{(\gamma)}}{\partial \xi^\beta} = \frac{\partial x^\alpha}{\partial x^\gamma} \frac{\partial^2 x^\beta}{\partial x^\gamma \partial x^\delta} = \frac{1}{2} \delta^{\alpha}_{\beta} (\frac{\partial g_{\gamma \eta}}{\partial \xi^\delta} + \frac{\partial g_{\delta \eta}}{\partial \xi^\gamma} - \frac{\partial g_{\delta \gamma}}{\partial \xi^\eta}).
\]  

(2.11)
The contravariant velocity components and the Cartesian components are related by
\[ u = U^\alpha a_{(\alpha)}, \quad U^\alpha = u \cdot a^{(\alpha)}. \] (2.12)
As discussed in [11], it is necessary to use the mass flux \( V^\alpha = \sqrt{g} U^\alpha \) through the coordinate surface \( \xi^\alpha = \text{constant} \), as primary unknowns.

In the case of Cartesian coordinates \( (\xi = x) \) our discretization is identical to the classical staggered discretization of Harlow and Welch [7]. The fluxes are defined at midside points and the scalar quantities at the centres of cells. The geometrical quantities (base vectors, metric tensors and Christoffel symbols), are defined by central differences. The mapping \( x = x(\xi) \) is given in the cell vertices, and extended to the whole domain by bilinear interpolation.

Hence \( a_{(1)} \) is continuous along the lines \( \xi^2 \) constant and \( a_{(2)} \) is continuous along the lines \( \xi^1 \) constant. At the two other sides of the cells these values are discontinuous. One easily verifies that due to the continuity of \( a_{(1)} \), \( a^{(1)} \) is continuous along the lines \( \xi^1 \) constant and \( a^{(2)} \) along the lines \( \xi^2 \) constant.

The Jacobian \( \sqrt{g} \) is only continuous inside the cells.

Before considering the improved discretization we first focus our attention on the computation of the base vectors and \( \sqrt{g} \).

### 3 Computation of the base vectors and interpolation of velocity components

One of the key issues in the curvilinear approach is the computation of the geometrical quantities and the interpolation of unknowns to points where they are not defined. There are several ways to generate consistent approximations of the geometrical quantities, but we impose the extra requirement that, if the solution of the Navier-Stokes equations consists of a constant velocity field with corresponding constant pressure, we want the numerical solution to be exact, regardless of the grid.

Although this demand formally does not increase the order of accuracy of the approximation, experiments show that if we satisfy these requirements, the numerical solution is much more acceptable in case of non-smooth grids. As pointed out in Section 2, \( a_{(1)} \) is continuous along the lines \( \xi^2 \) constant and \( a_{(2)} \) is continuous along the lines \( \xi^1 \) constant. At the two other sides of the cells these quantities are discontinuous and it is necessary to define them in a suitable way. For the same reason \( \sqrt{g} \) is not continuous across cell boundaries. Hence also for the Jacobian a definition at the cell boundaries is needed. We shall investigate how we have to define the discontinuous
quantities in such a way, that a constant velocity field is represented exactly, even for a staggered grid. For the sake of argument we limit ourselves to the centre of \( V^1 \)-cell. All other points of interest can be treated in exactly the same way.

Consider two adjacent cells in the computational space in Figure 3.1. Because

\[
\begin{array}{c}
(-1,1) \\
(0,0) \\
(-1,-1) \\
(1,-1)
\end{array}
\]

Figure 3.1: Two adjacent \( V^1 \)-cells in computational domain, with local numbering.

the mapping is piecewise bilinear, along the cell face with centre at \((0,0)\), \( a_{(2)} \) is given exactly by \( a_{(2)} = x_{(0,1)} - x_{(0,-1)} \) and is continuous at this cell face. Similarly, the base vectors \( a_{(1)} \) are continuous at the horizontal cell faces with centres \((\pm 1, \pm 1)\). The Cartesian velocity \( u \) is of course continuous over the complete domain. It follows that the flux \( V^1 \) defined by

\[
V^1 = u \otimes a_{(2)} = a_{(2)}^1 u_1 - a_{(2)}^2 u_2 \quad (3.1)
\]

is continuous in the vertical cell faces, whereas

\[
V^2 = u \otimes a_{(1)} = a_{(1)}^1 u_1 - a_{(1)}^2 u_2 \quad (3.2)
\]

is continuous at the horizontal cell faces.

