Surface Capturing and Multigrid for Steady Free-Surface Water Flows

Surface Capturing and Multigrid for Steady Free-Surface Water Flows

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus
prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen
op maandag 5 november 2007 om 15.00 uur

door

Jeroen WACKERS

ingenieur Luchtvaart- en Ruimtevaarttechniek
geboren te Nijmegen

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ir. B. Koren

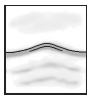
Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
prof.dr.ir. B. Koren,	Technische Universiteit Delft / CWI, promotor
prof.dr.ir. E. Dick,	Universiteit Gent
prof.dr. P.W. Hemker,	Universiteit van Amsterdam / CWI
prof.dr.ir. J.J.W. van der Vegt,	Universiteit Twente
prof.dr. A.E.P. Veldman,	Rijksuniversiteit Groningen
prof.dr.ir. C. Vuik,	Technische Universiteit Delft
dr.ir. H.C. Raven,	MARIN

The research in this thesis has been supported by the Dutch government in the BSIK/BRICKS project, theme MSV1.6. It was carried out at the Centrum voor Wiskunde en Informatica (CWI), the national research institute for mathematics and computer science, in Amsterdam.

The cover shows the wave pattern created by a swimming duck. Photo: author, design: Rutger Vos and Pieter Wackers.

ISBN 978-90-6464-178-7



Foreword

This Ph.D. thesis is the result of my research from 2003 to 2007 at CWI in Amsterdam and, in the summer of 2006, at NMRI in Tokyo. Before I let the reader go on, I would like to thank the people who helped me during those years.

First, I want to thank my supervisor prof.dr.ir. Barry Koren. He has given me the chance to work on such a beautiful subject as water flow, this thesis profited from his careful reading, and his suggestions have kept the project going at some crucial moments. He gave me the freedom to work on my own when I wanted to, which must have been difficult sometimes. And when I needed help, with anything, his door was always open for me. I am thankful for that, above everything else in our cooperation.

Then I thank the members of my committee for investing their time and effort in my work. I thank CWI for giving me a pleasant working environment. And I thank the Dutch voters who, through the ministry of Economic Affairs, provided the funding for this project.

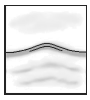
Many colleagues, both Dutch and international, gave me useful advice. I thank them all, and mention some of the most important. Prof.dr. Piet Hemker helped me with his explanation of multigrid and Fourier analysis, and with his views on science in general. Helpful comments came from the participants in the former CFD work-progress meetings at CWI, and the current MAS2 meetings. I profited from the work of my predecessors in water flow simulation at CWI, dr.ir. Harald van Brummelen and dr.ir. Mervyn Lewis. Discussions with the CFD researchers at MARIN were both useful and inspirational. Chapter 6 benefited from Jasper Kreeft's recent M.Sc. work on the same topic. And last but not least, I thank dr. Takanori Hino for allowing me to visit NMRI and for his care when I was there.

Then there are the people who shared my life, and made it worth living. A big 'thank you' to my roommates at CWI: Bob Planqué and Carolynne Montijn, Scott MacLachlan, and the two from Behind the Iron Curtain, Joost Batenburg and Willemien Ekkelkamp. Also to my other colleagues in MAS2, in MAS3, and in the rest of CWI. To Domenico Lahaye and Margreet Nool for dinners together, to the juggling people, the salsa people, the intersection of the two, and the fantastic people in CWI's supporting departments. To my fellow PhDays 2004 organisers. To the people at NMRI, 海技研のみなさん for giving me so much friendship in such a short time. To my friends in Haarlem, in the Netherlands and throughout the world. And finally, to my family.

With these thanks, I end here. Of my work, my thoughts, my feelings in the last four years, this thesis can speak for itself.

Haarlem, Amsterdam, Nantes,
August 2007,

Jeroen Wackers



Contents

Introduction

This thesis describes a surface capturing discretisation for 2D steady water flow that can be solved efficiently, and its fast multigrid solver. The Introduction starts with a description of water flow computation around ships, that motivates the need for such a discretisation (section 1.1). Then it explains what surface capturing is and why it offers advantages over other techniques (section 1.2). After an introduction to multigrid solution (section 1.3), the crucial section 1.4 shows why surface capturing forces most researchers to use slow solution techniques, but why we think that it can be solved fast. Section 1.5 gives an outline of the remaining chapters in this thesis, where this fast solution technique is presented.

1.1 Computing flows around ships

A ship moving through water causes a flow in that water. This water flow determines the ship's drag, its position with respect to the water surface, and its manoeuvring characteristics.

To design an energy-efficient ship, accurate computer simulation of this flow is indispensable. Optimizing a ship's shape after it has been built is virtually impossible. And even towing-tank tests with ship models cannot be used as the only means of optimizing a ship's hull shape. Building and testing scale models is very expensive and it is difficult to change the shape of a model after it has been built, so model tests are often used only to check the ship's final hull shape. Thus, computer simulation remains as the only tool for hull shape optimisation during the design.

Simulation of ship flow requires the modelling of many different flow phenomena. Accurately computing the waves around and behind the ship (figure 1.1 and front cover) is important. Creating waves costs a lot of energy, that has to be produced by the ship's engine(s). Therefore, a ship that generates less waves is more efficient.

A particularly interesting type of wave field is the flow around a ship with a transom stern. These ships have a flat, more or less vertical stern that is partially submerged when the ship is at rest or moving slowly (see figure 1.2a). But at higher velocities, the water flows directly aft from the bottom of the hull and the whole transom is dry (figure 1.2b), which gives low drag at high velocities. Thus, over some speed range, the flow changes its topology [41, 65]. From a computational point of view, this is very challenging.

On the hull, boundary layers appear: very thin regions in which the flow velocity decreases from the outside velocity to zero on the hull itself. The flow in the boundary layers is determined by the viscosity of the water; skin friction is created in these regions. Also, the flow often separates near the stern. There, the flow may not be able to follow the inward curvature of the hull. Instead, it detaches from the hull. Between the hull and this main flow, a turbulent wake appears with recirculating flow and low velocities (figure 1.3). The point of separation is determined by the development of the boundary layers.

Figure 1.1. Wave pattern generated by a ship (M-frigate, source: Royal Dutch Navy).

Accurate prediction of the separation region is important, for two reasons. First, the wake causes a large part of the ship's pressure drag. And second, the ship's propeller (or propellers) is positioned in the wake region; too much separation decreases the propeller efficiency and may even cause damage to the propeller. Thus, accurate computation of all aspects of the ship water flow is needed to give enough information for a good ship design.

To get a correct solution, it is essential that all physical processes in the flow are modelled together. This is not directly obvious. The ship's waves are caused by gravity effects on the water surface; viscous effects are negligible here. On the other hand, the boundary layer flow is dominated by viscosity and turbulence, but gravity

a) b)

Figure 1.2. Transom stern, that is submerged when the boat is at rest (a) and dry at high speed (b) (source: Tom Lihan).

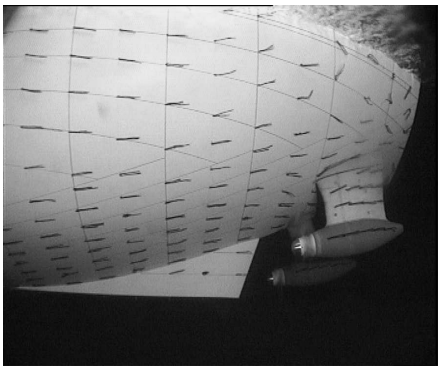


Figure 1.3. Flow separation and recirculation on the stern of a ship model, indicated by tufts (source: MARIN).

has almost no influence on it. Thus, it is tempting to compute the free-surface waves and the boundary layers separately.

However, especially near the stern, the two flow regimes are coupled. Flow separation changes the waves that are created at the stern, waves often become lower when the flow separates. The free surface flow also changes the pressure field at the stern; this may change the position of the separation region or even cause or prevent separation altogether. The flow in figure 1.3 is an example: computations at MARIN of the viscous flow around this ship, without wave formation, produced no separation. When waves were taken into account, the separation region was computed correctly [53]. Thus, for accurate results, a ship flow model has to incorporate viscosity, turbulence effects, and the wave development at the water surface.

To be suitable for design, a simulation method faces another requirement besides accuracy: speed. A designer needs to see the effect of a design change quickly, otherwise the design process is slowed down too much. A method that requires weeks or even months to compute a single flow can only be used in the same way as a towing-tank test: to check a final design. Optimisation of a design requires solutions that are found within a day, or at most in a couple of days.

The current state of the art in ship design is, that separate models for the waves and the viscous flow can be solved both accurately and efficiently. For combined models, this depends on the type of the model [25]. One class of models can be solved efficiently; unfortunately these models can only handle a limited group of ship geometries. Other models allow more general geometries, but their solution is time-consuming. What makes the difference between these models, is the treatment of the water surface. The different models are explained below.

1.2 Water surface models

Solving a flow problem starts with the definition of the flow domain: the region in which the flow takes place. Then this flow domain is divided into small cells, which form the computational grid. Finally, the flow is computed on this grid.

A special aspect of water flow is, that a part of the domain geometry – the water surface – is not known beforehand. One part of the domain boundary is the ship's hull, which has a known shape. Another part is the water surface, that has a shape which depends on the flow. Thus, a model is needed for the water surface; a model that can change the shape of the surface during the computation.

To compute steady flow, two water surface models are in use. For surface fitting, the grid is deformed during the computation, such that its upper boundary coincides with the surface. For surface capturing, the grid is not changed, but the water surface can move through the grid[1].

1.2.1 Surface fitting As mentioned, in surface fitting methods the grid is deformed such that its upper boundary gets the shape of the water surface. Therefore, the free-surface boundary conditions, that determine the flow behaviour at the water surface, are imposed on the upper boundary of the grid. A steady water flow satisfies two boundary conditions on the water surface:

1. Kinematic condition: the flow moves parallel to the surface, i.e. the surface is a streamline of the flow.
2. Dynamic condition: the pressure at the surface is a constant (ignoring the effect of air flow above the water, which is usually small).

These two conditions determine both the flow and the water surface shape; they can only be satisfied together when the water surface is in the right location.

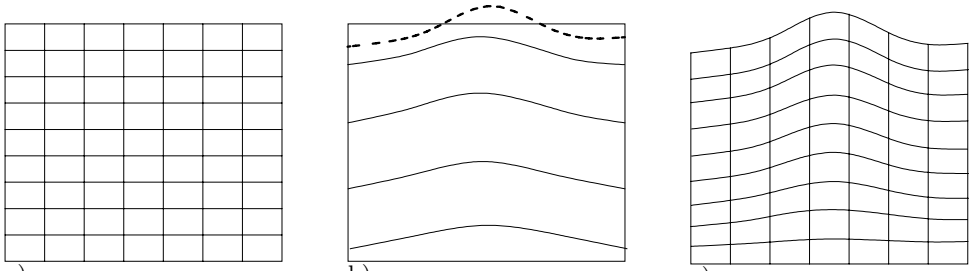


Figure 1.4. Surface fitting: an initial grid (a), the flow solution on this grid (isolines of pressure) that gives an estimate (- - -) of the water surface location (b), and a new grid, based on this estimate (c). This is a side view of a 2D flow, the water surface is at the top.

The surface fitting technique starts with an estimate of the water surface shape (this estimate may be a flat surface); a grid is made with this estimated surface shape (figure 1.4a). Then a flow solution on this grid is computed, that satisfies one free-surface boundary condition, usually the kinematic condition (although alternatives

[1] A third class of models, the surface tracking techniques, follow the surface with discrete marker particles [20]. These methods are most suitable for unsteady flows and are not discussed here.

The distinction between surface tracking and surface capturing varies in the literature. Some authors use 'surface tracking' to denote the methods that are called 'surface capturing with reconstruction' here.

exist [5]). If the water surface is not in the correct position, then this solution does not satisfy the dynamic boundary condition: the surface pressure is not constant (figure 1.4b). An improved estimate for the surface shape is derived from this non-constant pressure. Then a new grid is made with this shape (figure 1.4c), a new solution is computed, a new surface shape is estimated, and this process is repeated until both boundary conditions are satisfied. An improvement is to update the grid shape more often, each time before the flow solution is fully converged [39].

Surface fitting is a very fast solution technique, for three reasons. First, the grid always ends at the water surface; no cells are added above the surface and no computing time is used to solve the flow in these cells. Second, the flow field solver does not notice the free surface, the full free-surface conditions are only used in the grid deformation steps. Therefore, the flow field can be solved using all the fast techniques that have been developed for non-free-surface water or air flow. And finally, the grid near the free surface is aligned with the surface. Therefore, the boundary conditions there can be applied very accurately, which increases the overall accuracy of the solution. Thus, good solutions can be computed on relatively coarse grids.

The disadvantage of surface fitting is, that it is only useful for smooth ship hulls. It takes too much time to generate a completely new grid after each free-surface update, therefore the new grids are made by deforming the original grid. This deformation is difficult when the hull has sharp angles. For example, modelling a transom stern is hard when it is not known in advance whether the stern will be dry or wetted, because it is difficult to deform a grid with cells on the transom into one with no transom cells.

Thus, the surface fitting method is very efficient, but only for the selected cases where it is applicable.

1.2.2 Surface capturing For surface capturing, the grid is not deformed. Instead, the grid is continued above the water surface and a model of the water surface is added to the flow model.

The location of the water surface is determined as follows. By extrapolation or otherwise, the velocity field is extended to the cells above the surface, such that the velocity is continuous around the approximate location of the surface. Then a marker function ϕ is convected in this velocity field, i.e. we add

$$\boldsymbol{u} \cdot \nabla \phi = 0, \tag{1.1}$$

where $\boldsymbol{u}$ is the velocity field, to the flow equations. In this pure convection, ϕ follows the streamlines of the flow. Thus, by the kinematic condition, ϕ indicates the location of the water surface.

Several different surface capturing methods exist. The most recently developed capturing technique is the level set (LS) method [24, 48, 51, 69]. Here ϕ is specified on the inflow boundary of the flow field as the signed distance from the inflow water surface. So $\phi = 0$ on the inflow water surface, ϕ is positive above the surface and negative below it. If the velocity field is smooth, then ϕ stays smooth when it is convected through the whole domain. The value $\phi = 0$ follows the streamline

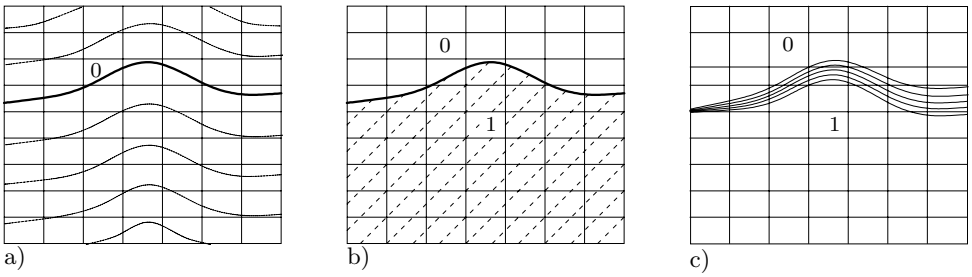


Figure 1.5. Surface capturing techniques. Level set, isolines of the LS function (a), volume-of-fluid with interface reconstruction (b), and VoF without reconstruction, isolines of the volume fraction (c).

starting at the inflow water surface. Therefore, in the whole domain, the $\phi = 0$ isoline indicates the water surface (figure 1.5a).

Since ϕ is smooth, it is easy to determine the position of the plane where $\phi = 0$. To couple the LS solution back to the flow field, the free-surface boundary conditions on the pressure and the flow field are imposed on this plane. The $\phi = 0$ isoline cuts through the cells, the application of boundary conditions in the interior of cells is the most difficult part of the LS technique.

An older alternative for the level set technique is the volume-of-fluid (VoF) method [27]. Here, ϕ has a more physical meaning: it is the fraction of each cell that is filled with water. Thus, $\phi = 1$ in cells below the surface, $\phi = 0$ in cells above the surface and ϕ lies between 0 and 1 in cells that contain the surface. On the inflow boundary, ϕ is a discontinuity (figure 1.5b).

Even though ϕ starts as a discontinuity, it is smeared out by numerical diffusion when it is convected with equation (1.1). (This effect is less important for the continuous LS function.) Therefore, many VoF methods use a geometric reconstruction [27, 30, 54, 66]. When ϕ is smeared out over a few cells, the location of the interface is reconstructed from this smeared ϕ and a new volume fraction is determined in each cell, based on this reconstruction. This volume fraction is no longer smeared (figure 1.5b). As with the level set method, free-surface boundary conditions are imposed on the reconstructed water surface.

For VoF, there is an alternative formulation that does not require free-surface boundary conditions. The flow can be computed both in the water and in the air above it. Then ϕ still indicates the amount of water in a cell, but the rest of the cell is filled with air (for reconstruction, it is considered empty). Now ϕ is computed with (1.1) but it is not reconstructed, so it smears out. Physically, this means that the sharp water surface is replaced by a mixture region: over a few cells, the fluid changes from pure water to pure air (figure 1.5c). The cells in between contain both water and air. This flow is fully determined by the flow equations plus (1.1) and boundary conditions on the edges of the domain: no boundary conditions are needed on the water surface.

The main advantage of all these surface capturing methods is, that they can handle arbitrary ship geometries because the grid is not deformed. Therefore, they are much more generally applicable than surface fitting. It is even possible to use unstructured grids; this is customary for surface capturing codes [21, 23, 24, 68]. A disadvantage of the reconstruction methods is, that it is difficult to impose boundary conditions accurately in the interior of cells. Also, the combination of the flow model with the completely different surface model forces most surface capturing methods to use relatively slow solvers. We get back to this in section 1.4, after the introduction of a very fast solution technique.

1.3 Multigrid solvers

One of the fastest solvers for flow equations available today is multigrid. By a solver, we mean a technique to find a flow solution, i.e. a flow field that satisfies the discretised flow equations on some grid plus the boundary conditions. In the next section, the specific problems of solving water flow equations are discussed. Here, we give a brief general introduction to multigrid solvers.

Multigrid (MG) is an iterative technique: it solves a system of equations $\mathcal{F}(\boldsymbol{q}) = 0$ by starting from an estimated initial state $\boldsymbol{q}^0$ and improving this state in a number of steps, until it satisfies the equations. The special feature of MG is, that this improvement is not only carried out on the computational grid itself, but also on a series of coarser grids with fewer grid cells, that are constructed from this grid. MG was initially developed for linear problems, it is now a mature technique that is applied successfully to both linear and nonlinear problems, including flow problems [19, 22, 31, 63, 80].

To see why multigrid iteration needs more than one grid, we consider classical iterative solution techniques, like point Jacobi or point Gauss-Seidel relaxation or time stepping. In one iteration step, the first two methods update the state cell by cell: in each cell, the state is set to such a value that the flow equations in that cell are satisfied, given fixed states in the neighbouring cells. For Jacobi, these states are the old states, for Gauss-Seidel they are the new states wherever these states are already available. For time stepping, the change in each cell is determined by the net inflow into that cell, this inflow is computed from the solution in the cell itself and in its neighbour cells. Thus, in each step of a classical solver, a cell exchanges information mainly with its neighbours, so changes in the state travel only one cell per iteration step.

Therefore, the classical solvers are ineffective for removing low-frequency errors. The error after an iteration is the difference between the state after that iteration and the final solution; the solver must remove this error. Using spatial Fourier decomposition, the error can be split into high-frequency and low-frequency components. The high-frequency errors vary rapidly in space, they are positive in one cell and negative in its neighbours. Information exchange between neighbour cells can quickly remove these errors, so the classical solvers are effective for removing the high-frequency errors. Low-frequency errors, on the other hand, are positive on one side of the domain and negative on the other side. The number of iterations needed to exchange information between cells on two sides of the domain is proportional

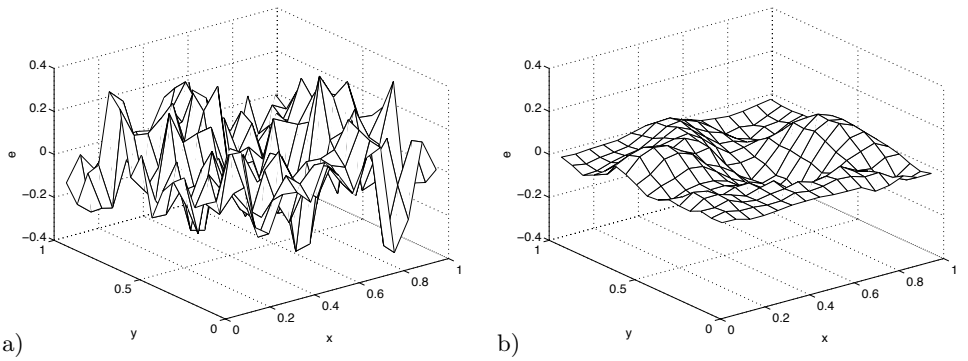


Figure 1.6. Gauss-Seidel smoothing applied to $\Delta\phi = 0$ with homogeneous Dirichlet boundary conditions, on a 16×16 cell grid. Random initial solution (a) and solution after two Gauss-Seidel iterations.

to the number of cells over the length of the grid, so this number increases when the grid gets finer. Therefore, low-frequency errors on fine grids are removed very slowly.

Figure 1.6 gives an illustration. Gauss-Seidel smoothing is applied to a problem with a random error. After two iteration steps, we see that the rapidly oscillating components have almost disappeared. The low-frequency part remains.

The coarser the grid, the more efficient the removal of low-frequency errors. That is why MG uses both coarse and fine grids; the high-frequency terms are removed with a solver on the finest grid, the low-frequency errors with solvers on coarser grids.

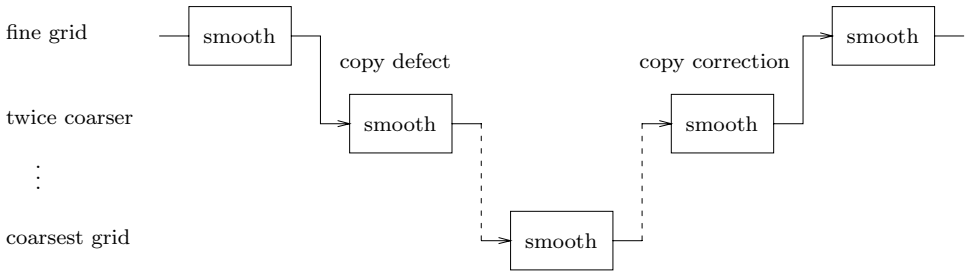


Figure 1.7. Multigrid, a single V-cycle iteration.

An iteration of the most common MG algorithm starts with one step of a classical solver (which is now called a smoother, since it is only used to remove high-frequency errors), on the finest grid (figure 1.7). Then the defect $\boldsymbol{d} = \mathcal{F}(\boldsymbol{q})$ is computed and copied to the next coarser grid. The smoother is applied to the solution on that grid, with the copied defect as a source term. A defect is computed on this grid, which is copied to the next coarser grid, and this is repeated until the coarsest grid is reached. Then, after smoothing, a correction is computed: this is the change in

the solution during this iteration step. This correction is copied to the next finer grid and added to the solution there. Because the correction is based on the defect that was copied from the next finer grid, it reduces the low-frequency errors on this finer grid. A new smoothing step is then applied to the finer grid, a correction is computed there and copied to the next finer grid. The final correction, copied to the finest grid, contains the effect of the smoothers on all the coarse grids.

Theoretically, the number of iterations that a MG solver needs to solve a problem is independent of the number of cells in the finest grid. This means that the amount of computer time needed to solve a problem increases linearly with the number of cells. For the classical solvers, this increase is at least quadratic.

We see that a successful MG method needs three things:

1. A good smoother, i.e. an efficient technique for removing high-frequency errors.

2. A sensible way to copy defects from fine to coarse grids and solution corrections from coarse to fine grids.

3. The flow equations on the coarse grids must resemble those on the fine grid. The equations on the coarse grids are used as an approximation to those on the finest grid; if they are not similar, then the corrections become meaningless.

1.4 Multigrid for surface capturing models

Multigrid is usually not applied directly to surface capturing discretisations. Instead, steady problems are solved by computing an unsteady solution, until it has converged to steady state. Unsteady water flow is usually computed with a time stepper in which the flow and surface updates are alternated. First, a time step is made for the flow, with the surface in a fixed location. Then the surface is updated by time stepping the VoF or LS equation (1.1), based on the new flow field. To compute these time steps, MG can be used (see e.g. [23, 24]).

But this solution technique is relatively slow. As pointed out in the previous section, the information exchange speed is reduced when the grids become finer. Therefore, time stepping in general is a slow technique. It is even worse for water flow, because gravity waves are hardly affected by viscosity. Thus, transient waves damp out very slowly; this further increases the number of time steps required. Still, the time stepping technique is often used for water flow, because it is generally believed to be the only possible technique.

This thesis shows that it is possible to solve surface capturing discretisations by directly applying MG to the steady flow equations and solving the flow field and the free surface together. In this way, solution speeds can be obtained that are similar to those of the fastest MG solvers for flows without free surfaces.

The only thing that is needed to make fast solution possible, is not using reconstruction of the surface. It is this reconstruction that makes it hard to fulfil the three requirements for a multigrid method. First, it is difficult to build a collective smoother when the interface is reconstructed. Such a smoother generally requires

derivatives of the defect in a cell with respect to the state in that cell. Therefore, in interface cells, the effect of all state variables on the reconstructed interface location is needed, plus the effect of the location on the defect. Computing these is very complicated, so alternately smoothing the flow and the free surface is the only option. This is not very efficient.

Because of the reconstruction, copying defects and corrections between coarse and fine grids is also very complex. The reconstruction is non-local: information in several cells is used collectively, to place the interface in some of these cells. It is very hard to translate this procedure into defects in individual cells, that can be copied to coarser grids. Also, it is difficult to define corrections when reconstruction is used on the coarse grids. Reconstruction sets the value of the marker function in all cells, based only on the location of the interface. Thus, in all non-interface cells, corrections that were copied from coarser grids to these cells may be lost. Therefore, the coarse grid corrections do not necessarily make sense.

And finally, in the interface region the coarse grid equations may be totally different from the fine grid equations. If the interface locations on a coarse and a fine grid differ by just one cell, then there are already cells where the velocity on one grid comes from the flow equations and on the other grid from extrapolation beyond the interface. Since it is never possible to get the interface exactly in the same location on two different grids, these situations always occur.

These problems disappear when VoF is used without interface reconstruction. The VoF equation (1.1) itself is a conservation law, it is of the same type as the flow equations. So for a water flow discretisation where we only add equation (1.1) to the flow equations and not reconstruct the interface, all equations are of the same type. Furthermore, all these equations are similar to the flow equations for flows without a free surface. Such a discretisation can be solved well with multigrid.

1.5 Outline

This thesis presents a VoF model without reconstruction and shows that it can be solved efficiently. All aspects of this model are studied; flow equations are developed that are adapted to multigrid solution and a multigrid method is constructed that is suitable for water flow.

Chapter 2 develops the foundation for the method, a first-order accurate model for 2D steady laminar flow. In the first part of this chapter, physical principles are used to derive a two-fluid VoF model as suggested in section 1.4, plus a stable finite-volume discretisation. In the second part, the multigrid solver is studied; Fourier analysis is used to select a smoother and a multigrid technique that form a good combination with the flow equations. And finally, results are given for three test problems. These confirm the efficiency of the method and provide insight in its practical application.

Shortened versions of this chapter are published as [77], and in conference proceedings as [73, 74].

Chapter 3 gives the extension of the method to turbulent flow: the change to the Reynolds-Averaged Navier-Stokes equations and the inclusion of a turbulence model. Using a turbulence model is essential for our method, because all water flows of practical interest are turbulent. The chapter shows how the MG method can be changed to solve the turbulent flow equations with similar efficiency as the laminar flow equations. Also, it is shown that the multigrid solution of the turbulent flow equations poses problems that are very similar to those caused by the two-fluid model. Therefore, the new multigrid method is effective for both.

The main part of this chapter appears as [79] (single-fluid results only). A shorter version including two-fluid results is included in the conference proceedings [75].

Chapter 4 concludes the discussion of the main method, giving the extension to second-order accuracy. Two aspects are discussed. The first part describes the use of a special, compressive discretisation for the volume fraction, to resolve the solution near the water surface as accurately as possible. The second part concerns the solution process, which is based on iterative improvement of the first-order accurate solution with defect correction, combined with multigrid. With the method now complete, two test cases show both its efficiency and its accuracy.

The contents of this chapter are contained in [78] and appear in the conference proceedings [76].

Chapter 5 describes work together with Dr. Takanori Hino, while the author was at the National Maritime Research Institute in Tokyo, Japan. Although it concerns a separate project, it is closely connected with the research in the previous chapters. It further investigates the accuracy of surface capturing methods: using NMRI's level set code, the accuracy of ship flow simulations is studied. But, as opposed to chapter 4, the focus is on the effect of the grid near the water surface on the accuracy. As a reference, the solutions are compared with the results of a surface fitting method, extending the discussion in section 1.2. Using this analysis, a grid refinement method is developed that is specially adapted for ship flow computations.

The research in this chapter will be published in the near future.

Chapter 6, the final chapter, introduces a second research topic that is related to the main research in this thesis. The ideas in chapter 2 for modelling two-fluid flow are used for compressible two-fluid flow. The result is a physical concept model for compressible flow, that gives insight in existing two-fluid models and leads to an alternative formulation of these models, with numerical advantages. Furthermore, the model gives a guideline for treating shock waves in two-fluid media exactly. Results show the validity of the model in continuous and discontinuous flow.

Early versions of this chapter (with an incomplete physical model, see section 6.2.3) have appeared as [72], and in conference proceedings as [71].

Chapter 7 gives a summary of the main conclusions in this thesis and opportunities for future research.

First-order accurate laminar model

This chapter introduces the basis of our method, a first-order accurate surface capturing discretisation for laminar steady water–air flow and its multigrid solution technique. The method is developed in three parts. First, in section 2.1, a physical model is proposed for the water–air flow and this model is used to derive continuous flow equations. As explained in the Introduction, this model is chosen such that it can be solved with multigrid.

The second part is the discretisation of the continuous flow equations. We use a finite-volume method on curved quadrilateral meshes. The flow equations are split into a convective part, a diffusive part, and a gravity term; the convective part is discretised with the aid of artificial compressibility [10, 11, 12]. Section 2.2 explains the discretisation, focusing on the aspects that are specific for the water–air flow: the use of variable densities in the artificial-compressibility fluxes, the modelling of gravity, and the boundary conditions.

The third part of the method is the multigrid solver. Section 2.3 introduces the multigrid method and the smoother; furthermore it addresses the specific issues that arise when this solver is applied to the highly nonlinear flow equations. In section 2.4, the solver is optimised for water–air flow using Fourier analysis.

Finally, in section 2.5, a numerical test of the method is presented. We study the accuracy of the discretisation, the water surface model, and the influence of the boundary conditions. Furthermore, the performance of the multigrid solver and the factors that influence the MG convergence are determined. We show that the method gives excellent solution speeds.

2.1 Flow equations

The first step in constructing a two-fluid solution method is the formulation of continuous flow equations. Here, we specifically want a system that can be solved well with multigrid. As shown in section 1.4, this can be done by choosing a model that consists of conservation laws only and does not use a geometric reconstruction of the water surface. Such a model closely resembles a single-fluid model, so it can be solved with the multigrid techniques available for single-fluid flow.

To find flow equations in which the water surface does not appear explicitly, we base the equations on a physical model for a mixture fluid: the fluid everywhere in the domain is modelled as if it is a mixture of water and air. When the inflow boundary conditions specify inflow of pure water and pure air only, then we get a flow field with a sharp water surface, where the 'mixture' is in fact pure water below the surface and pure air above it. But the model itself does not see this, the flow equations are the same everywhere in the domain. When the model is discretised, the surface appears as a smooth transition from water to air, over a few cells.

We assume that the fluid is a mixture on the macroscopic scale only. On the microscopic scale, the fluids do not mix. Instead, the fluid is divided into small

elements of the pure fluids; these elements have equal pressures and velocities. Thus, we get a mixture fluid that moves as a single fluid, the two fluid components do not move with respect to each other. But the fluid elements have the densities of the pure fluids, which are not necessarily equal. This way, the density is well defined in the whole domain and an average ('bulk') density can be easily found.

The flow we consider is incompressible Navier-Stokes (NS) flow, which includes friction and gravity effects. The two pure fluids have constant densities, since the flow is incompressible. However, the bulk density of the mixture is not constant. This has to be taken into account in the flow equations.

Derivation of the flow equations starts with the full compressible NS equations in two dimensions, which allow variable densities. These equations are valid for the mixture fluid defined above, because they are derived without any assumptions about the type of the flow medium. To distinguish between the two fluids, one extra equation is added, a mass conservation equation for the water only. The resulting system is:

$$\begin{aligned}
\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}\left(p + \rho u^2\right) + \frac{\partial}{\partial y}(\rho u v) &= \frac{\partial}{\partial x}(\mu u_x) + \frac{\partial}{\partial y}(\mu u_y), \\
\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho u v) + \frac{\partial}{\partial y}\left(p + \rho v^2\right) &= \frac{\partial}{\partial x}(\mu v_x) + \frac{\partial}{\partial y}(\mu v_y) - \rho g, \\
\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) &= 0, \\
\frac{\partial}{\partial t}(\rho_w \alpha) + \frac{\partial}{\partial x}(\rho_w u \alpha) + \frac{\partial}{\partial y}(\rho_w v \alpha) &= 0,
\end{aligned} \tag{2.1}$$

where α is the volume fraction of water: in the model above, the part of the volume occupied by elements of water. This system is not closed. For real compressible flow, we would add energy equations and equations of state for each of the two fluids (see chapter 6). But here, the pure fluids are incompressible so a single equation for the bulk density ρ is enough to close the system:

$$\rho = \rho_w \alpha + \rho_a (1 - \alpha), \tag{2.2}$$

with ρ_w and ρ_a, the densities of the pure fluids, constant. This equation can be seen as the incompressible equivalent of an equation of state for the bulk fluid. Substituting it into (2.1) gives:

$$\begin{aligned}
\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}\left(p + \rho u^2\right) + \frac{\partial}{\partial y}(\rho u v) &= \frac{\partial}{\partial x}(\mu u_x) + \frac{\partial}{\partial y}(\mu u_y), \\
\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho u v) + \frac{\partial}{\partial y}\left(p + \rho v^2\right) &= \frac{\partial}{\partial x}(\mu v_x) + \frac{\partial}{\partial y}(\mu v_y) - \rho g, \\
\frac{\partial}{\partial x}(u) + \frac{\partial}{\partial y}(v) &= 0, \\
\frac{\partial}{\partial t}(\alpha) + \frac{\partial}{\partial x}(u \alpha) + \frac{\partial}{\partial y}(v \alpha) &= 0.
\end{aligned} \tag{2.3}$$

The density cannot be divided out in the two momentum equations. However, it disappears from both continuity equations. This result is generally true for incompressible flow (see [81], sect. 1.4). The bulk mass conservation equation has its standard form (no time derivative!) and mass conservation for the water reduces to a volume-of-fluid equation in conservation form (compare with equation (1.1)). This shows that, for incompressible two-fluid flow, the 'natural' model is VoF.

A definition of the bulk viscosity μ is needed too. One possibility is the arithmetic mean

$$\mu = \alpha\mu_w + (1-\alpha)\mu_a. \tag{2.4}$$

This definition of μ follows when we assume equal strains in the fluid elements (which is true, since the two fluids have equal velocities everywhere), compute the stresses in the pure-fluid elements with Hooke's law and average the stress.

The system to be solved here, for steady flow, is (2.3) without the time derivatives:

$$\begin{aligned}
\frac{\partial}{\partial x}\left(p+\rho u^2\right) + \frac{\partial}{\partial y}\left(\rho uv\right) &= \frac{\partial}{\partial x}\left(\mu u_x\right) + \frac{\partial}{\partial y}\left(\mu u_y\right), \\
\frac{\partial}{\partial x}\left(\rho uv\right) + \frac{\partial}{\partial y}\left(p+\rho v^2\right) &= \frac{\partial}{\partial x}\left(\mu v_x\right) + \frac{\partial}{\partial y}\left(\mu v_y\right) - \rho g, \\
\frac{\partial}{\partial x}\left(u\right) + \frac{\partial}{\partial y}\left(v\right) &= 0, \\
\frac{\partial}{\partial x}\left(u\alpha\right) + \frac{\partial}{\partial y}\left(v\alpha\right) &= 0.
\end{aligned} \tag{2.5}$$

This system satisfies our requirements: it is completely in conservation form and the only difference with the single-fluid NS equations is the weighting with ρ in the momentum equations and the additional continuity equation. As is shown in the remainder of this chapter, the system can indeed be solved efficiently with multigrid.

2.2 Finite-volume discretisation

The system (2.5) is discretised with a cell-centered finite-volume technique on curved, structured grids. Fluxes are based on a Riemann solver with artificial compressibility. All aspects of this discretisation, i.e. the flux functions, the source term, and the treatment of boundary conditions, are described and motivated in this section.

2.2.1 Finite-volume on curved grids We discretise the NS equations on structured curvilinear grids, with a technique similar to e.g. that used in [31]. A typical cell is shown in figure 2.1. All state variables $\boldsymbol{q}$ are stored in the cell centres.

Define the state $\boldsymbol{q}$, the fluxes $\boldsymbol{f}$ and $\boldsymbol{g}$, and the gravity term $\boldsymbol{z}$ as:

$$\boldsymbol{q} = \begin{pmatrix} u \\ v \\ p \\ \alpha \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} p+\rho u^2 - \mu u_x \\ \rho uv - \mu v_x \\ u \\ u\alpha \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} \rho uv - \mu u_y \\ p+\rho v^2 - \mu v_y \\ v \\ v\alpha \end{pmatrix}, \quad \boldsymbol{z} = \begin{pmatrix} 0 \\ -\rho g \\ 0 \\ 0 \end{pmatrix}, \tag{2.6}$$

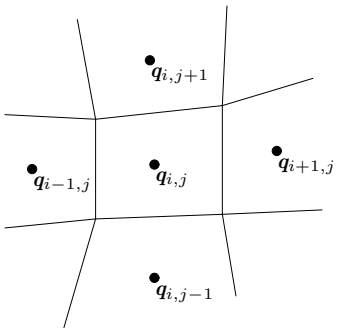


Figure 2.1. A typical cell $\Omega_{i,j}$, with four neighbours.

then the system (2.5) is written as:

$$\frac{\partial}{\partial x}\boldsymbol{f}(\boldsymbol{q}) + \frac{\partial}{\partial y}\boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{z}(\boldsymbol{q}). \tag{2.7}$$

Integrating this over a cell $\Omega_{i,j}$ yields:

$$\oint_{\partial\Omega_{i,j}} (\boldsymbol{f}n_x + \boldsymbol{g}n_y)\, ds = \iint_{\Omega_{i,j}} \boldsymbol{z}\, dx\, dy, \tag{2.8}$$

where $\partial\Omega_{i,j}$ denotes the cell faces of $\Omega_{i,j}$ and $(n_x, n_y)^T$ is the outward normal vector on the cell faces. This finite-volume equation is still exact. We discretise it by assuming $\boldsymbol{f}$, $\boldsymbol{g}$ constant over the cell faces and $\boldsymbol{z}$ constant in the cell:

$$\sum_{m=1}^{4} \left(\bar{\boldsymbol{f}}_{i,j,m} n_{x\,i,j,m} + \bar{\boldsymbol{g}}_{i,j,m} n_{y\,i,j,m}\right) l_{i,j,m} = A_{i,j}\bar{\boldsymbol{z}}_{i,j}. \tag{2.9}$$

Here $l_{i,j,m}$ is the length of the m^{th} cell face of $\Omega_{i,j}$, $A_{i,j}$ is the cell area. It is not necessary to compute the approximate fluxes $\bar{\boldsymbol{f}}$ and $\bar{\boldsymbol{g}}$ separately for each cell face, they can be combined to give a single flux over a cell face. Therefore we define the approximate, discretised flux $\boldsymbol{F}$ on cell face m as:

$$\boldsymbol{F}_{i,j,m} = \bar{\boldsymbol{f}}_{i,j,m} n_{x\,i,j,m} + \bar{\boldsymbol{g}}_{i,j,m} n_{y\,i,j,m}. \tag{2.10}$$

Face m is the same cell face as the one on the opposite side $m+2$ of the neighbour cell, so conservation requires:

$$\boldsymbol{F}_{i,j,m} = \boldsymbol{F}_{i(\pm1),j(\pm1),m\pm2}, \tag{2.11}$$

i.e. the flux out of cell $\Omega_{i,j}$ is equal to the flux into its neighbour cell. These fluxes are computed from the states in the cell centres, in two parts: we divide the flux vectors (2.6) in a convective and a diffusive part and compute these separately. The convective flux is discretised with an approximate Riemann solver based on artificial compressibility; the diffusive flux is computed with central differences.

The following sections describe these fluxes in detail and explain the incorporation of the gravity source term, as well as the choice of boundary conditions.

2.2.2 Convective fluxes An important aspect of water flows around ships is their very high Reynolds number. Thus, the influence of diffusion in the flow equations is low in most places. From a numerical perspective, this means that the diffusive part cannot be counted upon to stabilise the solution: we need a discretisation of the convective fluxes that is stable in itself. This is the main reason for discretising the convective and diffusive fluxes separately; thus, we can directly control the stability of the convective part.

To stabilise a convective flux discretisation, some type of numerical diffusion must be added. Also, we would like to have a strong pressure–velocity coupling in the flux function: when the local pressure and velocity influence each other directly, then an iterative error smoothing process (see section 2.4) becomes more robust.

Such convective fluxes are obtained with a Riemann solver. These stable flux functions have been used successfully for high-Reynolds Navier-Stokes flow [10, 11, 12, 63]. Furthermore, they can be used with Gauss-Seidel smoothing.

As Riemann solvers compute fluxes from elementary solutions of hyperbolic systems, they cannot be derived directly for the incompressible NS equations because the equations are not hyperbolic. Therefore, the flux functions are derived for an approximate system that is made hyperbolic with a technique called artificial compressibility. These flux functions are then used in the original, steady NS equations.

Below, it is shown how the artificial compressibility technique is adapted for the two-fluid system and how a flux function is derived from the resulting hyperbolic system.

Artificial compressibility. The time-dependent incompressible NS equations (2.3) cannot be used directly in a Riemann solver because the continuity equation has no time derivative. So an artificial time derivative is added to this equation. In the original artificial compressibility method [8], the resulting system is time-marched to convergence. The steady solution, with time-derivatives equal to zero, is equal to the solution of the steady NS equations. Here, we do not time-march. Instead, we use the modified system to define a flux function.

For artificial compressibility, it is assumed that the density of a fluid is still constant, but that it has a time derivative *in the continuity equations only*. Thus, for two fluids, the density is still given by (2.2), but now:

$$\rho_w,\ \rho_a \text{ have constant values,} \tag{2.12a}$$

$$\nabla\rho_w \equiv \nabla\rho_a \equiv 0, \tag{2.12b}$$

$$\rho_{w\,t} = \frac{p_t}{c_w^2}, \quad \rho_{a\,t} = \frac{p_t}{c_a^2} \quad \text{in the continuity equations.} \tag{2.12c}$$

The constants c_w and c_a have the dimension of velocity, they resemble the speed of sound c in compressible flow, where $c^2 = \frac{\partial p}{\partial \rho}$ for constant entropy. However, c_w and c_a are not sound speeds, as will be shown later on.

Now take the convective part of the compressible Navier-Stokes equations (2.1), i.e. ignore the diffusion and the gravity term, and substitute (2.2) and (2.12). The

resulting system has time derivatives in each equation:

$$\begin{aligned}
\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}\left(p + \rho u^2\right) + \frac{\partial}{\partial y}(\rho u v) &= 0,\\
\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho u v) + \frac{\partial}{\partial y}\left(p + \rho v^2\right) &= 0,\\
\left(\frac{\alpha}{\rho_w c_w^2} + \frac{1-\alpha}{\rho_a c_a^2}\right) p_t + u_x + v_y &= 0,\\
\alpha_t + \frac{\alpha}{\rho_w c_w^2} p_t + \frac{\partial}{\partial x}(u\alpha) + \frac{\partial}{\partial y}(v\alpha) &= 0.
\end{aligned} \tag{2.13}$$

The spatial derivatives are in conservation form, but the time derivatives are not; they cannot be written in conservation form either. However, this is not a problem, since the system is not going to be time-integrated anyway.

The c_w and c_a are parameters, they must be chosen somehow. But this choice is free, the parameters have no physical meaning. Therefore, we can make a special choice that simplifies the equations considerably. This choice is to set:

$$\rho_w c_w^2 = \rho_a c_a^2 = \rho c^2, \tag{2.14}$$

for some c. In that case, the last two equations of (2.13) reduce to:

$$\begin{aligned}
\frac{1}{\rho c^2} p_t + u_x + v_y &= 0,\\
\alpha_t + \frac{\alpha}{\rho c^2} p_t + \frac{\partial}{\partial x}(u\alpha) + \frac{\partial}{\partial y}(v\alpha) &= 0.
\end{aligned} \tag{2.15}$$

Characteristic analysis. The system (2.13) in one spatial dimension is hyperbolic, it has a set of four real-valued characteristic speeds and corresponding Riemann invariants. These are found by writing the system in primitive form, as yet without assumption (2.14):

$$\begin{aligned}
u_t + u u_x + v u_y + \frac{p_x}{\rho} &= 0,\\
v_t + u v_x + v v_y + \frac{p_y}{\rho} &= 0,\\
p_t + \frac{1}{\frac{\alpha}{\rho_w c_w^2} + \frac{1-\alpha}{\rho_a c_a^2}}(u_x + v_y) &= 0,\\
\alpha_t - \alpha(1-\alpha)\frac{\frac{1}{\rho_w c_w^2} - \frac{1}{\rho_a c_a^2}}{\frac{\alpha}{\rho_w c_w^2} + \frac{1-\alpha}{\rho_a c_a^2}} u_x + u\alpha_x + v\alpha_y &= 0,
\end{aligned} \tag{2.16}$$

and then considering only the x-derivatives:

$$\begin{pmatrix} u \\ v \\ p \\ \alpha \end{pmatrix}_t + \begin{bmatrix} u & 0 & \frac{1}{\rho} & 0 \\ 0 & u & 0 & 0 \\ \rho c_*^2 & 0 & 0 & 0 \\ D_c & 0 & 0 & u \end{bmatrix} \begin{pmatrix} u \\ v \\ p \\ \alpha \end{pmatrix}_x = 0. \tag{2.17}$$

The abbreviations are:

$$\rho c_*^2 = \frac{1}{\frac{\alpha}{\rho_w c_w^2} + \frac{1-\alpha}{\rho_a c_a^2}}, \qquad D_c = -\alpha(1-\alpha)\rho c_*^2 \left(\frac{1}{\rho_w c_w^2} - \frac{1}{\rho_a c_a^2} \right).$$

The characteristic speeds, the eigenvalues of the Jacobian matrix, are

$$\lambda^- = \tfrac{1}{2}u - \sqrt{c_*^2 + \left(\tfrac{1}{2}u\right)^2}, \quad \lambda^{0,1} = \lambda^{0,2} = u, \quad \lambda^+ = \tfrac{1}{2}u + \sqrt{c_*^2 + \left(\tfrac{1}{2}u\right)^2}. \tag{2.18}$$

Note that $\lambda^- < \lambda^0 < \lambda^+$, $\lambda^- < 0$ and $\lambda^+ > 0$, always. The Riemann invariants are found from the left eigenvector matrix, they are:

$$dJ^- = dp + \rho\lambda^- du, \qquad J^{0,1} = v, \qquad dJ^{0,2} = D_c\left(\rho u du + dp\right) - \rho c_*^2 d\alpha,$$
$$dJ^+ = dp + \rho\lambda^+ du. \tag{2.19}$$

These wave speeds and Riemann invariants are comparable with the results for the compressible Euler equations: there are two pressure characteristics, running left and right into the flow. However, there is no 'sound speed', the pressure waves have different velocities with respect to the flow. The tangential velocity v is convected with the flow, the volume fraction is not: there is an extra compression / expansion correction term in $dJ^{0,2}$, containing D_c. But under the assumption (2.14), D_c becomes zero. Then we find $J^{0,2} = \alpha$, the volume fraction is convected with the flow too. From now on, we shall assume that (2.14) holds.

Linearised Osher solver. The Riemann-solver flux uses the states on two sides of a cell face as left and right initial states $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ in a 1D shock tube flow problem, with the change between the states occurring at $x = 0$. After $t = 0$, waves start traveling from the discontinuity into the fluids, the flow with these waves can usually be computed analytically. For our system, two pressure waves appear: one always runs left and the other always runs right. In the middle, a contact discontinuity appears which runs either left or right, depending on the velocities. The output flux is set from the state $\boldsymbol{q}_{\frac{1}{2}}$ at $x = 0$, which always lies in between the pressure waves.

In approximate Riemann solvers, the solution of the Riemann problem is not computed exactly. We use Osher's approximation: the outer pressure waves in the Riemann problem are modelled as isentropic expansion or compression waves, which can be computed by integration of the Riemann invariants in (2.19). The pressure waves are weak, because incompressible NS solutions are continuous in pressure. Therefore, it is acceptable to linearise these waves. The pressure and the velocity between the waves are equal:

$$p_{\frac{1}{2}} = p_0 - \psi_0(u_{\frac{1}{2}} - u_0) = p_1 - \psi_1(u_{\frac{1}{2}} - u_1), \tag{2.20}$$

with the wave strengths:

$$\psi_0 = \rho_0 \left(\tfrac{1}{2}u_0 + \sqrt{c^2 + \left(\tfrac{1}{2}u_0\right)^2} \right), \qquad \psi_1 = \rho_1 \left(\tfrac{1}{2}u_1 - \sqrt{c^2 + \left(\tfrac{1}{2}u_1\right)^2} \right), \tag{2.21}$$

and thus,

$$\begin{aligned} u_{\frac{1}{2}} &= u_0 + \frac{p_1 - p_0 + \psi_1(u_1 - u_0)}{\psi_1 - \psi_0}, \\ p_{\frac{1}{2}} &= p_0 - \psi_0 \frac{p_1 - p_0 + \psi_1(u_1 - u_0)}{\psi_1 - \psi_0}. \end{aligned} \tag{2.22}$$

The volume fraction α and the tangential velocity v are convected with the middle wave, so they are discretised purely upwind:

$$\begin{aligned} v_{\frac{1}{2}} &= v_0, \qquad \alpha_{\frac{1}{2}} = \alpha_0 \qquad \text{if } u_{\frac{1}{2}} \geq 0, \\ v_{\frac{1}{2}} &= v_1, \qquad \alpha_{\frac{1}{2}} = \alpha_1 \qquad \text{if } u_{\frac{1}{2}} < 0. \end{aligned} \tag{2.23}$$

Now that the state $\boldsymbol{q}_{\frac{1}{2}}$ is known, the flux $\boldsymbol{F}$, the output of the Riemann solver, is computed from this state.

The artificial compressibility parameter c can be chosen arbitrarily. In practice, a useful definition for c was found to be:

$$\rho c^2 = C^2 = \text{constant}. \tag{2.24}$$

Also, to make the flux function more stable during smoothing, it is useful to compute both wave strengths with the same, averaged density:

$$\bar{\rho} = \frac{\rho_0 + \rho_1}{2}. \tag{2.25}$$

Therefore, (2.21) is replaced by:

$$\psi_0 = \bar{\rho}\left(\tfrac{1}{2}u_0 + \sqrt{C^2/\bar{\rho} + \left(\tfrac{1}{2}u_0\right)^2}\right), \qquad \psi_1 = \bar{\rho}\left(\tfrac{1}{2}u_1 - \sqrt{C^2/\bar{\rho} + \left(\tfrac{1}{2}u_1\right)^2}\right). \tag{2.26}$$

The average density $\bar{\rho}$ is only used in the wave strengths; for the density $\rho_{\frac{1}{2}}$ in the flux, we still use the upwind-defined density.

This Riemann solver is similar to some existing solvers [10, 15], although it is derived differently. Actually, for incompressible Navier-Stokes flow with artificial compressibility, most Riemann solvers resemble each other. The pressure waves are modeled in similar ways and besides, they are usually weak since NS flow is smooth. In shear layers, the jump in tangential velocity may be significant, but nearly all Riemann solvers compute these states in the same way, purely upwind.

Curved meshes. In a general curved quadrilateral mesh, a cell face may have any orientation with respect to the reference axes (x, y). But the Riemann solver computes fluxes normal and tangential to the cell faces. Therefore, the input states must be transformed to a cell face coordinate frame first. Then the Riemann flux is computed and this flux is transformed back to the standard reference frame.

2.2.3 Comments on convective flux In the previous section, the convective flux function is derived from physical arguments. It is worthwhile to take a step back from this derivation and make a short analysis of the resulting flux function.

Numerical viscosity. The convective flux, with its artificial compressibility, can be seen as a hyperbolisation of a non-hyperbolic system. It can also be interpreted as a central discretisation with a variable amount of numerical diffusion added. This becomes clear when we write the solution (2.22) of the Riemann problem as:

$$\begin{aligned} u_{\frac{1}{2}} &= \tfrac{1}{2}(u_0+u_1) + \tfrac{1}{2}\frac{\psi_1+\psi_0}{\psi_1-\psi_0}(u_1-u_0) + \frac{1}{\psi_1-\psi_0}(p_1-p_0), \\ p_{\frac{1}{2}} &= \tfrac{1}{2}(p_0+p_1) - \tfrac{1}{2}\frac{\psi_1+\psi_0}{\psi_1-\psi_0}(p_1-p_0) - \frac{\psi_0\psi_1}{\psi_1-\psi_0}(u_1-u_0). \end{aligned} \tag{2.27}$$

As we shall see in section 2.4.1, all these terms lead to positive numerical diffusion in the flux.

Effects of C. The constant C has a strong influence on the behaviour of the flux function. We compare two extreme cases: the effects of choosing C much larger or much smaller than the local velocity.

If C is small, i.e. $|u_0|, |u_1| \gg C/\sqrt{\bar\rho}$, then we get:

$$\begin{aligned} \psi_0 &\approx \bar\rho\left(\tfrac{1}{2}u_0 + \sqrt{(\tfrac{1}{2}u_0)^2}\right) = \tfrac{1}{2}\bar\rho\left(u_0 + |u_0|\right), \\ \psi_1 &\approx \bar\rho\left(\tfrac{1}{2}u_1 - \sqrt{(\tfrac{1}{2}u_1)^2}\right) = \tfrac{1}{2}\bar\rho\left(u_1 - |u_1|\right). \end{aligned}$$

If the velocities are positive, then $\psi_0 \approx \bar\rho u_0$ and $\psi_1 \approx 0$. Therefore

$$\begin{aligned} u_{\frac{1}{2}} &\approx u_0 - \frac{1}{\bar\rho u_0}(p_1-p_0), \\ p_{\frac{1}{2}} &\approx p_1. \end{aligned} \tag{2.28}$$

The velocity is discretised in an upwind way with a pressure term, the pressure itself is fully downwind. For comparison, there are schemes which use an upwind-biased velocity / downwind-biased pressure discretisation in the main flow direction [52]. Because of this combination, information still travels both upstream and downstream.

On the other hand, if $C/\sqrt{\bar\rho} \gg |u_0|, |u_1|$, then:

$$\begin{aligned} \psi_0 &\approx \sqrt{\bar\rho}C, \\ \psi_1 &\approx -\sqrt{\bar\rho}C, \end{aligned}$$

so

$$\begin{aligned} u_{\frac{1}{2}} &\approx \tfrac{1}{2}(u_0+u_1) - \frac{1}{2\sqrt{\bar\rho}C}(p_1-p_0), \\ p_{\frac{1}{2}} &\approx \tfrac{1}{2}(p_0+p_1) - \tfrac{1}{2}\sqrt{\bar\rho}C(u_1-u_0). \end{aligned} \tag{2.29}$$

Now both states are computed centrally, with diffusion terms in the other variable.

These results have two important consequences. First, they show the effect of globally varying C. For low C, the discretisation is close to upwind – downwind, for

higher C it becomes more central in the variables themselves. However, the numerical diffusion in the pressure actually increases for higher C, because the pressure $p_{\frac{1}{2}}$ has a velocity diffusion term scaling with $\sqrt{\overline{\rho}}C$. The higher diffusion for higher C is confirmed by numerical experiments. So choosing C very large is not smart, even though this makes the artificial compressibility system (2.15) equal to the original (2.3). In this case, the Riemann-solver approach is unrealistic (λ^- and λ^+ become infinite), which causes the high diffusion.

And second, the analysis shows that the type of the discretisation changes through the flow field. Near stagnation points and in fluxes normal to the main flow direction, the situation $C/\sqrt{\overline{\rho}} \gg |u|$ occurs, so the discretisation becomes central. In stagnation points, $u_1 - u_0$ is not small, therefore the numerical diffusion remains important too.

Pressure–velocity coupling. A criticism against the artificial compressibility technique is that, in both the pressure and velocity discretisations, the pressure and velocity diffusion terms are of the same order of magnitude (equation 2.27). Thus, in regions with low velocity and large pressure gradients, the pressure may start to dominate even the continuity equations, which should have no pressure dependence at all. Other flux functions, like AUSM+, do not have this property [67].

It is true that the Riemann solver is based on artificial physics. Therefore, in regions with high gradients, there may be a large difference between the states coming from the flux functions and the cell centre states. The latter satisfy the global flow equations, that are non-hyperbolic and contain friction. The largest differences occur near singularities in the flow, such as corners and stagnation points.

On the other hand, the errors in singular points are local. On grid refinement, they remain constant in the maximum norm, but their influence on the global solution becomes smaller. Thus, the problem disappears eventually. When a higher-order accurate scheme is used (chapter 4), the errors become even smaller. This gives us the confidence to use the artificial compressibility scheme.

2.2.4 Diffusive fluxes The diffusive part of (2.3) is discretised with central differences, which give a second-order accurate and stable discretisation. The only complication is the formulation on curved grids.

The diffusive terms in the finite-volume equations (2.8) may be rewritten as:

$$\oint_{\partial\Omega_{i,j}} -\mu \left(u_x n_x + u_y n_y\right)\, ds = \oint_{\partial\Omega_{i,j}} -\mu \frac{\partial u}{\partial n}\, ds, \tag{2.30}$$

i.e. the flux is the integral of the normal derivative of u over the cell faces. The same goes for v in the second momentum equation. This normal derivative is not available, it must be approximated. That can be done with an integration over a Peyret control volume. The control volume is as in figure 2.2. We get:

$$\iint_{\Omega_d} \frac{\partial u}{\partial n}\, d\Omega = \oint_{\partial\Omega_d} u\, dx_t, \tag{2.31}$$

where x_t is the tangential coordinate on the cell face. Approximating this integral

with an average derivative and average states on the control volume faces, it is found that

$$\frac{\partial u}{\partial n} \approx \frac{1}{A_d}\left(\frac{u_1+u_2}{2}(x_{t2}-x_{t1}) + u_3(x_{t4}-x_{t2}) + \frac{u_4+u_5}{2}(x_{t5}-x_{t4}) + u_i(x_{t1}-x_{t5})\right), \quad (2.32)$$

with $A_d \approx \frac{1}{4}\left(A_1 + A_2 + A_4 + A_5 + 2\left(A_3 + A_i\right)\right)$. This discretisation is second-order accurate for sufficiently smooth grids. Furthermore, it is relatively simple and reduces to the straightforward discretisation $\frac{\partial u}{\partial n} \approx (u_3 - u_i)/(x_{n3} - x_{ni})$ on uniform rectangular grids.

