Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Three-dimensional computation of non-hydrostatic free-surface flows

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

SEBASTIAN ULLMANN

Delft, the Netherlands
July 2008

Deltares

TU Delft

MSc THESIS APPLIED MATHEMATICS

"Three-dimensional computation of non-hydrostatic free-surface flows"

SEBASTIAN ULLMANN

Delft University of Technology

Daily supervisor

Dr. P. Wilders

Responsible professor

Prof.dr.ir. A.W. Heemink

Other thesis committee members

Prof.dr.ir. G.S. Stelling

Ir. J.A.Th.M. van Kester

Dr.ir. M.J.A. Borsboom

July 2008

Delft, the Netherlands

Preface

This thesis contains the results of my studies performed at the Deltares Institute and at the Delft Institute of Applied Mathematics of the Delft University of Technology. This work could not have been finished without the support of a number of people. I want to thank the members of my thesis committee:

- Mart Borsboom for having time for me whenever I had questions or something to proof-read, for motivating me and for our many discussions about non-hydrostatic modeling,
- Jan van Kester for introducing me to the Delft3D-Flow code, for his useful suggestions with respect to this thesis and for nice debugging sessions,
- Peter Wilders for providing me with this thesis work, for giving me both freedom and guidance for my work and for valuable discussions,
- Arnold Heemink for being my responsible professor,
- Guus Stelling for his interest in my work and for a good talk about splitting methods.

Further I want to thank some people from Deltares who contributed to this thesis in several ways: Menno Genseberger for giving me advice with the chapter about the iterative solver; Marcel Zijlema for providing me with some interesting literature and with the data for the Baji-Battjes test case; Bert Jagers for the ship lock model; Eric de Goede for helping me with Delft3D-Flow; all the others at Deltares who should be in this list.

This thesis is the result of my two years studying in the Netherlands. I thank my fellow students for making this time such a good one. A very special and sincere thank-you goes to those who gave me their moral support and love during this time abroad: my family and my girlfriend Maria.

Summary

This thesis describes the improvement of the accuracy and performance of the Delft3D-FLOW software for strongly non-hydrostatic flow applications. A non-hydrostatic free-surface flow model in z-grid formulation based on a pressure-correction method is introduced. The model differs from the original non-hydrostatic Delft3D-FLOW model in the way the pressure is treated. The splitting in hydrostatic and non-hydrostatic pressure components is not applied anymore and the pressure terms are implemented with a θ-scheme, allowing for a second-order accuracy in time.

The dynamic boundary condition at the free surface is realized with a linear interpolation of pressure values, leading to a non-symmetric pressure-correction matrix. The BiCGSTAB method is implemented as a non-symmetric iterative solver for the resulting system of equations. The convergence of the iterative solver is improved by replacing the existent tridiagonal Jacobi preconditioner with a modified incomplete LU decomposition.

To assess the capabilities of the model, test simulations are done for a standing wave in a closed basin, for the wave propagation over a submerged bar and for a buoyancy-driven flow in a ship lock. It is found that the new model is computationally efficient and gives more accurate results than the previous model. In particular, the interpolated pressure condition at the surface improves the accuracy of the phase speed of traveling waves, while adding little computational cost to the overall solution process.

Contents

Chapter 1

Introduction

The Dutch research and consultancy institute WL|Delft Hydraulics, now part of Deltares, developed the software Delft3D-FLOW originally for the solution of the three-dimensional shallow-water equations. The code is able to efficiently and accurately calculate environmental flows that feature a small vertical velocity component compared to the horizontal velocity components and waves that are much longer than the water depth. In this case the pressure field can be assumed to be hydrostatic, which means that the pressure is determined only by the weight of the water. Due to the increase of computing power it became feasible over the years to resolve finer and finer details in the simulations. When approaching smaller time and length scales, however, some of the approximations inherent to the Delft3D-FLOW shallow-water code are not valid anymore and have to be revised.

In order to compute flows with a significant vertical velocity more accurately, by the end of the 1990s several *non-hydrostatic* models were proposed in the literature. The idea behind these models is to add an extension to existing shallow-water software in order to obtain a more realistic pressure field. This is done by introducing a non-hydrostatic pressure component as a correction to the hydrostatic pressure. By switching off the non-hydrostatic extension the original shallow-water code is obtained. Especially in weakly non-hydrostatic flows, where the deviation from the shallow-water condition is small, these methods give accurate results with a moderate increase in computational work.

The non-hydrostatic extension of Delft3D-FLOW is based on a z-grid formulation in the vertical direction, while the original model is formulated using a σ-grid. Bijvelds [2001] implemented a hydrostatic z-grid model in Delft3D-FLOW to be able to simulate stratified flows in estuaries with a steep bottom topography. In Bijvelds [2003] he extended this model with a non-hydrostatic module, which is still present in the current Delft3D-FLOW version 3.26. The scheme is based on the method of Casulli [1999], but there are some important differences: Bijvelds uses a pressure-correction method instead of a fractional step method, he uses an only first-order accurate alternating direction implicit (ADI) method in the hydrostatic prediction step instead of solving a directionally coupled system for the preliminary surface elevation, and he does not realize a θ-scheme for the time integration of the pressure terms, as Casulli did.

Ever since the non-hydrostatic model was available in Delft3D-FLOW, attempts were made to use it for simulations of strongly non-hydrostatic flows, like buoyant jets, short waves and flows over hydraulic structures. Although the results were qualitatively much more accurate than the results obtained with the hydrostatic scheme, the results were quantitatively unsatisfactory in some cases and the solution process was computationally expensive. There were several possible reasons why the non-hydrostatic model of Bijvelds did not perform as well as it was supposed to: Several features of the model were still tailored to a shallow-water environment, it was not known whether the code was free of bugs, and the splitting of the pressure in hydrostatic and

non-hydrostatic components was thought to introduce errors. To address the last problem, a new non-hydrostatic model, with a different pressure handling, was introduced by Borsboom [2007] as a possible alternative to the model of Bijvelds. During the development of the code for the Borsboom model, several programming errors were found in the code of the Bijvelds model. After removing these errors some of the problems that had actually triggered the development of the new model were solved, and two non-hydrostatic models became available.

By the end of 2007, the Borsboom model could not be fully tested, yet, because it required a non-symmetric iterative solver, which was not available in Delft3D-FLOW at that time. This is the starting point of the work presented in this thesis. The first aim is the development and implementation of a fast and reliable iterative solver for the Borsboom model. Having the solver available, the second aim is to test the model and find the differences between the two available approaches.

The new non-hydrostatic pressure-correction scheme is derived and presented in chapter 2, with a focus on the correction step and the handling of the free-surface. The implementation of an efficient iterative method to solve the pressure-correction equation is described in chapter 3. The accuracy and performance of the non-hydrostatic model is tested in chapter 4 and compared with the model of Bijvelds. Finally, in chapter 5, the results of this study are summarized and suggestions for further research are given.

1.1 The free-surface problem

The incompressible Navier-Stokes equations for the fluid flow in three dimensions consist of three momentum equations and a continuity equation. Formulated in Cartesian coordinates, they relate the velocity components $u(x,y,z,t)$, $v(x,y,z,t)$ and $w(x,y,z,t)$ in x-, y- and z-direction and the pressure to each other. As will be done in all equations presented in this thesis, we assume the density and the atmospheric pressure to be constant and define the pressure variable p as the pressure minus the atmospheric pressure, divided by the density. Using the velocity vector $\vec{u} = (u,v,w)^T$ the Navier-Stokes equations can be written as

$$\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u} = -\nabla p + \nabla\cdot\sigma + \vec{f} \tag{1.1}$$

$$\nabla\cdot\vec{u} = 0. \tag{1.2}$$

The set of the three equations in (1.1) are the momentum equations and (1.2) is the continuity equation. The external force field, $\vec{f}(x,y,z,t)$ is assumed to be given and the deviatoric stress tensor is defined as

$$\sigma = \nu \begin{pmatrix} 2\frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} & 2\frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} & \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} & 2\frac{\partial w}{\partial z} \end{pmatrix},$$

with $\nu(x,y,z,t)$ the molecular viscosity.

To obtain a well-posed problem from the Navier-Stokes equations, a full set of boundary conditions needs to be applied. In this thesis only the boundary conditions in the vertical direction are of interest. The domain boundaries in the vertical direction are given as height functions. The water surface elevation ζ and the bottom depth d with respect to an arbitrary reference plane $z = 0$ are defined as $\zeta = \zeta(x,y,t)$ and $d = d(x,y)$, where $\zeta \geq -d$. The water depth, which is the distance between the water surface and the bottom depth is given as $h(x,y,t) = \zeta(x,y,t) + d(x,y)$, as sketched in figure 1.1. With H we describe the mean water depth. The definition of the surface and the bottom as height functions is a restriction on the class of flows that can be described by the equations, excluding for example overturning waves.

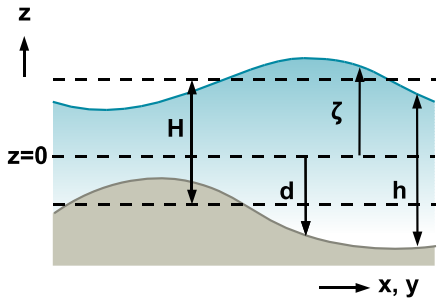

Figure 1.1: Sketch of a free surface problem with definitions of the depth variables.