Once the contravariant velocity vector is known in each point, one can compute the Cartesian velocity vector. In order that a constant vector field is represented exactly it is necessary that the transformation from Cartesian components, to contravariant components and again the transformation
from contravariant components to Cartesian components produces the original Cartesian vector at least for constant velocity fields. This might seem trivial, however, since only one component of the flux is continuous in the grid point, the other one must be derived by interpolation. Hence the interpolation must satisfy certain rules.

First of all we have to define the base vector $a_{(1)}$ in the point $(0,0)$. A simple definition is the trivial interpolation

$$a_{(1)}|_{(0,0)} = \frac{1}{4}(a_{(1)}|_{(1,1)} + a_{(1)}|_{(-1,1)} + a_{(1)}|_{(-1,-1)} + a_{(1)}|_{(1,-1)}). \quad (3.3)$$

Note that (3.3) is identical to

$$a_{(1)}|_{(0,0)} = x_{(1,0)} - x_{(-1,0)}. \quad (3.4)$$

Since $V^2$ is defined by

$$V^2 = u \otimes a_{(1)}, \quad (3.5)$$

it is clear that $V^2$ in $(0,0)$ must be interpolated in exactly the same way as $a_{(1)}$. Only then the relation (3.5) holds in $(0,0)$ for constant vector fields.

The Cartesian velocity in $(0,0)$ follows from:

$$u = \frac{a_{(1)}V^1 + a_{(2)}V^2}{\sqrt{g}} \quad (3.6)$$

Substitution of the relations (3.1) and (3.5) into (3.6) yields

$$u = \frac{a_{(1)}(u \otimes a_{(2)}) + a_{(2)}(u \otimes a_{(1)})}{\sqrt{g}} \quad (3.7)$$

Left-hand side and right-hand side of (3.7) are identical provided $\sqrt{g}|_{(0,0)}$ is defined by

$$\sqrt{g} = a_{(1)} \otimes a_{(2)}, \quad (3.8)$$

which is the standard definition in case of a differentiable transformation. Hence in order that a constant vector field is represented exactly in computational domain it is necessary that the following two requirements are met:

- The contravariant components and its corresponding covariant base vectors must be interpolated in exactly the same way.
- The determinant of the transformation $\sqrt{g}$ must in each point of the grid be defined such that relation (2.8) holds.
It is a simple exercise to show that the area of a grid cell is exactly equal to the numerical value of $\sqrt{g}$ in the centre of the cell, provided the definitions in this section are applied. However, the value of $\sqrt{g}$ in the other points is not equal to the area of the corresponding finite volume. Consider for example the area of a $V^1$-cell. One can show, that this area $\Omega(0,0)$ is identical to
\[
\Omega(0,0) = \frac{1}{2} \sqrt{g}(0,0) + \frac{1}{4} (\sqrt{g}(-1,0) + \sqrt{g}(1,0))
\] (3.9)
where $\sqrt{g}$ is computed as described in this section. It does not seem easy to define an simple interpolation of $a(2)$ in the point $(0,0)$ such that $\sqrt{g}(0,0)$ is equal to $\Omega(0,0)$ and also a constant vector field is represented exactly. In the next section we consider the improved discretization.

4 Improved discretization of the incompressible Navier-Stokes equations

The idea of the improved discretization is that we try to use quantities only in those points where they are continuous. In that way it is possible to overcome trouble due to non-smooth grids. In all other points it is necessary to use such an approximation that constant and if possible even linear fields are represented exactly regardless of the grid used.

A major difference between the discretization derived in [11] and our improved discretization is that we do not start with the general tensor formulation, but use the momentum equations in Cartesian form:
\[
\frac{\partial \rho u^\alpha}{\partial t} + (\rho u^\alpha u^\beta)_{,\beta} + (\delta^\alpha_\beta p)_{,\beta} - \tau^\alpha_\beta = \rho f^\alpha
\] (4.1)

Application of the relation
\[
\frac{\partial \sqrt{g} a^\alpha}{\partial \xi^\alpha} = 0,
\] (4.2)
and the chain rule implies that
\[
\frac{\partial \phi}{\partial x^\beta} = \frac{1}{\sqrt{g}} \frac{\partial \sqrt{g} a^{(\gamma)}_\beta}{\partial \xi^\gamma} \phi
\] (4.3)

