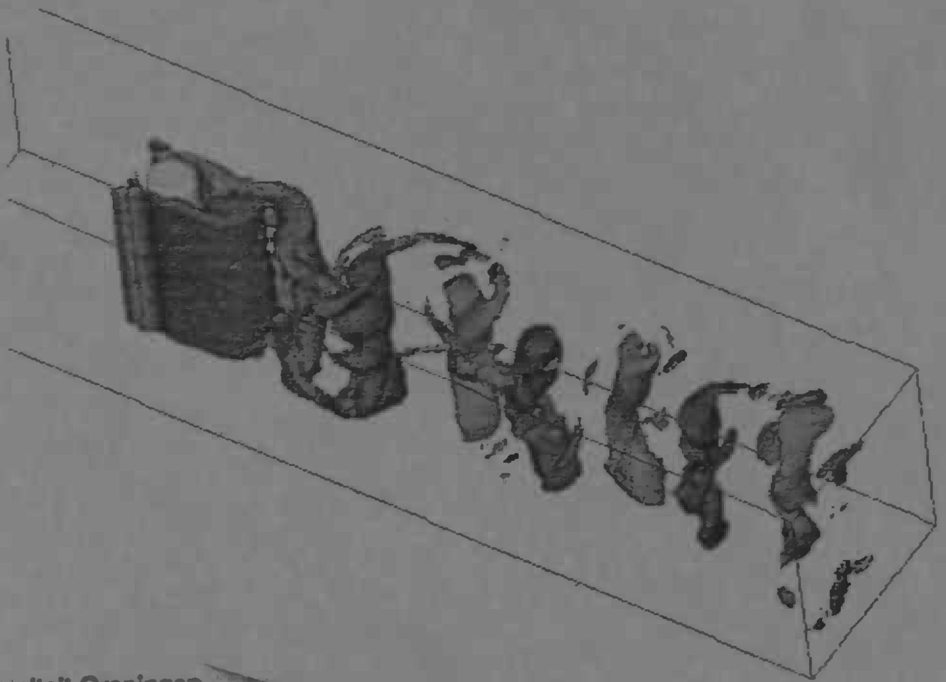
[illegible]

Simulation of Flow past Complex Geometries using Cartesian Grids

Jos Dijkstra



Department of
Mathematics

RuG

Master's thesis

Simulation of Flow past Complex Geometries using Cartesian Grids

Jos Dijkstra

Cover figure : shown is a
3D contour plot of the vorticity
of flow past a circular cylinder

University of Groningen
Department of Mathematics
P.O. Box 800
9700 AV Groningen

August 1997

Contents

Chapter 1

Introduction

Mathematics as a purely theoretical science wouldn't have had such an impact on society if it had not been used in the applied sciences. Even the Egyptians used mathematics to calculate all sorts of things. Nowadays mathematics is used in all branches of science, for example in those areas where fluid mechanics is involved. The analytical equations which describe the evolution of fluid flow in time were presented by Navier and Stokes halfway the previous century. Solving them however is not a trivial thing to do, even with the modern day computers. First of all the equations have to be discretised. With computer aid a solution in a discrete number of points can be obtained. Discretising involves choosing a computational grid covering the flow domain. There are two types of grids we can use: structured and unstructured grids. Structured grids are much easier to produce than unstructured grids but when dealing with complex shaped domains the latter can be made fitting the boundaries exactly. So when using a structured grid, which we will use, special attention must be paid to the boundary handling.

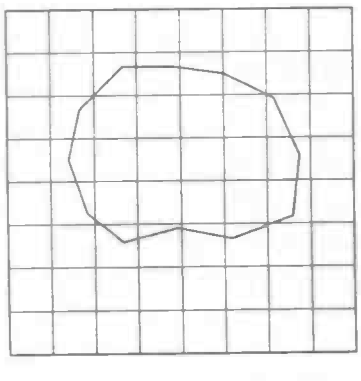


Figure 1.1: *Structured, Cartesian grid*

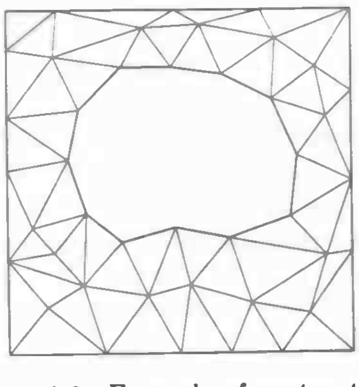


Figure 1.2: *Example of unstructured grid*

In this thesis a refinement is presented of a numerical method using structured Cartesian grids as presented by Gerrits [2]. The cry for more accuracy resulted in a better boundary handling while more stability in time and space was obtained by a adding a higher (4^{th}) order method. Theoretical aspects are explained in chapter 3.

Chapter 4 deals with results, starting with exploring several parameters using flow problems which have an analytical solution. This is followed by examining flow around a circular cylinder, in 2D as well in 3D, at Re=20, creating recirculating eddies behind the object, as well at Re=100 where these eddies are shed off with a characteristic frequency. We will compare the new method for boundary treatment with the old method in combination with a comparison in results using a 2^{nd} and a 4^{th} order method.
Chapter 4 continues with a 2D turbulent pipe flow, a problem which was brought forward by the Gasunie (Dutch gascarrier) who wanted to know the pressure drop in gaspipes as a result of surface roughness such as welch ribblets. Due to the high Re-number, which is about 10^4, special attention had to be paid to stability at the outflow region, a common problem with DNS-solvers.
Conclusions and future developments are discussed in chapter 5 which formally ends this thesis.
The Appendix deals with details about the computerprogram `ComFlo` which was written by J.Gerrits and which was extended in the current research project with inflow and outflow, refined boundary treatment and higher-order discretisation.

Chapter 2

Mathematical Model

2.1 The Navier-Stokes Equations

Consider a complex shaped domain Ω filled with an incompressible fluid. The equations describing the evolution of this system are

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.1}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j^2} \tag{2.2}$$

In these equations p stands for the pressure and u_i are the velocity components in the three directions. The kinematic viscosity ν is equal to $\frac{\mu}{\rho}$ where μ is the dynamic viscosity. The equations (2.1) and (2.2) are the local volume, momentum and mass balances in the bulk of the liquid. Written in vectorform they look like this :

$$\text{div } \boldsymbol{u} = 0 \tag{2.3}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \text{grad})\boldsymbol{u} = -\frac{1}{\rho} \text{ grad } p + \nu \text{ div grad } \boldsymbol{u} \tag{2.4}$$

with $\boldsymbol{u} = (u_1, u_2, u_3)^T$. The above equations can be made dimensionless by scaling the lengths with a characteristic length L and the velocities by a characteristic velocity U. Furthermore, we will normalize ρ to one and we will use the conservative form of the equations since our numerical model is convervation based. Then (2.3) and (2.4) can be written as

$$\text{div } \boldsymbol{u} = 0 \tag{2.5}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \text{div}(\boldsymbol{u}\boldsymbol{u}^T) = -\text{grad } p + \frac{1}{Re} \text{ div grad } \boldsymbol{u} \tag{2.6}$$

With the introduction of the dimensionless quantities another quantity is introduced, the Reynolds number $Re = \frac{UL}{\nu}$. This number represents the relative importance of the convective term in relation to the diffusive term.

2.2 Boundary Conditions

Next we specify the boundary conditions for the velocity at the walls. The stress at a wall is composed of two components, a tangential part and a normal part. A wall is called stress-free or free-slip if the tangential stress vanishes at the wall. Further, the normal velocity always has to be zero due to the inpenetrability of the wall. Hence the conditions are given by

$$\boldsymbol{u} = 0 \text{ on no-slip part of } \partial\Omega \tag{2.7}$$

$$u_n = 0, \; \frac{\partial}{\partial n} u_t = 0 \text{ on free-slip part of } \partial\Omega \tag{2.8}$$

Inflow is defined by Dirichlet boundary conditions on the inflow part $\Gamma_{in} \subset \partial\Omega$

$$\boldsymbol{u} = \boldsymbol{U} \text{ at } \Gamma_{in} \tag{2.9}$$

In case of outflow, many conditions are possible (compare [1]) :

- $\frac{\partial \boldsymbol{u}_n}{\partial n} = 0$
- $\frac{\partial p}{\partial n} = 0, \; \frac{\partial^2 \boldsymbol{u}_n}{\partial n^2} = 0$
- $-p + \nu \frac{\partial \boldsymbol{u}_n}{\partial n} = 0$
- $\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = 0$ (Sommerfeld radiation condition)

Depending on the numerical method it's a matter of trial and error to find out what works best. Since we have to define both the velocity and pressure conditions at the outflow boundary, we have chosen for

$$\frac{\partial u_n}{\partial n} = 0 \text{ and } p = 0 \text{ on } \Gamma_{out} \subset \partial\Omega \tag{2.10}$$

These turned out to work very well in all problems we encountered.
Another combination of in- and outflow boundary conditions is given by

$$p = p_{in}, \; \frac{\partial \boldsymbol{u}_n}{\partial n} = 0 \text{ at } \Gamma_{in}, \tag{2.11}$$

$$p = p_{out}, \; \frac{\partial \boldsymbol{u}_n}{\partial n} = 0 \text{ at } \Gamma_{out} \tag{2.12}$$

In this case the fluid will be driven by a pressure difference. Note that the inflow becomes a "negative" outflow ($p_{in} > p_{out}$).
In the tangential direction we will always use a free-slip condition :

$$\frac{\partial}{\partial n} \boldsymbol{u}_t = 0.$$

Now the mathematical model is complete we will show in the next chapter the numerical model based on the given equations.

