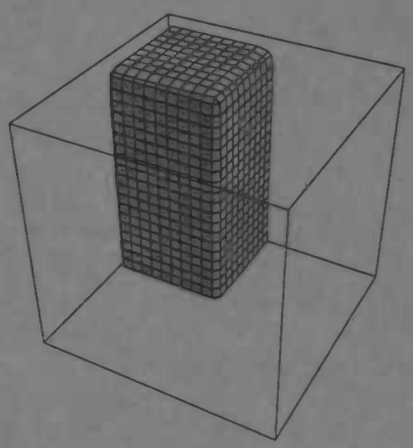
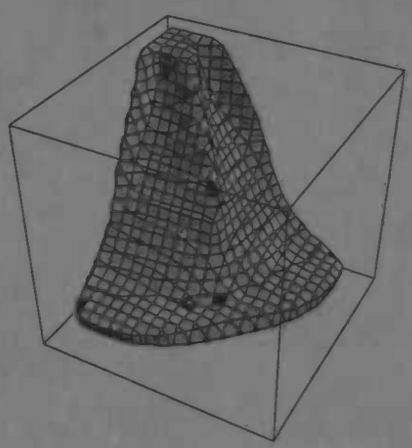
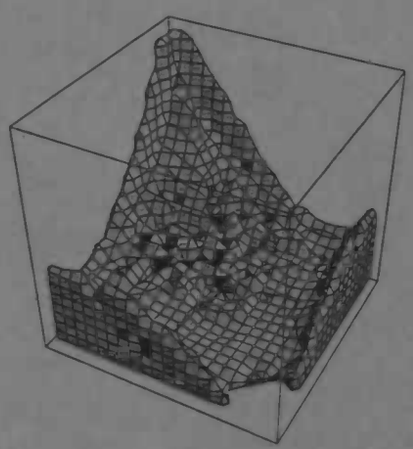
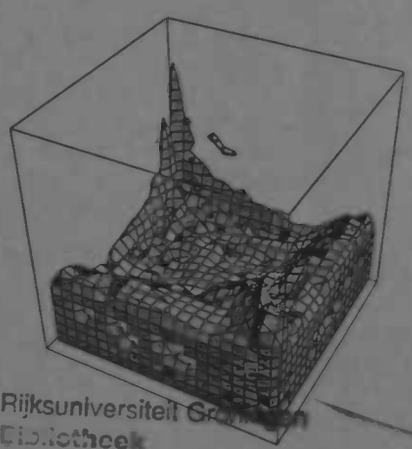
Three-Dimensional Liquid Sloshing in Complex Geometries

Jeroen Gerrits









Department of Mathematics

RuG

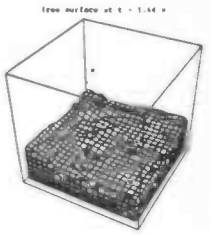
Master's thesis

Three-Dimensional Liquid Sloshing in Complex Geometries

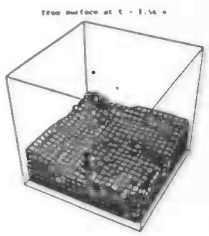
Jeroen Gerrits

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

August 1996



Contents



Chapter 1

Introduction

Free-surface flow or liquid sloshing is one of the most important applications of computational fluid dynamics (CFD). Just think of the motion of fluid in satellites, spacecraft and oil tankers or, more familiar, water waves at the beach and a cup of tea shaking in your hand. Simulating free-surface flow, i.e. solving the Navier-Stokes equations numerically, is a complex task since the position of the free surface varies in time making it part of the problem to be solved.

In the history of CFD lots of computer programs have been developed capable of simulating liquid sloshing in specific situations, for example

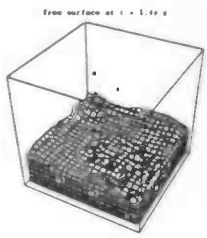
- Two-dimensional dambreak in a rectangular container [16]. In this reference the container is covered with a Cartesian grid and the fluid is moved using the VOF-method allowing arbitrary free-surface shapes.
- Liquid motion in a two-dimensional rectangular container [13]. The container is subjected to a forced pitching oscillation. In this reference the governing equations are solved using the finite element method. Hereto the region inside the container occupied by fluid is covered with an unstructured grid (a process that is repeated every time step).
- Axisymmetric sloshing in a cylindrical container [17]. In this paper the Navier-Stokes equations are rewritten in cylindrical co-ordinates.
- Three-dimensional sloshing in a rotating sphere using spherical co-ordinates [3]. The free-surface height is assumed to be a single-valued function of space. This reference is one of the few describing 3-D liquid sloshing in complex geometries.

Most of the available solution algorithms make the use of body-fitted co-ordinates. Hence all of the references above and lots of others are quite restrictive in the sense that only one type of problem can be solved. In this report we will discuss liquid sloshing (with arbitrary free-surface shapes) in arbitrary three-dimensional geometries resulting in a computer program capable of simulating a variety of flow problems.

This report is an extension of previous work by the author concerning fluid flow (without free surfaces) in three-dimensional complex geometries [5]. Herein we showed how the Navier-Stokes equations are discretized on a Cartesian grid, also for non-rectangular geometries. This was accomplished by introducing a sophisticated labeling method that

avoids stability problems. Further we used so called apertures for representing the complex geometry on the rectangular grid. Also in the current project the governing equations are discretized on a Cartesian grid. Hereby we use the labeling in [5] as a starting point. For tracing the free surface new labels are introduced and the VOF-method is implemented.

In chapter 2 of this report the mathematical model is formulated describing the governing equations and the boundary conditions at the solid wall and the free surface. The numerical model is derived in chapter 3. In this chapter we will explain the use of apertures and the labeling method. Further we discuss the solution algorithm for the discrete Navier-Stokes equations and the incremental procedure for the volume of fluid convection. Based on the numerical model a computer program COMFLO-SLOSH has been written of which a detailed description is given in appendix A. Some results of the program are discussed in chapter 4 showing the validation and capabilities of COMFLO-SLOSH. Finally in chapter 5 some conclusions are drawn.



Chapter 2

Mathematical Model

2.1 Navier-Stokes Equations

In this report we study the flow of a viscous fluid (with kinematic viscosity equal to ν) inside an arbitrary complex three-dimensional domain Ω. To allow free-surface flow the domain is partially filled with a liquid. We assume the fluid is incompressible and normalize the density to unity ($\rho \equiv 1$). Then the motion of the fluid is governed by the unsteady Navier-Stokes equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{2.1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + F_x + f_x, \tag{2.2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + F_y + f_y, \tag{2.3}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{\partial p}{\partial z} + \nu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + F_z + f_z. \tag{2.4}$$

In these equations p is the pressure and u, v and w are the velocity components in x-, y- and z-direction respectively. Further $\boldsymbol{F} = (F_x, F_y, F_z)^T$ is an external body force (e.g. gravity) and $\boldsymbol{f} = (f_x, f_y, f_z)^T$ is a virtual body force due to motion of the domain Ω (see below). Equation (2.1) expresses conservation of mass, while (2.2), (2.3) and (2.4) express conservation of momentum in x-, y- and z-direction. In the momentum equations the convective terms (first-order derivatives) and the diffusive terms (second-order derivatives) can be recognized.

If we write

$$\boldsymbol{u} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

the Navier-Stokes equations can be written in short (vector) form, using the divergence (div) and gradient (grad) operators:

$$\text{div } \boldsymbol{u} = 0,$$

$$\frac{\partial u}{\partial t} + (u \cdot \text{grad})\, u = -\text{ grad } p + \nu \text{ div grad } u + \boldsymbol{F} + \boldsymbol{f},$$

or equivalent, using that the velocity field is divergence-free,

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

$$\frac{\partial u}{\partial t} + \text{div}\left(uu^T\right) = -\text{ grad } p + \nu \text{ div grad } u + \boldsymbol{F} + \boldsymbol{f}. \tag{2.6}$$

The Cartesian reference frame $\mathcal{O}xyz$ is taken fixed to the domain Ω which may be moving with respect to an inertial reference frame $\overline{\mathcal{O}}\overline{xyz}$. Hereto a virtual body force $\boldsymbol{f}$ is introduced [17] in equation (2.6):

$$\boldsymbol{f} = -\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} - \boldsymbol{\omega} \times (\boldsymbol{\omega} \times r) - \frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} \times r - 2\boldsymbol{\omega} \times u, \tag{2.7}$$

where $\boldsymbol{q}$ is the velocity of the point $\mathcal{O}$ (with respect to the inertial reference frame), $\boldsymbol{\omega}$ is the angular velocity of the flow domain (with respect to $\mathcal{O}xyz$), and r is the radius vector pointing away from $\mathcal{O}$. See figure 2.1 for an illustration of the above-mentioned.

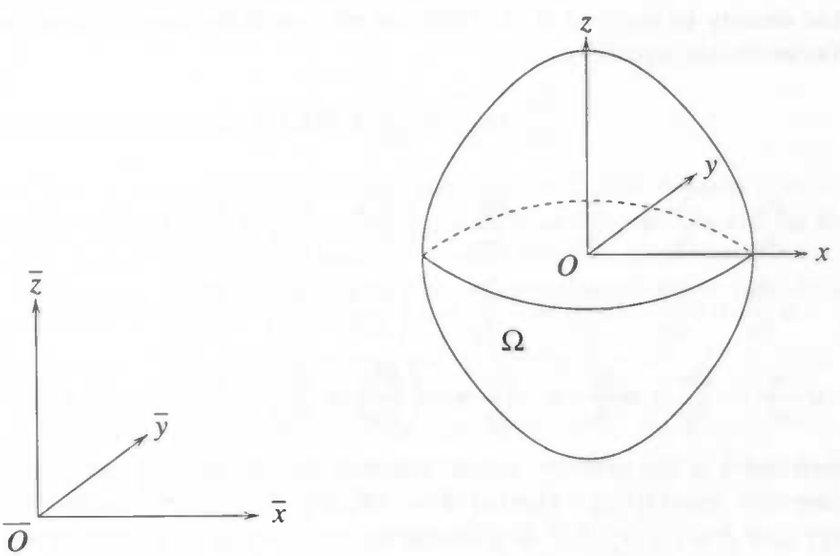
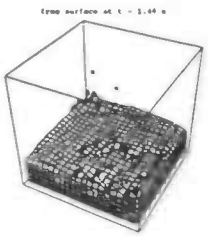


Figure 2.1: *Flow domain Ω and reference frames.*

2.2 Boundary Conditions

We will denote the region inside Ω that is occupied by fluid with Ω_f. This means that in every point of Ω_f the fluid must satisfy equations (2.5) and (2.6). We mention that Ω_f itself is part of the problem since it varies in time ($\Omega_f = \Omega_f(t)$). On the boundary of Ω_f boundary conditions are needed. Since the domain is partially filled we have to make a distinction between the solid boundary $\partial\Omega$ and the free surface $\partial\Omega_f$.



2.2.1 Solid Boundary

At the solid boundary of the flow domain the velocity must satisfy the no-slip conditions for a viscous fluid:

$$u = 0 \text{ on } \partial\Omega.$$

These guarantee that the normal component of the velocity is equal to zero (the fluid can not flow through the boundary) and that the tangential component of the velocity is equal to zero (the fluid sticks to the wall because of the viscosity).

2.2.2 Free Surface

At the free surface both the velocity and the pressure need boundary conditions. If we neglect the surface tension these are [18]:

$$\frac{\partial u_n}{\partial t} + \frac{\partial u_t}{\partial n} = 0 \text{ on } \partial\Omega_f, \tag{2.8}$$

$$-p + 2\mu\frac{\partial u_n}{\partial n} = -p_0 \text{ on } \partial\Omega_f, \tag{2.9}$$

where $u_n = u \cdot n$ is the velocity normal to the free surface (pointing away from the fluid) and $u_t = u \cdot t$ the velocity in tangential direction. Further μ is the coefficient of dynamic viscosity which is related to the kinematic viscosity by $\nu = \frac{\mu}{\rho}$ and p_0 is the atmospheric pressure. Equations (2.8) and (2.9) express the continuity of the tangential and normal stresses respectively at the free surface.

In this report we simplify the free-surface condition (2.9) to

$$p = p_0 \text{ on } \partial\Omega_f. \tag{2.10}$$

This assumption enables a very direct application of the pressure boundary condition in the numerical model as will be explained in the next chapter. The free-surface condition (2.8) is simplified as well. In two dimensions usually $\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} = 0$ is prescribed, which is more easy to compute on a Cartesian grid, instead of (2.8). This ensures that (2.8) is correct at least for horizontal and vertical surfaces. In three dimensions we use a similar simplification. We demand that (2.8) is correct for surfaces normal to the x-, y- and z-direction which results in three equations:

$$\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} = 0, \quad \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = 0, \quad \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} = 0. \tag{2.11}$$

In the next chapter we will explain how and where these equations are discretized.