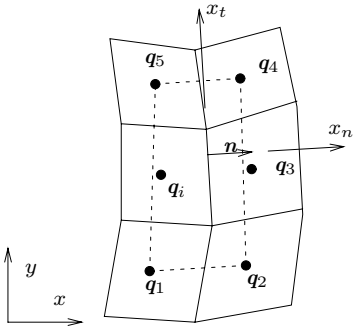


Figure 2.2. Control volume for diffusive flux discretisation.

2.2.5 Gravity The gravity term in the y-momentum equation is added as a source term $\bar{\boldsymbol{z}}_{i,j}A_{i,j} = -\rho_{i,j}gA_{i,j}$ to the sum of the fluxes in each cell. The gravity cannot be homogenised into the flux terms. When the interface is sharp instead of smeared out, the pressure is usually replaced by the hydrodynamic pressure $\varphi = p - \int_y^{y=0} \rho g dy$. In case of a sharp interface, this reduces to $\varphi = p + Fr^{-2}y$, see e.g. [39, 60]. Substituting this in equation (2.5) yields the Navier-Stokes equations without a source term, with p replaced by φ. But here, the interface is not sharp, so $p_x \neq \varphi_x$. Therefore, the homogenisation fails.

With the source term present, the cell-face states for the Riemann solver must be adapted. This solver has a strong coupling between pressure and velocity. But in a uniform horizontal flow with gravity, there is a vertical pressure gradient *without* a velocity gradient. So when simple first-order cell face states are fed to the Riemann solver, then there is a pressure jump across each lower – upper cell face. The Riemann solver reacts to this by specifying an (incorrect) vertical velocity at the cell face. We reduce this problem by subtracting the pressure gradient due to gravity from the input states for the Riemann solver. The local gravity effect is well-known, so we take:

$$\begin{aligned} p^{\text{lower}}_{i,j+\frac{1}{2}} &= p_{i,j} - \rho_{i,j}g(y_{i,j+\frac{1}{2}} - y_{i,j}), \\ p^{\text{upper}}_{i,j+\frac{1}{2}} &= p_{i,j+1} + \rho_{i,j+1}g(y_{i,j+1} - y_{i,j+\frac{1}{2}}). \end{aligned} \quad (2.33)$$

Thus, in a horizontal uniform flow, the Riemann solver sees no pressure jumps (in that case, the pressures in (2.33) are equal) and computes zero vertical velocities. In other flows, the erroneous vertical velocities are reduced an order of magnitude.

2.2.6 Boundary conditions The convective and diffusive parts of the NS equations may require a different number of boundary conditions. For the convective part, with the artificial compressibility flux function, the number of boundary conditions is determined by the number of characteristic waves running into the flow domain (equation (2.18)). For the diffusive part, the number is always two: both velocity components or their derivatives have to be specified. With the discretisation described in the previous sections, it is easy to impose different boundary conditions on the convective and diffusive operator.

Below, boundary conditions are given for different types of boundaries. All the equations are for a boundary on a right cell face.

Inflow. At an inflow boundary, three characteristics run into the flow: λ^-, $\lambda^{0,1}$, and $\lambda^{0,2}$. Therefore, three convective boundary conditions, u_b, v_b, and α_b are specified:

$$u_{\frac{1}{2}} = u_b, \qquad p_{\frac{1}{2}} = p_0 - \psi_0(u_b - u_0), \qquad v_{\frac{1}{2}} = v_b, \qquad \alpha_{\frac{1}{2}} = \alpha_b. \tag{2.34a}$$

Two diffusive boundary conditions; the velocities u_b and v_b are used here too:

$$(\mu u_x)_{\frac{1}{2}} = \mu_{\frac{1}{2}} \frac{u_b - u_0}{\frac{1}{2}\Delta x}, \qquad (\mu v_x)_{\frac{1}{2}} = \mu_{\frac{1}{2}} \frac{v_b - v_0}{\frac{1}{2}\Delta x}. \tag{2.34b}$$

Outflow. On an outflow boundary, there is one ingoing characteristic, the pressure wave λ^-. Thus one convective boundary condition, p_b, is specified. This Dirichlet boundary condition (pressure itself, not a normal derivative) improves the convergence of the multigrid solver.

$$u_{\frac{1}{2}} = u_0 - \frac{p_b - p_0}{\psi_0}, \qquad p_{\frac{1}{2}} = p_b, \qquad v_{\frac{1}{2}} = v_0, \qquad \alpha_{\frac{1}{2}} = \alpha_0. \tag{2.35a}$$

There is no physical guideline for the choice of the diffusive boundary conditions. It was decided to impose weak conditions on the velocity derivatives:

$$(\mu u_x)_{\frac{1}{2}} = \mu_{\frac{1}{2}} \frac{u_{\frac{1}{2}} - u_0}{\frac{1}{2}\Delta x}, \qquad (\mu v_x)_{\frac{1}{2}} = 0. \tag{2.35b}$$

No-slip wall. At a wall, the middle wave speeds are zero. Therefore, it is not clear how many characteristics run into the flow. However, the choice of v_b and α is completely arbitrary, because $u_b = 0$ makes the flux of these variables identical to zero. We choose:

$$u_{\frac{1}{2}} = 0, \qquad p_{\frac{1}{2}} = p_0 + \psi_0 u_0, \qquad v_{\frac{1}{2}} = v_0, \qquad \alpha_{\frac{1}{2}} = \alpha_0. \tag{2.36a}$$

The two diffusive boundary conditions follow from the zero velocity at the wall:

$$(\mu u_x)_{\frac{1}{2}} = -\mu_{\frac{1}{2}} \frac{u_0}{\frac{1}{2}\Delta x}, \qquad (\mu v_x)_{\frac{1}{2}} = -\mu_{\frac{1}{2}} \frac{v_0}{\frac{1}{2}\Delta x}. \tag{2.36b}$$

Slip wall / symmetry plane. The convective part of a slip wall is treated the same as a no-slip wall:

$$u_{\frac{1}{2}} = 0, \qquad p_{\frac{1}{2}} = p_0 + \psi_0 u_0, \qquad v_{\frac{1}{2}} = v_0, \qquad \alpha_{\frac{1}{2}} = \alpha_0. \tag{2.37a}$$

Two diffusive boundary conditions: zero normal velocity and zero tangential velocity derivative (symmetry!)

$$(\mu u_x)_{\frac{1}{2}} = -\mu_{\frac{1}{2}} \frac{u_0}{\frac{1}{2}\Delta x}, \qquad (\mu v_x)_{\frac{1}{2}} = 0. \tag{2.37b}$$

Water surface. Without a geometric reconstruction, the flow equations (2.5) do not contain the idea of a water surface; in principle, they can model any mixture of water and air. The water surface in the solution is obtained by specifying the correct boundary conditions.

The water surface is created at the inflow boundary. The α inflow condition is discontinuous, $\alpha = 1$ below the desired inflow water height and $\alpha = 0$ above this height. Thus, the exact solution for α in the whole domain is a contact discontinuity, the fluid is either pure water or pure air and the solution represents a physically correct water–air flow. In a solution of the discretised model, the surface is smeared by numerical diffusion, but this smearing disappears on grid convergence.

On the other hand, the location of the water surface in the domain is mostly determined by the outflow water height (in an experimental water channel, this height is usually set with an adjustable flap at the end of the channel). Only when the whole flow is supercritical (see section 2.5.1) does the inflow water height determine the whole flow. The most straightforward way to set the outflow water level is to set the outflow pressure boundary condition equal to the hydrostatic pressure, given the desired water level. The flow then adapts the water level to fit this pressure. These boundary conditions are further discussed when numerical results are analysed, in section 2.5.1.

2.3 Multigrid and smoothing

The multigrid method that is used to solve the discretised flow equations from section 2.2, is described here. Section 2.3.1 formally describes the MG technique, that was already introduced in section 1.3. Then section 2.3.2 describes two possible smoothers, the point and line Gauss-Seidel techniques. In section 2.4, the MG method is optimised for our two-fluid model using linearised flow equations and Fourier analysis; here, the final section 2.3.3 describes the specific requirements on the multigrid method due to the highly nonlinear character of the two-fluid flow equations.

2.3.1 Multigrid algorithm The multigrid method iteratively removes high-frequency errors on the fine grid and lower-frequency errors on a series of underlying coarser grids. Many variants exist for different applications [19, 63, 80]. Here, the method is constructed for a cell-centered finite-volume discretisation on curvilinear structured meshes (section 2.2.1). The finest grid is called Ω_K. A set of underlying

coarse grids Ω_k with $0 \leq k \leq K-1$ is made by merging 2×2 blocks of cells in the next finer grid into single cells, so each cell $(\Omega_k)_{i,j}$ in grid Ω_k corresponds to four cells $(\Omega_{k+1})_{2i(+1),2j(+1)}$ in the next finer grid Ω_{k+1}.

Denote the state on a grid k by $\boldsymbol{q}_k$ and the Navier-Stokes equations (2.9), discretised on this grid, by the operator $\mathcal{F}_k$:

$$(\mathcal{F}_k \boldsymbol{q}_k)_{i,j} = \sum_{m=1}^{4} \left(\boldsymbol{F}_{\mathrm{conv}\,i,j,m} + \boldsymbol{F}_{\mathrm{diff}\,i,j,m} \right) l_{i,j,m} - A_{i,j} \bar{\boldsymbol{z}}^k_{i,j}. \tag{2.38}$$

Then the general problem on each grid is $\mathcal{F}_k \boldsymbol{q}_k = \boldsymbol{s}_k$, for some source term $\boldsymbol{s}_k$. The final problem to be solved is $F_K \boldsymbol{q}_K = 0$. We call the smoothing operator (to be defined in section 2.3.2) M_k and introduce a prolongation operator P^k_{k-1} that moves a solution from one cell on grid $k-1$ to the four cells on grid k that lie in the same location:

$$(\boldsymbol{q}_k)_{2i(+1),2j(+1)} = P^k_{k-1}(\boldsymbol{q}_{k-1})_{i,j} = (\boldsymbol{q}_{k-1})_{i,j}. \tag{2.39}$$

In the same way, a restriction operator R^{k-1}_k is defined for defects $\boldsymbol{d}$:

$$(\boldsymbol{d}_{k-1})_{i,j} = R^{k-1}_k (\boldsymbol{d}_k)_{2i(+1),2j(+1)} = (\boldsymbol{d}_k)_{2i,2j} + (\boldsymbol{d}_k)_{2i+1,2j} + (\boldsymbol{d}_k)_{2i,2j+1} + (\boldsymbol{d}_k)_{2i+1,2j+1}. \tag{2.40}$$

The nonlinear multigrid procedure (for iteration n) is defined recursively as follows. It is started on the finest grid, with $\boldsymbol{s}_K = 0$.

$\boldsymbol{q}^{n+1}_k =$ **recursive function** $NMG(k, \boldsymbol{q}^n_k, \boldsymbol{s}^n_k)$

$\tilde{\boldsymbol{q}}^n_k = (M_k)^{q_1}\, \boldsymbol{q}^n_k$ — q_1 pre-relaxation steps, (2.41a)

if $k \neq 0$ **then**

$\boldsymbol{d}^n_{k-1} = R^{k-1}_k \left(\mathcal{F}_k \tilde{\boldsymbol{q}}^n_k - \boldsymbol{s}^n_k \right)$ — defect on coarse grid, (2.41b)

$\boldsymbol{s}^n_{k-1} = \mathcal{F}_{k-1} \boldsymbol{q}^n_{k-1} - w^n_{k-1} \boldsymbol{d}^n_{k-1}$ — source term, (2.41c)

$\boldsymbol{q}^{n+1}_{k-1} = NMG^{\sigma} \left(k-1, \boldsymbol{q}^n_{k-1}, \boldsymbol{s}^n_{k-1} \right)$ — σ MG steps on coarse grid, (2.41d)

$\tilde{\tilde{\boldsymbol{q}}}^n_k = \tilde{\boldsymbol{q}}^n_k + \dfrac{1}{w^n_{k-1}} P^k_{k-1} \left(\boldsymbol{q}^{n+1}_{k-1} - \boldsymbol{q}^n_{k-1} \right)$ — prolongation of correction, (2.41e)

end if

$\boldsymbol{q}^{n+1}_k = (M_k)^{q_2}\, \tilde{\tilde{\boldsymbol{q}}}^n_k$ — q_2 post-relaxation steps. (2.41f)

We mostly use $q_1 = q_2 = 1$. Either $\sigma = 1$ (V-cycle) or $\sigma = 2$ (W-cycle) can be used, σ is chosen based on the experiments in section 2.5.3. Figure 1.7 shows a method with $q_1 = q_2 = 1$ and $\sigma = 1$. The choice of the scaling parameter w^n_{k-1} and the initial solution $\boldsymbol{q}^n_{k-1}$, are explained in section 2.3.3.

The MG procedure is used in a full multigrid (FMG) framework: we actually start on the coarsest grid, find a converged solution on this grid, then prolongate this solution to the next finer grid, use it as an initial solution to find a converged solution there, etc. FMG gives a good initial solution on the finest grid, so it reduces the required number of (expensive) fine grid iterations.

2.3.2 Smoothers For the removal of high-frequency errors, two smoothers are considered: the collective point and line Gauss-Seidel techniques. The smoothers are described here, their properties are analysed in section 2.4. For the discussion, we limit ourselves to five-point stencils; the operator in each cell uses only the state in that cell and in its four neighbours.

Point Gauss-Seidel. Point Gauss-Seidel smoothing is one of the classical iterative solution techniques for discretised PDE's. In each step, the solution is updated cell by cell. The state in each cell is changed such, that the residual in that cell becomes zero, given the current state in all other cells. These states are the old states in cells that have not been updated yet and the new, updated states in all other cells. Lexicographic ordering is used, which means that the cells are updated column by column or row by row.

Consider a cell $(\Omega_k)_{i,j}$ after the n^{th} iteration. If the direction of the sweep is upwards and to the right, then the lower and left neighbour cell have already been changed to state $n+1$. We want to find $(\boldsymbol{q}_k^{n+1})_{i,j}$ such, that:

$$\mathcal{F}_k\left((\boldsymbol{q}_k^{n+1})_{i,j}, (\boldsymbol{q}_k^{n+1})_{i-1,j}, (\boldsymbol{q}_k^{n+1})_{i,j-1}, (\boldsymbol{q}_k^{n})_{i+1,j}, (\boldsymbol{q}_k^{n})_{i,j+1}\right) = (\boldsymbol{s}_k^n)_{i,j}. \tag{2.42}$$

For nonlinear collective smoothing, this means solving a set of four nonlinear equations in the four state variables $(\boldsymbol{q}_k^{n+1})_{i,j}$. This is done with a numerical root finder, the matrix – vector version of the Newton-Raphson (NR) method. From the initial state $\boldsymbol{q}_{\text{NR}}^0 = (\boldsymbol{q}_k^n)_{i,j}$, each NR iteration step is defined as:

$$\boldsymbol{q}_{\text{NR}}^{l+1} = \boldsymbol{q}_{\text{NR}}^{l} - \left(\frac{\partial \mathcal{F}_k}{\partial \boldsymbol{q}_k}\right)_{i,j}^{-1} \left((\mathcal{F}_k)_{i,j}\left(\boldsymbol{q}_{\text{NR}}^{l}\right) - (\boldsymbol{s}_k^n)_{i,j}\right). \tag{2.43}$$

On convergence ($\left\|(\mathcal{F}_k)_{i,j}\left(\boldsymbol{q}_{\text{NR}}^{l}\right)\right\| < \epsilon$), we set $(\boldsymbol{q}_k^{n+1})_{i,j} = \boldsymbol{q}_{\text{NR}}^{l}$. The derivative $\frac{\partial \mathcal{F}_k}{\partial \boldsymbol{q}_k}$ is the sum of the convective and diffusive flux derivatives over the four cell faces, plus the α-derivative of the gravity source term. For the fluxes, the gravity correction (2.33) must be taken into account.

The process can be changed by under / overrelaxation, i.e. introducing a parameter $\omega > 0$ and choosing:

$$(\boldsymbol{q}_k^{n+1})_{i,j} = (\boldsymbol{q}_k^{n})_{i,j} + \omega\left(\boldsymbol{q}_{\text{NR}}^{l} - (\boldsymbol{q}_k^{n})_{i,j}\right). \tag{2.44}$$

For nonlinear Gauss-Seidel, underrelaxation ($\omega < 1$) is the usual choice. This will be explained in section 2.4.3.

So the Gauss-Seidel smoothing contains two, nested iterative processes. The outer iteration is the Gauss-Seidel sweep over all cells, the inner iteration is the solution of (2.42) in each cell, with Newton-Raphson. The latter process converges very fast, usually in one or two steps. The outer iteration converges slowly, if it is not accelerated with multigrid.

Line Gauss-Seidel. An alternative to point Gauss-Seidel is line Gauss-Seidel smoothing. The principle is the same, but now the states are adapted in a whole

row or column at the time, not in a single cell. This makes the smoother much more powerful.

We solve (for row smoothing, marching in positive y-direction):

$$\mathcal{F}_k\left((\boldsymbol{q}_k^{n+1})_{i,j},(\boldsymbol{q}_k^{n+1})_{i-1,j},(\boldsymbol{q}_k^{n+1})_{i,j-1},(\boldsymbol{q}_k^{n+1})_{i+1,j},(\boldsymbol{q}_k^{n})_{i,j+1}\right)=(\boldsymbol{s}_k^n)_{i,j},\quad \forall i\in[1,i_{\max}]\,. \tag{2.45}$$

This is a nonlinear system in $4i_{\max}$ unknowns. Like the point Gauss-Seidel system, it is solved with vector Newton-Raphson. But now, the state and residual vector are:

$$\boldsymbol{q}_{\mathrm{NR}}^0=\begin{pmatrix}(\boldsymbol{q}_k^n)_{1,j}\\ \vdots\\ (\boldsymbol{q}_k^n)_{i_{\max},j}\end{pmatrix},\quad \mathcal{F}_{\mathrm{line}}(\boldsymbol{q}_{\mathrm{NR}}^0)-(\boldsymbol{s}_k^n)_{\mathrm{line}}=\begin{pmatrix}(\mathcal{F}_k)_{1,j}-(\boldsymbol{s}_k^n)_{1,j}\\ \vdots\\ (\mathcal{F}_k)_{i_{\max},j}-(\boldsymbol{s}_k^n)_{i_{\max},j}\end{pmatrix}. \tag{2.46}$$

The derivative matrix is:

$$\left(\frac{\partial\mathcal{F}_k}{\partial\boldsymbol{q}_k}\right)_{\mathrm{line}}=\begin{bmatrix}\left(\frac{\partial\mathcal{F}_k}{\partial\boldsymbol{q}_k}\right)_{1,j} & \frac{\partial(\mathcal{F}_k)_{1,j}}{\partial(\boldsymbol{q}_k)_{2,j}} & & 0\\ \frac{\partial(\mathcal{F}_k)_{2,j}}{\partial(\boldsymbol{q}_k)_{1,j}} & \left(\frac{\partial\mathcal{F}_k}{\partial\boldsymbol{q}_k}\right)_{2,j} & \ddots & \\ & \ddots & \ddots & \frac{\partial(\mathcal{F}_k)_{i_{\max}-1,j}}{\partial(\boldsymbol{q}_k)_{i_{\max},j}}\\ 0 & & \frac{\partial(\mathcal{F}_k)_{i_{\max},j}}{\partial(\boldsymbol{q}_k)_{i_{\max}-1,j}} & \left(\frac{\partial\mathcal{F}_k}{\partial\boldsymbol{q}_k}\right)_{i_{\max},j}\end{bmatrix}. \tag{2.47}$$

This matrix is block-tridiagonal, because each cell has only two other cells in the line that influence it: its two neighbours. Therefore, $(\partial\mathcal{F}_k/\partial\boldsymbol{q}_k)_{\mathrm{line}}^{-1}\,\mathcal{F}_{\mathrm{line}}$ can be computed with a block Thomas algorithm; the amount of work for this direct algorithm is proportional to the number of cells in the line. Therefore, the amount of work needed for a line smoothing step on a row is proportional to the work needed for a point Gauss-Seidel iteration step in every cell of that row. So even for line Gauss-Seidel, the total work per smoothing step is proportional to the number of cells. Including flux computation, it is about two times more than for the point smoothing.

The computation of the derivatives in (2.47) requires exactly the same amount of work as the computation for point Gauss-Seidel smoothing: the $(\partial\mathcal{F}_k/\partial\boldsymbol{q}_k)_{i,j}$ appear again and the $\partial(\mathcal{F}_k)_{i,j}/\partial(\boldsymbol{q}_k)_{i\pm1,j}$ are derivatives of the fluxes over a single cell face, that require no extra computation. The flux out of cell $\Omega_{i,j}$ is the flux into cell $\Omega_{i\pm1,j}$ (equation (2.11)), so the derivatives are the same too:

$$\frac{\partial\boldsymbol{F}_{i,j,m}}{\partial\boldsymbol{q}_{i\pm1,j}}=-\frac{\partial\boldsymbol{F}_{i\pm1,j,m\mp2}}{\partial\boldsymbol{q}_{i\pm1,j}}, \tag{2.48}$$

which is already computed, since it is needed to find $(\partial\mathcal{F}_k/\partial\boldsymbol{q}_k)_{i\pm1,j}$.

All the equations above describe a smoothing by rows. If j is chosen as the running index, we get the equivalent smoothing by columns. When row and column sweeps are mixed, we have the so-called alternating line Gauss-Seidel smoothing.

2.3.3 Solving nonlinear equations Our discretised system of flow equations is highly nonlinear. The artificial compressibility flux functions have quadratic terms in the pressure and rational terms in the normal velocities. The volume fraction α complicates the system too: changes in the density change the pressure–velocity coupling (section 2.2.3). Also, the volume fraction at the cell faces is discretised upwind, so $\alpha_{\frac{1}{2}}$ in the Riemann solver depends discontinuously on the normal velocity.

Although multigrid is basically a linear technique (the restrictions and prolongations are linear), it is suitable for solving nonlinear equations. The main limitation comes from the flow equations themselves: because they are nonlinear, the discretised equations in one cell may not have a solution for arbitrary states in the other cells. Therefore, it is not always possible to find a solution of the equations (2.42) or (2.45), so the Gauss-Seidel smoothing does not always work.

This section describes how the MG procedure is adapted to make sure that the GS smoothers in the individual cells or lines always find a solution. When these solutions can be found, then GS is an effective smoother. And when the smoother works, the MG method works as a whole.

Allowable source terms. The source terms $\boldsymbol{s}_k^n$ in equation (2.42) or (2.45) can make the Gauss-Seidel smoothing impossible. The $\boldsymbol{s}_k^n$ contain the defects on the fine grids (equation (2.41c)). For some source terms, the GS equations have no solutions.

From practice, it appears that (2.42) and (2.45) have solutions for arbitrary neighbour states when $\boldsymbol{s}_k^n = 0$. This is proved for the volume fraction α: when mass is conserved in a cell, then that cell has at least one face with inflow and one face with outflow, and the inflow and outflow of mass are equal. With the upwind discretisation, the α's at the inflow faces do not depend on the state in the cell itself, but the α's at the outflow faces are equal to that state. So if we set α in a cell equal to the average value over the inflow faces, then water mass is conserved: the equation has a solution. Moreover, if $0 \leq \alpha \leq 1$ in the cells neighbouring the inflow faces, then $0 \leq \alpha \leq 1$ in the cell itself, after the update.

When the source term $\boldsymbol{s}_k$ is nonzero, then the existence of a solution to (2.42) or (2.45) is not guaranteed. Again, this is shown with α. For example, if a large negative source of mass is specified, we may get a cell with inflow on all cell faces. It is impossible to set the net inflow of water into such cells, since the flux of α is determined purely upwind: with inflow on all faces, it does not depend on α in the cells itself. Also, a source term could specify a sink for water in a cell whose neighbours contain only air. The only way to get a net inflow of water in that cell is to take a negative α in the cell itself. This may lead to negative densities, which disturb the pressure–velocity coupling. For multigrid, we have to allow negative α: if volume fractions on coarse grids are reset to a value between 0 and 1 after smoothing, then the wrong corrections are passed to the fine grids. The result is, that the multigrid method does not converge.

The solution is to keep the source terms small. The equations (2.42) and (2.45) have a solution for $\boldsymbol{s}_k = 0$, so it is likely that they also have solutions for $\boldsymbol{s}_k$ close to zero. In the multigrid algorithm from section 2.3.1, the two terms in the source

term (2.41c) are made small as follows:

$\mathcal{F}_{k-1}\boldsymbol{q}_{k-1}^n$: This term is reduced by selecting a good starting solution $\boldsymbol{q}_{k-1}^n$ on the coarse grids. The ideal $\boldsymbol{q}_{k-1}^n$ is that state, for which $\mathcal{F}_{k-1}\boldsymbol{q}_{k-1}^n \approx 0$. This state is available, as we use a FMG procedure: the first solution $\boldsymbol{q}_k^0$ on each grid k, from grid 1 upwards, is set by solving $\mathcal{F}_{k-1}\boldsymbol{q}_{k-1} = 0$ on the next coarser grid and prolongating this solution to grid Ω_k (section 2.3.1). As a bonus, we get $\boldsymbol{q}_{k-1}$ for which $\mathcal{F}_{k-1}\boldsymbol{q}_{k-1} \approx 0$. These states are used as the initial solution $\boldsymbol{q}_{k-1}^n$ for each coarse grid correction step.

$w_{k-1}^n \boldsymbol{d}_{k-1}^n$: This term is made small by choosing w_{k-1}^n small whenever $\boldsymbol{d}_{k-1}^n$ is large; the correction is unscaled through multiplication with $1/w_{k-1}^n$ (equation (2.41e)). The coarse grid correction does not depend much on the value of w, so a straightforward choice for w_{k-1}^n is used:

$$w_{k-1}^n = \min\left(1, \frac{1}{D \max\left|\boldsymbol{d}_{k-1}^n\right|}\right). \tag{2.49}$$

$D = 10^2$ works in practice. Also, more elaborate expressions for w are possible.

These choices for w and $\boldsymbol{q}_{k-1}^n$ slightly reduce the maximum convergence rate of the multigrid solver. Practice shows that it makes no difference whether $w = 0.1$ or $w = 0.001$, but that the MG convergence is a little better when $w = 1$. By taking $\boldsymbol{q}_{k-1}^n$ constant, the resemblance of the fine and the coarse grid operators is not optimal (see section 1.3). The best way to make the coarse grid operators resemble the fine grid operator is to restrict the fine grid state to the coarse grids for each step. But we cannot do that, as those states cause large source terms.

But despite these minor disadvantages, it is very useful to keep $\boldsymbol{s}_k^n$ small: this guarantees the proper functioning of the GS smoother.

Damped Newton-Raphson. Even when a solution to the Gauss-Seidel equations exists, it may be difficult to find. This is caused mainly by the volume fraction α. Above, it is shown that a solution for α in a single cell always exists when $\boldsymbol{s}_k^n = 0$. But this proof requires that the continuity equation in the cell is satisfied. When a Newton-Raphson solution is started, with $\boldsymbol{q}_{\mathrm{NR}}^0$, this is not yet true. In fact, in low-velocity regions, cells may appear where the ratio of inflow and outflow is far from one, or even cells where all the faces have inflow. Here, the flux derivatives with respect to α are almost zero. Under these circumstances, normal NR may not converge; damped Newton-Raphson gives a more stable iteration.

Damped Newton-Raphson still solves (2.42) or (2.45). However, (2.43) is modified to make smaller steps per iteration. The process resembles backward-Euler implicit time stepping in a single cell or line, with the fluxes linearised around the old state:

$$T_{i,j}\left(\boldsymbol{q}_{\mathrm{NR}}^{l+1} - \boldsymbol{q}_{\mathrm{NR}}^{l}\right) = (\mathcal{F}_k)_{i,j}\left(\boldsymbol{q}_{\mathrm{NR}}^{l}\right) + \left(\frac{\partial \mathcal{F}_k}{\partial \boldsymbol{q}_k}\right)_{i,j}\left(\boldsymbol{q}_{\mathrm{NR}}^{l+1} - \boldsymbol{q}_{\mathrm{NR}}^{l}\right) - (\boldsymbol{s}_k^n)_{i,j},$$

or

$$\boldsymbol{q}_{\mathrm{NR}}^{l+1} = \boldsymbol{q}_{\mathrm{NR}}^{l} - \left(\left(\frac{\partial \mathcal{F}_k}{\partial \boldsymbol{q}_k}\right)_{i,j} - T_{i,j}\right)^{-1}\left((\mathcal{F}_k)_{i,j}\left(\boldsymbol{q}_{\mathrm{NR}}^{l}\right) - (\boldsymbol{s}_k^n)_{i,j}\right). \tag{2.50}$$

$T_{i,j}$ is a time-derivative Jacobian matrix:

$$T_{i,j} = \frac{A_{i,j}}{\Delta t} \begin{bmatrix} \rho^l_{\mathrm{NR}} & 0 & 0 & u^l_{\mathrm{NR}}(\rho_w - \rho_a) \\ 0 & \rho^l_{\mathrm{NR}} & 0 & v^l_{\mathrm{NR}}(\rho_w - \rho_a) \\ 0 & 0 & \frac{1}{C^2_l} & 0 \\ 0 & 0 & \frac{\alpha_{\mathrm{NR}}}{C^2} & 1 \end{bmatrix}. \tag{2.51}$$

For $\Delta t \to \infty$, normal Newton-Raphson is recovered. Thus, the only change from the original NR method is, that the derivative matrix $(\partial \mathcal{F}_k / \partial \boldsymbol{q}_k)_{i,j}$ is replaced by $(\partial \mathcal{F}_k / \partial \boldsymbol{q}_k)_{i,j} - T_{i,j}$. The size of Δt determines the relative importance of the extra terms.

The damped Newton-Raphson procedure is started only for those lines where the normal Newton-Raphson fails (this usually occurs just a few times, in the first iteration steps). In most cases, the damped Newton-Raphson finds a solution for these lines.

2.4 Fourier analysis

For the analysis and optimisation of multigrid methods, local Fourier analysis is one of the most effective tools. This technique directly shows how fast the MG method reduces different errors. A detailed description of the method can be found in [63], chapter 4. In this section, Fourier analysis is used to determine the best smoother for the two-fluid system and the effectiveness of the coarse grid correction.

Fourier analysis is only suitable for linear equations. Therefore, the two-fluid flow equations are linearised in section 2.4.1. This linearisation is not only necessary for the Fourier analysis, it also gives insight in the numerical diffusion of the system. In the following sections, Fourier analysis is applied to point smoothing (section 2.4.2), to line smoothing (section 2.4.3), and to the complete multigrid solver (section 2.4.4).

2.4.1 Linearised equations To enable Fourier analysis, the nonlinear Navier-Stokes equations (2.38) are linearised. We choose a state $\boldsymbol{Q}_k$ on grid Ω_k and write states close to $\boldsymbol{Q}_k$ as $\boldsymbol{q}_k = \boldsymbol{Q}_k + \epsilon \tilde{\boldsymbol{q}}_k$, with ϵ small. Substitution in (2.38) gives the linearised NS operator L_k:

$$\mathcal{F}(\boldsymbol{Q} + \epsilon \tilde{\boldsymbol{q}}) = \mathcal{F}(\boldsymbol{Q}) + \epsilon L_k \tilde{\boldsymbol{q}} + \mathcal{O}(\epsilon^2). \tag{2.52}$$

When $\boldsymbol{Q}_k$ is a solution of $\mathcal{F}_k(\boldsymbol{q}_k) = 0$, then $\epsilon \tilde{\boldsymbol{q}}_k$ can be seen as a small disturbance around that solution. The removal of this disturbance with nonlinear MG is equivalent up to $\mathcal{O}(\epsilon^2)$ with applying linear MG to $L_k \tilde{\boldsymbol{q}}_k$. Therefore, the analysis of smoothing and MG on L_k accurately predicts the behaviour of the nonlinear solver for small errors.

For $\boldsymbol{Q}_k$ a uniform flow is taken: $\boldsymbol{Q}_k = [U, V, P, A]^T$, the uniform flow density is R. Then the state in cell $(\Omega_k)_{i,j}$ is:

$$u_{i,j} = U + \epsilon \tilde{u}_{i,j}, \quad v_{i,j} = V + \epsilon \tilde{v}_{i,j}, \quad p_{i,j} = P - gRy + \epsilon \tilde{p}_{i,j}, \quad \alpha_{i,j} = A + \epsilon \tilde{\alpha}_{i,j}. \tag{2.53}$$

The state (2.53) is substituted into (2.38) on a uniform grid with cell sizes Δx and

Δy, to find L_k. The linear system is written as a five-point stencil:

$$(L_k\tilde{\boldsymbol{q}}_k)_{i,j} = A_{0,0}(\tilde{\boldsymbol{q}}_k)_{i,j}+A_{-1,0}(\tilde{\boldsymbol{q}}_k)_{i-1,j}+A_{1,0}(\tilde{\boldsymbol{q}}_k)_{i+1,j}+A_{0,-1}(\tilde{\boldsymbol{q}}_k)_{i,j-1}+A_{0,1}(\tilde{\boldsymbol{q}}_k)_{i,j+1}. \tag{2.54}$$

The (constant) matrices are:

$$A_{-1,0} = \Delta y \begin{bmatrix} -RU - \frac{C^2+RU^2}{2\psi_x} - \frac{\mu}{\Delta x} & 0 & -\frac{1}{2} - 1\frac{1}{2}\frac{U}{2\psi_x} & -\Delta\rho U^2 \\ -\frac{1}{2}RV - \frac{1}{2}\frac{RUV}{2\psi_x} & -RU - \frac{\mu}{\Delta x} & -\frac{V}{2\psi_x} & -\Delta\rho UV \\ -\frac{1}{2} - \frac{1}{2}\frac{U}{2\psi_x} & 0 & -\frac{1}{2R\psi_x} & 0 \\ -\frac{1}{2}A - \frac{1}{2}\frac{UA}{2\psi_x} & 0 & -\frac{A}{2R\psi_x} & -U \end{bmatrix}, \tag{2.55a}$$

$$A_{1,0} = \Delta y \begin{bmatrix} RU - \frac{C^2+RU^2}{2\psi_x} - \frac{\mu}{\Delta x} & 0 & \frac{1}{2} - 1\frac{1}{2}\frac{U}{2\psi_x} & 0 \\ \frac{1}{2}RV - \frac{1}{2}\frac{RUV}{2\psi_x} & -\frac{\mu}{\Delta x} & -\frac{V}{2\psi_x} & 0 \\ \frac{1}{2} - \frac{1}{2}\frac{U}{2\psi_x} & 0 & -\frac{1}{2R\psi_x} & 0 \\ \frac{1}{2}A - \frac{1}{2}\frac{UA}{2\psi_x} & 0 & -\frac{A}{2R\psi_x} & 0 \end{bmatrix}, \tag{2.55b}$$

$$A_{0,-1} = \Delta x \begin{bmatrix} -RV - \frac{\mu}{\Delta y} & -\frac{1}{2}RU - \frac{1}{2}\frac{RUV}{2\psi_y} & -\frac{U}{2\psi_y} & \Delta\rho\left(\frac{1}{2}\frac{RUg\Delta y}{\psi_y} - UV\right) \\ 0 & -RV - \frac{C^2+RU^2}{2\psi_y} - \frac{\mu}{\Delta y} & -\frac{1}{2} - 1\frac{1}{2}\frac{V}{2\psi_y} & \Delta\rho\left(\frac{1}{4}g\Delta y - V^2\right) \\ 0 & -\frac{1}{2} - \frac{1}{2}\frac{V}{2\psi_y} & -\frac{1}{2R\psi_y} & \Delta\rho\frac{1}{2}\frac{g\Delta y}{\psi_y} \\ 0 & -\frac{1}{2}A - \frac{1}{2}\frac{VA}{2\psi_y} & -\frac{A}{2R\psi_y} & \Delta\rho\frac{1}{2}\frac{Ag\Delta y}{\psi_y} - V \end{bmatrix}, \tag{2.55c}$$

$$A_{0,1} = \Delta x \begin{bmatrix} -\frac{\mu}{\Delta y} & \frac{1}{2}RU - \frac{1}{2}\frac{RUV}{2\psi_y} & -\frac{U}{2\psi_y} & -\Delta\rho\frac{1}{2}\frac{RUg\Delta y}{\psi_y} \\ 0 & RV - \frac{C^2+RU^2}{2\psi_y} + \frac{\mu}{\Delta y} & \frac{1}{2} - 1\frac{1}{2}\frac{V}{2\psi_y} & \Delta\rho\frac{1}{4}g\Delta y \\ 0 & \frac{1}{2} - \frac{1}{2}\frac{V}{2\psi_y} & -\frac{1}{2R\psi_y} & -\Delta\rho\frac{1}{2}\frac{g\Delta y}{\psi_y} \\ 0 & \frac{1}{2}A - \frac{1}{2}\frac{VA}{2\psi_y} & -\frac{A}{2R\psi_y} & -\Delta\rho\frac{1}{2}\frac{Ag\Delta y}{\psi_y} \end{bmatrix}, \tag{2.55d}$$

$$A_{0,0} = \Delta y \begin{bmatrix} \frac{C^2+RU^2}{\psi_x} + 2\frac{\mu}{\Delta x} & 0 & 1\frac{1}{2}\frac{U}{\psi_x} & \Delta\rho U^2 \\ \frac{RUV}{2\psi_x} & RU + 2\frac{\mu}{\Delta x} & \frac{V}{\psi_x} & \Delta\rho UV \\ \frac{U}{2\psi_x} & 0 & \frac{1}{R\psi_x} & 0 \\ \frac{UA}{2\psi_x} & 0 & \frac{A}{R\psi_x} & U \end{bmatrix} \tag{2.55e}$$

$$+ \Delta x \begin{bmatrix} RV + 2\frac{\mu}{\Delta y} & \frac{RUV}{2\psi_y} & \frac{U}{\psi_y} & \Delta\rho UV \\ 0 & \frac{C^2+RU^2}{\psi_y} + 2\frac{\mu}{\Delta y} & 1\frac{1}{2}\frac{V}{\psi_y} & \Delta\rho\left(V^2 + \frac{1}{2}g\Delta y\right) \\ 0 & \frac{V}{2\psi_y} & \frac{1}{R\psi_y} & 0 \\ 0 & \frac{VA}{\psi_y} & \frac{A}{R\psi_y} & V \end{bmatrix}. \tag{2.55f}$$

In these, the abbreviations are

$$\psi_x = \sqrt{C^2/R + \left(\tfrac{1}{2}U\right)^2}, \qquad \psi_y = \sqrt{C^2/R + \left(\tfrac{1}{2}V\right)^2}, \qquad \Delta\rho = \rho_w - \rho_a. \tag{2.56}$$

The discretisations of the fluxes on the left and right cell faces are equal and so are the discretisations of the top and bottom fluxes. Therefore, $A_{0,0} = -(A_{-1,0}+A_{1,0}+A_{0,-1}+A_{0,1})$, with just one exception: the α-term in the y-momentum equation in $A_{0,0}$ contains an extra $g\rho\Delta x\Delta y$ source term. The system does not depend on the uniform pressure P. The upwind character of the equations can be clearly seen in a few places, among others in the RU-term in the y-momentum equation in $A_{-1,0}$ and $A_{0,0}$ and in the U- and V-terms in the α-part of the second continuity equation. The effect of gravity is only seen in the α-derivatives.

To study the effect of numerical viscosity, the equivalent differential equation for the system (2.54), (2.55) is constructed. Terms in Δx^2 and higher are neglected. The result is, in matrix form, with ∂_x, ∂_{xx} etc. representing first and second derivatives, and with $\Delta = \partial_{xx} + \partial_{yy}$:

$$
\begin{aligned}
L_k = &\begin{bmatrix} 2RU\partial_x + RV\partial_y - \mu\Delta & RU\partial_y & \partial_x & \Delta\rho(U^2\partial_x + UV\partial_y) \\ RV\partial_x & RU\partial_x + 2RV\partial_y - \mu\Delta & \partial_y & \Delta\rho(UV\partial_x + V^2\partial_y + g) \\ \partial_x & \partial_y & 0 & 0 \\ A\partial_x & A\partial_y & 0 & U\partial_x + V\partial_y \end{bmatrix} \\
&-\Delta x \begin{bmatrix} \frac{C^2+RU^2}{2\psi_x}\partial_{xx} & 0 & 1\frac{1}{2}\frac{U}{2\psi_x}\partial_{xx} & \frac{1}{2}\Delta\rho U^2\partial_{xx} \\ \frac{1}{2}\frac{RUV}{2\psi_x}\partial_{xx} & \frac{1}{2}RU\partial_{xx} & \frac{V}{2\psi_x}\Delta x\partial_{xx} & \frac{1}{2}\Delta\rho UV\partial_{xx} \\ \frac{1}{2}\frac{U}{2\psi_x}\partial_{xx} & 0 & \frac{1}{2R\psi_x}\partial_{xx} & 0 \\ \frac{1}{2}\frac{UA}{2\psi_x}\partial_{xx} & 0 & \frac{A}{2R\psi_x}\partial_{xx} & \frac{1}{2}U\partial_{xx} \end{bmatrix} \\
&-\Delta y \begin{bmatrix} \frac{1}{2}RV\partial_{yy} & \frac{1}{2}\frac{RUV}{2\psi_y}\partial_{yy} & \frac{U}{2\psi_y}\partial_{yy} & \frac{1}{2}\Delta\rho UV\partial_{yy} + \frac{RUg}{\psi_y}\partial_y \\ 0 & \frac{C^2+RV^2}{2\psi_y}\partial_{yy} & 1\frac{1}{2}\frac{V}{2\psi_y}\partial_{yy} & \frac{1}{2}\Delta\rho V^2\partial_{yy} \\ 0 & \frac{1}{2}\frac{V}{2\psi_y}\partial_{yy} & \frac{1}{2R\psi_y}\partial_{yy} & \frac{1}{2}\frac{g}{\psi_y}\partial_y \\ 0 & \frac{1}{2}\frac{VA}{2\psi_y}\partial_{yy} & \frac{A}{2R\psi_y}\partial_{yy} & \frac{1}{2}V\partial_{yy} + \frac{1}{2}\frac{Ag}{\psi_y}\partial_y \end{bmatrix}.
\end{aligned}
\tag{2.57}
$$

The first matrix contains L, the equivalent of the continuous system (2.5). The determinant of this part gives an indication of the type of smoothing that is needed to solve the system:

$$
\det(L) = -\left(U\partial_x + V\partial_y\right)\left(\mu\Delta^2 - \left(U\partial_x + V\partial_y\right)\Delta\right). \tag{2.58}
$$

This determinant has convection and diffusion terms, even in the part with the highest derivatives. Therefore, a smoother is needed that is effective for both diffusion and convection. In the following two sections, we get back to this.

The last two matrices contain the numerical diffusion terms. Almost all variables get numerical diffusion terms; furthermore, all the diffusion terms are positive. The pressure–velocity coupling is indeed very strong.

2.4.2 Point Gauss-Seidel In this section, local Fourier analysis is applied to point Gauss-Seidel smoothing (section 2.3.2). This shows how the smoother is affected by changes in the flow and in the settings of the solver parameters. Especially, it shows whether point smoothing is effective for two-fluid flow. Before that, the Fourier transform of the point GS smoother is derived.

First, we construct the linearisation of point GS, by applying it to the linear system (2.54). For underrelaxation, $\boldsymbol{q}^l_{\mathrm{NR}}$ is taken from equation (2.44) and substituted as the state for the $A_{0,0}$-term. The result is an equation for $(\tilde{\boldsymbol{q}}_k^{n+1})_{i,j}$. For the smoothing direction upwards and to the right (denoted $\nearrow$), this is:

$$
\begin{aligned}
&A_{0,0}\left(\tfrac{1}{\omega}(\tilde{\boldsymbol{q}}_k^{n+1})_{i,j} + \left(1 - \tfrac{1}{\omega}\right)(\tilde{\boldsymbol{q}}_k^{n})_{i,j}\right) = \\
&\quad - A_{-1,0}(\tilde{\boldsymbol{q}}_k^{n+1})_{i-1,j} - A_{1,0}(\tilde{\boldsymbol{q}}_k^{n})_{i+1,j} - A_{0,-1}(\tilde{\boldsymbol{q}}_k^{n+1})_{i,j-1} - A_{0,1}(\tilde{\boldsymbol{q}}_k^{n})_{i,j+1}.
\end{aligned}
\tag{2.59}
$$

This linear GS operator is called M_{pGS}.

Then we search the eigenfunctions of M_{pGS}, i.e. those $\tilde{\boldsymbol{q}}_k$ for which $(M_{\mathrm{pGS}}\tilde{\boldsymbol{q}}_k)_{i,j} = \Lambda(\tilde{\boldsymbol{q}}_k)_{i,j}$. On an infinite grid, the eigenfunctions of a linear operator with constant coefficients are complex exponentials:

$$\boldsymbol{e}_{\omega_x,\omega_y,\boldsymbol{v}}(i,j) = \boldsymbol{v}e^{-I(\omega_x i\Delta x+\omega_y j\Delta y)}, \qquad (\omega_x,\omega_y)\in T, \tag{2.60}$$

where $T = \{\omega_x,\omega_y\,|\omega_x \in (-\pi/\Delta x,\pi/\Delta x],\omega_y \in (-\pi/\Delta y,\pi/\Delta y]\,\}$. To avoid confusion with indices, the imaginary unit is denoted I. The vector $\boldsymbol{v}$ is constant. The ω_x and ω_y are the horizontal and vertical frequency of all the waves that can be represented on the infinite grid; higher frequencies mean shorter wavelengths, which require a finer grid. On the other hand, each finite grid can represent a finite subset of these frequencies, so by studying an infinite grid, we get results for all possible finite grids with cell size Δx, Δy.

The eigenvalue Λ is a 4×4 matrix, that depends on the frequencies ω_x and ω_y. We call Λ the Fourier transform of M_{pGS}:

$$\widehat{M_{\mathrm{pGS}}}(\omega_x,\omega_y) = \Lambda, \tag{2.61}$$

it indicates how a wave with frequency ω_x, ω_y is transformed by point smoothing. M_{pGS} is effective for a frequency ω_x, ω_y if it makes the wave $\boldsymbol{e}_{\omega_x,\omega_y,\boldsymbol{v}}$ smaller. Because $\boldsymbol{e}$ is a vector, this happens when all eigenvalues of the matrix Λ are (in absolute value) smaller than 1.

To find Λ, we substitute $(M_{\mathrm{pGS}}\boldsymbol{e})_{i,j} = \Lambda\boldsymbol{e}_{i,j}$ in equation (2.59). After division by $e^{-I(\omega_x i\Delta x+\omega_y j\Delta y)}$, we get:

$$\begin{aligned}\big(\Lambda\left(\tfrac{1}{\omega}A_{0,0} + A_{-1,0}e^{I\omega_x\Delta x} + A_{0,-1}e^{I\omega_y\Delta y}\right)\\ + \left(1-\tfrac{1}{\omega}\right)A_{0,0} + A_{1,0}e^{-I\omega_x\Delta x} + A_{0,1}e^{-I\omega_y\Delta y}\big)\,\boldsymbol{v} = 0.\end{aligned} \tag{2.62}$$

This equation is solved for Λ, it only holds for all $\boldsymbol{v}$ when:

$$\begin{aligned}\Lambda = \left(\tfrac{1}{\omega}A_{0,0} + A_{-1,0}e^{I\omega_x\Delta x} + A_{0,-1}e^{I\omega_y\Delta y}\right)^{-1}\\ \left(\left(1-\tfrac{1}{\omega}\right)A_{0,0} + A_{1,0}e^{-I\omega_x\Delta x} + A_{0,1}e^{-I\omega_y\Delta y}\right).\end{aligned} \tag{2.63}$$

Equivalent expressions are found for other smoothing directions.

The behaviour of $\Lambda(\omega_x,\omega_y)$ is analysed for different smoother settings and flow conditions. For all ω_x and ω_y, Λ is computed and the absolute values of the four eigenvalues of Λ are determined. The largest of these eigenvalues is of most interest here. Let $T_L = \{\omega_x,\omega_y\,|\omega_x \in (-\pi/(2\Delta x),\pi/(2\Delta x)],\omega_y \in (-\pi/(2\Delta y),\pi/(2\Delta y)]\,\}$ be the set of low-frequency waves that can be resolved on a twice coarser grid. Multigrid removes these waves, the smoother must be effective for all frequencies in $T_H = T\backslash T_L$. Thus, the highest value over T_H of the largest eigenvalue of $\Lambda(\omega_x,\omega_y)$ determines the convergence speed of the multigrid method. Even the values in T_L are interesting: if the largest eigenvalue becomes >1 in T_L, the multigrid solver diverges.

The absolute values of the four eigenvalues of Λ are shown in figure 2.3, sorted from small to large. The inner squares in this figure represent T_L. The flow and

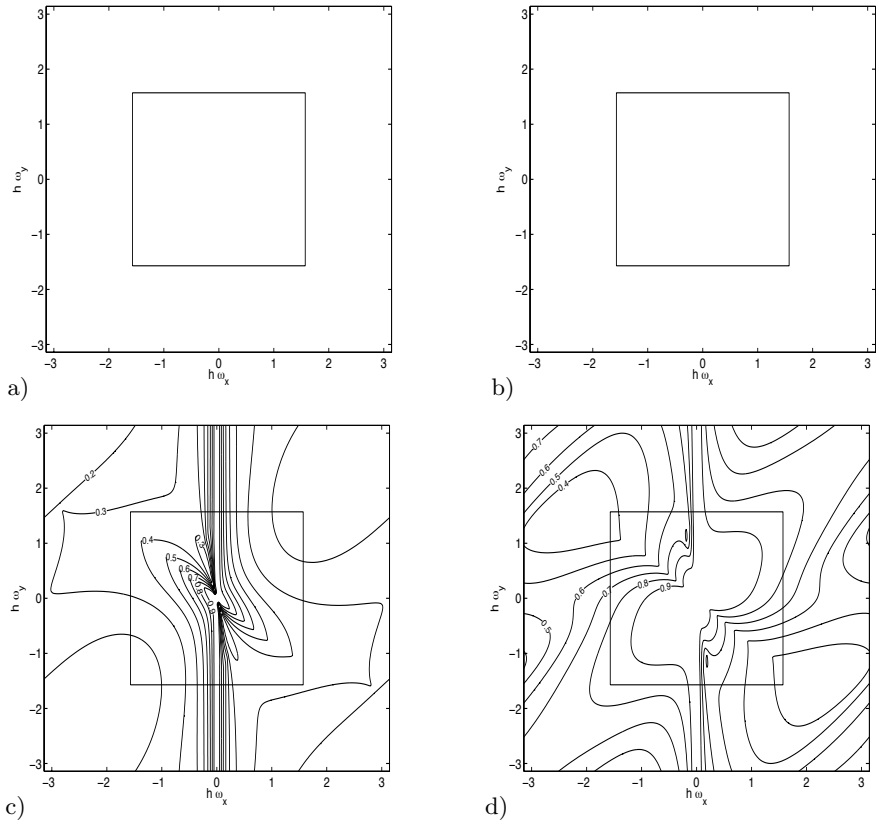


Figure 2.3. Four eigenvalues of Λ. (d) is the largest, (a) and (b) are both zero. $\mu = 0$, $C = 1$. Smoothing direction is $\nearrow$.

solver settings are the 'baseline' settings for this analysis, a horizontal flow with $U = 1$, $A = 1$, and zero viscosity. The artificial compressibility parameter $C = 1$, the smoothing direction is $\nearrow$, no over- or underrelaxation is used, and $\Delta x = \Delta y$.

The first eigenvalue, which is associated with the convection of α, is identically zero: downwind smoothing is an exact solver for pure convection problems. The second eigenvalue is zero here but nonzero in the presence of viscosity; the last two have eigenvectors that contain both pressure and velocity terms. In the following comparisons, we study the largest eigenvalue only.

In two bands of frequencies, the largest eigenvalue is close to 1. The first is the diagonal $\omega_x = \omega_y$, its appearance here is caused by the direction-dependence of the smoother. The other badly damped band is that around $\omega_x = 0$. For convection in a horizontal flow, the frequencies with $\omega_x = 0$ always have damping factor 1. This is because, in the absence of viscosity, any contact discontinuity is a valid solution of the flow equations. So errors in u with $\omega_x = 0$, i.e. 'zebra' errors with horizontal stripes, are not seen and therefore not eliminated by the operator. In practice, the errors with $\omega_x = 0$ are eliminated by the application of proper boundary conditions.

But the badly damped frequencies *around* $\omega_x = 0$ must be removed by smoothing. So in a good smoother, the band must be narrow.

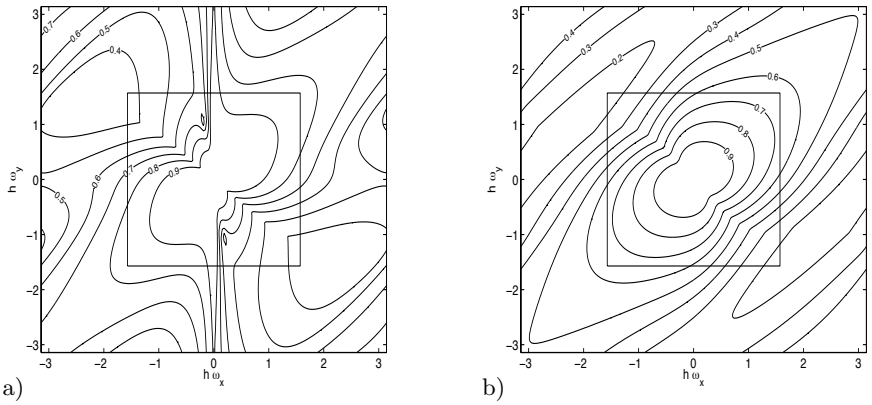


Figure 2.4. Effect of viscosity, $\mu/\Delta x = 10^{-6}$ (a), $\mu/\Delta x = 10^{-4}$ (b). $C = 1$.

Effect of viscosity. Gauss-Seidel smoothing for pure diffusion problems is direction-independent. Therefore, adding physical diffusion to the pure convection baseline problem reduces the direction-dependence of the smoothing (figure 2.4). The badly-damped bands are reduced and overall damping is improved. This effect is small for low viscosity ($\mu/\Delta x = 10^{-6}$, figure 2.4a), but clear for higher viscosity ($\mu/\Delta x = 10^{-4}$, figure 2.4b). So physical viscosity can be used to get good smoothing, but this cannot be depended on for high Reynolds numbers.

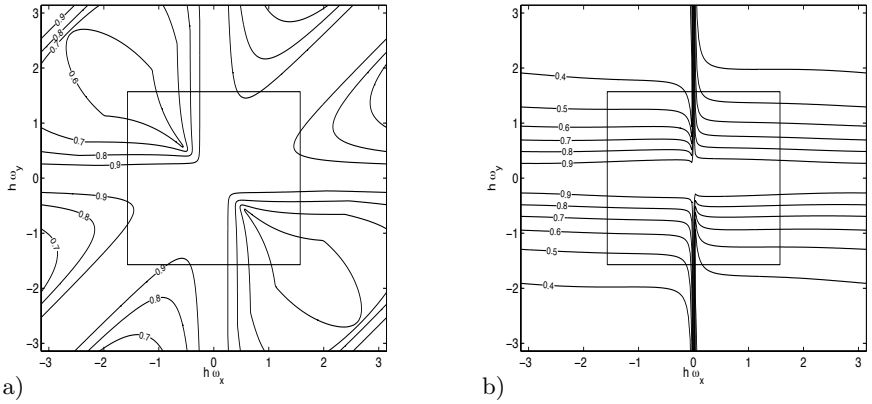


Figure 2.5. Effect of C, $C = 100$ (a), $C = 0.01$ (b). $\mu = 0$.

Effect of C. Changing the artificial compressibility parameter C has the effect of changing the type of the discretisation (see section 2.2.3), from upwind–downwind for small C to central for high C, both with pressure–velocity coupling. The effects

of changing C on the smoothing are shown in figure 2.5. Extreme values for C are chosen to make these effects clear. At the high $C = 100$, we see three broad bands of bad damping (figure 2.5a): around $\omega_x = 0$, $\omega_y = 0$, and $\omega_x = \omega_y$. For the low $C = 0.01$, there is only one broad band, that around $\omega_y = 0$ (figure 2.5b). The case of high C is significant, as it corresponds to the situation with a normal C, but small U, i.e. the situation in stagnation points and boundary layers. The point Gauss-Seidel smoother gives bad smoothing in these locations.

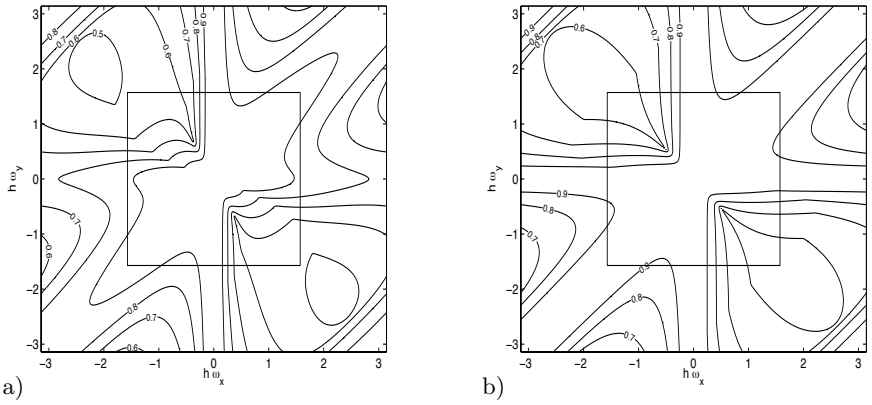


Figure 2.6. Effect of volume fraction, $A = 0.1$ (a), $A = 0.0$ (b). $\mu = 0$, $C = 1$.

Effect of α. When the volume fraction A is changed, the smoothing is mainly influenced by the changes in the density R, which varies from $\rho_w = 1.0$ for $A = 1$ to $\rho_a = 0.001$ for $A = 0$. The density changes the wave speed component $\sqrt{C^2/R + (\frac{1}{2}U)^2}$, reducing R has the same effect as increasing C. This is clearly seen in figure 2.6: the figures resemble figure 2.5a. So in the air-filled parts of the flow domain, the point smoother gives bad smoothing too.

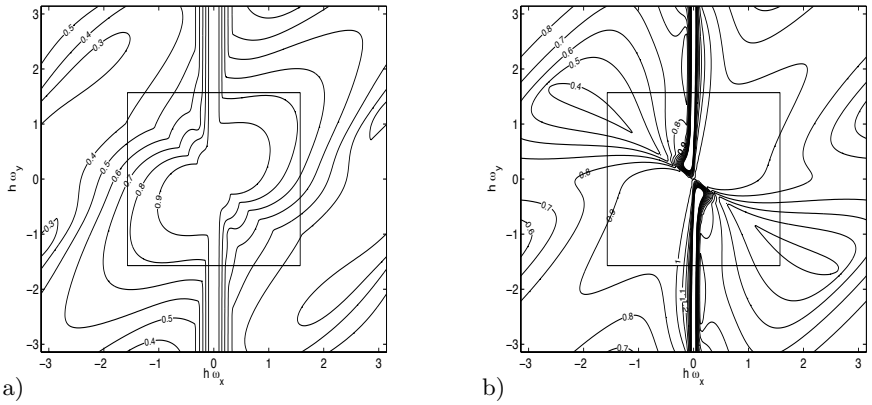


Figure 2.7. Effect of under / overrelaxation, $\omega = 0.9$ (a), $\omega = 1.1$ (b). $\mu = 0$, $C = 1$.