When lowering the contravariant index $\alpha$ and (4.3) is taken into account, while retaining the pressure gradient and the stress tensor as they stand, the momentum equations (4.1) can be written in the following form:
\[
\frac{\partial \rho u_\alpha}{\partial t} + \frac{1}{\sqrt{g}} \frac{\partial \rho V^\tau u_\alpha}{\partial \xi^\gamma} + \frac{\partial p}{\partial x^\alpha} - \frac{1}{\sqrt{g}} \frac{\partial \sqrt{g} a^{(\gamma)}_\beta}{\partial \xi^\gamma} \tau^\beta_\alpha = \rho f_\alpha,
\] (4.4)
with
\[ \tau^\beta_\alpha = \mu \left( \frac{\partial u^\beta}{\partial x^\alpha} + \frac{\partial u^\alpha}{\partial x^\beta} \right), \]

(4.5)
or in vector form
\[ \frac{\partial u}{\partial t} + \frac{1}{\sqrt{g}} \frac{\partial P}{\partial \xi} + \nabla p - \frac{1}{\sqrt{g}} \frac{\partial \sqrt{g} u^\gamma}{\partial \xi^\gamma} \tau_\beta^\beta = \rho f, \]

(4.6)
where
\[ \tau^\beta = (\tau_1^\beta, \tau_2^\beta)^T. \]

(4.7)
In order to apply the finite volume method the momentum equations are integrated over the \( V^1 \) and \( V^2 \) cells and the continuity equations over the \( p \)-cells. Figure 4.1 shows a \( V^1 \)-cell with local numbering in the computational domain.

![Figure 4.1: \( V^1 \)-cell in computational domain, with local numbering](image)

With respect to the discretization we consider all terms separately.

Time-derivative:
\[ \int_\Omega \frac{\partial u}{\partial t} d\Omega \approx \Omega_{(0,0)} \frac{\partial u}{\partial t}_{(0,0)}. \]

(4.8)
Note that the derivative at the right-hand side is a physical quantity and hence continuous.
Right-hand side:

\[ \int_{\Omega} \rho f \, d\Omega \approx \Omega_{(0,0)} \rho f|_{(0,0)}. \]  

(4.9)

Convective terms:

\[ \int_{\Omega} \frac{1}{\sqrt{g}} \frac{\partial \rho V^\gamma u}{\partial \xi^\gamma} \, d\Omega = \int_{\Omega_{\xi}} \frac{1}{\rho V^1} u_{\xi^1=1} \, d\xi^2 + \int_{\Omega_{\xi}} \frac{1}{\rho V^2} u_{\xi^2=1} \, d\xi^1, \]

(4.10)

where \( \Omega_{\xi} \) denotes the volume in the computational domain. Application of the midpoint rule to the integrals at the right-hand side of (4.10) results in the term:

\[ \rho V^1 u|_{(-1,0)}^{(1,0)} + \rho V^2 u|_{(0,-1)}^{(0,1)}. \]

(4.11)

The components of \( u \) can be expressed in terms of the unknown fluxes by the relation (3.6). Fluxes must be interpolated according to the rules of Section 3.

Pressure gradient:

\[ \int_{\Omega} \nabla p \approx \Omega_{(0,0)} \nabla p|_{(0,0)}. \]

(4.12)

In order to approximate \( \nabla p|_{(0,0)} \) in such a way that the result is exact for constant \( \nabla p \) on arbitrary non-uniform grids, we follow the so-called integration path method, introduced by Wesseling et al [16], also described in [13]. The idea is quite simple. We choose two integration paths in physical space as depicted in Figure 4.2. With respect to the \( \xi^1 \)-direction the trivial path (-1,0), (0,0), (1,0) is chosen. In the \( \xi^2 \)-direction the points ((0,-1) and (0,1) do not contain pressure unknowns. Therefore we use both the path (-1,-2), (-1,0) (0,0), (1,0), (1,2) and the path (1,-2), (1,0), (0,0), (-1,0) and (-1,2).

First consider the path in the \( \xi^1 \)-direction. Integrating of \( \nabla p \) along this path gives

\[ p|_{(-1,0)}^{(1,0)} = \int_{(-1,0)}^{(1,0)} \nabla p \cdot dx \approx \nabla p|_{(0,0)} \cdot \int_{(-1,0)}^{(1,0)} dx = \nabla p|_{(0,0)} \cdot c_{(1)}, \]

(4.13)

with

\[ c_{(1)} = \int_{(-1,0)}^{(1,0)} dx = x_{(1,0)} - x_{(-1,0)} = a_{(1)}|_{(0,0)}, \]

(4.14)
with $\mathbf{a}_{(1)}(0,0)$ given by (3.4).