Chapter 3

Numerical Model

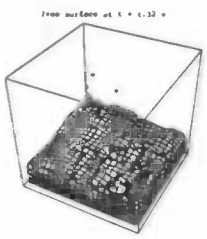
In this chapter the numerical model is deduced. Although this thesis deals with three-dimensional geometries we will explain most of the numerical method in two dimensions since this improves readability. Furthermore an extension to 3-D is in general straightforward. A part of the numerical model is exactly the same as in the author's previous master's thesis [5]. Therefore in some situations we refer to this thesis although readability remains the highest priority.

3.1 Apertures and VOF-Function

As mentioned in the introduction we cover the (complex) flow domain Ω with a three-dimensional Cartesian grid. The computational cells of the grid are cut by the curved boundary of Ω in a wide variety. Hence cells with different characters originate. This difference in character is incorporated in the numerical method by introducing *apertures* [15].

In the centre of every computational cell a *volume-aperture* F^b is defined which indicates the fraction of the cell-volume that is open to flow (so $0 \leq F^b \leq 1$). Analogously we define in the centre of every cell-face an *edge-aperture* A^x (in x-direction) or A^y (in y-direction). Of course in 3-D a third edge-aperture A^z is introduced. We note that edge-apertures contain information that is one dimension lower than the information given by volume-apertures (e.g. three-dimensional edge-apertures indicate the fraction of a cell-*surface* open to flow, which is two-dimensional information). The use of apertures is illustrated in figure 3.1 The apertures F^b, A^x and A^y (and A^z in 3-D) are used to discretize (2.5) and (2.6) near the boundary and to compute the boundary velocities (see sections 3.4 and 3.6 in [5]).

Since Ω is partially filled with fluid we need additional information to be able to trace the free surface. This is accomplished by introducing one more volume-aperture F^s (also defined in the centre of every cell) better known as a volume of fluid (VOF) function [8]. This (discrete) function indicates the fraction of a computational cell that is occupied by fluid. As a consequence we have $F^s \leq F^b$ throughout the computational grid. An important difference between the volume-aperture F^b and the VOF function F^s is the time-dependency. As F^b can be computed once before the start of the time integration, the VOF function has to be adjusted every single time step. This procedure



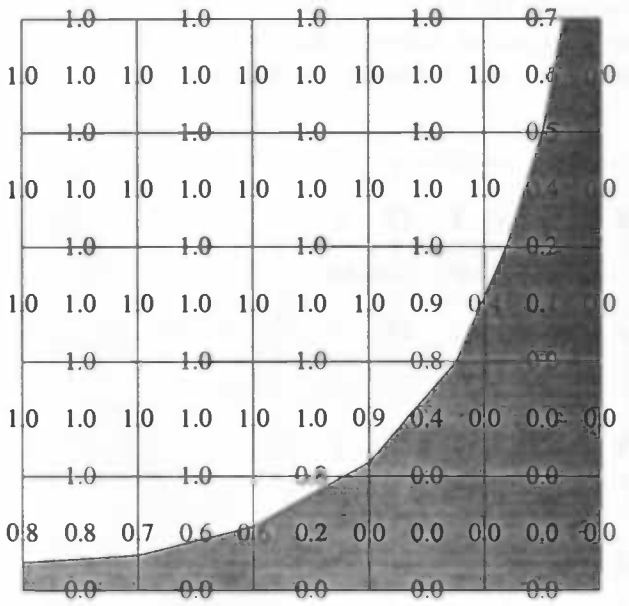


Figure 3.1: *Example of a (discrete) aperture-field (rounded to one decimal place).*

is explained in section 3.6.2.

3.2 Labeling

The continuity equation (2.5) and momentum equations (2.6) are discretized on a staggered grid, i.e. the continuity equation is discretized in cell-centres while the velocity components are solved from the momentum equations in the centre of cell-faces.

In section 3.2 of [5] we examined two different methods to apply the boundary conditions. In the first (conventional) method all the boundary velocities (i.e. velocities solved from the solid boundary conditions) were located outside the flow domain, while the second method allowed boundary velocities inside the flow domain. The latter turned out to have some advantages compared to the former. For example we have shown that the second method did not cause stability problems in contrary to the first method. This is the reason why we choose in this report the second method to set up the numerical model (this method is explained in the section below). Further some new labels are introduced compared to [5] since the current model allows free-surface flow (section 3.2.2).

3.2.1 Geometry Labels

Prior to the time integration the cells are labeled on the basis of the geometry only. First all the cells with $F^b \geq \frac{1}{2}$ are labeled as an **F**(low)-cell. This means that at least half of an **F**-cell is open to flow. A computational cell adjacent to an **F**-cell but with $F^b < \frac{1}{2}$ will be called a **B**(oundary)-cell. Finally all the remaining cells are labeled as an **O**(utside)-cell. These cells have no significant role in the numerical model.

Based on the cell labeling the velocities are labeled by a combination of two letters; for example a velocity component between an **F**- and a **B**-cell will be called an **FB**-velocity. The geometry labels are illustrated in figure 3.2.

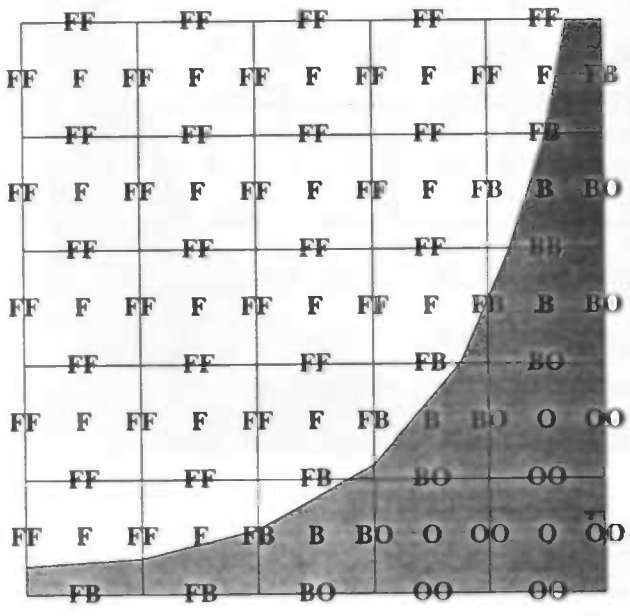


Figure 3.2: *Cell and velocity labels based on geometry only.*

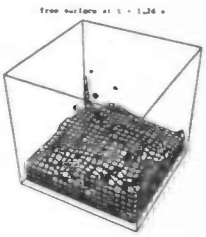
3.2.2 Free-Surface Labels

The geometry labels are the basis of a more comprehensive labeling to be able to trace the free surface. Since the **B**- and **O**-cells consist of more than 50% solid boundary these labels remain unchanged during the rest of the computational process. Thus, every time step, only the **F**-cells have to be examined on the occurence of liquid. If a cell contains no fluid, i.e. $F^s = 0$, then the cell is called an **E**(mpty)-cell. All the cells adjacent to an **E**-cell with $F^s > 0$ are labeled as an **S**(urface)-cell.[1] The remaining **F**(low)-cells now will be called **F**(luid)-cells. Figure 3.3 illustrates the free-surface labels in combination with the geometry labels.

Based on the cell labels we decide the following:

- In **E**-, **B**- and **O**-cells no pressure is computed or prescribed since these cells contain (almost) no fluid.
- In **S**-cells the pressure is set equal to the atmospheric pressure according to the free-surface condition (2.10). Hereby an error is introduced since the centre of a computational cell (where the pressure is prescribed) need not be on the actual free surface. A more accurate solution would be to compute the pressure in an

[1] In the computer program a cell is labeled an **E**-cell if $F^s < \varepsilon$, with ε a small constant, to take rounding errors into account. Analogously an **S**-cell satisfies $F^s \geq \varepsilon$.



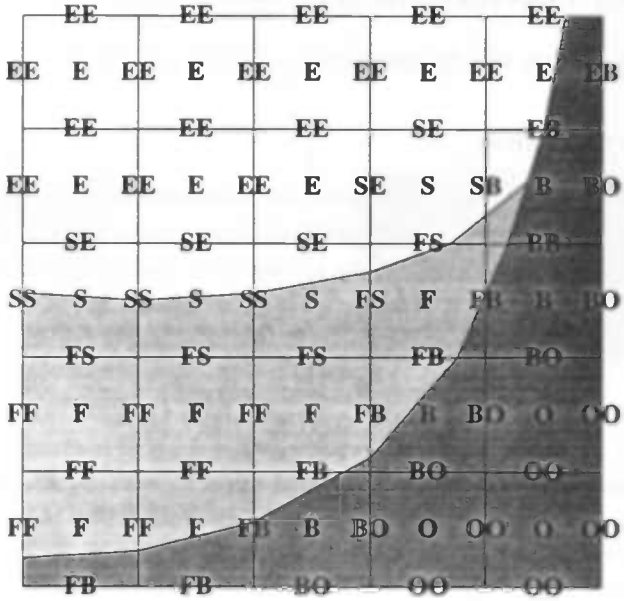


Figure 3.3: *Free-surface labels and geometry labels.*

S-cell by interpolation or extrapolation with a pressure inside the fluid. However an assignment $p = p_0$ in all the **S**-cells is much easier to implement. Moreover the results turn out to be sufficiently precise.

- In all the **F**-cells the pressure is solved from the continuity equation (2.5) or to be more precise from the pressure Poisson equation (see section 3.6).

The velocities are labeled by a combination of two letters as in the previous section. A priori 15 different combinations can be made. However an **O**-cell can not be adjacent to an **F**-, **S**- or **E**-cell and an **F**-cell can not be next to an **E**-cell. Further **BO**- and **OO**-velocities are ignored since these are not needed in the finite-difference formulas representing the first- and second-order derivatives in the Navier-Stokes equations. Thus only the following 9 combinations remain for further study:

- **FF**, **FS**, **SS**, **SE** and **EE**. Based on only the geometry these are the **FF**-velocities.
- **FB**, **SB** and **EB**. These are the **FB**-velocities if only the geometry labels are considered.
- **BB**.

Since in both the **F**-cells and the **S**-cells a pressure is defined a pressure gradient can be computed between two **F**-cells, between an **F**- and an **S**-cell and between two **S**-cells. Hence all the **FF**-, **FS**- and **SS**-velocities are solved from the momentum equations. In the remainder of this report we will call these velocities *momentum velocities*. We will call the **SE**- and **EE**-velocities *free-surface velocities* and deal with these in section 3.4. The rest of the velocities (**FB**, **SB**, **EB** and **BB**) are the *boundary velocities* and will be discussed in section 3.5.

3.3 Discretized Navier-Stokes Equations

In this section we will discuss the discretization of (2.5) and (2.6).

3.3.1 Time Discretization

In [5] we have shown that the labeling method as discussed in the previous section does not cause stability problems. Hence the Navier-Stokes equations are discretized explicit in time. If we use the most elementary time integration method *forward Euler* we get:

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

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

where δt is the time step, u^n the velocity field at time $t_n = n \cdot \delta t$ and p^{n+1} the pressure distribution at time t_{n+1}. The pressure gradient is discretized at the 'new' time level to ensure that the 'new' velocity field from (3.2) is divergence-free.

3.3.2 Spatial Discretization

The time-discretized equations (3.1) and (3.2) have to be discretized in space as well. This procedure is exactly the same as in [5]. Hence we only give here the main features of the spatial discretization.

For the spatial discretization conservation cells are used. In these cells both conservation of mass and momentum is required. The conservation cell for the continuity equation is identical to a computational cell, while a conservation cell for the momentum equations consists of two computational cells (see figures 3.4 and 3.5 respectively). Since the continuity equation expresses that the net amount of mass flowing through the boundary of a computational cell must be equal to zero the continuity equation is discretized based on apertures. This way we take into account that no mass can flow through the solid boundary.

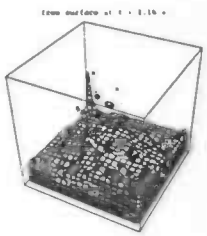
The same applies to the convective terms of the momentum equations. These indicate the increase of momentum in a conservation cell because of transportation of momentum through the boundary of that cell. Since momentum can not flow through $\partial\Omega$ the convective terms are discretized based on apertures too. The diffusive terms, the pressure gradient and the body forces indicate the increase of momentum due to stresses rather than transportation. Hence these terms in the momentum equations are *not* discretized with apertures.