Over- and underrelaxation. Adjusting the overrelaxation parameter ω changes the smoothing (figure 2.7). For underrelaxation, the damping increases in most locations, but the bad band around $\omega_x = 0$ has broadened. Therefore, overall smoothing has not improved. Overrelaxation causes strong instability near $\omega_x = 0$.

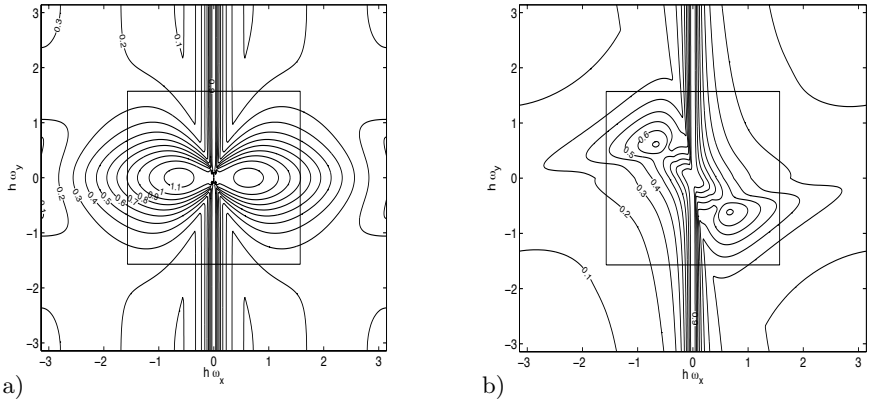


Figure 2.8. Effect of two combined smoothing directions, combined downwind ↗ + ↘ (a) and symmetric smoothing ↗ + ↙ (b). $\mu = 0$, $C = 1$.

Multiple smoothing directions. In the baseline figure 2.3d, bad damping on the one diagonal is combined with good damping on the other diagonal. Therefore, it makes sense to combine successive sweeps of ↗ and ↘ smoothing. The effect of one ↗ sweep plus one ↘ sweep is shown in figure 2.8a. Indeed, the badly damped diagonals have disappeared and the damping in most of the domain is good. Unfortunately, the $\omega_x = 0$ band has not disappeared, so the solver still does not function well.

Another possibility is symmetric smoothing, the combination of ↗ with upwind ↙ smoothing. α is discretised purely upwind, so upwind smoothing on its own has no damping effect on the volume fraction errors. The combination with downwind smoothing is effective for most frequencies (figure 2.8b), but the badly damped band around $\omega_x = 0$ has not disappeared. Thus, even this combination is not useful.

Conclusion. Point Gauss-Seidel is not suitable as a smoother for two-fluid flow. It gives bad damping in regions with low densities or low velocities (stagnation flow etc.), which is not improved by underrelaxation or variation in the smoothing direction. The smoother damps well when the flow has sufficient physical viscosity, but we cannot rely on that for the high-Reynolds water flows considered here.

2.4.3 Line Gauss-Seidel Line Gauss-Seidel is more powerful than point smoothing, it has been used successfully for high-Reynolds flow [15]. This section shows its performance for two-fluid flow. For the line smoother, all the cells in a line (row or column) are updated simultaneously. However, it is still possible to do a Fourier analysis of the smoothing for one cell at the time; the update in cell $\Omega_{i,j}$ depends only on its four neighbours after all. But now, to include overrelaxation, $\boldsymbol{q}^l_{\mathrm{NR}}$ (equa-

tion (2.44)) is used not only in $(\Omega_k)_{i,j}$ but also in its two neighbour cells in the line. So for horizontal lines and smoothing direction $\uparrow$, we solve in each cell:

$$\begin{aligned} A_{0,0}\left(\tfrac{1}{\omega}(\tilde{\boldsymbol{q}}_k^{n+1})_{i,j} + \left(1-\tfrac{1}{\omega}\right)(\tilde{\boldsymbol{q}}_k^{n})_{i,j}\right) + A_{-1,0}\left(\tfrac{1}{\omega}(\tilde{\boldsymbol{q}}_k^{n+1})_{i-1,j} + \left(1-\tfrac{1}{\omega}\right)(\tilde{\boldsymbol{q}}_k^{n})_{i-1,j}\right) \\ + A_{1,0}\left(\tfrac{1}{\omega}(\tilde{\boldsymbol{q}}_k^{n+1})_{i+1,j} + \left(1-\tfrac{1}{\omega}\right)(\tilde{\boldsymbol{q}}_k^{n})_{i+1,j}\right) \\ = -A_{0,-1}(\tilde{\boldsymbol{q}}_k^{n+1})_{i,j-1} - A_{0,1}(\tilde{\boldsymbol{q}}_k^{n})_{i,j+1}. \end{aligned} \quad (2.64)$$

The Fourier transform $\widehat{M_{\mathrm{lGS}}} = \Lambda$ is found again by substituting the eigenfunctions (2.60) in this expression. The resulting eigenvalue problem is:

$$\begin{aligned} &\left(\Lambda\left(A_{0,-1}e^{I\omega_y\Delta y} + \tfrac{1}{\omega}\left(A_{0,0} + A_{-1,0}e^{I\omega_x\Delta x} + A_{1,0}e^{-I\omega_x\Delta x}\right)\right) + \right. \\ &\quad \left.\left(1-\tfrac{1}{\omega}\right)\left(A_{0,0} + A_{-1,0}e^{I\omega_x\Delta x} + A_{1,0}e^{-I\omega_x\Delta x}\right) + A_{0,1}e^{-I\omega_y\Delta y}\right)\boldsymbol{v} = 0, \end{aligned} \quad (2.65)$$

with solution:

$$\begin{aligned} \Lambda = &\left(A_{0,-1}e^{I\omega_y\Delta y} + \tfrac{1}{\omega}\left(A_{0,0} + A_{-1,0}e^{I\omega_x\Delta x} + A_{1,0}e^{-I\omega_x\Delta x}\right)\right)^{-1} \\ &\left(\left(1-\tfrac{1}{\omega}\right)\left(A_{0,0} + A_{-1,0}e^{I\omega_x\Delta x} + A_{1,0}e^{-I\omega_x\Delta x}\right) + A_{0,1}e^{-I\omega_y\Delta y}\right). \end{aligned} \quad (2.66)$$

Equivalent expressions to (2.66) are found for $\downarrow$ horizontal line smoothing and for vertical line smoothing.

Damping factors for horizontal and vertical line smoothing are given in figure 2.9a and b; both figures show badly damped bands. Horizontal smoothing has a wide band around $\omega_x = 0$, vertical smoothing has a comparable wide band around $\omega_y = 0$ and the inevitable, but narrow, band around $\omega_x = 0$. Apparently, the smoother is only effective in the direction of the lines, in the other direction it is less effective than the more diagonally oriented point smoother. So neither horizontal nor vertical line smoothing gives good results.

However, the alternating application of horizontal and vertical line smoothers gives excellent smoothing for all high frequencies (figure 2.9c). Even taking into account that this figure is the result of two smoothing steps, it is still much better than anything offered by point smoothing. Theoretically, the $\omega_x = 0$ frequencies still have damping factor 1, but the band around them is so narrow that it is not even resolved in the figure. A noticeable problem is the two small zones of instability (the maximum amplification factor is more than 2) in the centre of figure 2.9c, which appear in figure 2.8b too. These, however, have no influence on the smoothing of high-frequency errors. As is shown later on, underrelaxation removes these zones.

In the following, the effects of flow and setting changes are tested. All these examples have two smoothing sweeps, one in each direction.

Effect of viscosity. When a little viscosity is added, the smoothing does not change much (figure 2.10a). The only noticeable effect is that the $\omega_x = 0$ band becomes a bit wider! Overall smoothing is not affected. For higher viscosity, the band disappears completely and overall smoothing is improved. The instability near the centre of the figure has disappeared too.

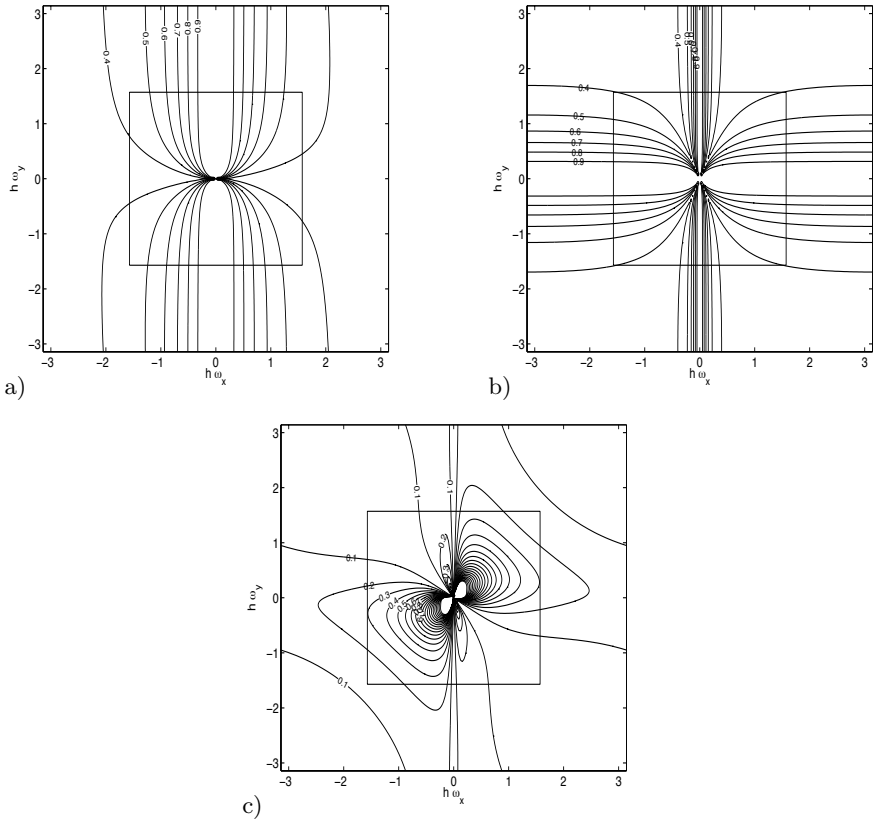


Figure 2.9. Line smoothing, horizontal lines ↑ (a), vertical lines → (b) and alternating directions ↗ (c). $\mu = 0$, $C = 1$.

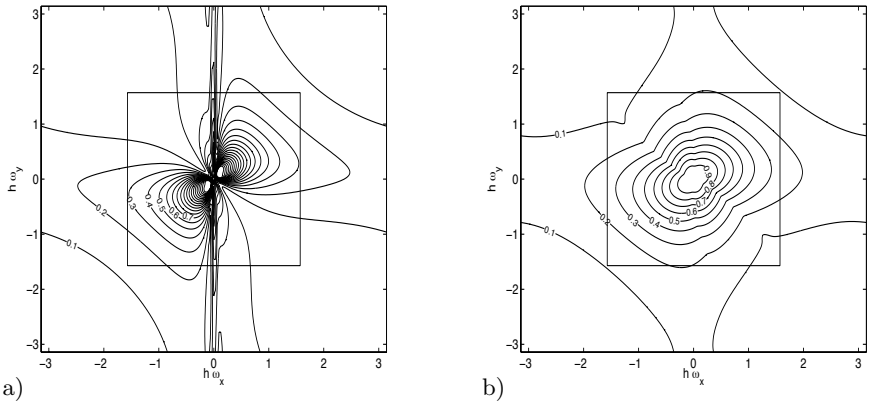


Figure 2.10. Effect of viscosity, $\mu/\Delta x = 10^{-6}$ (a), $\mu/\Delta x = 10^{-4}$ (b). $C = 1$.

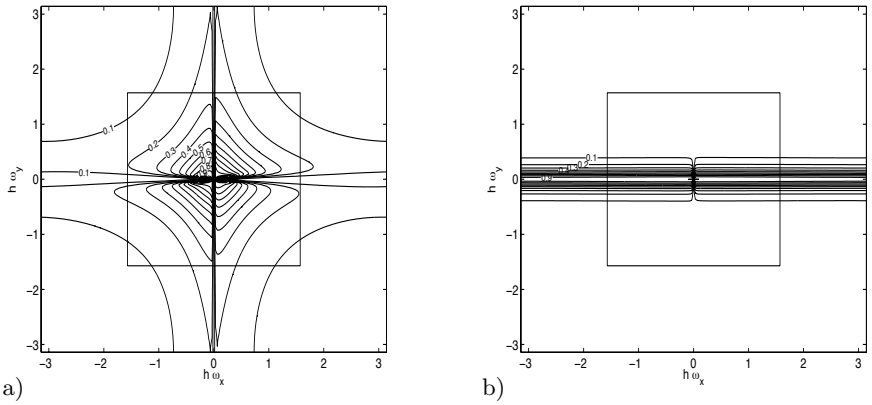


Figure 2.11. Effect of C, $C = 100$ (a), $C = 0.01$ (b). $\mu = 0$.

Effect of C. As pointed out in the previous section, good behaviour of the smoother when $|U| < C/\sqrt{\rho}$ is essential for solving flows with boundary layers or stagnation points. The line smoother works well with high C (figure 2.11). The smoothing of the high frequencies is still good and the instability in the centre has disappeared. The figure for low C shows a narrow band of undamped frequencies around $\omega_y = 0$. This problem can be easily avoided by choosing C large enough, i.e. $C/\sqrt{\rho}$ of the order of the free-stream velocity.

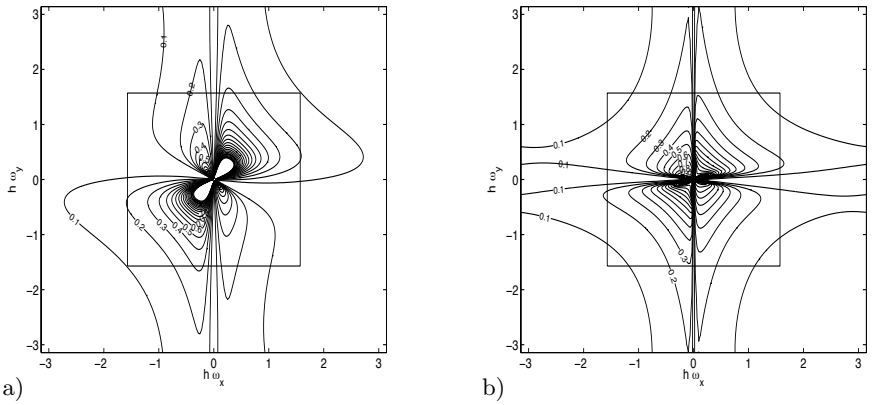


Figure 2.12. Effect of volume fraction, $\alpha = 0.1$ (a), $\alpha = 0.0$ (b). $\mu = 0$, $C = 1$.

Effect of volume fraction. Here too, the effect of reducing α is the same as the effect of increasing C (figure 2.12). Line smoothing works well for low densities.

Under- and overrelaxation. A little underrelaxation ($\omega \approx 0.9$) is essential for line smoothing (figure 2.13). It hardly affects the smoothing of high-frequency errors and it removes the instability in the centre of the figure. This makes the smoother

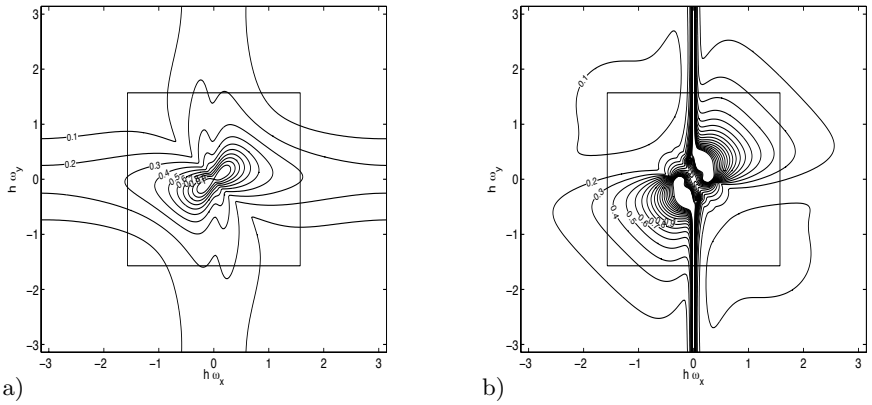


Figure 2.13. Effect of under / overrelaxation, $\omega = 0.9$ (a), $\omega = 1.1$ (b). $\mu = 0$, $C = 1$.

suitable as a stand-alone solver and improves the robustness in a combination with multigrid. So in practice, underrelaxation is used. Overrelaxation causes instability, even in the high-frequency domain.

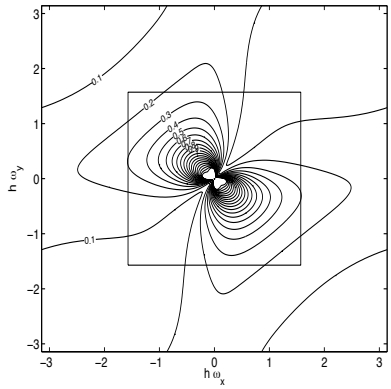


Figure 2.14. Effect of upwind ↖ smoothing. $\mu = 0$, $C = 1$.

Effect of smoothing direction. The original smoothing direction is ↗, i.e. horizontal lines ↑ and vertical lines →. Changing the direction to ↘ has no effect, since the flow is horizontal. Changing to upwind smoothing ↖ has some effect (figure 2.14), but there are no real changes in the damping. This is because the 'upwind' label denotes the sequence in which the vertical lines are updated, while the main smoothing of upwind – downwind errors is performed by the horizontal line sweeps. More generally said, the main smoothing takes place in the updating of each individual line. Therefore, the sequence of updating the lines is not so important.

For the same reason, combinations like fully symmetric smoothing (↗ + ↙) give little difference from the normal alternating smoothing.

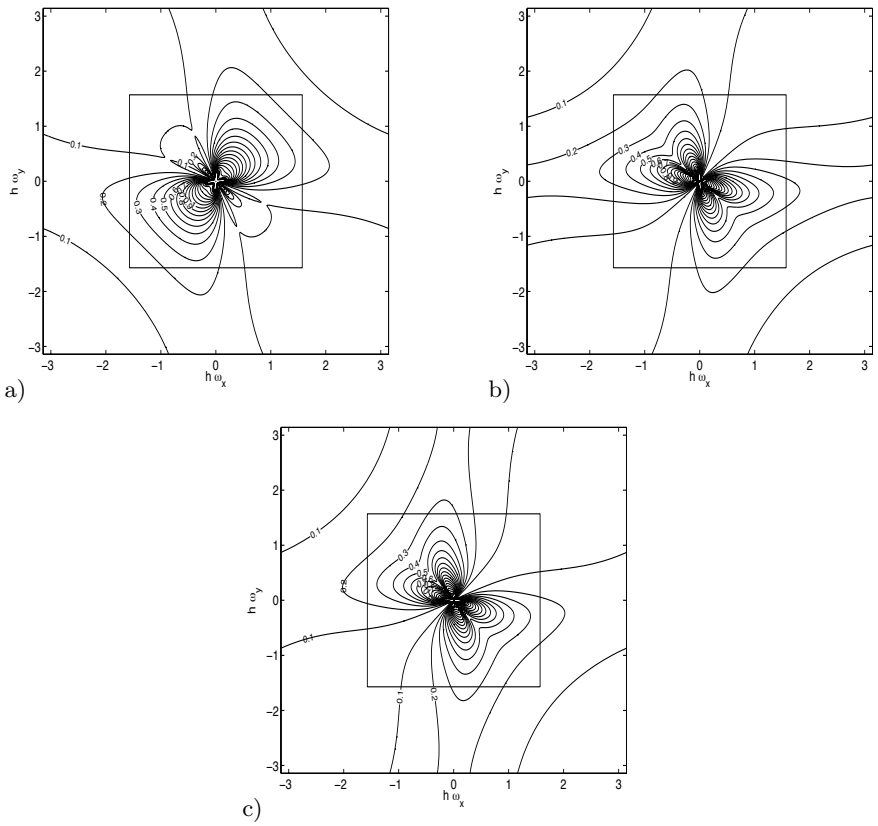


Figure 2.15. Flow at an angle $\theta = \frac{1}{4}\pi$ (direction ↗), effect of smoothing direction. Downwind ↗ (a), crosswind ↘ (b) and upwind ↙ (c). $\mu = 0$,$C = 1$.

Flow direction. The smoothing changes when the underlying uniform flow is put at an angle of $\frac{1}{4}\pi$ (figure 2.15). However, the overall smoothing is still good and the smoothing does not depend much on the smoothing direction, which can now be downwind ↗, crosswind ↘ or upwind ↙.

Conclusion. Alternating line Gauss-Seidel smoothing is a very good smoother for our two-fluid system. It does not need viscosity for good smoothing, it is insensitive to changes in density, in velocity and in velocity direction. Furthermore, the direction of smoothing is not really important. Some underrelaxation is necessary for stability.

2.4.4 Analysis of multigrid Fourier analysis can also be used to study the complete multigrid solver. Here, we show how most of the properties of the smoother are carried over directly to the MG solver. On the other hand, the coarse grid correction can sometimes reduce the MG convergence speed; this is also analysed. The section starts with the derivation of the MG Fourier transform, following e.g. [63].

Linear two-grid algorithm. For Fourier analysis, the multigrid algorithm in section 2.3.1 is linearised. Furthermore, we study a two-grid process, where only one coarser grid is used and where the system is solved exactly on this coarser grid. The coarse grid is denoted with the subscript H, the fine grid with h.

The linearised multigrid method uses the linear operator L_h from equation (2.54) instead of $\mathcal{F}_h$. The coarse grid operator L_H is also linear, so the source term $\boldsymbol{s}_H^n$ (equation (2.41c)) is not computed. Instead, the coarse grid correction is found by directly applying $(L_H)^{-1}$ to the defect $\boldsymbol{d}_H^n$. The prolongation and the restriction are both linear operators, they are not changed. The whole coarse grid correction can be written as a single linear operator:

$$M_{CGC} = \bar{I} - P_H^h L_H^{-1} R_h^H L_h, \tag{2.67}$$

where $\bar{I}$ is the identity matrix. A complete two-grid smoothing step, with pre- and post-smoothing on the fine grid, is:

$$M_{TG} = M_{\text{sm}}^{q_2} \left(\bar{I} - P_H^h L_H^{-1} R_h^H L_h\right) M_{\text{sm}}^{q_1}. \tag{2.68}$$

M_{sm} is either the point or the line smoother. The two-grid algorithm converges when the smoother damps all high-frequency errors and the coarse grid operator (2.67) is sufficiently close to the zero matrix. Also, when the two-grid algorithm converges, the corresponding multigrid algorithm converges too [19]. For multigrid, the exact inverse L_H^{-1} on the coarse grid is replaced by one or more multigrid sweeps on the coarse grid. This inverse is not exact, so the multigrid convergence may be a little slower than the two-grid convergence.

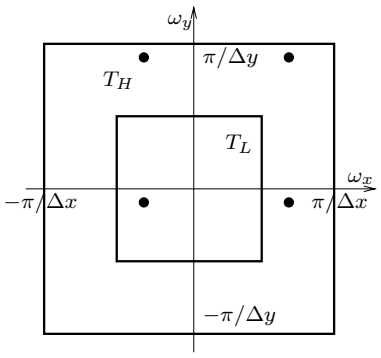


Figure 2.16. Frequency domain, high and low frequencies, four corresponding frequencies.

Fourier transform of two-grid operator. To find the Fourier transform of the two-grid operator (2.68), we compute the Fourier transforms of its individual components. The prolongation and restriction are different from the linear operators encountered before: they can change the frequency of an input grid function. Consider T_L, the grid functions that can be represented on a coarser grid. Each grid

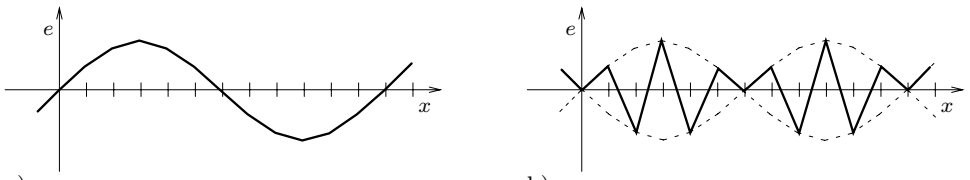


Figure 2.17. One-dimensional example of a grid function in T_L (a) and a corresponding function in T_H (b).

function in T_L has three corresponding functions in T_H that have the same absolute value in each grid point, but change sign between grid points in x-direction, y-direction, or both (figure 2.16, 2.17). The restriction maps all these functions to the one in T_L. On the other hand, the prolongation applied to this single grid function gives contributions in all four frequencies. Therefore, Fourier transforms are computed for these four grid functions together, a 'super' grid function with 16 components:

$$\boldsymbol{e}^*_{\omega_x,\omega_y,\boldsymbol{v}} = \begin{bmatrix} \boldsymbol{e}_{\omega_x,\omega_y,\boldsymbol{v}_1} \\ \boldsymbol{e}_{\omega_x+\frac{\pi}{\Delta x},\omega_y,\boldsymbol{v}_2} \\ \boldsymbol{e}_{\omega_x+\frac{\pi}{\Delta x},\omega_y+\frac{\pi}{\Delta y},\boldsymbol{v}_3} \\ \boldsymbol{e}_{\omega_x,\omega_y+\frac{\pi}{\Delta y},\boldsymbol{v}_4} \end{bmatrix}, \qquad (\omega_x,\omega_y) \in T_L. \tag{2.69}$$

The Fourier transform of the fine grid discretisation L_h, for a single eigenfunction, is found by substituting this function in (2.54):

$$\widehat{L_h}(\omega_x,\omega_y) = A_{0,0} + A_{-1,0}e^{I\omega_x\Delta x} + A_{1,0}e^{-I\omega_x\Delta x} + A_{0,-1}e^{I\omega_y\Delta y} + A_{0,1}e^{-I\omega_y\Delta y}. \tag{2.70}$$

The four grid functions in $\boldsymbol{e}^*$ are transformed independently of each other, so the Fourier transform matrix for $\boldsymbol{e}^*$ is block-diagonal:

$$\widehat{L_h^*}(\omega_x,\omega_y) = \begin{bmatrix} \widehat{L_h}(\omega_x,\omega_y) & & & \emptyset \\ & \widehat{L_h}(\omega_x+\frac{\pi}{\Delta x},\omega_y) & & \\ & & \widehat{L_h}(\omega_x+\frac{\pi}{\Delta x},\omega_y+\frac{\pi}{\Delta y}) & \\ \emptyset & & & \widehat{L_h}(\omega_x,\omega_y+\frac{\pi}{\Delta y}) \end{bmatrix}, \tag{2.71}$$

with $(\omega_x,\omega_y) \in T_L$.

The restriction (2.40) is linear, so its Fourier transform is obtained by substituting an eigenfunction $\boldsymbol{e}$ in this expression. However, the restriction changes both the frequency and the grid on which the function is defined. The grid function on the fine grid can either be the one in T_L or one of its three connected high-frequency grid functions, the grid function after restriction is always the one with $(\omega_x,\omega_y) \in T_L$. Also, the fine grid functions are defined in the small cell centres $(2i(+1)\Delta x, 2j(+1)\Delta y)$, the coarse grid function is defined in the coarse cell centres $(2i+\frac{1}{2})\Delta x, (2j+\frac{1}{2})\Delta y)$, see figure 2.18.

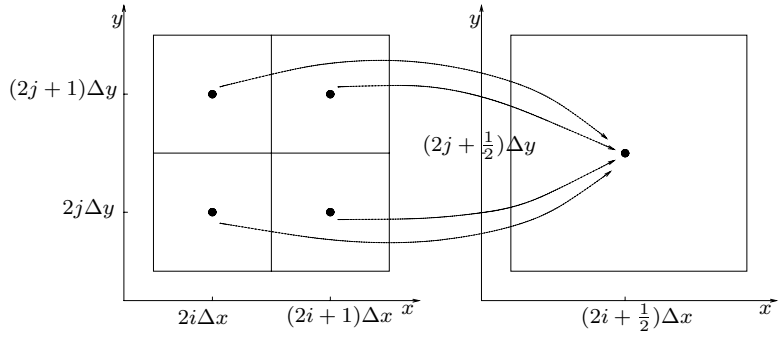


Figure 2.18. Grid points change during restriction.

Upon substitution of these grid functions in (2.40), we get:

$$\Lambda \boldsymbol{v} e^{-I(\omega_x(2i+\frac{1}{2})\Delta x+\omega_y(2j+\frac{1}{2})\Delta y)} = \boldsymbol{v}\left(e^{-I((\omega_x\Delta x+p_x\pi)2i+(\omega_y\Delta y+p_y\pi)2j)}+\right.$$
$$e^{-I((\omega_x\Delta x+p_x\pi)(2i+1)+(\omega_y\Delta y+p_y\pi)2j)}+e^{-I((\omega_x\Delta x+p_x\pi)(2i+1)+(\omega_y\Delta y+p_y\pi)(2j+1))}+$$
$$\left.e^{-I((\omega_x\Delta x+p_x\pi)2i+(\omega_y\Delta y+p_y\pi)(2j+1))}\right), \qquad p_x, p_y = 0, 1.$$

Division by $e^{-I(\omega_x 2i\Delta x+\omega_y 2j\Delta y)}$ and removal of the term $e^{-I(p_x 2i\pi+p_y 2j\pi)}$, which is always 1 (this is where the high-frequency effect disappears), gives:

$$\widehat{R}_{p_x,p_y} = \Lambda = \bar{I}e^{\frac{1}{2}I(\omega_x\Delta x+\omega_y\Delta y)}\left(1+e^{-I(\omega_x\Delta x+p_x\pi)}+\right.$$
$$\left.e^{-I((\omega_x\Delta x+p_x\pi)+(\omega_y\Delta y+p_y\pi))}+e^{-I(\omega_y\Delta y+p_y\pi)}\right), \qquad p_x, p_y = 0, 1. \quad (2.72)$$

The first term is a phase shift, the movement from the fine to the coarse grid. The term between brackets is the Fourier transform of the summation over four cells. As the grid functions for the four related frequencies are mapped onto the same low-frequency grid function, their contributions are summed to get the Fourier transform of the restriction for the super grid function $\boldsymbol{e}^*$:

$$\widehat{R_h^{H^*}}(\omega_x, \omega_y) = \left[\widehat{R}_{0,0} \quad \widehat{R}_{1,0} \quad \widehat{R}_{1,1} \quad \widehat{R}_{0,1}\right]. \quad (2.73)$$

The Fourier transform of the coarse grid operator L_H is almost the same as that of L_h, only the Δx and Δy are changed to $2\Delta x$ and $2\Delta y$:

$$\widehat{L_H}(\omega_x, \omega_y) = A^H_{0,0}+A^H_{-1,0}e^{I\omega_x 2\Delta x}+A^H_{1,0}e^{-I\omega_x 2\Delta x}+A^H_{0,-1}e^{I\omega_y 2\Delta y}+A^H_{0,1}e^{-I\omega_y 2\Delta y},$$
$$(\omega_x, \omega_y) \in T_L. \quad (2.74)$$

This matrix is 4×4, since it is applied to the single low-frequency grid function only. The Fourier transform of L_H^{-1} is the inverse of $\widehat{L_H}$.

The prolongation operator (2.39) maps the single low-frequency grid function on the coarse grid back to the four related grid functions on the fine grid. These

transforms cannot be computed separately for each fine grid function: the four fine grid functions *together* must satisfy (2.39) in the four centres of the fine cells:

$$\Big(\widehat{P}_{0,0}e^{-I(\omega_x\Delta x(2i+q_x)+\omega_y\Delta y(2j+q_y))} + \widehat{P}_{1,0}e^{-I((\omega_x\Delta x+\pi)(2i+q_x)+(\omega_y\Delta y)(2j+q_y))}$$
$$+\widehat{P}_{1,1}e^{-I((\omega_x\Delta x+\pi)(2i+q_x)+(\omega_y\Delta y+\pi)(2j+q_y))} + \widehat{P}_{0,1}e^{-I(\omega_x\Delta x(2i+q_x)+(\omega_y\Delta y\pi)(2j+q_y))}\Big)\,\boldsymbol{v}$$
$$= \bar{I}e^{-I(\omega_x\Delta x(2i+\frac{1}{2})+\omega_y\Delta y(2j+\frac{1}{2}))}\boldsymbol{v}, \qquad q_x, q_y = 0, 1.$$

This is a system of four equations for the matrices $\widehat{P}$, the transform matrices for the individual fine grid functions. Solving them gives:

$$\widehat{P}_{p_x,p_y} = \bar{I}e^{-\frac{1}{2}I(\omega_x\Delta x+\omega_y\Delta y)}\Big(1 + e^{I(\omega_x\Delta x+p_x\pi)} + e^{I((\omega_x\Delta x+p_x\pi)+(\omega_y\Delta y+p_y\pi))}$$
$$+e^{I(\omega_y\Delta y+p_y\pi)}\Big), \qquad p_x, p_y = 0, 1. \quad (2.75)$$

Taken together, the Fourier transform of the prolongation is:

$$\widehat{P_H^{h}{}^*}(\omega_x, \omega_y) = \begin{bmatrix} \widehat{P}_{0,0} \\ \widehat{P}_{1,0} \\ \widehat{P}_{1,1} \\ \widehat{P}_{0,1} \end{bmatrix}. \quad (2.76)$$

The Fourier transform of the full two-grid operator (2.68) is found by multiplying the Fourier transforms of the individual operators:

$$\widehat{M_{TG}} = \left(\widehat{M_{sm}}\right)^{q_2}\left(I - \widehat{P_H^{h}{}^*}\,\widehat{L_H}^{-1}\,\widehat{R_h^{H}{}^*}\,\widehat{L_h{}^*}\right)\left(\widehat{M_{sm}}\right)^{q_1}. \quad (2.77)$$

Note that the dimensions of the matrices fit.

Fourier results. Equation (2.68) shows that the effectiveness of the two-grid algorithm is determined both by the smoothing and by the coarse grid correction. While the choice of the right smoother depends very much on the type of problem to be solved, the coarse grid correction is usually a standard procedure that works well for most problems. Thus, good smoothing usually implies good two-grid convergence.

Using two-grid Fourier analysis, we investigate a few cases where the good smoother performance is carried over to the two-grid algorithm. The smoother has already been thoroughly investigated, it is not necessary to repeat that investigation here. Instead, the last part of this section discusses the cases where the coarse grid correction, instead of the smoother, limits the two-grid convergence.

Figure 2.19 gives results for the two-grid algorithm with point Gauss-Seidel smoothing. The plots show the largest of the 16 eigenvalues for the 'super' grid function (2.69), plotted against the lowest of the four frequencies, the one in T_L. As expected, these results are not good. The first plot shows badly damped regions for errors with horizontal or vertical frequency zero and some frequencies that are

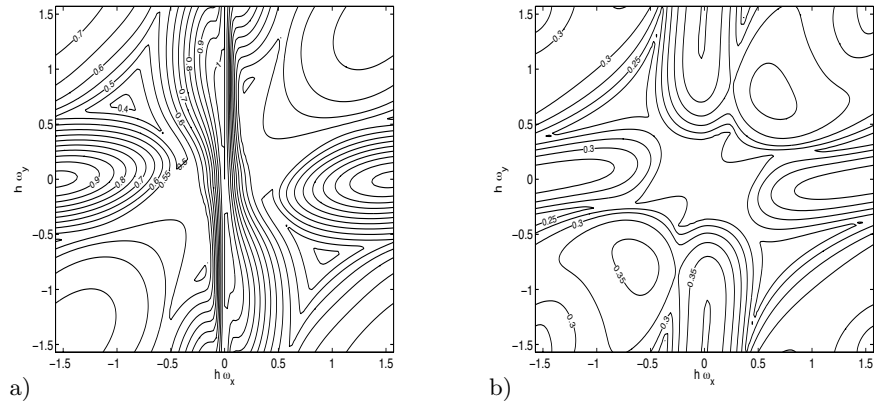


Figure 2.19. Multigrid smoothing with point Gauss-Seidel, without viscosity (a) and with $\mu/\Delta x = 10^{-4}$ (b). Horizontal flow, $\omega = 1$, $C = 1$.

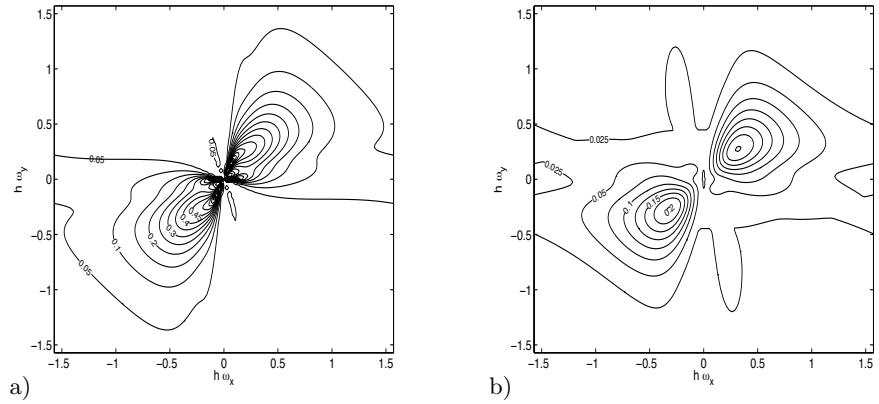


Figure 2.20. Multigrid, line smoothing without underrelaxation (a) and with $\omega = 0.9$ (b). Horizontal flow, $\mu = 0$, $C = 1$.

not damped at all. To achieve damping for all frequencies, point Gauss-Seidel needs viscosity (figure 2.19b).

Line smoothing performs better. In figure 2.20a, most of the domain has a damping factor of 0.05 or less. The only problem is the undamped region in the middle of the figure; underrelaxation (figure 2.20b) eliminates this region. This figure shows an excellent damping factor of 0.20 or less in the whole domain. In fact, the two-grid algorithm with line smoothing performs well in all cases where the line smoother itself works (section 2.4.3). The only exception is discussed below.

Coarse grid correction. When the operator (2.67) is close enough to the zero matrix, the coarse grid correction is effective. Therefore, the approximate inverse $P_H^h \mathcal{F}_H^{-1} R_h^H$ must be close to the real inverse of $\mathcal{F}_h$, which means that the coarse grid operators must resemble the fine grid operator. If that is not the case, then the two-

grid damping suffers. The effect on multigrid convergence is even worse, because MG uses coarse grid corrections to approximate $\mathcal{F}_H^{-1}$. Normally, the convergence rate of MG is independent of the number of grids. But when the coarse grid correction is not good enough, the convergence becomes dependent on the number of grids and decreases with every added grid.

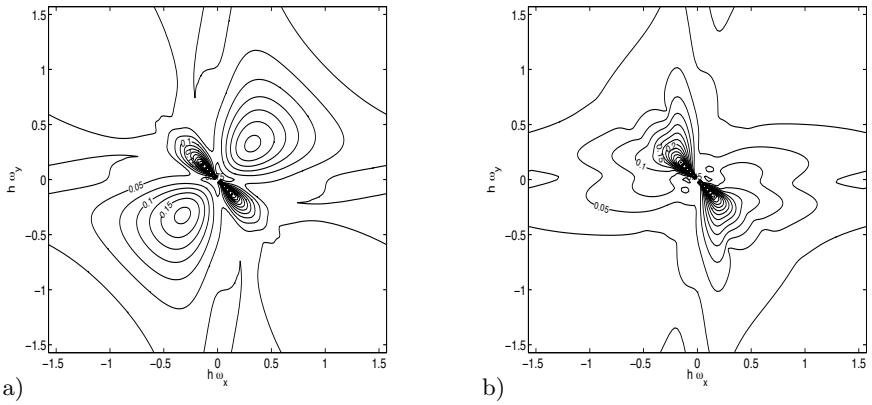


Figure 2.21. Multigrid, flow at an angle $\theta = \frac{1}{4}\pi$, downwind smoothing $\nearrow$ (a) and crossflow smoothing $\searrow$ (b). $\mu = 0$, $C = 1$, $\omega = 0.9$.

For our problem, two factors make the coarse grid operators different from the fine grid operator. The first factor is the numerical diffusion: the numerical viscosity added on the coarse grid is twice as high as the viscosity on the fine grid. This limits the convergence rate of the two-grid algorithm to $\frac{1}{2}$ [15, 63]. Fourier analysis of our two-grid algorithm shows that this 'half' problem does not occur when the flow is horizontal (figure 2.20b). But in figure 2.21, where the flow is at an angle to the grid, there is a band of frequencies normal to the flow direction where the two-grid smoothing factor is about 0.5. So the 'half' problem occurs indeed, but only when the flow is at an angle to the grid. This conclusion agrees with the findings of Frohn-Schauf [15].

Another cause for differences between the coarse- and the fine-grid operator is the difference in the solutions on the fine and coarse grid. In the two-grid analysis, both operators are linearised around the same state U, V, A. But in a real computation, the solutions on the fine and the coarse grid are not the same. This is very important near the water–air interface, which is a smeared discontinuity in α. Different smearing on coarse and fine grids, as well as small differences in the interface location between two grids, may cause large differences in the α-values in some locations.

To study this effect, we perform a Fourier analysis with the fine-grid operator linearised around $A_{\text{fine}} = 1$, but the coarse-grid operator linearised around a different A_{coarse}. The result is given in figure 2.22. For a moderate difference in the operators, $A_{\text{coarse}} = 0.9$, the damping is not affected, although the figure has changed. But for a large difference, i.e. $A_{\text{coarse}} = 0$, the damping has worsened and the damping for the lowest frequencies has become 1. Thus, the highest damping factor increases

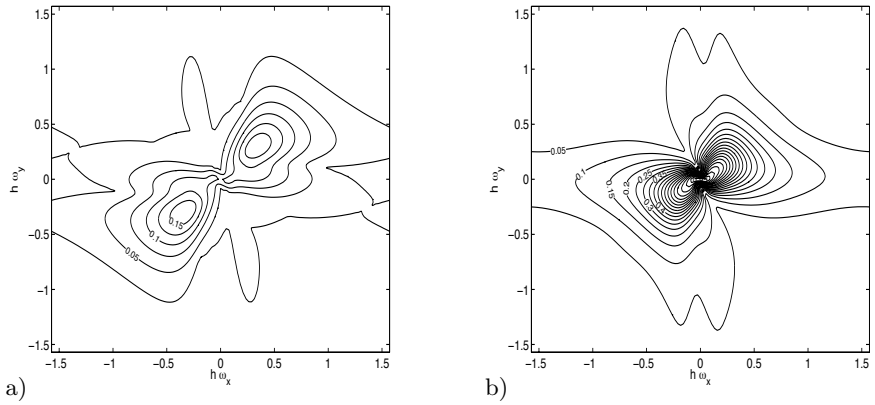


Figure 2.22. Change in the coarse grid operator. $A_{\text{fine}} = 1$, $A_{\text{coarse}} = 0.9$ (a) or $A_{\text{coarse}} = 0.0$ (b). Horizontal flow, $\mu = 0$, $C = 1$, $\omega = 0.9$.

more or less proportional to the difference in the two operators.

Conclusion. In general, the multigrid solver shows the good convergence rates expected from the line smoother. Multigrid convergence on fine grids with many coarser levels may be worse than on coarse grids, but this decrease in convergence should be small when the grid is aligned with the flow and when the interface location is resolved well on the coarser grids.

2.5 Test cases

The two-fluid method from the previous sections is tested on three related test cases: flows in a 2D channel with a bottom bump, as measured by Cahouet [6]. Numerical results were reported among others by Cahouet himself [6] and by Van Brummelen [4, 5]. Our tests show, that efficient solution of a surface capturing discretisation is indeed possible. The tests also show the limitations of the current method, suggesting directions for further development. This section describes the test and our numerical setup (section 2.5.1), the flow results and the properties of the flow discretisation (section 2.5.2), and the performance of the multigrid solver (section 2.5.3).

2.5.1 Cahouet test setup This section describes the experimental configuration and the way in which it is adapted for the numerical tests.

Experiment. Cahouet's experiments were conducted in a water channel with a rectangular cross-section, open at the top (figure 2.23). The flow rate is set with the pump at the inflow side, the water height in the channel is set with the outflow valve. The obstacle on the bottom has a length L, a thickness $E = 0.1L$, and a polynomial shape:

$$y = \tfrac{27}{4} E \frac{x}{L} \left(\frac{x}{L} - 1 \right)^2 . \tag{2.78}$$

This shape gives non-breaking waves of reasonable size without flow separation over the obstacle.

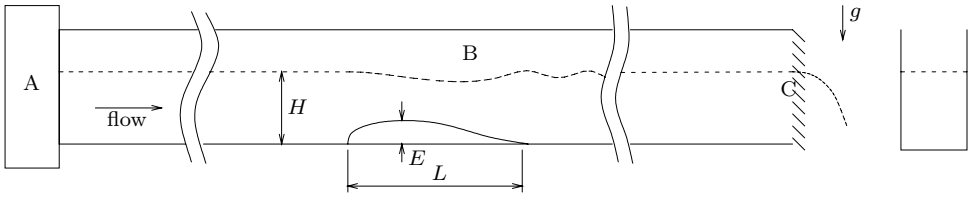


Figure 2.23. Water channel with pump and settling chamber (A), test section with bottom bump (B), and venetian blind type outflow valve (C).

Cahouet performed measurements on flows with different Froude numbers:

$$Fr = \frac{U}{\sqrt{gH}}, \tag{2.79}$$

both with $Fr < 1$ (subcritical flow, écoulement fluvial) and with $Fr > 1$ (supercritical flow, écoulement torrentiel). The Reynolds numbers based on the water height H,

$$Re_H = \frac{\rho U H}{\mu}, \tag{2.80}$$

are of $\mathcal{O}(10^5)$. Three cases are used here, two subcritical cases and one supercritical case. Their parameters are given in table 2.1.

Table 2.1. Cahouet test cases.

nr.	Fr	E/H	Re_H
1	0.43	0.20	$1.3 \cdot 10^5$
2	0.52	0.15	$2.4 \cdot 10^5$
3	2.05	0.44	$1.9 \cdot 10^5$

Turbulent boundary layers appear on the bottom and side walls of the channel. The boundary layer on the bottom of the channel determines the velocity profile of the flow in front of the obstacle, the boundary layers on the side walls introduce 3D effects in the basically 2D flow. For the subcritical cases, the boundary layer thickness is about half the obstacle height; the 3D effects are small. But for the supercritical case, the boundary layer thickness is more than half the water height and the 3D effects are significant: the measured wave was higher in the middle than at the channel walls.

Numerical simulation. The above test cases are simulated on curved quadrilateral grids (figure 2.24), with $n \times 4n$ cells. The cells are basically rectangular, but are deformed to follow the obstacle on the bottom. We model only the test section of the channel and ignore 3D effects.

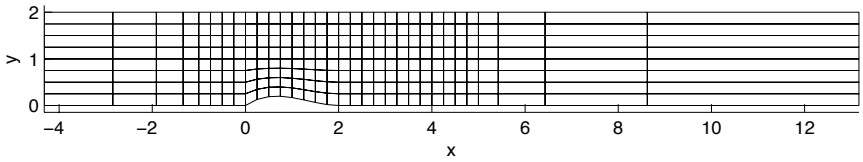


Figure 2.24. Grid for Cahouet test, 8×32 cells, with stretched cells at the inflow and outflow boundaries.

In subcritical flow, gravity waves can travel upstream. Thus, when Dirichlet outflow boundary conditions are imposed close to the obstacle, spurious waves are reflected from this boundary, hampering the convergence of the solver. To find steady solutions, it was found that the waves must be completely damped out before they reach one of the boundaries. Therefore, the grids for the subcritical test cases are stretched exponentially over the first $n/2$ cells in x-direction at the outflow and inflow side. Such a 'numerical beach' is customary in subcritical flow simulation. In the supercritical case, no waves can travel upstream, so stretching is not needed. In that case, the grid runs from $x = -2$ to $x = 6$.

The inflow and outflow boundary conditions must be chosen with care. From a physical point of view, it makes sense to impose the same type of conditions as the 'boundary conditions' in the real channel, i.e. to set the flow rate at the inflow boundary and the water height at the outflow boundary. Also, to get good convergence, it is useful to have Dirichlet boundary conditions. Therefore, we set inflow conditions on u, v and α, thus fixing the flow rate and the inflow water height. On the outflow boundary, p is specified. In subcritical flow, the outflow water height is set by taking this pressure equal to the hydrostatic pressure for the desired water height. The solver then automatically adjusts the water height to fit this pressure profile. For supercritical flows, the outflow height is determined by the flow in front of it; the outflow pressure has only a local influence. We set this pressure, after each iteration, equal to the hydrostatic pressure for the current outflow water height. This adaptation proved unnecessary, so in chapters 3 and 4, the pressure is set like for subcritical flow and the resulting small local jump in water height is ignored.

On the upper side of the domain, a slip wall is specified. This condition is more robust than a constant-pressure boundary with both inflow and outflow. Because of the low air density, the influence of the upper wall on the water flow is small.

Table 2.2. Numerical tests.

nr.	Fr	Re_H	H	g	μ
1	0.43	3333	1.00	5.41	0.0003
2	0.52	4433	1.33	2.78	0.0003
3	2.05	1517	0.46	0.52	0.0003

The numerical tests are computed at lower Reynolds numbers than the experiments. The experimental flows are turbulent, so the laminar NS equations have no

steady solutions for the experimental conditions. Therefore, the Reynolds numbers are lowered until steady laminar flow is obtained. To get realistic boundary layers, the bottom wall boundary condition is adapted too. In the subcritical cases, the leftmost part of the bottom wall is modeled as a slip wall; the start of the no-slip condition is chosen such, that the boundary layer hitting the obstacle is about as thick as the turbulent boundary layer in the experiment. In the supercritical case, the entire bottom wall is a slip wall. To simulate the effect of a boundary layer, the inflow velocity field is given the shape of Cahouet's measured boundary layer.

The settings for the three cases are summarised in tables 2.2 and 2.3. With these, convective and diffusive boundary conditions are set as described in section 2.2.6. In all cases, $\rho_w = 1$ and $\rho_a = 0.001$.

Table 2.3. Numerical tests: boundary conditions.

nr.	Inflow	Outflow	Lower wall
1, 2	$u = 1$, $v = 0$, $\alpha = 0, 1$.	Fixed hydrostatic pressure, water level at H.	No-slip for $x \geq -2$, slip for $x < -2$.
3	Varying u, $v = 0$, $\alpha = 0, 1$.	Adjusted hydrostatic pressure.	Slip wall.

2.5.2 Flow solution This section gives the steady flow solutions obtained with our algorithm. The focus is on the properties of the flow model, the flux function, and the boundary conditions; the performance of the multigrid solver is not yet discussed.

Case 1, $Fr = 0.43$. The experimental solution for this test case has relatively smooth waves. However, the waves are short: fine grids are needed for sufficient resolution. The solution on a 128×512 grid is shown in figures 2.25 and 2.26. These figures give the absolute value of the velocity, the vertical velocity component, and the volume fraction, i.e. the water surface.

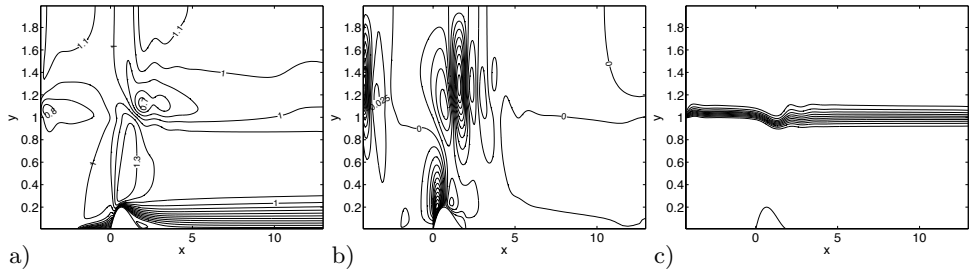


Figure 2.25. $Fr = 0.43$, flow in the whole computational domain. Isoline plots of speed $|\boldsymbol{u}|$ (a), vertical velocity v (b), and volume fraction α (c). 128×512 grid.

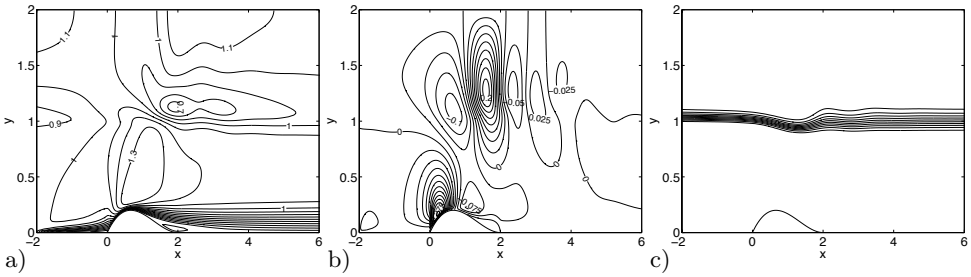


Figure 2.26. $Fr = 0.43$, detail of figure 2.25.

The water surface moves down over the obstacle and then evolves into a wave train (figure 2.26c), which is typical for a flow with $Fr < 1$. Although the grid is rather fine, the waves damp out quickly, even to the left of the stretched cells starting at $x = 5$. This damping is caused by the high numerical viscosity of the first-order accurate method, which also causes the smearing of the water surface over several cells. The highest vertical velocities do not appear in the water, but in the air above the waves.

The flow behind the obstacle is dominated by the boundary layer at the bottom wall. The start of the boundary layer can be seen at $x = -2$ and behind the obstacle, the boundary layer becomes very thick. In figure 2.25c, we see that the desired outflow water level $y = 1$ is indeed obtained. Right behind the inflow boundary, the water level jumps to adapt itself to the main flow. This shows that the water height in the entire flow domain is determined by the outflow height.

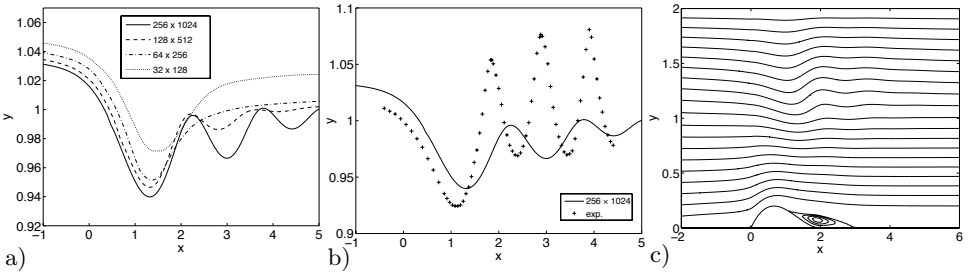


Figure 2.27. $Fr = 0.43$, water surface and streamlines. Shown is a grid convergence study of the $\alpha = 0.5$ isoline (a), a comparison of this isoline with the water height in Cahouet's experiment (b), and a streamline plot (c). The streamlines have a difference of 0.1, or 0.002 in the recirculation zone.

The accuracy of the solution is studied with a grid convergence test (figure 2.27a). The finest grid is twice finer than the 128×512 grid shown before: it has a well-developed wave train. Two coarser grids have no waves at all. In front of the wave train, the solution shows proper first-order convergence, but the flow in the wave train itself is less converged.

The real drama is shown in figure 2.27b. A comparison with Cahouet's experiment shows that the waves in our computation are too long and much too weak. The

cause of the low amplitude is seen in figure 2.27c: a streamline plot reveals a laminar separation bubble behind the obstacle, which lengthens the apparent obstacle for the main flow. This longer, smoother obstacle causes waves of a smaller amplitude than those in the experiment, where the (turbulent) boundary layer remains attached. The size of the laminar separation bubble depends on the numerical viscosity; on the coarsest grids, it is not even there. This, together with the wave damping by numerical viscosity, causes the bad grid convergence in figure 2.27a.

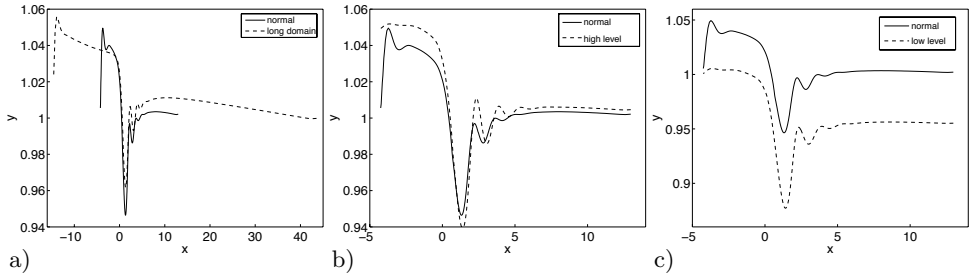


Figure 2.28. $Fr = 0.43$, effect of boundary conditions. $\alpha = 0.5$ isolines for a longer domain, with more cells stretched (a), for a higher inflow level of 1.05 (b), and for a lower outflow level of 0.95 (c). All grids have 128×512 cells.

In figure 2.28, the effect of the inflow and outflow boundary conditions is studied. In the first picture, more cells are stretched (all cells with $x < 0$ or with $x > 4$) and the flow domain is much larger. Thus, the effect of the boundary layers becomes stronger: the water level still ends at 1, but it is higher at the end of the wave train and it rises higher at the inflow. In the second picture, the inflow water level is increased to 1.05. This has almost no effect on the average water height, although the wave amplitude is increased because the water flow has become faster (same water level, larger flux). The inflow adjustment disappears. Finally, lowering the outflow level to 0.95 lowers the water level almost uniformly in the whole flow (figure 2.28c). This supports the conclusion that the outflow boundary condition determines the water height.

Summarising: for the $Fr = 0.43$ testcase a solution with a wave train is found, so our two-fluid model can simulate the physics of water waves. The high physical and numerical viscosity in the test cause strong damping of the waves. Together with laminar separation over the obstacle, these reduce the wave amplitude. The water height in the whole flow is determined by the outflow boundary.

Case 2, $Fr = 0.52$. The second test case has a greater water depth and stronger waves than the first case. Its first wave is close to breaking, which makes it difficult to compute numerically. Our method computes this case without problems, albeit partly because the wave amplitudes are reduced by laminar separation (figures 2.29 and 2.30).

The solution resembles the previous case: it has a similar boundary layer (figure 2.29a) and a separation bubble, although this bubble is a bit smaller and shorter than

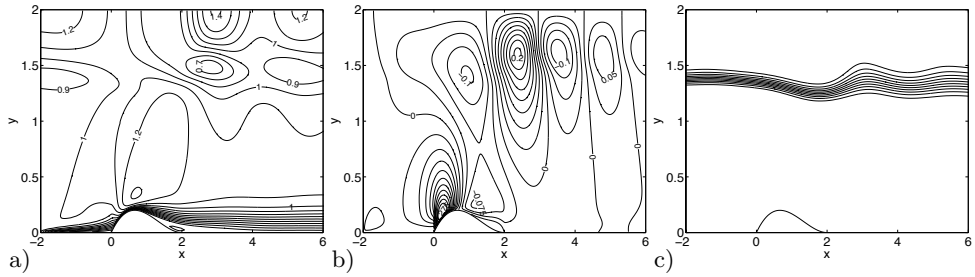


Figure 2.29. $Fr = 0.52$, flow in a part of the computational domain. Isoline plots of speed $|\boldsymbol{u}|$ (a), vertical velocity v (b), and volume fraction α (c). 128×512 grid.

the previous one (figure 2.30c). The wave resolution is better, because the longer waves are resolved on more cells. Therefore, we see less damping in figure 2.29c and better grid convergence in figure 2.30a. Indeed, the wave train is resolved even on the coarsest grid. Also, the correspondence with the experiment is better: the wave amplitude is too low, but the wavelengths are reasonable. The waves clearly show nonlinear behaviour: the wave crests are steeper than the troughs.