The bottom and the free surface are assumed to be impermeable. This is expressed by the kinematic boundary conditions

$$\frac{\partial \zeta}{\partial t} + u|_\zeta \frac{\partial \zeta}{\partial x} + v|_\zeta \frac{\partial \zeta}{\partial y} - w|_\zeta = 0, \tag{1.3}$$

$$u|_{-d} \frac{\partial d}{\partial x} + v|_{-d} \frac{\partial d}{\partial y} + w|_{-d} = 0. \tag{1.4}$$

Further we want to express a free slip condition along the surface boundary and along the bottom boundary, as given by

$$\vec{n} \cdot \sigma \cdot \vec{s} = 0, \qquad \vec{n} \cdot \sigma \cdot \vec{t} = 0, \tag{1.5}$$

where $\vec{s}$ and $\vec{t}$ are linearly independent tangent vectors and $\vec{n}$ is the outer normal vector (compare Wesseling [2001], section 6.2). For the surface boundary, the respective formulas are given as

$$\vec{n} = \begin{pmatrix} -\frac{\partial \zeta}{\partial x} \\ -\frac{\partial \zeta}{\partial y} \\ 1 \end{pmatrix}, \qquad \vec{s} = \begin{pmatrix} 1 \\ 0 \\ \frac{\partial \zeta}{\partial x} \end{pmatrix}, \qquad \vec{t} = \begin{pmatrix} 0 \\ 1 \\ \frac{\partial \zeta}{\partial y} \end{pmatrix}.$$

The formulas for the bottom boundary are analog and not shown here. Depending on the problem, the boundary conditions (1.5) may be modified to include shear stresses due to wind effects or bottom roughness.

Neglecting the surface tension, we also prescribe a dynamic boundary condition

$$p|_\zeta = 0 \tag{1.6}$$

for the pressure at the free surface. The atmospheric pressure and the density, which were assumed constant, are already contained in the variable p.

The model described so far is a macroscopic free-surface problem based on the incompressible Navier-Stokes equations. Given some suitable initial conditions and side boundary conditions, the equations describe various flow and wave phenomena in three dimensions. However, the resulting systems of discretized equations can be computationally expensive to solve. Therefore, several simplifications of the equations are often made by taking into account typical length and time scales of the problem.

1.2 Hydrostatic and non-hydrostatic modeling

The first simplification of the free-surface problem described in section (1.1) is related to the effect of turbulence. Nearly all macroscopic environmental flows are turbulent. Direct numerical modeling of turbulent flows with the Navier-Stokes equations requires a grid that resolves

all turbulent length scales. This is computationally not feasible for most engineering or environmental applications. At the same time one is usually not interested in the details of the turbulent flow pattern. Therefore, by time-averaging over the velocity field the turbulent effect is reduced to a turbulent viscosity or eddy viscosity. Since the eddy viscosity is usually much higher than the molecular viscosity, the latter is neglected. The resulting equations are called the Reynolds-averaged Navier-Stokes equations (RANS equations). The turbulent viscosity field has to be determined by an adequate turbulence model, or, as another approximation, may be set to a constant value.

In order to achieve a higher efficiency of the model, the relation between the vertical and horizontal length scales can also be taken into account. This leads to the notions of hydrostatic and non-hydrostatic modeling and to the discrimination between weakly and strongly non-hydrostatic applications. In order to classify these concepts, we consider the aspect ratio L/H, where H is the mean water depth and L is the minimum horizontal length scale to be resolved accurately. For wave problems L stands for a typical wave length, for flow problems it may also be the length of a large eddy or the size of a jet or a plume. For pure wave problems, it is possible to base the discussion about length scales on the dispersion relation, as it will be done in the beginning of chapter 4. For general problems, however, the discussion is a bit more involved and the classification of a flow problem may even be ambiguous in some cases. However, based on the size of L/H, we can roughly distinguish:

- *Hydrostatic flows:* If L/H is around 100 or larger, then typically $|u| + |v| \gg |w|$, and the terms containing w in the third momentum equation may be neglected. We assume further that the external force originates fully from earth's gravity, so that $\vec{f} = (0, 0, g)^T$. Now the w-momentum equation of (1.1) can be simplified to

$$\frac{\partial p}{\partial z} = g \qquad \text{or} \qquad p(x, y, z, t) = g(\zeta(x, y, t) - z), \tag{1.7}$$

  which is called the hydrostatic pressure relation. The term $g(\zeta - z)$ is the hydrostatic pressure, which is scaled in the same way as the pressure variable p. Substituting (1.7) in the other two momentum equations of (1.1) we get

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -g\frac{\partial \zeta}{\partial x} + \nu_h\frac{\partial^2 u}{\partial x^2} + \nu_h\frac{\partial^2 u}{\partial y^2} + \frac{\partial}{\partial z}\Big(\nu_v\frac{\partial u}{\partial z}\Big), \tag{1.8}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -g\frac{\partial \zeta}{\partial y} + \nu_h\frac{\partial^2 v}{\partial x^2} + \nu_h\frac{\partial^2 v}{\partial y^2} + \frac{\partial}{\partial z}\Big(\nu_v\frac{\partial v}{\partial z}\Big), \tag{1.9}$$

  where we used some approximations with respect to the viscous terms: Although the viscous terms can usually not be neglected, the variations of the viscosity may assumed to be small. This means that we can put ν in front of the derivatives. In practice often an anisotropic turbulence model is used, leading to different horizontal and vertical eddy viscosity coefficients ν_h and ν_v, respectively. The form of equations (1.8) and (1.9) is used in the hydrostatic model of Delft3D-Flow.

  To obtain w, we integrate the continuity equation over the depth,

$$\begin{aligned} 0 &= \int_{-d}^{z}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)\mathrm{d}z \\ &= \frac{\partial}{\partial x}\int_{-d}^{z} u\,\mathrm{d}z + \frac{\partial}{\partial y}\int_{-d}^{z} v\,\mathrm{d}z + w|_z - w|_{-d} - u|_{-d}\frac{\partial d}{\partial x} - v|_{-d}\frac{\partial d}{\partial y} \\ &= \frac{\partial}{\partial x}\int_{-d}^{z} u\,\mathrm{d}z + \frac{\partial}{\partial y}\int_{-d}^{z} v\,\mathrm{d}z + w|_z, \end{aligned}$$

where we have used the kinematic condition (1.4) at the bottom. To obtain ζ we also use a depth integration,

$$\begin{aligned}
0 &= \int_{-d}^{\zeta} \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \mathrm{d}z \\
&= \frac{\partial}{\partial x} \int_{-d}^{\zeta} u \, \mathrm{d}z + \frac{\partial}{\partial y} \int_{-d}^{\zeta} v \, \mathrm{d}z + w|_{\zeta} - w|_{-d} - u|_{\zeta} \frac{\partial \zeta}{\partial x} - v|_{\zeta} \frac{\partial \zeta}{\partial y} - u|_{-d} \frac{\partial d}{\partial x} - v|_{-d} \frac{\partial d}{\partial y} \\
&= \frac{\partial}{\partial x} \int_{-d}^{\zeta} u \, \mathrm{d}z + \frac{\partial}{\partial y} \int_{-d}^{\zeta} v \, \mathrm{d}z + \frac{\partial \zeta}{\partial t},
\end{aligned}$$

where we have additionally used the kinematic condition (1.3) at the free surface. Using a depth-integrated continuity equation is more robust than using the kinematic condition directly.

The momentum equations together with the depth integrated continuity equation form the three-dimensional shallow-water equations with the unknowns $u(x, y, z, t)$, $v(x, y, z, t)$, $w(x, y, z, t)$ and $\zeta(x, y, t)$. By depth integrating the momentum equations in a similar way as the continuity equation, the vertical velocity drops out of the momentum equations and we arrive at the two-dimensional shallow-water equations for the depth-averaged horizontal velocities (see Vreugdenhil [1994] for details).

We can also simplify the bottom and the surface boundary conditions related to the shear stress: Assuming a sufficiently small gradient of the bottom and the free surface, (1.5) may be reduced to

$$\frac{\partial u}{\partial z} = 0 \qquad \frac{\partial v}{\partial z} = 0,$$

which means that there is no horizontal stress.

- *Weakly non-hydrostatic flows:* When L/H is, say, between about 10 and 100, then the w-momentum equation of the Navier-Stokes equations can not be simplified to a hydrostatic pressure relation anymore. However, since the typical horizontal length scales are still much longer than the water depth, it is reasonable to assume that $p \approx g(\zeta - z)$. Introducing a non-hydrostatic pressure q that describes the deviation of p from the hydrostatic pressure, we can write $p = g(\zeta - z) + q$, knowing that q is small compared to $g(\zeta - z)$, and, more importantly, that the gradient of q is smaller than the gradient of $g\zeta$.

- *Strongly non-hydrostatic flows:* When L/H is about 10 or smaller, then the non-hydrostatic pressure is no longer small compared to the hydrostatic pressure. The same holds if the flow field contains local effects like buoyancy, discharges or the flow around an obstacle. In this case the pressure gradient is mainly influenced by these local effects and the deviation from the hydrostatic assumption is large. We can still write $p = g(\zeta - z) + q$, but we have to expect that the gradient of q can be much larger than the gradient of $g\zeta$.