The above-mentioned results in the following discretized Navier-Stokes equations:

$$D_h^a u^{n+1} = 0, \tag{3.3}$$

$$\frac{u^{n+1} - u^n}{\delta t} + G_h p^{n+1} = \nu D_h G_h u^n - D_h^a\left(u^n u^{n^T}\right) + \boldsymbol{F}^n + \boldsymbol{f}^n. \tag{3.4}$$

Here D_h and G_h are the discrete versions of the divergence and gradient operators. D_h^a indicates that the divergence operator is discretized with apertures. The solution of equations (3.3) and (3.4) is discussed in section 3.6.



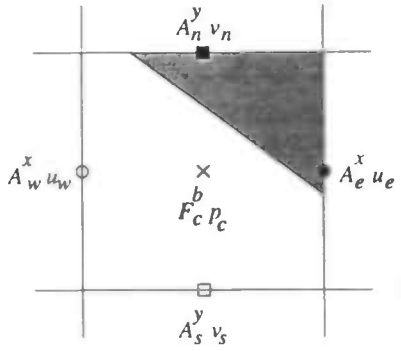


Figure 3.4: *Conservation cell for the continuity equation.*

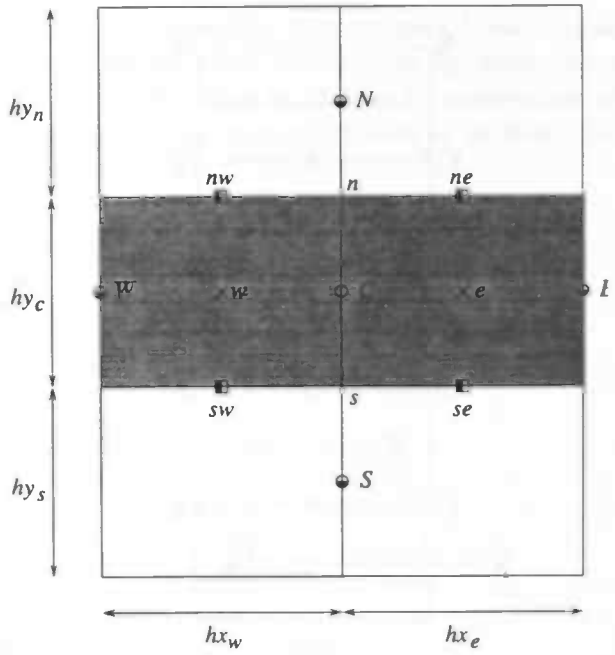


Figure 3.5: *Conservation cell for the momentum equation in x-direction.*

3.4 Free-Surface Velocities

The **FF**-, **FS**- and **SS**-velocities are solved from the discrete momentum equations (3.4). These equations contain first- and second-order spatial derivatives which means that for computing the momentum velocities the free-surface velocities **SE** and **EE** are needed. More precisely: the free-surface velocities adjacent to the momentum velocities are needed. In this section we will explain how these velocities are computed.

3.4.1 EE-velocities

We already mentioned that **EE**-velocities have to be computed if they appear in a discrete momentum equation. Hereto the free-surface condition (2.11), which consists of three equations, is discretized. We will illustrate this by examining an **EE**-velocity in x-direction and the neighbouring momentum velocities (see figure 3.6). Note that the momentum velocities can only be **SS**-velocities since an **F**-cell can not be next to an **E**-cell.

Since we consider a velocity in x-direction the first equation in (2.11) is redundant which leaves the following two:

$$\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = 0, \quad \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} = 0. \tag{3.5}$$

For each configuration or combination of configurations in figure 3.6 we have to decide in what way (3.5) is applied. First we mention that an **EE**-velocity without **SS**-neighbours

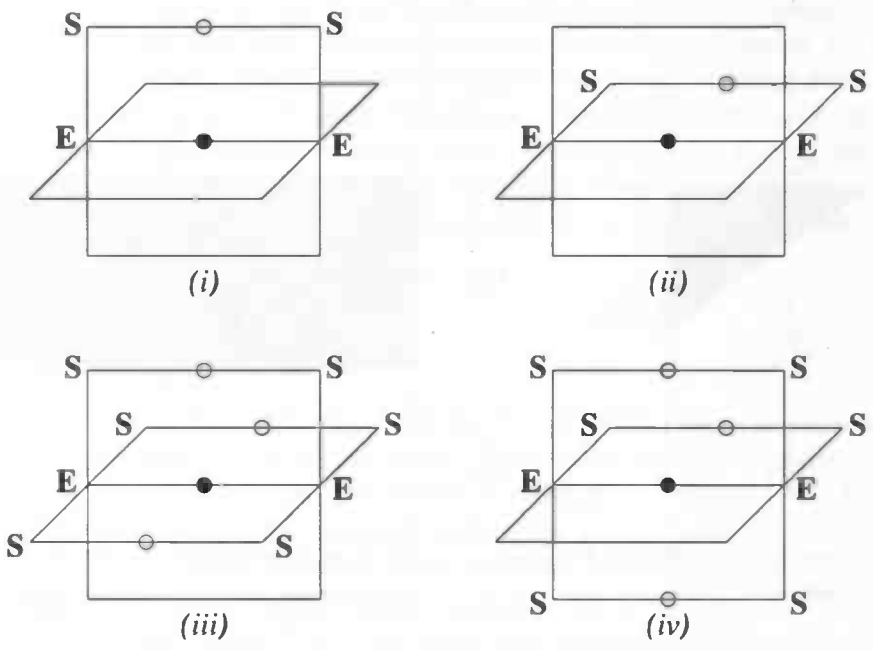


Figure 3.6: *Momentum velocities* ∘ *near an* **EE***-velocity* •.

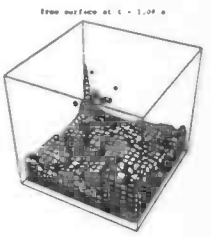
the value 0 is assigned to. Roughly spoken this is the part of the flow domain containing no fluid. In configuration *(i)* (one **SS**-neighbour in z-direction) $\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = 0$ is discretized as follows

$$\frac{u_T - u_B}{h_z} + \frac{w_R - w_L}{h_x} = 0,$$

where h_x and h_z are the mesh sizes in x- and z-direction (for a uniform mesh), u_T is the **SS**-velocity, u_B is the required **EE**-velocity and w_L and w_R are the **SE**-velocities at the left and the right. So in order to compute an **EE**-velocity the **SE**-velocities are required (see section 3.4.2). In configuration *(ii)* of course $\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} = 0$ is discretized since the **SE**-velocities now are velocities in y-direction. If a combination of the first two configurations appears (either one velocity in y-direction and one velocity in z-direction or two velocities in y- or z-direction) then the above-mentioned procedure is executed twice and an average is taken of the two resulting **EE**-velocities. In configuration *(iii)* the **EE**-velocity is surrounded by three momentum velocities. Because of symmetry reasons the free-surface velocity is solved from $\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = 0$. Similarly $\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} = 0$ is prescribed in configuration *(iv)*. Finally if the free-surface velocity is enclosed between four momentum velocities then each of the equations in (3.5) is discretized twice and the four **EE**-velocities are averaged to produce a final **EE**-velocity.

3.4.2 SE-velocities

The free-surface velocities with label **SE** have to be computed if they appear in the discrete Navier-Stokes equations. Furthermore some **SE**-velocities are used to be able to compute certain **EE**-velocities (see the previous section). This means there are a lot



of possible configurations surrounding an **SE**-velocity leading to a complex branching in the computer program. Hence we decide to compute every **SE**-velocity by discretizing the continuity equation in the corresponding **S**-cell. Thus we demand conservation of mass without discretizing the pressure Poisson equation (remember that in an **S**-cell the pressure is set equal to the atmospheric pressure).

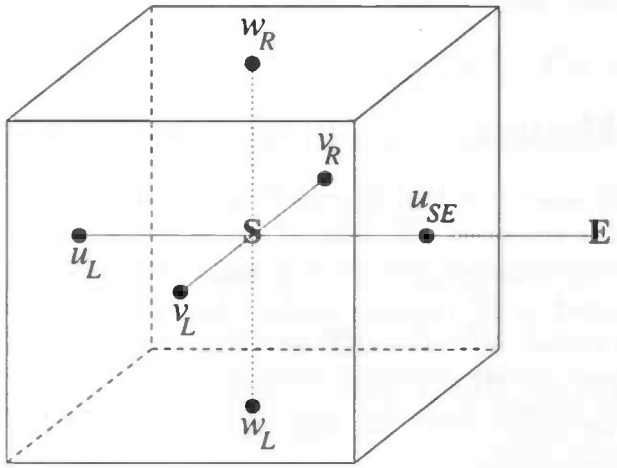


Figure 3.7: *Computation of an* **SE***-velocity in* x*-direction.*

In the **S**-cell illustrated in figure 3.7 the continuity equation is discretized as:

$$\frac{u_{SE} - u_L}{h_x} + \frac{v_R - v_L}{h_y} + \frac{w_R - w_L}{h_z} = 0, \tag{3.6}$$

where h_x, h_y and h_z are the mesh sizes in x-, y- and z-direction and u_{SE} is the required **SE**-velocity. The free-surface velocity can be solved from (3.6) if the other five velocities (u_L, v_L, v_R, w_L and w_R) have a meaningful value. These five velocities are labeled either **FS**, **SS**, **SB** or **SE**. The former three labels give no difficulties since these are computed from the momentum equations (**FS** and **SS**) or the boundary conditions (**SB**). However if one (or more) of the five velocities is an **SE**-velocity itself then u_{SE} can not be solved from (3.6) anymore. In such a situation the following two distinctions are made:

- u_L is an **SE**-velocity. In this case u_{SE} is set equal to zero.
- u_L is not an **SE**-velocity. Now u_{SE} is solved from $\frac{\partial u}{\partial x} = 0$, i.e. u_{SE} is taken equal to u_L.

Free-surface velocities in the other spatial directions are treated in a similar way. At this stage all the free-surface velocities have been accounted for.

3.5 Boundary Velocities

Only the boundary velocities **FB**, **SB**, **EB** and **BB** have not been discussed yet. In [5] we have encountered the boundary velocities **FB** and **BB** already. In the current project

an **F**-cell may be changed into an **S**- or **E**-cell which explains the occurence of the new boundary velocities **SB** and **EB**. Nevertheless these boundary velocities are treated as if they were **FB**-velocities. This is no restriction since an **S**-cell contains fluid; hence the no-slip condition can be applied safely. Furthermore most of the **EB**-velocities are not used elsewhere in the computation. Hence these velocities are set to 0 if the boundary conditions are applied. This means that by computing all of the boundary velocities no extra error is introduced. For a detailed discussion of the boundary velocities we refer to section 3.6 in [5].

3.6 Solution Method

In this section we will show how the discrete Navier-Stokes equations are solved and which algorithm is used to 'move' the fluid. First some notation is introduced. The set of points in the computational grid where a momentum velocity (**FF**, **FS** and **SS**) is defined will be denoted by Ω_h^f (roughly spoken the points inside the fluid). Further Ω_h^s contains the free-surface velocities (**SE** and **EE**) and Ω_h^b contains the boundary velocities (**FB**, **SB**, **EB** and **BB**). Finally we define $\Omega_h = \Omega_h^f \cup \Omega_h^s \cup \Omega_h^b$ as the entire computational grid (see figure 3.8). We note that the sets Ω_h^f and Ω_h^s vary in time whereas Ω_h^b is time-independent.

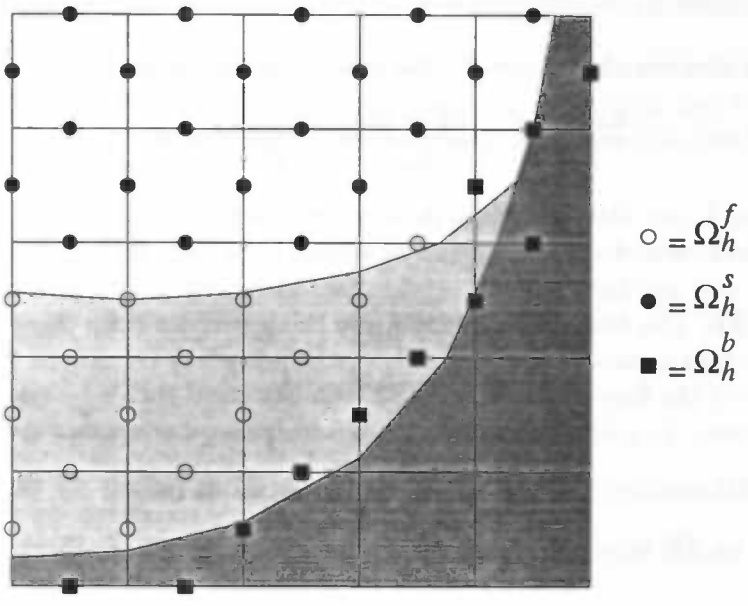
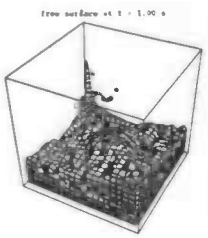


Figure 3.8: *Momentum, free-surface and boundary velocities.*

3.6.1 Solution of the Discrete Navier-Stokes Equations

We assume an initial velocity field u^n is given on Ω_h (for $n = 0$ we simply take $u^0 = 0$). Our goal is to compute a new velocity field u^{n+1} and a new pressure distribution p^{n+1}



from the discrete Navier-Stokes equations (3.3) and (3.4).