Chapter 3

Numerical Model

In this chapter we will deduce the numerical model. Due to presentation reasons this is done in 2D, extension to 3D is straightforward.

3.1 Apertures

Since our domain Ω is covered with a structured Cartesian grid it is difficult to represent the curved boundaries in the numerical method. Boundaries can cross the gridlines at all places so it's important to represent this fact in a numerical method. Therefore we introduce apertures which were intended to be useful when discretising the equations, a method explained in [2]. We however will use them to estimate a distance from velocities to boundaries. In the middle of each cell a *volume*-aperture is placed which indicates the fraction of fluid in this cell. In the middle of each cell-face an *edge*-aperture is placed which indicates how much of the cell-face is open. All apertures take values between 0 and 1.

3.2 Labeling

Each cell will receive a label which indicates the function of this cell in the numerical method. How this works is made clear in later sections, for now we will only describe these labels. Based on their volume-apertures and those of their neighbours (6 in 3D) cells can be labeled a Fluid cell (**F**-cell), a Boundary cell (**B**-cell) or an Xterior cell (**X**-cell). A cell is called an **F**-cell when it has a volume-aperure $\geq 1/2$. A cell is called a **B**-cell when it is not an **F**-cell but has at least one neighbour **F**-cell. All other cells become an **X**-cell but they don't have any use. These labels are enough to make a model for flow inside geometries work. When inflow and ouflow is needed some cell labels can be changed. An **I**-cell indicates an inflow cell and is usualy positioned near an **F**-cell. The same can be said about an **O**-cell which represents outflow. When dealing with an incompressible fluid it is very common to use a staggered grid. This means positioning the pressure in the middle of a cell and the velocities in the middles of the cell-faces. However, we will only place a pressure in an **F**-cell. Pressures in **B** or **I** cells are not needed since they are eliminated by using a boundary velocity (see section 3.5) while the pressure in **O**-cells is given as part

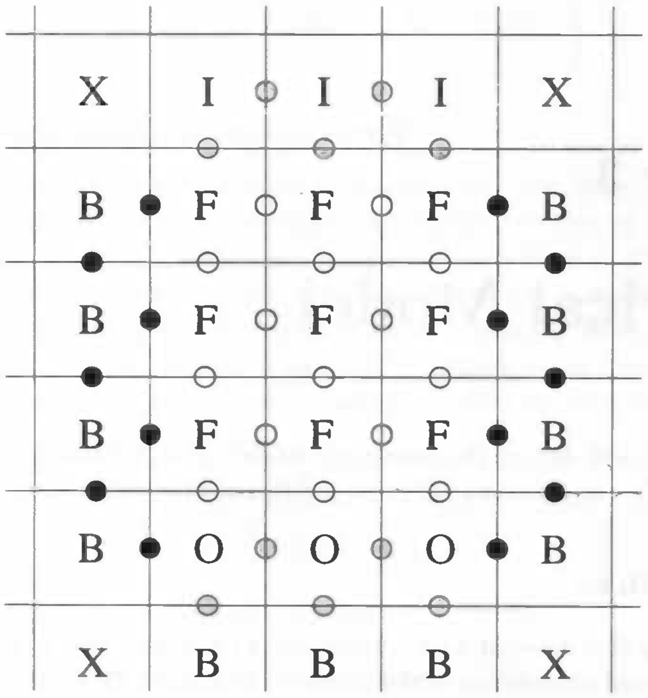


Figure 3.1: *Cellnames and positioning of the velocities*

of the outflow condition. Velocities are only placed between combinations of **F**,**O**,**I** and **B**-cells. The labels of the velocities are formed with the 2 cell labels adjacent to this velocity. This results in the following variety of labels :

- **FF**,**FO**-velocities are computed from the momentum equation
- **BF** and **FB** velocities are computed from the boundary conditions, see section 3.6
- **FI** velocities are given by means of Dirichlet conditions
- **BB** velocities are also computed from boundary conditions
- **OB** velocities have to be set to an appropriate discretised outflow condition
- **FX** velocities cannot appear since there must always be a **B**-cell in between
- **BX** velocities don't have to be computed because they are not in the domain of our discretised equations

Figure 3.2 shows a sample grid with inflow and outflow. Note that the inflow cells are treated as **B**-cells, while the outflow cells are treated as **F**-cells and as **B**-cells, that is, **FO** velocities are solved from the momentum equation, while OO velocities are boundary velocities, in particular a free-slip condition is applied here.

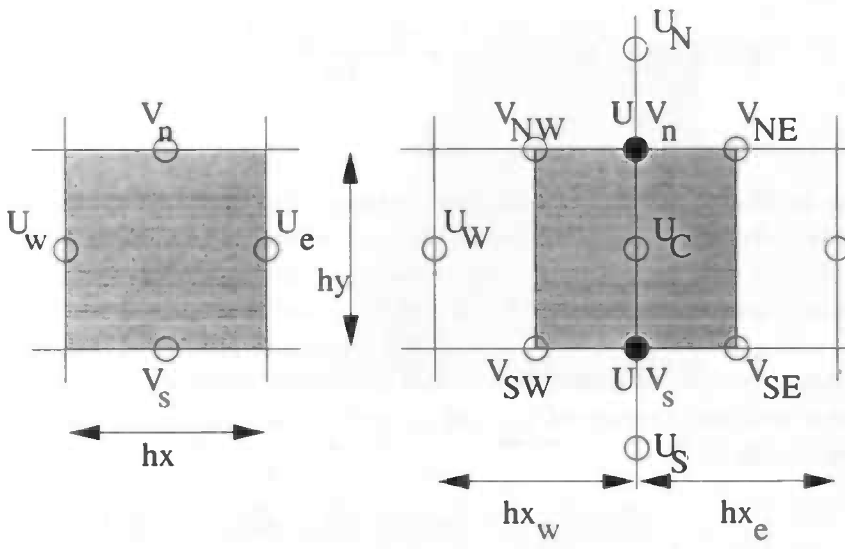


Figure 3.2: *Control volumes for continuity and momentum equations*

3.3 Discretised Navier-Stokes Equations

The governing equations were given in chapter 2. In order to solve them we have to discretise them in time as well in space. How this works is explained in the next section.

Time Discretisation

The equations (2.5) and (2.6) are discretised explicitly in time. If we use the most elementary explicit time integration method *Forward Euler* we get:

$$\text{div } \boldsymbol{u}^{n+1} = 0, \tag{3.1}$$

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\delta t} + \text{grad } p^{n+1} = -\text{div } (\boldsymbol{u}^n \boldsymbol{u}^{n^T}) + \nu \text{ div grad } \boldsymbol{u}^n \tag{3.2}$$

where the superscript n indicates the quantity at $t = n\delta t$. The pressure gradient is discretised at the new time t_{n+1} to ensure that the new velocity field is divergence free.

Spatial Discretisation

In this subsection the spatial discretisation is explained. Formulae are valid for situations when all occuring velocities are **FF**-velocities. In the vicinity of boundaries a modified scheme is used which is explained in the next section.

Continuity equation

We will begin with discretising the continuity equation. This is done in a standard manner, dividing the difference of velocities by the distance between them :

$$\text{div } \boldsymbol{u}^n \doteq \left(\frac{u_e^{n+1} - u_w^{n+1}}{hx} + \frac{v_n^{n+1} - v_s^{n+1}}{hy} \right) \tag{3.3}$$

Momentum equation - convective terms

Next we have to discretise the momentum equations. For both the convective and diffusive terms a small conservation cell is used, contrary to the one which was used in the old method. This is because we don't have problems with the volume-apertures now. Furthermore, it's essential to use a small cell when we also want to make use of the 4^{th} order method discussed later on. It uses a three times larger control volume which would otherwise become too large. If we take u_n as a weighted average of u_N and u_C and so on we can discretise the convective terms like (3.3)

$$\text{div} \begin{pmatrix} u^n & u^n \\ u^n & v^n \end{pmatrix} |_C \doteq \left(\frac{\frac{1}{4}((u_e + u_c)^2 - (u_c + u_w)^2)}{\frac{1}{2}(hx_w + hx_e)} + \frac{u_n^n v_n^n - u_s^n v_s^n}{hy_c} \right) \tag{3.4}$$

Momentum equation - diffusive terms

The pressure gradient is discretised in the standard manner, i.e.

$$\frac{\partial p^{n+1}}{\partial x} |_C \doteq \frac{p_e^{n+1} - p_w^{n+1}}{\frac{1}{2}(hx_w + hx_e)} = G_h p^{n+1} \tag{3.5}$$

with G_h being the discretised grad operator. Furthermore,

$$\begin{aligned} \text{div grad } u^n |_C \quad &\doteq \quad \frac{hx_e u_w^n - (hx_w + hx_e) u_C^n + hx_w u_e^n}{\frac{1}{2}(hx_w + hx_e) hx_w hx_e} \\ &+ \quad \frac{(hy_c + hy_s) u_n^n - (hy_n + 2hy_c + hy_s) u_C^n + (hy_n + hy_c) u_s^n}{\frac{1}{2}(hy_n + hy_c)(hy_c + hy_s) hy_c} \end{aligned}$$