Non-hydrostatic models are models to solve the incompressible Navier-Stokes equations, so they are based on the full set of equations (1.1) and (1.2). However, they were developed as extensions to shallow-water models. Therefore, they usually involve a splitting of the pressure in hydrostatic and non-hydrostatic components and employ the concept of horizontal and vertical eddy viscosity. The basic equations for non-hydrostatic models are therefore often given in a

form like

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{\partial q}{\partial x} - g\frac{\partial \zeta}{\partial x} + \nu_h\frac{\partial^2 u}{\partial x^2} + \nu_h\frac{\partial^2 u}{\partial y^2} + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial u}{\partial z}\right) \tag{1.10}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -\frac{\partial q}{\partial y} - g\frac{\partial \zeta}{\partial y} + \nu_h\frac{\partial^2 v}{\partial x^2} + \nu_h\frac{\partial^2 v}{\partial y^2} + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial v}{\partial z}\right) \tag{1.11}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{\partial q}{\partial z} + \nu_h\frac{\partial^2 w}{\partial x^2} + \nu_h\frac{\partial^2 w}{\partial y^2} + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial w}{\partial z}\right) \tag{1.12}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \tag{1.13}$$

In comparison to the three-dimensional shallow-water equations we notice the presence of the third momentum equation and the gradient of the non-hydrostatic pressure q. Non-hydrostatic numerical schemes, however, are not only characterized by the governing field equations they are supposed to solve. They may also be distinguished by the type of grid that is used, the way the different terms of the equations are discretized and the way in which the equations are split.

The two types of coordinate systems that are used most often in non-hydrostatic modeling are σ-coordinates and z-coordinates. The difference between them is the way they incorporate the boundaries at the bottom and at the top in the grid. A sketch of the two approaches is given in figure 1.2.

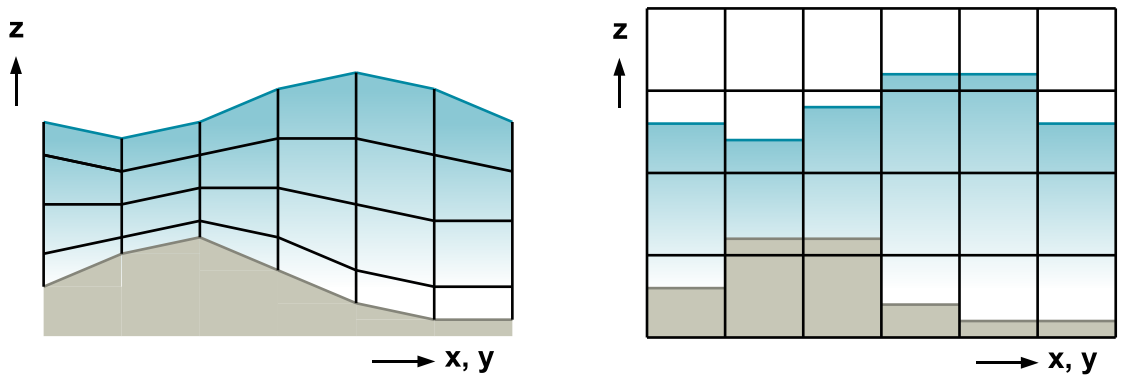

Figure 1.2: Sketches of two frequently used grid formulations in non-hydrostatic modeling, σ-grid (left) and the z-grid (right).

In the σ-coordinate approach the grid is aligned with the bottom topography and the moving surface. In every grid node the thickness of each individual grid layer is specified as a fixed fraction of the water depth $h(x, y, t)$. The fraction is constant throughout each layer. The transformation of the equations into the σ-grid is done by rescaling the vertical axis, which makes the three-dimensional equations and their discretization more complicated.

In the z-coordinate approach the grid is fixed and consists of layers with a prescribed layer thickness. At the bottom and at the surface there are cells which are partly outside and partly inside the domain. The boundary handling in the z-grid approach is more difficult than in the σ-grid approach, because of the book-keeping of the boundary cells and because of the special cases that arise when cells have neighboring cells outside the computational domain. However, since the equations do not have to be transformed, the discretization in the interior domain is more straightforward.

1.3 Non-hydrostatic schemes in the literature

Since the first non-hydrostatic schemes were proposed in the mid-1990s, they have been implemented in several software packages for simulating environmental free-surface flow problems.

Mahadevan et al. [1996a,b] developed a non-hydrostatic model for weakly non-hydrostatic ocean applications to resolve flow phenomena with typical horizontal length scales of 100 km. In vertical direction σ-coordinates are used and in the horizontal plane a boundary-fitted curvilinear mesh is applied. The method consists of two steps: In the first step a hydrostatic computation is carried out and in the second step the effect of a non-hydrostatic pressure component q is taken into account. A similar model was proposed by Casulli and Stelling [1998], using a z-grid in the vertical direction and an orthogonal grid in the horizontal plane.

In the model of Mahadevan and in the model of Casulli and Stelling, there is only a one-way coupling between the hydrostatic and the non-hydrostatic pressure components, because the surface elevation is determined in the first step, without taking into account the effect of the non-hydrostatic pressure. To couple both pressure components more tightly, Casulli [1999] proposed a scheme, where a correction of the surface elevation was incorporated in the second step. The approach of Casulli was implemented in the software TRIM-3D of the German Bundesanstalt für Wasserbau (BAW). The scheme was extended to unstructured horizontal grids in Casulli and Zanolli [2002]. Koyigit et al. [2002] described a method similar to Casulli [1999], but in σ-coordinates.

The methods of the above-mentioned references are fractional step methods. This means that the second step of the computations always features an unknown of the dimension of a pressure. Fringer et al. [2006] proposed a scheme which uses a pressure-correction as an unknown, which has the dimension of a pressure times the time step. This approach is called a pressure-correction method. It is shown in Armfield and Street [2002] that pressure-correction methods have a higher order of convergence with respect to the time step than fractional step methods. The method of Fringer et al. was implemented in the SUNTANS code for coastal ocean simulations.

In most non-hydrostatic models the unknowns are arranged on a staggered grid, which means that the velocity unknowns are located at the faces of the discretization cells, while the pressures are located in the cell centers. A different approach is followed by Stelling and Zijlema [2003], Zijlema and Stelling [2005] and Zijlema and Stelling [2008]. They propose a pressure-correction method where the pressure unknowns are located at the top faces of each cell. This approach is realized in the code TRIWAQ, which was developed by the Dutch Rijkswaterstaat and is now maintained by the Deltares Institute.

All the methods listed in this section have been developed with particular types of applications in mind. This means that there is no model that is generally better than others, but there will be always models that are more suited than others for a certain application. To be able to assess the suitability of a model in a certain application one has to consider the physical and numerical approximations that have been made and the errors that result from these approximations.

1.4 Errors in non-hydrostatic models

The essence of computational modeling is finding a reasonably simple algorithm that approximates the solution of a problem well enough, thereby requiring a limited amount of computational time and memory. The accuracy of the solution is assessed by its error, which is the difference between the numerical solution and the true physics. In practice only an estimate of the error is available, because the true solution can not be measured or analytically calculated.

Algorithms to calculate non-hydrostatic free-surface flows can be relatively complex, because

the field variables are coupled in a non-linear algebraic way and many physical aspects have to be taken into account to obtain realistic results. Depending on the scales and the physical effects to be resolved, different errors are predominant in the numerical solution. The identification of the most important error sources can be done using mathematical analysis or by performing numerical experiments of test cases where either an exact solution or measurements are available. Once the origin of the predominant error is found, steps can be taken to reduce it, for example by changing the numerical scheme or the physical model, changing parameters or refining the mesh.

The following error sources are the most important in the context of the numerical modeling of physical processes:

- *Data errors*: Simulations of realistic problems contain uncertainties for example in the parameters, initial conditions and boundary conditions because of measurement errors, misfits between the state space of the measurements and of the discretized model or a lack of information.

- *Modeling errors*: When a physical problem is formulated in terms of equations, usually certain approximations are made. Often linearizations of non-linear relations are used and effects of minor importance are ignored completely. This leads to an error that is even present if the resulting equations are solved perfectly.

- *Discretization errors*: When a continuous solution is approximated by a solution at a finite number of grid points and time steps, we make an error, because values of derivatives and integrals have to be approximated using information at neighboring grid points. The resulting errors only diminish if the space and time step approach zero, provided the scheme is consistent and stable.

- *Convergence errors:* When an iterative method is used to solve a system of equations, then the iteration is usually stopped before the best possible accuracy is reached, which leads to a convergence error. Good stopping criteria and threshold values are necessary to bind this error while still requiring a minimum number of iterations.

- *Round-off errors*: Since numbers are represented with finite precision in computer memory, mathematical operations with these numbers lead to a rounding of the last digits. When these round-off error accumulate, they can spoil the numerical solution.

Data errors are problem specific and not inherent to the model itself. It is not possible to decrease data errors by changing the model, so these errors are out of scope for the here presented work. The model equations and the respective approximations of the physical processes are chosen on the basis of the typical time- and length scales to be resolved and by weighting the importance of the processes involved in typical applications. The discretization of the equations includes the generation of a computational mesh, the approximation of the continuous variables in terms of discrete variables at the mesh points and the definition of a set of equations for the discretized variables. The discretization should be done in such a way that the respective error is smaller than or at most of the same order as the data and modeling errors. The systems of equations that are found in the context of the numerical solution of partial differential equations typically have discretized variables as unknowns. Therefore it makes sense to require that the convergence error of an iterative solution method is smaller than the discretization error. Round-off errors are usually much smaller than the other errors mentioned above. The accumulation of round-off errors can be avoided by the use of stable numerical schemes.