First in all the **S**-cells the pressure is set equal to the atmospheric pressure

$$p = p_0 \text{ in } \mathbf{S}\text{-cells},$$

and a temporary vector field $\tilde{u}$ is created on Ω_h^f by integrating the momentum equations (3.4) without the pressure gradient, i.e.

$$\tilde{u} = u^n + \delta t \left\{ \nu D_h G_h u^n - D_h^a \left(u^n u^{n^T} \right) + \boldsymbol{F}^n + \boldsymbol{f}^n \right\} \text{ on } \Omega_h^f. \tag{3.7}$$

Note that $\tilde{u}$ is *not* a velocity field. At this stage u^{n+1} on Ω_h^f and p^{n+1} in all the **F**-cells have to be solved from

$$D_h^a u^{n+1} = 0 \text{ in } \mathbf{F}\text{-cells}, \tag{3.8}$$

$$u^{n+1} + \delta t \; G_h p^{n+1} = \tilde{u} \text{ on } \Omega_h^f. \tag{3.9}$$

The operator D_h^a works on unknown velocities in Ω_h^f and Ω_h^b, i.e. on momentum and boundary velocities. Hence (3.9) can not be substituted directly into (3.8) since then the pressure in **B**-cells would be needed; remember that the continuity equation is discretized in **F**-cells while **FB**-velocities are not solved from the momentum equations [18]. This problem does not occur at the free surface because **FS**-velocities are solved from the momentum equations. To avoid this problem of the missing boundary condition for the pressure we write

$$D_h^a = D_h^{a,f} + D_h^{a,b},$$

where $D_h^{a,f}$ and $D_h^{a,b}$ operate on velocities in Ω_h^f and Ω_h^b respectively. Yet the problem is not solved because we can not compute the boundary velocities at the new time level since therefore we would need the entire velocity field at the new time level. That is why these velocities are set equal to the boundary velocities at the old time level, i.e. $u^{n+1} = u^n$ on Ω_h^b. This procedure causes an error $\mathcal{O}(\delta t)$ which already has been introduced by the time integration method forward Euler. Then it remains to solve

$$D_h^{a,f} u^{n+1} = -D_h^{a,b} u^n \text{ in } \mathbf{F}\text{-cells}, \tag{3.10}$$

$$u^{n+1} + \delta t \; G_h p^{n+1} = \tilde{u} \text{ on } \Omega_h^f. \tag{3.11}$$

Now it is possible to substitute (3.11) into (3.10) resulting in the *pressure Poisson equation*

$$D_h^{a,f} G_h p^{n+1} = \frac{1}{\delta t} \left(D_h^{a,b} u^n + D_h^{a,f} \tilde{u} \right) \text{ in } \mathbf{F}\text{-cells}, \tag{3.12}$$

which has to be solved in every **F**-cell in order to produce a new pressure distribution p^{n+1}. The solution of (3.12) is then used to compute the new momentum velocities

$$u^{n+1} = \tilde{u} - \delta t \; G_h p^{n+1} \text{ on } \Omega_h^f. \tag{3.13}$$

Finally the free-surface velocities and boundary velocities are adjusted (this has been explained in sections 3.4 and 3.5) such that the new velocity field u^{n+1} has been computed on the entire grid Ω_h.

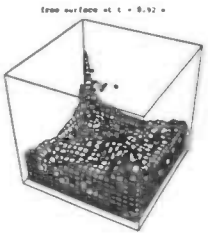
3.6.2 Volume of Fluid Convection

Once a new velocity field has been computed the position of the fluid has to be changed based on this velocity field. Hereto the donor-acceptor method [8] has been implemented and slightly adjusted to account for the complex geometries.

In the current labeling method fluid can flow from an **F**- or **S**-cell towards an **F**-, **S**- or **E**-cell. Hence at every cell-face with label **FF**, **FS**, **SS** or **SE** a flux is computed indicating the amount of fluid flowing from the donor to the acceptor cell during one time step. Hereby the following considerations have to be made:

- Fluid can not flow out of an **E**-cell since it is empty ($F^s = 0$).
- No more fluid can flow out of the donor cell than the donor cell contains. So the maximum amount of fluid that can be transported is $F^s h_x h_y h_z$.
- An acceptor cell can not accept more fluid than the void volume in the acceptor cell, i.e. $\left(F^b - F^s\right) h_x h_y h_z$.
- The edge-apertures are used for computing the fluxes because part of the cell-face may be solid boundary. Of course through this part of the cell-face no fluid can flow.
- Due to rounding errors F^s can become less than zero or greater than F^b. Hereto these values of F^s are reset at the end of the volume of fluid convection.

After the volume of fluid convection the cells and velocities are relabeled based on the new volume of fluid distribution. At this stage one time cycle has been completed and the entire process is repeated until the maximum simulation time has been reached.



Chapter 4

Results

A computer program (COMFLO-SLOSH) has been written based on the numerical model presented in chapter 3. A description of this program can be found in appendix A. In this chapter some results of COMFLO-SLOSH will be discussed.

4.1 A Rotating Cylinder

As a first test we consider a cylinder with both diameter and height equal to unity. The Cartesian origin is taken in the center of the cylinder's axis. The bottom half of the cylinder is filled with liquid. Initially the fluid is in rest but beginning at time $t = 0$ the cylinder is rotated around the central axis (z-axis) with a constant angular velocity of ω radians per second causing the fluid to move away from the axis. The flow reaches a steady state when the rotational force balances the gravitational force g, i.e.

$$\frac{1}{2}\omega^2 r^2 = gh,$$

where $r = \sqrt{x^2 + y^2}$ is the distance of a point in the fluid to the axis and h is the relative free-surface height in that point (compared to the free-surface height in the origin). This shows that the free-surface height is parabolic with respect to the radius. The height of the free surface in the origin follows from the total amount of fluid. At $t = 0$ half of the cylinder is filled with liquid i.e. a volume of $\frac{\pi}{8}$. This means that also at $t = t_{\max}$, when the flow has reached a steady state, the total amount of fluid must be equal to $\frac{\pi}{8}$. Using the parabolic profile it follows that the free-surface height in the origin is $\frac{1}{2} - \frac{\omega^2}{16g}$. Further the flow should be axisymmetric providing another easy check on validation of the computer program.

For $\omega = 5$ and $g = 10$ a calculation has been carried out. The Reynolds number $\frac{UL}{\nu}$ based on the velocity U of the cylinder's casing and the diameter L of the cylinder was set to 250. The Cartesian grid consisted of 40 grid points in every spatial direction. The flow turned out to be stationary after approximately two seconds ($t_{\max} = 2$). In figures 4.1 and 4.2 the initial free-surface configuration and the steady free-surface configuration are plotted respectively, showing clearly the axisymmetry. For comparison with theory the computed free-surface configuration is plotted in the symmetry plane in figure 4.3 together with the theoretical solution discussed above. It shows that the simulation

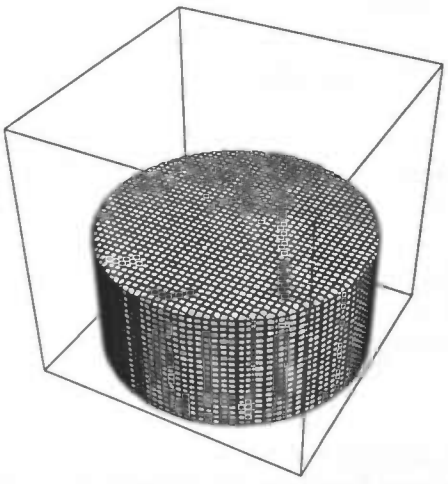


Figure 4.1: *Initial free-surface configuration.*

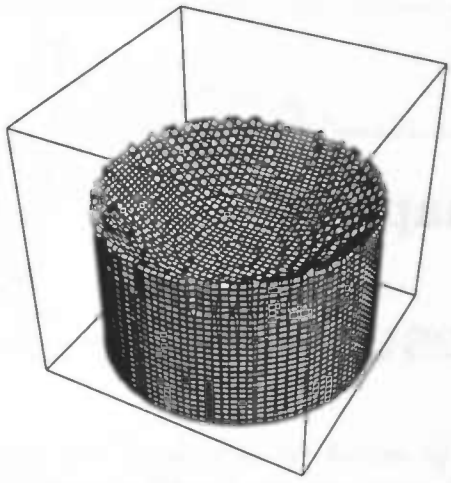


Figure 4.2: *Steady state solution after two seconds.*

agrees closely with the theory with regard to both the parabolic profile and the free-surface height. In figure 4.4 the free surface in the symmetry plane is plotted once more together with the velocity field showing the secondary flow pattern. Finally we take a

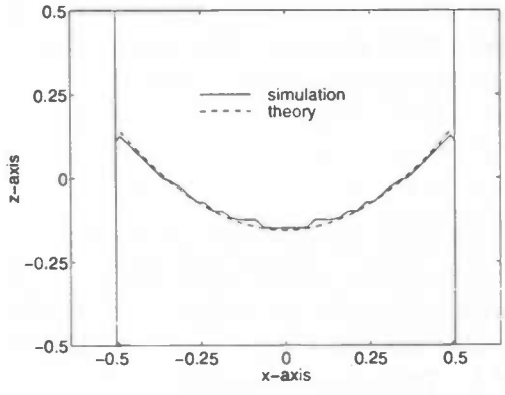


Figure 4.3: *Computed solution versus theoretical solution in the symmetry plane at time $t = 2$.*

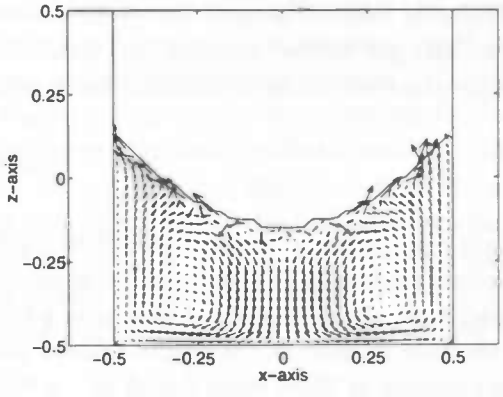
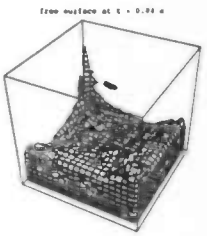


Figure 4.4: *Velocity field in the symmetry plane together with the computed free surface at time $t = 2$.*

look at the pressure along the bottom of the cylinder. For stationary flow the pressure must be equal to the hydrostatic pressure $p_h = gh + p_0$ where h is the free-surface height and p_0 the atmospheric pressure (in this simulation p_0 is set equal to unity). Since the free surface is parabolic the same must be true for the hydrostatic pressure p_h. In the origin the theoretical free-surface height is $\frac{11}{32}$ implying a pressure of $\frac{71}{16} = 4.4375$. This corresponds to the computed value of 4.4247. The pressure along the bottom of the cylinder is shown in figure 4.5.