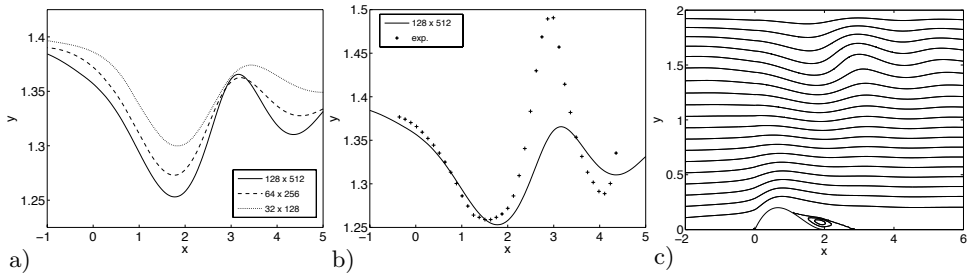


Figure 2.30. $Fr = 0.52$, water surface and streamlines. Grid convergence study of the $\alpha = 0.5$ isoline (a), comparison with Cahouet's experiment (b), and streamline plot (c). Streamlines as in figure 2.27.

Case 3, $Fr = 2.05$. The last test case is a supercritical flow. The water surface follows the obstacle and no wave train appears; figure 2.31 gives the solution. This figure shows the whole domain, since the grids are not stretched at the boundaries. We see upflow in front of the obstacle and downflow behind, both in the water and in the air (figure 2.31b). The air gets a high velocity above the obstacle, because there is a top wall (figure 2.31a). As the case was run with a slip lower wall (this proved necessary to get a steady solution), we see no boundary layer, except the velocity profile specified at the inflow boundary.

A grid convergence study in figure 2.32a shows that the wave is resolved well on all grids and that the solution on the finest grid is almost converged. Indeed, correspondence with the experiment is good (figure 2.32b)[1]. The slightly flatter

[1]Since the experimentally measured wave was three-dimensional, the experimental solution given

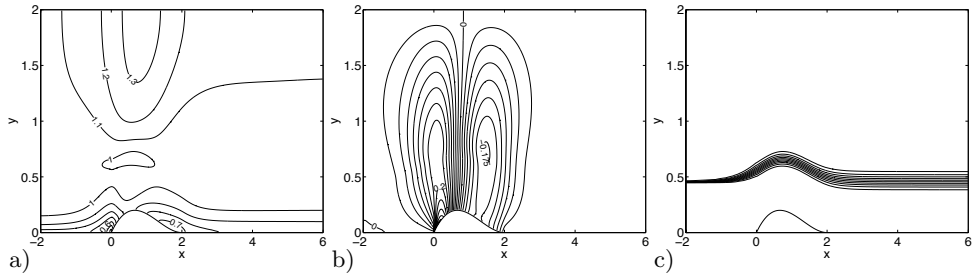


Figure 2.31. $Fr = 2.05$, flow in the whole computational domain. Isoline plots of speed $|\boldsymbol{u}|$ (a), vertical velocity v (b), and volume fraction α (c). 128×512 grid.

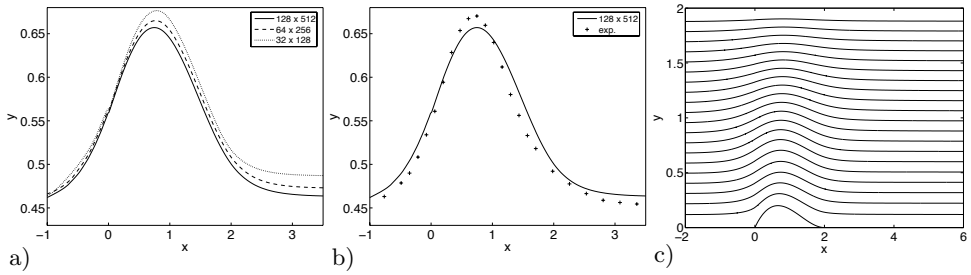


Figure 2.32. $Fr = 2.05$, water surface and streamlines. Grid convergence study of the $\alpha = 0.5$ isoline (a), a comparison of this isoline with the (average) water height in Cahouet's experiment (b) and a streamline plot (c). All streamlines have a difference of 0.1.

wave may be caused by the absence of boundary layer effects or by the top wall. The streamline plot 2.32c shows a smooth flow, without separation.

2.5.3 Multigrid performance The previous section discussed the quality of the solutions obtained with our method. Now, we consider the efficiency of the computation. The effect of the flow type on the multigrid convergence is studied, as well as the best multigrid strategy (V-cycle or W-cycle?).

Problems on the 128×512 grid are solved with multigrid on 7 grids: the coarsest of these has 2×8 cells. Initially, a W-cycle is used with one pre- and one post-smoothing step. The underrelaxation parameter for the smoother is $\omega = 0.9$. The coarsest grid gets four relaxations per iteration, instead of two, to bring a smoothing step on the coarsest grid as close as possible to exact inversion. This is necessary because the coarsest grid has no coarse-grid correction, so even the low-frequency errors on that grid must be removed by the smoothing.

Convergence rate. A plot of the residual during computation of the $Fr = 0.43$ testcase is given in figure 2.33a. The 'sawtooth' character of the plot is caused by the full multigrid algorithm: it shows the computations on the coarser grids first. Only the last 26 iterations are the expensive ones on the finest grid. The plot is compared

here is the average of the measured wave height at the centre and the walls of the canal.

with the computation of the same problem, with line Gauss-Seidel smoothing on a single grid (figure 2.33b). Even though line Gauss-Seidel on its own is already a powerful solver, this computation takes much more time than with multigrid. We get back to that comparison later and focus on the analysis of the multigrid convergence first.

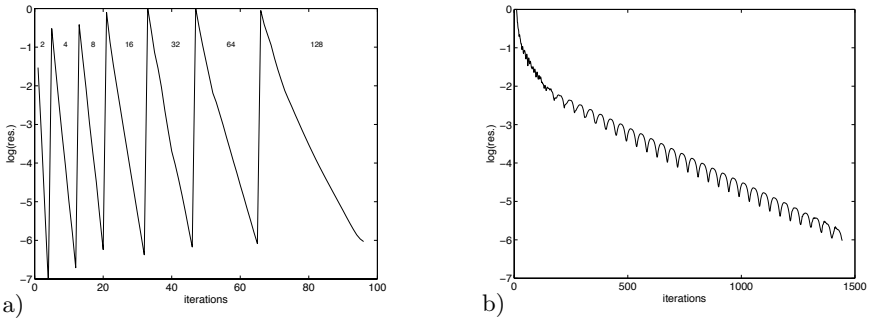


Figure 2.33. Total residual during the iterative solution for the $Fr = 0.43$ testcase, with multigrid (a) and single grid line smoothing (b).

Three factors determine the convergence speed of multigrid. The first is the smoother performance, which should be good for the powerful line smoother (section 2.4.3). The second factor is the difference between the fine and coarse grid operators (section 2.4.4): the 'half' problem (figure 2.21) and the difference between the states on the coarse and fine grids (figure 2.22). We use converged solutions on the coarse grids for the coarse grid operators (section 2.3.3), while the coarse and fine solutions in the test differ significantly; in figure 2.33a, the finest grids have a wave train and laminar separation, which are absent on the coarse grids. Therefore, the convergence on the fine grids is decreased. The last factor is the flow problem itself: when a flow is nearly unsteady, the MG convergence becomes slower. Laminar separation is an indication of approaching unsteadiness, so this may be one of the causes of the slow convergence on the finest grid in figure 2.33a.

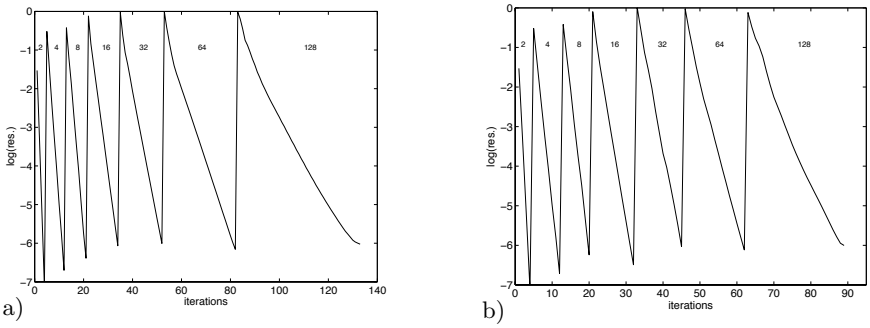


Figure 2.34. Residual during the solution, V-cycles (a) and VW-cycles (b).

Table 2.4. Convergence rates for V-cycles, W-cycles (standard), and VW-cycles, on 6 grids, compared with the theoretical two-grid convergence rates σ from Fourier analysis.

case	σ $(A=1)$	σ $(A=0)$	4	8	16	32	64	128
V	0.171	0.182	0.141	0.182	0.331	0.466	0.611	0.763
W	”	”	0.132	0.147	0.274	0.401	0.508	0.677
VW	”	”	0.132	0.148	0.267	0.323	0.438	0.636

If the convergence rate is determined by problems in the coarse grid correction, then the convergence rates change when the multigrid strategy is changed: more multigrid cycles on each grid improve the coarse grid correction. To test this, the $Fr = 0.43$ case is computed with V-cycles and VW-cycles (three coarse grid multigrid-cycles per grid), see figure 2.34 and table 2.4. The V-cycle performs worse than the others, so the coarse grid corrections are indeed a limiting factor for the convergence. The VW-cycle gives no improvement on the coarsest grids, so convergence on these grids is limited by the smoother if a W-cycle or more is used. On two finer grids, it gives a better coarse grid correction. But on the finest grid, there is little improvement. This means that approaching unsteadiness limits the convergence here, rather than the coarse grid corrections.

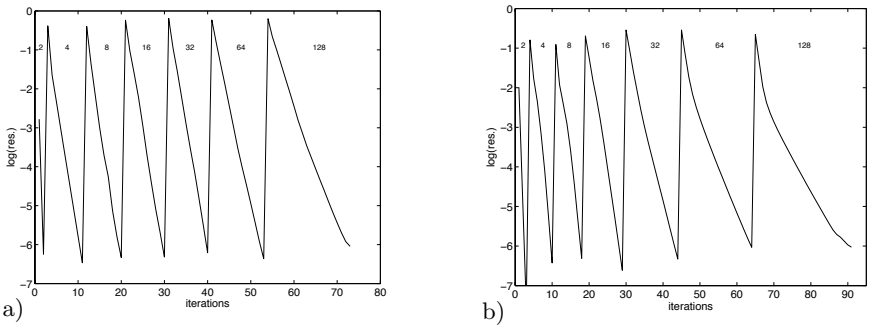


Figure 2.35. Residual for the $Fr = 0.52$ test (a) and for the $Fr = 2.05$ test (b), both with W-cycles.

Table 2.5. Convergence rates for the $Fr = 0.52$ and $Fr = 2.05$ problem.

case	σ $(A=1)$	σ $(A=0)$	4	8	16	32	64	128
$Fr = 0.43$	0.171	0.182	0.132	0.147	0.274	0.401	0.508	0.677
$Fr = 0.52$	”	”	0.203	0.200	0.214	0.219	0.310	0.523
$Fr = 2.05$	”	”	0.077	0.119	0.230	0.430	0.590	0.675

Figure 2.35 and table 2.5 give convergence results for the $Fr = 0.52$ and $Fr = 2.05$ test cases, with W-cycles. Comparing these with figure 2.33a shows that the

Table 2.6. $Fr = 0.43$ CPU times.

grid	levels	time (s)
16×64	4	5
32×128	5	21
64×256	6	103
128×512	7	572
128×512 (V)	7	634
128×512 (VW)	7	758
128×512	1	18690

$Fr = 0.52$ case has better convergence on fine grids. This case has larger waves, that are resolved on the coarser grids; it is also more stable, because the separation bubble is smaller. This suggests that coarse–fine solution differences and approaching unsteadiness, rather than the 'half' problem, limit the convergence of the $Fr = 0.43$ case. On the other hand, the $Fr = 2.05$ case shows slow convergence on the fine grids. A possible reason is the slow adaptation of the outflow boundary condition to the water height.

Computation times. A good measure of computational efficiency is the required CPU time for a computation. These are measured here for the $Fr = 0.43$ case. All computations are performed on a 2.2 GHz PC. Table 2.6 gives CPU times for the total computation, for different sizes of the finest grid; the coarsest grid is always 2×8, so the total number of grids increases.

When the number of iterations per grid is the same for all grids, then the CPU time for the computation on a grid is between four and five times higher than the CPU time on a twice coarser grid. This estimate seems accurate for the coarsest grids, but underestimates the CPU times for the finer grids. This is not surprising, we have already seen that the number of iterations per grid is not constant on the fine grids.

Table 2.6 also gives the CPU times for the V-cycle and VW-cycle computations. The original W-cycle is the best choice: both the others take more time. The V-cycle is fast per iteration, but requires too many iterations. The VW-cycle on the other hand is very expensive per iteration, but converges only a little faster than the W-cycle.

Finally, the MG solution of the $Fr = 0.43$ test case is compared with line Gauss-Seidel smoothing on a single grid (table 2.6, figure 2.33). This solution process takes about 33 times longer than the multigrid process. No comparison with time stepper solvers is made, but such a solver is not expected to perform better than the single-grid line relaxation. Therefore, the test case shows that for our two-fluid flow model, multigrid solution is far superior to classical relaxation techniques.

2.6 Conclusion

Simulation of steady water flow with surface capturing and multigrid is possible, and efficient.

We derived flow equations that are based on a volume-of-fluid model for the water surface, without interface reconstruction. The resulting system has no explicit model for the water surface. It consists of conservation laws only, so it can be solved efficiently with multigrid.

For the discretisation of the flow equations, the convective and diffusive fluxes are split. Convective fluxes are based on artificial compressibility, gravity is added as a source term. The discretisation allows low viscosity, is robust in combination with a line Gauss-Seidel smoother and can handle the nonlinearity and sensitivity caused by the large density gradients in water–air flow. The splitting of convective and diffusive fluxes allows a straightforward and natural treatment of boundary conditions.

Multigrid requires a smoother that is suitable for the flow equations. Two smoothers have been studied with local Fourier analysis. With the high Reynolds numbers in water–air flow, point smoothers do not give sufficient performance. An alternating line Gauss-Seidel smoother, which is not much more expensive than a point smoother, gives good smoothing under all flow conditions. For stability, a slight underrelaxation is needed.

Because of the nonlinear fluxes, the coarse grid operators for multigrid cannot handle large source terms. This problem is solved by scaling the defects passed to the coarse grids and unscaling the corrections, and by using converged solutions on the coarse grids as starting solutions for the multigrid iterations. The result is that the solutions on the coarse and fine grids differ. This effect is reinforced by the water surface, where the gradient of the density is large. A W-cycle, that compensates for reduced convergence on the coarser grids, is found to be the most efficient multigrid strategy.

The method is applied to three channel flow test cases. A supercritical flow test case is simulated accurately. For two subcritical testcases, the model produces wave trains. However, the laminar model is not able to accurately compute the boundary layer behaviour, it produces laminar separation. Therefore the amplitude of the waves is underestimated.

The performance of the multigrid solver is compared with single-grid techniques. This comparison shows that multigrid is a very efficient solver for the water–air flow equations: it gives a great reduction in computation time.

Outlook. The challenge for the further development of the current method is to further improve the accuracy of the solutions, while the efficiency of the multigrid solver is kept. In the following chapters, two measures are taken to get better accuracy.

First, the method will be adapted for turbulent flow. The water flows for which our model is meant, are all turbulent. The test cases show that it is not possible to simulate these flows correctly with a laminar flow model. Therefore, a turbulence model must be included with the flow equations. The major challenge is the

combination of turbulence modelling with the multigrid solver.

Furthermore, the first-order accurate model is not accurate enough to resolve the short waves that appear in subcritical problems, unless very fine grids are used. Therefore, it is necessary to change to second-order accurate discretisations. Also, compressive schemes for the discretisation of the volume fraction will appear to be useful, to keep the interface sharp.

Solving turbulent flow equations

This chapter describes a multigrid solution technique for turbulent flow equations. For the simulation of water waves, turbulence modelling is a necessity; the previous chapter showed that turbulence is an essential part of the physics of water flow. But the method presented here has a wider application area than water flow alone. Most real-life flows are turbulent, be they flows around vehicles or flow in pipes and channels. So in design processes for these devices, a fast turbulent flow solver is a very useful tool.

Quasi-steady turbulent flows are usually modelled with the steady Reynolds-Averaged Navier-Stokes (RANS) equations. To obtain the RANS equations, the unsteady turbulent flow field is averaged; the result is a set of flow equations that allow steady solutions. These equations contain unknown closure terms for the effect of turbulence, which are modelled approximately with a turbulence model: one or more differential equations for the turbulent stress. As opposed to the laminar Navier-Stokes equations, these turbulence models contain source terms, that represent the production and dissipation of turbulence.

Multigrid solution of the RANS equations is not straightforward. For the single-fluid laminar Navier-Stokes equations, the steady flow equations can be solved directly with a combination of nonlinear multigrid and Gauss-Seidel smoothing, i.e. our solver from chapter 2; examples are found in the work of Hemker et al. [22, 31], of Dick et al. [12, 59], and of Trottenberg et al. [15, 63]. But due to the source terms in the turbulence model, the RANS equations cannot be solved with these techniques. Instead, multigrid is usually combined with a time stepping approach. Either time integration is used as a smoother in the multigrid algorithm or multigrid is used for the individual time steps in an implicit time integrator, that time-marches the unsteady RANS equations to convergence. In the first category, Mavriplis [43, 44] uses Jameson's multigrid method with a Runge-Kutta time integration smoother, on unstructured grids. Liu and Zheng [40] present a finite-volume method on structured meshes. Carré [7] uses linear multigrid to solve implicit time steps and increases these time steps as the solution converges. Steelant et al. [59] form an exception: they use damped multigrid with a line smoother.

In the solution process, the link between the Navier-Stokes equations and the turbulence model is important. In many flow solvers, the turbulence model is considered loosely coupled to the other flow equations, therefore it is solved separately: alternately, the flow field is updated with the turbulence fixed and the turbulence is updated with the flow fixed. For the flow field step, an existing laminar flow solver can be used. However, Liu and Zheng [40] claim that this technique is inefficient; they report improved convergence when all equations are solved together. On the other hand, Steelant et al. [59] get the best results with a loosely coupled approach.

We show that the steady Reynolds-Averaged Navier-Stokes equations with Menter's turbulence model [46] can be solved with multigrid and Gauss-Seidel smoothing,

without the need for time stepping, and that convergence rates can be obtained that are similar to the most efficient multigrid solvers for laminar flow. We also explain why a fully coupled solution of the flow field and the turbulence is necessary to obtain this convergence. Our novel multigrid technique is a combination of nonlinear line Gauss-Seidel smoothing on the finest grid and linearised coarse grid corrections. Local damping is applied in the initial part of the solution process; this does not reduce the convergence rate because it is not needed in most of the domain. The nonlinear smoothing is used to estimate the need for damping.

The turbulent flow solver is developed first for single-fluid flow, for simplicity and because its use is not limited to water–air problems. In the introductory section 3.1, a brief overview is given of the RANS equations and of Menter's one-equation turbulence model: a simple model that still contains all the relevant features of more complicated turbulence models. The sections 3.2 and 3.3 form the heart of the chapter: section 3.2 shows why the RANS equations cannot be solved with classical nonlinear multigrid and presents a suitable Gauss-Seidel smoother. Furthermore, it explains why the coupled solution is essential for fast convergence. Section 3.3 introduces the linear multigrid algorithm as an adaptation of the algorithm from section 2.3. This algorithm is valid for general spatial discretisations; section 3.4 shows how turbulence is added to the finite-volume discretisation from section 2.2.

The chapter then returns to water–air flow modelling. Section 3.5 shows how the turbulence and water surface models are combined and it explains why the linear multigrid method is a good choice for the solution of both. The chapter ends with two sets of test cases. Four single-fluid problems in section 3.6 test the turbulent flow solver itself: the method proves efficient for different flows. The final section 3.7 uses the turbulence model to return to the Cahouet test case from section 2.5. The solver has retained the efficiency of the laminar method, while the results have become much more accurate.

3.1 Flow equations

This section gives a brief overview of the flow equations that we use, the Reynolds-Averaged Navier-Stokes (RANS) equations and Menter's turbulence model [46]. The equations are given here in the variable-density form of our two-fluid flow model (section 2.1). For the analysis in the following sections, the equations are simplified to a single-fluid form.

3.1.1 RANS equations For the RANS equations, the turbulent flow field is ensemble-averaged: the average is defined over many realisations of the same flow, with slightly different initial conditions [47]. In the equations for the averaged flow quantities, the only contribution from the turbulence that remains is a turbulent stress term. Under the Boussinesq hypothesis, this term can be modelled by simply adding a non-constant turbulent viscosity to the laminar viscosity in the standard Navier-Stokes equations. But due to this non-constant viscosity, the complete stress tensor must be taken into account, there are no more terms that cancel. Therefore, the diffusion part of the equations is more complex than in (2.5).

In two dimensions, the steady two-fluid RANS equations are:

$$\begin{aligned}
\frac{\partial}{\partial x}\left(p+\rho u^2\right)+\frac{\partial}{\partial y}\left(\rho uv\right) &= \frac{\partial}{\partial x}\left(\left(\mu+\mu_T\right)2u_x\right)+\frac{\partial}{\partial y}\left(\left(\mu+\mu_T\right)\left(u_y+v_x\right)\right),\\
\frac{\partial}{\partial x}\left(\rho uv\right)+\frac{\partial}{\partial y}\left(p+\rho v^2\right) &= \frac{\partial}{\partial x}\left(\left(\mu+\mu_T\right)\left(u_y+v_x\right)\right)+\frac{\partial}{\partial y}\left(\left(\mu+\mu_T\right)2v_y\right)-\rho g,\\
\frac{\partial}{\partial x}\left(u\right)+\frac{\partial}{\partial y}\left(v\right) &= 0,\\
\frac{\partial}{\partial x}\left(u\alpha\right)+\frac{\partial}{\partial y}\left(v\alpha\right) &= 0.
\end{aligned} \tag{3.1}$$

Here μ_T is the turbulent viscosity. As μ_T is not known exactly, it is modelled with an approximate turbulence model.

3.1.2 Menter's turbulence model To approximate μ_T, one or more equations are added to the system (3.1). Menter's turbulence model is a robust and accurate one-equation model, that computes the turbulent viscosity directly. It is similar to the Spalart-Allmaras model [57] and is used successfully for ship flows [52].

Menter derived his model for constant-density flow, by isolating the turbulent viscosity from the two-equation $k-\varepsilon$ model. The same procedure can be used to obtain a variable-density formulation of the model. However, using this formulation requires some care, because Menter tuned the parameters in his model using incompressible flows only. As we shall see in section 3.5, this poses no problems for water–air flows. The variable-density model in two dimensions is given by:

$$\frac{\partial(\rho u\tilde{\nu}_T)}{\partial x}+\frac{\partial(\rho v\tilde{\nu}_T)}{\partial y}=\frac{\partial}{\partial x}\left(\left(\mu+\frac{\rho\tilde{\nu}_T}{\sigma_m}\right)\frac{\partial\tilde{\nu}_T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\left(\mu+\frac{\rho\tilde{\nu}_T}{\sigma_m}\right)\frac{\partial\tilde{\nu}_T}{\partial y}\right)+P-D. \tag{3.2}$$

This is a convection–diffusion–reaction equation for the turbulent kinematic viscosity $\tilde{\nu}_T$: P and D are source terms modelling the production and dissipation of turbulence. To get correct behaviour of the model near walls, the actual μ_T to be used in (3.1) is scaled:

$$\mu_T=\rho\left(1-e^{-\left(\frac{\rho\tilde{\nu}_T}{A^+\kappa\mu}\right)^2}\right)\tilde{\nu}_T. \tag{3.3}$$

A^+ and κ are constants. The boundary conditions for $\tilde{\nu}_T$ are straightforward; this is one of the advantages of Menter's model. On a wall, $\tilde{\nu}_T=0$ as turbulence dies out near walls; no wall function is needed. And on inflow boundaries, a small positive value for $\tilde{\nu}_T$ is set, usually about $0.01\mu/\rho$; if $\tilde{\nu}_T$ at the inflow is zero, it remains zero throughout the domain because no turbulence is produced. The solution is not sensitive to the inflow $\tilde{\nu}_T$, as long as it is significantly smaller than the maximum $\tilde{\nu}_T$ [46].

The production and dissipation terms are the heart of the model. The production term is:

$$P=c_1\frac{\mu+\mu_T}{\mu+\rho\tilde{\nu}_T}\rho\tilde{\nu}_T\sqrt{2(u_x^2+v_y^2)+(u_y+v_x)^2}, \tag{3.4}$$

and the dissipation is:

$$D = c_2 c_3 D_{BB} \tanh\left(\frac{D_{k-\varepsilon}}{c_3 D_{BB}}\right), \tag{3.5}$$

with

$$D_{k-\varepsilon} = \rho\tilde{\nu}_T^2 \frac{(u_{xx}+u_{yy})^2 + (v_{xx}+v_{yy})^2}{u_x^2+u_y^2+v_x^2+v_y^2}, \qquad D_{BB} = \rho\left((\tilde{\nu}_{T_x})^2 + (\tilde{\nu}_{T_y})^2\right). \tag{3.6}$$

$D_{k-\varepsilon}$ is the main dissipation term. Equation (3.5) reduces to $D \approx c_2 D_{k-\varepsilon}$ whenever $D_{k-\varepsilon}$ is small, the limiting with $c_3 D_{BB}$ is only needed for regions where $D_{k-\varepsilon}$ is large due to small velocity derivatives. The $D_{k-\varepsilon}$ in (3.6) is actually an alternative form that is suggested, but not used, by Menter. It is slightly more complex than Menter's original form, but it can be discretised on a five-point stencil (see section 3.4). Both the production and the dissipation term are invariant under rotation and the production term P is always positive, while the dissipation $-D$ is always negative.

The model constants have the values $c_1 = 0.144$, $c_2 = 1.86$, and $A^+ = 13.0$. The von Karman constant $\kappa = 0.41$. Furthermore, $c_3 = 7$ and $\sigma_m = 1$.

3.2 Line smoothing for Menter's model

The RANS equations with Menter's turbulence model will be solved with a multigrid technique combined with alternating line Gauss-Seidel smoothing. Standard multigrid, as presented in section 2.3 for the laminar Navier-Stokes equations, cannot be used directly for the RANS equations. This has two reasons. One: because of the source terms in the turbulence equation, the flow equations are no longer positive definite, so the line smoothing does not converge. And two: the solutions on the coarse grids do not resemble the solutions on the fine grid enough.

The following sections give the solution to these problems. In section 3.2.1, the problem with the flow equations is analysed; an improved smoother is presented in section 3.2.2. And section 3.3 gives a working coarse grid correction algorithm.

For the analysis, the single-fluid RANS equations are used: ρ is divided out from the equations in the previous section and μ is replaced by the kinematic viscosity ν. The system (3.1) plus the Menter model (3.2) is denoted by $\mathcal{F}(\boldsymbol{q})$, $\boldsymbol{q} = [u, v, p, \tilde{\nu}_T]^T$. Also, we limit ourselves to five-point stencils: for each cell, the state in that cell and in its four neighbours is used.

3.2.1 Source term and negative eigenvalues Classical relaxation techniques, like line smoothing, only work when the discretised system $\mathcal{F}_k$ is of vector-positive type [10]. This means that the system has no unstable eigenmodes: when $\mathcal{F}_k$ is linearised, all eigenvalues λ of the resulting linear system must have $\Re(\lambda) > 0$. Discretisations of convection-diffusion equations, like the laminar Navier-Stokes equations, may be vector-positive; this depends on the discretisation. For example, the artificial compressibility discretisations by Dick and his co-workers [10, 12] are always vector-positive, but their later AUSM+ based discretisations [50, 67] are not.

In the RANS equations, a more fundamental problem appears. There, the *continuous* system makes it impossible to find a discretisation that is always vector-positive. The occurrence of eigenvalues with $\Re(\lambda) \leq 0$ is caused by the source term in the turbulence equation.

Linearised flow equations. To study the effect of the source term, we construct a linearised version of the continuous single-fluid system $\mathcal{F}$, like we did for the discretised laminar two-fluid system, in section 2.4.1. We choose a function $\boldsymbol{Q} = [U, V, P, N_T]^T$ and write functions close to $\boldsymbol{Q}$ as $\boldsymbol{q} = \boldsymbol{Q} + \epsilon \boldsymbol{q}'$. Substituting this in (3.1) and (3.2) gives a linear operator L for the small disturbances $\epsilon \boldsymbol{q}'$, such that:

$$\mathcal{F}(\boldsymbol{Q} + \epsilon \boldsymbol{q}') = \mathcal{F}(\boldsymbol{Q}) + \epsilon L \boldsymbol{q}' + \mathcal{O}(\epsilon^2). \tag{3.7}$$

We find:

$$L = \begin{bmatrix}
2U\partial_x + V\partial_y & U\partial_y & \partial_x & -2U_x\partial_x \\
-N(2\partial_{xx} + \partial_{yy}) & -N\partial_{xy} & & -(U_y + V_x)\partial_y \\
V\partial_x & U\partial_x + 2V\partial_y & \partial_y & -(U_y + V_x)\partial_x \\
-N\partial_{xy} & -N(\partial_{xx} + 2\partial_{yy}) & & -2V_y\partial_y \\
\partial_x & \partial_y & 0 & 0 \\
N_T\partial_x & N_T\partial_y & 0 & U\partial_x + V\partial_y - N(\partial_{xx} + \partial_{yy}) \\
-P_2(2U_x\partial_x + (U_y + V_x)\partial_y) & -P_2((U_y + V_x)\partial_x + 2V_y\partial_y) & & -N_{T\,x}\partial_x - N_{T\,y}\partial_y \\
+D_2(U_{xx} + U_{yy})(\partial_{xx} + \partial_{yy}) & +D_2(V_{xx} + V_{yy})(\partial_{xx} + \partial_{yy}) & & -P_1 + D_1 \\
+D_3(U_x\partial_x + U_y\partial_y) & +D_3(V_x\partial_x + V_y\partial_y) & &
\end{bmatrix}. \tag{3.8a}$$

For simplicity, we use $D = c_2 D_{k-\varepsilon}$ instead of equation (3.5) and the low-Reynolds correction (equation (3.3)) is not used, so $\nu_T = \tilde{\nu}_T$. This makes no large difference to the system. The symbols like ∂_x and ∂_{xx} denote differentiation operators, the abbreviations are:

$$N = \nu + N_T, \tag{3.8b}$$

representing the total viscosity, and:

$$P_1 = c_1\sqrt{2(U_x^2 + V_y^2) + (U_y + V_x)^2}, \quad P_2 = c_1\frac{1}{\sqrt{2(U_x^2 + V_y^2) + (U_y + V_x)^2}}, \tag{3.8c}$$

$$\begin{aligned}
D_1 &= 2c_2 N_T \frac{(U_{xx} + U_{yy})^2 + (V_{xx} + V_{yy})^2}{U_x^2 + U_y^2 + V_x^2 + V_y^2}, \quad D_2 = 2c_2 N_T^2 \frac{1}{U_x^2 + U_y^2 + V_x^2 + V_y^2}, \\
D_3 &= -2c_2 N_T^2 \frac{1}{(U_x^2 + U_y^2 + V_x^2 + V_y^2)^2},
\end{aligned} \tag{3.8d}$$

representing parts of the production and dissipation terms.

The operator in the first three rows, first three columns is the laminar Navier-Stokes operator. We see the continuity equation $\partial_x u + \partial_y v = 0$ in the third row and the momentum equations with convection and viscous diffusion in the first two rows.

The fourth row and column add the effect of turbulence: in the fourth column, we see terms due to the non-constant viscosity; the fourth row contains the turbulence equation, where the terms are combinations of convection-diffusion effects and source term contributions.

These combined terms cause zero eigenvalues in the system (the limiting case of eigenvalues with $\Re(\lambda) \leq 0$), when they cancel each other out. For example, in the (4,4) term, the convection and the non-constant diffusion term are comparable, but they have different constants. The source term also has an important effect on the (4,4) term. Altogether, situations can arise where the fourth equation does not change when the local $\tilde{\nu}'_T$ changes. In that case, the operator has a zero eigenvalue. In a realistic flow solution, any combination of values for U, V, N_T and the source term parameters P_1, P_2, D_1, D_2, and D_3 may appear, so zero eigenvalues are bound to arise. They appear either by this cancelling of the (4,4) term or by interaction of the (1–2,4) terms with the (4,1–2) terms. This effect is physically explainable and not limited to the Menter model: it happens for any turbulence model, in regions where the turbulence production and the convection and diffusion cancel each other.

One could think that the use of stable discretisations, e.g. upwinding, can make a discretised version of this system vector-positive. Unfortunately, that is not true: the terms in (3.8) that may cancel each other are part of totally different parts of the operator. Those parts, like the convection operator and the source term, cannot be combined, so they cannot be discretised with one stable technique. Therefore, in a discretised system, situations where the system is not vector-positive cannot be prevented.

Determinant. To understand the nature of the system, it is useful to study the determinant of (3.8a). This determinant is:

$$\begin{aligned}\det(L) = -\Delta\left[N\Delta - (U\partial_x + V\partial_y)\right]\left[N\Delta - \left((U - N_{T_x})\,\partial_x + \left(V - N_{T_y}\right)\partial_y\right) + P_1 - D_1\right] \\ +(U_y + V_x)\,(\partial_{yy} - \partial_{xx}) \\ \left[P_2(U_y + V_x)\,(\partial_{yy} - \partial_{xx}) + D_2\left((V_{xx} + V_{yy})\,\partial_x - (U_{xx} + U_{yy})\,\partial_y\right)\Delta + D_3\,(V_x\partial_{xx} - U_y\partial_{yy})\right],\end{aligned} \tag{3.9}$$

where Δ is the Laplace operator $\partial_{xx} + \partial_{yy}$. This determinant is much more complex than the one for the laminar Navier-Stokes operator, given here for comparison:

$$\det(L_{\text{lam}}) = -\Delta\left(N\Delta - (U\partial_x + V\partial_y)\right). \tag{3.10}$$

See also the two-fluid determinant (2.58). The main difference is, that the turbulent determinant has two separate terms, while the laminar determinants (even the two-fluid one) have only one product of terms. The second term is a combination of the turbulent viscosity effect on the momentum equations and the velocity effects in the turbulent source term. The operator has zero eigenvalues when $\det(L) = 0$. In the laminar case, this happens only when one of the terms of the product is zero. In the turbulent case, it happens whenever the two terms cancel each other out.

In multigrid, one usually determines the type of the system by looking at the highest derivatives in the determinant. But since this system is singularly perturbed (ν is small), we study the terms with the highest two derivatives, i.e. those of fifth

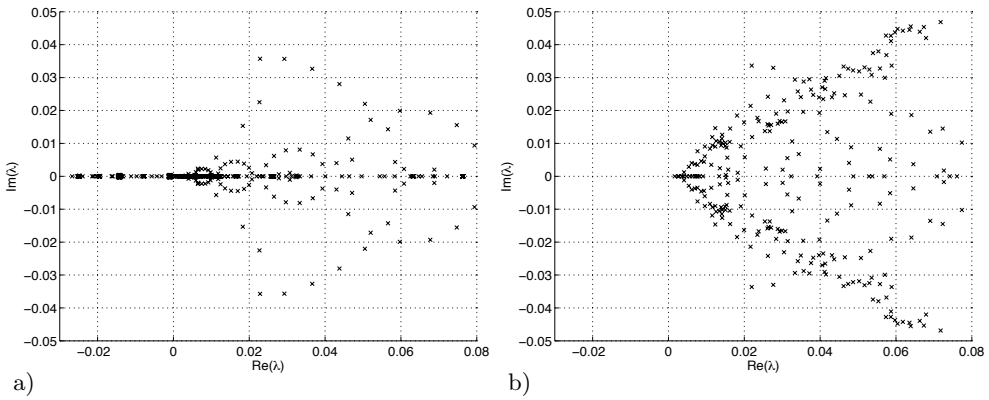
and sixth order. This leads to:

$$\det(L) \approx -N^2\Delta^3 + N\Delta^2 \left((2U - N_{T_x})\,\partial_x + \left(2V - N_{T_y}\right)\partial_y\right) + D_2(U_y + V_x)\left(\partial_{yyyy} - \partial_{xxxx}\right)\left((V_{xx} + V_{yy})\,\partial_x - (U_{xx} + U_{yy})\,\partial_y\right). \quad (3.11)$$

Even here, a part of the second term remains. And even though the term D_2 contains N_T^2, it is $\mathcal{O}(N_T)$ in a boundary layer, where U_y is $\mathcal{O}(N_T^{-1/2})$. Therefore, its contribution is non-trivial: the source term really has an effect on the occurrence of zero eigenvalues.

Stability near a solution. An interesting property of the system saves us: if the turbulence model is stable, then the operator has no eigenvalues with $\Re(\lambda) \leq 0$ near a solution of the steady RANS equations. This can be seen by considering the time-dependent flow equations and substituting a steady solution with a small disturbance. If any of the eigenvalues of L would have a negative real part, then the disturbance would grow in time, so the flow would be unstable. Thus, the existence of a stable steady solution guarantees that all λ have $\Re(\lambda) > 0$ near that solution. This is good news, as a properly designed turbulence model must have stable steady solutions. It is the task of the turbulence model designer to guarantee this. Therefore, we may assume that near a solution, L always has $\Re(\lambda) > 0$ for all eigenvalues λ.

Figure 3.1 gives an example. It shows a part of the eigenvalue spectrum for a discretised version of (3.1), (3.2). In the first figure, the spectrum is given for a state that is far away from the converged solution. We see a large number of eigenvalues with negative real parts. When the flow is converged (figure 3.1b), all these λ have moved to the right and crossed the imaginary axis. No eigenvalues with $\Re(\lambda) \leq 0$ remain.



a) b)

Figure 3.1. Detail of the eigenvalue spectrum (leftmost eigenvalues) for the discretised RANS operator (the discretisation from section 3.4, applied to the zero pressure-gradient boundary layer problem from section 3.6.2, on a 32×32 cell grid). Graph (a): state after one line GS iteration, starting with a uniform flow. Graph (b): converged solution.

Concluding, we have found:

1. that the linearised operator L may have eigenvalues with $\Re(\lambda) \leq 0$, that prevent the convergence of line Gauss-Seidel,
2. that this is *not* the case for converged solutions,
3. that the interaction between the turbulent viscosity in the momentum equations and the velocity contributions to the source term plays an important role in the behaviour of the system.

In the following, these findings are used to construct an efficient multigrid algorithm.

3.2.2 Improved line smoothing The previous section showed why normal line smoothing cannot be used to solve the RANS equations. Since line smoothing works well for the laminar Navier-Stokes equations, it is tempting to consider separate smoothing of the laminar state variables (u, v, and p) and the turbulence $\tilde{\nu}_T$, alternately updating $\tilde{\nu}_T$ or the laminar state variables while the other(s) are kept fixed. This technique works for time steppers, but not for the more powerful Gauss-Seidel smoothing. The coupling between velocity and $\tilde{\nu}_T$ in the source term is so strong that smoothing u and v while keeping $\tilde{\nu}_T$ frozen may send the state in a wrong direction, increasing the error. In some of our experiments, errors increased with a factor 100 or more in a single smoothing step! Therefore, it is essential that all four state variables are smoothed together. This section describes how such a smoother is made stable.

Damped Gauss-Seidel. The alternating line Gauss-Seidel smoothing is stabilised by adding local damping. Instead of applying the line smoother to $\mathcal{F}_k(\boldsymbol{q}_k) = 0$, we apply it, in each smoothing step n, to the system:

$$\mathcal{F}_k(\boldsymbol{q}_k^{n+1}) + \tau I_4 \boldsymbol{q}_k^{n+1} = \tau I_4 \boldsymbol{q}_k^n. \tag{3.12}$$

Applying one Gauss-Seidel step gives an approximate solution of this system. The positive function τ may be non-constant in space and different for different n. I_4, for each cell, is a 4×4 zero matrix with 1 in the (4,4) position. Thus, we damp only the corrections of the turbulent viscosity; equation (3.12) is similar to implicit time stepping for $\tilde{\nu}_T$ only. Linearised, it reads:

$$(L_k + \tau I_4)\,(\boldsymbol{q}_k')^{n+1} = \tau I_4 (\boldsymbol{q}_k')^n, \tag{3.13}$$

where L_k is the linearisation of $\mathcal{F}_k$.

We see:

1. that, if the damped Gauss-Seidel process converges, it converges to $\mathcal{F}_k(\boldsymbol{q}_k) = 0$,
2. that line smoothing is a good solver for the system (3.12) if the linear operator $(L_k + \tau I_4)$ has $\Re(\lambda) > 0$, $\forall\lambda$,
3. that, since (3.12) resembles an implicit time stepping procedure for $\tilde{\nu}_T$, it is expected to converge if a stable steady solution exists.

As seen in section 3.2.1, the first three rows and columns of L_k form the laminar Navier-Stokes operator, which has no eigenvalues with $\Re(\lambda) \leq 0$. Therefore, L_k can be made diagonally dominant by increasing the (4,4) term only; $L_k + \tau I_4$ has no $\Re(\lambda) \leq 0$ eigenvalues when τ is chosen sufficiently large.

However, it is not necessary — rather, it is wasteful — to set τ large everywhere, all the time. As we have seen, the $\Re(\lambda) \leq 0$ eigenvalues disappear when the solution process converges, so damping is only needed in the early stages of the solution. And in most of the flow field, certainly outside boundary layers and turbulent wakes, no $\Re(\lambda) \leq 0$ eigenvalues appear anyway, so no damping is needed there either.

Therefore, we take τ constant in a line. We set τ for each individual line, for every smoothing step, to the smallest value that gives only positive eigenvalues. Thus, the τ in a certain cell may be different for successive horizontal and vertical sweeps. The choice of τ is explained below.

Estimating τ. The smallest possible τ can be estimated, very elegantly, with the Newton-Raphson algorithm that is used to solve the nonlinear flow equations in the individual lines. Newton-Raphson (NR) relies on linearisations $\partial \mathcal{F}_k / \partial \boldsymbol{q}_k$ of the flow equations $\mathcal{F}_k$ in each line to find the roots of these sets of nonlinear equations. But if one or more of the eigenvalues of the linearised flow equations are zero, then the corresponding part of the nonlinear system is dominated by higher-order effects. In that case, a linearisation is no longer a good approximation of the nonlinear system, so NR loses its quadratic convergence. It converges slowly, or not at all.

Therefore, we choose τ locally by monitoring the convergence rate of NR for each line. Each smoothing step is started with a low value for τ, constant in the whole domain. When the NR iteration in a line does not converge fast enough, it is restarted with a larger τ for that line. This is repeated, if necessary, until a sufficient convergence rate for NR is obtained. Then we can be sure that $L_k + \tau I_4$ has no eigenvalues close to zero, that are associated with that line. And, as the eigenvalues near 0 lie close together (figure 3.1), this indicates that all eigenvalues have $\Re(\lambda) > 0$. The advantage of this procedure is, that the convergence of NR has to be monitored anyway, to determine when to stop the iteration; no costly, complicated estimation of eigenvalues is needed.

The basis for a definition of sufficiently fast convergence for NR is the residual of the flow equations in the line, after l Newton-Raphson steps:

$$r^l = \sum_{\text{line}} \left| \mathcal{F}(\boldsymbol{q}^l_{\text{NR}}) + \tau I_4 (\boldsymbol{q}^l_{\text{NR}} - (\boldsymbol{q}^n_k)_{i,j \in \text{line}}) \right| . \tag{3.14}$$

When applied to a system without zero eigenvalues, NR ideally shows quadratic convergence: $r^l \sim (r^{l-1})^2$. But in practice, this is seldom obtained, even with τ sufficiently large. For high residuals, nonlinear effects dominate the system behaviour and for very small residuals, little inaccuracies in the computer code often prevent convergence below a certain threshold. Therefore, basing τ on a test for quadratic convergence is too restrictive. Instead, we require a specified reduction in r within a fixed number of steps:

$$r^l / r^0 < \epsilon \text{ for some } l \leq l_{\max}. \tag{3.15}$$

If this criterion is not met for a line, then τ is increased. If it is, we consider the iteration converged.

The convergence criterion (3.15) is different from the usual NR convergence criterion (which ends the iteration when $r^l < \epsilon$), for a very good reason. It is expensive to find the smallest τ exactly by trial and error, so each time when the criterion (3.15) is not met, we increase τ significantly (by a factor 10, in our experiments). But in situations where the criterion is *almost* met without increasing τ, a very small reduction in the initial residual r^0 could mean that the criterion $r < \epsilon$ is suddenly met without a higher τ. This leads to many sudden changes in τ for different sweeps in the same line; such 'jittery' behaviour can prevent convergence of the Gauss-Seidel smoothing. When the relative convergence criterion (3.15) is used, τ is more consistent between successive sweeps of the same line.

Smoothing algorithm. Summarizing, the algorithm for the Gauss-Seidel smoothing in one line is:

$\tau = \tau_0$ — initial τ

while $\tau \leq \tau_{\max}$

  $\boldsymbol{q}_{\mathrm{NR}}^0 = (\boldsymbol{q}_k^n)_{i,j\in\mathrm{line}}$ — initial state

  for $l = 1, l_{\max}$

    $\boldsymbol{q}_{\mathrm{NR}}^l = \boldsymbol{q}_{\mathrm{NR}}^{l-1} - \left(\dfrac{\partial \mathcal{F}_k}{\partial \boldsymbol{q}_k} + \tau I_4\right)^{-1}_{\mathrm{line}} (\mathcal{F}_k + \tau I_4)_{\mathrm{line}} \left(\boldsymbol{q}_{\mathrm{NR}}^{l-1}\right)$ — NR step

    if $r^l/r^0 < \epsilon$ **end**

  end for

  $\tau := \tau_{\mathrm{factor}} \cdot \tau$ — new τ

end while

As stated above, this is an estimator for the smallest τ, it is not exact. But practical experience shows that this algorithm functions well. We found that suitable values for the parameters (for flows with $u, v, p = \mathcal{O}(1)$) are $\tau_0 = 10^{-2}$, $\tau_{\max} = 10^7$, $\tau_{\mathrm{factor}} = 10$, $\epsilon = 10^{-5}$, and $l_{\max} = 10$.

We remark that in many solution methods for the RANS equations, e.g. [40, 45], the source term is split in a positive and a negative part. For each smoothing step or time step, the positive part (the production) is kept constant, while the negative part (the dissipation) is updated together with the convective and diffusive fluxes. This is a kind of damping too, it increases the diagonal dominance of L_k. The difference with our algorithm is, that this damping by source-term splitting is not switched off, even when it is no longer needed because the solution approaches convergence. Therefore, our damping algorithm is more efficient.

3.3 Linear multigrid

Nonlinear multigrid (NMG), as described in section 2.3, does not work for the RANS equations. In tests that we did, application of NMG usually gives worse convergence

than Gauss-Seidel smoothing alone, or even causes divergence of the solution process. The $\Re(\lambda) \leq 0$ eigenvalues described in the previous section are *not* the cause of this problem: multigrid has been applied successfully to problems with both positive and negative eigenvalues (see e.g. [63], section A8.5.3 for the solution of the Helmholtz problem). Given a good smoother, which we have, multigrid ought to work. But it does not.

NMG operators fail. The problem is, that the coarse grid operators $\mathcal{F}_k$ do not resemble the fine grid operator $\mathcal{F}_K$ sufficiently well. In turbulent areas, the source term in Menter's model is the small difference of two large variables (production and dissipation), that contain products of both first and second spatial derivatives. Thus, small errors in these derivatives may cause large differences in the source term; the turbulence model only makes sense when the grid is fine enough to resolve the interior structure of a boundary layer well. Therefore, the model needs a minimum grid resolution to be accurate, typically about 20 cells over the thickness of a boundary layer. For coarser grids, the solution becomes highly grid-dependent. An example for a boundary layer flow is given in figure 3.2. The solution on the 32×32 cell grid is accurate (compare with the fine-grid solution in figure 3.5), but the solutions on the coarser grids are a lot worse. Note that the maximum value for $\tilde{\nu}_T$ is off by about 30% on the 16×16 grid and that this maximum is actually lower than the one on the 8×8 grid.

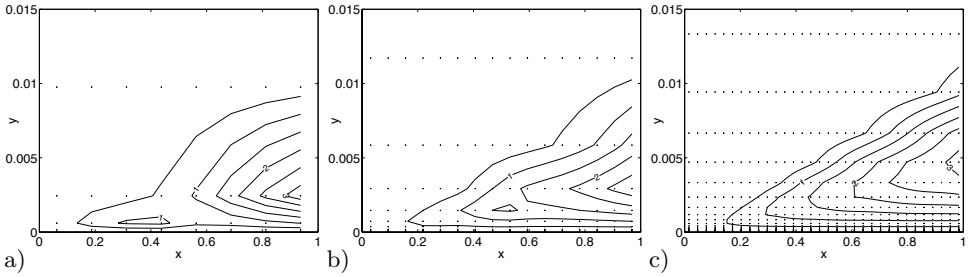


Figure 3.2. Grid convergence study for a zero pressure-gradient boundary layer flow at $Re = 10^7$. Solutions for $\tilde{\nu}_T$ on an 8×8 grid (a), 16×16 grid (b), and 32×32 grid (c). The cell centres are marked by ×. (The test case is also described in section 3.6.2.)

If the solutions on the coarse grids differ from the solutions on the fine grid, then the operators that produce these solutions differ too. Therefore, the coarse grid operators $\mathcal{F}_k$ are not suitable for constructing approximate inverses for $\mathcal{F}_K$ and the NMG coarse grid correction does not work. This is confirmed by our tests, which show that acceptable coarse grid corrections can be obtained with NMG, but only when the coarsest grid used is fine enough to accurately resolve the boundary layers. If coarser grids are used, the convergence deteriorates. However, for good multigrid convergence, proper smoothing on these coarse grids is essential.

Galerkin operators. So, to get proper coarse grid corrections, we use Galerkin coarse grid operators. Instead of restricting the state on the fine grid to the coarser

grids and constructing coarse grid operators with these states, we directly restrict the fine grid operator to the coarse grids. Thus, we can be sure that the coarse grid operators resemble the fine grid operator reasonably well. The Galerkin operators are found by first constructing a linearised version of the fine grid operator and then restricting this linear operator to the coarse grids. So we switch from nonlinear to linearised coarse grid corrections. Therefore, we no longer compute states $\boldsymbol{q}$ on the coarse grids, but small corrections $\boldsymbol{u}$.

The linearised version of the fine grid operator $\mathcal{F}_K$ is the discretised equivalent of the operator L from equation (3.7), i.e. the Jacobian of $\mathcal{F}_K$:

$$L_K = \frac{\partial \mathcal{F}_K(\boldsymbol{q}_K)}{\partial \boldsymbol{q}_K}. \tag{3.16}$$

For a finite-volume discretisation with a five-point stencil, the linear operator in one cell is:

$$\begin{aligned}(L_K \boldsymbol{u}_K)_{i,j} = L^{0,0}_{i,j}(\boldsymbol{u}_K)_{i,j} + L^{1,0}_{i,j}(\boldsymbol{u}_K)_{i+1,j} + L^{0,1}_{i,j}(\boldsymbol{u}_K)_{i,j+1} \\ + L^{-1,0}_{i,j}(\boldsymbol{u}_K)_{i-1,j} + L^{0,-1}_{i,j}(\boldsymbol{u}_K)_{i,j-1},\end{aligned} \tag{3.17a}$$

where

$$L^{0,0}_{i,j} = \frac{\partial(\mathcal{F}_K)_{i,j}}{\partial(\boldsymbol{q}_K)_{i,j}}, \qquad L^{1,0}_{i,j} = \frac{\partial(\mathcal{F}_K)_{i,j}}{\partial(\boldsymbol{q}_K)_{i+1,j}}, \qquad L^{0,1}_{i,j} = \frac{\partial(\mathcal{F}_K)_{i,j}}{\partial(\boldsymbol{q}_K)_{i,j+1}}, \quad \text{etc.} \tag{3.17b}$$

So for each cell $(\Omega_K)_{i,j}$, the operator L_K consists of five 4×4 matrices.

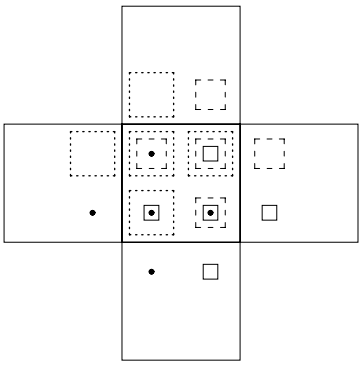


Figure 3.3. Construction of a coarse grid linear operator. The matrices in the coarse five-point stencil (—) are made by summing four fine grid stencils ($\cdots$, - -, —, •).

The coarse grid operators then follow from Galerkin's principle:

$$L_{k-1}\boldsymbol{u}_{k-1} = R^{k-1}_k L_k P^k_{k-1} \boldsymbol{u}_{k-1}. \tag{3.18}$$

With the prolongation (2.39) and the restriction (2.40), this means that the coarse grid operator L_{k-1} can be evaluated by copying the correction $\boldsymbol{u}_{k-1}$ in all coarse cells

$(\Omega_{k-1})_{i,j}$ to the four fine cells $(\Omega_k)_{2i(+1),2j(+1)}$ that lie in the same location as these coarse cells, then evaluating the fine grid operator L_k for this $\boldsymbol{u}_k$ and finally summing the fluxes from each group of four cells. This, in turn, means that the matrices for L_{k-1} are found by a summation of the matrices in L_k: the 20 matrices of the four operators in the fine cells corresponding to coarse cell $(\Omega_{k-1})_{i,j}$ are summed to form the five matrices in $(L_K)_{i,j}$, such that a matrix for a fine cell is added to the matrix for that coarse cell in which the fine cell lies. Figure 3.3 demonstrates this.

LMG algorithm. With these coarse grid operators, the linear multigrid algorithm is constructed. It is different for the finest level and the coarser levels as only the operator on the finest grid is nonlinear:

$\underline{\boldsymbol{q}_K^{n+1} = \textbf{function } LMG(\boldsymbol{q}_K^n):}$

$\tilde{\boldsymbol{q}}_K^n = (M_K)^{q_1}\, \boldsymbol{q}_K^n$	q_1 pre-relaxation steps,
$L_K^n = \partial \mathcal{F}_K(\tilde{\boldsymbol{q}}_K^n)/\partial \boldsymbol{q}_K$	linearisation,
$\boldsymbol{s}_{K-1}^n = R_K^{K-1}\mathcal{F}_K(\tilde{\boldsymbol{q}}_K^n)$	source term,
$L_{K-1}^n = R_K^{K-1} L_K^n P_{K-1}^K$	linearised system,
$\boldsymbol{u}_{K-1}^n = 0$	initial solution,
$\boldsymbol{u}_{K-1}^{n+1} = LMGC^\sigma\left(K-1, \boldsymbol{u}_{K-1}^n, \boldsymbol{s}_{K-1}^n, L_{K-1}^n\right)$	MG on coarser grid,
$\tilde{\tilde{\boldsymbol{q}}}_K^n = \tilde{\boldsymbol{q}}_K^n + P_{K-1}^K \boldsymbol{u}_{K-1}^{n+1}$	correction,
$\boldsymbol{q}_K^{n+1} = (M_K)^{q_2}\, \tilde{\tilde{\boldsymbol{q}}}_K^n$	q_2 post-relaxation steps.

$\underline{\boldsymbol{u}_k^{n+1} = \textbf{recursive function } LMGC(k, \boldsymbol{u}_k^n, \boldsymbol{s}_k^n, L_k^n):}$

$\tilde{\boldsymbol{u}}_k^n = \left(M_k^L\right)^{q_1}\, \boldsymbol{u}_k^n$	q_1 pre-relaxation steps,
if $k \neq 0$ **then**	
$\boldsymbol{s}_{k-1}^n = R_k^{k-1} L_k \tilde{\boldsymbol{u}}_k^n$	source term,
$L_{k-1}^n = R_k^{k-1} L_k^n P_{k-1}^k$	linearised system,
$\boldsymbol{u}_{k-1}^n = 0$	initial solution,
$\boldsymbol{u}_{k-1}^{n+1} = LMGC^\sigma\left(k-1, \boldsymbol{u}_{k-1}^n, \boldsymbol{s}_{k-1}^n, L_{k-1}^n\right)$	MG on coarser grid,
$\tilde{\tilde{\boldsymbol{u}}}_k^n = \tilde{\boldsymbol{u}}_k^n + P_{k-1}^k \boldsymbol{u}_{k-1}^{n+1}$	correction,
end if	
$\boldsymbol{u}_k^{n+1} = \left(M_k^L\right)^{q_2}\, \tilde{\tilde{\boldsymbol{u}}}_k^n$	q_2 post-relaxation steps.

For the smoothing on the coarser grids, we use line Gauss-Seidel, just like on the fine grid. Only, the Newton-Raphson iteration is not needed anymore, as the linear flow equations in a line can be solved exactly in one step. As for the NMG algorithm, each smoothing step M_k consists of both a horizontal and a vertical sweep. This smoothing is relatively expensive, but it increases the robustness of the procedure.

Comments on the algorithm. The change from nonlinear to linear coarse grid corrections is not that big. For small residuals, the NMG algorithm from section 2.3 is more or less linear anyway: the computation of $\mathcal{F}_{k-1}(\boldsymbol{q}_{k-1}^{n+1}) = \mathcal{F}_{k-1}(\boldsymbol{q}_{k-1}^{n}) + \boldsymbol{d}_{k-1}^{n}$ is a matrix-free evaluation of the Jacobian of $\mathcal{F}_{k-1}$ when $\boldsymbol{d}_{k-1}^{n}$ is small, i.e. when $\mathcal{F}_{k-1}(\boldsymbol{q}_{k-1}^{n+1}) \approx \mathcal{F}_{k-1}(\boldsymbol{q}_{k-1}^{n}) + L_{k-1}(\boldsymbol{q}_{k-1}^{n+1} - \boldsymbol{q}_{k-1}^{n})$. And as complex systems like the RANS equations cannot handle large source terms, the defect $\boldsymbol{d}$ is *always* made small by reducing the parameter w in equation (2.41c) when $\boldsymbol{d}$ is large (see section 2.3.3); reducing w is equivalent to linearising the coarse grid operators.

The only real difference between the NMG and LMG algorithms is that the LMG coarse grid operators are Galerkin operators. According to the literature, Galerkin operators may cause problems (see e.g. [63], section 5.4). In particular, Galerkin coarse grid operators for convection problems are less diagonally dominant than the fine grid operator. As a consequence, the convergence rate for multigrid becomes worse when more coarse grids are added. In our experiments (section 3.6) this effect is noted, but it causes no problems.

The unusual combination of a linear coarse grid correction with nonlinear smoothing on the finest grid is chosen for two reasons. First, the nonlinear smoother is robust, it acts like a 'safety net' for the solution process. It can correct small unphysical solutions ($\tilde{\nu}_T < 0$, for example) that arise from the coarse grid correction. Furthermore, it performs well in extreme situations, like the start of the solution process from a uniform flow, where the linear multigrid method is not (yet) effective. And second, we use the damping factor τ from the line smoothing (equation (3.12)) in the coarse grid correction too, i.e. we replace (3.16) with:

$$L_K = \frac{\partial \left(\mathcal{F}_K(\boldsymbol{q}_K) + \tau I_4 \boldsymbol{q}_K \right)}{\partial \boldsymbol{q}_K}. \tag{3.19}$$

As explained in the introduction to this section, this damping is not really needed for the convergence of multigrid, but it makes sure that the smoothing on all coarse grids is stable (we cannot set τ with the aid of Newton-Raphson on the coarse grids, because the coarse grid operators are linear). The nonlinear smoothing on the finest grid is needed to find the τ in equation (3.19): for each cell, we choose τ as the minimum of the τ's for that cell in the horizontal and vertical line sweep.