Next consider the paths in the $\xi^2$-direction. Integrating of $\nabla p$ along these paths gives

$$p|_{(1,-2)} + p|_{(-1,-2)} = \int_{(1,-2)}^{(-1,2)} \int_{(-1,-2)}^{(1,2)} \nabla p \cdot d\mathbf{x} \approx \nabla p|_{(0,0)} \cdot \mathbf{c}(2), \quad (4.15)$$

with

$$\mathbf{c}(2) = \mathbf{x}_{(-1,2)} - \mathbf{x}_{(-1,-2)} + \mathbf{x}_{(1,2)} - \mathbf{x}_{(1,-2)}. \quad (4.16)$$

(4.13) and (4.15) can be considered as two equations to express $\nabla p$ in terms of the six surrounding pressure unknowns. If we define

$$c^{(1)} = \frac{1}{C}(c_{(2)} - c_{(1)}, -c_{(1)})^T; \mathbf{c}^{(2)} = \frac{1}{C}(-c_{(2)}, c_{(1)})^T; C = c_{(1)} \otimes c_{(2)}, \quad (4.17)$$

then it follows that

$$\nabla p|_{(0,0)} \approx p|_{(-1,0)} c^{(1)} + (p|_{(-1,-2)} + p|_{(1,-2)}) c^{(2)}. \quad (4.18)$$
Stress tensor:

\[
\int_{\Omega} \frac{1}{\sqrt{g}} \frac{\partial \sqrt{g} a_\beta^{(\gamma)}}{\partial \xi^\gamma} d\Omega = \int_{\Omega_{\xi}} \frac{\partial \sqrt{g} a_\beta^{(\gamma)}}{\partial \xi^\gamma} d\Omega_{\xi} \approx \sqrt{g} \sigma_\beta^{(1)} \tau_\beta^{(1)(0,1)} + \frac{1}{2} \left( \sqrt{g} a_\beta^{(2)} \tau_\beta^{(1,1)} + \sqrt{g} a_\beta^{(2)} \tau_\beta^{(-1,1)} \right).
\]

(4.19)

All terms in these expressions are continuous and well defined.

The remaining problem is of course the discretization of \( \tau_\beta \). This is done in the same way as the evaluation of \( \nabla p \), i.e. the integration path method is applied. Using the definition of \( \tau_\beta \) we get

\[
\tau_\beta^{(1)(0,1)} \approx \mu^{(1)(0,1)} \left( \frac{\partial a_\rho^{(1)}}{\partial x^\rho} + \frac{\partial a_\rho^{(2)}}{\partial x^\rho} \right) \left( \frac{\partial a_\rho^{(1)}}{\partial x^\rho} + \frac{\partial a_\rho^{(2)}}{\partial x^\rho} \right)_{(1,0)},
\]

(4.20)

and similar expressions for the other integration points. For some of the integration points it may be necessary to interpolate \( \mu \). At this stage we do not specify this interpolation explicitly. The derivatives of \( u_\alpha \) are evaluated by the integration path method. In the point \((-1,0)\) we have

\[
u\alpha^{(-1,1)}_{(-1,-1)} = \int_{(-1,-1)} \nabla u_\alpha \cdot dx \approx \nabla u_\alpha^{(-1,1)}_{(-1,0)} \cdot a_{(2)}^{(-1,0)}, \]

(4.21)

\[
u\alpha^{(0,0)}_{(-2,0)} = \int_{(-2,0)} \nabla u_\alpha \cdot dx \approx \nabla u_\alpha^{(0,0)}_{(-1,0)} \cdot a_{(1)}^{(-1,0)}. \]

(4.22)

Solution of these two equations with two unknowns gives

\[
\nabla u_\alpha^{(-1,0)} = a^{(1)}_{(-1,0)} u_\alpha^{(0,0)}_{(-2,0)} + a^{(2)}_{(-1,0)} u_\alpha^{(-1,1)}_{(-1,-1)}.
\]

(4.23)

The Cartesian velocity must be expressed in terms of the contravariant components using the relation (3.6).