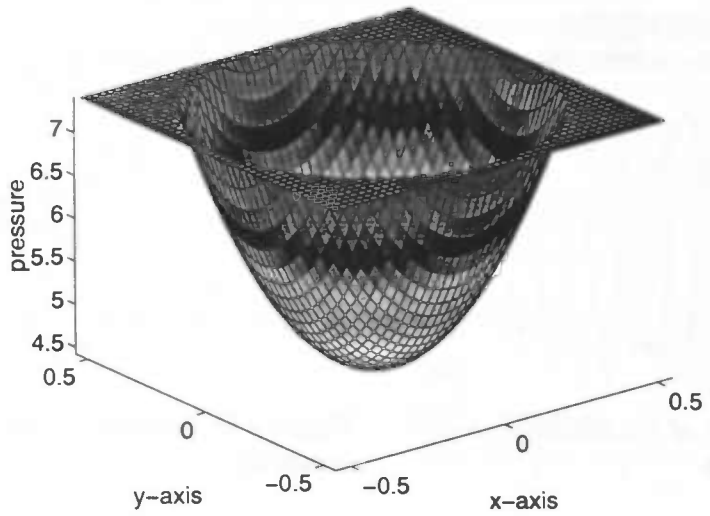


Figure 4.5: *Computed pressure along the bottom of the cylinder showing the parabolic hydrostatic pressure profile.*

4.2 Oscillation in a Cubic Container

In this section we simulate the free-surface flow in a cubic container with unity volume. The Cartesian origin is put in the center of mass of the cube and at the beginning of the simulation half of the container is filled with fluid. For the computations in this section we used 20 grid points in every spatial direction. We make a distinction between free oscillation and forced oscillation.

This simulation is similar to that described in [13], however we simulate three-dimensional viscous flow instead of two-dimensional inviscid flow. Yet the qualitative behaviour can be used as a reference.

4.2.1 Free Oscillation

To enable a free oscillation the container is filled with liquid for $z < \frac{1}{2}x$ (so half of the container contains fluid in an asymmetrical configuration: at the right wall the free surface is positioned at $z = 0.25$ while at the left wall this is at $z = -0.25$). The container is subjected to a gravitational force of $g = 10$ in negative z-direction. Hence the fluid will move to the left wall and after some period in time bounce back to the right wall and so on. Eventually the flow reaches an equilibrium position at which the container will contain fluid for $z < 0$. We will examine the free-surface height at the left wall and at the the right wall in the symmetry plane (i.e. $(x, y) = (-\frac{1}{2}, 0)$ and $(x, y) = (\frac{1}{2}, 0)$ respectively). The free-surface height in these points as it changes in time is shown in figure 4.6. After approximately 3.5 seconds the flow reaches the steady state solution. From the figure we can derive that the fluid oscillates with a natural frequency of about 5.3 radians per second. From potential theory (e.g. [14]) it is known that the lowest

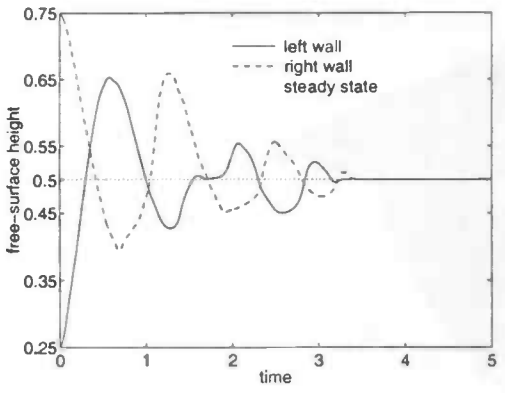


Figure 4.6: *Height of the free surface during free oscillation.*

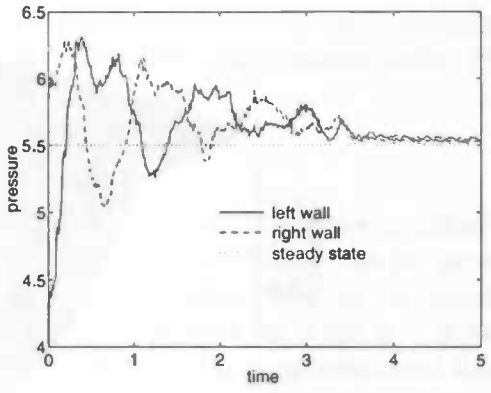


Figure 4.7: *Pressure in the bottom of the container.*

natural frequency is equal to

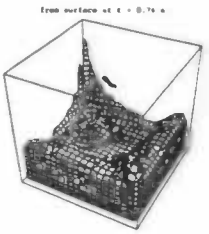
$$\omega_1 = \sqrt{\frac{\pi g}{L} \tanh\left(\frac{\pi h_0}{L}\right)} = 5.36 \text{ rad/s},$$

where $L = 1$ is the length of the container and $h_0 = \frac{1}{2}$ is the steady state free-surface height. Figure 4.7 shows the pressure in the points $(x, y, z) = (-\frac{1}{2}, 0, -\frac{1}{2})$ and $(x, y, z) = (\frac{1}{2}, 0, -\frac{1}{2})$, i.e. in the bottom of the container. The pressure oscillates with the same frequency as the free surface. When the steady state is reached the pressure must be equal to the hydrostatic pressure $p_h = gh + p_0$ as in the previous section. For the stationary flow the free-surface height h is $\frac{1}{2}$. However this has to be corrected for the positions where the pressure is solved from the Poisson equation. In this simulation the Cartesian grid coincides with the boundary $z = -\frac{1}{2}$ and the free surface $z = 0$. Hence the center of a computational cell is half a mesh size $(\frac{1}{2}h_z)$ above the bottom boundary and half a mesh size $(\frac{1}{2}h_z)$ under the free surface. So $h = \frac{1}{2} - \frac{1}{20} = \frac{9}{20}$ in this case (remember that $h_z = \frac{1}{20}$ since we used 20 grid points in the z-direction). For $g = 10$ and $p_0 = 1$ this results in a hydrostatic pressure of 5.5 in accordance with the figure.

4.2.2 Forced Oscillation

In the next simulations the container is filled with fluid for $z < 0$ and is subjected to a forced sinusoidal pitching oscillation around the y-axis with a frequency of ω radians per second and an amplitude of 5 degrees. We will see that the dynamic behaviour will be quite different for different ω. We recall that the theoretical first natural frequency of the liquid is $\omega_1 = 5.36$ radians per second. For all the computations 20 grid points in every spatial direction are used and in each case we examine the free-surface height at the left and the right wall in the symmetry plane $y = 0$.

First we take $\omega = 4.25$. The time development of the free-surface height is shown in figure 4.8. The position of the free surface changes very little due to a forced frequency that is less than the first natural frequency ω_1. If the forced frequency is increased to a value of $\omega = \omega_1 = 5.36$ the fluid starts to resonate as is clearly shown in figure 4.9.



Further the well-known nonlinear characteristic of the fluid is observed: the upward wave amplitude becomes greater than the downward one as the wave amplitude becomes larger [13]. In figure 4.10 the same forced frequency of 5.36 radians per second is used

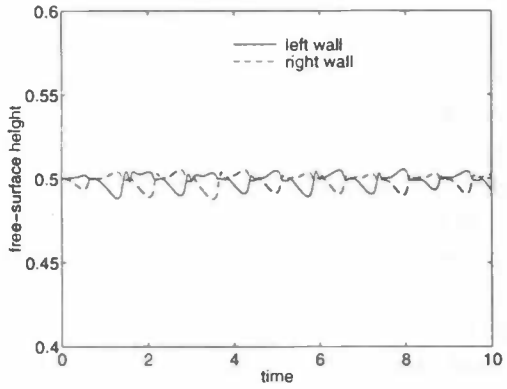


Figure 4.8: *Free-surface height during forced oscillation with* $\omega = 4.25$.

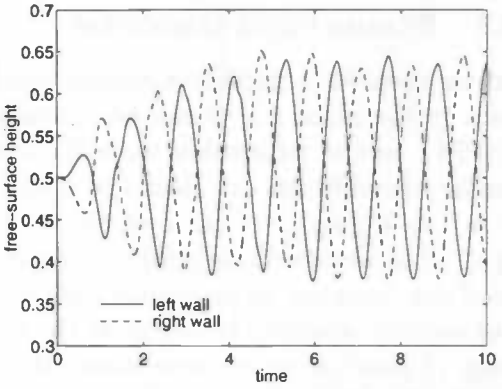


Figure 4.9: *Resonant free-surface height during forced oscillation with* $\omega = 5.36$.

but now the time development of the free surface is shown in the entire symmetry plane. Finally a calculation has been performed with $\omega = 10.72$, i.e. twice the natural frequency (figure 4.11). It is observed that the fluid starts to response in the natural frequency, whereas the final response is in the forced frequency.

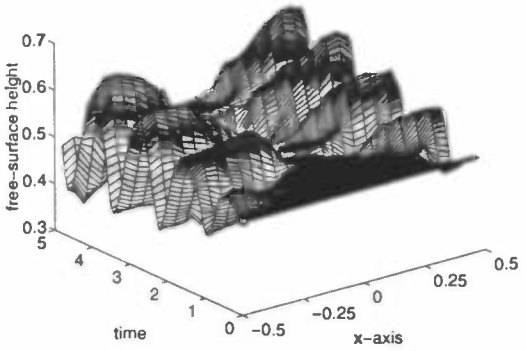


Figure 4.10: *Same as figure 4.9 but now in the entire symmetry plane* $y = 0$.

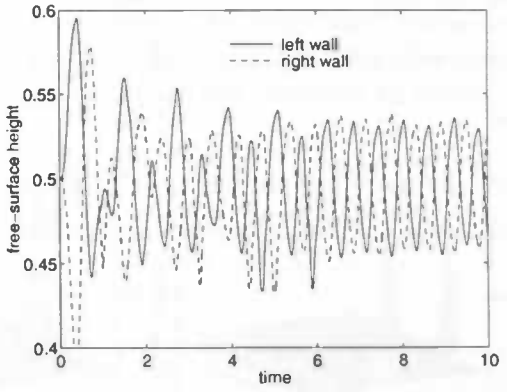


Figure 4.11: *Free-surface height during forced oscillation with* $\omega = 10.72$.

4.3 The Dambreak Problem

A classical problem in CFD is the dambreak problem in wich the free fall of a block of fluid inside a rectangular container is simulated. Most of the available references solve the dambreak problem in two dimensions and apply free-slip boundary conditions. In this section we will examine some three-dimensional dambreak problems with no-slip

boundary conditions. All of the calculations were performed inside a cubic container of dimensions $1 \times 1 \times 1$ which was divided into $24 \times 24 \times 24$ computational cells. The Cartesian origin is taken in the center of mass of the container.

4.3.1 Symmetrical Dambreak

First we consider a dambreak symmetrical in the y-direction such that the qualitative results in the plane $y = 0$ can be compared to two-dimensional results (see for example [16]). In the y-direction we will encounter typical 3-D effects due to the no-slip boundary conditions. At time $t = 0$ the container is filled with fluid for $x < 0$ and $z < 0$ (i.e. a quarter of the container). The fluid is subjected to a gravitational force of $g = 10$ in negative z-direction producing a transient wave which hits the right-hand side of the container at an impact time of approximately $t_{\text{imp}} = 0.28$ seconds. A violent jet of vertical velocities is formed at the right-hand side of the container when a sudden change of interface orientation takes place. Note that the maximum height of the wave front at the right wall is less than in [16] due to the no-slip boundaries we used in the present calculations. Furthermore the fluid is able to flow also in the y-direction instead of in the z-direction only. After some time the wave front falls back causing the free surface to overturn. Note that at this stage the free surface no longer is a single-valued function of space. After approximately 4.0 seconds the flow reaches the steady state solution. The above-mentioned is illustrated in figure 4.12. In figure 4.13 the pressure

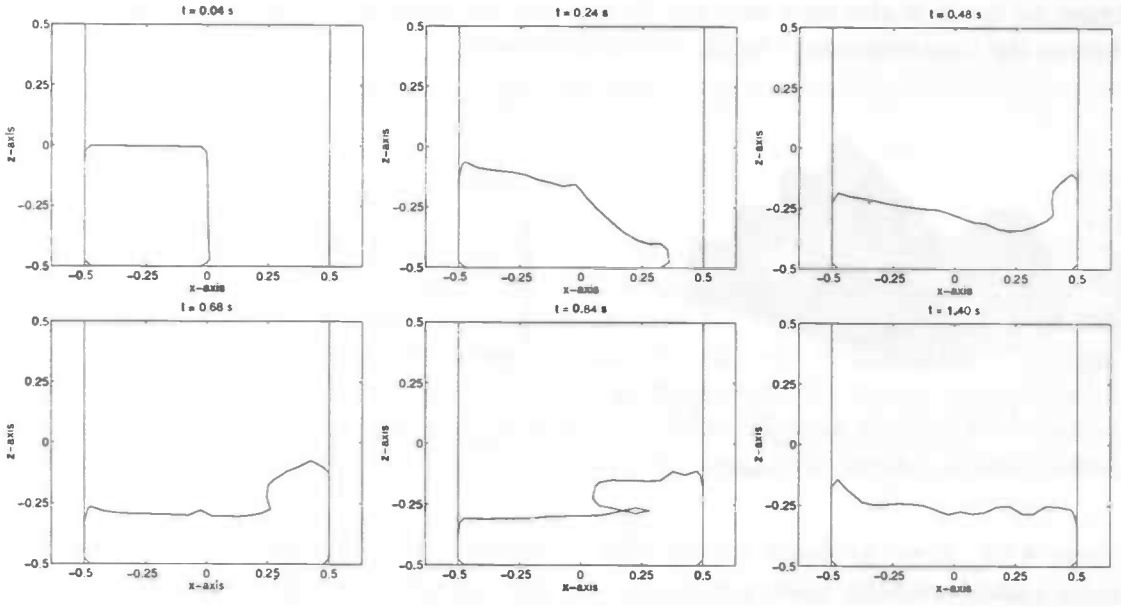
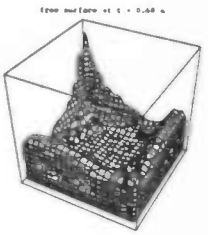