3.4 Near Boundaries

When discretising near boundaries the standard method can be used, using virtual velocities on the 'missing places'. In the search for higher accuracy another method was implemented. The idea is to use the 'exact' given boundary condition directly in the numerical scheme. To that purpose we have to give the molecule a more flexible form, it has to reach towards a boundary. This is illustrated in figure 3.4. The left figure shows the changed control volume for the *div* operator. The control volumes for the momentum equations are changed too. This is shown in the right part of figure 3.4. The combinations between exact velocites and operators are the following :

- continuity : exact boundary conditions are used in all directions

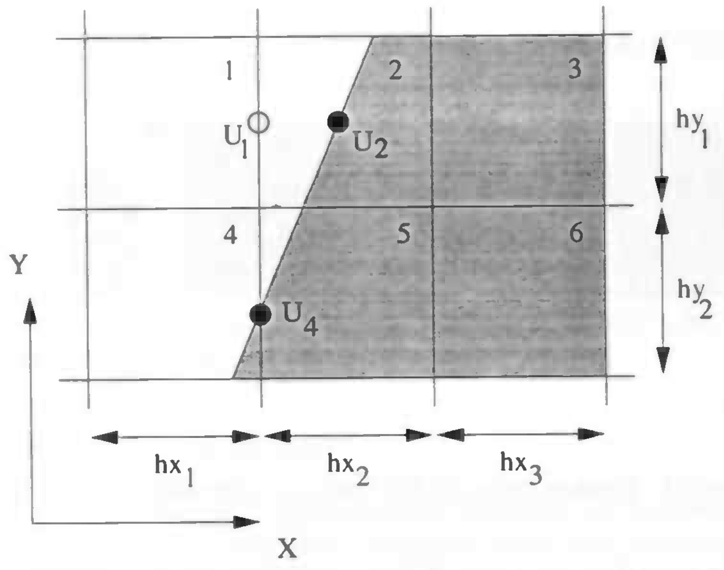


Figure 3.3: *Estimating distances with apertures*

- convective terms only make use of exact conditions in the longitudinal direction. In the other directions they make use of virtual velocities. This is because these velocities are used to calculate derivatives in other directions than the directions they were extrapolated in
- diffusive terms make use of all the exact velocities in all directions

In very rare cases it might happen that a boundary is too far away to use the exact boundary velocity. But since we can't stretch the molecule into the infinite we will use a virtual velocity. This can only happen for the momentum equations discretised in the other 2 than the longitudinal directions, e.g. when discretising in x-direction the $\partial^2 y$ and $\partial^2 z$ might give problems (diffusive terms). Returning to our discretised operators, this means that $hx, hy, ..$ are changed according to the distance towards a boundary. These distances are calculated with the help of apertures. This is shown in figure 3.3. The distance from U_1 towards U_2 is estimated with the help of volume-apertures :

$$\text{distance} \doteq hx_2 * F_b^2 + hx_3 * F_b^3$$

The distance between U_1 and U_4 is given by

$$\text{distance} \doteq 0.5 * hy_1 * A_x^1 + hy_2 * A_x^4$$

where we make use of the edge-apertures.

3.5 Solution Method

In this section we will explain how the discretised equations are solved. First we introduce some notation. Let Ω_h^f be points in the computational grid (**FF-**

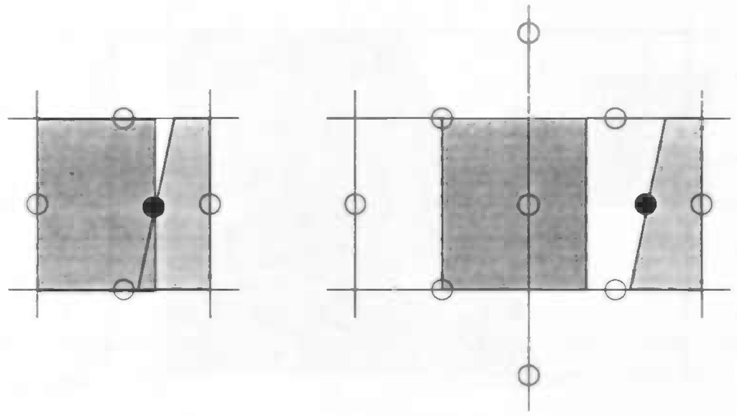


Figure 3.4: *Control volumes for enhanced boundary treatment*

velocities) and let Ω_h^b be the set of boundary velocities (**FB**,**BF** and **BB**-velocities). Furthermore let u_{exact} be the given exact boundary condition. The initial velocity u^0 is given on Ω_h^f, followed by computing the boundary velocities on Ω_h^b, eventually followed by setting u_{exact}. Next step is to compute the new velocity field and a new pressure distribution. To do that, we first construct a temporary vector field $\tilde{u}$ (not velocity field) by integrating the momentum equations without the pressure gradient :

$$\tilde{u} := u^n - \delta t D_h(u^n u^{n^T}) - \nu D_h G_h u^n \text{ on } \Omega_h^f \tag{3.6}$$

where D_h is the discretised divergence operator. At this stage we can use the exact boundary velocities and/or the virtual velocities. For notational convenience the same D_h is used. Now it remains to solve

$$D_h u^{n+1} = 0, \tag{3.7}$$

$$u^{n+1} + \delta t G_h p^{n+1} = \tilde{u} \tag{3.8}$$

Substituting (3.7) in (3.8) requires the pressure in the boundary cells because D_h works on the entire grid. Therefore we split D_h into two parts :

$$D_h = D_h^f + D_h^b \tag{3.9}$$

This results in

$$(D_h^f + D_h^b)u^{n+1} = 0, \tag{3.10}$$

$$u^{n+1} + \delta t G_h p^{n+1} = \tilde{u} \tag{3.11}$$

We cannot compute $D_h^b u^{n+1}$ since the new velocity field on Ω_h^f is yet unknown so we will use the exact boundary conditions and/or the virtual velocities, the latter based on a velocity field u^n. Then it remains to solve :

$$D_h^f u^{n+1} = -D_h^b u_b, \tag{3.12}$$

$$u^{n+1} + \delta t G_h p^{n+1} = \tilde{u} \tag{3.13}$$

Now we can substitute (3.13) in (3.12) from which we get the **Pressure Poisson equation** :

$$D_h^f G_h p^{n+1} = \frac{1}{\delta t}(D_h^f \tilde{u} + D_h^b u_{exact}) \tag{3.14}$$

From (3.14) we can compute the new pressure distribution p^{n+1} which leads with

$$u^{n+1} = \tilde{u} - \delta t G_h p^{n+1} \text{ on } \Omega_h^f \tag{3.15}$$

to new velocities on Ω_h^f. Finally the new boundary velocities are computed after which we have the velocity field on the entire grid Ω_h.

3.6 Boundary Velocities

Next we will briefly explain how the boundary velocities are computed from the new velocity field. For more information we refer to [2]. The general principle is that an **FB** or **BB**-velocity is calculated based on interpolation or extrapolation of the desired condition on the wall with an **FF**-velocity. An example is shown in figure 3.5 with a no-slip boundary condition.
Inflow velocities are Dirichlet conditions and can be set easily to the desired value(s). With the help of **II**-velocities the direction can be changed but they can also be made free-slip so the inflow becomes more flexible when e.g. pressure waves bounce into the inflow area.
At the **outflow** region we have two types of conditions. The first concerns the normal part of the flow and is $\frac{\partial u}{\partial n} = 0$, the actual outflow condition. The second is a free-slip condition in both tangential directions in outflow cells. Here the **BB**-velocities are set equal to their neighbour **FF**-velocities.

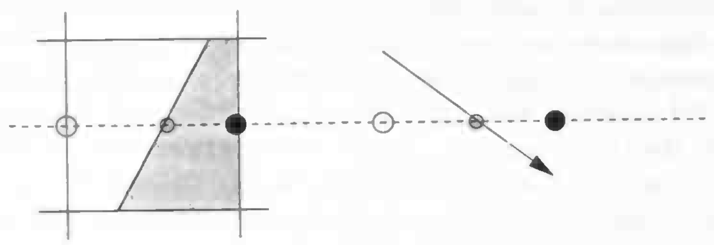


Figure 3.5: *Principle of extrapolation*

3.7 Higher Order

In section 3.4 is explained how to solve the equations. This was done by applying a control volume for integrating the momentum equations. To eliminate the leading term of the truncation error of the second order method the momentum equations are also integrated over a three times larger control volume. This

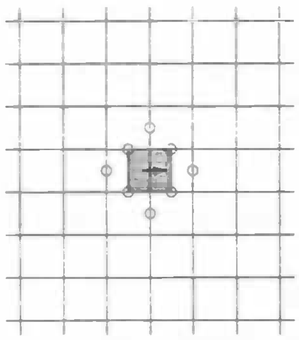


Figure 3.6: *velocities occuring in 2nd order scheme*

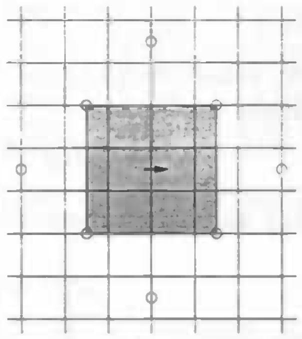


Figure 3.7: *velocities used in 4th order scheme*

second integration has a truncation error which is exactly 9 times larger than the error when using a small control volume. By substracting 1/8 of the larger volume from 9/8 times the small cell eliminates the second order term of the truncation error resulting in a fourth-order spatial discretisation. By using a three times larger volume we don't have problems with missing velocities in the vertical directions. This method can also be applied on weakly stretched grids. In the vicinity of boundaries the three times larger cell does not fit into the flow domain. Hence we will not use the larger cell here so near boundaries we still have a second order method which is justified by the fact that the boundary $\partial\Omega$ is of lower dimension than Ω itself. This approach in very common for higher order methods yet still resulting in largely improved results when compared to second order methods. More details of this approach can be found in [10].