Concerning the computational cost of the linear multigrid algorithm, a single coarse grid correction step takes less CPU time than a nonlinear one. Computing L_K, the linearisation of $\mathcal{F}_K$, is cheap because the Jacobian of $\mathcal{F}_K$ in each cell is already computed for the nonlinear line smoothing. The restriction of the operators L_k does not take much time either, as it consists of additions only. And finally, the coarse grid smoothing becomes faster, because it does not need Newton-Raphson anymore and because the evaluation of the linear operators is faster than the computation of nonlinear fluxes. If the Galerkin operators require more iterations to reach convergence, the total CPU time may go up a little; this depends on the individual problem.

The main increase is in the memory usage. Storing the linearised operators takes $5 \cdot (4 \cdot 4) = 80$ reals per cell, much more than the storage needed for nonlinear

multigrid. There are two ways to reduce these costs: one is to restrict the fine grid operator L_K to the next coarser grid while it is being computed. This means that storing L_K itself is not needed, a significant gain as more than 75% of all cells lie in the finest grid. The other is to store the linearised operators with a low precision. Their inverse is not computed exactly anyway, so double precision accuracy is not needed for the L_k. This may save another 50% – 75% in memory.

Full multigrid. As initial condition for the multigrid solution, we usually choose $\tilde{\nu}_T$ very close to zero. Then we can see two distinct 'stages' in the solution process. During the first iterations the boundary layers develop and the turbulence intensity grows, often with a factor $\mathcal{O}(10^4)$ or more. Typically, we see the residual in the turbulence equation increase in this first stage, because the turbulence intensity increases. Then, when the boundary layers have more or less developed, the residuals start to decrease: this is the second stage. Experiments show that multigrid is only effective in this second stage.

Therefore, full multigrid (FMG) is essential for our method. We do not start the solution on the finest grid, but on the coarsest grid on which the boundary layers can be accurately resolved. When the solution on this grid is computed, it is prolongated as initial solution to the next finer grid where the solution is computed too. This is continued until the finest grid is reached. Thus, all but the first computations start at the second stage, with developed boundary layers. The time-consuming development of turbulence is only needed on the coarsest grid.

3.4 Finite-volume discretisation

The multigrid solution technique presented in the previous sections can, in principle, be applied to different types of discretisations. Therefore, the description of the discretisation for the system (3.1), (3.2) was kept general. In this section, we show how turbulence is included in the finite-volume discretisation from section 2.2. Turbulent flows often have high Reynolds numbers, so the separate stable discretisation of the convective and diffusive fluxes is useful, even for single-fluid flow. This section focuses on those aspects that are specific for the RANS equations: the non-constant viscosity for the diffusion, the turbulent source terms, and the boundary conditions for $\tilde{\nu}_T$.

Convective fluxes. The convective part of the RANS equations is the same as the convective part of the laminar NS equations (2.5). Therefore, it is discretised in the same way, with the artificial compressibility Riemann solver from section 2.2.2. The convective part of the turbulent viscosity equation (3.2) is similar to the equation for the volume fraction α, so $\tilde{\nu}_T$ at the cell faces is also chosen upwind:

$$\tilde{\nu}_{T\frac{1}{2}} = \tilde{\nu}_{T0} \qquad \text{if } u_{\frac{1}{2}} \geq 0, \qquad \tilde{\nu}_{T\frac{1}{2}} = \tilde{\nu}_{T1} \qquad \text{if } u_{\frac{1}{2}} < 0. \tag{3.20}$$

The convective flux is constructed with this $\tilde{\nu}_{T\frac{1}{2}}$.

Diffusive fluxes. On our non-rectangular grids, the derivatives in the diffusive fluxes are discretised with Peyret control volumes (figure 2.2). But here, it is not

sufficient to compute the normal derivatives only. The cross-derivative terms like u_{xy} (u_y in the x-direction flux) do not cancel, because the viscosity is not constant: the diffusive operator is similar to the one for the laminar compressible Navier-Stokes equations. Therefore, velocity derivatives in two directions are needed on each face. For simplicity, the x- and y-derivative are computed directly:

$$\begin{aligned} u_x &\approx \frac{1}{A_d} \oint_{\partial\Omega_d} u \, dy \\ &\approx \frac{1}{A_d} \left(\frac{u_1 + u_2}{2}(y_2 - y_1) + u_3(y_4 - y_2) + \frac{u_4 + u_5}{2}(y_5 - y_4) + u_i(y_1 - y_5) \right), \end{aligned} \tag{3.21a}$$

$$\begin{aligned} u_y &\approx -\frac{1}{A_d} \oint_{\partial\Omega_d} u \, dx \\ &\approx \frac{1}{A_d} \left(\frac{u_1 + u_2}{2}(x_1 - x_2) + u_3(x_2 - x_4) + \frac{u_4 + u_5}{2}(x_4 - x_5) + u_i(x_5 - x_1) \right). \end{aligned} \tag{3.21b}$$

The same equation is used for v and $\tilde{\nu}_T$.

Compared with the diffusive fluxes in the laminar incompressible Navier-Stokes equations, the RANS equations cause two complications. First, the cross-derivative terms do not cancel. For the discretisation, this means that the diffusion operator requires a nine-point stencil, even on rectangular grids, as each cell needs the states in its four neighbours and in the four cells on its diagonals. We see that equation (3.21a) reduces to a central difference equation in u_3 and u_i when the grid is rectangular, but that equation (3.21b) does not. In the linear multigrid algorithm (section 3.3), this is currently ignored: only the five-point part of the stencil is restricted. But as the convection and the source term are discretised on the five-point stencil only, no convergence problems appear in practice.

A second point is the choice of $\tilde{\nu}_T$ at the cell faces. In the convective fluxes, $\tilde{\nu}_T$ is chosen upwind (equation (3.20)), but using this same viscosity in the diffusion operators causes severe instability in the Gauss-Seidel smoothing. This is caused by the cell faces parallel to the flow, where a small change in velocity, from positive to negative or vice versa, causes a discontinuous change in the face viscosity. Using a central approximation, the average of the two cells next to the cell face, is a possibility, but tests showed that this choice sometimes causes instability too. At this moment, we use the $\tilde{\nu}_T$ from one of the two cells. Which cell, is determined in advance and not changed during the computation. We pick the upwind cell for faces normal to the flow direction and the cell closest to the nearest wall for parallel faces.

Turbulent source term. The source term in (3.2) contains first- and second-order derivatives. Furthermore, being a source term, it cannot be converted to a boundary integral over the cell faces. And computing it with the cell face states is not a good idea: these upwind states can change discontinuously with small changes in the velocity (equation (3.20)). Therefore, the source term is approximated with finite

differences based on the cell centre states. On our curved, non-uniform grids, two adaptations of the standard finite-difference stencils are needed.

First, when the grid is curved, the centres of a cell and its neighbours are not in line and the line between the upper and lower neighbour is not orthogonal to the line between the left and right neighbour. Therefore, we fit a local orthogonal axis system to each cell (figure 3.4) and project the cell centre locations on these axes. Then the derivatives are computed with the cell states in these projected cell centres. This projection step causes some errors, but these are of second-order in the grid size if the grid is sufficiently smooth. The rotation of the local axes has no influence, as the source term is invariant under rotation. Compared with Peyret control volumes this discretisation is simpler, it uses five-point stencils, and it is easier to compute its derivatives with respect to the state for the line smoothing.

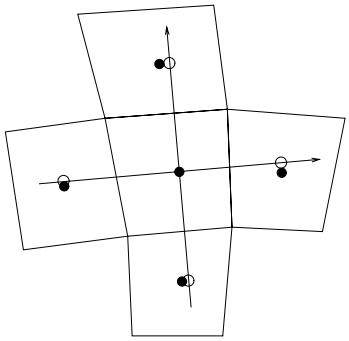


Figure 3.4. Local axes and projected cell centre locations for source term computation.

On the orthogonal axes, finite-difference stencils for non-uniform grids are used. In x-direction, these stencils are:

$$u_x \approx \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}}, \qquad u_{xx} \approx \frac{\frac{u_{i+1}-u_i}{x_{i+1}-x_i} - \frac{u_i-u_{i-1}}{x_i-x_{i-1}}}{\frac{1}{2}(x_{i+1} - x_{i-1})}. \tag{3.22}$$

With these derivatives known in each cell centre, the source terms (3.4) and (3.5) can be computed in the cell centres. The schemes (3.22) are in principle first-order accurate, but if the grids are smooth, the accuracy increases to second order.

Boundary conditions. For u, v, and p, the boundary conditions are the same as those for laminar flow (section 2.2.6); here, we specify a boundary condition for $\tilde{\nu}_T$. The diffusive operator and the source term need a condition at each boundary; for the convective operator, this depends on the characteristics.

Boundary conditions for Menter's model come from section 3.1.2. On an inflow boundary, a Dirichlet condition is specified for $\tilde{\nu}_T$, which is the same for the convection and the diffusion. $\tilde{\nu}_T$ is set to a small positive value, typically 0.01ν. A no-slip wall has Dirichlet conditions too, there $\tilde{\nu}_T$ is zero; at a symmetry wall, $\tilde{\nu}_T$ has a zero normal derivative. Once again, these conditions are the same for convection and

diffusion. Finally, at an outflow boundary, convection requires no boundary condition for $\tilde{\nu}_T$. For the diffusion and the source term, a weak condition is imposed: a homogeneous Neumann condition.

3.5 Turbulence in the two-fluid model

The turbulent flow solution technique given here is applicable for many different types of flows, so it was presented in a general single-fluid formulation first. But in the end, the technique is intended to make accurate solution of water–air flow possible. Therefore, this section shows how the RANS discretisation and the linear multigrid solver are combined with our two-fluid model. We also explain why linear MG is an excellent solver for the two-fluid model itself.

Flow equations. The combined system of turbulent two-fluid flow equations is already given in section 3.1, where the two-fluid RANS equations (3.1) are combined with Menter's model for variable densities (3.2). The coupling between the turbulence and the two-fluid model is weak: the volume fraction equation has no turbulent diffusion and the only effect of α on the turbulence equation is the variable density.

From a physical point of view, the model (3.1) and (3.2) is not yet suitable for modelling the occurrence of turbulence near the free surface. Firstly, physical turbulence near a water surface is often associated with bubbly flow, foam, and wave breaking. The current steady water–air model cannot yet simulate these unsteady phenomena. And second, the Menter model is developed for constant-density flow and is not meant for our mixture surface model. For example, laminar flows exist where the water and the air have different velocities, for example when wind blows over water. For such flows, the numerical mixture layer also becomes a shear layer. Thus, the Menter model would produce turbulence in the mixture layer, which is not desired.

Therefore, we use our current model only for flows where the turbulence occurs away from the interface, for example Cahouet's flow. For these flows, Menter's model is only active in constant-density areas, where it is known to work well. Section 3.8 suggests how the model can be adapted to allow turbulence at the surface.

Like the continuous flow equations, the discretised system is a straightforward combination of the turbulence and two-fluid discretisations.

Linear multigrid. The combined system is solved with the linear multigrid method developed for the turbulence model. This solver was originally chosen because the solutions for the turbulence on the fine and coarse grids are not similar. The same is true for the volume fraction: α is a smeared out discontinuity that is resolved more sharply on fine grids. And, as figure 2.27 shows, waves may be much higher on fine grids than on coarse grids. Therefore, there are locations in a flow with $\mathcal{O}(1)$ differences between fine and coarse grid solutions for α. For these situations, Galerkin coarse grid operators are more effective than nonlinear multigrid.

As stated before, the coarse grid corrections from section 2.3.1 are linear as well. This is caused by the choice (2.49) for w: choosing w low when $\boldsymbol{d}$ is large, effectively eliminates the nonlinear contributions of the coarse grid operators. The

difference with the LMG algorithm is, that the old coarse grid operators are linearised around their respective coarse grid solutions, while the Galerkin operators are all linearised around the present fine grid solution. This advantage of the linear MG technique compensates for the inherent disadvantage of Galerkin operators (section 3.3). Therefore, the linear multigrid is a good solver for the two-fluid flow model.

Damped line smoothing. The turbulence operator may have unstable eigenmodes, that are removed with local damping of the line smoother (section 3.2.2). The determination of the damping factor τ from the convergence rate of Newton-Raphson only works when nothing besides the turbulence model reduces the NR convergence. This is true for single-fluid flows but not for two fluids: in section 2.3.3, we saw that undamped NR may not converge when the initial solution in a line has inflow over all cell faces for some cells.

In practice, this phenomenon is very rare: it may occur in the first two MG steps of a solution, in a few lines only. And as the turbulence regions and the water surface are separate, a single line usually does not have both turbulence and volume fraction convergence problems. Therefore, the equations in each line are solved as follows:

1. First, the equations are solved, while a turbulence damping factor is determined, with the algorithm from section 3.2.2.

2. When $\tau_{\max}$ is reached and NR still does not converge, this is probably caused by α. In those cases, the damping factor τ is copied from the previous line and kept fixed, while the equations are solved with the damped NR from equation (2.50).

This algorithm gives convergence in almost all lines. In the rare cases where it still does not converge, the state in the line is not updated at all. In the next line smoothing step, the line can usually be smoothed again.

3.6 Single-fluid test cases

The following two sections present different test cases. Here, the performance of the multigrid algorithm for the RANS equations is assessed with four single-fluid test cases. Section 3.7 contains a two-fluid flow test. The first test case here is a laminar flow, that is computed with linear and nonlinear multigrid. After that, turbulent boundary layer flows are computed. The last two test cases are more complex flows over airfoils.

3.6.1 Laminar flow To compare the current linear multigrid algorithm with nonlinear multigrid, a laminar flow is computed with both methods. The test case is the flow in a laminar flat-plate boundary layer with zero pressure gradient, the Reynolds number is $Re = 2000$. Our grid has 128×128 cells and is highly stretched in y-direction. Multigrid computations are done on six grids, the coarsest grid has 4×4 cells. For both algorithms, the same Gauss-Seidel smoothing is used on the

Table 3.1. Laminar boundary layer, iterations, and computation time per grid.

Grid	Iterations		t (s)	
	NMG	LMG	NMG	LMG
4×4	4	4	0.1	0.1
8×8	5	5	0.1	0.1
16×16	5	5	0.5	0.4
32×32	5	7	1.5	1.9
64×64	6	9	7.7	9.4
128×128	7	10	37.3	39.7

fine grid and in both cases, the discretisation is the same first-order accurate one. The only difference is in the coarse grid corrections.

Table 3.1 gives a comparison of the results for the FMG solution process. For each grid, from the coarsest to the finest, the table gives the number of iterations needed to reach convergence on that grid (sum of the residuals $< 10^{-6}$). It also lists the computation time needed to get the solution on that particular grid. The computations were performed on a 2.2 GHz PC.

The table shows an adverse effect of the Galerkin operators (see section 3.3): on the finer grids, more iterations are needed for the LMG than for the NMG solution process. The LMG computation times are higher too, but they do not increase as fast as the number of iterations. Therefore, the LMG coarse grid corrections are indeed cheaper than the NMG coarse grid corrections. All in all, this laminar test case shows that the LMG gives similar efficiency as the standard NMG.

3.6.2 Boundary layers The first turbulent test cases are two boundary layer flows over flat plates: simple flows that are dominated by the development of turbulence in the boundary layer. These cases illustrate the efficiency of linear multigrid for turbulent flows and compare the performance on coarse and fine grids.

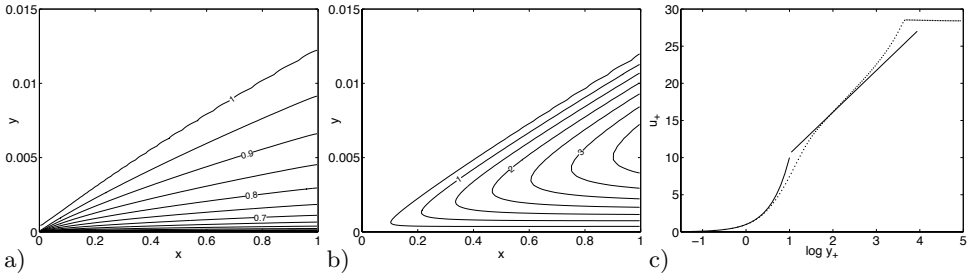


Figure 3.5. Turbulent flat-plate flow at $Re = 10^7$, zero pressure gradient. Isoline plots of velocity $|\boldsymbol{u}|$ (a), turbulent viscosity $\tilde{\nu}_T \cdot 10^5$ (b), and the velocity profile at $x = 1$ in wall coordinates $(\cdots)$ compared with analytical solution (—) (c).

Boundary layer flows. The first flow is a boundary layer with no pressure gradient, at a Reynolds number of 10^7 based on the length of the plate. This test case is used in Menter's paper [46]. The grid is the same as for the laminar boundary layer and multigrid is used with six grids. Results are given in figure 3.5. The first figure gives the velocity profile and the second figure the turbulent viscosity $\tilde{\nu}_T$. We see that this viscosity is (almost) zero in the far field, then increases in the boundary layer and returns to zero at the wall. Compared with the velocity profile, the highest turbulence intensity occurs high in the boundary, where the velocity is close to 1. The last figure gives the velocity profile in wall coordinates; good agreement is found with the theoretical profiles in the viscous sublayer and the logarithmic layer.

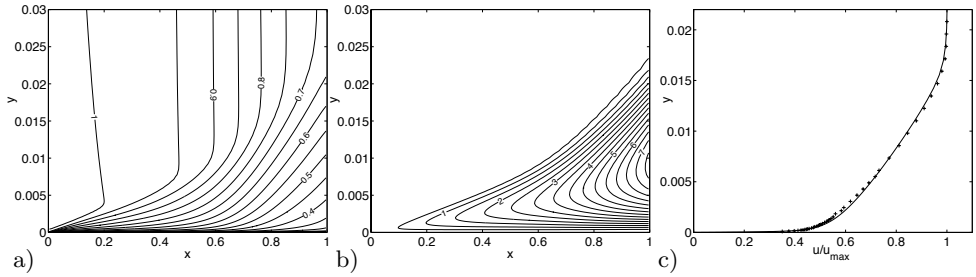


Figure 3.6. Turbulent flat-plate flow at $Re = 7.055 \cdot 10^6$, adverse pressure gradient. Isoline plots of velocity $|\boldsymbol{u}|$ (a), turbulent viscosity $\tilde{\nu}_T \cdot 10^5$ (b), and the velocity profile at $x = 0.82$ (—) compared with measurements (+) from [55] (c).

Similar results are found for a more difficult flow, a boundary layer with an increasingly adverse pressure gradient over the plate. This flow was investigated experimentally by Samuel and Joubert [55], it has a Reynolds number of $7.055 \cdot 10^6$ based on the plate length. The grid is the same as for the previous test case. Results in figure 3.6 show how the adverse pressure gradient slows down the flow. The boundary layer is thicker than in the previous case and the turbulence intensity is higher. A comparison of a velocity profile with experimental measurements shows excellent agreement (figure 3.6c).

Multigrid convergence. Figure 3.7a and 3.7b give the convergence of the residual during the multigrid computation. The finest grid is very fine for a boundary layer grid (128 cells in vertical direction), the flow can be resolved on most coarse grids too. Therefore, we start the FMG computation on the second (8×8) grid. Here, the development of the boundary layer, when the residual rises and falls, is clearly seen. On all subsequent grids, the convergence is excellent. The convergence rate does not deteriorate much on the finer grids, which means that the Galerkin operators cause few problems. In fact, convergence on the fine grids is faster than in the preceding laminar case. Also, the convergence is similar for both boundary layers; multigrid performance does not get worse in adverse-pressure conditions.

For comparison, figure 3.7c shows the convergence when the zero pressure-gradient flow is solved with line Gauss-Seidel alone, on a single grid. This takes 120 iterations on the fine grid, compared to 7 for the multigrid solution. 40 iterations are needed

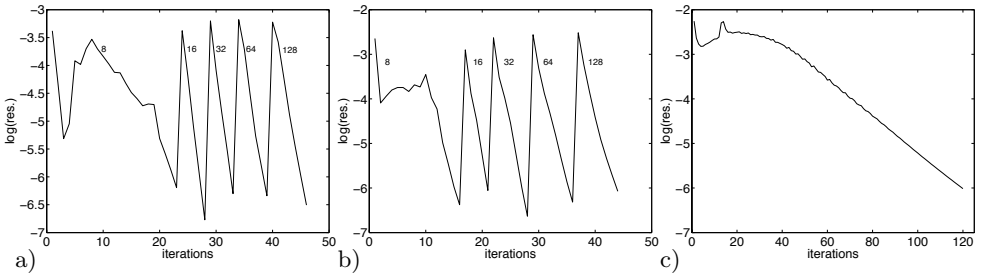


Figure 3.7. FMG convergence for boundary layer flows under zero (a) and adverse (b) pressure gradients. Graph (c) gives single-grid convergence for the same case as (a).

for the development of the boundary layer. The total CPU time is about 5 times higher than for the multigrid solution.

3.6.3 NACA 0012 airfoil A more complex test case than the flat-plate boundary layer is the flow over a NACA 0012 airfoil. It features stagnation points, curved walls, and the transition from a boundary layer to a turbulent wake. The angle of attack $\alpha = 0$ and the Reynolds number is $Re = 10^6$. Because the NACA 0012 airfoil is symmetric, the flow is computed in the upper half of the flow domain only. An H-type grid is used with 512 cells in x-direction and 256 cells in y-direction, the grid is stretched near the airfoil surface and near the leading and trailing edges. The problem is used to test multigrid convergence for a more difficult flow.

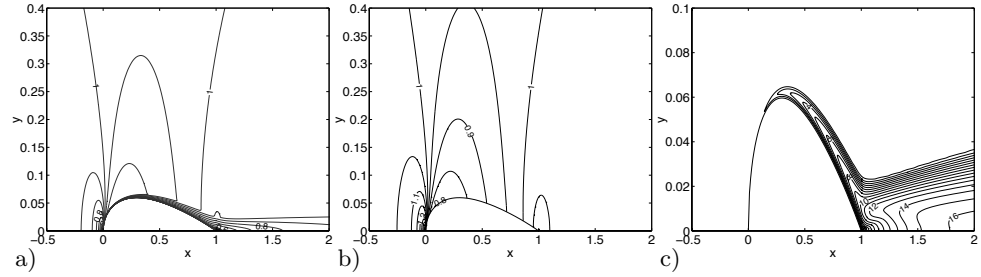


Figure 3.8. Flow over a NACA 0012 airfoil. These plots show the velocity $|\boldsymbol{u}|$ (a), the pressure p (b), and the turbulent viscosity $\tilde{\nu}_T \cdot 10^5$ (c). For clarity, all plots are stretched in y-direction.

Flow field. The flow around the airfoil is given in figure 3.8. The velocity plot 3.8a shows the stagnation point at the leading edge, the suction area above the airfoil, the growing boundary layer and the transition from the boundary layer to the wake. The second figure shows the pressure and the last figure the turbulent viscosity. We see here that the turbulence intensity is very low near the leading edge, it starts to grow where the pressure gradient becomes adverse. Behind the airfoil, the location of the maximum value for the turbulence gradually shifts from the centre of the

boundary layer to the centre of the wake.

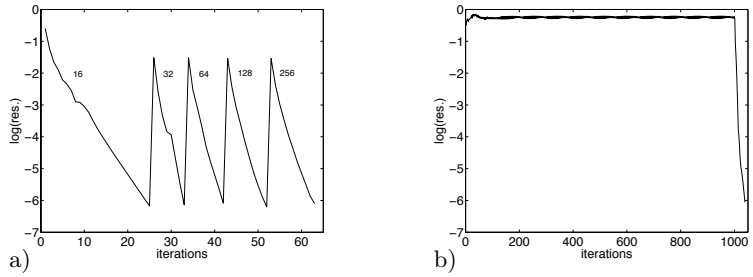


Figure 3.9. NACA 0012, convergence of the residual on the 256×512 grid, for multigrid (a) and single-grid Gauss-Seidel (b).

Multigrid convergence. The flow is solved with multigrid on six grids; the coarsest grid has 8×16 cells. As in the boundary layer case, the FMG computation is started on the second grid. But the NACA 0012 flow has more different features than the simple boundary layer flows, so it cannot be resolved accurately on the 16×32 cell grid. Therefore, multigrid does not work on this grid. To retain the full advantage of the FMG algorithm, the flow is solved with single-grid smoothing on the 16×32 grid. From the 32×64 grid on, multigrid is used.

The convergence of the residual is shown in figure 3.9a. The number of iterations per grid is a little higher than for the preceding boundary layer flows, but the initial residual on each grid is higher too (because the flow is more complex), so that is to be expected. Convergence does not deteriorate much on the finer grids, which shows once again that the Galerkin operators work correctly. There is a little 'bump' in the convergence on the 32×64 grid, probably caused by the development of the boundary layer that is not sufficiently resolved on the 16×32 grid.

For comparison, the single-grid convergence is shown in figure 3.9b. This convergence is odd: the residual stays constant for a long time and then suddenly decreases. This behaviour is abnormal for these types of flows, the usual single-grid convergence plots look like figure 3.7c. But it does show that there are cases where the FMG solver can compute flows that cause problems for single-grid smoothing.

3.6.4 Supercritical airfoil The low-Mach flow over a supercritical airfoil was measured by Nakayama [49]. This airfoil is placed at an angle of attack $\alpha = 4^{\circ}$, the Reynolds number is $1.2 \cdot 10^6$. Computation of this flow is very challenging, as the flow field has high curvature and strong pressure gradients near the trailing edge. In the same location, two boundary layers of different strength merge to form the turbulent wake. The flow is computed on a 256×256 cell H-type grid.

Flow field. The most typical feature of the Nakayama wing is the concave lower side near the trailing edge. The strong curvature in the flow field there can be seen in the results (figure 3.10). The plots show the stagnation point and a strong suction peak near the leading edge, and a moderate high pressure region in the hollow below the

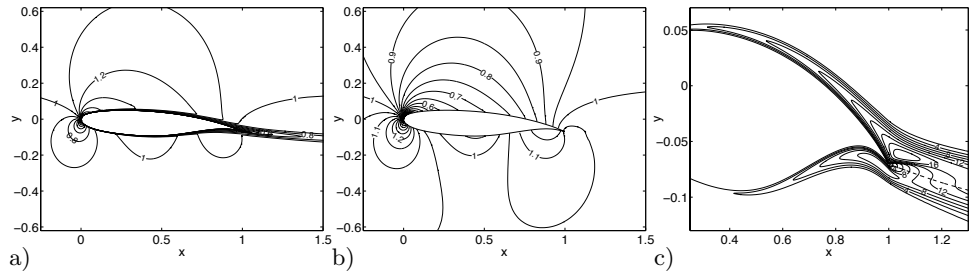


Figure 3.10. Flow over a supercritical airfoil; velocity $|\boldsymbol{u}|$ (a), pressure p (b), and turbulent viscosity $\tilde{\nu}_T \cdot 10^5$ (c). Plot (c) is stretched in y-direction, - - -: trailing edge streamline.

trailing edge. The turbulence intensity near the trailing edge is interesting: because of the adverse pressure gradient above the airfoil, the turbulence intensity is higher above than below the airfoil. The two boundary layers merge, so the turbulence levels have to adapt. For the upper boundary layer, the reduction in turbulence intensity happens in a thin layer, clearly visible in figure 3.10c. This layer is *not* aligned with the flow, it runs upwards into the flow.

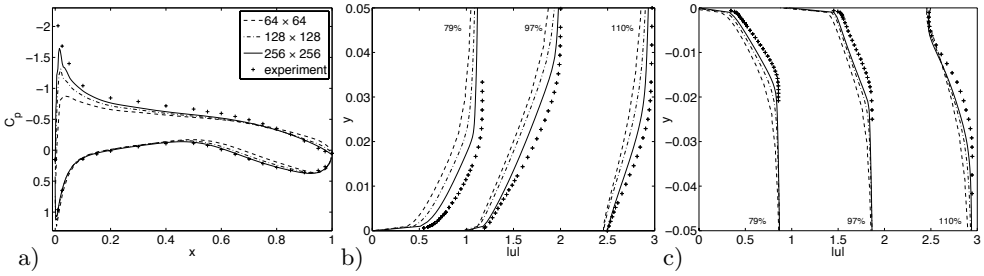


Figure 3.11. Supercritical airfoil, grid convergence study. Solutions for c_p (a) and for the velocity profiles above (b) and below the airfoil (c) are compared with measurements (+) from [49].

Grid convergence, comparison with experiment. A grid convergence study is done for the supercritical airfoil. Solutions on three grids are compared with Nakayama's experimental data [49]. Figure 3.11 shows the results: velocity profiles in three chordwise positions and the pressure coefficient $c_p = (p - p_\infty)/(\frac{1}{2}u_\infty^2)$, where the reference pressure $p_\infty = 1$ and the reference velocity $u_\infty = 1$ too. The c_p plot agrees reasonably well with the experiment, although the pressure in the stagnation point is a little too high and the suction peak is not strong enough. The same is seen in the velocity contours: the solution approaches the experiment, but it is not yet converged. A very accurate solution with the current first-order discretisation would require finer grids.

Multigrid. The Nakayama flow is solved with multigrid on 6 levels, the coarsest

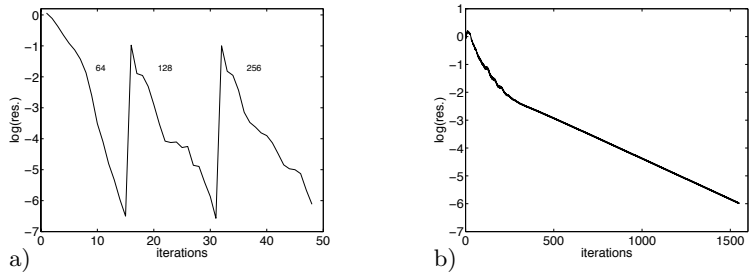


Figure 3.12. Supercritical airfoil, convergence of the residual on the 256×256 grid, for multigrid (a) and single-grid Gauss-Seidel.

level is 8×8 cells. Due to the difficulty of the flow, the coarsest grid on which a solution could be obtained is the fourth grid of 64×64 cells. There, 6 iterations are done with single-grid Gauss-Seidel to develop the boundary layers, then multigrid is started; this can be seen in the multigrid convergence graph 3.12a. The solution on the fifth and sixth grid is found with multigrid from the start.

The multigrid convergence on the two finest grids is slower than in the previous problems and it is a bit 'jittery'. There are two possible causes for this. First, the airfoil has both an upper and a lower side; to get good smoothing, it is necessary to start the horizontal line smoothing alternately from the lowest line going up and from the highest line going down. This sometimes causes large residuals near the outer boundaries of the domain, that have nothing to do with the turbulence modelling: they appear in laminar flow too. This may be due to an error in the code.

And second, to get stability near the trailing edge, a small region there (a circle with radius 0.01) needs a higher damping in the coarse grid corrections. In that circle, the τ for the coarse grid corrections is set at a constant high value. This stabilises the computation, but it decreases the convergence a little. This problem can probably be solved by performing coarse grid corrections with the full nine-point stencils instead of the five-point stencils, as diffusion is dominant in the high gradients near the trailing edge. This is an area for further study (see section 3.8).

At this moment, the multigrid convergence is acceptable. Compared with a single-grid solution (figure 3.12b), the computation time is reduced by a factor of about 11.5.

3.6.5 Parameter settings The linear multigrid algorithm from sections 3.2.2 and 3.3 has some free parameters, that can be tuned if desired. However, for our numerical experiments, we found that this was not necessary for most parameters: the values given in section 3.2 are satisfactory for all problems. The parameter ϵ, the required relative convergence for the line smoothing residual, has the strongest influence. A lower ϵ means, that a higher convergence rate is required for NR in the lines, which leads to higher values for the damping τ. This means better stability, but slower convergence for the multigrid method. Still, $\epsilon = 10^{-5}$ is used for all our problems, with good results.

The user's most important choice is, on which grid the FMG solution is started.

For a linear coarse grid correction we can use as many grids as desired, but FMG requires nonlinear solutions on coarse grids. The problems above show, that the coarsest possible starting grid depends on the problem: for more complex flows, we need finer starting grids. We found, as a good rule of thumb, that the starting grid must not have less than about 8 – 10 cells over the thickness of the expected boundary layers.

3.7 Two-fluid test

In this section, we return to the Cahouet test problem [6] from section 2.5. The tests in the previous section 3.6 have confirmed that the turbulence model works well. It is now used as a tool to eliminate the laminar separation that occurred in section 2.5, to answer the question that remained from that section: can our water–air model accurately reproduce the short, steep waves that Cahouet's experiment suggest?

Table 3.2. Numerical tests.

nr.	Fr	Re_H	H	g	μ
1	0.43	$1.3 \cdot 10^5$	1.00	5.41	$7.7 \cdot 10^{-6}$
2	0.52	$2.4 \cdot 10^5$	1.33	2.78	$5.5 \cdot 10^{-6}$
3	2.05	$1.9 \cdot 10^5$	0.46	0.52	$2.4 \cdot 10^{-6}$

The test cases are the same three as in the laminar test, but now with the correct viscosity (table 3.2). The grids are stretched near the bottom and are also refined near the water surface (figure 3.13). Due to these changes, the grids now have the same number of cells in horizontal and vertical direction. For the subcritical tests, horizontal stretching at the inflow and outflow boundaries is used to damp out waves and allow steady solutions, but this is not enough. The laminar viscosity is so low for these tests, that it has to be increased by a factor 100 when $x < -2$ or $x > 8$; only this higher viscosity gives enough damping.

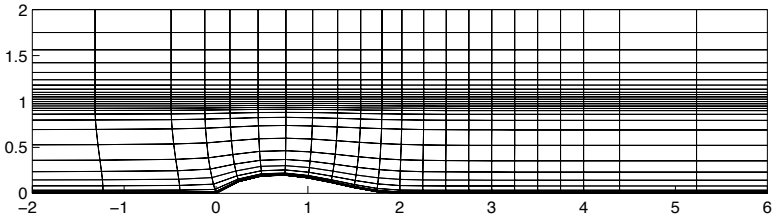


Figure 3.13. Grid for Cahouet test, 32×32 cells, with stretched cells at the inflow and outflow boundaries. The figure shows a detail, the stretched zones run from $x = -72$ to $x = 76$.

3.7.1 Flow results This section focuses on the effectiveness of the flow model. Multigrid performance is studied in the next section.

Case 1, $Fr = 0.43$. This subcritical test case is first computed with a bottom bump that is thinner than in Cahouet's experiment: $E = 0.075L$ instead of $E = 0.10L$. The same test case is computed by Van Brummelen [5]. Figure 3.14 gives the solution. The velocity profiles look good: they clearly show upward–downward motion in the waves and areas of low velocity on the wave crests. The waves are notably shorter than in the laminar case (figure 2.26).

Figures 3.14d and 3.14e show, that the turbulence model is successful. $\tilde{\nu}_T$ has a normal development for a boundary layer starting at $x = -2$ (the start of the no-slip condition, see table 2.3) and the streamline plot shows a fully attached flow. Even for the current high Reynolds number, the turbulence model prevents the appearance of a separation bubble.

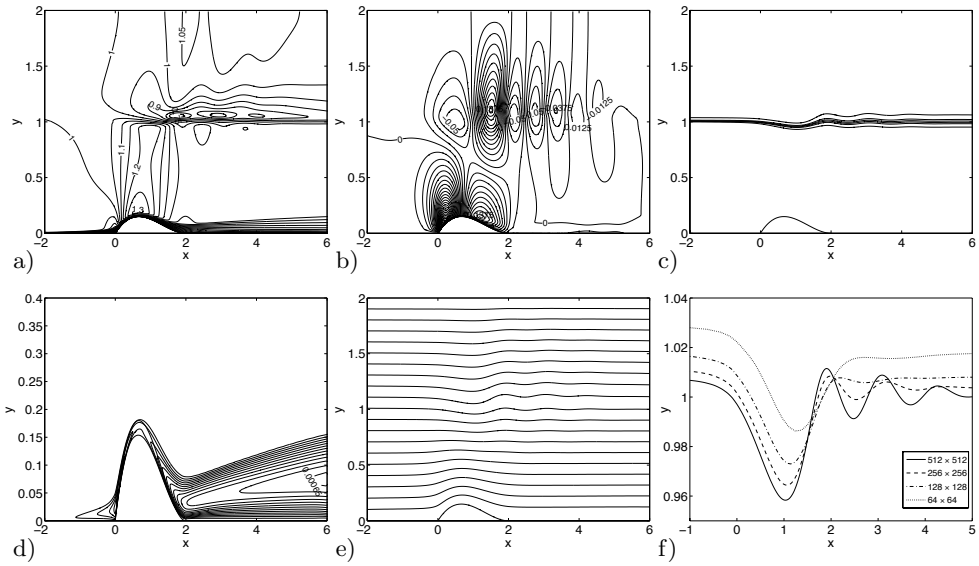


Figure 3.14. $Fr = 0.43$ flow, with thin bottom bump ($E = 0.075L$). Isoline plots of speed $|\boldsymbol{u}|$ (a), vertical velocity v (b), volume fraction α (c), and turbulent viscosity $\tilde{\nu}_T$ (d); streamline plot (e); all on 512×512 grid. Grid convergence of $\alpha = 0.5$ isoline (f).

The grid convergence study (figure 3.14f) shows the shorter waves. Despite the different dimensions, the grids have the same number of cells as those in figure 2.27a. The figure shows the expected first-order behaviour: the waves are damped out quickly and the difference between the coarse and fine grid solutions is high. But, compared with figure 2.27a, the waves are resolved notably better on the coarse grids. The refined cells near the water surface are one explanation; the absence of laminar separation also improves the wave formation.

The reason why we did not start with the normal Cahouet bump is, that we did not find steady solutions for this case on grids like in figure 3.13. The cause is revealed in figure 3.15, which shows an intermediate result for the solution of the flow over a bump with $E = 0.09L$. We see that the upper part of the mixture layer breaks;

therefore, the solution process does not converge.

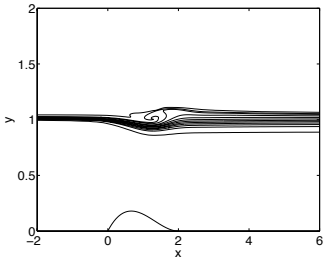


Figure 3.15. Partial breaking of the water surface. Isolines for α in an intermediate solution for the $Fr = 0.43$ case with $E = 0.09L$, 128×128 grid.

The breaking of a wave depends on the slope of the wave crest: above a certain steepness, the wave starts breaking (see e.g. [13, 65]). The waves in Cahouet's experiment did not break. For our water–air mixture model, the thickness of the mixture layer increases when going downstream; this is caused by the numerical diffusion. Because of this spreading, the upper part of the mixture layer near a crest is steeper than the centre. When the grid is not fine enough, the layer may smear so much that the upper part reaches the required steepness for breaking, even though the exact solution has non-breaking waves. This happens in figure 3.15. When the grid is sufficiently refined, a non-breaking solution is found eventually. But for the $Fr = 0.43$ flow with Cahouet's original bump, our 512×512 grid is not fine enough for this.

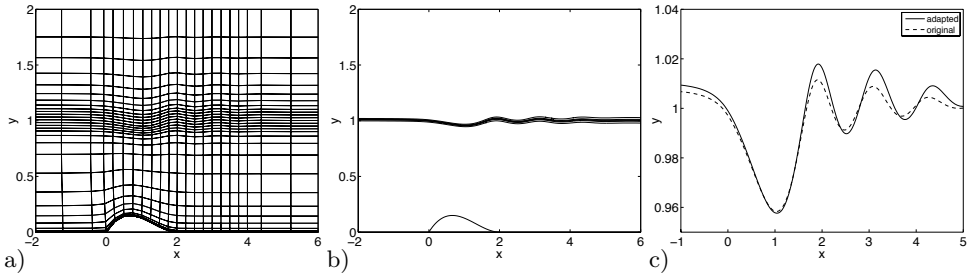


Figure 3.16. $Fr = 0.43$ flow with thin bump, computed on an adapted 512×512 grid. The flow-aligned grid (a), isolines for α (b), and a comparison of the $\alpha = 0.5$ isoline with the solution on the original grid.

To prove that our two-fluid method is effective, even for the $E = 0.10L$ bump, we eliminate some of the smearing of the interface by using a flow-aligned grid. Figure 3.16a shows such a grid for the thin bump, where the shape of the fine cells near the surface is set from the solution in figure 3.14c. On this grid, the flow at the interface is parallel to the grid lines: this reduces the smearing of the interface significantly (figure 3.16b). It also increases the accuracy of the solution: figure 3.16c shows that the reduced diffusion near the surface increases the wave amplitude.

The solution for the thick bump is found with a continuation process. Using the water surface from figure 3.16b, a grid is made with a thicker bump. A solution is then computed on this grid and the water surface for this solution is used to make a grid with an even thicker bump, etc. Thus, $E = 0.10L$ is reached in four steps. The computation time is reduced by starting the solution on each fine grid from the solution on the previous fine grid, instead of using the FMG procedure.

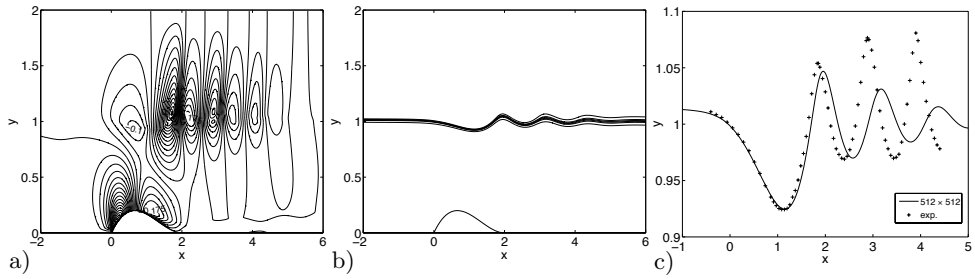


Figure 3.17. $Fr = 0.43$ flow with $E = 0.10L$ bump, flow-aligned 512×512 grid. Isolines of v (a) and α (b), comparison of the $\alpha = 0.5$ isoline with Cahouet's measurements.

Figure 3.17 gives the solution. The wave train is well developed and the surface is smeared just a little. The wave amplitude is much higher than for the thin bump, even though the bump thickness has increased by only 25%. The comparison with Cahouet's experiment shows that the first-order method damps the waves too much. On the other hand, the trough above the bump and the first wave are resolved well and the wavelength is predicted much better than in figure 2.27b. Altogether, the figure shows that the combination of our two-fluid model with turbulence is sensible and gives physically correct solution behaviour.

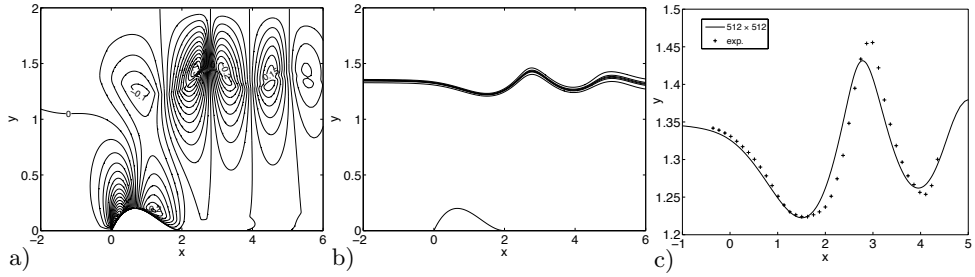


Figure 3.18. $Fr = 0.52$ flow with $E = 0.10L$ bump, flow-aligned 512×512 grid. Isolines of v (a) and α (b), comparison of the $\alpha = 0.5$ isoline with Cahouet's measurements.

Case 2, $Fr = 0.52$. The second subcritical test case is considered more difficult than the first one by Cahouet, because the first wave is steeper and closer to breaking. But the solution with our model is actually easier than the previous case, because the longer wave is resolved on more cells. The solution in figure 3.18 is obtained from an $E = 0.075L$ solution in only two continuation steps. The figure shows that

the wave is indeed higher and steeper than in the previous case. The agreement with the experiment, both in wavelength and in amplitude, is good.

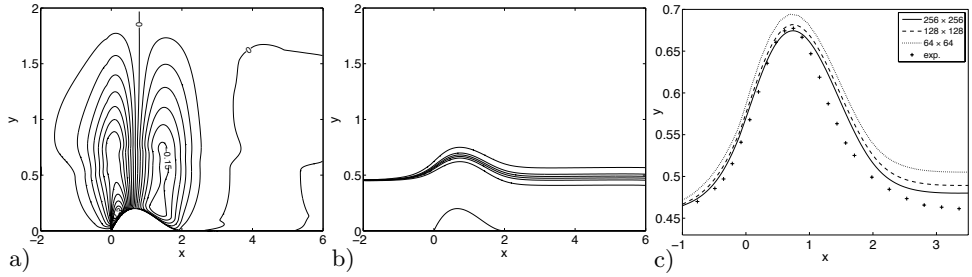


Figure 3.19. $Fr = 2.05$ supercritical flow. Isolines of v (a) and α (b) on a 512×512 grid and comparison of the $\alpha = 0.5$ isolines with Cahouet's measurements.

Case 3, $Fr = 2.05$. The supercritical test case is computed on grids that are not stretched at the inflow and outflow boundary. The wave is far away from breaking, so normal FMG is used for the solution on non-adapted grids. The test case is modelled slightly different from the laminar case in section 2.5.1: now the flow over the bottom is modelled with a turbulent boundary layer, starting at $x = -2$. This choice is also an approximation: the experimental boundary layer is more developed when it reaches the bottom bump. In all our tests, the boundary layer is similar to the one in figure 3.14d. Also, the prescribed pressure at the outflow boundary is kept constant (compare with section 2.5.1).

Despite these changes, the flow (figure 3.19) is similar to the laminar case in figure 2.31. The smearing of the interface is about the same, and the solution shows good first-order grid convergence (figure 3.19c). Due to the real boundary layer used now, the wave height agrees better with the experiment than the laminar wave (figure 2.32b).

3.7.2 Multigrid convergence The efficiency of the MG solver is studied for the two test cases where FMG solution is possible: the $Fr = 0.43$ case with the thin bump and the $Fr = 2.05$ case. Both are solved on 256×256 grids, that have the same number of cells as the 128×512 grids in the laminar test (section 2.5.3). Multigrid is used with 8 grids, FMG is started on the third grid.

There are many differences between these computations and the laminar ones. The first is, obviously, the inclusion of the turbulence model, with its extra flow equation and source term. Second, we switch to Galerkin coarse-grid operators, that could make the convergence worse (section 3.6.1), but may also be an improvement over the nonlinear coarse grid corrections because they resemble the fine grid operator better near the water surface. Third: the grids are different: we now use grids with more cells in y-direction and refinement near the water surface and the boundary layer. And finally, the turbulence model has eliminated the separation bubble. All these changes have an effect on the multigrid convergence.

The FMG convergence for both test cases (figure 3.20) shows the effect of the

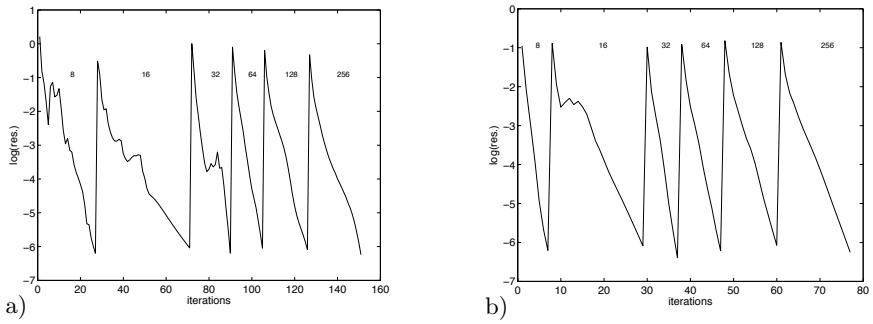


Figure 3.20. Total residual during the multigrid solution for the $Fr = 0.43$ (a) and $Fr = 2.05$ (b) test case.

Table 3.3. Average convergence rates for the $Fr = 0.43$ and $Fr = 2.05$ test case.

case	8	16	32	64	128	256
$Fr = 0.43$	0.572	0.746	0.453	0.381	0.507	0.566
$Fr = 2.05$	0.139	0.566	0.172	0.262	0.367	0.458

turbulence model mostly on the coarser grids. The 'bumpy' convergence on these grids is similar to the convergence for the single-fluid tests on coarse grids (section 3.6). On the finer grids, the convergence is less linear than for the laminar flow (figures 2.33a and 2.35b), but not slower. Comparing table 3.3 with table 2.5 shows that the turbulent computation actually converges faster on the three finest grids. So for two-fluid flows, the better similarity with the fine-grid operator near the water surface outweighs the slower convergence speed of the Galerkin operators. Also, the reduction in convergence rate in the last iterations of figure 2.33a, caused by the separation, is absent from figure 3.20a. The same is true for figures 2.35b and 3.20b; here, the improvement may be caused by the no-slip bottom wall, or the outflow pressure that is now kept constant.

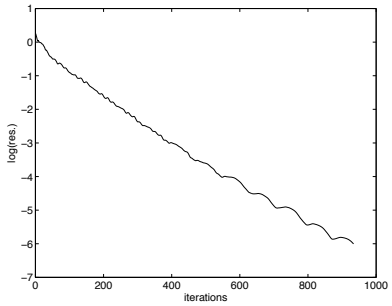


Figure 3.21. Residual for the $Fr = 0.43$ test, solved with single-grid line smoothing.

Table 3.4. CPU times.

	$Fr = 0.43$		$Fr = 2.05$	
Grid	levels	time (s)	levels	time (s)
64×64	6	37.5	5	35.7
128×128	7	187	6	131
256×256	8	884	8	616
256×256 (V)	8	not conv.	7	845
256×256 (VW)	8	1194	8	744
256×256	1	16058	1	not conv.

Table 3.4 shows computation times on a 2.2 GHz PC. These are higher than for the laminar case (table 2.6) but lower than expected, given the increase in work from solving five instead of four flow equations. The fast convergence on fine grids helps reduce the computation times. Once again, the W-cycle is the most efficient for both problems.

The $Fr = 0.43$ solution is compared with a line smoothing computation on a single grid. This computation converges faster than its laminar counterpart (compare figures 3.21 and 2.33), because the line smoother is most effective when the number of horizontal and vertical cells is equal. Still, multigrid reduces the computation time with a factor of almost 20. So also for the combined turbulent two-fluid system, multigrid proves to be an excellent solver.

3.8 Conclusion

We have presented a novel multigrid method for the Reynolds-Averaged Navier-Stokes equations with Menter's turbulence model. The method has two new aspects: a linear coarse grid correction with nonlinear smoothing on the finest grid and an efficient locally damped line Gauss-Seidel smoothing.

Standard nonlinear multigrid does not work for the RANS equations. This has two reasons. The first is, that the combination of the non-constant viscosity and the source term in the turbulence equation makes it possible for the system to have unstable eigenmodes, that make Gauss-Seidel smoothing impossible. However, in the neighbourhood of a converged solution, these eigenmodes become stable. Our line Gauss-Seidel smoothing is stabilised by adding local damping to the turbulence equation in each smoothing step, enough to remove the unstable eigenmodes. We showed that the amount of damping necessary can be estimated with the Newton-Raphson root finder in the individual lines: if NR converges quickly, then enough damping is used.

The second problem is, that the nonlinear RANS operators on coarse grids do not resemble the operators on fine grids sufficiently well. This problem is caused by the sensitivity of the turbulence source terms to the grid size on too coarse grids. The problem is solved by switching to a linearised coarse grid correction with Galerkin operators. These resemble the fine grid operator reasonably well and their

construction is simple for a finite-volume discretisation. A disadvantage is the higher memory requirement. The combination with nonlinear smoothing on the finest grid adds robustness and allows the computation of the Gauss-Seidel damping factor, that is used to stabilise the smoothing on the coarse grids too.

The multigrid method is combined with a cell-centred finite-volume discretisation. Convective fluxes are constructed with artificial compressibility, diffusive fluxes with central differences. The source terms are computed with finite differences on local orthogonal grids.

Numerical results show that the multigrid convergence is very good. The convergence rates for a laminar problem are compared with nonlinear multigrid: they are a little lower due to the use of Galerkin operators, but the effect on the computation times is small, because the linear method requires less time per multigrid cycle. Turbulent flow problems show comparable convergence. Full multigrid is needed for good convergence, because turbulent boundary layers have to develop first when the initial condition is a uniform flow. In this initial stage, multigrid is of little use. By using full multigrid, the solutions on all but the coarsest grid start with developed boundary layers. Thus, the extra iterations for boundary layer development are only needed on the coarsest grid.

When combined with our water-air model, the turbulence model provides physically correct solutions. Tests show that the laminar separation, found in the previous chapter, disappears with the introduction of turbulence. While, in some cases, grid deformation is needed to get convergence, the resulting solutions show good agreement with experiments.

The water surface is a smeared discontinuity. Therefore, the solutions for the volume fraction on coarse and fine grids are different, just like for the turbulence. So linear multigrid with Galerkin coarse grid operators is not only effective for turbulence, it is also an excellent solver for the two-fluid model. Despite the addition of the turbulence equations, the multigrid convergence rate for two-fluid tests is found to be higher than for the laminar tests in the previous chapter. Thus, the combination of the two-fluid model with turbulence and multigrid is successful.

Outlook. The present method works well. However, there is room for improvements to make the method even more robust and accurate. For instance, a larger study of the parameter settings for the multigrid algorithm can lead to better guidelines for the choice of the parameters. Also, it is worthwhile to study aspects of the discretisation, like the choice of the turbulent viscosity at the cell faces that influences the smoothing efficiency. And finally, it is useful to compare coarse-grid corrections using nine-point stencils with the present method, to see if there are any differences.

The present method can probably be extended to other turbulence models and even to different equations than RANS. The current analysis is done for Menter's turbulence model, but most one- and two-equation turbulence models consist of the same type of equations: convection-diffusion equations with similar production and dissipation terms. Therefore, we expect that the present method can be used

for other RANS turbulence models. And in other fields, for example in chemistry, equations with steady solutions and dominant source terms appear too. Our method can be a useful tool for the fast solution of these equations.

The current water–air model can be extended to allow the interaction of turbulence with the free surface. For fitting methods and capturing with reconstruction (section 1.2), the free surface is a boundary of the flow, so a turbulence model can be used there like at any other boundary. Probably, the turbulence model in our discretisation can be modified such, that it does not really notice the free surface either, by switching it off altogether in the mixture layer or by somehow removing the density dependence. For flows where the turbulence and the water surface mix but do not interact strongly (such as the flow at the ship's hull), this likely gives accurate results.

A more daring solution is to make the steady model suitable for unsteady flow at the interface. The present model does not converge when the surface is close to breaking. But it is likely that a technique like for the RANS equations can produce averaged flow equations that have steady solutions, even for breaking waves. The solution for α then becomes an average volume fraction, that is smeared in breaking regions. The major challenge for such a technique is the derivation of a 'turbulence model' for the free surface breaking. A model like this even prevents problems like with the $Fr = 0.43$ test case, where FMG failed because of breaking on coarse grids even when a non-breaking solution on a fine grid exists. Thus, such a model would be a very flexibe and robust tool for water wave simulation.

4 Second-order accuracy

To get good flow solutions without using very fine grids, the two-fluid discretisation is improved to second-order accuracy. This requires two changes to the method. First, the flux discretisation itself is changed from first- to second-order accuracy. And second, the solution method is adapted. This chapter describes both changes.

For most of the fluxes, either central differences or a TVD limited reconstruction are used. Section 4.1 briefly explains these techniques.

Then section 4.2 describes the special discretisation used for the volume fraction. In many existing volume-of-fluid methods without interface reconstruction, for time-dependent flows, compressive schemes are used [16, 27, 64]. These schemes use numerical antidiffusion to keep the smeared water surface thin. This is possible because the solution for α is a discontinuity only, so it has no smooth gradients which the antidiffusion would distort. Here, a compressive scheme is developed especially for steady flow: this allows a much simpler scheme than the existing unsteady flow methods.

The second-order accurate discretisation is solved with defect correction [9, 63]; the robust first-order multigrid scheme is kept as the heart of the solver. Section 4.3 gives a preliminary investigation of defect correction for our two-fluid model. The resulting solver is not yet robust enough in all cases. However, the results in section 4.4 show, that the combination of defect correction and the limited flux discretisation gives very accurate solutions, in short computation times.

4.1 Second-order accurate fluxes

To get a stable second-order accurate discretisation of the fluxes in the Navier-Stokes equations, we use a TVD limited reconstruction for the convective fluxes. The Riemann solver from section 2.2.2 is not changed, but the input left and right states $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ are now determined with linear interpolation over four cells (figure 4.1). The solution is kept stable and monotone with a limited interpolation following Sweby [61]. For example, the left and right state on a vertical cell face are:

$$\begin{aligned} q_0^p &= q_{i,j}^p + \phi(r_0^p)\frac{q_{i,j}^p - q_{i-1,j}^p}{x_{i,j} - x_{i-1,j}}\left(x_{i+1/2,j} - x_{i,j}\right), & r_0^p &= \frac{\frac{q_{i+1,j}^p - q_{i,j}^p}{x_{i+1,j} - x_{i,j}}}{\frac{q_{i,j}^p - q_{i-1,j}^p}{x_{i,j} - x_{i-1,j}}}, \\ q_1^p &= q_{i+1,j}^p + \phi(r_1^p)\frac{q_{i+1,j}^p - q_{i+2,j}^p}{x_{i+1,j} - x_{i+2,j}}\left(x_{i+1/2,j} - x_{i+1,j}\right), & r_1^p &= \frac{\frac{q_{i,j}^p - q_{i+1,j}^p}{x_{i,j} - x_{i+1,j}}}{\frac{q_{i+1,j}^p - q_{i+2,j}^p}{x_{i+1,j} - x_{i+2,j}}}. \end{aligned} \tag{4.1}$$

The limiting is done individually for each state component q^p, $p = 1 \cdots 5$. In these equations, the non-uniformity of the grids is taken into account; this is necessary for the highly compressed grids in the boundary layers. On the other hand, the curvature of the cells is ignored which is acceptable for smooth grids.

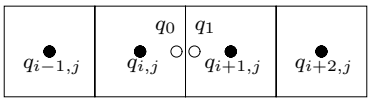


Figure 4.1. Reconstruction of cell face states from four cells.

The scheme is stable and monotone when the limiter $\phi(r)$ lies in Sweby's monotonicity domain (figure 4.2a) and it is second-order accurate when $\phi(1) = 1$. We use the limiter proposed by Koren [32], where Van Leer's $\kappa = \frac{1}{3}$ scheme [36] is followed as long as possible (figure 4.2b):

$$\phi_\kappa = \begin{cases} 0 & r \leq 0, \\ 2r & 0 < r \leq \frac{1}{4}, \\ \frac{2}{3}r + \frac{1}{3} & \frac{1}{4} < r \leq \frac{5}{2}, \\ 2 & r > \frac{5}{2}. \end{cases} \tag{4.2}$$

In one dimension, this limiter is third-order accurate. For 2D finite-volume discretisations, the formal order of accuracy is two, but the resulting scheme is still one of the most accurate limited schemes possible.

The limited linear reconstruction automatically takes into account the pressure gradient due to the gravity. Therefore, the gravity correction in the reconstruction, as introduced in equation (2.33), is no longer used.