This thesis we will mainly focus on the discretization errors of non-hydrostatic models and ways to reduce them. Different types of discretization errors can be distinguished:

- *Time discretization errors*: Suppose we are given a partial differential equation that describes the time evolution of a field variable f. The equation is written as

$$\frac{\partial f(\vec{x},t)}{\partial t} = g(f(\vec{x},t),\vec{x},t),$$

where g may be a complicated nonlinear function. Time discretization schemes are used to evolve the solution forward in time by stepping through discrete times t^n. At every time step n the partial time-derivative on the left-hand side can be approximated using $(f(\vec{x},t^{n+1})-f(\vec{x},t^n))/(t^{n+1}-t^n)$, which is identical to the derivative in the limit $t^{n+1}-t^n \to 0$. Depending on the discretization of the right-hand side different orders of convergence of this limit can be realized.

- *Space discretization error*: Similarly to the time discretization error, also the approximation of the spatial derivatives is done using information of surrounding points, leading to an error that is dependent on the size of the spatial step.

- *Splitting error*: If a system of equations contains complicated relations between the dependent variables, a simplification can be made by treating several terms independently in consecutive calculation steps. Consider a time-discretized system of linear equations for some vector variable f,

$$f^{n+1} = f^n - \Delta t(Af^n + Bf^{n+1}).$$

If the matrix B couples the unknowns strongly, one can often gain computational efficiency by treating some of the implicit terms separately. Therefore, we introduce the splitting $B = B_1 + B_2$ and use B_1 in a first step, where we solve for a preliminary variable f^{n+1*}. In a second step we take B_2 into account and solve for the final solution $\tilde{f}^{n+1}$ with f^{n+1*} given. The split system of equations is given as

$$\begin{aligned} f^{n+1*} &= f^n - \Delta t(Af^n + B_1 f^{n+1*}) \\ \tilde{f}^{n+1} &= f^{n+1*} - \Delta t B_2 \tilde{f}^{n+1}. \end{aligned}$$

For special choices of B_1 and B_2 it can be much simpler to solve the split system than to solve the original system. However, the splitting introduces an error $\epsilon = \tilde{f}^{n+1} - f^{n+1}$. It can easily be seen that the sum of both equations is not equivalent to the original system, because the preliminary f^{n+1*} can not be eliminated from the sum of both equations. It can be shown that in this case the splitting error is $\epsilon = \tilde{f}^{n+1} - f^{n+1} \approx -\Delta t^2 B_1 B_2 \tilde{f}^{n+1}$ (see appendix A), which is the error per time step, so the global error is of $O(\Delta t)$.

Chapter 2

Numerical model

For the presentation of the non-hydrostatic numerical model in this thesis, a few simplifications are made, which are not in the actual Delft3D-FLOW code:

- The horizontal grid is taken Cartesian.
- Only gravity is used as an external forcing. Coriolis effects, for example, are neglected.
- The density is taken constant.
- The atmospheric pressure is assumed constant.

As stated in the introduction, the pressure variable p denotes the pressure minus the atmospheric pressure, divided by the density.

2.1 Model equations

Following the discussion of section 1.2, the governing three-dimensional equations are given in the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{\partial p}{\partial x} + \nu_h\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial u}{\partial z}\right) \tag{2.1}$$

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

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{\partial p}{\partial z} + \nu_h\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial w}{\partial z}\right) - g \tag{2.3}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \tag{2.4}$$

Note that we have not split the pressure in its non-hydrostatic and hydrostatic components. At the bottom and at the free surface we prescribe a kinematic condition,

$$\frac{\partial \zeta}{\partial t} + u|_\zeta\frac{\partial \zeta}{\partial x} + v|_\zeta\frac{\partial \zeta}{\partial y} - w|_\zeta = 0, \tag{2.5}$$

$$u|_{-d}\frac{\partial d}{\partial x} + v|_{-d}\frac{\partial d}{\partial y} + w|_{-d} = 0 \tag{2.6}$$

and zero stress in horizontal direction,

$$\frac{\partial u}{\partial z}|_\zeta = 0, \qquad \frac{\partial v}{\partial z}|_\zeta = 0, \qquad \frac{\partial u}{\partial z}|_{-d} = 0, \qquad \frac{\partial v}{\partial z}|_{-d} = 0, \tag{2.7}$$

assuming that the free surface is sufficiently close to a horizontal plane, which can be questionable for some strongly non-hydrostatic applications. We apply a zero pressure condition,

$$p|_\zeta = 0, \tag{2.8}$$

neglecting the effect of the surface tension. The implementation of the side boundaries is not considered in this thesis. A description can be found in the Delft3D-FLOW user manual [Del, 2006].

2.2 Computational grid

The three-dimensional field variables are discretized on a staggered, cell-centered grid (Arakawa C grid) [Arakawa and Lamb, 1977] in three dimensions, as sketched in figure 2.1. The two-dimensional fields of the bottom depth $d(x, y)$ and the surface elevation $\zeta(x, y, t)$ are discretized at the centers of the grid columns. In the horizontal plane the grid consists of spatially fixed, rectangular grid cells with dimensions $\Delta x_i \times \Delta y_j$ on a grid that is non-uniform in x- and y-direction. In z-direction the grid consists of layers with a non-uniform thickness $\Delta z_k = z_{k+1/2} - z_{k-1/2}$, where $z_{k+1/2}$ is the location of the top cell faces and $z_{k-1/2}$ is the location of the bottom cell faces of layer k.

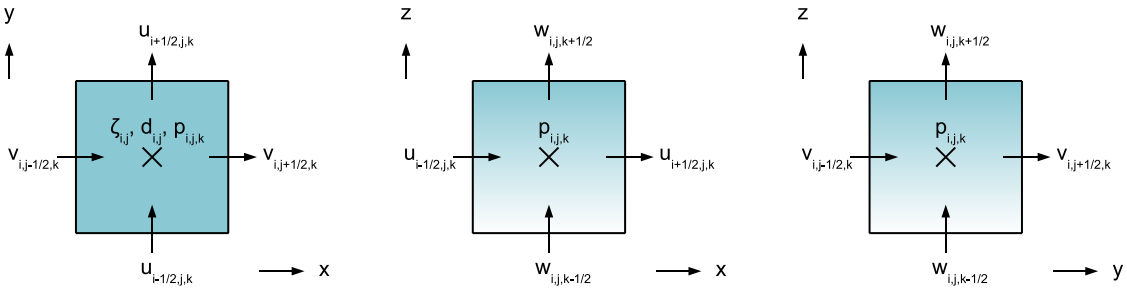

Figure 2.1: Sketch of the positions of the unknowns on the staggered grid.

To include the spatially discretized bottom depths $d_{i,j}$ and surface elevations $\zeta_{i,j}(t)$ in the discretized equations of the three-dimensional field variables, we define the volume coordinates

$$\begin{aligned} z_{i,j,k+1/2} &= \min(z_{k+1/2}, \zeta_{i,j}), \\ z_{i,j,k-1/2} &= \max(z_{k-1/2}, -d_{i,j}), \\ \Delta z_{i,j,k} &= \min(z_{k+1/2}, \zeta_{i,j}(t)) - \max(z_{k-1/2}, -d_{i,j}). \end{aligned} \tag{2.9}$$

Note that $z_{k\pm1/2}$ is related to a complete layer and $z_{i,j,k-1/2}$ is related to a single cell.

With the definitions given above, there can exist single cells or complete layers which are outside the domain, because they are situated below the bottom boundary or above the surface boundary. We decide at the beginning of each time step about what cell is inside and what cell is outside the domain. Therefore, we need the time-discretized free-surface variable $\zeta^n_{i,j}$. We define the variables $k^{i,j}_{\min}$ and $k^{i,j,n}_{\max}$ in a way that

$$\begin{aligned} k = k^{i,j}_{\min} \quad &\text{if} \quad z_{k-1/2} < h_{i,j} \leq z_{k+1/2}, \\ k = k^{i,j,n}_{\max} \quad &\text{if} \quad z_{k-1/2} < \zeta^n_{i,j} \leq z_{k+1/2}. \end{aligned}$$

We will use the following terminology: Cells with index $k^{i,j}_{\min} < k < k^{i,j,n}_{\max}$ are *filled* or *interior* cells. Cells with $k < k^{i,j}_{\min}$ or $k > k^{i,j,n}_{\max}$ are *empty* or *exterior* cells. Finally, cells with $k = k^{i,j}_{\min}$

are *bottom boundary* cells and cells with $k = k_{\max}^{i,j,n}$ are *surface boundary* cells. It is possible that in some grid columns $k_{\min}^{i,j} = k_{\max}^{i,j,n}$, so the bottom and the surface are contained in one grid cell. However, we will not consider the special case where also $\zeta_{i,j}^n = -d_{i,j}$, which appears for example in simulations that include a moving shore line. For the computations it is not important that a discretization point in the middle of a cell with vertical index $k_{\min}^{i,j}$ or $k_{\max}^{i,j,n}$ lies actually outside the *physical* domain if the cell is less than 50% filled with water.