3.8 2D Problems

Since most problems concerned with fluid flow past an object are 2D because of lack of computing power, we have to make clear how to compute 2D problems in a 3D environment. This is done by extending the 2D problem to 3D by using 1 cell in the 3rd direction and apply free-slip conditions on the two velocity components in the third direction. Hence no force is exerted on the velocity components in the 2D-plane. Furthermore, we set the velocity in the third direction equal to zero. Combined with the free-slip condition this guarantees a 3D flow which is in fact 2D.

Chapter 4

Results

4.1 Simple Testcases

In order to test the new implementations concerning inflow, outflow and free-slip and to determine the values of some important parameters we selected the following problems :

1. tube with uniform inflow and no-slip walls
2. tube with pressure difference on both ends with no-slip walls
3. tube with uniform inflow and free-slip walls
4. tube with pressure difference on both ends with free-slip walls

Since we can analytically calculate the velocity at the ouflow for all four problems it's easy to check if the results are of satisfactory accuracy. A tube was chosen to demonstrate the capability of computing flow in a round object on a (rectangular) Cartesian grid. In the first case the tube has diameter 1 and length 2 situated on a uniform grid of 15x15x30 (grid 1) and 30x30x60 (grid 2). In the other cases the tube has length 1 where the grids were also halfed in length direction. The finer grid was chosen in order to see the effects of the discretisation error.

Case 1

We expect the flow to become a Poiseuille flow, that is, the outflow velocity profile should be parabolic with a maximum velocity being twice the average inflow velocity, in this case $W = -1$. This follows from conservation of flux :

$$\text{flux out } = \pi \int_0^b a^2 \left(\sqrt{\frac{-y+b}{b}} \right)^2 dy = \frac{1}{2}\pi a^2 b \equiv a^2\pi = \text{ flux in } \Rightarrow b = 2$$

As can be seen in table 4.1 grid 1 differs 2.3% from the desired value. The second grid however is more accurate. A decrease of eps to $1 * 10^{-5}$ didn't change anything. Because of friction with the wall the velocity profile at the beginning of the tube won't be parabolic, it takes a long distance to become so. For this reason the tube was taken twice as long.

	grid1	grid2
w_{max}	1.9533	1.9855
expected	2.0000	2.0000
deviation	2.3 %	0.72 %

Table 4.1: *Results case 1*

	grid1	grid 2
w_{max}	3.3382	3.2339
expected	3.3482	3.2328
deviation	0.299 %	0.034 %

Table 4.2: *Results case 2*

Case 2

Next we prescribe on both ends a (different) pressure. In our case, $p_{in} = 2$ and $p_{out} = 0$. The velocity profile at a random cross-section of the tube has to satisfy ([7]) :

$$w = -\frac{1}{4\nu}\frac{dp}{dz}(R^2 - r^2) \tag{4.1}$$

with R the diameter of the tube and r the distance from the center of the tube. The outflow profile became parabolic with maximum velocities at the center of the ouflow area. Results are shown in table 4.2. Analytically we expect the follwing values for w_{max} (ν=0.04, $R = \frac{1}{2}$, dz measured from the centers of the cells) :

- grid 1 : $-\frac{1}{4*0.04}\frac{-2}{2*29/30}\frac{1}{4} \doteq 3.3482$
- grid 2 : $-\frac{1}{4*0.04}\frac{-2}{2*59/60}\frac{1}{4} \doteq 3.2328$

Case 3

Since we have free-slip nothing should happen to the fluid since no force is exerted on it. Therefore we expect the ouflow-velocity to be the same as the inflow-velocity. The second grid differs 0.03% (with eps=$1 * 10^{-4}$) from the

	eps=1e-4		eps=1e-5	
	grid 1	grid 2	grid 1	grid 2
$t = 0s$	-1.0000	-0.9997	-1.0000	-0.9999
$t > 0s$	-1.0000	-1.0000	-1.0000	-1.0000

Table 4.3: *Results case 3*

desired value at the first timestep which can be explained by local differences in flux combined with the large grid and the initial velocity field of $u = v = w = 0$. When the iteration error is decreased the results become more accurate.

Case 4

As in case 3 we prescribe different pressures on both ends of the tube. Because of the free-slip walls the fluid should not reach a steady velocity but should accelerate uniformly. The velocity of the flow is given by

$$w(t) = at \text{ with } w(0) = 0;\ u(t) = v(t) = 0 \tag{4.2}$$

	grid 1		grid 2	
time	w	a	w	a
0.0s	0.00000		0.00000	
0.1s	-0.21428	-2.1428	-0.20689	-2.0689
0.5s	-1.071428	-2.14286	-1.03448	-2.06896
1.0s	-2.14285	-2.14285	-2.06896	-2.06896

Table 4.4: *Results case 4*

The acceleration a is given by :

$$\Delta p = \frac{F}{A} = \frac{ma}{A} = \frac{\rho Aha}{A} = ha \Rightarrow a = \frac{\Delta p}{h}$$

Note that ρ was normalized to one. Furthermore h is the length of the tube measured from the centers of the cells at inflow and outflow. The results can be found in table 4.4. The iteration error was eps=10^{-6} to show that the results can be as precise as wanted. When eps was taken to be 10^{-4} only the first three digits would be significant, now this applies for the first 5 digits. Analytically we expect the following values for a :

- Grid 1 : $a = \frac{-2}{14/15} \doteq -2.14285$
- Grid 2 : $a = \frac{-2}{29/30} \doteq -2.06896$

Since the correct results are produced we can conclude that the program has passed this test as well.

4.2 2D Flow past a Circular Cylinder

In this section we consider one of the fundamental problems of fluid flow namely that of flow past an object. This has been examined in both computational and experimental studies and is considered a stringent test for flow solvers. We consider a flow past a cylinder at Reynolds numbers 20 and 100 with *Re* based on the diameter of the cylinder. For lower *Re*-numbers we expect the flow to become stationary creating recirculating eddies behind the cylinder. When *Re* is increased the flow may become unstable above some cirital value of *Re*, usualy around 45. Numerical errors are big enough to create disturbances in the flow. These will cause the flow to become chaotic. A well known property is vortex shedding where the eddies are shed off from behind the object with a certain frequency. Our first goal is to check several characteristic lengths and frequencies. Part of these computations was an analysis concerning the effects of varying meshsizes and numerical enhancements. We shall compare our data with that obtained from a workshop [8]. Furthermore, we shall compare the results using different numerical models or modifications on them. The inflow profile is parabolic and given by

$$u = 4.0 * U * (xp - 0.41)/(0.41 * 0.41)$$

	$Re = 20$		$Re = 100$			
	Length cell		Strouhal number			
grid	standard	'exact'	standard	'exact'	4^{th} order	both
20x100	0.07	0.07	0.278	0.286	0.287	0.283
40x200	0.079	0.082	0.285	0.283	0.302	0.299
80x400	0.080	0.083	0.297	0.294	0.304	0.300

Table 4.5: *Comparison between different models at different Re-numbers*

where U is given by $U = 0.3$ in case of $Re = 20$ and $U = 1.5$ when $Re = 100$. The Reynolds numbers are based on the average inflow velocity.

$Re = 20$

With Re=20 we are well below the critical Re-number marking the border between steady and unsteady flow. As mentioned above recirculating eddies are formed behind the object and don't change in time after arriving at their steady size which can also be seen in figure 4.2 where velocities in the two directions at a point behind the cylinder are plotted versus time. A characteristic length could be the length of this cell, made dimensionless by dividing it by the diameter of the object. Comparing many results contributed to the workshop a range of [0.084,0.085] was determined to be the interval in which the 'exact' value could be found. Comparing this with our result we may conclude that though not exactly in the right interval, the "exact" boundary treatment does a better job than the standard one though not dramatically. Furthermore we can see a change for the better by taking a finer grid which was to be expected.