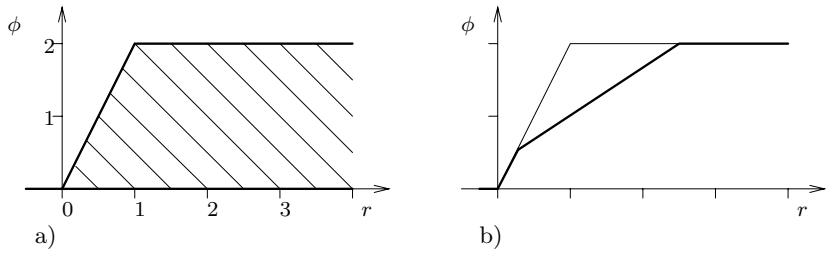


Figure 4.2. Sweby's monotonicity domain (a) and the $\kappa = \frac{1}{3}$ limiter (b).

For the diffusive fluxes and the turbulent source term, the discretisations are the same as for the first-order scheme. Formally, the discretisations (2.32) and (3.22) are first-order accurate on arbitrary meshes. But on smooth meshes, i.e. meshes for which the difference in dimensions of neighbour cells is of $\mathcal{O}(h^2)$, the schemes become second-order accurate. The only difference between the first- and second-order accurate discretisations is, that the cell face turbulent viscosities in the diffusion terms are now modelled with central interpolations (compare with section 3.4).

4.2 Compressive limiter for α

For the reconstruction of the volume fraction α, there exist better limiters than the standard second-order ones like (4.2). These compressive limiters keep the wa-

ter surface sharp with numerical antidiffusion. One such limiter, that is specially adapted for steady flow, is derived and tested in this section.

For the volume fraction solutions, second-order schemes do not give low errors. These solutions consist of a discontinuity only; $\alpha = 1$ below the water surface and $\alpha = 0$ above it. At the surface itself, the exact solution for α is discontinuous and its spatial derivatives are not defined. So in the numerical solutions, where the interface is smeared over a few cells, the spatial derivatives do not converge on grid refinement but continue to increase. Therefore, and since all terms in the truncation error of a scheme contain spatial derivatives of the solution, the higher-order terms in the truncation error do not necessarily become smaller than the low-order terms on grid convergence. And therefore, a scheme that eliminates the first-order truncation terms does not guarantee a lower truncation error than other schemes that have first-order terms. So for an α-discretisation, it is meaningless to require a second-order truncation error.

The main errors, both in the first-order scheme and in limited schemes, come from numerical diffusion. Therefore, the most important error in the α-solution is the smearing of the interface. This smearing can be reduced with a scheme that contains numerical antidiffusion: a compressive scheme. In the monotonicity domain, such a scheme must still lie within the boundaries of the domain; this guarantees stability, even for discontinuous solutions. But the $\phi(1) = 1$ requirement, that gives a second-order truncation error, no longer holds.

Here, section 4.2.1 introduces compressive schemes that are in use for time-dependent computations. Then section 4.2.2 shows how these schemes can be simplified by using the special properties of steady flow. The last two sections 4.2.3 and 4.2.4 test our discretisation on a linear problem.

4.2.1 Existing schemes for time stepping This section introduces compressive schemes and shows some of the schemes that are currently used for VoF discretisations of time-dependent flows.

The most basic stable compressive scheme is the limited downwind scheme (figure 4.3), whose limiter follows the top of the monotonicity domain:

$$\phi_{\mathrm{LD}} = \begin{cases} 0 & r \leq 0, \\ 2r & 0 < r \leq 1, \\ 2 & r > 1. \end{cases} \tag{4.3}$$

The first part of this limiter is the downwind scheme $\phi = 2r$, i.e. $\alpha_0 = \alpha_{i+1,j}$ in the notation of figure 4.1[1]. The second part, $\phi = 2$ or $\alpha_0 = 2\alpha_{i,j} - \alpha_{i-1,j}$, is similar to the downwind scheme: it extrapolates a value from the upwind cells to the downwind cell centre and uses that value for α_0.

These two downwind schemes are compressive, they sharpen any gradient into a step function. For example, consider a smooth solution for α and let $\alpha_{i+1,j}$ be larger than $\alpha_{i,j}$. The correct α_0 lies somewhere between $\alpha_{i,j}$ and $\alpha_{i+1,j}$, so taking

[1] Also $\alpha_1 = \alpha_{i,j}$, but for brevity, we discuss α_0 only. Since the Riemann solver takes pure upwind values for α, the face value $\alpha_{\frac{1}{2}}$ is equal to α_0 anyway when $u_{\frac{1}{2}} \geq 0$

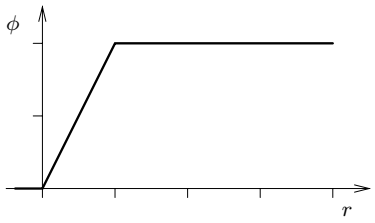


Figure 4.3. The limited downwind scheme.

$\alpha_0 = \alpha_{i+1,j}$ results in a flux that is too high. The result: $\alpha_{i+1,j}$ becomes higher and α_i lower; the difference between them increases. Thus, the gradient is sharpened. The extrapolated downwind scheme has the same effect.

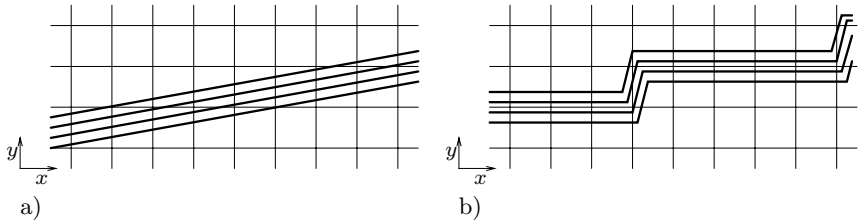


Figure 4.4. Horizontal compression turns a sharp interface (a) into a stairstep (b). The thick lines are α-isolines.

However, in two or more dimensions the limited downwind scheme is not desired in all directions. This is illustrated in figure 4.4a, where the interface in a 2D flow is almost parallel to the grid lines. Even though this interface is very sharp, the solution is smooth in horizontal direction: it takes several cells horizontally for α to change from 0 to 1. If this interface is compressed in horizontal direction, the result is an incorrect staircase profile (figure 4.4b), which continuously grows in unsteady problems and completely deforms the interface. Thus, compression must only be used in a direction normal to the interface.

The original VoF method by Hirt and Nichols [27] partially solves the staircase problem. It uses a compressive scheme based on downwind fluxes; these are limited with the donor–acceptor principle which says that the outflow of α out of a cell, in a single time step, can never be more than the total amount of α in that cell. The development of staircases is reduced by estimating the interface direction and switching to a first-order upwind scheme, instead of the compressive scheme, when the interface has an angle of more than 45^o with the cell face. In figure 4.4, this means that the vertical fluxes are computed with the compressive scheme and the horizontal fluxes with the upwind scheme.

Ubbink and Issa [64] present an improved VoF scheme. They reject the donor–acceptor scheme because it is not monotone; their CICSAM scheme is based on a limited downwind scheme in Leonard's normalised variable diagram (NVD) [38].

The NVD is completely equivalent to Spekreijse's extended monotonicity domain [58], which guarantees monotonicity when a CFL constraint is satisfied [28]. They also argue that parallel to the interface, where the solution is smooth, a scheme should be used that preserves this smooth solution, not a diffusive scheme. Thus, they use a second-order scheme, i.e. ULTIMATE-QUICKEST, the limited version of Leonard's QUICKEST scheme [37]. Finally, they propose a gradual transition between the two schemes, instead of the abrupt switch at 45^o.

Both the original VoF scheme and CICSAM are complicated. They require an estimation of the interface direction, which follows from the gradient of α; accurately computing this gradient is difficult, because the α-solution is not smooth. Also, neither scheme guarantees that α remains between 0 and 1. Many implementations of the VoF scheme reset too high or too low values for α after each time step: such schemes are not mass-conserving. Ubbink and Issa suggest a predictor–corrector approach for their scheme, which keeps α bounded and conserves mass. However, this further increases the complexity of the scheme.

A different approach is chosen by Visonneau et al. [21, 68], who use a time stepping method to compute steady flow around ships. Their MGDS scheme is based on a highly compressive limiter in the NVD. To prevent instability, this limiter is modified based on the face CFL numbers (i.e. CFL numbers based on the normal velocity on that face). For increasing face CFL numbers, the compressive scheme is gradually changed to the first-order upwind scheme. In other words, the compressive scheme is used for α-fluxes normal to the main flow direction and the upwind scheme is used parallel to the main flow.

Summarising: successful compressive VoF schemes exist for time-dependent problems. To keep α bounded and to prevent stairstep deformations, these schemes have to be complicated.

4.2.2 Steady compressive scheme The main problems of compressive schemes for unsteady flow disappear when the flow is steady. Using this, we have developed a very simple compressive scheme for our two-fluid flow model. That scheme is presented here.

In steady flow, large stairstep deformations cannot occur. For unsteady flow, the interface can have any orientation with respect to the velocity field. Therefore, disturbances in the interface shape can grow without being stopped. But when the flow is steady, the interface always follows a streamline of the flow, so the interface location is explicitly fixed by the velocity field. Therefore, every monotone scheme gives a reasonable prediction of the interface location; even the unmodified limited downwind scheme can only wrinkle the interface, not deform it.

Also, for steady flow, standard limiting is sufficient to guarantee that α stays between 0 and 1. For unsteady flow, the time stepping must be taken into account: if the fluxes into a cell cause a net inflow, then a sufficiently large time step gives $\alpha > 1$ in that cell. In the CICSAM scheme, the predictor–corrector procedure is needed to prevent this. But for steady flow, the fluxes in each cell are in equilibrium.

For 1D flow, the monotonicity principle guarantees that $0 \leq \alpha \leq 1$ in this case. In 2D that formal proof is not valid, but practice shows that α is also bounded here.

Concluding: to get reasonable steady solutions for α, the monotonicity of a limiter is a sufficient condition. Even the limited downwind scheme is acceptable.

Still, following Ubbink and Issa, we define a limiter that combines the limited downwind scheme with a second-order scheme. The resulting interfaces are a little thicker than those found with limited downwind alone, but they are smoother. And for our nonlinear two-fluid model, a non-smooth interface could disturb the velocity field, which in turn distorts the interface. Therefore, a smooth interface is desirable.

For our VoF scheme, the limited downwind scheme is combined with the $\kappa = \frac{1}{3}$ limited scheme. This is done simply with a linear combination of the limiters:

$$\phi_{\text{VoF}} = \gamma\phi_{\text{LD}} + (1-\gamma)\phi_{\kappa}. \tag{4.4}$$

The parameter γ depends on the angle between the cell face normal and the interface. For steady flow, it is not necessary to explicitly determine the orientation of the interface to choose γ. As the interface is always parallel to the flow field, γ can be set from the angle of the velocity vector. This is easier and more accurate than computing the gradient of α, which is difficult because the α solution is a discontinuity. We let γ vary linearly between $\gamma = 1$ when the interface is parallel to the cell face and $\gamma = 0$ when it is normal to the cell face:

$$\gamma = \frac{2}{\pi} \arccos \frac{|\boldsymbol{u} \cdot \boldsymbol{n}|}{\|\boldsymbol{u}\|}, \tag{4.5}$$

where $\boldsymbol{u} = [u, v]$ and $\boldsymbol{n}$ is the face normal. For α_0, γ is computed from $\boldsymbol{q}_{i,j}$, for α_1 from $\boldsymbol{q}_{i+1,j}$. Equations (4.2), (4.3), (4.4), and (4.5) form our VoF limiter. This limiter is more complex than ϕ_{LD}, but still much simpler than unsteady VoF schemes.

Let us return briefly to the MGDS scheme, which changes from compressive to first-order upwind when the face CFL numbers are high. As we saw, this means a switch to the upwind scheme when the velocity is normal to the cell face. And in steady flow, the velocity direction is the interface direction! So for steady flow, for which it is meant, the scheme depends implicitly on the interface direction with respect to the cell face. But contrary to our scheme, the discretisation parallel to the interface is not high-order accurate.

4.2.3 Numerical test To assess the performance of our scheme, we compute solutions of the linear convection equation:

$$\nabla \cdot (\boldsymbol{u}\alpha) = 0, \tag{4.6}$$

where the given velocity field satisfies $\nabla \cdot \boldsymbol{u} = 0$. The equation is discretised with a finite volume scheme. The test problem is a flow over the unit square, with velocities:

$$\begin{aligned} u &= 1, \\ v &= 1.5 \sin(2\pi x). \end{aligned} \tag{4.7}$$

The boundary conditions are $\partial\alpha/\partial n = 0$ on the top, bottom and right boundary, and $\alpha = 1$ ($y \leq 0.25$) or $\alpha = 0$ ($y > 0.25$) on the left boundary. This problem is a model for a single water wave.

As explained in section 4.2, the solution is a discontinuity only. Therefore, error measures for smooth solutions are not useful here. Instead, the global error in the interface solution is split into two parts: diffusion (the interface is smeared) and dispersion (the interface is in the wrong location). We define measures for these errors as follows:

$$e_{\text{diff}} = \int_0^1 |y_{\alpha=0.75} - y_{\alpha=0.25}|\, dx, \qquad e_{\text{disp}} = \int_0^1 |y_{\alpha=0.5} - y_{\text{exact}}|\, dx, \tag{4.8}$$

where the $y_{\alpha=\ldots}$ denote the height of α-isolines in the numerical solution and y_{exact} is the exact interface height.

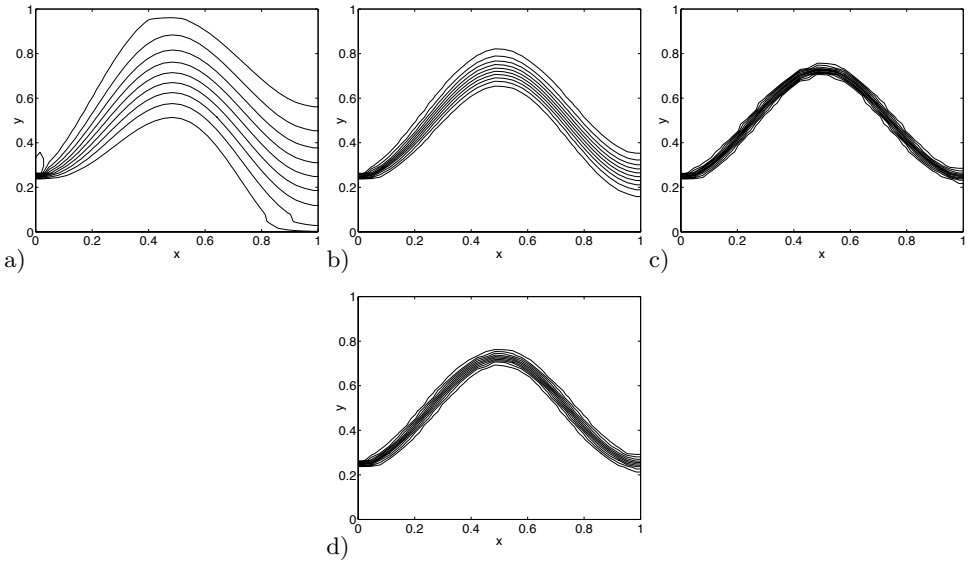


Figure 4.5. Test problem solution on 32×32 cell grid, for four different schemes: first-order upwind (a), limited $\kappa = \frac{1}{3}$ (b), limited downwind (c), and ϕ_{VoF} (d).

Solutions. Figure 4.5 gives the solutions[2] to the test problem, for the upwind scheme and three limited schemes. As expected, the first-order scheme (figure 4.5a) is very diffusive. The $\kappa = \frac{1}{3}$ limiter (figure 4.5b) also produces a diffused solution, although the interface is smeared a lot less than in the first-order case. The limited downwind scheme (figure 4.5c) is not diffusive: the interface is smeared over a few cells, but it keeps the same thickness as it progresses downstream. Instead, it is too compressive: the staircase deformations are obvious. This is not catastrophic, the

[2]The first-order solution is found with direct inversion, the limited discretisations are solved with a Runge-Kutta explicit time stepper.

interface is in the right location, but it is not smooth. Finally, our VoF limiter (figure 4.5d) also gives a constant interface thickness, but the interface is much smoother than for the limited downwind scheme.

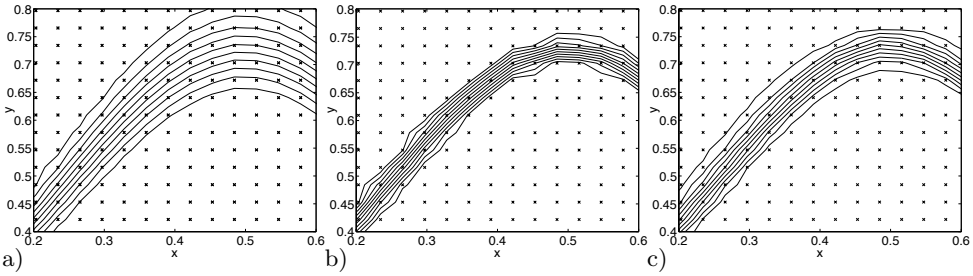


Figure 4.6. Close-up of figure 4.5, with $\times$ indicating the cell centres. The three limiters are ϕ_κ (a), ϕ_{LD} (b), and ϕ_{VoF} (c).

Looking at the interfaces in close-up (figure 4.6), we see that ϕ_{LD} spreads the interface over two cells, compared to about five cells for the second-order scheme (more than five, further downstream), and that it causes staircase deformations. The interface for our VoF limiter is one cell wider, but it is almost completely smooth.

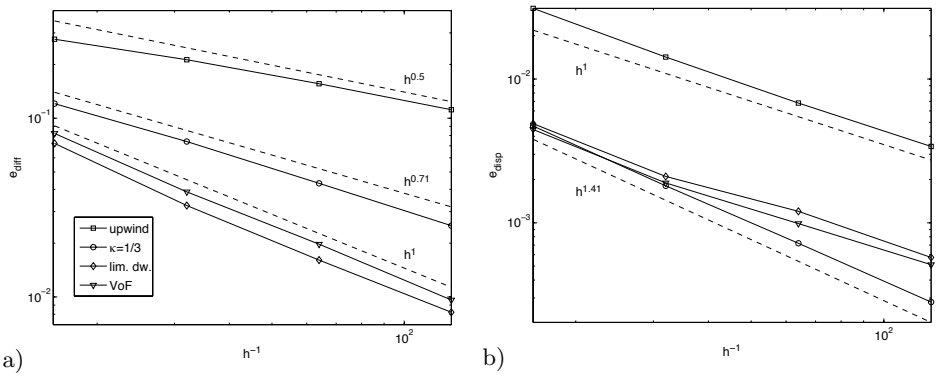


Figure 4.7. Grid convergence of the errors e_{diff} (a) and e_{disp} (b).

Diffusive errors. A curved interface can never be resolved exactly on a rectangular grid, it is always smeared over two or more cells. An ideal scheme keeps the interface thickness constant: then it is never smeared more than necessary. For such a scheme, the thickness decreases linearly with the mesh width. Therefore, the maximum order of convergence for e_{diff} is $\mathcal{O}(h)$. On the other hand, a diffusive scheme with a constant diffusion coefficient produces an interface that spreads with the square root of its length, and with the square root of the diffusion coefficient (this is the case for physical diffusion layers). So for a first-order scheme, where the numerical diffusion decreases linearly with the mesh width, the error e_{diff} converges with $\mathcal{O}(h^{\frac{1}{2}})$.

Figure 4.7a gives the grid convergence of e_{diff} for the four schemes. The first-order scheme converges asymptotically with $\mathcal{O}(h^{0.5})$, as expected. The limited $\kappa = \frac{1}{3}$ scheme is diffusive, but its order of convergence is higher than 0.5: as the interface spreads, it becomes smoother, so the r-values lie closer to 1. Then the limiter uses the $\kappa = \frac{1}{3}$ scheme more so less numerical diffusion is applied. The two other schemes are compressive, they converge with $\mathcal{O}(h)$. The ϕ_{LD} limiter gives the thinnest interface.

Dispersion errors. What we call 'dispersion error' is not dispersion in the original meaning, which is an error in the wavelength of the solution. Instead, e_{disp} is caused by oscillations of the $\alpha = 0.5$ isoline around the exact solution, within one wavelength. As opposed to e_{diff}, it is not explicitly bounded from below by the cell size; it can become zero on any grid.

In the results (figure 4.7b), we see $\mathcal{O}(h)$ convergence for the first-order solution. The $\kappa = \frac{1}{3}$ scheme does not show $\mathcal{O}(h^2)$ convergence, but it gives the lowest errors of all schemes. The ϕ_{LD} scheme has $\mathcal{O}(h)$ convergence, caused by the compression of the interface: a staircase solution gives an $\mathcal{O}(h)$ dispersion error. The ϕ_{VoF} scheme also has asymptotic $\mathcal{O}(h)$ convergence, but less staircase effects, so the error is lower.

Concluding: to get a non-diffusive and yet smooth solution, the ϕ_{VoF} limiter is the best choice.

4.2.4 Wiggles Using the nonlinear limited schemes for discontinuous problems has a disadvantage: the steady solutions sometimes do not completely converge (figure 4.8a). This problem may appear when a part of the interface is almost parallel to the grid.

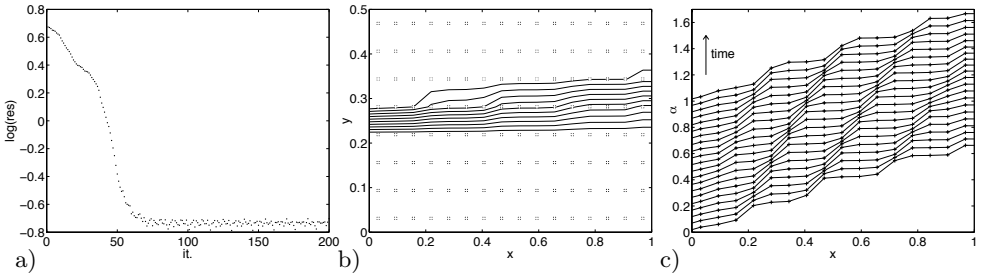


Figure 4.8. Non-convergence of a solution with the flow nearly parallel to the grid. Convergence of the residual for a time stepping solver (a), the solution for α (b) (where the $\times$ mark the cell centres), and the solution in the cells at $y = 0.275$, for 20 consecutive time steps starting after 200 steps (c).

The explanation is, that the limiter in the direction parallel to the flow keeps switching between schemes in some cells. Consider for example figure 4.8b, a uniform flow with $v \ll u$. In a horizontal row of cells, α varies very smoothly. On the other hand, in vertical direction, the interface is sharp. Therefore, a small disturbance in the vertical position of the interface creates a large change in α in the cells. The result is that, in horizontal direction, r can easily change from $r > 1$ to $r < 1$. For

the ϕ_{LD} limiter, this means a change from the extrapolated downwind scheme to the normal downwind scheme. It is this changing of schemes that causes the oscillations.

The disturbances propagate downstream as waves, which carry the change in the limiter with them. Figure 4.8c shows these waves in a horizontal row of cells, for 20 consecutive time steps of the solver. The same phenomenon may appear for curved interfaces. There, the oscillations start where the interface is almost parallel to the grid, but the waves still travel downstream all the way, even though most of the interface is not parallel to the grid.

The problem appears if r can change enough in a single cell. The chance of such a cell appearing is high when the interface is sharp, so the oscillations happen mostly when a compressive limiter is used. However, they can also happen for the ϕ_κ limiter, even though this limiter gives a linear scheme near $r = 1$: when r changes enough, oscillations appear.

Luckily, for our application the oscillations cause no problems. They are relatively small and do not disturb the interface shape much (The oscillations in figure 4.8c are extreme). Also, as we shall see in the next section, the defect correction solver does not require complete convergence anyway. Therefore, the limited schemes are acceptable.

4.3 Defect correction

To solve our second-order accurate two-fluid model, the multigrid solver is combined with defect correction (DC). This section introduces the defect correction technique and explains the changes that are necessary when it is used for the two-fluid model.

4.3.1 Defect correction procedure In general, defect correction is a broad class of methods that solve a difficult problem iteratively using the inverse of a similar, but easier problem. Multigrid itself can be seen as a defect correction method.

Here, DC is used to solve the second-order accurate flow equations, by approximating them with the first-order equations [9, 63]. For convection-dominated problems, solving second-order discretisations with MG is difficult because effective smoothers are not available (some smoothers exist, but these give rather slow convergence rates [63]). Defect correction is an attractive alternative because it solves the second-order equations using the effective multigrid solver for the first-order equations.

The defect correction algorithm is defined as follows. Let the first-order discretisation of the flow equations be $\mathcal{F}_{1,K}$ (this is $\mathcal{F}_K$ in equation (2.38)) and the second-order discretisation $\mathcal{F}_{2,K}$. Then one DC iteration is basically:

$$\boldsymbol{q}_K^{n+1} = \tilde{\mathcal{F}}_{1,K}^{-1} \left(\mathcal{F}_{1,K} \boldsymbol{q}_K^n - \mathcal{F}_{2,K} \boldsymbol{q}_K^n \right),$$

where $\tilde{\mathcal{F}}_{1,K}^{-1}$ is the approximate inverse of $\mathcal{F}_{1,K}$, obtained with one MG cycle. When the iteration converges, i.e. $\boldsymbol{q}_K^{n+1} = \boldsymbol{q}_K^n$, then $\mathcal{F}_{1,K} \boldsymbol{q}_K^n = \mathcal{F}_{1,K} \boldsymbol{q}_K^n - \mathcal{F}_{2,K} \boldsymbol{q}_K^n$, so $\mathcal{F}_{2,K} \boldsymbol{q}_K^n = 0$. To allow scaling of the defect, a parameter w is introduced again (see

also section 2.3.1). Furthermore, we introduce underrelaxation ω:

$$\boldsymbol{q}_K^{n+1} = \boldsymbol{q}_K^n + \omega \frac{1}{w_K^n} \left(\tilde{\mathcal{F}}_{1,K}^{-1} \left(\mathcal{F}_{1,K} \boldsymbol{q}_K^n - w_K^n \mathcal{F}_{2,K} \boldsymbol{q}_K^n \right) - \boldsymbol{q}_K^n \right). \tag{4.9}$$

The choice of ω and w_K^n is discussed below, in section 4.3.2.

The residual of the DC process converges very slowly, or may not converge at all. Désidéri and Hemker [9] have shown that convergence of the residual is not necessary when the DC is started from the converged solution of the first-order equations, i.e. $\mathcal{F}_{1,K} \boldsymbol{q}_0^n = 0$. They prove that for linear problems, a single DC step improves this solution to second-order accuracy. Their explanation is, that the error components that make the solution first-order accurate are removed quickly by the defect correction. Other error components may damp slowly or even diverge, but they are so small initially that the solution with these errors is still second-order accurate.

For nonlinear problems, more than one iteration is needed to get second-order accuracy; for discontinuous problems, formal second-order accuracy can never be obtained, as shown in the previous section. Still, section 4.4 shows that a few DC steps greatly improve our first-order accurate solutions. This suggests that defect correction is a very efficient solution technique for our two-fluid flow model.

4.3.2 Solving the two-fluid model To solve the complicated, nonlinear two-fluid flow equations, standard defect correction needs some changes. These changes are explained here.

Defect correction gives the best results when $\mathcal{F}_{1,K}$ and $\mathcal{F}_{2,K}$ resemble each other. For two-fluid flow, this resemblance is good in the flow away from the water surface, where the two operators produce similar results. But near the surface, the operators are very different. This leads to large changes in the solution in the first DC steps and to the need for damping.

The second-order discretisation, with the compressive ϕ_{VoF} limiter for α, produces much thinner interfaces than the smeared-out results of the first-order scheme. Therefore, the defect correction procedure near the surface gives large changes in α in the first steps. As these changes are computed with the inverse of $\mathcal{F}_{1,K}$, instead of the inverse of $\mathcal{F}_{2,K}$, they are not always completely correct. In fact, the experiments in section 4.4 and a test on the linear problem from section 4.2.3 (figure 4.9) show that DC always produces an over- and undershoot alongside the interface. These overshoots may grow and cause the divergence of the solver.

The problem is solved by resetting all $\alpha > 1$ to 1 and all $\alpha < 0$ to 0, after each DC iteration. This stabilises the defect correction. After a few iterations, when the solution is close to its converged state, the overshoots no longer appear and the reset is not needed anymore. The reset is not mass-conservative. However, as opposed to unsteady flow, the iteration converges to a steady solution that does conserve mass. The reset makes even the intermediate solutions physically more realistic.

Related to the undershoots in α are the overshoots in the velocity updates. The experiments in section 4.4 show that the velocities in the air region, in the zones where α undershoots, become too high. This often leads to instabilities in the

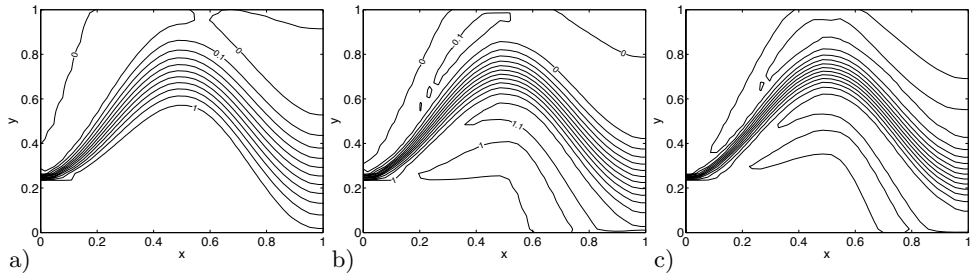


Figure 4.9. Over- and undershoot of α produced by defect correction on the problem of section 4.2.3. The limiter is ϕ_{VoF}, the result is shown after 1 (a), 2 (b), and 3 (c) DC steps.

iteration. This is solved with a simple damping: in the air region, the correction to all state variables in a cell is multiplied with the same factor, that is chosen as the maximum value that still keeps $\alpha \geq 0$, after the correction. In practice, this factor is either 1 or 0; the correction is completely suppressed in the zones where α undershoots. Section 4.4 shows, that this crude damping is effective.

Finally, the stability of the DC iteration requires the scaling w_K^n and underrelaxation with ω. The operator $\mathcal{F}_{1,K}$ can handle states $\boldsymbol{q}_K^n$ away from its own solution $\boldsymbol{q}_K^0$, but not large source terms. Therefore, w_K^n is used to keep the source terms small, like in section 2.3.3. Equation (2.49) is used for w_K^n, mostly with $D = 10^1$. Unfortunately, the solution depends strongly on the choice of D. The usual choice for the underrelaxation is $\omega = 0.9$. The following section further discusses the choice of the parameters.

4.4 Test cases

This section tests the combination of defect correction with the second-order accurate discretisation and studies the effectiveness of the compressive ϕ_{VoF} limiter. First, DC is used on a single-fluid flow, the supercritical airfoil flow from the previous chapter. Then the combination of DC and the compressive limiter is tested with the Cahouet flow. The accuracy of the solutions and the convergence of the defect correction is studied.

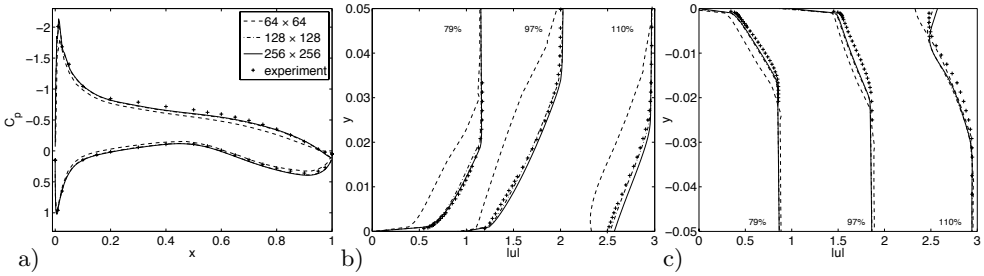


Figure 4.10. Supercritical airfoil, grid convergence study. Solutions for c_p (a) and for the velocity profiles above (b) and below the airfoil (c) are compared with measurements from [49].

4.4.1 Supercritical airfoil The grid convergence study for the supercritical airfoil from section 3.6.4 is repeated here with the limited discretisation. The limiter is ϕ_κ; the defect correction uses $w_K^n = 1$ and $\omega = 1$ (no scaling, no underrelaxation) and the solutions after 5 DC steps are presented.

The pressure coefficient and the velocity profiles are given in figure 4.10. These results are excellent. The grid convergence rate is clearly higher than first-order and the solution on the middle, 128×128 grid is essentially converged apart from a small zone in the wake. Compared with the first-order solution (figure 3.11), the stagnation point with $c_p = 1$, the suction peak, and the shape of the boundary layers are predicted much better. The remaining differences with the experimental results are probably due to the approximate turbulence model.

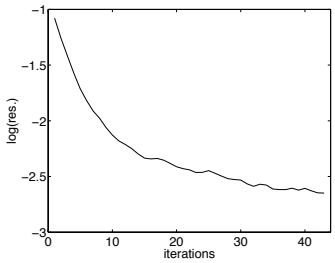


Figure 4.11. Supercritical airfoil, convergence of the defect correction residual on the 256×256 grid.

Figure 4.11 shows the convergence of the residual on the finest grid. The rapid convergence in the first steps and the slow asymptotic convergence are typical for defect correction [9]. Figure 4.10 confirms that full convergence is not needed to get good results.

4.4.2 Cahouet test Now defect correction is combined with our two-fluid model, to obtain accurate results for Cahouet's channel flow (section 2.5.1). We use undeformed grids (section 3.7.1) and solve the two test cases where first-order solutions exist on these grids. The current defect correction procedure is not yet robust enough: instabilities appear near the boundaries and the residual eventually diverges. But despite this, we show that the results obtained are reliable and accurate.

$Fr = 0.43$. Figure 4.12 shows the results for the $Fr = 0.43$ test case with the thin, $E = 0.075L$ bump. These are obtained on the 256×256 cell grid, with underrelaxation $\omega = 0.9$ and w_K^n given by equation (2.49) with $D = 10^1$, after 8 DC steps. The limiter is ϕ_κ, with ϕ_{VoF} for α.

The figures show much more pronounced waves than in the first-order case (figure 3.14). Especially the vertical velocity plot 4.12b shows that the waves damp out slowly. Although the damped region just above the interface and the approaching instability in the air (see section 4.3.2) are clearly visible, these do not affect the solution in the water much, because the air influences the water very little. To check this, the damping was switched off. In the solution after 3 DC steps, the water surface with and without damping was identical.

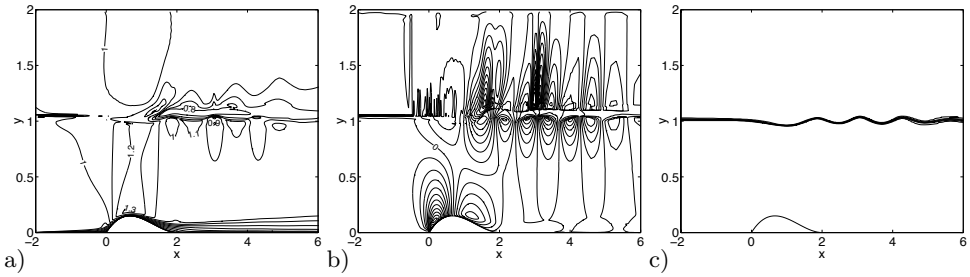


Figure 4.12. $Fr = 0.43$ flow, with thin bottom bump ($E = 0.075L$). Isoline plots of speed $|\boldsymbol{u}|$ (a), vertical velocity v (b), and volume fraction α (c), after 8 DC steps.

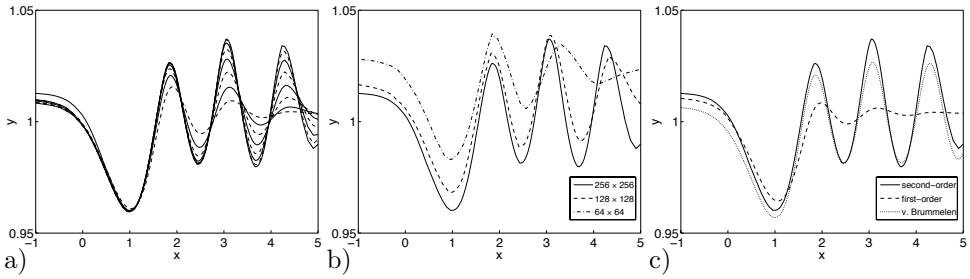


Figure 4.13. Convergence studies for the $Fr = 0.43$ flow. The $\alpha = 0.5$ isolines are shown in the first 8 DC steps on the 256×256 grid (a) (- - -: odd-numbered steps, —: even-numbered steps) and after 8 steps on three grids (b), and the first- and second-order solution on the 256×256 grid are compared with Van Brummelen's solution [5].

Figure 4.13 shows convergence studies. In figure 4.13a, the solution on the finest grid is given in the first 8 DC steps. The shape of the first wave has converged after only four steps and the next wave is essentially converged after 8 steps. Even the third wave crest, which lies in the zone of stretched cells (figure 2.24), is almost converged. This figure shows that, even if the residual has not converged after 8 steps, the solution itself is very close to convergence.

Figure 4.13b gives the solution on three grids, after 8 steps[3]. Even the solution on the coarsest grid has waves, which are absent from the first-order solution (figure 3.14f). The shape of the first two waves has converged on the finest grid; the difference with the middle grid is mainly due to the differences in the average water height, that is determined by boundary layer effects downstream. The finest grid has about 40 cells per wavelength, which is reasonable. The accuracy of the solution is also confirmed by figure 4.13c: the solution on the finest grid is compared with the solution that Van Brummelen [5] obtained with surface fitting. The agreement, especially in the wave length, is excellent. The difference in the amplitudes is probably due to a different starting point for the boundary layer (section 2.5.1).

Finally, figure 4.14 shows the solution for α near the water surface. The interface

[3]Both here and for the $Fr = 2.05$ test case, the solution on the coarsest grid is computed with $\omega = 0.6$. Therefore, the solutions are used after 12 DC steps ($Fr = 0.43$) or 9 DC steps ($Fr = 2.05$).

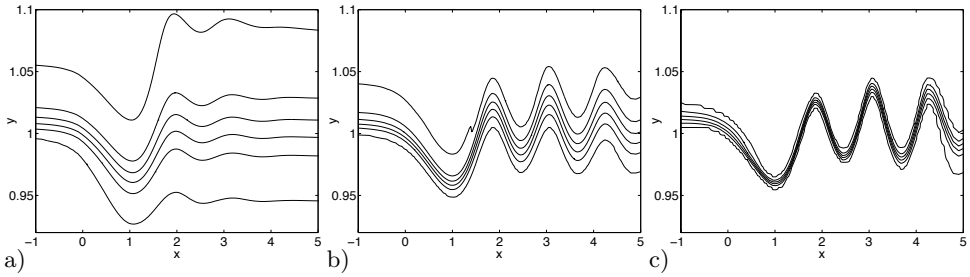


Figure 4.14. $Fr = 0.43$, a detail of the α-solution. The isolines are equally spaced between $\alpha = 0.01$ and $\alpha = 0.99$. First-order solution (a) and the solution with the ϕ_κ (b) and ϕ_{VoF} limiter (c), after 8 DC steps.

with the ϕ_{VoF} limiter is much thinner than the first-order solution and the solution with ϕ_κ. Also, the wave amplitude is a little higher for ϕ_{VoF} than for ϕ_κ. This is different from the linear test, where ϕ_κ gave the best prediction of the wave location (figure 4.7b). It is purely a discretisation effect, not an effect of the defect correction: the ϕ_κ solution is nearly converged, just like in figure 4.13a. Apparently, the lower diffusion of the interface reduces the numerical damping of the whole flow. The grid refinement study in figure 4.13b shows steeper waves on finer grids, so the steeper waves are more accurate. Thus, not only does the ϕ_{VoF} limiter give a sharper resolution of the interface, it also produces more accurate wave shapes.

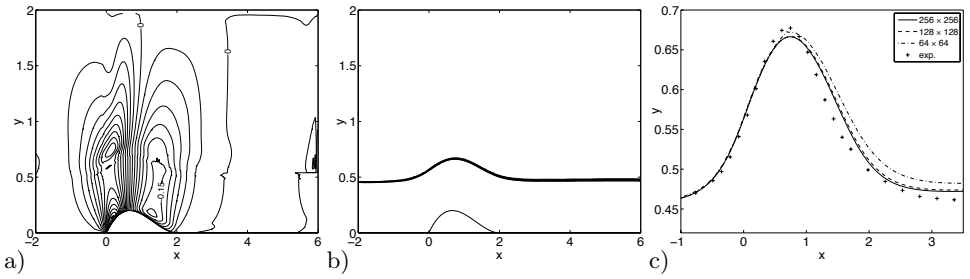


Figure 4.15. $Fr = 2.05$ flow. Isoline plots of vertical velocity v (a) and volume fraction α (b). Grid convergence of $\alpha = 0.5$ isoline and comparison with Cahouet's experiment (c). 6 DC steps.

$Fr = 2.05$. A second test is done on the $Fr = 2.05$ test case, using $\omega = 0.9$ and $D = 10^{-1}$. The solution is given in figure 4.15. Compared with the first-order solution in figure 3.19, the interface is much thinner; it has a constant thickness of about four cells. The grid convergence (figure 4.15c) is excellent, the agreement with Cahouet's experiment is good too.

Figure 4.16 shows a detail of the interface, for three different limiters (compare with figure 4.6). The ϕ_κ limiter smears the interface, while ϕ_{LD} gives a very thin solution. However, to keep the solution with the ϕ_{LD} limiter stable, the interface had to be reset to a monotone function of y after each DC step; the figure shows the

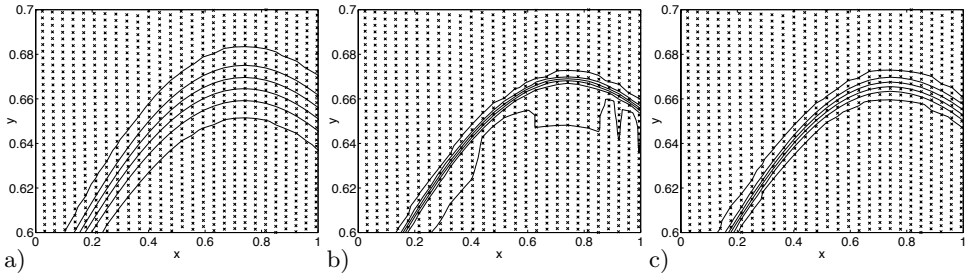


Figure 4.16. $Fr = 2.05$, detail of α-isolines (equally spaced between 0.01 and 0.99) for the ϕ_κ (a), ϕ_{LD} (b), and ϕ_{VoF} (c) limiters. 6 DC steps.

dents in the lowest isoline where this happened. (For the $Fr = 0.43$ problem, the DC iteration with ϕ_{LD} immediately became unstable, even with this correction.) The ϕ_{VoF} solution does not need this reset: it is a little thicker than the ϕ_{LD} solution, but it is smoother and can be obtained by resetting the $\alpha > 1$ and $\alpha < 0$ only.

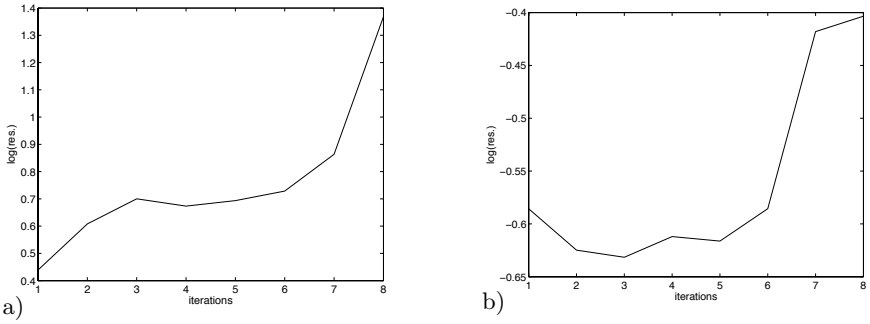


Figure 4.17. Total residual during the defect correction solution for the $Fr = 0.43$ (a) and $Fr = 2.05$ (b) test case.

DC convergence. At this moment, the DC solver for the two-fluid model is not robust enough. The residuals during the iteration do not converge (figure 4.17) and both iterations fail catastrophically after about 9 or 10 steps. This failure is caused by the gradual development of oscillations in the solution. These oscillations develop in the air region, in two locations: just above the damped band next to the interface (see figure 4.12b) and on the aft boundary (see figure 4.15a). The appearance of oscillations is sensitive to the values of w_K^n. Tests show, that larger values of D give more oscillations above the surface, while lower values of D cause oscillations at the boundary.

At this moment, the cause of the instabilities is not known, so we cannot guarantee that the defect correction solver gives a solution for any given problem. However, the results that we do obtain indicate that the main flow in the water is not affected by the first appearance of the oscillations. Also, in these cases the quality of the results is excellent; the convergence of a solution can be ascertained by comparing

Table 4.1. Defect correction CPU times, compared with the first-order solution times. (64×64 grid, see footnote 3.)

	$Fr = 0.43$ (8 DC it.)			$Fr = 2.05$ (6 DC it.)		
Grid	levels	time (s)	% of MG	levels	time (s)	% of MG
64×64	6	13.6	36	5	12.1	34
128×128	7	52.5	28	6	34.1	34
256×256	8	215	24	8	166	27

the states after each DC step like in figure 4.13a. Therefore, DC still is a promising technique for the solution of the two-fluid flow model.

In terms of CPU time, DC is not expensive (table 4.1). A single DC step takes more time than an MG cycle on the finest grid, because the DC needs to compute the source term $\mathcal{F}_{1,K}\boldsymbol{q}_K^n - \mathcal{F}_{2,K}\boldsymbol{q}_K^n$ before the MG cycle is started. However, the number of DC steps needed is lower than the number of MG iterations for convergence on the finest grid. Therefore, the CPU time for the DC is about a third of the solution time for the first-order equations. For this little extra computation time, the quality of the first-order solutions is greatly improved.

4.5 Conclusion

An extension of the two-fluid model to second-order accuracy has been presented. The convective fluxes are discretised with the Riemann-solver flux function from the first-order scheme, combined with limited reconstruction of the cell face states. The diffusive fluxes and the turbulence source terms are modelled with the existing central discretisations.

A compressive limiter has been developed for the volume fraction α. The α-solutions consist of a discontinuity only, so a second-order accurate limiter does not give good solutions. Instead, an antidiffusive scheme is used that keeps the water–air interface thin. Because our flow solutions are steady, the complicated features of compressive schemes for unsteady flow are not needed. Any monotonicity-preserving limiter keeps the values for α between 0 and 1. Also, the velocity field explicitly determines the interface location, so compressive limiters can never cause large stairstep deformations of the interface.

Still, a limiter is used that is a combination of a compressive scheme and a second-order scheme. The two schemes are mixed according to the angle between the interface and the cell face. Because the flow is steady, the interface is parallel to the flow, so this angle is estimated easily from the velocity field. The result is a scheme that keeps the interface both sharp and smooth.

Numerical tests on a linear problem show that compressive schemes keep the thickness of the interface constant over its entire length: no diffusion occurs. Our mixed limiter indeed gives smooth solutions.

The second-order discretisation is solved with defect correction: the residual of the second-order operator is fed as a source term into the MG solver for the first-order operator. The highly-nonlinear flow equations cannot handle large source terms, therefore the source term is scaled while the correction from each DC step is unscaled. Also, for better stability, underrelaxation is used.

Near the water surface, the first- and second-order operators are very different. This causes over- and undershoots in the corrections to the velocity and the volume fraction, as well as instabilities in the velocity solution. These effects are reduced by damping all corrections in the air region such that α remains ≥ 0 and by resetting the $\alpha > 1$ to 1.

Results are computed for two cases: the single-fluid supercritical airfoil and the Cahouet channel flow. The DC convergence is slow, the two-fluid computations are even unstable. However, excellent solutions are obtained after only a few DC steps. The solutions effectively converge in less than 10 steps, the wave damping is greatly reduced with respect to the first-order solutions, and the agreement with experiments and existing solutions is good. The compressive scheme is effective, both in keeping the water surface sharp and in reducing the numerical wave damping. Starting from a first-order solution, the CPU time for the DC solution is about one third of the time for the first-order computation.

Outlook. At this moment, the defect correction process for the two-fluid model is not robust enough. Instabilities appear in the air region and near the outflow boundary, that grow and cause the iteration to fail at some point. It is not guaranteed that the flow solution has converged enough before the iteration fails.

It is expected that an adaptation of the damping or another change in the current defect correction procedure can eliminate these problems. The instabilities only appear in the air region of the two-fluid solutions; both in the single-fluid solutions and in the water region, defect correction causes no instability. The precise cause of the instability could be related to the non-convergence of the α-solution in the linear test. When the cause is further investigated and found, it is expected that defect correction in the air region can be adapted to function as efficiently as in the water region.

If the instability is eliminated, then the second-order discretisation with defect correction becomes an efficient tool for the simulation of water flow; the solution quality that can be obtained is proved by the results in this chapter.

Local grid coarsening for ship flow grids

In this chapter, the accuracy of surface capturing models is studied again, for three-dimensional flow around ships. While the focus of this thesis is mainly on solution speed, the previous chapter also analyses how a good spatial discretisation can improve the global accuracy of the solution. There is yet another way to increase the solution accuracy: local grid refinement. This chapter presents a local grid refinement method that is tailor-made for surface capturing solution of ship flow.

As explained in the Introduction (chapter 1), there are two numerical techniques for computing steady water flow around ships: surface fitting and surface capturing. For surface fitting, the grid is deformed to fit the water surface and free surface boundary conditions are imposed on the upper boundary of the grid. For surface capturing, the mesh is not deformed. With surface reconstruction, that is used here (as opposed to the previous chapters), the free surface boundary conditions are imposed on a plane that runs through the grid. The location of this plane is determined with a volume-of-fluid or level set approach.

The main advantage of surface capturing is its flexibility: it can compute flows around complex-shaped ship hulls like wetted transoms. On the other hand, surface fitting methods compute solutions faster. Also, for similar grids, surface capturing usually gives less accurate solutions (figure 5.1). This is due partly to the smearing of the marker function ϕ that indicates the free surface (equation (1.1)) and partly due to the fact that the boundary conditions are not imposed on the grid points.

To improve the accuracy of a surface capturing technique, local grid refinement is a possibility. The errors in the solution are likely to be concentrated in some parts of the flow domain, such as near the ship's hull and the water surface. The rest of the solution domain can contain coarser cells than these parts, so local refinement can be advantageous. Also, as opposed to surface fitting, the capturing meshes need not be deformed automatically, so unstructured and locally refined grids can be easily used.

On the other hand, unstructured grids have disadvantages for surface capturing. In the boundary layer, with its high aspect ratio cells, tetrahedral cells can have high skewness and may cause instability of the solution. And for the stability of the free surface, it is useful to have grid planes there that are more or less aligned with the surface. Therefore, standard unstructured grids cannot be used.

We present a method of local grid refinement for an unstructured grid solver, that can be used with surface capturing. First, we perform flow computations on structured grids to find the best locations for grid refinement. A grid convergence study is used to determine the locations of the largest errors. Then two tests are done on structured grids where the cells are clustered in these locations, to see if local refinement there is useful.

This chapter is joint work with Dr. Takanori Hino, National Maritime Research Institute, Tokyo, Japan.

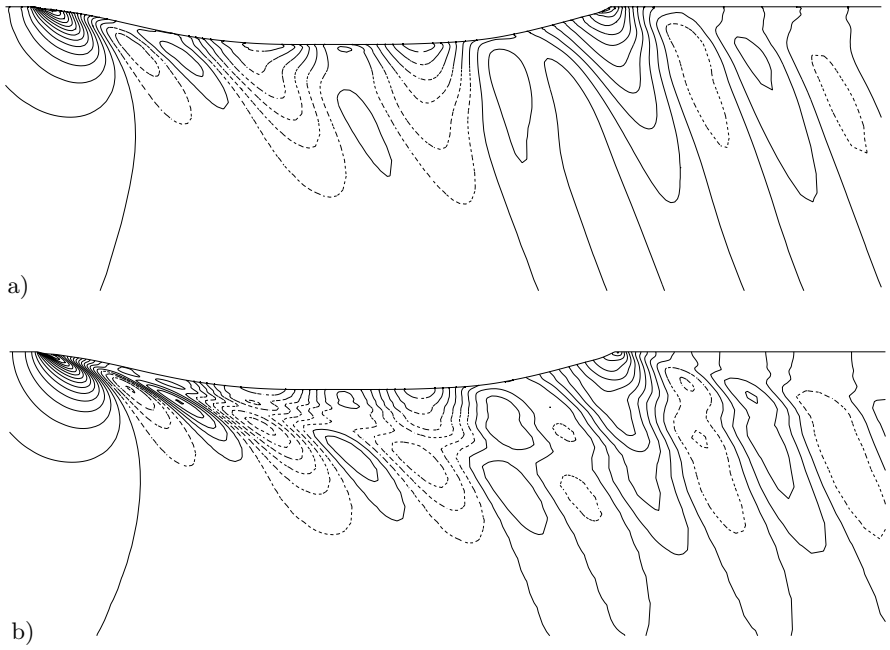


Figure 5.1. The difference between a surface capturing (SURF) (a) and a surface fitting (NEPTUNE [26]) (b) solution. Free surface height, Series 60 hull, $Fr = 0.20$. Isoline spacing 0.0005 (- - -: negative values). See also the experimental results in figure 5.16.

The grid refinement method is then described. It constructs locally refined unstructured grids that can be used directly in a normal unstructured solver, but consist of structured hexahedral cells as much as possible. Thus, our grids consist of nested blocks of hexahedrals, that are connected by single-cell layers of other cells. Even these layers are kept as regular as possible. For the greatest accuracy in the geometry of the fine cells and for good grid smoothness, the grids are constructed with grid coarsening, a structured grid generator is used to make a very fine mesh and cells of this mesh are either used directly or merged to form coarser cells. This way, the best possible definition of the hull shape is obtained.

The grids are used with SURF, the unstructured surface capturing code developed at NMRI [23, 24]. SURF is a finite-volume solver for the Reynolds-Averaged Navier-Stokes equations, with a level set representation of the free surface. The code accepts tetrahedral, pyramid, prism, and hexahedral cells and can therefore be used with both structured and unstructured grids. Here, the code is first used for the structured grid tests, then it is used with a locally refined grid.

In all these tests, the test case is the flow around a Series 60 (block coefficient $C_b = 0.6$) hull at $Fr = 0.2$ and $Re = 3.746 \cdot 10^6$, with fixed attitude, for which experimental results are reported by Kusaka [35]. The Spalart-Allmaras one-equation turbulence model is used [57].

This chapter starts by describing the tests to find the optimum locations for local refinement (section 5.1). Then section 5.2 describes our basic method for constructing coarsened grids. In section 5.3, this method is used to make a grid around the Series 60 hull. Finally, section 5.4 presents the results obtained with this grid.

5.1 Coarsening location tests

For local grid coarsening, the most important choice is where to coarsen the grid. To establish guidelines for the coarsening, we perform tests on structured grids. This section describes these tests: a grid convergence study to estimate the global errors in the solution and two local refinement tests.

5.1.1 Test grids All structured grid tests are performed on H-O type grids; the grid geometry is sketched in figure 5.2. The basis fine grid has $216 \times 64 \times 96$ cells in i-, j-, and k-direction, runs from $x = -1.5$ to $x = 2.5$ and out to $y = -2.0$. The ship is located between $x = -0.5$ and $x = 0.5$. To enable surface capturing, the grid has horizontal j = constant grid planes near the free surface. Figure 5.3 shows the grid from above, on the hull, and in two x cross-sections.

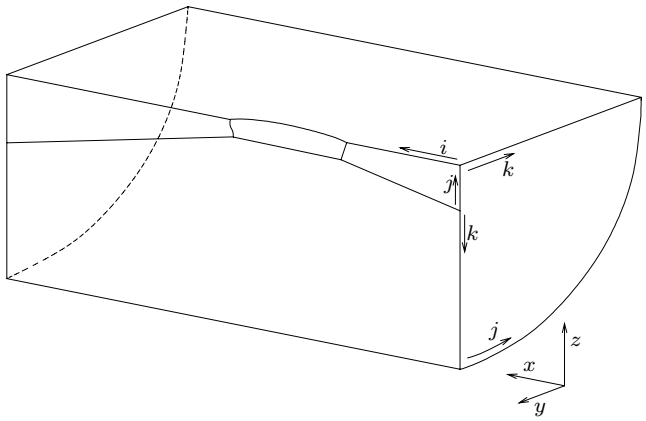


Figure 5.2. H-O grid geometry for Series 60 tests.

For the convergence test, the grid is coarsened twice or four times in each direction, to create a middle grid with $108 \times 32 \times 48$ cells and a coarse grid with $54 \times 16 \times 24$ cells. To be used with SURF, the structured grids are stored as hexahedrals in an unstructured grid format.

5.1.2 Grid convergence study To estimate the location and size of the global solution errors, the test case flow is computed with SURF on the structured fine, middle and coarse grid. As a reference, the solution is also computed with NMRI's surface fitting code NEPTUNE [26], on a grid that is similar to the SURF fine grid. As seen in figure 5.1, this solution is more accurate than the SURF fine grid solution: it has stronger waves and more wave detail.