In the finite volume discretization of the continuity equation, given in section 2.4, we will encounter the vertical volume spacings $\Delta z_{i\pm1/2,j,k}$ and $\Delta z_{i,j\pm1/2,k}$. These spacings are linear combinations of the respective vertical volume sizes of horizontally neighboring volumes. It is important that the vertical spacings defined at the border of two horizontally neighboring volumes are compatible. The definitions used in Delft3D-FLOW are

$$\Delta z_{i+1/2,j,k} = \min(\Delta z_{i,j,k}, \Delta z_{i+1,j,k}), \qquad \Delta z_{i,j+1/2,k} = \min(\Delta z_{i,j,k}, \Delta z_{i,j+1,k}). \tag{2.10}$$

This is only one possible choice. It is not straightforward how to choose these spacings, in particular, when the special cases are considered that arise when a non-empty cell has one or several empty horizontal neighbors, which is the case when the free surface or the bottom are varying between different layers. The peculiarities of the z-grid formulation related to these "grid crossings" are not subject of this thesis.

Up to now we have described the discretization of the spatial domain. The discretization in time is much simpler, because it only involves variables at two subsequent time levels n and $n+1$ as long as only one-step schemes are considered. In the definition of the solution method we will encounter preliminary variables denoted with superscript $n+1*$. These are predictions of variables at time step $n+1$.

2.3 Momentum equations

The time discretization of the governing equations given in section 2.1 is done with a pressure-correction method. The method involves a splitting of the momentum equations. First, a preliminary velocity is calculated, neglecting the pressure at the new time level in the momentum equations. Then, a pressure-correction is calculated that, when added to the momentum equations, leads to a divergence free velocity field. Finally, all unknowns are updated to the new time level.

The pressure-correction method is derived starting with the time discretization of the momentum equations (2.1) to (2.3),

$$\begin{aligned}\frac{u^{n+1}-u^n}{\Delta t} + u^n\frac{\partial u^n}{\partial x} + v^n\frac{\partial u^n}{\partial y} + w^n\frac{\partial u^{n+1}}{\partial z} \\ = -\theta\frac{\partial p^{n+1}}{\partial x} - (1-\theta)\frac{\partial p^n}{\partial x} + \nu_h\left(\frac{\partial^2 u^n}{\partial x^2} + \frac{\partial^2 u^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial u^{n+1}}{\partial z}\right),\end{aligned} \tag{2.11}$$

$$\begin{aligned}\frac{v^{n+1}-v^n}{\Delta t} + u^n\frac{\partial v^n}{\partial x} + v^n\frac{\partial v^n}{\partial y} + w^n\frac{\partial v^{n+1}}{\partial z} \\ = -\theta\frac{\partial p^{n+1}}{\partial y} - (1-\theta)\frac{\partial p^n}{\partial y} + \nu_h\left(\frac{\partial^2 v^n}{\partial x^2} + \frac{\partial^2 v^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial v^{n+1}}{\partial z}\right),\end{aligned} \tag{2.12}$$

$$\begin{aligned}\frac{w^{n+1}-w^n}{\Delta t} + u^n\frac{\partial w^n}{\partial x} + v^n\frac{\partial w^n}{\partial y} + w^n\frac{\partial w^{n+1}}{\partial z} \\ = -\theta\frac{\partial p^{n+1}}{\partial z} - (1-\theta)\frac{\partial p^n}{\partial z} + \nu_h\left(\frac{\partial^2 w^n}{\partial x^2} + \frac{\partial^2 w^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial w^{n+1}}{\partial z}\right) - g.\end{aligned} \tag{2.13}$$

The decisions about location of the advective and viscous terms in time were made in Bijvelds [2001]. The reason for the implicit treatment of the vertical derivatives is that usually Δz is very small and, therefore, these terms cause the strongest stability restrictions. The pressure terms are included using a θ-scheme. The pressure-correction Δp is defined as the difference between the pressure at the new time level and the pressure at the old time level. It is given as

$$\Delta p = p^{n+1} - p^n. \tag{2.14}$$

Substitution of Δp in the momentum equations (2.11), (2.12) and (2.13) yields

$$\begin{aligned}
&\frac{u^{n+1}-u^n}{\Delta t} + u^n\frac{\partial u^n}{\partial x} + v^n\frac{\partial u^n}{\partial y} + w^n\frac{\partial u^{n+1}}{\partial z} \\
&\qquad = -\theta\frac{\partial \Delta p}{\partial x} - \frac{\partial p^n}{\partial x} + \nu_h\left(\frac{\partial^2 u^n}{\partial x^2} + \frac{\partial^2 u^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial u^{n+1}}{\partial z}\right), \\
&\frac{v^{n+1}-v^n}{\Delta t} + u^n\frac{\partial v^n}{\partial x} + v^n\frac{\partial v^n}{\partial y} + w^n\frac{\partial v^{n+1}}{\partial z} \\
&\qquad = -\theta\frac{\partial \Delta p}{\partial y} - \frac{\partial p^n}{\partial y} + \nu_h\left(\frac{\partial^2 v^n}{\partial x^2} + \frac{\partial^2 v^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial v^{n+1}}{\partial z}\right), \\
&\frac{w^{n+1}-w^n}{\Delta t} + u^n\frac{\partial w^n}{\partial x} + v^n\frac{\partial w^n}{\partial y} + w^n\frac{\partial w^{n+1}}{\partial z} \\
&\qquad = -\theta\frac{\partial \Delta p}{\partial z} - \frac{\partial p^n}{\partial z} + \nu_h\left(\frac{\partial^2 w^n}{\partial x^2} + \frac{\partial^2 w^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial w^{n+1}}{\partial z}\right) - g.
\end{aligned}$$

The pressure-correction has to be calculated in such a way that the continuity requirement is fulfilled. Since the pressure-correction is accounted for in a second step, we exclude Δp from the momentum equation and replace the final velocities u^{n+1}, v^{n+1} and w^{n+1} with preliminary velocities u^{n+1*}, v^{n+1*} and w^{n+1*}, which yields

$$\begin{aligned}
&\frac{u^{n+1*}-u^n}{\Delta t} + u^n\frac{\partial u^n}{\partial x} + v^n\frac{\partial u^n}{\partial y} + w^n\frac{\partial u^{n+1*}}{\partial z} \\
&\qquad = -\frac{\partial p^n}{\partial x} + \nu_h\left(\frac{\partial^2 u^n}{\partial x^2} + \frac{\partial^2 u^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial u^{n+1*}}{\partial z}\right), &(2.15)\\
&\frac{v^{n+1*}-v^n}{\Delta t} + u^n\frac{\partial v^n}{\partial x} + v^n\frac{\partial v^n}{\partial y} + w^n\frac{\partial v^{n+1*}}{\partial z} \\
&\qquad = -\frac{\partial p^n}{\partial y} + \nu_h\left(\frac{\partial^2 v^n}{\partial x^2} + \frac{\partial^2 v^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial v^{n+1*}}{\partial z}\right), &(2.16)\\
&\frac{w^{n+1*}-w^n}{\Delta t} + u^n\frac{\partial w^n}{\partial x} + v^n\frac{\partial w^n}{\partial y} + w^n\frac{\partial w^{n+1*}}{\partial z} \\
&\qquad = -\frac{\partial p^n}{\partial z} + \nu_h\left(\frac{\partial^2 w^n}{\partial x^2} + \frac{\partial^2 w^n}{\partial y^2}\right) + \frac{\partial}{\partial z}\left(\nu_v\frac{\partial w^{n+1*}}{\partial z}\right) - g. &(2.17)
\end{aligned}$$

Using the results of the momentum prediction equations, the pressure-correction Δp is calculated with the pressure-correction equation that will be derived in section 2.5. Subsequently, the velocity at the new time level is obtained using the momentum correction equations

$$\frac{u^{n+1}-u^{n+1*}}{\Delta t} = -\theta\frac{\partial \Delta p}{\partial x}, \tag{2.18}$$

$$\frac{v^{n+1}-v^{n+1*}}{\Delta t} = -\theta\frac{\partial \Delta p}{\partial y}, \tag{2.19}$$

$$\frac{w^{n+1}-w^{n+1*}}{\Delta t} = -\theta\frac{\partial \Delta p}{\partial z}. \tag{2.20}$$

Note that for the final determination of w^{n+1}, instead of using the momentum correction equation directly, an alternative technique based on the depth-integrated continuity equation may be applied, as described in section 2.6.

Having considered the time discretization of the momentum prediction and correction equations, it is also necessary to introduce the discretization in space. The discretization of these terms has not changed since the hydrostatic model in z-grid formulation was introduced in Bijvelds [2001]. Therefore, only a brief summary is given here and the reader is referred to the reference for details.

In the momentum prediction equations the horizontal advection terms are discretized using a first order multi-directional upwind scheme. The vertical advection terms and all viscosity terms are discretized with second order central schemes. The pressure is included with a central discretization as well. The spatial discretization of the momentum correction equations can be found at the beginning of section 2.5, because they are a starting point for the derivation of the pressure-correction equation.