$Re = 100$

Comparing this situation with the one described above we see that the recirculating eddies are shed off. Due to the higher inflow velocity the eddies are

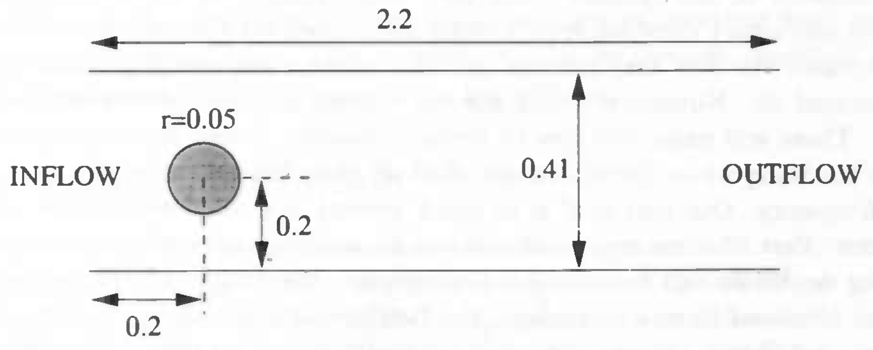


Figure 4.1: *Situation for flow past circular cylinder*

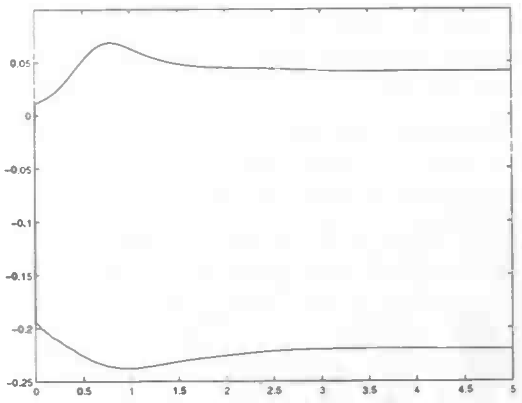


Figure 4.2: *Evolution of velocities in time at* $Re = 20$

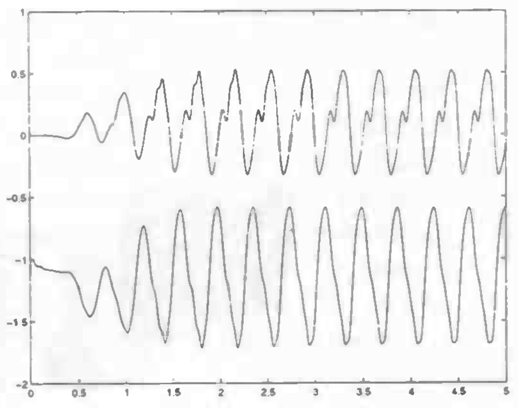


Figure 4.3: *Velocities versus time at* $Re = 100$

becoming larger than in the $Re = 20$ case. Because they are larger they become sensitive for disturbances which are usualy caused by numerical errors. One eddie starts to grow more than the other eddie thus leaving less space for the latter. It will not grow into infinity but starts to move away from the object after a while giving the other eddie an opportunity to grow, and to move away and the whole process will start all over. This shedding has a characteristic frequency f. The Strouhal number is a dimensionless version of this frequency and is defined as

$$St = \frac{fD}{U}$$

where D the diameter of the cylinder and U a characteristic velocity, in this case the avarage inflow velocity. As in the $Re = 20$ situation, we will compare the results from different models and a higher order method with results obtained from the workshop ([8]) which gave values of St in the range of $[0.299, 0.301]$. In table 4.5 we show the results using the standard method and modifications. First we notice the improvement in results when using the exact boundary treatment. The higher order method pushes the St-number further into the desired direction, sometimes causing an overshoot. When both methods are combined we see an almost perfect result, even on relatively coarse grids. Comparing "standard" with "both" we see we can win a factor 2 in every direction. Combined with the factor time we may conclude we can win at least a factor 2*2*2 = 8 to achieve the same results, in 3D this may even be larger.
Finally in figure 4.3 the evolution of the two velocities in time are shown at some point behind the cylinder. After approximately 2 seconds the flow becomes periodic. With help of this data it is easy to calculate the frequencies.

4.3 3D Flow past a Circular Cylinder

The 2 dimensional cases described above are good enough to test programs for accuracy. Most problems though arise from the 3D world so the next is to

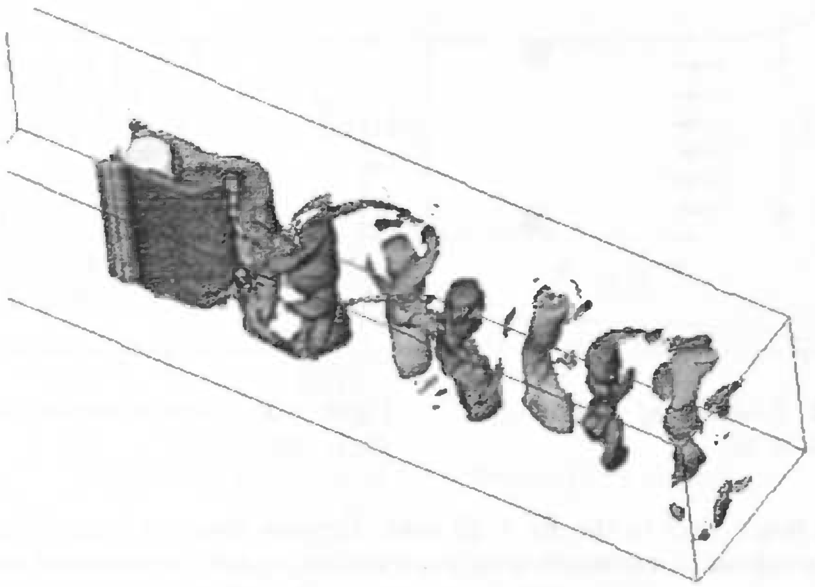


Figure 4.4: *3D flow past circular cylinder*

show that the method also works with 3D flow problems. We selected the same problem as above but now, surprise, the 3D case. The cylinder however was moved so the distance between inflow and center cylinder became 0.5. Again we prescribe a (3D)-parabolic inflow profile yielding a Re-number of 100. Due to instability of the problem we did not even try it without the higher order method. We will only show the possibility of 3D calculations at low Reynolds numbers. The grid we used was $40 * 40 * 200$ yielding a total number of 320.000 cells. On a normal workstation the calculation time was 13h with a timestep $dt = 5 * 10^{-4}$ and a simulation time of 6 sec. The obtained Strouhal number was $St = 0.352$ which is not exactly in the interval of [3.1900, 3.2100], again results from a workshop [8] but considering the grid a very good answer.
When looking at figure 4.3 we can clearly see that the vortices are bending when dragged along with the flow, a typical 3D effect. Eventually they will collapse and disappear, in contrary to the 2D situation where the vortices don't seem to disappear after some time. This is partly due to the free-slip condition in the third direction so no force is exerted on the vortices.

4.4 Industrial Problem

Next step in our development process was to examine whether it was possible to do industrial problems with Re numbers in the range of 10^4 till 10^8. A problem was supplied by the Gasunie, the Dutch gascarrier. They make use of pipes to transport the gas. These pipes are welded together causing a little bump inside the pipes. The question was how these bumps influence the transportation of gas.

First of all, considering speed of gas, pressure and density, the normal Re-number for this problem would be around 10^6 till 10^8. Because it's a realistic problem it would be nice to do the simulations in 3D. This however is not yet possible with the modern computers in combination with our numerical method which is called DNS or Direct Numerical Simulation. The idea behind this is that the Navier Stokes equations are believed to be correct for every scale of length in the problem, even when the flow becomes very turbulent thus possessing very small scales. But unfortunately, a very fine grid has to be used to catch these details which automatically results in a huge number of grid points and very small timesteps. In order to 'ontlopen' these fine grids the turbulence can be modelled thus "simulating" it on a coarser grid which is time consuming but less acurate. Various methods have been developed according to this idea. We didn't use any turbulence models so we had to restrict our testcase to 2D with $Re = 10^4$, yet causing enough problems. Goal was to explore the problem regarding parameters (gridsize, timestep,etc) and possibilities of the numerical method for use in future research.
Empirically we can say a few things. First of all there is a formula deduced by Dean :

$$\frac{dp}{dx} = -\frac{\rho U_b^2}{h} * C_f$$

with C_f defined as

$$C_f = 0.073 Re^{-0.25}$$

which gives a relation between the Re-number and the pressure drop. With ρ, U_b and h all normalized to 1 we get

$$\frac{dp}{dx} = -0.0073$$

Another relation, also deduced by Dean, is given by

$$\frac{U_{max}}{U_b} = 1.28 Re^{-0.0116}$$

which gives a relation between the ratio of the maximum and average velocity and the Re-number.