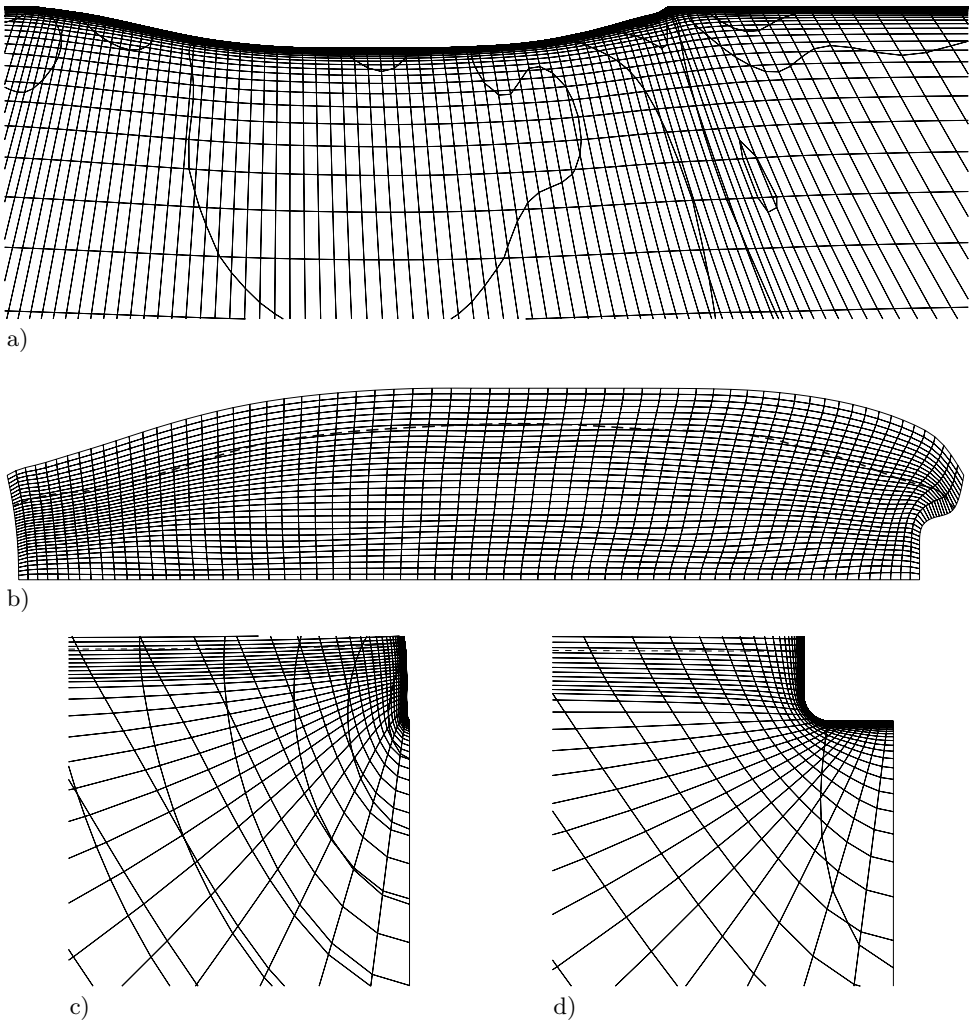
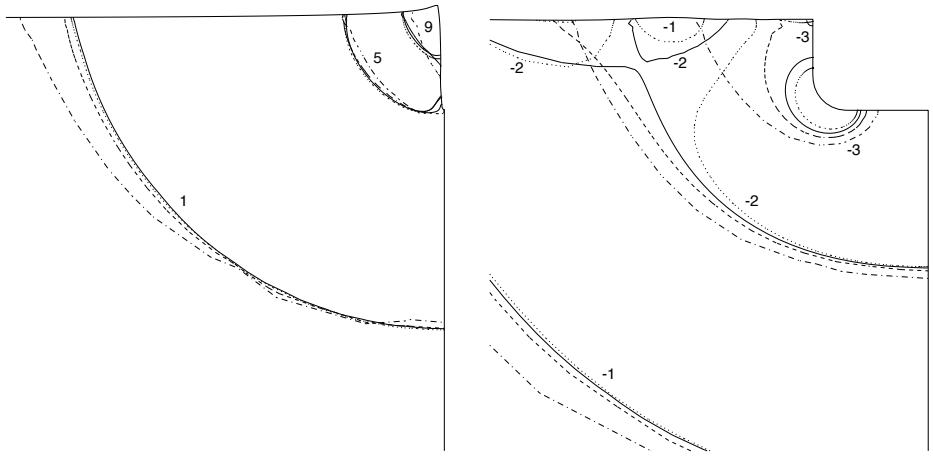


Figure 5.3. Structured middle grid. Cross-section at the water surface (detail) (a), grid on the hull, seen from diagonally below (b), and cross-sections (detail) at $x = -0.48$ (c) and $x = 0$ (d). The ship's bow is to the left. The irregular curves are cross-sections through j = constant faces (a) or i = constant faces (c, d). - - -: free surface.



a) b)

Figure 5.4. x cross-section pressure$\cdot 10^2$ on three structured grids. $x = -0.48$ (a) and $x = 0$ (b). —: fine grid, - - -: middle grid, - · -: coarse grid, $\cdots$: NEPTUNE fine grid.

Figure 5.4 gives pressure contours in two x-cross sections. The figure clearly shows that the main errors in the SURF solution are associated with the free surface. Far below the surface, the solution is accurate on the finer grids and is even approaching asymptotic convergence. But near the surface, the wave amplitudes are not converged, they increase a lot as the grid becomes finer; this is related to the larger pressure differences seen in figure 5.4. Another region of large errors, just visible close to the ship hull in the upper right corner of figure 5.4a, is the pressure peak on the bow.

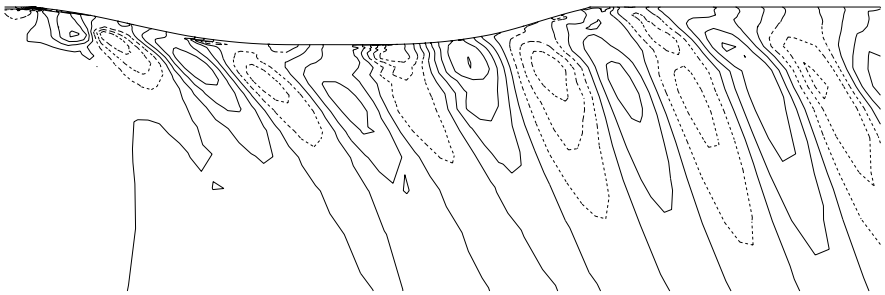


Figure 5.5. Difference between the pressures on the fine and the middle grid, for $z = -0.01$. Isoline spacing is 0.0025.

Figure 5.5 shows the difference between the fine and the middle grid solution in a horizontal plane just below the free surface. A comparison with figure 5.1 shows that the locations of the largest differences in the solutions correspond exactly to the locations of the wave crests and troughs.

So concluding: the largest errors in the SURF solutions are associated with the

waves; they are found below the free surface in an x, y zone that contains the largest waves. Another location of large errors is the pressure peak near the bow. These zones are relatively small, so locally refining the grid there is efficient: it does not increase the grid size much.

5.1.3 Local refinement tests Two structured-grid tests are done to study the effects of local grid refinement. In these tests, whole planes of cells are refined, so the grids remain structured. The first test concerns refinement in j-direction, in the zone identified in the previous section 5.1.2. The second test studies very local j-refinement near the free surface.

Local j-refinement. Two different grids are derived from the standard middle grid, by refining this grid in j-direction. The first grid has the cell sizes of the middle grid in i- and k-direction, but the (twice finer) spacing of the fine grid in j-direction. The second grid has this fine j-spacing only in the upper half of the grid, which is roughly the region higher than the ship's bottom. If these grids give comparable accuracy in the solutions, then local refinement near the free surface is effective.

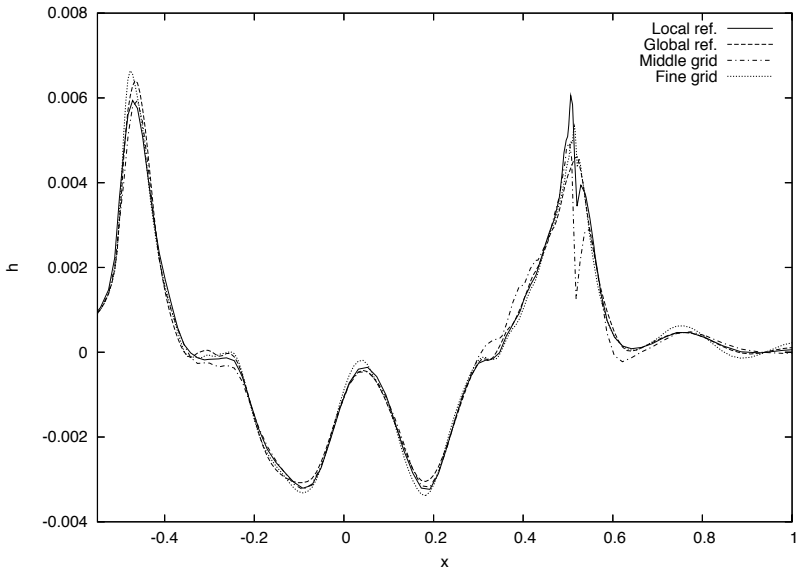


Figure 5.6. Wave profiles on the hull and centreline, for the locally and globally j-refined grid.

Figure 5.6 compares the wave profiles on the hull for the two grids. As a reference, it also shows the solution on the standard middle and fine grid. The figure proves that the two grids give almost the same solution. In some locations, the locally refined solution is actually better: this is because the computation of the globally j-refined solution did not converge completely, a transient wave with a very long wavelength is still present. Both solutions are not as accurate as the

fine grid solution, because they lack the i- and k-resolution. But both show a clear improvement over the middle grid solution.

So as expected, j-refinement near the free surface is effective for improving the accuracy of the solution.

Free-surface j-refinement. In this test, the SURF middle grid is deformed, such that the upper grid planes are aligned with the NEPTUNE fine-grid free surface. For one grid, the j-spacing of the original grid is kept. For another, the j-spacing of a NEPTUNE grid is used, which has locally very fine cells near the free surface.

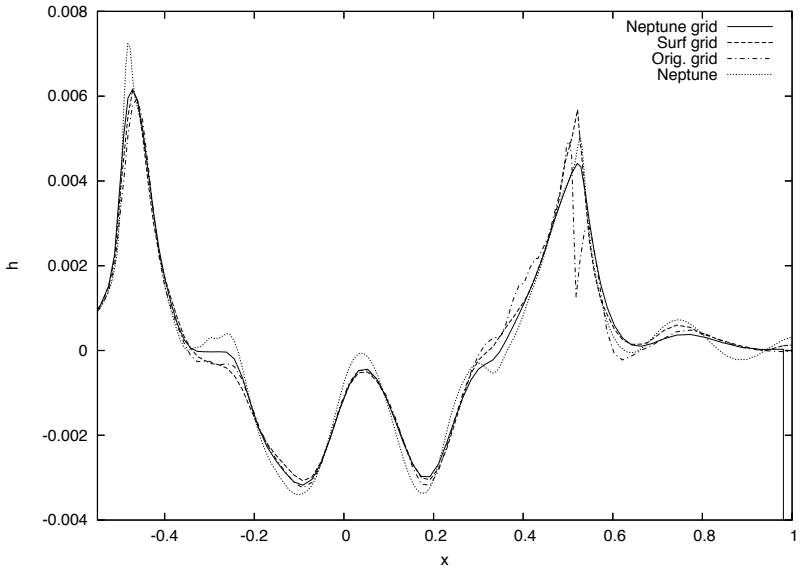


Figure 5.7. Wave profiles on the hull and centreline, for the deformed middle grids.

The original idea behind this experiment was, that the local alignment of the grid with the free surface would improve the solution by reducing the numerical diffusion in the level set equation. However, the wave profiles in figure 5.7 show that this does not happen: the solution with the original SURF grid spacing offers no improvement over the middle grid solution. This is different from the results in section 3.7.1, where a surface-aligned grid does improve the accuracy[1]. Therefore, the main surface errors for SURF are probably caused by the free-surface boundary conditions, rather than the numerical diffusion.

The solution on the NEPTUNE-spaced grid, on the other hand, is better than the middle-grid solution. The only explanation is that the accuracy of the level set solution and the free-surface boundary conditions is improved by the fine cells near the surface. Only, the band of fine cells in this grid is too narrow: behind the ship,

[1] Although they appear earlier in this thesis, the tests in section 3.7.1 were inspired by the SURF test and were performed about half a year later.

the free surface lies outside the fine cell band, the solution is worse there than on the original grid.

Concluding: a sufficiently wide band of fine cells around the free surface improves the accuracy of the solution. Grid deformation gives no better accuracy, but it can be used to form a narrower band of fine cells that still captures the surface.

5.2 Grid coarsening

In this section, our basic technique for constructing coarsened grids is explained. Our goal is to make locally refined grids, based on a standard unstructured data structure so they can be used directly in an unstructured solver, but resembling structured grids, as these give optimal accuracy and stability of solutions. We achieve this by using nested blocks of structured hexahedral cells, where the blocks with fine cells lie inside the coarse cell blocks. The blocks are connected with one cell thick shells of non-hexahedral unstructured cells. For stability, these unstructured cells cannot be too skewed; therefore, we must not make too abrupt changes from very fine to very coarse structured cells. Also, the shells of unstructured cells must be regular.

This is achieved by coarsening, per shell, in one direction only. So if a block of structured cells is finer in i- and k-direction than the surrounding block, then these blocks are connected with two unstructured shells, the second laying around the first.

The grids are generated by coarsening a very fine structured grid, that is made with a standard structured grid generator. This grid acts as a background, on which the unstructured grid is built. This has two advantages: one, the finest parts of the grid are smooth and the geometry of the hull is accurate (this is more difficult to achieve when a coarse grid is refined). And two: because of the structured background grid, the single-direction coarsening shells can be made in a very regular way, with all the unstructured cells having similar sizes and orientations.

Eventually, both hexahedral and non-hexahedral cells are stored in the same unstructured data structure. Directions like i and k are only used in the grid construction.

5.2.1 Procedure The coarsening procedure is as follows. It is started from the fine structured grid. First, hexahedral cells are defined in all the structured cells

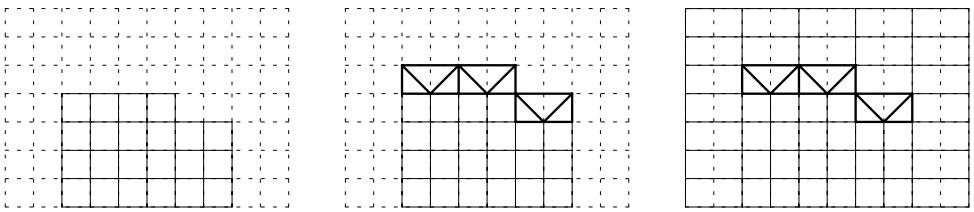


Figure 5.8. Fine cells (a), coarsening shell (b), and coarse cells (c). 2D cross-section.

where the finest grid is required. These cells have coarsening level $(1, 1, 1)$, i.e. they have the fine grid size in all directions (fig 5.8a). Around this block of cells, a coarsening shell is placed. In case of an i-coarsening for example, the hexahedral cells on the outside of the block will have coarsening level $(2, 1, 1)$, i.e. coarsened once in i-direction only.

To connect these cells with the $(1, 1, 1)$ cells inside, the block of fine cells is surrounded by one layer of $(2, 1, 1)$ cells (figure 5.8b). Then, these shell cells are divided into triangular cells, such that the faces of these cells match the faces of the fine and coarse hexahedrals around them. Thus, the outside of the shell now looks exactly as if it is filled completely with $(2, 1, 1)$ cells. Therefore, the rest of the domain can be filled with $(2, 1, 1)$ cells and the block fits in perfectly (figure 5.8c). Some fine cell faces already have the right size (like the side faces in figure 5.8): no coarsening cells are placed there.

The triangular cells are only used to fill hexahedral cells; they never stick out of their hexahedral boxes. This guarantees that the cells in the coarsening shell are regular. Also, all hexahedrals are subdivided in the same way, which adds to the regularity of the shell.

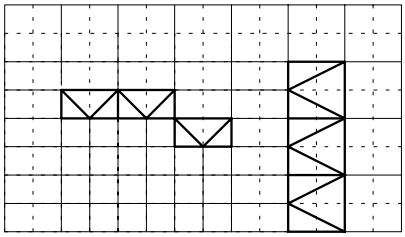


Figure 5.9. Two coarsening shells, 2D cross-section.

In most cases, the grid is not filled completely with the new coarse cells. Instead, some hexahedral cells are placed, then another coarsening layer is made (figure 5.9). If this is a k-coarsening, then the outside cells of that shell have coarsening level $(2, 1, 2)$. In this way, the whole fine structured grid is filled.

5.2.2 Filling cell planes Because the coarsening is so regular, the coarsening shells need not be generated in one piece. Instead, they can be generated in two cell-planes at the time. This is illustrated in figure 5.10, the 3D version of figure 5.8. In an i-coarsening, two planes of cells with constant i (the planes $i = i^*$ and $i = i^* - 1$) are merged into one plane, with coarsening cells. For each set of two planes, this merging can be done completely independently of all the other $i =$ constant planes, as the fine $(1, 1, 1)$ cells and the coarse $(2, 1, 1)$ cells have the same size of cell face in common with the other planes (figure 5.10). No matter where the coarsening cells are placed, the planes fit in with their neighbour planes.

This makes the generation of the coarsening shells easy and it increases the flexibility of the method: even the locations of the fine cells may be different in different planes. Thus, irregularly shaped blocks can be used, as seen in figure 5.8.

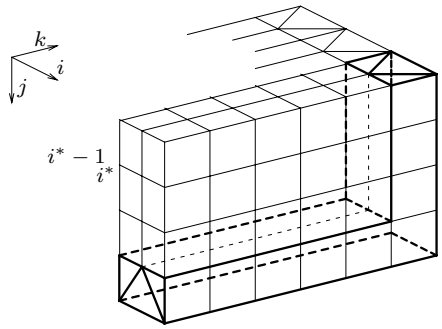


Figure 5.10. Two i = constant planes are merged.

Also, in the first and last columns in that figure, no coarsening shell is present at all because no fine cells exist in those planes. Even this can be done without any difficulty.

5.2.3 Coarsening cells Finally, we define the exact way in which the shell cells are filled with triangular cells. This distribution must be such that the cell fits in with all its neighbours. So for a normal shell cell (figure 5.11a), the front has two fine cell faces, while the back has one coarse cell face. The left and right side have one fine cell face, that fits in with the neighbouring planes. And the top and bottom are arbitrary, since they are connected with identical coarsening cells in the same shell; they fit as long as the top and bottom faces are equal.

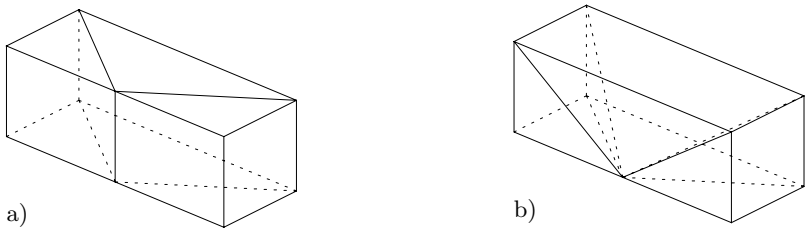


Figure 5.11. Coarsening cells, normal (a) and corner cell (b).

These outside faces are obtained by filling the hexahedral with three prism cells. These prisms have a great advantage: in one direction, they have no triangles. This is very useful near the free surface, as explained in the following section 5.3.

The coarsening shell can have corners, like the bottom corner in figure 5.10. In these corners, special cells are placed that are filled with four pyramid cells (figure 5.11b). These have the triangular faces on two adjacent hexahedral faces and coarse faces on the other two. Of course, both cells from figure 5.11 can be rotated to fit any desired orientation.

With the coarsening cells in figure 5.11, we can surround blocks of fine cells that form rectangles. All other shapes of fine-cell blocks have at least one corner that

points inwards into the block; these have to be filled with a corner cell that has two adjacent faces with triangles and two with fine cell faces. These could be filled with 15 pyramids. However, these pyramid cells can become very skewed. Therefore, we decided not to allow inward corners. So in each set of two planes to be merged, the fine cells that are surrounded by the shell must form a rectangle. This is the only restriction on the shape of the fine cell blocks.

5.3 Ship grid

To show how the grid coarsening from the previous section is used in practice, a grid is presented here for the Series 60 test case from section 5.1. The grid is constructed from the fine grid of section 5.1.1, it is coarsened to give a grid that is comparable in size with the middle grid.

Requirements and limitations. Middle-grid size cells are desired in the zone identified in section 5.1, i.e. in the region where the strongest waves appear. Near the bow, where the waves are high and where narrow secondary waves appear, even more refinement is useful.

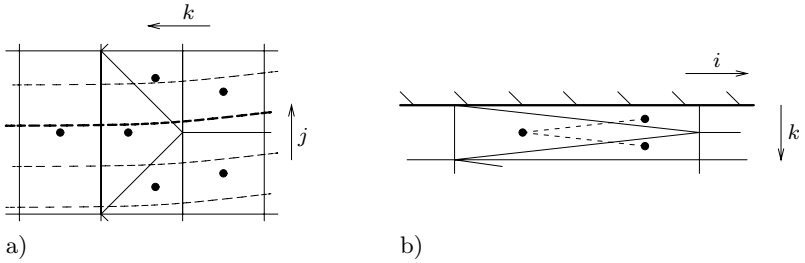


Figure 5.12. Forbidden coarsenings. j-coarsening at the free surface (a) gives large jumps in the level set values between neighbour cells (- - -: level set isolines). k-coarsening in the boundary layer (b) gives skewed cells (- - -: lines between cell centres).

Two factors pose limitations on the coarsening. The first comes from the free surface: the grid there must be structured in the direction normal to the surface. Unstructured cells would have different, irregular distances from the free surface to their cell centres, which means irregular cell values of the level set function (figure 5.12a). This causes large interpolation errors and instability. As a result, the prism coarsening cells cannot be used near the free surface, unless the prisms have their triangular faces in horizontal planes. Therefore, j-coarsening near the free surface is impossible (the j-direction is vertical there). We decided not to use j-coarsening whatsoever; instead, cells are clustered in j-direction near the free surface.

The second limitation is, that k-coarsening in the boundary layer is impossible (the k-direction is normal to the wall). The structured cells in the boundary layer have such high aspect ratios, that k-coarsening would produce highly skewed prism cells where the lines between cell centres are almost parallel to the cell faces, instead of normal to the faces (figure 5.12b). These cells cause instability. On the other hand, i-coarsening in the boundary layer is no problem.

These considerations define the i- and k-coarsening, described below.

Coarsening in i-, k-direction. When choosing the i, k coarsening sequence, the challenge is to get fine and coarse cells in the right locations, without placing coarsening cells where they are not acceptable.

We want the finest grid (level 1, the structured fine-grid size) in the bow region, near the free surface. But in most of the boundary layer, we accept the middle-grid spacing as fine enough. As k-coarsening in the boundary layer is not possible, this means that the whole boundary layer gets k-level 2.

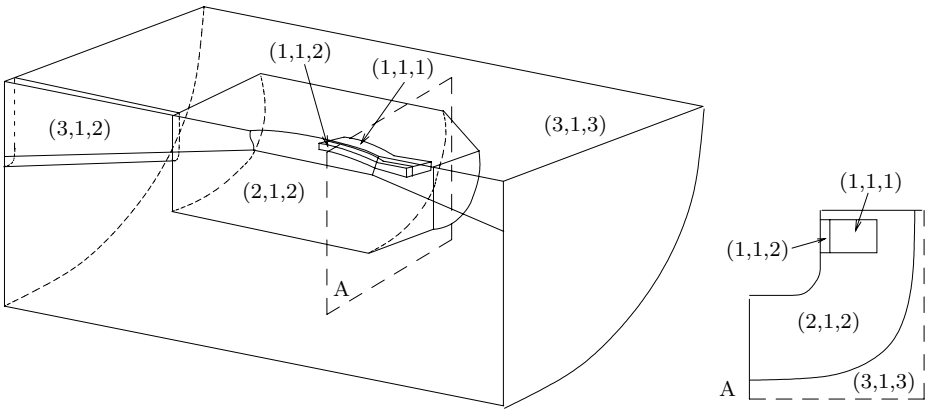


Figure 5.13. Fine and coarse blocks of cells: the whole grid and a cross-section of the finest part.

Therefore, we choose the finest grid (1,1,1) in a block near the bow, *outside* the boundary layer (figure 5.13). Since the structured grid is stretched in k-direction, these cells are still larger than the boundary layer cells. A k-coarsening shell is placed around this block, with coarsening cells on the forward, aft, upper and lower faces of the block. No coarsening cells are needed on the inboard face, in the boundary layer.

Then (1,1,2) hexahedrals are used to fill the boundary layer between the first block and the hull; this part of the boundary layer now has the desired k-level 2. Around these two blocks, an i-coarsening shell is placed with coarsening cells on the upper, outboard and lower faces. This shell extends into the boundary layer, but that is allowed for i-coarsening.

We are now at level (2,1,2), the middle grid size that is desired for most of the domain near the hull and the waves. A large block is filled with these cells, the block runs through all cells in j-direction. Near the front, the block is cut off at an oblique angle; this is allowed, as the cells in the planes to be merged still form rectangles.

Around this block, an i-coarsening and a k-coarsening are placed. Experiments showed that the boundary layer outside the block can be k-coarsened in front of the ship, where the flow is undisturbed. In the wake behind the block, k-coarsening is not possible. Therefore, i-coarsening is applied first, with prism cells on the

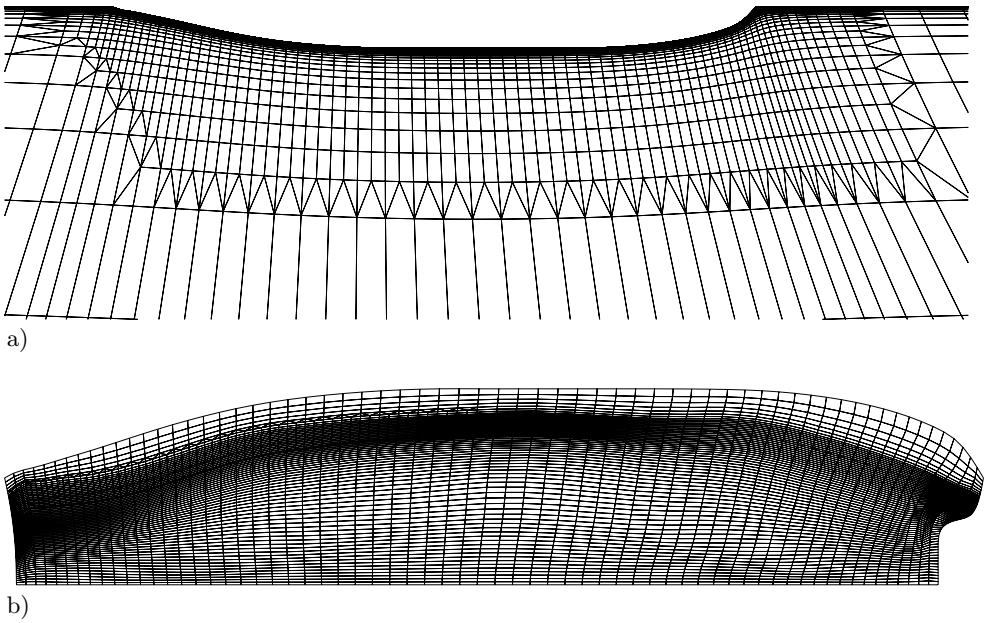


Figure 5.14. Adapted grid, near the free surface (a) and on the hull (b).

outside face of the block only. Then the aft part of the boundary layer is filled with (3,1,2) hexahedrals. Thus, the boundary layer is filled with three different sizes of hexahedrals, but they all have k-level 2. Finally, a k-coarsening is applied and the remaining part of the domain is filled with (3,1,3) cells.

In principle, it would be possible to fit another coarsening shell around the last one. But the reduction in cells from this shell is not much and it was found that the resulting very large cells decrease the stability of the solution. So this extra coarsening is not used.

Figure 5.14 shows the grid from above and on the hull. In the first figure, the outer shell can be seen. The second figure shows the i-refinement on the first block.

Clustering in j-direction. Since local j-coarsening at the free surface is not possible, j-coarsening is not used at all. Instead, cells are clustered in j-direction at the free surface. A uniform band of fine cells is placed around the free surface (see section 5.1.3). This band is deformed according to the free surface of a very coarse grid solution. This solution does not have to be accurate, so it can be computed very quickly in advance. It only has to capture the largest waves in the flow (bow, stern). Thus, it is possible to use a fine-cell band with a thickness that is smaller than the height of the bow wave, that still captures the free surface everywhere. This is impossible when the band is placed around $z = 0$. The adapted shape can be seen in figure 5.14.

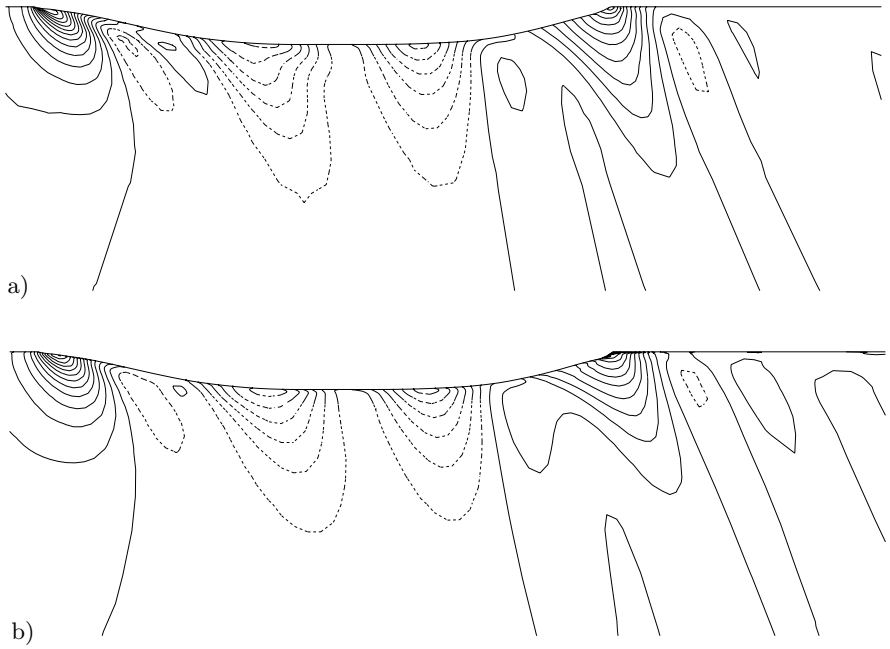


Figure 5.15. Free surface height for the coarsened solution (a), compared with the structured middle grid solution (b). Isoline spacing 0.0005.

Grid size. The finished grid has 282 998 cells, 277 758 nodes, and 843 859 faces. Of the cells, 251 781 are hexahedrals, 30 633 are prisms, and 584 are pyramids. The coarsening has reduced the number of cells to 21% of the fine grid cells, which is 171% of the middle grid cells. The grid is larger than the middle grid because it is not coarsened in j-direction.

5.4 Results

With the coarsened grid of the previous section, the test case of section 5.1 is computed; this solution is presented here and compared with the structured middle grid solution. The coarsened grid has more cells than the middle grid. Furthermore, it has the fine region near the bow and clustered cells near the interface. Therefore, the solution is expected to be more accurate than the middle grid solution.

It is indeed more accurate. The wave pattern (figure 5.15) has more detail in the shape of the waves and higher amplitudes near the back of the hull. Certainly in the bow region, where the grid is fine, the amount of detail has increased a lot; this shows that very local refinement can improve local accuracy. The wave profile on the hull (figure 5.16) has improved too. However, the effect of the coarse outer grid can be seen behind the hull: there, the wave amplitudes have become smaller. These waves are of little interest for the ship, so that is no problem.

The grid coarsening has an important side effect: the convergence of solutions

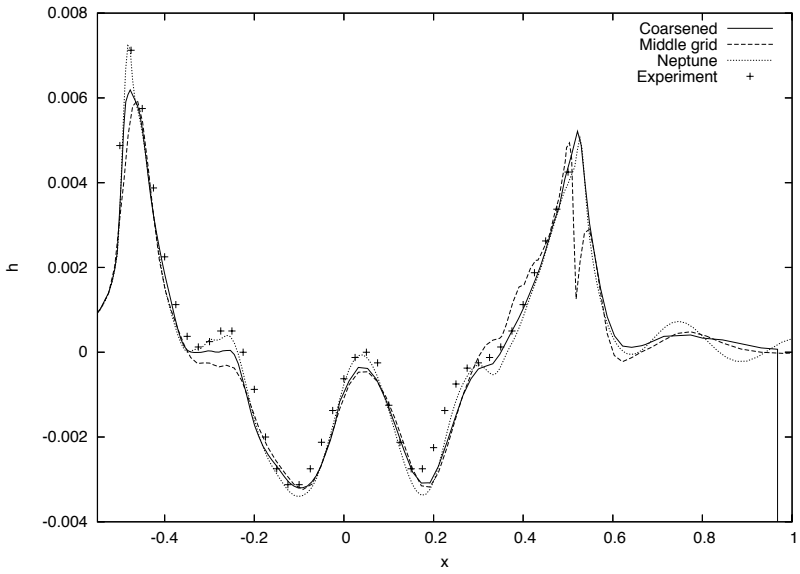


Figure 5.16. Wave profiles on the hull and centreline, coarsened grid. The profile is compared with NEPTUNE and with experimental data from [35].

on coarsened grids is much faster than on structured grids. Figure 5.17 shows the convergence of the drag for the coarsened and middle grid solution, for SURF (whose solver combines local time stepping with multigrid). This figure shows that some transients damp away quickly, but that one oscillating mode remains for a long time. This mode comes from a wave travelling between the front and the back of the domain; because of the coarse cells on the outside, this wave is damped much stronger on the coarsened grid. So the number of iterations needed for convergence is decreased at least 3 times. Because the grid is larger, the CPU time is reduced less, but it is still at least 2 times faster.

5.5 Conclusion

A method is presented for the construction of locally refined grids. The grids are unstructured, but are made to resemble structured grids as much as possible. The method is applied to the grid around a ship hull.

Tests on structured grids reveal that the errors in a ship flow solution with surface capturing are concentrated near the free surface, in the locations of the wave crests and troughs. The error is also large near the bow. Local refinement in these regions is almost as effective as global refinement for the improvement of the solution.

Smooth refined grids are made by coarsening a very fine grid. The coarsened grid consists of nested blocks of hexahedral cells, connected with shells of prism cells. By coarsening, per shell, in one direction only, the shells become very regular and can be constructed easily.

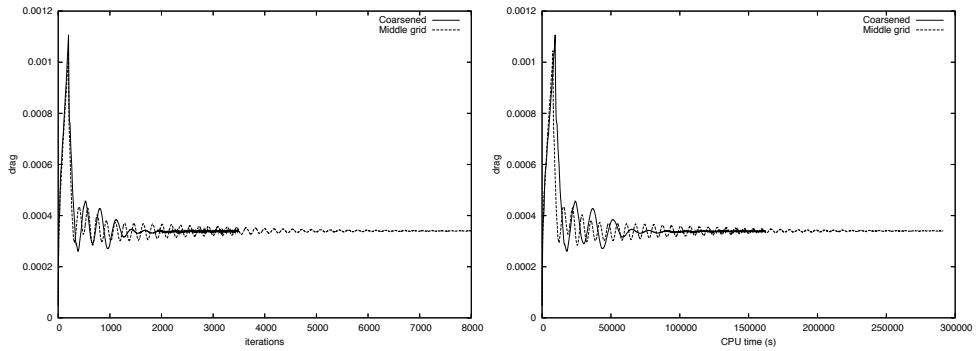


Figure 5.17. Convergence of the total drag, for the coarsened and the original middle grid solution. Drag against number of iterations (a) and against CPU time (b).

When constructing a ship grid, the free surface treatment makes j-coarsening near the surface impossible. Therefore, the grids are made with global cell clustering around the free surface and no j-coarsening. The shape of the fine cell region near the surface is adapted from a coarse-grid solution; thus, it can be placed effectively. Also, k-coarsening in the boundary layer leads to instability. Therefore, an i, k coarsening sequence is applied that gives no k-coarsening there.

Results show that coarsened grids, compared with structured grids, give better solutions in less computation time.

Outlook. The most important topic for further study is the development of coarsening criteria. The current work has produced a coarsening method, a technique for constructing locally coarsened grids once it is known where fine and coarse cells are desired. We also showed that local coarsening is effective for ship flows. However, the optimum locations for fine grids are not yet determined. The geometry in figure 5.13 is sensible. But is the fine grid needed in the whole wave domain? What is the optimum location and size of the fine (1,1,1) zone, and is it really efficient? Further experiments and experience in the use of the method will provide guidelines for the grid design.

This grid design can be partially automated. If a very coarse grid solution is computed in advance anyway, to deform the grid planes near the water surface, this solution can be used to set the coarsened zones too. With the locations of the largest waves known, the regions that cause the largest errors can be estimated. This estimation can be used when making the final grid.

Compressible two-fluid flow

In this last chapter, a model like the one in the first chapters is made for compressible two-fluid flow; the pure-fluid densities are no longer constant and energy equations plus equations of state have to be added to the model. On the other hand, friction and heat conduction are not modelled: the Euler equations are used. Most compressible two-fluid flows are unsteady, these flows appear mainly in explosions. Their main difficulty is the interaction of the interface between the two fluids with strong shock waves.

For compressible two-fluid flow, models exist that combine the standard Euler equations with one extra equation, which distinguishes between the two fluids. This can be, for example, a volume fraction equation or an equation for a parameter in a mixture equation of state. Since the compressible Euler equations have an energy equation, such systems consist of four differential equations in 1D; they are relatively simple. These models can produce large pressure errors near the interface (so-called pressure oscillations), that do not disappear when the grid is refined [1]. An alternative four-equation model is the ghost-fluid method [14], that uses a level set equation (section 1.2) to model the interface. While this method is pressure-oscillation free, it is not mass-conservative [33], which makes it less suitable for very strong shocks.

A different approach is, to use a full two-phase model that allows physical mixing of the fluids, for the non-mixing two-fluid flow. Two-phase models have different densities, pressures, and velocities for the two fluids. The seven-equation model by Baer and Nunziato [3] is well known, it is used among others by Abgrall and Saurel [2, 56]. These models give mass-conservative solutions without pressure oscillations. On the other hand, the large number of differential equations makes them expensive.

Kapila et al. [29] construct a five-equation model by taking the limit of the Baer and Nunziato model for equal fluid pressures and velocities. Their model has become popular, it is used for example in [18]. A disadvantage of this and similar models is, that their behaviour in shock waves is not well defined. An overview of these different models is given in [17].

We develop a physical concept model for, basically a way of thinking about, compressible two-fluid flow with equal fluid pressures and velocities. This concept gives an elegant and insightful derivation of a system of flow equations that is equivalent to Kapila's. Furthermore, it provides insight in the behaviour of the five-equation model in shocks. We present a general technique that could be used to model shock waves, and we derive shock relations for one specific case. While discretisations are not extensively discussed here, the model, like the ones described above, is intended for numerical methods.

The first section of this chapter presents the physical two-fluid model. Then section 6.2 uses this model to derive and analyse the five flow equations. Section 6.3 studies the model in a shock wave. And finally, section 6.4 presents two tests which

confirm that the model and the shock relations obtained are physically correct and give good agreement with experiments and analytical results.

6.1 Physical model for two-fluid flow

In this section, a physical model for compressible two-fluid flow without friction and heat conduction is developed. The model is based on the same idea as the incompressible model in section 2.1. The entire flow domain is filled with a mixture of the two fluids, but the fluids are not mixed on the molecular level: the 'mixture' consists of very small elements of the two pure fluids, arranged in an irregular pattern. So the fluid is a mixture in the macroscopic sense, but on a microscopic level the two fluids keep their own pure-fluid behaviour. The length scale of the fluid elements is much smaller than any other length scale in the model; thus, adjacent fluid elements are usually considered to lie in the *same* place.

In most parts of the domain, the quantity of one fluid is zero or very close to zero. The interface between the two fluids appears as a gradual transition from fluid 1 to fluid 2. In this way, the concept of an interface between the two fluids disappears from the model. The interface region is not fundamentally different from the rest of the flow, only the concentration of the two fluids changes faster there than in the rest of the domain. As we already saw in chapters 2–4, this is a great advantage numerically, since no special interface model is needed to act on the interface cells only.

Each fluid has its own density, but a single pressure and a single velocity are assumed for the two fluids. Thus, the fluid elements do not move relative to each other. In a numerical model, the two fluids may penetrate into each other, but by numerical diffusion only; they cannot convect into each other.

The fluid elements can interact by exerting forces on each other, to keep the velocity of the fluid elements equal. While the model has no friction, i.e. no other forces than pressure between two pieces of fluid lying some distance apart, the adjacent pure-fluid elements lie in the same location (see above), they are allowed to exchange any type of force. However, we do not allow heat conduction, neither in space nor between the two fluids in the same location (these are two different concepts!). As the two fluids in one location have different densities but the same pressure, they are not necessarily in thermal equilibrium. Physically, this is only possible when heat conduction is absent.

Our two-fluid model has five unknown state variables: pressure, velocity, two pure-fluid densities, and a measure for the relative amount of fluid 1. To successfully integrate this model in time, we need five differential equations. Four-equation models lose information about the flow during time integration; this may be a cause of the pressure oscillations.

6.2 Flow equations

Five differential equations for continuous two-fluid flow are derived and analysed in this section.

6.2.1 Differential equations To accurately solve problems with strong discontinuities, we strive for a model that is based on conservation laws as much as possible. In this section, suitable equations are selected. For convenience, we do this in 1D.

Like the incompressible Navier-Stokes equations (section 2.1), the Euler equations are derived without assumptions about the microscopic behaviour of the flow medium. Therefore, these equations are valid for the two-fluid model:,

$$(\rho)_t + (\rho u)_x = 0, \tag{6.1a}$$
$$(\rho u)_t + \left(\rho u^2 + p\right)_x = 0, \tag{6.1b}$$
$$(\rho E)_t + (\rho E u + p u)_x = 0. \tag{6.1c}$$

However, expressions are needed for the bulk quantities ρ and E, based on the states of the pure fluids 1 and 2. First α, the volume fraction of fluid 1, is chosen to distinguish between the fluids. The volume fraction can be used to define any bulk quantity; the bulk density ρ and bulk total energy E are:

$$\begin{aligned} \rho &= \alpha\rho_1 + (1-\alpha)\rho_2, \\ \rho E &= \alpha\rho_1 E_1 + (1-\alpha)\rho_2 E_2, \end{aligned} \tag{6.2}$$

with the total energy for each fluid defined as:

$$\begin{aligned} E_1 &= e_1 + \tfrac{1}{2}u^2, \\ E_2 &= e_2 + \tfrac{1}{2}u^2. \end{aligned} \tag{6.3}$$

The symbols e denote the internal energy of the fluids.

Two more conservation laws are needed to close the system. There are no more bulk conservation laws available, so the only option is to look for conserved quantities of one of the fluids. The first one is, of course, the conservation of mass for fluid 1:

$$(\rho_1\alpha)_t + (\rho_1 u\alpha)_x = 0. \tag{6.4}$$

Together with equation (6.1a), this equation gives mass conservation for both fluids.

For the last equation, only one option remains: an equation for the energy of fluid 1. Momentum for fluid 1, instead of energy, cannot be used: the bulk momentum is known and the two fluids move at the same speed, so we already know the momentum of fluid 1 (this is shown in detail in section 6.2.3). So the energy equation for fluid 1 is the only possible choice. This equation has a special property: the fluid elements exert forces on each other, so they *exchange* energy. This exchange appears as a source term in the energy equation:

$$(\rho_1 E_1\alpha)_t + (\rho_1 E_1 u\alpha + pu\alpha)_x = S. \tag{6.5}$$

The remainder of this chapter is largely devoted to this source term. Although, in a strict mathematical sense, equation (6.5) is not in conservation form, it is a genuine conservation law for $S = 0$ (i.e., everywhere outside the numerical mixture layer).

Summarising, the 1D system of equations is:

$$\boldsymbol{q}_t + \boldsymbol{f}_x = \boldsymbol{s}, \tag{6.6a}$$

with

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \\ \rho_1 \alpha \\ \rho_1 E_1 \alpha \end{pmatrix}, \qquad \boldsymbol{f} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u E + p u \\ \rho_1 u \alpha \\ \rho_1 E_1 u \alpha + p u \alpha \end{pmatrix}, \qquad \boldsymbol{s} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ S \end{pmatrix}. \tag{6.6b}$$

This system can be extended to more dimensions by adding a bulk-momentum conservation law for each additional dimension and fluxes in more directions.

6.2.2 Primitive variables To close the system (6.6), equations are needed for the thermodynamic behaviour of the fluid. These are the equations of state (EOS) for the two pure fluids. In their most general form, these equations are

$$\begin{aligned} e_1 &= e_1\left(\rho_1, p\right), \\ e_2 &= e_2\left(\rho_2, p\right). \end{aligned} \tag{6.7}$$

To keep the equations consistent with the physical model from section 6.1, no mixed-fluid equation of state is defined for the bulk fluid. The EOS (6.7) are valid in their respective fluid elements, bulk quantities can be defined as in equation (6.2).

The system of differential equations (6.6), combined with the equations of state (6.7) and the expression for S that is derived in the following section, is closed. It can be solved for the primitive variables, although this may require an iterative method for complex EOS. For simple EOS, like the ideal gas law and the stiffened gas EOS, explicit formulas can be found for the primitive variables.

6.2.3 The source term The source term S in the last equation of the system (6.6) has a physical meaning. In the energy equation for fluid 1, it represents the energy that is transferred from fluid 2 to fluid 1. In our flow model there is no heat conduction between pure-fluid elements, so the fluids can only exchange energy by means of work: the work generated by the forces on the interfaces between the microscopic fluid elements. This work can be split in two parts: the work generated by the inter-fluid forces acting in the flow direction and the expansion / compression work that appears when one of the fluids expands more than the other[1].

Inter-fluid force. There are two types of forces that act on the interfaces. The first are pressure forces, the second are a kind of friction forces. Regular friction forces occur in viscous flow in two or three dimensions, when fluid particles slide alongside each other. In inviscid flow, even in two-fluid flow, this force is absent. But another type of friction occurs in two-phase flow, where the two fluids have different velocities. When the pure fluids at the same location move through each other, they exert friction forces on each other. This friction does exist in our two-fluid flow model, albeit in a limit case. The two fluids have the same velocity, they can only maintain this if any velocity difference between the fluids causes inter-fluid

[1] In our early publications [71, 72], this second part is not yet taken into account.

friction forces that immediately make the velocities equal again. Thus, the friction forces provide instantaneous relaxation of velocity differences.

The requirement that the two fluids have the same velocity makes it possible to derive the work done by the inter-fluid forces. For this purpose, it is convenient to study the momentum equation for fluid 1 first:

$$(\rho_1 u\alpha)_t + \left(\rho_1 u^2\alpha + p\alpha\right)_x = S_M. \tag{6.8}$$

The source term S_M in this equation is the force exerted by fluid 2 on fluid 1. The equation offers no new time derivative information: u_t is known from equations (6.1b) and (6.1a), $(\rho_1\alpha)_t$ from equation (6.4). Instead, the equation defines S_M. While the expression for S_M can be found directly by eliminating the time derivatives from (6.8), we give another, physically insightful derivation of S_M.

Consider a small section of a 1D shock tube with height 1 (to simplify notation, the 1D tube has no real height), see figure 6.1. For convenience, all of fluid 1 has been lumped down at the bottom of this fluid element, but this lumped model is not really different from the microscopic-element flow model as discussed in section 6.1.

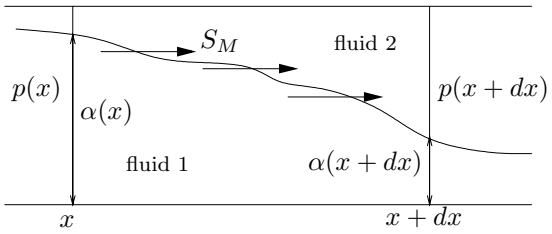


Figure 6.1. Flow element in 1D smooth flow.

The force on the entire fluid element is $p(x) - p(x+\Delta x)$. The bulk mass is $\bar{\rho}\Delta x$, with $\bar{\rho}$ the average density, so the acceleration of the entire element is:

$$u_t = \frac{p(x) - p(x+\Delta x)}{\bar{\rho}\Delta x}. \tag{6.9}$$

Now take the fluid 1 part of the element. The force on this element is $(p\alpha)(x) - (p\alpha)(x+\Delta x) + \overline{S_M}\Delta x$. The mass is $\overline{\rho_1\alpha}\Delta x$. The fluid velocities are equal, so the acceleration of this part is equal to the acceleration of the entire element:

$$\begin{aligned} \frac{p(x) - p(x+\Delta x)}{\bar{\rho}\Delta x} &= \frac{(p\alpha)(x) - (p\alpha)(x+\Delta x) + \overline{S_M}\Delta x}{\overline{\rho_1\alpha}\Delta x} \quad \Rightarrow \\ \frac{\overline{\rho_1\alpha}}{\bar{\rho}}\frac{p(x) - p(x+\Delta x)}{\Delta x} &= \frac{(p\alpha)(x) - (p\alpha)(x+\Delta x)}{\Delta x} + \overline{S_M}, \end{aligned}$$

and thus, in the limit for $\Delta x \downarrow 0$:

$$S_M = p\alpha_x + \alpha p_x - \frac{\rho_1\alpha}{\rho}p_x. \tag{6.10}$$

The first term $p\alpha_x$ expresses the pressure force on the interface: pressure times the projected height of the interface. The second and third term can be combined. By introducing β, the mass fraction of fluid 1,

$$\beta = \frac{\rho_1 \alpha}{\rho}, \tag{6.11}$$

this force is written as $(\alpha - \beta)p_x$. This is the friction force that keeps the velocities of the two fluids equal. For example, if for a bulk fluid $\alpha > \beta$ somewhere, then the mass of its fluid 1 there is low, but its volume is high. Therefore, fluid 1 receives a pressure force which would accelerate it faster than fluid 2, which is heavy and dense. So fluid 1 passes some of its pressure force to fluid 2, as friction. The opposite happens when $\alpha < \beta$. If the two fluids have the same density ($\alpha = \beta$), each fluid gets exactly enough pressure force for equal acceleration, so the friction force vanishes. And in pure contact discontinuities ($p_x = 0$), there is *no* acceleration, which means that the friction force vanishes too.

The contribution of this velocity relaxation force to the source term S in the energy equation (6.5) is the work done by the force S_M. The velocity of the interfaces is equal to the velocity u of the elements. Therefore the work is:

$$S_{u\text{-rel}} = S_M u = pu\alpha_x + (\alpha - \beta)\, up_x. \tag{6.12}$$

Expansion work. The work done by the velocity relaxation force is not the only energy exchange between the two fluids. There is also the pressure relaxation. Consider again a fluid element that moves with the flow, like in figure 6.1. When the volume of this element is changed, the pressure in the microscopic pure-fluid elements changes. The change in pressure with volume is not equal in the two fluids: one may be 'stiffer' than the other. So to keep the pressure in the two fluids equal, pressure relaxation occurs: the volume fraction changes. The stiffer gas gets a higher volume fraction on compression and a lower volume fraction on expansion. A change in the volume fraction means that the inter-fluid interfaces move; this movement creates pressure work.

The interface movement for the Lagrangian fluid element is given by the total derivative of the volume fraction:

$$\frac{D\alpha}{Dt} = \alpha_t + u\alpha_x.$$

Therefore, the expansion/compression inter-fluid work is:

$$S_{p\text{-rel}} = -p\alpha_t - pu\alpha_x. \tag{6.13}$$

Total work. The total energy exchange between the two fluids is found by adding equation (6.12) and (6.13). The source term becomes:

$$S = -p\alpha_t + (\alpha - \beta)up_x. \tag{6.14}$$

With the explicit expression (6.14) for S, the system (6.6) with (6.2), (6.3), and (6.7) is closed. The new physical information which the source term offers, that makes the energy equation (6.5) independent while the momentum equation (6.8) is not, is the specification of the heat transfer between the fluids. The kinetic energy of both fluids is fixed by the requirement that they have the same velocity, but the implicit specification in S that the heat transfer is zero (see section 6.1) is needed in the system to determine the internal energies of the two fluids.

6.2.4 Characteristic analysis A characteristic analysis like the one in section 2.2.2 is done for the system (6.6) to determine, among others, the sound speed. As our system in primitive form is equal to Kapila's [29], only the result is given here.

First, the conservative system is written in the primitive variables ρ, u, p, β, and α, and the derivatives are expanded to obtain five primitive flow equations. The derivatives of the internal energy are found by rewriting (6.2) as:

$$e = \beta e_1 + (1-\beta)e_2, \tag{6.15}$$

and substituting the general EOS (6.7). This equation can be differentiated with respect to t and x by using equation (6.11) and the chain rule. For example:

$$\begin{aligned} e_t = &\left(e_1 - e_2 + \frac{\beta\rho}{\alpha}e_{1_{\rho_1}} + \frac{(1-\beta)\rho}{1-\alpha}e_{2_{\rho_2}}\right)\beta_t + \left(\beta e_{1_p} + (1-\beta)e_{2_p}\right)p_t \\ &+ \left(\frac{\beta^2}{\alpha}e_{1_{\rho_1}} + \frac{(1-\beta)^2}{1-\alpha}e_{2_{\rho_2}}\right)\rho_t + \left(-\frac{\beta^2\rho}{\alpha^2}e_{1_{\rho_1}} + \frac{(1-\beta)^2\rho}{(1-\alpha)^2}e_{2_{\rho_2}}\right)\alpha_t. \end{aligned} \tag{6.16}$$

The derivatives of e_1 are found in a similar way. The primitive flow equations are, for $\boldsymbol{q} = [\rho, u, p, \beta, \alpha]$:

$$\boldsymbol{q}_t + \begin{pmatrix} u & \rho & 0 & 0 & 0 \\ 0 & u & \frac{1}{\rho} & 0 & 0 \\ 0 & \rho c^2 & u & 0 & 0 \\ 0 & 0 & 0 & u & 0 \\ 0 & A_u & 0 & 0 & u \end{pmatrix} \boldsymbol{q}_x = 0, \tag{6.17}$$

with the sound speed c defined by:

$$\rho c^2 = \frac{1}{\frac{\beta e_{1_p}}{\frac{p}{\rho} - \frac{\beta^2\rho}{\alpha^2}e_{1_{\rho_1}}} + \frac{(1-\beta)e_{2_p}}{\frac{p}{\rho} - \frac{(1-\beta)^2\rho}{(1-\alpha)^2}e_{2_{\rho_2}}}}, \tag{6.18}$$

and A_u by:

$$A_u = \frac{p\left(\alpha(1-\beta)e_{2_p} - \beta(1-\alpha)e_{1_p}\right) - \rho^2\beta(1-\beta)\left(\frac{\beta}{\alpha}e_{1_{\rho_1}}e_{2_p} - \frac{1-\beta}{1-\alpha}e_{2_{\rho_2}}e_{1_p}\right)}{p\left(\beta e_{1_p} + (1-\beta)e_{2_p}\right) - \rho^2\beta(1-\beta)\left(\frac{\beta}{\alpha^2}e_{1_{\rho_1}}e_{2_p} + \frac{1-\beta}{(1-\alpha)^2}e_{2_{\rho_2}}e_{1_p}\right)}. \tag{6.19}$$

The Jacobian of the primitive flow equations has eigenvalues:

$$\lambda_1 = u - c, \quad \lambda_{2,3,4} = u, \quad \lambda_5 = u + c, \tag{6.20}$$

and corresponding Riemann invariants:

$$dJ_1 = dp + \rho c\, du, \;\; dJ_2 = dp - c^2\, d\rho, \;\; J_3 = \beta, \;\; dJ_4 = d\alpha - \frac{A_u}{\rho c^2} dp, \;\; dJ_5 = dp - \rho c\, du. \tag{6.21}$$

The fifth equation in (6.17) is the original fifth equation in the Kapila system, a convection equation with source term for α. For continuous flow, it can be used to reformulate the energy exchange source term (6.14) by eliminating the time derivative for α.

6.3 Discontinuous flow

The system of differential equations (6.6) is not valid in discontinuities, where the derivatives of the state variables are not defined. To allow discontinuities in the solution, a well-posed weak formulation of the system is needed, with a jump condition that describes the flow behaviour over a discontinuity. Unlike the single-fluid Euler equations, most two-fluid models do not have an exact jump condition; they need approximations in shocks.

Using the concept of section 6.1, we suggest a way in which the formulation of exact jump conditions for our model may be possible. For a special case, we present an exact jump condition.

For any conservation law, the jump condition is the well-known Rankine-Hugoniot condition, which is found by integrating the conservation law over a control volume around the discontinuity:

$$[\boldsymbol{f}] = c_s\, [\boldsymbol{q}]\,, \tag{6.22a}$$

with [ ] denoting the jump over the discontinuity ($[\boldsymbol{q}] = \boldsymbol{q}_R - \boldsymbol{q}_L$ etc.), c_s is the speed of the discontinuity. The first four equations in (6.6) have jump conditions of this form. But when the fifth equation is integrated over a discontinuity, the contribution of the source term leads to an extra integral:

$$[\boldsymbol{f}] = c_s\, [\boldsymbol{q}] + \int_{x_L}^{x_R} S\, dx. \tag{6.22b}$$

With the spatial derivatives in S not defined over the discontinuity, this integral cannot always be evaluated. For a contact discontinuity, where u and p are constant, its value is $pu[\alpha]$[2]. But for a shock, the integral has no value. Mostly, this problem is avoided by assuming an approximate internal structure for the shock and integrating over this shape.

However, this approximation is not strictly necessary; if we properly define the weak solution, the integral in (6.22b) gets a well-defined value. We say that a flow

[2] When u and p are constant, the pressure relaxation $S_{p\text{-rel}}$ is zero, while the velocity relaxation $S_{u\text{-rel}}$ reduces to the pressure work $pu\alpha_x$. Thus, $\int_{x_L}^{x_R} S\, dx = pu \int_{x_L}^{x_R} \alpha_x\, dx$. (See section 6.2.3.)

is a weak solution of the system (6.6) if it is the inviscid limit of solutions of (6.6) with viscous effects and heat conduction added. This is the normal definition of weak solutions for single-fluid flow; it guarantees that the weak solution contains no entropy-violating expansion shocks. But the definition has another important implication: it guarantees that 'discontinuities' have *some* internal structure. Hence, the integration of S through a shock is possible.

In this internal structure, viscous effects and heat conduction cannot be ignored. Typically, the thickness of a shock wave scales to lowest order with the viscosity, so the velocity gradients in the shock, which are proportional to the inverse of the shock thickness, scale with the inverse of the viscosity. Thus, the viscous forces, proportional to viscosity times velocity gradient, have terms that are independent of the viscosity. The result is that the viscous effects in the shock do not vanish when the viscosity goes to zero.

The idea that we pursue here is to take the full flow equations in the shock, including viscosity and heat conduction, and *eliminate* the dependence on the space coordinate, such that we get relations between the flow states on the two sides of the shock that are independent of its precise internal structure. Thus, we get shock relations that function like the Rankine-Hugoniot relations for single-fluid flow.

6.3.1 General viscous shock Here, we show how to derive a set of ordinary differential equations that can be integrated through a shock. In a flow with viscosity and heat conduction, a shock wave is studied in a coordinate system attached to the shock. If the shape of the shock is constant, this flow is steady. Conservation laws can be applied to a control volume that partially lies *in* the shock and partially outside (figure 6.2). The control volume lies between the inflow state L and the shock (this is an arbitrary choice, we can also integrate from the right).

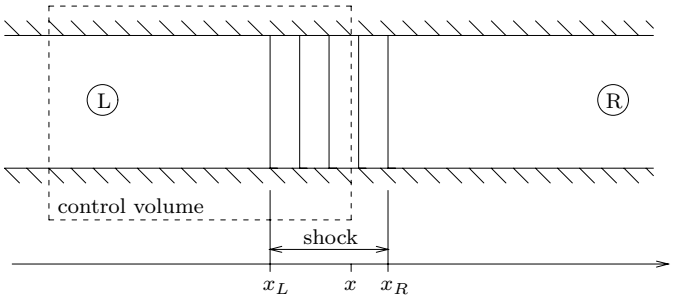


Figure 6.2. Control volume for integration across a shock with internal structure.

The conservation laws (6.6) with viscous terms and heat conduction are integrated over this control volume. Since the mixture fluid moves as a whole, the viscous friction τ is proportional to the velocity derivative $\tilde{u}_x$, where $\tilde{u}$ denotes the velocity in the shock-fixed coordinate system. However, when the two pure fluids have different viscosities, the friction forces in the fluids may be different. The heat conduction Q is complicated, because the fluids are not in thermal equilibrium. We shall not yet attempt to model τ and Q. With the extra terms, integration of (6.6)

gives:

$$\rho\tilde{u} = \rho_L \tilde{u}_L \qquad \text{(mass)}, \qquad (6.23a)$$

$$p - \tau + \rho\tilde{u}^2 = p_L + \rho_L \tilde{u}_L^2 \qquad \text{(mom.)}, \qquad (6.23b)$$

$$\rho\tilde{u}\left(e + \tfrac{1}{2}\tilde{u}^2\right) + (p-\tau)\tilde{u} - Q = \rho_L \tilde{u}_L \left(e_L + \tfrac{1}{2}\tilde{u}_L^2\right) + p_L \tilde{u}_L \qquad \text{(en.)}, \qquad (6.23c)$$

$$\beta = \beta_L \qquad \text{(ma. fl.1)}, \qquad (6.23d)$$

$$\rho\tilde{u}\beta\left(e_1 + \tfrac{1}{2}\tilde{u}^2\right) + (p-\tau_1)\alpha\tilde{u} - Q_1\alpha = \rho_L \tilde{u}_L \beta_L \left(e_{1,L} + \tfrac{1}{2}\tilde{u}_L^2\right)$$
$$+ p_L \alpha_L \tilde{u}_L + \int_{x_L}^{x} S_s \, dx \qquad \text{(en. fl.1)}, \qquad (6.23e)$$

where $[\rho, \tilde{u}, p, \alpha, \beta]$ is the state at x in the shock (see figure 6.2). The first four equations are pure conservation laws, the energy equation for fluid 1 has the integral over the shock source term S_s. Compared to S, this term also includes viscous effects and inter-fluid heat conduction. The equations (6.23) are only valid in shocks; they are meaningless in a contact discontinuity, where $\tilde{u} = 0$. Of course, (6.23d) does not hold in a contact discontinuity.