In the point \((0,1)\) we have a similar expression.

\[
u\eta^{(1,1)}_{(-1,1)} = \int_{(-1,1)} \nabla u_\alpha \cdot dx \approx \nabla u_\alpha^{(1,1)}_{(0,1)} \cdot c_{(1)}^{(0,1)}, \]

(4.24)

\[
u\alpha^{(0,0)}_{(0,2)} = \int_{(0,2)} \nabla u_\alpha \cdot dx \approx \nabla u_\alpha^{(0,0)}_{(0,1)} \cdot c_{(2)}^{(0,1)}. \]

(4.25)
with \( c_{(1)}|_{(0,1)} = x|_{(-1,1)}^{(1,1)} \) and \( c_{(2)}|_{(0,1)} = x|_{(0,0)}^{(0,2)} \). Solution of these equations gives

\[
\nabla u^\alpha|_{(0,1)} = c^{(1)}|_{(0,1)} u^\alpha|_{(-1,1)}^{(1,1)} + c^{(2)}|_{(0,1)} u^\alpha|_{(0,0)}^{(0,2)},
\]

(4.26)

where \( c^{(1)} \) and \( c^{(2)} \) follow from \( c_{(1)} \) and \( c_{(2)} \) by (4.17).

We have now computed the spatial discretization of (4.1), and obtained a hybrid formulation containing both \( u \) and \( V^\alpha \). In order to transform to an invariant (tensor) formulation by taking the inner product of the discretized vector equations and \( a^{(1)}|_{(0,0)} \) for a \( V^1 \)-cell. For a \( V^2 \)-cell we use of course \( a^{(2)}|_{(0,0)} \). Since \( \Omega_{(0,0)} \) is not equal to \( \sqrt{f}|_{(0,0)} \) and we would like to have an equation for \( \frac{dV^\alpha}{dt}(0,0) \) the equations are premultiplied by \( \frac{\sqrt{f}|_{(0,0)}}{\Omega_{(0,0)}} \).

The resulting stencil is a 21-point stencil for each \( V^\alpha \)-unknown. Figure 4.3 shows the stencil for a \( V^1 \) unknown.

![Stencel for a V1-cell](image)

Figure 4.3: Stencil for a \( V^1 \)-cell

5 Numerical experiments

One of the main problems with the discretization in the computational space is that the transformation is not known as an analytical function. Since the transformation is completely determined by the coordinates of the grid

13
points, a sudden jump in the grid spacing produces a discontinuous transformation. The geometrical quantities used in the transformed Navier-Stokes equations (2.1) to (2.3), are first derivatives of the transformation \( a^{(a)} \), \( a^{(b)} \) and \( g^{ab} \), or even second derivatives (Christoffel symbols). So one may expect problems as soon as the grid spacing is a non-smooth function.

In order to investigate the sensitivity of the discretization with respect to the smoothness of the solution we consider a simple channel flow. Figure 5.1 shows the configuration of a straight channel.

![Figure 5.1: Flow through a straight channel](image)

At the inlet we prescribe a quadratic velocity profile, at the fixed boundaries the velocity is zero (no-slip condition) and at the outlet we prescribe the outflow condition: \( u_t = 0, \sigma_n = 0 \).

The exact solution is the well known Poiseuille flow with a quadratic velocity profile, constant in the flow direction and with a linear pressure distribution. In first instance this example has been run with the discretization as described in [11]. For an equidistant grid the solution looks perfect. However, if we make a local refinement of the mesh as shown in Figure 5.2 (1:2 jump), we see the effect of the jump. The velocity field is still quadratic (Figure 5.3), but the pressure field is not constant any more in the cross sections. Figure 5.4 shows a typical jump in the pressure, exactly at the place where the mesh is discontinuous.

By replacing the discretization of the pressure gradient by a finite difference based finite volume method, with the property that the derivative of a constant pressure is zero, we were able to improve the classical discretization considerably.

Figure 5.5 shows the contour plot of the computed pressure for the straight channel problem of Figures 5.2 - 5.4 using this adapted approach. Clearly the pressure is much more accurate than in the original case. If we enlarge the jump in the mesh from 1:2 to 1:3 also this new discretization gives inaccurate results as can be seen in Figure 5.6. Application of the discretization treated in Section 4 appears to be insensitive to the size of the jump. Even
Figure 5.2: Mesh with local refinement for straight channel problem

Figure 5.3: Vector plot of computed velocity for straight channel problem

Figure 5.4: Contour plot of computed pressure for straight channel problem

for a 1:100 jump the pressure contours where all straight and with constant distance, proving that a linear pressure field is computed. Hence the new discretization is superior to the classical one with respect to the treatment of jumps in a grid.