When dealing with Re-numbers in the turbulent region it is very common to use a buffer zone at the end of the channel/pipe to reduce the effect of pressure waves bouncing back towards the inflow. This causes numerical instability of the worst kind. Therefore the Re-number is reduced by linearly increasing the viscosity in this buffer zone to 10^{-2} yielding a Re-number of 100 at the outlet. The buffer had a length of 16.6% compared to the total length. An overview of the situation is given in figure 4.4.
As a grid we used 60x480 with large stretching ($\alpha = 10$) towards the middles of the bumps. As timestep we used $dt = 10^{-4}$. The inflow profile was uniform $U = 1$. Due to the no-slip on the wall the flow has to adjust to this which takes three times the diameter of the inflow. First of all a comparison can be made by an actual 3D flow profile as was measured and the computed 2D profile (see 4.6). A striking difference is the flat middle part of the computed profile

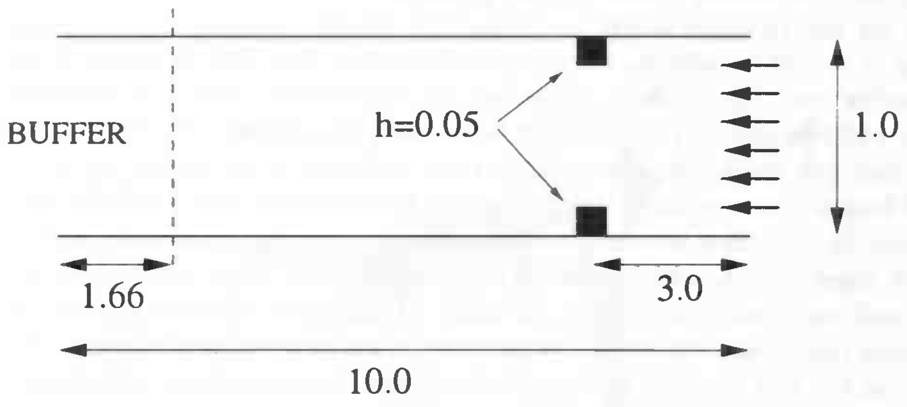


Figure 4.5: *Overview of the situation (with bump)*

compared to the measured profile. This is probably due to 3D effects which we didn't have with our 2D simulation. Several 2D calculations can be found in literature which are all flat. When looking at the pressure drop without the bump we see it isn't linear as it was with the low *Re*-cases. We can calculate the drop of pressure per unit of length :

$$\frac{dp}{dx} = \frac{0.08}{8.16} \doteq 0.0098$$

where is 8.16 is the length of the tube without the buffer zone. Compared to the predicted value of 0.0073 we are about 34% higher. This could be explained by the use of a relative coarse grid, new calculations on a much finer grid should prove this. And one can only speculate about the effect of the buffer zone on the results. It's evident that it disturbs the flow, it acts like a semi-permeable cork. Comparing the pressure drop without the bump and with the bump one can clearly see that the pressure has dropped a little. This bump in 2D has a smaller area than compared with the 3D situation so it is to be expected that 3D calculations show more effect of the bump on the pressure drop. In figure 4.7 we see plots of the pressure avareged in time along several slices in z-direction plotted in one figure. Note the non-linear drop of the pressure. Again several plots combined in one can be seen in figure 4.8, this time with a bump inside the channel. A snapshot can be seen in figure 4.9 where a contour plot of the vorticity and a vector (velocity) field are plotted. From this it is clear that vortices are shed off from behind the objects.

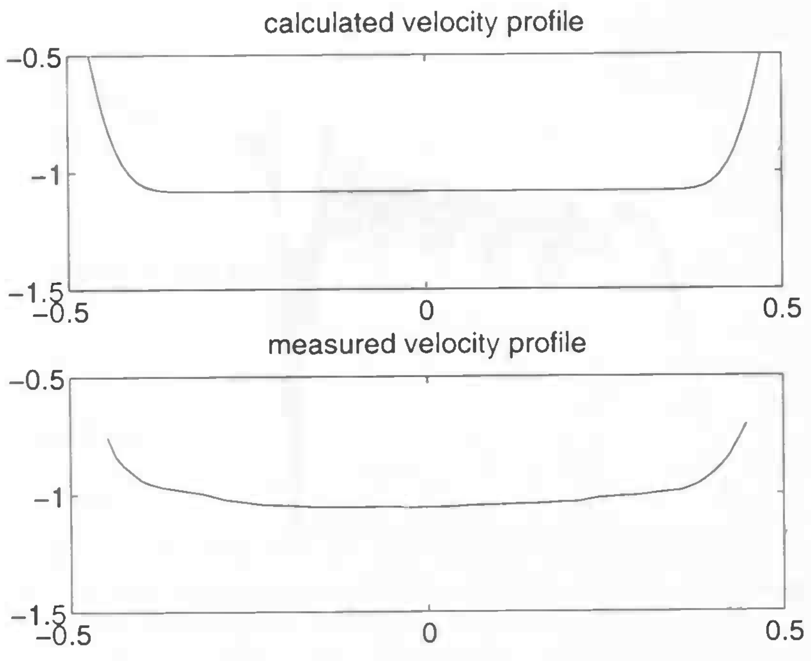


Figure 4.6: *Comparison of flow profile, calculated 2D v.s. measured 3D*

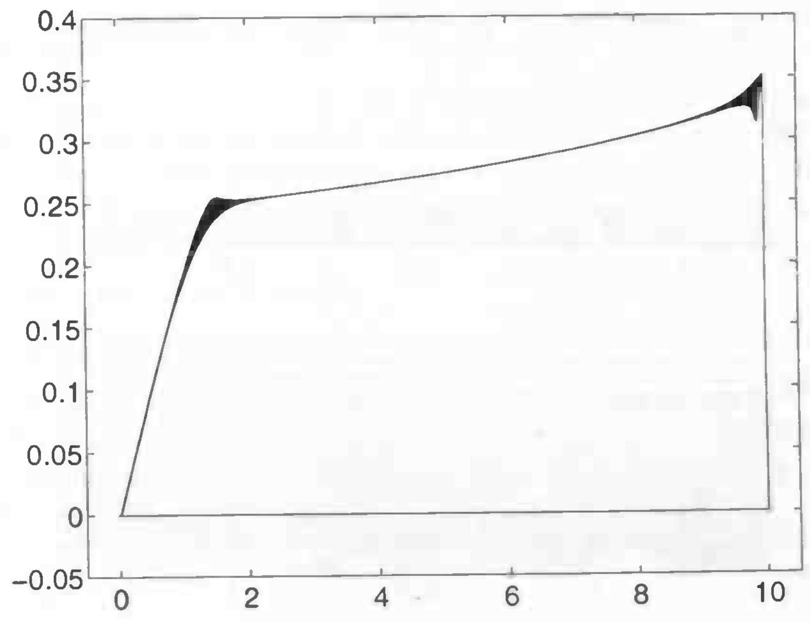


Figure 4.7: *Pressure drop in channel without bump*

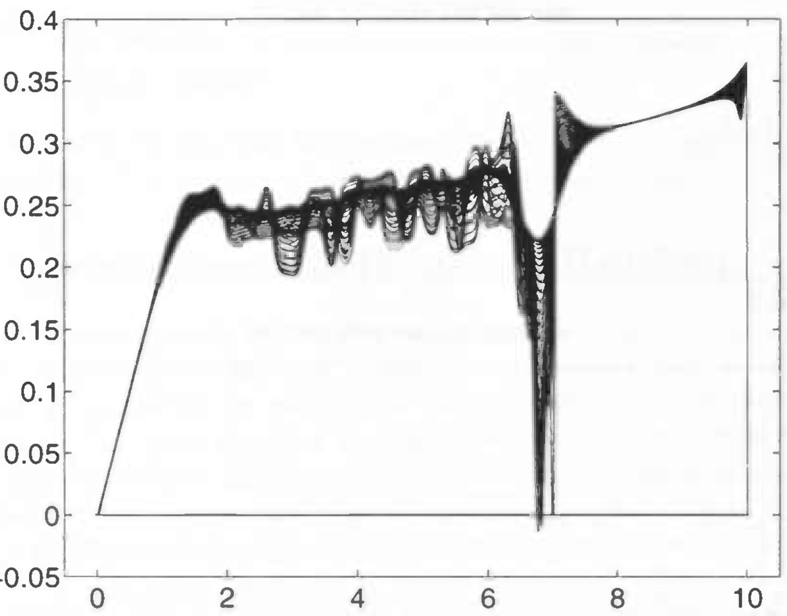


Figure 4.8: *Pressure drop in channel with bump*

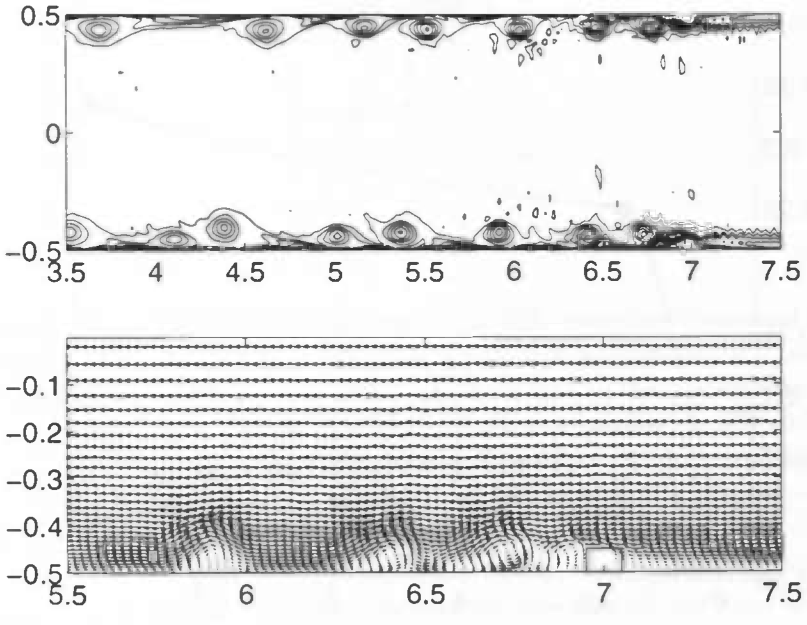


Figure 4.9: *Fully developed 2D turbulent flow past bumps*

Chapter 5

Conclusions

5.1 Results

Low *Re* cases

Evaluating the results from the previous chapter considering flow past a circular cylinder we may conclude that `ComFlo` has become an accurate solver for instationairy flow problems at low Reynold numbers. In particular the easy-to-implement 4th order method does a great job. This in combination with the "exact" treatment of the boundaries ensures very good results. Another way to use the apertures is to use them in the discretisation. This was done in the previous version of `ComFlo` but was eliminated due to inaccuracy. A combination of apertures and exact boundary conditions could be better than the last one alone, see next section.