The equations (6.23) give the five state variables as a function of x. But we are not interested in the precise shape of the shock, only in the relation between the states on the two sides of the shock. If that relation is independent of the shock thickness, then it is also valid in the limit when this thickness goes to zero. The following is a conjecture on how the shock structure can be eliminated. The idea is to write all state variables, not as a function of x, but of the velocity $\tilde{u}$. The change in velocity over a shock remains constant, even when the viscosity approaches zero. So if the shock has a smooth internal structure, the state as a function of $\tilde{u}$ does not steepen or become discontinuous when the viscosity vanishes. We assume that the friction and heat conduction expressions contain first-order spatial derivatives of the states. Then the procedure is:

1. Eliminate the x-derivatives in $\tau_1\alpha\tilde{u}$ and $Q_1\alpha$ from the left-hand side of equation (6.23e), with equations (6.23b) and (6.23c). Eliminate the second-order x-derivatives in S_s, if they appear, with the x-derivatives of these equations.

2. Differentiate the resulting equation with respect to x, to get rid of the integral. The derivative of e_1 is found as in (6.16). Because of the previous step, this equation contains only first-order spatial derivatives.

3. In all the spatial derivatives in (6.23b), (6.23c), and the rewritten (6.23e), make a change of variables: $(\cdot)_x = (\cdot)_{\tilde{u}}\tilde{u}_x$.

4. Rewrite these equations to get separate equations for $p_{\tilde{u}}$, $\alpha_{\tilde{u}}$, and $\tilde{u}_x$.

The result is two equations, for $p_{\tilde{u}}$ and $\alpha_{\tilde{u}}$, that do not depend on x. Together with (6.23a) and (6.23d), they define p, α, ρ, and β as functions of $\tilde{u}$. The state $\boldsymbol{q}_R$ at the right of the shock wave is that state for which $\tilde{u}_x = 0$, as given by the third equation.

6.3.2 Friction only In the general case above, the state jump over the shock depends on the model chosen for the friction and the heat conduction. When $\tau_1 \alpha \tilde{u}$ in equation (6.23e) is eliminated with $\tau \tilde{u}$ in (6.23c), the result depends at least on the ratio τ_1/τ. Also, the heat conduction Q_1 and Q are not equal in general, and the heat conduction between the fluids in S is not equal to either of them. The resulting equations may contain ratios of the heat conductions, or even of Q and τ.

This suggests simplifications to the viscous model, that can be made to get a shock relation that is independent of the viscous and heat conduction parameters.

1. No heat conduction, $Q = 0$. Physically, this represents the limit case where heat conduction is a much slower process than viscous friction and can be ignored in the shock.

2. While the viscosity of the mixture fluid may depend on the state vector, the viscosity in the two fluids, at the same location, is identical. Thus the added viscous force is similar to single-fluid friction:

$$\tau = \tau_1 = \mu(\boldsymbol{q})\tilde{u}_x. \tag{6.24}$$

Under these assumptions, all viscous parameters disappear from the shock relations. In general, the last assumption is physically not correct. However, tests in the next section show that the resulting shock relations accurately model shock behaviour.

Viscous source term. Given these assumptions, we study the energy exchange in a small part of the viscous shock. With a momentum balance like in section 6.2.3, it is found that:

$$S_{M_s} = (p - \mu\tilde{u}_x)\alpha_x + (\alpha - \beta)(p - \mu\tilde{u}_x)_x. \tag{6.25}$$

For the pressure relaxation work, it is reasonable to assume that the term $p - \mu\tilde{u}_x$ is now the normal stress that should be used in the work equation:

$$S_{(p\text{-rel})_s} = -(p - \mu\tilde{u}_x)(\alpha_t + \tilde{u}\alpha_x). \tag{6.26}$$

Because the flow is steady, the α_t term is zero. No heat conduction is present, so the complete source term is:

$$S_s = (\alpha - \beta)\tilde{u}(p - \mu\tilde{u}_x)_x. \tag{6.27}$$

Elimination of internal structure. First, $\tilde{u}_x$ (the only term directly dependent on x) is eliminated from the energy equations by subtracting $\tilde{u}$ times equation (6.23b) from (6.23c) and $\tilde{u}\alpha$ times (6.23b) from (6.23e). The source term (6.27) is rewritten by noticing that:

$$(p - \mu\tilde{u}_x)_x = -(\rho\tilde{u}^2)_x = -\rho_L\tilde{u}_L\tilde{u}_x,$$

and by making a change of variables in the integral. The result is:

$$\rho_L\tilde{u}_L\left(e - \tfrac{1}{2}\tilde{u}^2 + \tilde{u}_L\tilde{u}\right) + p_L\tilde{u} = \rho_L\tilde{u}_L\left(e_L + \tfrac{1}{2}\tilde{u}_L^2\right) + p_L\tilde{u}_L, \tag{6.28a}$$

$$\rho_L\tilde{u}_L\left(\beta_L\left(e_1 + \tfrac{1}{2}\tilde{u}^2\right) + \alpha\left(\tilde{u}_L\tilde{u} - \tilde{u}^2\right)\right) + p_L\alpha\tilde{u} + \rho_L\tilde{u}_L\int_{\tilde{u}_L}^{\tilde{u}}(\alpha - \beta_L)\tilde{u}\,d\tilde{u} =$$
$$\rho_L\tilde{u}_L\beta_L\left(e_{1,L} + \tfrac{1}{2}\tilde{u}_L^2\right) + p_L\alpha_L\tilde{u}_L. \tag{6.28b}$$

To eliminate the integral, both these expressions are differentiated with respect to $\tilde{u}$. The chain rule and the general EOS (6.7) are used.

$$\begin{aligned}
&\rho_L\tilde{u}_L\left(\left(\beta_L e_{1p}+(1-\beta_L)e_{2p}\right)p_{\tilde{u}}+\left(-\frac{\beta_L^2\rho}{\alpha^2}e_{1\rho_1}+\frac{(1-\beta_L)^2\rho}{(1-\alpha)^2}e_{2\rho_2}\right)\alpha_{\tilde{u}}\right.\\
&\qquad\left.+\left(\frac{\beta_L^2}{\alpha}e_{1\rho_1}+\frac{(1-\beta_L)^2}{1-\alpha}e_{2\rho_2}\right)\rho_{\tilde{u}}+\tilde{u}_L-\tilde{u}\right)+p_L=0.\\
&\rho_L\tilde{u}_L\left(\beta_L e_{1p}p_{\tilde{u}}+\left(-\frac{\beta_L^2\rho}{\alpha^2}e_{1\rho_1}-\tilde{u}^2+\tilde{u}_L\tilde{u}\right)\alpha_{\tilde{u}}+\frac{\beta_L^2}{\alpha}e_{1\rho_1}\rho_{\tilde{u}}+\alpha(\tilde{u}_L-\tilde{u})\right)\\
&\qquad+p_L\alpha+p_L\tilde{u}\alpha_{\tilde{u}}=0.
\end{aligned}$$

Now we substitute $\partial\rho/\partial\tilde{u}=-\rho_L\tilde{u}_L/\tilde{u}^2$ and eliminate $p_{\tilde{u}}$. A lot of rewriting gives:

$$\alpha_{\tilde{u}}=\frac{\left[\begin{array}{r}\rho_L\tilde{u}_L\Bigg(-\beta_L(1-\beta_L)\frac{e_{1\rho_1}e_{2p}\frac{\beta_L}{\alpha}-e_{2\rho_2}e_{1p}\frac{1-\beta_L}{1-\alpha}}{\beta_L e_{1p}+(1-\beta_L)e_{2p}}\frac{\rho_L\tilde{u}_L}{\tilde{u}^2}+\alpha(\tilde{u}_L-\tilde{u})\\ +\left(\tilde{u}-\tilde{u}_L-\frac{p_L}{\rho_L\tilde{u}_L}\right)\frac{\beta_L e_{1p}}{\beta_L e_{1p}+(1-\beta_L)e_{2p}}\Bigg)+p_L\alpha\end{array}\right]}{\rho_L\tilde{u}_L\left(-\beta_L(1-\beta_L)\frac{e_{1\rho_1}e_{2p}\frac{\beta_L}{\alpha^2}+e_{2\rho_2}e_{1p}\frac{1-\beta_L}{(1-\alpha)^2}}{\beta_L e_{1p}+(1-\beta_L)e_{2p}}\frac{\rho_L\tilde{u}_L}{\tilde{u}}+\tilde{u}(\tilde{u}_L-\tilde{u})\right)+p_L\tilde{u}}. \tag{6.29}$$

This is an ordinary differential equation that gives $\alpha_{\tilde{u}}$ in terms of α and $\tilde{u}$ only. So implicitly, it defines α as a function of $\tilde{u}$. The pressure can be found directly from equation (6.28a), the density and mass fraction from (6.23a) and (6.23d). The result is a series of four equations, coupling the five state variables in the shock, that are *independent* of x and of μ: these relations are unique for a given left state $\boldsymbol{q}_L$.

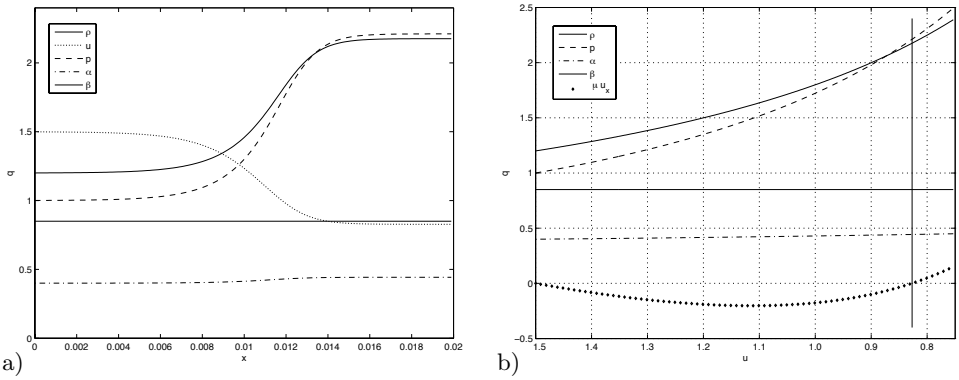


Figure 6.3. Example of a two-fluid viscous shock (ideal gases). Internal structure for $\mu=1.5\cdot10^{-3}$ (a) and state as a function of $\tilde{u}$, valid for all μ (b). The vertical line indicates the right state, where $\tilde{u}_x=0$.

These four equations give a Rankine-Hugoniot relation that connects the states to the left and to the right of the shock, which is independent of x and μ as well.

The right state is found by substituting $\tilde{u}$, $p(\tilde{u})$ and $\rho(\tilde{u})$ into equation (6.23b) and finding that value of $\tilde{u}$ for which $\tilde{u}_x = 0$:

$$\mu\tilde{u}_x = 0 = p + \rho\tilde{u}^2 - p_L - \rho_L\tilde{u}_L^2 \qquad \Leftrightarrow \qquad \boldsymbol{q} = \boldsymbol{q}_R. \tag{6.30}$$

Equation (6.29) has to be integrated numerically; an example is found in figure 6.3.

Concluding: under the above assumptions, there is a unique Rankine-Hugoniot condition that connects the inflow and outflow states in a two-fluid shock.

6.4 Test cases

This section shows two tests for our two-fluid model. In the first one, the shock relations from the previous section are tested by comparing them with experiments on shocks in mixtures. In the second test, a finite-volume numerical shock tube solver based on our model is used to compute 1D two-fluid flows.

Shocks in alloys. This test studies shocks in physical mixtures, that satisfy the assumptions of section 6.1: equal fluid pressures and velocities. Although our model is meant for two-fluid flows where the fluids mix only in the numerically diffused interface, the mixture model can compute shocks in such physical mixtures. Few experimental data are available for these shocks, because the kind of mixture on which our model is based does not often appear in nature. Metal alloys are an example. Two mixed metals do not really blend up to atom level, but one metal forms small elements in the other. The alloy can thus be seen as a two-fluid mixture and for strong shocks, it behaves like a compressible fluid. Experimental shock speed data for alloys are available [42].

Four alloys are studied here, uranium–rhodium, copper–zinc (brass), epoxy–spinel (not a metal alloy, probably spinel powder in epoxy), and gold–germanium. Properties of these alloys are given in table 6.1. The equation of state which is used for the pure metals is the stiffened gas EOS,

$$p = (\gamma - 1)\rho e - \gamma\pi. \tag{6.31}$$

For this equation, the stiff-gas parameters γ and π for the pure materials are needed. These are determined by fitting single-fluid shock relations to the experimental curves for the pure materials as found in [42]; the results are given in table 6.2.

Table 6.1. Material properties for four alloys (from Marsh [42]). These properties are for $p = 10^5$ Pa, α and β refer to the first material.

Alloy	α	β	ρ (kg/m^3)
Epoxy–Spinel	0.600	0.328	2171
Uranium–Rhodium	0.809	0.890	17204
Brass	0.710	0.754	8406
Gold–Germanium	0.736	0.909	15536

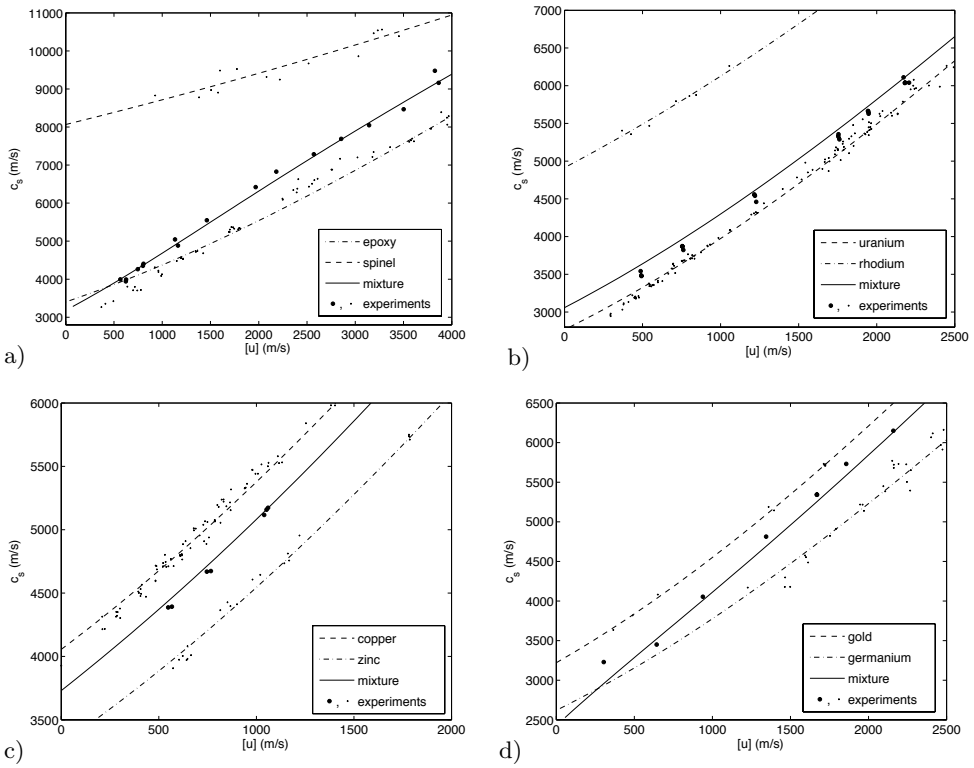


Figure 6.4. Shock speed – velocity curves found with the mixture shock relations from chapter 6.3.2, compared with experiments. For reference, the pure-fluid curves are plotted too.

The single-fluid shock-speed relation for the stiffened gas equation is:

$$c_s = \frac{\gamma+1}{4}[u] + \sqrt{\left(\frac{\gamma+1}{4}[u]\right)^2 + \gamma\frac{p_{pre}+\pi}{\rho_{pre}}}. \tag{6.32}$$

where c_s is the shock speed and $[u]$ the velocity jump over the shock.

Mixture shock relations are computed with the equations from section 6.3.2, based on the experimentally fitted EOS for the pure materials. These curves are compared with measurements of the relation between shock speed and material speed for the alloys. The tests are done with the alloys in initial conditions as given in table 6.1, with an initial pressure $p = 10^5$ Pa.

From figure 6.4, we see that the correspondence of the mixture-fluid curves with the experiments is very good for all four alloys. The fit is as good as the stiffened gas approximations to the pure-fluid measurements. This suggests that the concept of integration through shocks, as presented in section 6.3, is correct. It also shows that the assumptions in section 6.3.2 are reasonable here. This means either that the effects of the heat conduction and different fluid viscosities on a shock are generally

Table 6.2. Stiff-gas parameters for the six pure materials (derived from Marsh [42]).

Material	ρ (kg/m^3)	γ	π (GPa)
Epoxy	1185	2.45	5.6
Spinel	3622	1.50	157.1
Uranium	18930	3.10	46.8
Rhodium	12492	3.37	89.4
Copper	8924	3.63	40.4
Zinc	7139	3.27	23.9
Gold	19204	3.54	56.3
Germanium	5328	2.92	12.5

small enough to ignore, or that the assumptions (slow heat conduction, similar material viscosities) model the physics of alloys accurately.

Shock tube simulation. In the second test, the compressible flow equations are used in a 1D time-dependent numerical flow solver. Results are given here, together with a short description of the solver. More details on similar methods are found in [34, 71, 72].

The solver uses uniform grids. The fluxes are found with the limited reconstruction (4.1), using the ϕ_κ limiter (4.2). The flux function is an exact Riemann solver (see e.g. [62]), that uses the shock relations from section 6.3.2 for shocks and numerical integration of the Riemann invariants (section 6.2.4) for expansions. The Riemann solver uses two-variable Newton-Raphson to find the strengths of the left and right pressure wave. Explicit time integration is used with a two-step scheme as described in [70]. For the time integration, the source term is combined with the fluxes to give the time derivative in the fifth equation. The source term in each cell is computed in two parts: the energy exchange in the interior of the cells and in the waves of the Riemann solver at the cell faces.

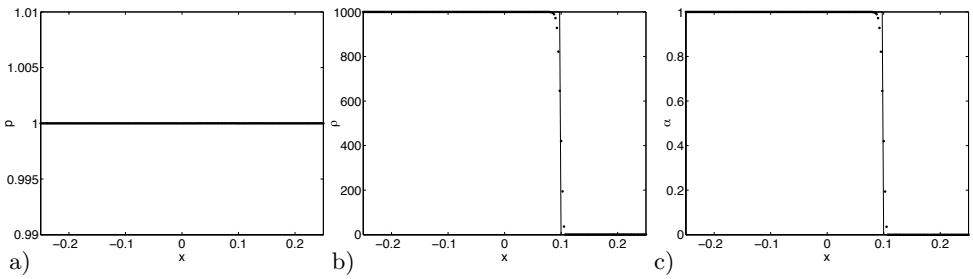


Figure 6.5. 1D contact discontinuity. $p = 1$, $u = 1$, $\rho_L = 1000$, $\rho_R = 1$, $\gamma_L = 1.4$ and $\gamma_R = 1.6$. The grid has 200 cells, 80 time steps and $\Delta t/\Delta x = 0.5$ (CFL = 0.5). Solid lines: exact solution.

The solver is tested on three Riemann problems: shock tube flows where the initial conditions to the left and to the right of $x = 0$ are constant. The EOS is ideal

gas (equation (6.31) with $\pi = 0$). The first test is a strong contact discontinuity with a 1000:1 density ratio (figure 6.5). This test shows that the flow model is pressure-oscillation free: the pressure over the contact discontinuity is constant.

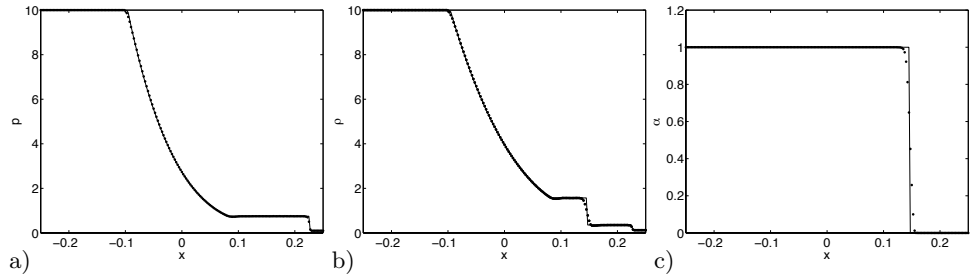


Figure 6.6. High-pressure, two-fluid Sod problem. $(\rho, u, p)_L = (10, 0, 10)$, $(\rho, u, p)_R = (0.125, 0, 0.1)$, $\gamma_L = 1.4$ and $\gamma_R = 1.6$. The grid has 200 cells, 160 time steps, $\Delta t/\Delta x = 0.2$ (CFL = 0.56).

The second problem (figure 6.6) is a two-fluid version of the Sod problem [62], with a 100:1 pressure ratio. It consists of a shock, the central two-fluid interface, and an expansion fan. Each feature is represented well by the model. The shock and the interface are in the right position, the pressure is constant over the interface, and the volume fraction is constant in both pressure waves.

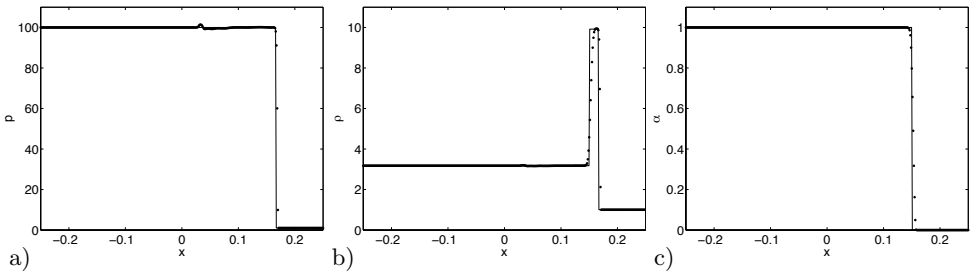


Figure 6.7. Strong shock hitting an interface, without creating a reflection wave. $(\rho, u, p)_L = (3.1748, 9.4350, 100)$, $(\rho, u, p)_R = (1, 0, 1)$, $\gamma_L = 1.667$ and $\gamma_R = 1.2$. The grid has 400 cells, 320 time steps, $\Delta t/\Delta x = 0.04$ (CFL = 0.42).

The final test is the hardest. It models a strong shock that hits a two-fluid interface; the conditions are such that no pressure wave is reflected. The problem is solved well (figure 6.7), despite the difficulty that the shock wave and the interface stay close together. The only visible error is the wiggle in the pressure near $x = 0.05$. This may be caused by a small inaccuracy in the source term modelling; it could also be a feature of the finite-volume discretisation, as single-fluid problems show similar errors. The wiggle disappears on grid refinement.

Concluding: Both these shock tube tests and the alloy tests confirm the validity of the two-fluid model.

6.5 Conclusion

A physical model is presented for compressible inviscid two-fluid flow, in which the interface is modelled as a mixture region. In this mixture, small elements of the two pure fluids have their own densities, but the same pressure and velocity. The elements of the two fluids can exert forces on each other, but not exchange heat.

Based on this physical idea, a five-equation model is derived. This model is equivalent to the existing Kapila five-equation model that is obtained by taking a limit of a more complex two-phase model. However, the current physical mixture concept suggests the energy equation for one of the fluids, rather than a modified volume fraction convection equation, as the fifth equation in the system. This formulation is helpful for the construction of numerical methods and it allows the study of the model behaviour in shock waves.

In a shock wave, flow equations are used with viscous friction and heat conduction; we argue that these may be rewritten such that the explicit dependence on the spatial coordinate disappears from the equations. Then these equations are also valid in the limit case where the viscosity goes to zero. By assuming that heat conduction is slower than viscous friction and can be ignored, and that the local viscosities of the fluids are equal, a system of shock relations is derived that depends neither on the shock thickness nor on the viscosity: it is therefore a possible shock model for the two-fluid system.

With this shock model, shock speeds are computed in metal alloys. These show excellent agreement with measurements. A finite-volume shock tube solver, constructed with the two-fluid model, accurately computes two-fluid Riemann problems. These two tests show that the model is sensible for both continuous and discontinuous flow.

Outlook. The ideas and the model developed in this chapter can be used to formulate accurate numerical methods for difficult two-fluid flow problems. Our two-fluid model, especially the fifth equation with its physically meaningful energy equation source term, can help with this development. The main challenge related to numerical methods is to find an approximate Riemann solver, that is more efficient than the very expensive exact solver used in section 6.4, but that still agrees with the physical two-fluid model.

An open question is the development of shock relations when both viscous friction and heat conduction are taken into account. The basic analysis in section 6.3.1 shows that it may be possible to derive such relations; this depends on the choice of the model for the friction and the heat conduction. Then, the question is if such a complex shock model gives different results from the simpler model in section 6.3.2. Despite the assumptions about the viscosity and the heat conduction, this model gives such excellent agreement with the experimental data in the alloy shock test, that a more complex model can hardly bring any real improvement there. The study of complex shock models may reveal that the simple model already contains all the relevant physics.

Conclusion

Conclusions and suggestions for future work have been given at the end of each chapter. The main findings are summarised here.

7.1 Conclusions

Surface capturing and multigrid. Efficient solution of a surface capturing model for steady water flow is possible. Chapter 2 presents a surface capturing discretisation without interface reconstruction, where the flow is computed both in the water and in the air above it; the distinction between water and air is made with a mass conservation equation for the water. The result is a volume-of-fluid formulation that consists of conservation laws only, which can be solved in a coupled way. Therefore, the system is suitable for multigrid solution.

The model is discretised with a finite-volume method that uses linearised Riemann solver fluxes: this gives a stable discretisation at all Reynolds numbers and the possibility to use Gauss-Seidel smoothing. Fourier analysis shows that point Gauss-Seidel smoothing is inefficient unless sufficient physical viscosity is present, while line Gauss-Seidel is effective for all flow states. Furthermore, it is not much more expensive than point smoothing, so line smoothing is used. The system can be solved with standard multigrid, except that the source terms for the coarse grid operators must be made small. This is done by scaling the defects coming from the fine grids, while unscaling the coarse grid corrections, and by starting each coarse grid correction from the converged solutions on the coarse grids.

Results are computed for a channel flow with a bottom bump. While these indicate the need for a turbulence model and a second-order accurate discretisation, they show that the multigrid solver is effective for the capturing model.

In chapter 3 a new multigrid method is presented, that is effective when a RANS turbulence model is added to the system. Also with this model, coupled solution of all equations is essential for efficiency. The turbulence model causes two problems. First, the model contains a source term which can make the discretisation unstable and the Gauss-Seidel smoothing ineffective. Close to a steady solution, the model is stable, so standard Gauss-Seidel can be used. But for the first iterations, a local damping is added to the smoother. The amount of damping is set automatically, based on the convergence of the Newton-Raphson solver for the individual lines.

And second, the turbulence model cannot resolve boundary layers well on coarse grids, so large differences appear between coarse and fine grid solutions. This makes normal coarse grid corrections ineffective. The solution is to switch to linear Galerkin coarse grid operators. The linearisation of the fine grid operator, that is available from the line smoothing, is copied directly to the coarse grids. The water surface too is resolved differently on coarse and fine grids; it is thicker on coarse grids. Therefore, the new multigrid method is also effective for the capturing model.

Results show that the convergence rates of the linear multigrid method, even with the turbulence model, are the same as for the laminar model in chapter 2. The boundary layers in the test problems are now modelled correctly: this improves the computed wave shapes significantly. Good agreement with experiments is obtained.

The capturing method is completed in chapter 4, where the discretisation is made second-order accurate. A limited reconstruction of the state variables is used, to guarantee stability and monotonicity of the solution. The solution for the volume fraction of water is a smeared discontinuity only; a standard second-order accurate limiter does not give the best results for this variable. Instead, an antidiffusive limiter is used that compresses the water surface layer. As opposed to unsteady flow, such a limiter can never deform a steady surface much; furthermore, a monotone limiter guarantees that the volume fraction is between 0 and 1. Therefore, complicated limiters and correction procedures are not needed. We use a limiter that depends on the interface direction, determined from the velocity vector. This limiter keeps the interface both sharp and smooth.

The second-order discretisation is solved with defect correction. In a few steps, the first-order accurate solutions are greatly improved. As in chapter 2, the defects are scaled and the corrections unscaled. Numerical tests show that this defect correction is not yet robust in all cases. But when it works, the results are excellent. Solutions are essentially converged in less than 10 defect correction steps, grid convergence is reached on realistically sized grids, and agreement with experiments and existing numerical solutions is good. The compressive limiter keeps the thickness of the water surface constant at $3-4$ cells. Thus, our now complete surface capturing method can solve problems both accurately and very fast.

Ship grids. In chapter 5, a method is presented for the construction of locally refined grids around ship hulls. Tests with a level set code on structured grids show that the main errors in a ship flow solution are concentrated in the wave region near the water surface, and near the pressure peak at the bow. Tests with structured, refined grids show that local refinement in these zones is as effective for the reduction of errors as global refinement. Very local refinement at the surface also proves effective.

The refined grids are made such, that they can be used in a standard unstructured data structure, but contain as many structured parts as possible. The grids consist of nested blocks of hexahedral cells, connected by unstructured cells. To keep even this connection as structured as possible, it is made by dividing layers of structured cells into prisms. The grid is constructed by coarsening a very fine structured grid.

In a test, a coarsened grid gave a better solution than a structured reference grid. Furthermore, the coarse cells at the outer boundaries cause rapid damping of transient waves; although the coarsened grid contains more cells than the reference grid, the solution was computed more than 2.5 times faster.

Compressible flow. In the final chapter 6, a model like the one in the first chapters is developed for compressible two-fluid Euler flow. To get physically correct solutions

without pressure oscillations, two flow equations for one of the fluids must be added to the standard Euler equations. Apart from the mass conservation equation, an energy equation for one of the fluids is added; this equation has a source term that models the energy exchange between the fluids. This source term contains two terms: the work done by the inter-fluid force that keeps the velocity of the fluids equal and the expansion / compression work caused by changes in the volume fraction. The system is equivalent to existing methods, but the energy equation formulation gives physical insight and guidelines for developing numerical methods.

By considering solutions with discontinuities as the limit case of flows with viscosity, the model behaviour over shocks can be studied. The conservation laws, in a viscous and heat conducting shock, can be rewritten with the velocity as the independent variable, instead of the spatial coordinate. Thus, the spatial shock structure can be eliminated from the equations. These equations contain viscosity parameters; when the heat conduction is assumed to be zero and the viscosity is locally uniform, these parameters are eliminated too. A test on metal alloy shock speeds shows that the resulting model gives excellent shock predictions. Tests with a numerical shock tube solver further confirm the validity of the method.

7.2 Future work

Surface capturing and multigrid. The most important step in the further development of the water flow method is the extension to three dimensions. The present method is ready for this; technically, there is nothing that prevents the addition of a third dimension. Adding extra fluxes, and an extra tangential velocity in the Riemann solver, is straightforward: only the Peyret control volume for the diffusion operator may require attention. The finite-difference stencils for the turbulent source term can also be extended.

For the line smoother, a third smoothing direction must be added, but the solution of the individual lines and the damping for the turbulence model remain the same. It will be interesting to see if the good smoother performance is kept in 3D.

Finally, multigrid is not intrinsically 2D, the restrictions and prolongations can be made three-dimensional and for the linear multigrid, seven-cell stencils will be restricted. Doubtlessly, unexpected problems will appear when the third dimension is added. But the method is ready for those problems to be discovered.

Another crucial question is the modelling of turbulence at the water surface, as already discussed in section 3.8. If situations are computed where the water surface touches a wall, then turbulence appears in the water–air mixture region. As a first approximation, the turbulence model could be modified such, that it does not notice the mixture at all. The result would be similar to turbulence in a surface fitting model, or a capturing model with free-surface boundary conditions. The resulting model can already compute flows where the water surface interacts with walls, which is of course necessary for ship flow computations.

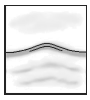
A much more challenging approach is to develop an approximate steady model for the physical unsteady phenomena at the water surface, like breaking waves and foam. Such a model would increase the accuracy and the robustness of the method,

since it would not have convergence problems due to breaking waves.

The major remaining question in the current research is the robustness of the defect correction iteration. At this moment, the iteration often diverges due to instabilities in the air region. But, as already noted in section 4.5, defect correction is stable for single-fluid flow and for the water region. So it is expected that a damping or another fix can be found that makes the air region behave the same as the water region, thus stabilising the defect correction iteration.

Ship grids. As the technique of the grid coarsening in chapter 5 is completely developed, the main question is how it should be used in practice. The technique is very flexible and can be used for many different types of ship flow, even for other flows where the good properties of both structured and unstructured grids are needed. For all these flows, coarsening criteria and guidelines for the grid block structure must be investigated. Both mathematical analysis and rules of thumb, based on experience, can provide these.

Compressible flow. Finally, chapter 6 gives ideas, rather than precise results. The analysis of continuous flow has produced no really new model, but the physical insight may help people who work on similar models. The shock analysis raises questions: the viscous shock model leads, in general, to shock relations that are dependent on viscous parameters. Then, making assumptions, these parameters are removed. And then, in the alloy shock test, this simple model gives near-perfect shock speed prediction. If this holds in general, the shock model without the viscous parameters already contains all the shock physics. Thus, a more detailed study of the shock model is worthwhile.



Bibliography

[1] R. Abgrall and S. Karni. Computations of compressible multifluids. *J. Comp. Phys.*, 169:594–623, 2001.

[2] R. Abgrall and R. Saurel. Discrete equations for physical and numerical compressible multiphase mixtures. *J. Comp. Phys.*, 186:361–396, 2003.

[3] M. R. Baer and J. W. Nunziato. A two-phase mixture theory for the deflagration-to-detonation transition (DDT) in reactive granular materials. *Int. J. Multiphase Flows*, 12:861–889, 1986.

[4] E. H. v. Brummelen. Numerical solution of steady free-surface Navier-Stokes flow. Report MAS-R0018, CWI, Amsterdam, June 2000.

[5] E. H. v. Brummelen. *Numerical Methods for Steady Viscous Free-Surface Flows*. PhD thesis, Universiteit van Amsterdam, Amsterdam, Feb. 2002.

[6] J. Cahouet. Etude numérique et expérimentale du problème bidimensionnel de la résistance de vagues non-linéaire. Rapport de recherche ENSTA n^o 185, Ecole Nationale Supérieure de Techniques Avancées, Paris, Jan. 1984.

[7] G. Carré. An implicit multigrid method by agglomeration applied to turbulent flows. *Computers & Fluids*, 26(3):299–320, 1997.

[8] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *J. Comp. Phys.*, 2:12–26, 1967.

[9] J.-A. Désidéri and P. W. Hemker. Convergence analysis of the defect-correction iteration for hyperbolic problems. *SIAM J. Sci. Comput.*, 16(1):88–118, 1995.

[10] E. Dick. A flux-vector splitting method for steady Navier-Stokes equations. *Int. J. Num. Meth. Fluids*, 8:317–326, 1988.

[11] E. Dick. A multigrid method for steady incompressible Navier-Stokes equations based on partial flux splitting. *Int. J. Num. Meth. Fluids*, 9:113–120, 1989.

[12] E. Dick and J. Linden. A multigrid method for steady incompressible Navier-Stokes equations based on flux difference splitting. *Int. J. Num. Meth. Fluids*, 14:1311–1323, 1992.

[13] J. H. Duncan. The breaking and non-breaking wave resistance of a two-dimensional hydrofoil. *J. Fluid Mech.*, 126:507–520, 1983.

[14] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the Ghost Fluid Method). *J. Comp. Phys.*, 152:457–492, 1999.

[15] C. Frohn-Schauf. *Flux-Splitting-Methoden und Mehrgitterverfahren für hyperbolische Systeme mit Beispielen aus der Strömungsmechanik*. Number 211 in Berichte der Gesellschaft für Mathematik und Datenverarbeitung. R. Oldenbourg Verlag, München; Wien, 1993.

[16] D. Greaves. A quadtree adaptive method for simulating fluid flows with moving interfaces. *J. Comp. Phys.*, 194:35–56, 2004.

[17] H. Guillard and M. Labois. Numerical modelling of compressible two-phase flows. In P. Wesseling, E. Oñate, and J. Périaux, editors, *Proceedings of EC-COMAS CFD 2006*, Egmond aan Zee, 2006.

[18] H. Guillard and A. Murrone. A five equation reduced model for compressible two phase flow problems. Rapport de recherche N^o 4778, INRIA, Sophia Antipolis, Mar. 2003.

[19] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer-Verlag, Berlin, 1985.

[20] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, Dec. 1965.

[21] A. Hay, P. Queutey, and M. Visonneau. Computation of three-dimensional free-surface flows with an automatic adaptive mesh refinement and coarsening strategy. In *Proceedings of the 25th Symposium on Naval Hydrodynamics*, St. Johns, Newfoundland and Labrador, Aug. 2004.

[22] P. W. Hemker and S. P. Spekreijse. Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations. *Appl. Num. Math.*, 2:475–493, 1986.

[23] T. Hino. A 3D unstructured grid method for incompressible viscous flows. *J. of the Society of Naval Architects, Japan*, 182:9–15, 1997.

[24] T. Hino. An interface capturing method for free surface flow computations on unstructured grids. *J. of the Society of Naval Architects, Japan*, 186:177–183, 1999.

[25] T. Hino, editor. *CFD Workshop Tokyo 2005*, Tokyo, 2005.

[26] N. Hirata and T. Hino. An efficient algorithm for simulating free-surface turbulent flows around an advancing ship. *J. of the Society of Naval Architects, Japan*, 185:1–8, 1999.

[27] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comp. Phys.*, 39:201–225, 1981.

[28] W. Hundsdorfer, B. Koren, M. v. Loon, and J. G. Verwer. A positive finite-difference advection scheme. *J. Comp. Phys.*, 117:35–46, 1995.

[29] A. K. Kapila, R. Menikoff, J. B. Bdzil, S. F. Son, and D. S. Stewart. Two-phase modeling of deflagration-to-detonation transition in granular materials: Reduced equations. *Physics of Fluids*, 13(10):3002–3024, 2001.

[30] K. M. T. Kleefsman. *Water Impact Loading on Offshore Structures*. PhD thesis, Rijksuniversiteit Groningen, Groningen, Nov. 2005.

[31] B. Koren. *Multigrid and Defect Correction for the Steady Navier-Stokes Equations, Application to Aerodynamics*. Number 74 in CWI Tracts. CWI, Amsterdam, 1985.

[32] B. Koren. A robust upwind discretization method for advection, diffusion and source terms. In C. B. Vreugdenhil and B. Koren, editors, *Numerical Methods for Advection-Diffusion Problems*, volume 45 of *Notes on Numerical Fluid Mechanics*, pages 117–138, Braunschweig, 1993. Vieweg.

[33] B. Koren, M. R. Lewis, E. H. v. Brummelen, and B. v. Leer. Riemann-problem and level-set approaches for homentropic two-fluid flow computations. *J. Comp. Phys.*, 181:654–674, 2002.

[34] J. J. Kreeft. Unsteady compressible two-fluid flow model for interface capturing. On the dynamics of a shock-bubble interaction. Report to appear, CWI, Amsterdam, 2007.

[35] Y. Kusaka. Report of ALM on the 18th ITTC cooperative experiments – conducted at two similar towing tanks with different size. Technical report, ALM (Mitsui Zosen), 1986.

[36] B. v. Leer. Upwind-difference methods for aerodynamic problems governed by the Euler equations. In *Lectures in Applied Mathematics*, volume 22, part 2, pages 327–336, Providence, RI, 1985. American Mathematical Society.

[37] B. Leonard. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Comp. Meth. Appl. Mech. Eng.*, 19:59–98, 1979.

[38] B. Leonard. Simple high-accuracy resolution program for convective modelling of discontinuities. *Int. J. Num. Meth. Fluids*, 8:1291–1318, 1988.

[39] M. R. Lewis. *Numerical Methods for Water Flows with Free-Surface Gravity Waves*. PhD thesis, Technische Universiteit Delft, Delft, June 2004.

[40] F. Liu and X. Zheng. A strongly coupled time-marching method for solving the Navier-Stokes and $k-\omega$ turbulence model equations with multigrid. *J. Comp. Phys.*, 128:289–300, 1996.

[41] K. J. Maki, A. Iafrati, S. H. Rhee, R. F. Beck, and A. W. Troesch. The transom-stern modeled as a backward facing step. In *Proceedings of the 26th ONR Symposium on Naval Hydrodynamics*, Rome, 2006.

[42] S. P. Marsh. *LASL Shock Hugoniot Data*. Univ. of California Press, Berkeley, 1980.

[43] D. J. Mavriplis. Algebraic turbulence modeling for unstructured and adaptive meshes. *AIAA Journal*, 29(12):2086–2093, 1991.

[44] D. J. Mavriplis and V. Venkatakrishnan. Agglomeration multigrid for two-dimensional viscous flows. *J. Comp. Phys.*, 24:553–570, 1995.

[45] F. R. Menter. Zonal two-equation $k-\omega$ turbulence models for aerodynamic flows. AIAA Paper 93-2906, American Institute of Aeronautics and Astronautics, July 1993.

[46] F. R. Menter. Eddy viscosity transport equations and their relation to the $k-\epsilon$ model. *J. Fluids Engineering*, 119:876–884, 1997.

[47] B. Mohammadi and O. Pironneau. *Analysis of the $k-\varepsilon$ Turbulence Model*.

John Wiley & Sons, Ltd., Chichester, 1994.

[48] W. Mulder, S. Osher, and J. A. Sethian. Computing interface motion in compressible gas dynamics. *J. Comp. Phys.*, 100:209–228, 1992.

[49] A. Nakayama. Characteristics of the flow around conventional and supercritical airfoils. *J. Fluid Mech.*, 160:155–179, 1985.

[50] K. Nerinckx, J. Vierendeels, and E. Dick. Mach-uniformity through the coupled pressure and temperature correction algorithm. *J. Comp. Phys.*, 206:597–623, 2005.

[51] S. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[52] H. C. Raven, A. v. d. Ploeg, and B. Starke. Computation of free-surface viscous flows at model and full scale by a steady iterative approach. In *Proceedings of the 25th Symposium on Naval Hydrodynamics*, St. Johns, Newfoundland and Labrador, Aug. 2004.

[53] H. C. Raven and B. Starke. Efficient methods to compute steady ship viscous flow with free surface. In *Proceedings of the 24th Symposium on Naval Hydrodynamics*, Fukuoka, July 2002.

[54] W. J. Rider and D. B. Kohte. Reconstructing volume tracking. *J. Comp. Phys.*, 141:112–152, 1998.

[55] A. E. Samuel and P. N. Joubert. A boundary layer developing in an increasingly adverse pressure gradient. *J. Fluid Mech.*, 66:481–505, 1974.

[56] R. Saurel and R. Abgrall. A multiphase Godunov method for compressible multifluid and multiphase flows. *J. Comp. Phys.*, 150:425–467, 1999.

[57] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale*, 1:5–21, 1994.

[58] S. P. Spekreijse. Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws. *Math. Comput.*, 49:135–155, 1987.

[59] J. Steelant, E. Dick, and S. Pattijn. Analysis of robust multigrid methods for steady viscous low Mach number flows. *J. Comp. Phys.*, 136:603–628, 1997.

[60] J. Stoker. *Water Waves*. Interscience Publishers Inc., New York, 1957.

[61] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Num. Anal.*, 21:995–1011, 1984.

[62] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer-Verlag, Berlin, 1997.

[63] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001.

[64] O. Ubbink and R. I. Issa. A method for capturing sharp fluid interfaces on arbitrary meshes. *J. Comp. Phys.*, 153:26–50, 1999.

[65] J.-M. Vanden-Broeck. Nonlinear stern waves. *J. Fluid Mech.*, 96(3):603–611, 1980.

[66] A. E. P. Veldman, K. M. T. Kleefsman, and G. Fekken. Numerical computation of hydrodynamic wave impact. In P. Bergan, J. García, E. Oñate, and T. Kvamsdal, editors, *Proceedings of the International Conference on Computational Methods in Marine Engineering*, Oslo, 2005.

[67] J. Vierendeels, K. Riemslagh, and E. Dick. A multigrid semi-implicit line-method for viscous incompressible and low-Mach-number flows on high aspect ratio grids. *J. Comp. Phys.*, 154:310–341, 1999.

[68] M. Visonneau, P. Queutey, and G. B. Deng. Numerical and physical aspects of model and full-scale free-surface ship flow computations. In P. Bergan, J. García, E. Oñate, and T. Kvamsdal, editors, *Proceedings of the International Conference on Computational Methods in Marine Engineering*, Oslo, 2005.

[69] M. Vogt and L. Larsson. Level set methods for predicting viscous free-surface flows. In *Proceedings of the 7th International Conference on Numerical Ship Hydrodynamics*, Nantes, July 1999.

[70] J. Wackers and B. Koren. A simple and efficient space-time adaptive grid technique for unsteady compressible flows. AIAA Paper 2003-3825, American Institute of Aeronautics and Astronautics, July 2003.

[71] J. Wackers and B. Koren. A fully conservative model for compressible two-fluid flow. In P. Neittaanmäki, T. Rossi, K. Majava, and O. Pironneau, editors, *Proceedings of ECCOMAS 2004*, Jyväskylä, 2004.

[72] J. Wackers and B. Koren. A fully conservative model for compressible two-fluid flow. *Int. J. Num. Meth. Fluids*, 47:1337–1343, 2005.

[73] J. Wackers and B. Koren. A multigrid method for the computation of steady water waves. In P. Wesseling, C. W. Oosterlee, and P. W. Hemker, editors, *Proceedings of the ECCOMAS Conference on Multigrid, Multilevel and Multiscale Methods*, Scheveningen, 2005.

[74] J. Wackers and B. Koren. A surface capturing method for the efficient computation of steady water waves. In P. Bergan, J. García, E. Oñate, and T. Kvamsdal, editors, *Proceedings of the International Conference on Computational Methods in Marine Engineering*, Oslo, 2005.

[75] J. Wackers and B. Koren. Accurate and efficient computation of steady water flow with surface waves and turbulence. In P. Wesseling, E. Oñate, and J. Périaux, editors, *Proceedings of ECCOMAS CFD 2006*, Egmond aan Zee, 2006.

[76] J. Wackers and B. Koren. Efficient computation of steady water flow with waves. In P. Bergan, J. García, E. Oñate, and T. Kvamsdal, editors, *Proceedings of the International Conference on Computational Methods in Marine Engineering*, Barcelona, 2007.

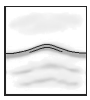
[77] J. Wackers and B. Koren. A surface capturing method for the efficient computation of steady water waves. To appear in *J. Comp. Appl. Math.*

[78] J. Wackers and B. Koren. Efficient computation of steady water flow with waves. To appear in *Int. J. Num. Meth. Fluids*.

[79] J. Wackers and B. Koren. Multigrid solution method for the steady RANS equations. To appear in *J. Comp. Phys.*

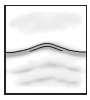
[80] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons Ltd., Chichester, 1992.

[81] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer-Verlag, Berlin Heidelberg, 2001.



Index

References are grouped according to their main term. Bold numbers denote a page where a definition of the reference is given.



Samenvatting

Surface capturing en multirooster voor stationaire waterstromingen met vrij oppervlak – Jeroen Wackers

Surface capturing ('oppervlak oplossen') is een techniek voor het modelleren van het wateroppervlak in numerieke berekeningen van waterstroming. Hierbij loopt het rekenrooster door tot boven het wateroppervlak en wordt aan de stromingsvergelijkingen een apart oppervlaktemodel toegevoegd, dat de plaats van het wateroppervlak in het rooster weergeeft. Voorbeelden zijn de level-set- ('isoverzameling') en volume-of-fluid- ('vloeistofvolume') methodes. Voor de berekening van bijvoorbeeld de stroming rond schepen hebben surface-capturingmethodes het voordeel dat zij algemeen toepasbaar zijn en ingewikkelde scheepsgeometrieën aankunnen. Voor stationaire stromingsproblemen is het grote nadeel van veel capturingmethodes dat zij geen snelle oplostechnieken toelaten. Meestal worden deze stromingen opgelost door het tijdstappen van de instationaire stromingsvergelijkingen tot convergentie.

Dit proefschrift laat zien dat snelle oplossing van een surface-capturingmethode mogelijk is. Hiertoe wordt een mengselmodel gebruikt: de stroming wordt zowel in het water als in de lucht erboven berekend en de overgang tussen lucht en water is een numeriek mengsel dat is uitgesmeerd over een aantal cellen. Het onderscheid tussen lucht en water wordt gemaakt met een massabehoudsvergelijking voor het water; het stromingsmodel bestaat daarom enkel uit behoudswetten. Omdat deze vergelijkingen gekoppeld kunnen worden opgelost, is snelle oplossing van stationaire problemen mogelijk. Hier wordt gebruik gemaakt van multirooster.

De stromingsvergelijkingen worden gediscretiseerd met een eindig-volumemethode. Hierbij worden het convectieve en het diffusieve gedeelte van de stromingsvergelijkingen apart gediscretiseerd, om er zeker van te zijn dat het convectieve deel uit zichzelf stabiel is. Dit is essentieel voor waterstroming, waar de Reynoldsgetallen vaak zeer hoog zijn. Het convectieve deel wordt gediscretiseerd met gelineariseerde Riemannfluxen, die worden afgeleid vanuit kunstmatig gehyperboliseerde stromingsvergelijkingen. Deze fluxen garanderen de stabiliteit van de discretisatie en het goed functioneren van de relaxatiemethodes.

Multirooster wordt gecombineerd met Gauss-Seidelrelaxatie voor het verwijderen van hoogfrequente fouten. Uit Fourieranalyse blijkt dat puntrelaxatie niet effectief is voor het water–luchtmodel bij hoge Reynoldsgetallen. Alternerende horizontale en verticale lijnrelaxatie daarentegen is effectief onder alle stromingscondities. Omdat lijnrelaxatie niet veel duurder is dan puntrelaxatie, wordt deze gebruikt.

Om de fysica van waterstroming correct weer te geven, is een turbulentiemodel vereist. Hier worden de Reynolds-gemiddelde Navier-Stokesvergelijkingen gebruikt, met het één-vergelijkingmodel van Menter. Een turbulentiemodel zorgt voor twee problemen bij het oplossen van de vergelijkingen. Ten eerste kan de bronterm in het turbulentiemodel de lijnrelaxatie instabiel maken. Dit probleem treedt echter niet op in de buurt van een stationaire oplossing. Voor het begin van het oplosproces

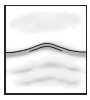
wordt lokale demping van de lijnrelaxatie gebruikt; de benodigde hoeveelheid demping kan worden bepaald uit de convergentie van de Newton-Raphson-oplosmethode in de individuele lijnen. Ten tweede: door het turbulentiemodel, maar ook door het mengsel-oppervlakmodel, lijken de oplossingen op grove roosters niet op de fijnroosteroplossing, waardoor niet-lineair multirooster geen effectieve grofroostercorrecties oplevert. Dit probleem wordt opgelost met een nieuwe multiroostermethode, waarbij niet-lineaire relaxatie op het fijnste rooster wordt gecombineerd met lineaire grofroostercorrecties. De fijnroosteroperator wordt gelineariseerd en deze linearisaties worden rechtstreeks gekopieerd naar de grove roosters; dit garandeert goede grofroostercorrecties.

Als laatste wordt de discretisatie tweede-orde nauwkeurig gemaakt met behulp van een schema met een limiter. Voor de volumefractie, die de plaats van het wateroppervlak aangeeft, wordt een compressieve limiter gebruikt. Met behulp van numerieke antidiffusie houdt deze limiter het uitgesmeerde oppervlak zeer scherp. In tegenstelling tot instationaire methodes geeft een stationaire methode met een zeer simpel compressief schema al goede resultaten. De tweede-orde nauwkeurige vergelijkingen worden opgelost met defect-correctie. In een paar iteraties levert dit een sterke verbetering van de eerste-orde nauwkeurige oplossingen op.

Resultaten worden gepresenteerd voor een 2D kanaalstroming met een drempel in de bodem. Deze tests laten zien dat het capturingmodel fysisch juiste stromingen oplevert die goede overeenkomst vertonen met experimenten en bestaande berekeningen. Multirooster zorgt voor zeer snelle berekeningen: vergeleken met lijnrelaxatie op één rooster is multirooster tot twintig keer sneller.

Als tweede onderwerp in dit proefschrift wordt een roosterverfijningsmethode voor stroming rond schepen gepresenteerd. Allereerst zijn berekeningen gedaan met een level-setcode op gestructureerde roosters. Deze worden gebruikt om de plaats van de grootste fouten in de oplossing te bestuderen en om te zien of roosterverfijning deze fouten effectief vermindert. Dan wordt een lokale roosterverfijningsmethode beschreven, die blokken gestructureerde cellen verbindt met dunne lagen prismacellen. Deze roosters passen in een standaard ongestructureerde datastructuur. Een test op een Series 60 scheepsromp laat zien dat de methode nauwkeuriger resultaten geeft dan gestructureerde roosters, in een kortere rekentijd.

Als afsluiting wordt het eerder beschreven mengselmodel gebruikt voor instationaire, compressibele twee-stofstroming. Het model leidt tot een stelsel van vijf stromingsvergelijkingen in 1D, waaronder een energievergelijking voor één van de stoffen. Deze vergelijking bevat een bronterm die de energieuitwisseling tussen de twee stoffen geeft. Dit model is equivalent met bestaande modellen, maar geeft door zijn afleiding fysisch inzicht. Het kan bovendien worden bestudeerd in schokken. Voor een vereenvoudigd geval worden schokrelaties afgeleid: deze leveren zeer goede overeenkomsten op met experimenteel bepaalde schoksnelheden in metaallegeringen. Numerieke simulaties van schokbuisproblemen laten zien dat het model zowel in continue als in discontinue stroming goed functioneert.



Summary

Surface capturing and multigrid for steady free-surface water flows – Jeroen Wackers

Surface capturing is a technique for modelling the water surface in numerical computations of water flow. For this technique, the computational grid is extended above the water surface and a separate surface model is added to the flow equations, which gives the location of the water surface in the grid. Examples are the level set and volume-of-fluid techniques. The advantage of surface capturing for e.g. the computation of the flow around ships is, that they are generally applicable and can handle complicated ship geometries. For steady flow problems, the major disadvantage is that most capturing methods do not allow the use of fast solution methods. Usually, these flows are solved by time stepping the unsteady flow equations to convergence.

This thesis shows that fast solution of a surface capturing model is possible. For this, a mixture model is used: the flow is modelled both in the water and in the air above it, the interface between air and water is a numerical mixture that is smeared over a few cells. The model distinguishes between air and water with a mass conservation equation for the water; therefore, the flow model consists of conservation laws only. As these equations allow coupled solution, they can be solved efficiently for steady flows. Here, a multigrid method is used.

The flow equations are discretised with a finite-volume method. The convective and diffusive part of the equations are discretised separately, to make sure that the convective part is stable in itself. This is essential for water flow, where the Reynolds numbers are often very high. The convective part is discretised with linearised Riemann fluxes, that are derived from artificially hyperbolised flow equations. These fluxes guarantee the stability of the discretisation and good performance of the relaxation methods.

Multigrid is combined with Gauss-Seidel smoothing to remove the high-frequency errors. Fourier analysis shows that point relaxation is ineffective for the water–air model at high Reynolds numbers. Alternating horizontal and vertical line relaxation, on the other hand, is effective for all flow conditions. As line relaxation is not much more expensive than point relaxation, the former is used.

To correctly represent the physics of water flow, a turbulence model is essential. Here, the Reynolds-averaged Navier-Stokes equations are used, with Menter's one-equation model. A turbulence model causes two problems with the solution of the equations. First, the source term in the turbulence model can make the line smoothing unstable. However, this problem does not appear in the neighbourhood of a steady solution; for the beginning of the solution process, the line smoothing is damped locally. The amount of damping required is determined from the convergence of the Newton-Raphson solver in the individual lines. And second: the turbulence model, but also the mixture surface model, cause large differences in the solutions on fine and coarse grids. Hence, nonlinear multigrid does not produce ef-

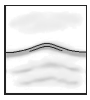
fective coarse grid corrections. This problem is solved with a new multigrid method, that combines nonlinear smoothing on the finest grid with linear coarse grid corrections. The fine grid operator is linearised and these linearisations are copied directly to the coarser grids; this guarantees good coarse grid corrections.

Finally, the discretisation is made second-order accurate with a limited scheme. For the volume fraction, which indicates the location of the water surface, a compressive limiter is used. This limiter keeps the smeared water surface very sharp, using numerical antidiffusion. As opposed to unsteady methods, a steady method gives good results with a very simple compressive scheme. The second-order accurate equations are solved with defect correction. In a few iterations, this results in a great improvement of the first-order accurate solutions.

Results are presented for a 2D channel flow with a bottom bump. These tests show that the capturing model computes physically correct flows that show good agreement with experiments and existing numerical models. Multigrid gives very fast computations: compared with line smoothing on a single grid, multigrid is up to 20 times faster.

The second subject in this thesis is a grid refinement method for flows around ships. First, computations have been performed with a level set code on structured grids. These are used to study the location of the largest errors in the solutions and to see if grid refinement effectively reduces these errors. Then a local grid refinement method is described, that links blocks of structured cells with thin layers of prism cells. These grids fit in a standard unstructured data structure. A test on a Series 60 ship hull shows that the method gives more accurate results than structured grids, in shorter computing times.

To conclude the thesis, the mixture model described before is used for unsteady, compressible two-fluid flow. The model leads to a system of five flow equations in 1D; one of these is an energy exchange equation for one of the two fluids. This equation contains a source term that gives the energy exchange between the fluids. This model is equivalent with existing models, but the derivation gives physical insight. Furthermore, the model can be studied in shocks. For a simplified case, shock relations are derived: these give very good agreement with experimentally determined shock speeds in metal alloys. Numerical simulations of shock tube problems show that the model functions well in continuous and in discontinuous flow.



Curriculum vitae

Jeroen Wackers was born in Nijmegen on August 20, 1978. He attended the Stedelijk Gymnasium Nijmegen from 1991 until his graduation in 1997.

From 1997, he studied Aerospace Engineering at Delft University of Technology. He completed his first year with honours in 1998. During his study, he spent six months of practical training at Totalförsvarets Forskningsinstitut (FOI) in Bromma, Sweden. He graduated with honours in 2003. For his master's thesis entitled "An adaptive-gridding solution method for the 2D unsteady Euler equations", supervised by prof.dr.ir. B. Koren, he received the 2003 award for the best Aerospace Engineering graduate student.

After his graduation, he worked at CWI in the group of prof.dr. U. Ebert. In the summer of 2003, he started his Ph.D. work at CWI with prof. Koren. During this work, he spent two months at the National Maritime Research Institute (NMRI) in Mitaka, Tokyo, Japan.

He is currently employed as a postdoctoral researcher at the Ecole Centrale de Nantes, France.