Figure 4.12: *Selected snapshots of the free-surface configuration in the symmetry plane for the dambreak problem.*

in $(x, z) = (\frac{1}{2}, -\frac{1}{2})$ (in the bottom of the right wall) is shown (again in the plane of symmetry). At impact time a sharp peak of the pressure is revealed (impact pressure) followed by a lower peak short after first impact. The pressure ends by a decaying oscillation towards hydrostatic pressure. This same behaviour was found in [16]. Since we used no-slip boundary conditions the first impact of the fluid with the right wall will be



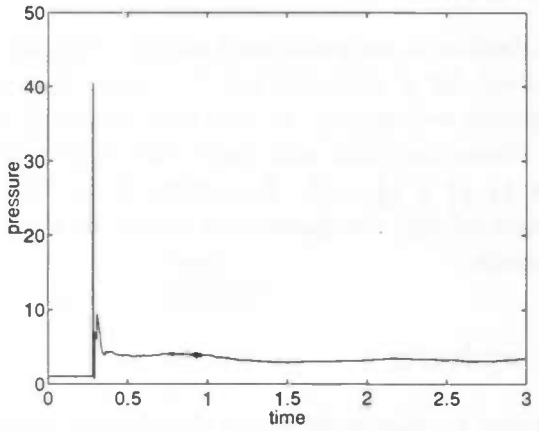


Figure 4.13: *Time history of the pressure (in the plane of symmetry) in the bottom of the right wall for the symmetrical dambreak problem.*

in the center of the right wall (i.e. in the symmetry plane $y = 0$). Short after the first impact the fluid near the symmetry plane will reach the right wall. So it is interesting to examine the pressure in these points as well. In figure 4.14 the pressure in the bottom of the entire container is shown at different points in time. In the first and second snapshot the pressure just before impact and just after first impact are recognized. Notice the relative low pressure in the symmetry plane in the second plot corresponding to a pressure drop immediately after first impact (see also figure 4.13). The third plot shows the impact at points well outside the symmetry plane. The fourth and fifth plot show the transition of impact towards final impact in the corners of the container in the final snapshot.

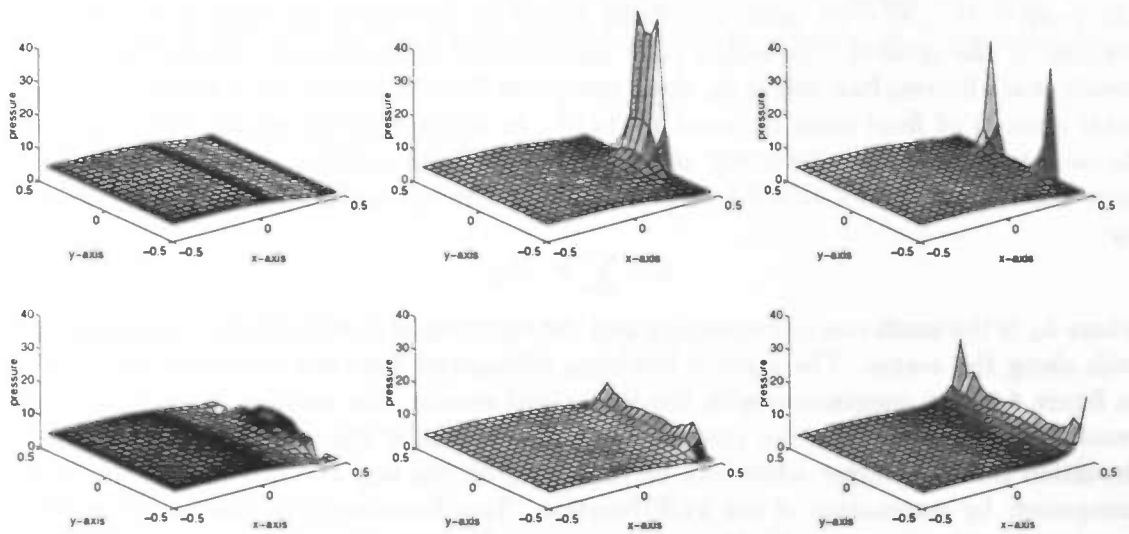


Figure 4.14: *Snapshots of the pressure in the bottom of the container.*

4.3.2 Asymmetrical Dambreak

Apart from symmetrical dambreak we performed another calculation in which the container initially contains fluid for $x < 0$ and $y > 0$. Again the fluid is subjected to a gravitational force in negative z-direction. In this case the fluid is able to flow in both the x- and y-direction. Hence the fluid will reach the walls $x = \frac{1}{2}$ and $y = -\frac{1}{2}$ first followed by an impact at $(x, y) = (\frac{1}{2}, -\frac{1}{2})$. Snapshots of the free-surface configuration during the first two seconds of this simulation are shown in the upper-right corners of every odd page in this report.

4.4 A Rotating Sphere

In the two previous sections we discussed liquid sloshing in rectangular domains. For these geometries it is natural to apply a Cartesian grid. In this section we will demonstrate that the computer program gives good results also for non-rectangular domains.

4.4.1 Axisymmetric Spin-Up

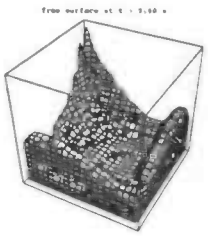
As a first example we consider the axisymmetric spin-up of a sphere of radius R. The Cartesian origin is taken in the center of the sphere. The spherical container is filled with liquid for $z < 0$ and is subjected to a rotational force by rotating it around the z-axis with a frequency of $\omega = 2\pi$ radians per second (i.e. one rotation per second). Further we apply a gravitational force of g = 9.81 in negative z-direction. The steady state solution follows from balancing the rotational and gravitational forces (see also section 4.1) resulting in

$$h = \frac{\omega^2}{2g} r + h_0, \tag{4.1}$$

where h_0 is the free-surface height along the z-axis and h is the free-surface height in the point $r = \sqrt{x^2 + y^2}$ (note that both h and h_0 are measured from $z = -R$; the bottom of the sphere). Equation (4.1) can be used for analytical calculation of the steady state free-surface height h_0 along the z-axis for different R, since for all $t > 0$ the total amount of fluid must be equal to $\frac{2}{3}\pi R^3$. In figure 4.15 the steady state value of h_0 (now measured from the center of the sphere) is plotted against R. Four calculations have been performed for $R = 0.1, 0.2, 0.3$ and 0.4. The free-surface height was computed by

$$h = \sum F^s \cdot h_z$$

where h_z is the mesh size in z-direction and the summation is over all the computational cells along the z-axis. The value h has been subtracted from the radius R and is plot in figure 4.15 for comparison with the theoretical results. For smaller R the computed results agree closely with the theory. When the radius of the sphere is increased the deviation becomes larger which can be explained by the way the free-surface height is computed: by summation of the VOF-function. This function is defined in cell-centres causing a maximum error of half a mesh size along both the free surface and the bottom of the sphere. The maximum error is plot also in figure 4.15 making clear that all of the computations are within this error bound.



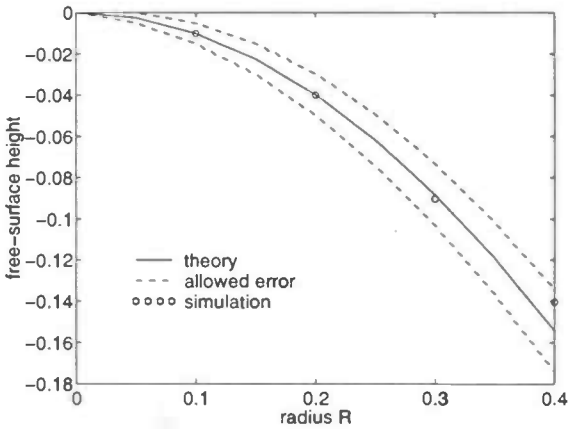


Figure 4.15: *Steady state free-surface height in the origin of the sphere.*

4.4.2 Asymmetric Spin-Up

As a final example we consider the asymmetric spin-up of a sphere (the radius r of the sphere is taken such that the volume is equal to one litre). Hereto we set the Cartesian origin outside the sphere. To be more precise: the center of the sphere has co-ordinates $(x, y, z) = (2r, 0, 0)$; so the axis of rotation (z-axis) is at a distance $2r$ away from the center of the sphere. Again the sphere is rotated with a frequency of 2π radians per second. In figure 4.16 snapshots of the free-surface configuration and the velocity field are shown (only the results in the plane through the z-axis and the center of the sphere are presented). The fluid starts to respond to the rotational force immediately due to the impulsive nature of the spin-up. After approximately 0.15 seconds the fluid passes the equilibrium position for the first time (overshoot). Hence the gravitational force becomes greater than the rotational force causing the velocity to change direction. This process is repeated until the fluid reaches the steady state solution after approximately one second.

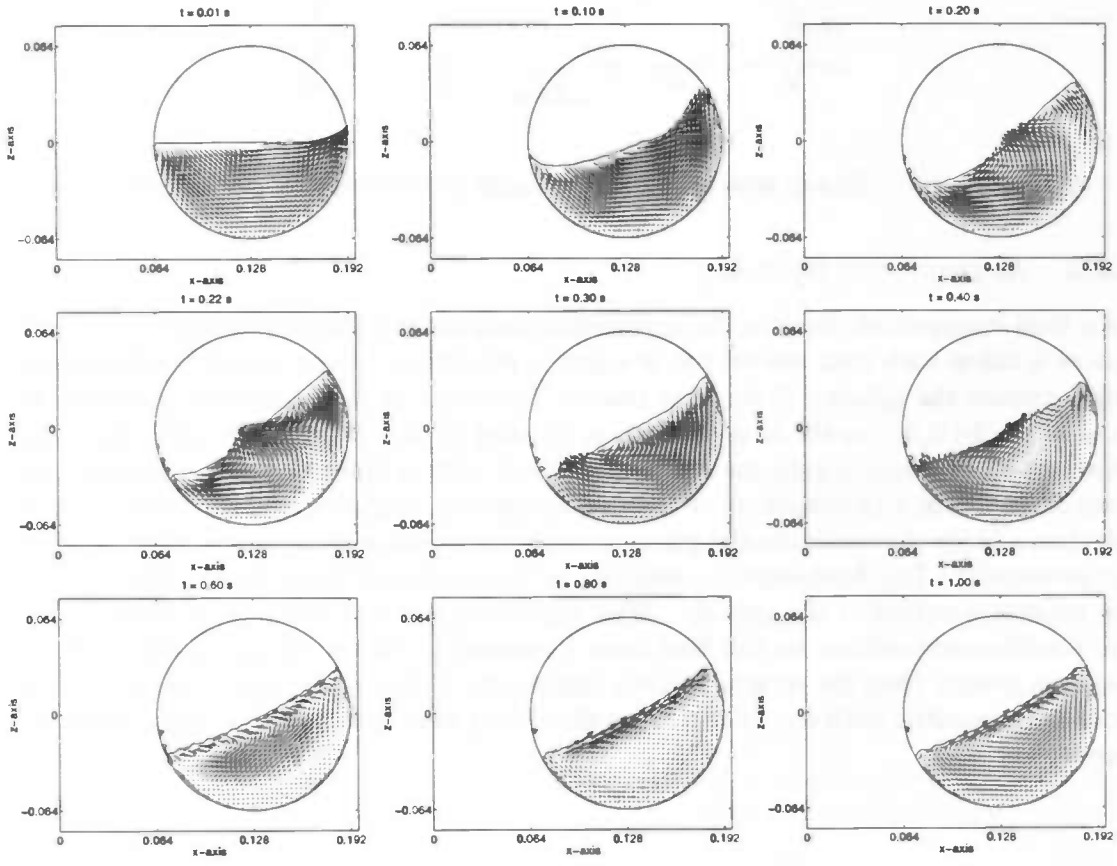
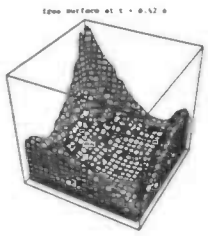


Figure 4.16: *Snapshots of the free surface and the velocity field for the asymmetric spin-up of a sphere.*



Chapter 5

Conclusions

In this report we studied free-surface flow in three-dimensional complex geometries. To avoid the complex task of generating a body-fitted grid, we simply covered the flow domain with a Cartesian grid. For the geometry recognition the labeling method described in [5] has been used. These geometry labels allow a simple application of the no-slip boundary conditions and, more important, avoid stability problems despite the occurence of irregular cells near the boundary of the flow domain.