High *Re* cases

The industrial problem turned out not as good as we wanted. Problems with numerical stability could be handled with a buffer zone at the end of the canal but the results concerning drop of pressure were not in accordance with analytical results. So some further investigation has to be done. Other things of interest could be :

- 2D canal flow at $Re = 100.000$
- 3D flow at $Re = 10.000$

The 3D case is of very much interest because it's still not known how the 2D canal and 3D tube are related to each other. Furthermore the use of different outflow (and inflow) conditions should be investigated because this is the only thing that can be changed without wrecking the numerical method itself. Suggestions in literature are for example periodic conditions imposed on inflow and outflow regions.

5.2 Future Development

As for future development the following features could be added :

- Heat transfer
- Turbulence models
- Graphical User Interface

We remark that a version with free surface flows has been completed but still needs perfection.

Further Improvement of Boundary Handling

New ideas concerning accuracy are explained here. The idea to use apertures by means of actual flow (aperture times velocity) seemes very natural. But when using extrapolation we should not use apertures since the extrapolated velocity has no physical meaning so multiplication with an aperture has no physical meaning either. We suggest the following. Try to move the velocities to places where they actually mean something, so move them to a place between a gridline and the boundary. Now it's relevant to use apertures. A velocity disappears when an aperture of the concerning cell-face equals zero. As could be seen when using the exact treatment we didn't have any stability problems due to moving certain velocities so we don't expect this to happen with the new method.

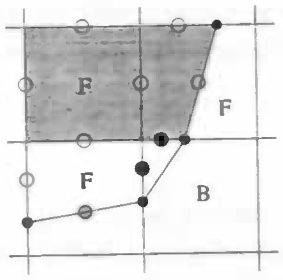


Figure 5.1: *new placement of velocities with boundary fitting conservation cell*

Comparing this to the previous methods, the **BB**-velocities disappear. Instead we introduce **node**-velocities, indicating a pair **u,v,w** situated in a node-point (small black dots) which provide information needed in the discretisation. Near boundaries we will use discretisation formulas as explained in [2] which make use of apertures, also the volume-apertures. Since it is difficult to make use of a small conservation cells due to problems with calculating the volume-aperure of a small cell, large conservation cells are used near boundaries. The discretisation of the diffusive terms remains the same, though experimenting with the use of apertures is an option. When no boundary velocities of any kind are needed we will use the discretisation discussed in this report (with small conservation cells) so we can still make use of the higher order method.

Appendix A

Program Description

The previous chapters dealt with the theoretical aspects of a numerical method. This method was implemented in the program ComFlo. The structure and subroutines of ComFlo are discussed in the next chapter.

A.1 Flow-chart

The program ComFlo consists of several subroutines which are called in the following order.

setup/pre-processing	SETPAR		
	GRID		
	LABEL	BDNDEF	
		BNDDEFIO	
	SETFLD or LDFLD	BCIO	
		BC	
	COEFL		
time integration	INIT		
	TILDE		
	SOLVEP	COEFR	
		PRESIT	SLAG
		BCIO	
		BC	
		MATLAB	
		SAVEFLD	
post-processing	SAVEFLD		

For a description of these subroutines we refer to section A.3
The program starts with initializing the apertures, labels and the velocity field. This can be done by using a previous field or by setting it to the desired values. Furthermore, the coefficients needed in the left side of the Poisson equation are calculated. The time integration is repeated until tmax is reached. This is the actual calculation of fluid motion evolving in time. Starting with initializing the flow field, this means copying the flow field to a temporary field followed by calculating the vector field $\tilde{u}$. The subroutine SOLVEP calculates the new

pressure field from which the new velocity field is calculated. Boundary/virtual velocities are calculated from this new field and we're back at start. The program ends with saving the complete velocity and pressure field so it can be used to restart the calculation.

A.2 Labels

Velocity labels

In order to determine how the velocities in each cell have to be computed the velocites have a label. The various labels are given below. In case of a boundary velocity the neigbours used to extrapolate or interpolate are given. In comparisson with the previous version of `ComFlo` 6 labels have been added.

velocity label	type	$u(i,j,k)$ needs	$v(i,j,k)$ needs	$w(i,j,k)$ needs
1	**FF**	-	-	-
2	**FF**		suitable for 4th order	
3	**FF**		suitable for exact boundary handling	
11	**FI**		inflow in negative direction	
12	**IF**		inflow in positive direction	
13	**FO**		outflow in negative direction	
14	**OF**		outflow in positive direction	
21	**FB**	i-1	j-1	k-1
22	**FB**	i+1	j+1	k+1
30	**BB**	-	-	-
31	**BB**	j-1	i-1	i-1
32	**BB**	k-1	k-1	j-1
33	**BB**	j+1	i+1	i+1
34	**BB**	k+1	k+1	j+1
35	**BB**	j-1, k-1	i-1,k-1	i-1, j-1
36	**BB**	j-1, k+1	i-1,k+1	i-1, j+1
37	**BB**	j+1, k-1	i+1, k-1	i+1, j-1
38	**BB**	j+1, k+1	i+1, k+1	i+1, j+1

Table A.1: *Defintion of the velocity labels*

As with all labels the main information they contain is their relative position on the grid with respect to velocites needed in boundary conditions. Labels 2 and 3 are to indicate where to use the higher order method or the exact handling of the boundary respectively.

pressure labels

In the calculations only cells filled with fluid and boundary cells are taken into account. To make it easier to determine wether a cell contributes or not the pressures are also labeled. These labels are defined in the following way :

pressure label	description
0	obstacle cell
1	fluid cell
2	no-slip boundary cell
3	inflow cell
4	outflow cell
5	free-slip boundary cell

Table A.2: *Definition of the pressure labels*

A.3 Subroutines

The program consists of several subroutines whose functions are described below.

`AVS`	produces data files for the post-processing program `AVS`
`BC`	computes the new boundary velocities and sets the exact boundary conditions
`BCIO`	sets velocities labeled as inflow or outflow velocities with the given conditions
`BNDDEF`	defines the domain and object and calculates the apertures.
`BNDDEFIO`	defines the inflow and outflow boundaries. They can be set by adjusting the `PLABEL` to the desired value.
`COEFL`	computes coefficients of left hand side of the Poisson equation
`COEFR`	computes coefficients of right hand side of the Poisson equation
`FREESLIP`	sets `PLABEL` to 5 when a wall and/or object is free-slip. They can be set independent of each other.
`GRID`	produces a stretched grid
`INIT`	initializes velocities for new time step
`LABEL`	gives each cell u,v,w and p-labels.
`LDFLD`	loads a backup field stored in `comflo-fld`
`MATLAB`	produces data fields for post-processing with `MATLAB`. It calculates the vorticity.
`PRESIT`	computes new pressure field
`SAVEFLD`	saves a velocity and pressure field to a backup file called `comflo-fld`
`SETFLD`	initializes the velocity and pressure field at $t = 0$
`SETPAR`	reads the inputfile `comflo-in` and sets the parameters
`SOLVEP`	calls `PRESIT` and calculates the new velocity field
`TILDE`	computes a temporary vector field

A.4 Common Blocks

Variables that belong together are grouped in a common block. `ComFlo` consists of several common blocks which are presented next.

ADAM

UNN(I,J,K),VNN(I,J,K),WNN(I,J,K) are velocities used in the Adams-Bashforth method
COEF1, COEF2 are coefficients indicating which time integration is used (see comflo-in)

APERT

AX(I,J,K),AY(I,J,K),AZ(I,J,K) are the edge-apertures in x,y and z-direction respectively.
FB(I,J,K) are the volume aptertures

COEFP

DIV(I,J,K),CC(I,J,K),CXL(I,J,K),CXR(I,J,K),CYL(I,J,K),CYR(I,J,K),CZL(I,J,K), CZR(I,J,K) are coefficients used in the Poisson equation
UEX(I,J,K),VEX(I,J,K),WEX(I,J,K) are the exact boundary conditions

GRIDAR

IMAX, JMAX, KMAX number of cells in each direction
X(I),Y(J),Z(K) position of the gridlines in the three directions
DXP(I),DYP(J),DZP(K) distance between gridlines, e.g. DXP(I)=X(I)-X(I-1)
DXU(I),DYV(J),DZW(K) distance between velocities e.g. DXU(I)=0.5*(DXP(I)+DXP(I+1))

LABELS

ULABEL(I,J,K),VLABEL(I,J,K),WLABEL(I,J,K),PLABEL(I,J,K) velocity and pressure labels

NEW

HIGH, EXACTB indicate whether or not to use 4th-order and boundary handling
DIM indicates the dimension of the problem. When DIM =2 a free-slip condition is applied in the third direction to ensure 2D flow.