We do not only want to investigate the behaviour of the improved discretization with respect to a jump in the grid, but also with a very curved grid. For that reason we solve the simple channel flow with the curved grid shown in Figure 5.7. Such a grid is very well suited for discovering weaknesses of curvilinear codes. The quantity that is the most sensitive for errors is al-

Figure 5.5: Contour plot of pressure in the case of 1:2 jump

Figure 5.6: Contour plot of pressure in the case of 1:3 jump
ways the pressure. Figure 5.8 shows the computed isobars using the classical method of [11] with the adapted pressure gradient discretization. Note that although the grid is smooth, its influence on the solution is clearly visible. The corresponding stream lines are shown in Figure 5.9. The improved discretization of Section 4 produces a much better pressure field as shown in Figure 5.10. The corresponding stream lines are shown in Figure 5.11. We see that even in this case the isobars are not parallel as one would expect from the exact solution.

In order to check the quality of these results we have compared the computations with the results of a finite element code using bilinear velocity, constant pressure, quadrilateral elements on exactly the same grid. Figures 5.12 and 5.13 show the isobars and stream lines of the finite element code. It is clear that are finite volume discretization produces nicer isobars and that the quality of the stream lines is comparable.
We must remark that if we refine the grid by doubling the number of points in both directions, both the finite element method and the finite volume method produce nearly perfect pictures.

The last example we consider is the flow in a bend. The exact solution is not known. In this example the convection terms are not negligible anymore. Consider the grid shown in Figure 5.14. The isobars and stream lines of the classical discretization are plotted in Figures 5.15 and 5.16, whereas Figures 5.17 and 5.18 show the results of our improved discretization. The improvement of accuracy is very clear if we compare them with the solution shown in Figures 5.19 and 5.20 computed on a very fine grid.
Figure 5.14: Grid for bend problem

Figure 5.15: Isobars for the bend problem using the classical discretization

Figure 5.16: Stream lines for the bend problem using the classical discretization

Figure 5.17: Isobars for the bend problem using the improved discretization

Figure 5.18: Stream lines for the bend problem using the improved discretization
Figure 5.19: Isobars for the bend problem using a fine smooth grid

Figure 5.20: Stream lines for the bend problem using a fine grid
6 Conclusions

In this report we have compared the classical discretization of the incompressible Navier-Stokes equations as derived in [11], with an improved scheme. This scheme, is a slight modification of a scheme especially developed for non-smooth grids which has been introduced in [15], [14], [16] and [13]. It has been shown with numerical experiments, that the improved scheme is superior to the original scheme. Jumps in grid size form no problem at all, whereas the classical scheme produces very inaccurate results for such grids. Besides that the results at a very curved grid show that our curvilinear discretization is at least as accurate as a finite element discretization based on the same grid.

Concludingly we may state that, curvilinear boundary fitted finite volume discretization, using staggered grids, is very well suited to solve the incompressible Navier-Stokes equations, even for non-smooth grids, provided the improved discretization of this report is used.
References


two-dimensional incompressible flows with complex boundaries. *Comp.

Invariant discretization of the incompressible Navier-Stokes equations in

culation using a body fitted coordinate system. *Numerical Heat Transfer*,

13. P. van Beek, R.R.P. van Nooyen, and P. Wesseling. Accurate discrete-
tization on non-uniform curvilinear staggered grids. *J. Comp. Phys.*,

14. P. van Beek and P. Wesseling. Finite volume discretization of the
three dimensional incompressible navier-stokes equation in general co-
dinates. Report 94-60, Faculty of Technical Mathematics and Infor-

15. P.C.W. van Beek and P. Wesseling. Finite volume discretization of the
incompressible Navier-Stokes equations in non-smooth boundary-fitted
coordinates in two dimensions. Report 93-57, Faculty of Technical Math-
ematics and Informatics, Delft University of Technology, Delft, 1993.

16. P. Wesseling, P. van Beek, and R.R.P. van Nooyen. Aspects of non-
smoothness in flow computations. In A. Peters, G. Wittum, B. Herrling,
U. Meissner, C.A. Brebbia, W.G. Gray, and G.F. Pinder, editors, *Com-
putational Methods in Water Resources X*, pages 1263 – 1271, Dordrecht,

17. H.Q. Yang, K.T. Yang, and J.R. Lloyd. Buoyant flow calculations with
non-orthogonal curvilinear co-ordinates for vertical and horizontal par-