2.4 Continuity equation

The pressure-correction is calculated in a way that the continuity equation (2.4) is fulfilled. Therefore, we have to discretize the continuity equation in space, which is done using the finite volume method. We express the velocity as a vector $\vec{u}$, integrate the continuity equation over the water volume $V_{i,j,k}$ within a cell with indices (i, j, k) and apply the divergence theorem:

$$0 = \int_{V_{i,j,k}} \nabla \cdot \vec{u} \, \mathrm{d}V = \oint_{S_{i,j,k}} \vec{u} \cdot \vec{n} \, \mathrm{d}S.$$

For interior cells a straightforward discretization of the fluxes through the volume areas $S_{i,j,k}$ gives

$$\begin{aligned} 0 = {} & u_{i+1/2,j,k}\Delta y_j \Delta z_k \\ & - u_{i-1/2,j,k}\Delta y_j \Delta z_k \\ & + v_{i,j+1/2,k}\Delta x_i \Delta z_k \\ & - v_{i,j-1/2,k}\Delta x_i \Delta z_k \\ & + w_{i,j,k+1/2}\Delta x_i \Delta y_j \\ & - w_{i,j,k-1/2}\Delta x_i \Delta y_j, \qquad k^{i,j}_{\min} < k < k^{i,j,n}_{\max}, \end{aligned}$$

because all volume faces are aligned with the coordinate axes.

For boundary cells, however, the discretization is more complicated. We start with a cell that contains both boundaries, the free surface and the bottom. Projected to the x-y-plane the cell is rectangular and aligned to the coordinate axes. However, at the bottom and at the surface the volume is bounded by functions in x-y-space. If we define the interface velocities and normal vectors at the surface and the bottom as

$$\vec{u}|_\zeta = \begin{pmatrix} u(x,y,\zeta) \\ v(x,y,\zeta) \\ w(x,y,\zeta) \end{pmatrix}, \quad \vec{S}|_\zeta = \begin{pmatrix} -\frac{\partial \zeta}{\partial x} \\ -\frac{\partial \zeta}{\partial y} \\ 1 \end{pmatrix}, \quad \vec{u}|_{-d} = \begin{pmatrix} u(x,y,-d) \\ v(x,y,-d) \\ w(x,y,-d) \end{pmatrix}, \quad \vec{S}|_{-d} = \begin{pmatrix} \frac{\partial d}{\partial x} \\ \frac{\partial d}{\partial y} \\ 1 \end{pmatrix},$$

then the integral over the volume faces is given by

$$\begin{aligned}
\oint_{S_{i,j,k}} \vec{u}\cdot\vec{n}\,\mathrm{d}S = &\int_{y_{j-1/2}}^{y_{j+1/2}}\int_{-d(x_{i+1/2},y)}^{\zeta(x_{i+1/2},y,t)} u(x_{i+1/2},y,z)\,\mathrm{d}y\,\mathrm{d}z \\
&- \int_{y_{j-1/2}}^{y_{j+1/2}}\int_{-d(x_{i-1/2},y)}^{\zeta(x_{i-1/2},y,t)} u(x_{i-1/2},y,z)\,\mathrm{d}y\,\mathrm{d}z \\
&+ \int_{x_{i-1/2}}^{x_{i+1/2}}\int_{-d(x,y_{i+1/2})}^{\zeta(x,y_{i+1/2},t)} v(x,y_{i+1/2},z)\,\mathrm{d}x\,\mathrm{d}z \\
&- \int_{x_{i-1/2}}^{x_{i+1/2}}\int_{-d(x,y_{i+1/2})}^{\zeta(x,y_{i-1/2},t)} v(x,y_{i-1/2},z)\,\mathrm{d}x\,\mathrm{d}z \\
&+ \int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}} \vec{u}|_\zeta\cdot\vec{S}|_\zeta\,\mathrm{d}x\,\mathrm{d}y \\
&- \int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}} \vec{u}|_{-d}\cdot\vec{S}|_{-d}\,\mathrm{d}x\,\mathrm{d}y \qquad (2.21)\\
&k^{i,j}_{\min} = k = k^{i,j,n}_{\max}.
\end{aligned}$$

We can substitute the kinematic boundary conditions (2.5) and (2.6) in the last two integrals of equation (2.21) to obtain

$$\int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}} \vec{u}|_\zeta\cdot\vec{S}|_\zeta\,\mathrm{d}x\,\mathrm{d}y = \int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}} \frac{\partial\zeta}{\partial t}\,\mathrm{d}x\,\mathrm{d}y,$$
$$\int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}} \vec{u}|_{-d}\cdot\vec{S}|_{-d}\,\mathrm{d}x\,\mathrm{d}y = 0.$$

If we now spatially discretize all variables in (2.21), we obtain

$$\begin{aligned}
0 = {} & u_{i+1/2,j,k}\Delta y_j\Delta z_{i+1/2,j,k} - u_{i-1/2,j,k}\Delta y_j\Delta z_{i-1/2,j,k} \\
& + v_{i,j+1/2,k}\Delta x_i\Delta z_{i,j+1/2,k} - v_{i,j-1/2,k}\Delta x_i\Delta z_{i,j-1/2,k} + \Delta x_i\Delta y_j\frac{\partial\zeta}{\partial t}, \qquad k^{i,j}_{\min} = k = k^{i,j,n}_{\max}.
\end{aligned}$$

It shall be mentioned once again that we have assumed an orthogonal grid, so the horizontal grid spacings have only a single index. This makes it possible that we can divide the equation by $\Delta x_i\Delta y_j$ to get

$$\begin{aligned}
0 = {} & \frac{u_{i+1/2,j,k}\Delta z_{i+1/2,j,k} - u_{i-1/2,j,k}\Delta z_{i-1/2,j,k}}{\Delta x_i} \\
& + \frac{v_{i,j+1/2,k}\Delta z_{i,j+1/2,k} - v_{i,j-1/2,k}\Delta z_{i,j-1/2,k}}{\Delta y_j} + \frac{\partial\zeta_{i,j}}{\partial t} \qquad k^{i,j}_{\min} = k = k^{i,j,n}_{\max}.
\end{aligned}$$

For brevity we introduce the operators

$$D_x(u) \equiv \frac{u_{i+1/2,j,k}\Delta z_{i+1/2,j,k} - u_{i-1/2,j,k}\Delta z_{i-1/2,j,k}}{\Delta x_i}, \qquad (2.22)$$

$$D_y(v) \equiv \frac{v_{i,j+1/2,k}\Delta z_{i,j+1/2,k} - v_{i,j-1/2,k}\Delta z_{i,j-1/2,k}}{\Delta y_j} \qquad (2.23)$$

to obtain

$$0 = D_x(u) + D_y(v) + \frac{\partial\zeta_{i,j}}{\partial t}, \qquad k^{i,j}_{\min} = k = k^{i,j,n}_{\max}.$$

Note that, by definitions (2.9) and (2.10), the operators D_x and D_y are continuously dependent on time, but the decision about what equation is applicable is done at the beginning of each time step.

In a similar way as shown above, we can use only one of the kinematic conditions (2.5) or (2.6) for cells that contain either the surface or the bottom. The full set of equations for all interior and boundary cells is given as

$$0 = D_x(u) + D_y(v) + w_{i,j,k+1/2} - w_{i,j,k-1/2}, \qquad k^{i,j}_{\min} < k < k^{i,j,n}_{\max}, \tag{2.24}$$

$$0 = D_x(u) + D_y(v) + w_{i,j,k+1/2}, \qquad k^{i,j}_{\min} = k < k^{i,j,n}_{\max}, \tag{2.25}$$

$$0 = D_x(u) + D_y(v) - w_{i,j,k-1/2} + \partial\zeta_{i,j}/\partial t, \qquad k^{i,j}_{\min} < k = k^{i,j,n}_{\max}, \tag{2.26}$$

$$0 = D_x(u) + D_y(v) + \partial\zeta_{i,j}/\partial t, \qquad k^{i,j}_{\min} = k = k^{i,j,n}_{\max}, \tag{2.27}$$

We have now defined the spatial discretization of the continuity equation for different vertical locations of a grid cell. The conditions for the side boundaries are not considered in this thesis.

2.5 Pressure-correction equation

In the following we will derive the pressure-correction equations. At the basis is a substitution of the momentum correction equations in the continuity equation to ensure mass conservation at time level $n+1$. The main difficulty of the derivation of the pressure-correction equations is the time derivative of the free surface, which is present in the discretized continuity equations of all surface cells. To handle this surface term, we will apply a dynamic condition at the free surface.

The set of pressure-correction equations will form a system of three-dimensionally coupled equations for the unknown Δp, which is the change of the pressure during one time step. The system of pressure-correction equations will contain centrally discretized second derivatives, which is the reason why the set of pressure-correction equations is often named pressure Poisson equation. Due to the particular treatment of the free surface in the current approach, the system matrix will not be symmetric.