NUMER

EPS is the allowed error when solving the Poisson equation OMEGA, OMSTRT are respectively the current and the starting relaxation parameter used to solve the Poisson equation
ITMAX, NOM is the total number of allowed iterations C1,C2 are the starting parameters regarding the method of time integration to be used

PHYS

UN(I,J,K),VN(I,J,K),WN(I,J,K) velocities at previous timestep $t = (n-1)\delta t$ U(I,J,K),V(I,J,K),W(I,J,K) new velocities at $t = n\delta t$ P(I,J,K) pressure at $t = n\delta t$

ROTATE

VOR, VORX,VORY,VORZ are the magnitude of the vorticity vector and its three components.

SPACE

DOMAIN indicates the flow domain
XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX are the (dimensionless) sizes of the flow domain

SMALL

HX,HY,HZ are the smallest cells in each direction, used to calculate the CFL-number

STRGRD

XP,YP,ZP is the concentration point of the grid
ALFAX, ALFAY, ALFAZ is the stretching of the grid in the three directions

TIME

TMAX, T, DT are the total (dimensionless) calculation time and current time $t = n * dt$. CYCLE indicates which cycle is going on
ITERTOT contains the total number of iterations from the beginning
FRTOT, TSTART are the number of frames and when to start producing them
LDFLD indicates loading a previous field to start with

VAR

OBJECT indicates which object to be used
VAR1, VAR2, VAR3 are parameters used to define the object. They can have different purposes when used with different objects.
FS,FSOBJ free-slip of walls respectively the object
NRINTP is the number of points in each direction used to calculate the apertures
POSX,POSY,POSZ is the position of the object on the grid

A.5 Input and Output

The input file Comflo-in consists of the following parameters :

```
xmin,    xmax,    ymin,    ymax,    zmin,    zmax
0.0      0.41     0.0      0.41     0.1      1.1

imax,    jmax,    kmax     dim
30       1        90       2

tmax,    dt,      fr-tot   tstart
5.0      1E-3     10       0.0
                    .freeslip.
nu,      domain,  fdomain  fobject
1E-3     11       0        0

eps,     omega,   itmax,   coef1,   coef2
1E-4     1.7      100000   1.0      0.0

object,  radius,  angle var3
3        0.05     0         0.0

posx,    posy,    posz
0.20     0.20     0.9

px,      py,      pz,      bifstrt  avgstart
20       20       100      4.0      16.0

xp,      yp,      zp
0.0      0.0      1.1

alfax,   alfay,   alfaz
1.0      1.0      1.0

#ip      O(4)     exact    loadfld
8        1        1        0
```

- `xmin, xmax, ymin, ymax, zmin, zmax` define the dimensionless coordinated between where the domain Ω is situated.
- `tmax, dt, fr-tot` define the maximum calculation time, time step and total number of data files to be written to disk
- nu is the viscosity of the fluid
- `omega, itmax` are parameters for SOR-iteration where omega is the starting relaxation parameter and itmax the maximum number of iterations
- `domain, object` are the parameters to choose several programmed domains and objects

- `fdomain, fobject` define wether the wall and object are no-slip (0) or free-slip (1)
- `var1, var2, var3` are variables that can be used throughout the program, for example the size of objects, rotationspeed,...
- `pox, posy, posz` define the position of the object on the grid
- `alfax, alfay, alfaz` are the factors between the sizes of cells near tthe walls and those near the concentration point in the three directions respectively
- `#ip` is number of points in one direction to be used for calculating the apertures
- `O(4)` and `exact` define whether or not to use the higher order method and the exact treatment of the boundaries
- `loadfld` uses `comflo-fld` to start a new calculation with

A.6 Domain Definition with Inflow and Outflow

In order to define the geometry a function $f(x,y,z)$ is needed which has the property that $f(x,y,z) < 0$ when (x,y,z) lies in a solid part of the geometry and $f(x,y,z) > 0$ when it is situated in the fluid, e.g.

$$F(X,Y,Z) = 0.5 - SQRT((X - XP) * *2 + (Y - YP) * *2 + (Z - ZP) * *2))$$

defines a sphere filled with fluid with radius 1/2 and center (xp, yp, zp). The program makes a distinction between domain and object, so two functions are needed to describe the geometries. With the operators min and max we can combine both functions to get one function which describes the complete domain.
Next step is to show how to define an inflow or outflow part. As can be seen in earlier tables, an inflow cell is defined with a *plabel* = 3, an outflow cell has *plabel* = 4. In the subroutine `BNDDEFIO` one can choose the cells which need to get the desired property. When `BNDDEFIO` is called the cells are only labeled 0 or 1, based on their level of filling. One can change both cells into inflow or outflow cells, but it is highly recommended to have inflow and outflow cells situated next to **F**-cells. After this the procedure `LABEL` generates the velocity labels fully automatically so no corrections have to be made.
Boundary velocities can be set in `BNDDEFIO`. This time you have to know which velocity labels are concerned due to the orientation. It's also possible to produce a flow driven by a pressure difference. Then the inflow becomes a "negative" outflow.

A.7 Outputfiles and Postprocessing

Visualisation of data can give a lot of information about the fluid flow. Several programs can be used to visualise the data, we used `Matlab` when dealing with 2D flow problems and switched to `AVS` when the problem became 3D. These programs require a different approach concerning their input files so different subroutines were written to produce these files.
The number of data files to be generated can be adjusted in the file `comflo-in`. Based on the total simulation time `tmax`, the starting time `tstrt` and the number of files `np` the program generates data files on uniform time intervals between `tstart` and `tmax`.

Matlab-files

`coord.m`

This `MATLAB`-file contains the gridpoints and information about numerical quantities which don't change in time e.g. apertures and labels. This information can be helpful when debugging the generated domain.

`comflo.xxx`

These files contain the velocity, vorticity and pressure information of the entire flow domain at a certain time. `xxx` is a number starting from 100 till $100 + np$. The first frame (`comflo.100`) is the initial frame at $t = 0$ (or the starting time when the `ComFlo` is started with a backup file). Furthermore, a file `coord.m` is produced for use with `MATLAB` and contains grid coordinates. The `AVS`-versions of this file are `coordx.dat, coordy.dat, coordz.dat` and `geometry.dat`. The `ComFlo` procedure `MATLAB` can be adjusted to write the desired data to a file.

`bif.m`

This file contains the velocities and pressure at some grid point (px, py, pz). This data is written to `bif.m` starting at `t=bifstart`. When for example the x-velocity is plotted versus time in a periodic flow the period can be easily determined. When dealing with 2D-flow $py = 4$ due to expanding the grid with 3 cells in each direction. To extract the data from `bif.m` it has to be loaded into `MATLAB` followed by

```
u=bifdata(:,1); v=bifdata(:,2); w=bifdata(:,3);
p=bifdata(:,4); t=bifdata(:,5);
```

`comflo-avg.m`

It contains the averaged velocity and pressure field in time. Averaging is started at `t=avg-start`. Furthermore, it has the same structure as the `comflo.xxx` files.

AVS-files

coordx.dat/coordy.dat/coordz.dat/geometry.dat

These files are the AVS-equivalent of the MATLAB-file `coord.m` described above. They contain the grid information and the geometry information (the latter based on the volume-apertures).

Backup-file

`comflo-fld`

contains the complete velocity and pressure field at some time t and is used for back-up and/or as a start-up file when further calculation is needed but **`tmax`** has been reached.

Bibliography

[1] P.M.Gresho,"Some current CFD issues relevant to the incompressible Navier-Stokes equation," *Comp. meth. in applied Mechanics and Engineering*, **87**, 201-252 (1991)

[2] J.Gerrits,"Fluid flow in 3D Complex geometries - a cartesian grid approach," *Master's thesis* (1996)

[3] P.Anagnostopoulos,G.Iliadis and S.Richardson , "Numerical study of the blockage effects on viscous flow past a circular cylinder,"*Int. J. Num. Methods in Fluids*, **22**, 1061-1074 (1996)

[4] S.Biringen and E.M.Saiki, "Numerical Simulation of a Cylinder in Uniform Flow: Application of a Virtual Boundary Method," *J.of Comp.Physics*, **123**, 450-465 (1996)

[5] M.Coutanceau and R.Bouard *J.Fluid Mech.*, **79** (1979)

[6] P.M.Gresho,R.Chan,C.Upson and R.Lee *Int.J.Numer.Methods Fluids*, **4** (1994)

[7] H.Hoogstraten, "Stromingsleer," *Lecture notes* (1992)

[8] Results from a DFG workshop "Flow Simulation on High Performance Computers"(March 1996, Heidelberg)

[9] A.E.P.Veldman, "Numerieke Stromingsleer," *Lecture notes* (1994)

[10] R.W.C.P.Verstappen and A.E.P.Veldman, "Direct Numerical Simulations of Turbulence at Lower Costs", *to appear in J.Eng.Meth*,(August 1997)