For the extension to free-surface flow we introduced new labels (free-surface labels) based on the VOF-method. This method has been widely used in simulating free-surface flow. In the current project the free-surface labels have been combined with the geometry labels to allow free-surface flow in arbitrary complex geometries. This was accomplished by careful examination of all the possible label combinations and trying to find for every situation a solution as simple as possible.

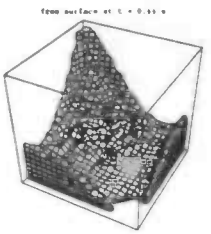
Summarized, we may conclude that

- the labeling method is capable of representing 3-D complex geometries.
- the VOF-method fits well in the current labeling method.
- the results are sufficiently accurate, in spite of a simple application of the boundary conditions and the free-surface conditions.

The current version of the computer program COMFLO-SLOSH is capable of simulating free-surface flow in 3-D complex geometries. For future research we suggest the following extensions.

- Inflow and outflow; to be able to simulate more industrial problems. We note that a version of COMFLO with inflow and outflow conditions, but without free surfaces, is already in preparation [4].
- Surface tension, making the simulation of fluid flow under low-gravity conditions possible.
- Dynamical interaction between the fluid and the flow domain; an extension also very useful for liquid motion in an extra-terrestrial environment.

- A more automated input of complex flow domains, i.e. domains that cannot be described by a simple mathematical formula.
- Heat transfer; for industrial problems.



Appendix A

Program Description

To simulate free-surface flow in three-dimensional complex geometries the computer program COMFLO-SLOSH has been written. Therefore the numerical model in chapter 3 has been implemented in FORTRAN77. In this appendix we will discuss the calling sequence, the most important variables, the individual subroutines and the input and output files of COMFLO-SLOSH.

A.1 Calling Sequence

The subroutines are called in the following order:

setup	SETPAR		
	GRID		
	BNDLAB	BNDDEF	
	SETFLD	SURDEF	
		SURLAB	
		BC	VELBC
time integration	INIT		
	TILDE	BDYFRC	
	SOLVEP	COEFL	
		COEFR	
		PRESBC	
		PRESIT	SLAG
		BC	VELBC
	VFCONV	SURLAB	
post-processing	MATH		
	MATLAB		
	AVS		

The setup subroutines are executed once, while the integration in time is repeated until the maximum simulation time is expired. The individual subroutines appearing in the calling sequence are discussed in section A.3.

A.2 Common Block Variables

The most important variables are grouped in common blocks, as stated below (in alphabetical order).

/APERT/

`AX(I,J,K)` Edge-aperture A^x between cells (i,j,k) and $(i+1,j,k)$.
`AY(I,J,K)` Edge-aperture A^y between cells (i,j,k) and $(i,j+1,k)$.
`AZ(I,J,K)` Edge-aperture A^z between cells (i,j,k) and $(i,j,k+1)$.
`FB(I,J,K)` Volume-aperture F^b in cell (i,j,k).
`FSN(I,J,K)`,`FS(I,J,K)` Volume of fluid function F^s in cell (i,j,k) at old and new time level.

/COEFP/

`DIV(I,J,K)` Right-hand side of pressure Poisson equation in cell (i,j,k).
`CC(I,J,K)` Coefficient of $p_{i,j,k}$ in pressure Poisson equation.
`CXL(I,J,K)`,`CXR(I,J,K)` Coefficients of $p_{i-1,j,k}$ and $p_{i+1,j,k}$.
`CYL(I,J,K)`,`CYR(I,J,K)` Coefficients of $p_{i,j-1,k}$ and $p_{i,j+1,k}$.
`CZL(I,J,K)`,`CZR(I,J,K)` Coefficients of $p_{i,j,k-1}$ and $p_{i,j,k+1}$.

/FLUID/

`NU` Kinematic viscosity coefficient ν.

/FORCE/

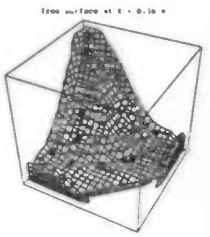
`AMPL` Amplitude of oscillation used in subroutine BDYFRC.
`FREQ` Frequency of oscillation used in subroutine BDYFRC.

/GRIDAR/

`IMAX,JMAX,KMAX` Number of grid points in x-, y- and z-direction.
`X(I)`,`Y(J)`,`Z(K)` Co-ordinates of grid points, such that the centre of a computational cell has co-ordinates $\left(\frac{x(i)+x(i-1)}{2}, \frac{y(j)+y(j-1)}{2}, \frac{z(k)+z(k-1)}{2}\right)$.
`DXP(I)`,`DYP(J)`,`DZP(K)` Mesh size: `DXP(I)=X(I)-X(I-1)`, etc.
`DXU(I)`,`DYV(J)`,`DZW(K)` Mesh size: `DXU(I)=0.5*(DXP(I)+DXP(I+1))`, etc.

/LABELS/

`ULABEL(I,J,K)`,`ULABFS(I,J,K)` Geometry label and free-surface label of velocity u between cells (i,j,k) and $(i+1,j,k)$.
`VLABEL(I,J,K)`,`VLABFS(I,J,K)` Geometry label and free-surface label of velocity v between cells (i,j,k) and $(i,j+1,k)$.
`WLABEL(I,J,K)`,`WLABFS(I,J,K)` Geometry label and free-surface label of velocity w between cells (i,j,k) and $(i,j,k+1)$.
`PLABEL(I,J,K)` Geometry label of pressure p in cell (i,j,k).
`PLABFSN(I,J,K)`,`PLABFS(I,J,K)` Free-surface label of pressure p in cell (i,j,k) at old and new time level.



/NUMER/

EPS Maximum error in pressure iteration.
OMEGA Relaxation parameter ω in SOR-iteration. The relaxation parameter is adjusted in subroutine PRESIT.
OMSTRT Initial relaxation parameter, i.e. the value of ω given in the input file.
ITMAX Maximum number of pressure iterations per time cycle.
ITER Number of pressure iterations in current time cycle.
NOM Number of time steps with same relaxation parameter.

/PHYS/

UN(I,J,K),U(I,J,K) Velocity u at old and new time level.
VN(I,J,K),V(I,J,K) Velocity v at old and new time level.
WN(I,J,K),W(I,J,K) Velocity w at old and new time level.
P(I,J,K) Pressure p at new time level.

/PLOTS/

PLOT23 Two-dimensional or three-dimensional plots in MATHEMATICA.
NPMATH Total number of plots in MATHEMATICA.
NPMATL Total number of plots in MATLAB.

/SPACE/

DOMAIN Type of flow domain (cube, cylinder, sphere etc.).
XMIN,XMAX Minimum and maximum x-co-ordinate of the mesh.
YMIN,YMAX Minimum and maximum y-co-ordinate of the mesh.
ZMIN,ZMAX Minimum and maximum z-co-ordinate of the mesh.

/TIME/

TMAX Maximum simulation time.
T Current time.
DT Time step δt.
CYCLE Current time cycle.

A.3 Subroutines

Next we discuss the individual subroutines in COMFLO-SLOSH (in alphabetical order). For every subroutine we give the input and output variables and a short description.

AVS

Input: FB,FS, IMAX,JMAX,KMAX, U,V,W.

Description: Create files for visualizing data in AVS.

Output: files comflo.fld and comflo.dat (see section A.4).

BC

Input: AX,AY,AZ,FB, IMAX,JMAX,KMAX, DXP,DYP,DZP,
ULABEL,VLABEL,WLABEL, U,V,W.

Description: The boundary velocities (**FB**, **SB**, **EB** and **BB**) are computed using the no-slip boundary conditions and the apertures. The various geometry labels are used to determine in which way the boundary velocities are computed (see section 3.5). At the end of the subroutine the free-surface velocities are computed by calling the subroutine VELBC.

Output: U,V,W.

BDYFRC

Input: AMPL,FREQ, DOMAIN, T.

Description: For every point in the computational grid this subroutine is called from the subroutine TILDE and returns the vectors DQDT = $\mathrm{d}\boldsymbol{q}/\mathrm{d}t$, OMET = $\boldsymbol{\omega}$ and DOMEDT = $\mathrm{d}\boldsymbol{\omega}/\mathrm{d}t$ in the virtual body force (2.7). Further a vector GRAV = $\boldsymbol{F}$ is returned representing the external body force in the Navier-Stokes equations (usually a gravitational force).

Output: DQDT,OMET,DOMEDT,GRAV.

BNDDEF

Input: DOMAIN, XPT,YPT,ZPT.

Description: If the boundary of a flow domain is described by $s(x,y,z) = 0$ then the interior of the flow domain is characterised by $s(x,y,z) > 0$ (for example a unit cube in 3-D: $s = \frac{1}{2} - \max(|x|,|y|,|z|)$). The variables XPT,YPT,ZPT are parameters of the subroutine and represent a certain point in the grid. Dependent of the type of domain DOMAIN this subroutine returns the value S in this point and a number VOF = 1 if $s > 0$ and VOF = 0 else.

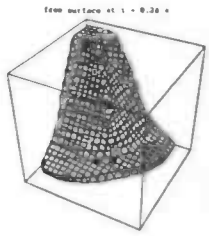
Output: S,VOF.

BNDLAB

Input: IMAX,JMAX,KMAX, X,Y,Z, S,VOF.

Description: First this subroutine computes the edge- and volume-apertures with the help of S and VOF (returned by the subroutine BNDDEF). Then the cells are labeled **F**, **B** or **O** based on the geometry only. Also the velocities are labeled based on the geometry. At this stage the boundary velocities have a correct label indicating how these velocities are computed. For more detailed information we refer to section 3.2.1 and the subroutines BC and BNDLAB in the computer program.

Output: AX,AY,AZ,FB, ULABEL,VLABEL,WLABEL,PLABEL.



COEFL

Input: AX,AY,AZ,FB, IMAX,JMAX,KMAX,DXP,DYP,DZP,DXU,DYV,DZW,
ULABFS,VLABFS,WLABFS,PLABFS.

Description: Every time step the matrix in the left-hand side of the pressure Poisson equation (3.12) is computed. In every **F**-cell a central coefficient and 6 neighbour coefficients are needed. At the end of this subroutine all the coefficients are scaled by the central coefficient.

Output: CC,CXL,CXR,CYL,CYR,CZL,CZR.

COEFR

Input: AX,AY,AZ,FB, CC, IMAX,JMAX,KMAX,DXP,DYP,DZP,
PLABFS, U,V,W, DT.

Description: Compute every time step the right-hand side of the pressure Poisson equation (3.12) in every **F**-cell. The right-hand side is scaled by the central coefficient CC (see subroutine COEFL).

Output: DIV.

GRID

Input: IMAX,JMAX,KMAX, XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX.

Description: Generate a (uniform) 3-D grid by dividing XMAX-XMIN into IMAX intervals etc. and compute the various mesh sizes.

Output: X,Y,Z,DXP,DYP,DZP,DXU,DYV,DZW.

INIT

Input: FS, IMAX,JMAX,KMAX, U,V,W, CYCLE.

Description: Start of a new time cycle. The momentum velocities U, V and W are copied to UN, VN and WN respectively and are set equal to zero again. Further the new volume of fluid function FS is copied to FSN and CYCLE is increased by one.

Output: FSN, UN,U,VN,V,WN,W, CYCLE.

MATH

Input: FS, IMAX,JMAX,KMAX, PLOT23,NPMATH, TMAX,DT,CYCLE.

Description: Create NPMATH files for visualizing the free surface in MATHEMATICA. Either two-dimensional or three-dimensional data is returned (based on the value of PLOT23).

Output: files comflo***.math (see section A.4).

MATLAB

Input: FS, IMAX,JMAX,KMAX,X,Y,Z, PLABFS, U,V,W,P,
NPMATL, TMAX,DT,CYCLE.