To derive the pressure-correction equations for the considered problem we start with the momentum correction equations (2.18), (2.19) and (2.20) that are discretized with central differences in space,

$$\begin{aligned}
\frac{u^{n+1}_{i+1/2,j,k} - u^{n+1*}_{i+1/2,j,k}}{\Delta t} &= -\theta \frac{\Delta p_{i+1,j,k} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta x_i + \Delta x_{i+1})}, \\
\frac{v^{n+1}_{i,j+1/2,k} - v^{n+1*}_{i,j+1/2,k}}{\Delta t} &= -\theta \frac{\Delta p_{i,j+1,k} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta y_j + \Delta y_{j+1})}, \\
\frac{w^{n+1}_{i,j,k+1/2} - w^{n+1*}_{i,j,k+1/2}}{\Delta t} &= -\theta \frac{\Delta p_{i,j,k+1} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta z^n_{i,j,k} + \Delta z^n_{i,j,k+1})}.
\end{aligned}$$

We rewrite the equations to have the new velocities at the left hand side,

$$u^{n+1}_{i+1/2,j,k} = -\Delta t\theta \frac{\Delta p_{i+1,j,k} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta x_i + \Delta x_{i+1})} + u^{n+1*}_{i+1/2,j,k}, \tag{2.28}$$

$$v^{n+1}_{i,j+1/2,k} = -\Delta t\theta \frac{\Delta p_{i,j+1,k} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta y_j + \Delta y_{j+1})} + v^{n+1*}_{i,j+1/2,k}, \tag{2.29}$$

$$w^{n+1}_{i,j,k+1/2} = -\Delta t\theta \frac{\Delta p_{i,j,k+1} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta z^n_{i,j,k} + \Delta z^n_{i,j,k+1})} + w^{n+1*}_{i,j,k+1/2}. \tag{2.30}$$

In the following we want to apply the discrete continuity equations of last section, which we still need to discretize in time. Based on (2.22) and (2.23) we introduce the time-discretized operators

$$D_x^n(u) \equiv \frac{u_{i+1/2,j,k}\Delta z^n_{i+1/2,j,k} - u_{i-1/2,j,k}\Delta z^n_{i-1/2,j,k}}{\Delta x_i}$$
$$D_y^n(v) \equiv \frac{v_{i,j+1/2,k}\Delta z^n_{i,j+1/2,k} - v_{i,j-1/2,k}\Delta z^n_{i,j-1/2,k}}{\Delta y_j},$$

where the Δz^n with half indices are given as

$$\Delta z^n_{i\pm1/2,j,k} = \min(\Delta z^n_{i,j,k}, \Delta z^n_{i\pm1,j,k}), \qquad \Delta z_{i,j\pm1/2,k} = \min(\Delta z^n_{i,j,k}, \Delta z^n_{i,j\pm1,k}), \tag{2.31}$$

where

$$\Delta z^n_{i,j,k} = \min(z_{k+1/2}, \zeta^n_{i,j}) - \max(z_{k-1/2}, -d_{i,j}).$$

The discretized continuity equation for interior cells is given by equation (2.24), which is formulated here for velocity values at the new time level $n+1$, but with the water level fixed at the old time level:

$$0 = D_x^n(u^{n+1}) + D_y^n(v^{n+1}) + w^{n+1}_{i,j,k+1/2} - w^{n+1}_{i,j,k-1/2}, \qquad k^{i,j}_{\min} < k < k^{i,j,n}_{\max}. \tag{2.32}$$

The substitution of (2.28), (2.29) and (2.30) in (2.32) yields the *pressure-correction equation for internal cells*,

$$\begin{aligned}\Delta t\theta\Big(D^n_{xx}(\Delta p) + D^n_{yy}(\Delta p) + \frac{\Delta p_{i,j,k+1} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta z^n_{i,j,k} + \Delta z^n_{i,j,k+1})} - \frac{\Delta p_{i,j,k} - \Delta p_{i,j,k-1}}{\frac{1}{2}(\Delta z^n_{i,j,k-1} + \Delta z^n_{i,j,k})}\Big) \\ = D_x^n(u^{n+1*}) + D_y^n(v^{n+1*}) + w^{n+1*}_{i,j,k+1/2} + w^{n+1*}_{i,j,k-1/2}, \qquad k^{i,j}_{\min} < k < k^{i,j,n}_{\max},\end{aligned} \tag{2.33}$$

where we have applied the operators D_x^n and D_y^n to u^{n+1} and v^{n+1} so that

$$D_x^n(u^{n+1}) = -\Delta t\theta D^n_{xx}(\Delta q^*) + D_x^n(u^{n+1*}),$$
$$D_y^n(v^{n+1}) = -\Delta t\theta D^n_{yy}(\Delta q^*) + D_y^n(v^{n+1*}),$$

with

$$D^n_{xx}(\Delta p) = \frac{\Delta p_{i+1,j,k} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta x_i + \Delta x_{i+1})}\frac{\Delta z^n_{i+1/2,j,k}}{\Delta x_i} + \frac{\Delta p_{i,j,k} - \Delta p_{i-1,j,k}}{\frac{1}{2}(\Delta x_{i-1} + \Delta x_i)}\frac{\Delta z^n_{i-1/2,j,k}}{\Delta x_i},$$
$$D^n_{yy}(\Delta p) = \frac{\Delta p_{i,j+1,k} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta y_j + \Delta y_{j+1})}\frac{\Delta z^n_{i,j+1/2,k}}{\Delta y_j} + \frac{\Delta p_{i,j,k} - \Delta p_{i,j-1,k}}{\frac{1}{2}(\Delta y_{j-1} + \Delta y_j)}\frac{\Delta z^n_{i,j-1/2,k}}{\Delta y_j}.$$

Similarly, the substitution of the momentum correction equations (2.28), (2.29) and (2.30) in the discretized continuity equation (2.25) at time level $n+1$ gives the *pressure-correction equation for bottom cells*

$$\begin{aligned}\Delta t\theta\Big(D^n_{xx}(\Delta p) + D^n_{yy}(\Delta p) + \frac{\Delta p_{i,j,k+1} - \Delta p_{i,j,k}}{\frac{1}{2}(\Delta z^n_{i,j,k} + \Delta z^n_{i,j,k+1})}\Big) \\ = D_x^n(u^{n+1*}) + D_y^n(v^{n+1*}) + w^{n+1*}_{i,j,k+1/2}, \qquad k^{i,j}_{\min} = k < k^{i,j,n}_{\max}.\end{aligned} \tag{2.34}$$

At the free surface we use a θ-scheme for the time discretization of $\partial\zeta/\partial t$ in the discretized continuity equation (2.26) to obtain

$$\begin{aligned}-\frac{\zeta^{n+1}_{i,j} - \zeta^n_{i,j}}{\Delta t} &= \theta_\zeta(D_x^n(u^{n+1}) + D_y^n(v^{n+1}) - w^{n+1}_{i,j,k-1/2}) \\ &\quad + (1-\theta_\zeta)(D_x^n(u^n) + D_y^n(v^n) - w^n_{i,j,k-1/2}), \qquad k^{i,j}_{\min} < k = k^{i,j,n}_{\max}.\end{aligned} \tag{2.35}$$

After substitution of the momentum correction equations (2.28), (2.29) and (2.30) we arrive at a preliminary form of the pressure-correction equation for free-surface cells,

$$\begin{aligned} -\frac{\zeta_{i,j}^{n+1}-\zeta_{i,j}^{n}}{\Delta t}+\theta_\zeta\Delta t\theta\Big(D_{xx}^n(\Delta p)+D_{yy}^n(\Delta p)-\frac{\Delta p_{i,j,k}-\Delta p_{i,j,k-1}}{\frac{1}{2}(\Delta z_{i,j,k-1}^n+\Delta z_{i,j,k}^n)}\Big) \\ =\theta_\zeta\Big(D_x^n(u^{n+1*})+D_y^n(v^{n+1*})+w_{i,j,k-1/2}^{n+1*}\Big) \\ +(1-\theta_\zeta)\Big(D_x^n(u^n)+D_y^n(v^n)+w_{i,j,k-1/2}^n\Big),\qquad k_{\min}^{i,j}<k=k_{\max}^{i,j,n}. \end{aligned} \tag{2.36}$$

Besides the actual unknown Δp, equation (2.36) still contains the surface elevation $\zeta_{i,j}^{n+1}$, which is an unknown, too. To eliminate the surface elevation from the pressure-correction equation we discretize the dynamic condition $p|_\zeta = 0$ (see equation (2.8) on page 12) at time level $n+1$ to get

$$0=p^{n+1}|_{\zeta^{n+1}} \tag{2.37}$$

and linearize it, so that we get an expression containing $\zeta^{n+1}-\zeta^n$. This can be done in different ways, two of which are shown in the following. We will express the pressure in components $p = g(\zeta - z) + q$ to be able to compare the two approaches easily with each other. It is important to note that it is possible to work without the non-hydrostatic pressure variable q, as will be shown in appendix B.

A commonly used approximation of equation (2.37) is the hydrostatic assumption in the surface cells, which is obtained by linearizing around $z = z_k$ and neglecting the non-hydrostatic pressure gradient at the free surface:

$$\begin{aligned} 0=p^{n+1}|_{\zeta^{n+1}} &\approx p^{n+1}|_{z_k}+(\zeta^{n+1}-z_k)\frac{\partial p^{n+1}}{\partial z}|_{z_k} \\ &\approx g(\zeta^{n+1}-z_k)+q^{n+1}|_{z_k}+(\zeta^{n+1}-z_k)\Big(-g+\frac{\partial q^{n+1}}{\partial z}|_{z_k}\Big) \\ &\approx q^{n+1}|_{z_k},\qquad k=k_{\max}^{i,j,n} \end{aligned} \tag{2.38}$$

where the largest error we make is the neglect of $(\zeta^{n+1}-z_k)\frac{\partial q^{n+1}}{\partial z}|_{z_k}$, which is of order Δz. In terms of discretized pressure variables we can write the linearized pressure condition at the surface as

$$q_{i,j,k}^{n+1}=0\qquad k=k_{\max}^{i,j,n}. \tag{2.39}$$

This approach is used in all non-hydrostatic models known to the author, for example, in particular in Casulli [1999].