Description: Create NPMATL files for visualizing the free surface in MATLAB. Further it is possible to create data files with the velocity field in an arbitrary plane of the grid, the pressure in certain points etc.

Output: files comflo***.m (see section A.4).

PRESBC

Input: IMAX,JMAX,KMAX, PLABFS.

Description: The free-surface condition (2.10) is applied. In every S-cell the pressure is set equal to the atmospheric pressure.

Output: P.

PRESIT

Input: IMAX,JMAX,KMAX, PLABFSN,PLABFS, EPS,OMEGA,OMSTRT,ITMAX,ITER,NOM, DELTA, CYCLE.

Description: The pressure Poisson equation is solved by SOR-iteration. In this subroutine the relaxation parameter is automatically adjusted to obtain a higher convergence rate [1]. The subroutine SLAG is called until either the 'error' DELTA is less than EPS or ITER exceeds ITMAX. In the latter case the pressure iteration did not converge and the program is terminated.

Output: P.

SETFLD

Input: IMAX,JMAX,KMAX.

Description: First the subroutines SURDEF and SURLAB are called in order to intialize the fluid configuration and set the free-surface labels respectively. Then the velocity field is initialized: every velocity is set equal to zero. Furthermore the pressure is set equal to unity throughout the computational grid. Finally the boundary velocities and free-surface velocities are computed by calling the subroutines BC and VELBC.

Output: U,V,W,P.

SETPAR

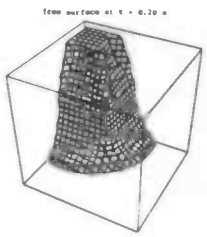
Input: file comflo-in (see section A.4).

Description: Read and initialize the variables listed below.

Output: NU, AMPL, FREQ, IMAX,JMAX,KMAX, EPS,OMEGA,OMSTRT,ITMAX,NOM, PLOT23,NPMATH,NPMATL, DOMAIN,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX, TMAX,DT,CYCLE.

SLAG

Input: DIV,CXL,CXR,CYL,CYR,CZL,CZR, IMAX,JMAX,KMAX, PLABFS, OMEGA,ITER, P.



Description: This subroutine executes one SOR-sweep (so ITER is increased by one) to solve the pressure Poisson equation; a relaxation parameter OMEGA is used. A new pressure solution is returned, as well as the Euclidean norm DELTA of the difference of the previous and the new pressure solution.

Output: ITER, P, DELTA.

SOLVEP

Input: IMAX,JMAX,KMAX,DXU,DYV,DZW, ULABFS,VLABFS,WLABFS,
U,V,W,P, DT.

Description: First the left-hand side and the right-hand side of the Poisson equation are computed by calling the subroutines COEFL and COEFR. Then the pressure Poisson equation is solved by calling PRESIT. The velocity field (3.13) at a new time level is computed and finally the boundary velocities and free-surface velocities are adjusted via the subroutines BC and VELBC.

Output: U,V,W,P.

SURDEF

Input: FB, IMAX,JMAX,KMAX,X,Y,Z, DOMAIN.

Description: The flow domain is partially filled with fluid at time $t = 0$, i.e. the volume of fluid function FS is initialized. Hereby we keep in mind that $F^s \leq F^b$.

Output: FS.

SURLAB

Input: FS, IMAX,JMAX,KMAX, ULABEL,VLABEL,WLABEL,PLABEL,PLABFS.

Description: This subroutine labels the cells and velocities based on the free-surface configuration. Since the free surface changes in time, this labeling is repeated every time step. The various free-surface labels indicate how the free-surface velocities are computed. For more detailed information we refer to section 3.2.2 and the subroutines VELBC and SURLAB in the computer program.

Output: ULABFS,VLABFS,WLABFS,PLABFS,PLABFSN.

TILDE

Input: AX,AY,AZ,FB, NU, IMAX,JMAX,KMAX,X,Y,Z,DXP,DYP,DZP,
DXU,DYV,DZW, ULABFS,VLABFS,WLABFS, UN,VN,WN, DT,
DQDT,OMET,DOMEDT,GRAV.

Description: Computation of the temporary vector field (3.7). Hereto for every point in the grid the subroutine BDYFRC is called returning the external body force $\boldsymbol{F}$ and the virtual body force $\boldsymbol{f}$.

Output: U,V,W.

VELBC

Input: FS, IMAX,JMAX,KMAX,DXP,DYP,DZP,DXU,DYV,DZW,
ULABFS,VLABFS,WLABFS, U,V,W.

Description: The free-surface velocities **SE** and **EE** are computed. Hereby the free-surface labels are used to determine how these velocities are computed using the free-surface condition (2.11) (see section 3.4).

Output: U,V,W.

VFCONV

Input: AX,AY,AZ,FB,FSN,FS, IMAX,JMAX,KMAX,DXP,DYP,DZP,
PLABEL,ULABFS,VLABFS,WLABFS,PLABFS, U,V,W, DT.

Description: First fluxes are computed at the cell-faces **FF**, **FS**, **SS** and **SE**. Then the fluid is transported from donor cells to acceptor cells, i.e. the volume of fluid function FS at the new time level is computed. At the end of the subroutine the cells and velocities are relabeled based on the new VOF-distribution by calling SURLAB.

Output: FS.

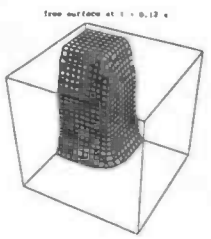
A.4 Files

Input File

The computer program reads the input file comflo-in to initialize variables (see the subroutine SETPAR in the previous section). This file has the following structure.

```
xmin,  xmax,  ymin,  ymax,  zmin,  zmax
-0.6   0.6    -0.6   0.6    -0.6   0.6

imax, jmax, kmax (SMALLER than 65)
24    24    24

tmax, dt
2.0   0.001

nu,    domain, ampl, freq
0.01   1       0.0   0.0

eps,    omega, itmax
1.0E-4  1.70   1000

plot23, npmath, npmatl
3       50      100
```

This specific file corresponds to the dambreak problem in section 4.3. For more information we refer to the file itself containing an explanation of all the variables in the input file.



Iteration File

The ouput file `comflo-iter` shows the supremum norm of two successive solutions of the three velocity components and the Euclidean norm of two successive pressure solutions. Furthermore the number of iterations per time step is listed and at the beginning and the end of the simulation the liquid percentage is printed.

AVS Files

The subroutine AVS writes output to the files `comflo.fld` and `comflo.dat`, which in turn can be used as input files for the visualization program AVS. The file `comflo.fld` tells AVS for example the number of spatial dimensions, how many grid points are used and the type of data that is returned. For example, the file `comflo.fld` corresponding to the input file `comflo-in` given above:

```
# AVS field
ndim=3
dim1=24
dim2=24
dim3=24
nspace=3
veclen=5
data=float
field=uniform
label=geometry
label=fluid
label=x-velocity
label=y-velocity
label=z-velocity
#
 variable 1 file=comflo.dat filetype=ascii skip=1 offset=0 stride=5
 variable 2 file=comflo.dat filetype=ascii skip=1 offset=1 stride=5
 variable 3 file=comflo.dat filetype=ascii skip=1 offset=2 stride=5
 variable 4 file=comflo.dat filetype=ascii skip=1 offset=3 stride=5
 variable 5 file=comflo.dat filetype=ascii skip=1 offset=4 stride=5
```

The actual data is contained in the file `comflo.dat`. This file consists of a number of columns, each representing a variable as described in the field-file `comflo.fld`. Every line of this file represents a certain point in the computational grid. We remark that an indication of the boundary of the flow domain and the free surface can be obtained by examining the isosurfaces $F^b = \frac{1}{2}$ and $F^s = \frac{1}{2}$ (the first and second column of the data-file respectively).

MATHEMATICA Files

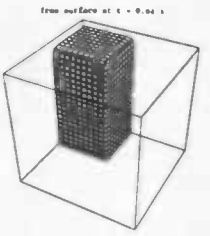
It is possible to make two-dimensional as well as three-dimensional movies of the free surface as it changes in time. Hereto the subroutine MATH creates a number of files

`comflo***.math` with *** a number starting with 101. Each file contains data representing the free-surface configuration at a certain point in time. These files can be read directly into MATHEMATICA by the command `<<comflo***`. Then a variable `fs` is introduced of which a `ListContourPlot` (2-D) or a `ListContourPlot3D` (3-D) can be made. By doing this for all the MATHEMATICA files a movie is built frame by frame. The notebook version of MATHEMATICA provides tools for combining these frames into a movie.

MATLAB Files

Finally the computer program returns files for visualizing the data in MATLAB. By default the files `coord.m` and `comflo***.m` are created and can be read directly into MATLAB. The former file introduces the vectors `x`, `y` and `z` with co-ordinates of the cell centres. The latter files contain the free-surface configuration in the plane $y = 0$ (but this may be changed to any other plane). The use of these files is similar to the MATHEMATICA files discussed above. A plot of the free surface is made by the command `contour(x,z,fs,[0.5 0.5])`.

Further it is possible to export the velocity field in a certain plane at a certain time, the pressure in a certain point during the entire simulation, the free-surface height in a certain point during the entire simulation, and so on. The MATLAB plots in this report have been made with the commands `plot`, `quiver` and `surf`. For more information we refer to the subroutine MATLAB in the computer program.



Bibliography

[1] E.F.F. Botta and M.H.M. Ellenbroek (1985) A Modified SOR Method for the Poisson Equation in Unsteady Free-Surface Flow Calculations, *J. Comput. Phys.*, **60**, 119-134.

[2] E.F.F. Botta (1992) Eindige differentie methoden, *Lecture notes* RuG.

[3] K.H. Chen and R.H. Pletcher (1993) Simulation of Three-Dimensional Liquid Sloshing Flows using a Strongly Implicit Calculation Procedure, *AIAA J.*, **31**, 901-910.

[4] J. Dijkstra (1997) Simulation of Flow past Complex Geometries using Cartesian Grid, *Master's thesis* RuG, in preparation.

[5] J. Gerrits (1996) Fluid Flow in 3-D Complex Geometries — A Cartesian Grid Approach, *Master's thesis* RuG.

[6] P. Hansbo (1995) Lagrangian Incompressible Flow Computations in Three Dimensions by Use of Space-Time Finite Elements, *Int. J. Numer. Methods Fluids*, **20**, 989-1001.

[7] F.H. Harlow and J.E. Welch (1965) Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface, *Phys. Fluids*, **8**, 2182-2189.

[8] C.W. Hirt and B.D. Nichols (1981) Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries, *J. Comput. Phys.*, **39**, 201-225.

[9] H.W. Hoogstraten (1992) Stromingsleer, *Lecture notes* RuG.

[10] D.B. Johnson, P.E. Raad and S. Chen (1994) Simulation of Impacts of Fluid Free Surfaces with Solid Boundaries, *Int. J. Numer. Methods Fluids*, **19**, 153-176.

[11] H. Miyata and S. Nishimura (1985) Finite-Difference Simulation of Nonlinear Ship Waves, *J. Fluid Mech.*, **157**, 327-357.

[12] H. Miyata (1986) Finite-Difference Simulation of Breaking Waves, *J. Comput. Phys.*, **65**, 179-214.

[13] T. Nakayama and K. Washizu (1980) Nonlinear Analysis of Liquid Motion in a Container Subjected to Forced Pitching Oscillation, *Int. J. Numer. Methods Eng.*, **15**, 1207-1220.

[14] A.R. Paterson (1985) *A First Course in Fluid Dynamics*, Cambridge University Press.

[15] R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutchfield and M.L. Welcome (1995) An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions, *J. Comput. Phys.*, **120**, 278-304.

[16] Z.A. Sabeur, W. Roberts and A.J. Cooper (1995) Development and Use of an Advanced Numerical Model using the Volume of Fluid Method for the Design of Coastal Structures, *Numerical Methods for Fluid Dynamics 5*, K.W. Morton and M.J. Baines Eds., Oxford Science Publishers, 565-573.

[17] A.E.P. Veldman and M.E.S. Vogels (1984) Axisymmetric Liquid Sloshing under Low-Gravity Conditions, *Acta Astronautica*, **11**, 641-649.

[18] A.E.P. Veldman (1994) Numerieke stromingsleer, *Lecture notes* RuG.

[19] J.A. Viecelli (1969) A Method for Including Arbitrary External Boundaries in the MAC Incompressible Fluid Computing Technique, *J. Comput. Phys.*, 4, 543-551.

[20] J.A. Viecelli (1971) A Computing Method for Incompressible Flows Bounded by Moving Walls, *J. Comput. Phys.*, **8**, 119-143.