Although the non-hydrostatic pressure at the surface is zero by definition, the *gradient* of the non-hydrostatic pressure can be far from zero in strongly non-hydrostatic applications. Therefore, a better approximation of the pressure distribution will be rewarding. We start by linearizing the pressure condition at the new time level around the water level at the old time level, which yields

$$\begin{aligned} 0=p^{n+1}|_{\zeta^{n+1}} &\approx p^{n+1}|_{\zeta^n}+(\zeta^{n+1}-\zeta^n)\frac{\partial p^{n+1}}{\partial z}|_{\zeta^n} \\ &\approx q^{n+1}|_{\zeta^n}+g(\zeta^{n+1}-\zeta^n)+(\zeta^{n+1}-\zeta^n)\frac{\partial p^{n+1}}{\partial z}|_{\zeta^n}, \\ &\qquad k_{\min}^{i,j}<k=k_{\max}^{i,j,n}. \end{aligned} \tag{2.40}$$

We have pressure variables only at discrete locations along the z-axis. Therefore we realize the non-hydrostatic pressure at the free surface with a linear inter-/extrapolation using the

non-hydrostatic pressure of the cell below the surface cell,

$$q^{n+1}|_{\zeta^n} \approx q^{n+1}|_{z_k} + \frac{z_k - \zeta^n}{z_k - z_{k-1}} \left(q^{n+1}|_{z_{k-1}} - q_k^{n+1}|_{z_k} \right), \qquad k_{\min}^{i,j} < k = k_{\max}^{i,j,n}.$$

The respective geometry is sketched in figure 2.2. For the second term we use a similar approximation as in equation (2.38) above,

$$(\zeta^{n+1} - \zeta^n) \frac{\partial p^{n+1}}{\partial z}|_{\zeta^n} = (\zeta^{n+1} - \zeta^n) \Big(-g + \frac{\partial q^{n+1}}{\partial z}|_{\zeta^n} \Big) \approx -g(\zeta^{n+1} - \zeta^n), \qquad k_{\min}^{i,j} < k = k_{\max}^{i,j,n},$$

so that the linearized surface pressure condition (2.40) becomes

$$0 = q^{n+1}|_{z_k} + \frac{z_k - \zeta^n}{z_k - z_{k-1}} \left(q^{n+1}|_{z_{k-1}} - q_k^{n+1}|_{z_k} \right), \qquad k_{\min}^{i,j} < k = k_{\max}^{i,j,n}, \tag{2.41}$$

where the largest error we make is the neglect of $(\zeta^{n+1} - \zeta^n) \frac{\partial q^{n+1}}{\partial z}|_{\zeta^n}$, which is of order Δt. Equation (2.41) can be written for discretized variables as

$$\alpha_{i,j}^n q_{i,j,k-1}^{n+1} + (1 - \alpha_{i,j}^n) q_{i,j,k}^{n+1} = 0, \qquad k_{\min}^{i,j} < k = k_{\max}^{i,j,n} \tag{2.42}$$

with

$$\alpha_{i,j}^n = \frac{z_k - \zeta_{i,j}^n}{z_k - z_{k-1}} = \frac{z_{k+1/2} + z_{k-1/2} - 2\zeta_{i,j}^n}{z_{k+1/2} - z_{k-3/2}}, \qquad k_{\min}^{i,j} < k = k_{\max}^{i,j,n}. \tag{2.43}$$

It is easy to verify that by fixing $\alpha_{i,j}^n = 0$ we obtain a formula equivalent to (2.39).

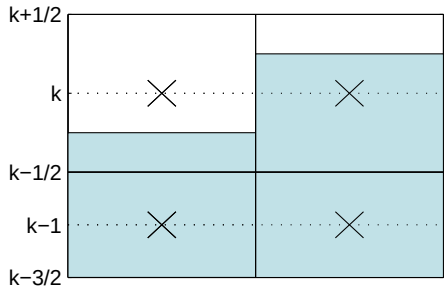

Figure 2.2: Sketch of the surface representation on a non-uniform grid ($k = k_{\max}^{i,j,n}$). Shown are four discretization cells, the centered crosses representing the pressure points.

The aim of using the pressure condition was to eliminate the water level term in the discretized momentum correction equation (2.36), where we have the unknown $\Delta p_{i,j,k}$. According to (2.14) on page 14 we have the discretized relation

$$\Delta p_{i,j,k} = p_{i,j,k}^{n+1} - p_{i,j,k}^n = \zeta_{i,j}^{n+1} + q_{i,j,k}^{n+1} - \zeta_{i,j}^n - q_{i,j,k}^n, \tag{2.44}$$

which gives the following relation for q^{n+1}:

$$q_{i,j,k}^{n+1} = \Delta p_{i,j,k} - \zeta_{i,j}^{n+1} + \zeta_{i,j}^n + q_{i,j,k}^n.$$

Substitution in (2.42) and rearranging gives

$$\zeta^{n+1} - \zeta^n = \frac{\alpha_{i,j}^n}{g} (\Delta p_{i,j,k-1} + p_{i,j,k-1}^{n+1}) + \frac{(1 - \alpha_{i,j}^n)}{g} (\Delta p_{i,j,k} + p_{i,j,k}^{n+1})$$

Now, finally, we are able to eliminate the water levels from (2.36) to obtain the *pressure-correction equation for surface cells*

$$\begin{aligned}
&-\frac{\alpha^n_{i,j}\Delta p_{i,j,k-1}+(1-\alpha^n_{i,j})\Delta p_{i,j,k}}{g\Delta t}\\
&\quad+\theta_\zeta\Delta t\theta\Big(D^n_{xx}(\Delta p)+D^n_{yy}(\Delta p)-\frac{\Delta p_{i,j,k}-\Delta p_{i,j,k-1}}{\frac{1}{2}(\Delta z^n_{i,j,k-1}+\Delta z^n_{i,j,k})}\Big)\\
&=\theta_\zeta\Big(D^n_x(u^{n+1*})+D^n_y(v^{n+1*})+w^{n+1*}_{i,j,k-1/2}\Big)\\
&\quad+(1-\theta_\zeta)\Big(D^n_x(u^n)+D^n_y(v^n)+w^n_{i,j,k-1/2}\Big)\\
&\quad+\frac{\alpha^n_{i,j}q^n_{i,j,k-1}+(1-\alpha^n_{i,j})q^n_{i,j,k}}{g\Delta t},\qquad k^{i,j}_{\min}<k=k^{i,j,n}_{\max}.
\end{aligned}\tag{2.45}$$

In cells that contain both the free surface and the bottom, we have to use the hydrostatic pressure relation (2.39). We can not use (2.42) and (2.43), as done for regular surface cells, because no pressure points are available at position $k^{i,j,n}_{\max}-1$ if the surface cell is a bottom cell at the same time. By performing a similar derivation as for regular surface cells, or by using (2.45) and setting $\alpha=0$, we obtain the *pressure-correction equation for cells that contain the bottom and the surface*:

$$\begin{aligned}
&-\frac{\Delta q^*_{i,j,k}}{g\Delta t}+\theta_\zeta\Delta t\theta\Big(D^n_{xx}(\Delta q^*)+D^n_{yy}(\Delta q^*)\Big)\\
&\qquad=\theta_\zeta\Big(D^n_x(u^{n+1*})+D^n_y(v^{n+1*})\Big)\\
&\qquad\quad+(1-\theta_\zeta)\Big(D^n_x(u^n)+D^n_y(v^n)\Big)+\frac{q^n_{i,j,k}}{g\Delta t},\qquad k^{i,j}_{\min}=k=k^{i,j,n}_{\max}.
\end{aligned}\tag{2.46}$$

The three-dimensionally coupled system of pressure-correction equations for the whole interior domain is formed by the set of equations (2.33), (2.34), (2.45) and (2.46). This is a system of equations with a number of unknowns that is equal to the number of non-empty grid cells. The unknowns are coupled with their neighbors in x-, y- and z-direction. When the unknowns are ordered lexicographically, then the matrix of the system of equations is sparse with a banded structure. The term containing α on the left hand side introduces an asymmetry in the otherwise symmetric system. In section 3 a method is explained that is able to solve the system of equations in an efficient way.

2.6 Update of variables

Once the pressure-correction Δp is calculated with a sufficient precision, all other unknowns at time step $n+1$ can be computed. The horizontal velocities are calculated using the first two discretized momentum correction equations (2.28) and (2.29). The surface elevation is obtained from (2.35), the pressure and the non-hydrostatic pressure component from (2.44). The vertical velocity may be computed by using a momentum equation, which we will call method A, or by using a continuity equation, which we will call method B.

In method A the momentum equation (2.30) is applied to compute the new vertical velocities. As in the other two momentum correction equations, the pressure-correction Δp is used to determine the velocity. The pressure-correction is the outcome of an iterative solution method, which introduces some convergence error. Assume now that $\Delta p_{i,j,k}=\Delta\tilde{p}_{i,j,k}+\epsilon_{i,j,k}$, where $\Delta\tilde{p}_{i,j,k}$ is the solution that makes the velocity field divergence free and $\epsilon_{i,j,k}$ is the related error

in each cell. When all three momentum equations are used to update the velocity field, it will not be divergence free and the error will be more or less randomly distributed throughout the whole domain. The error may sum up during time stepping, but provided the scheme is stable and the error sufficiently small, it will be damped out before it can be observed by inspecting the velocity field. However, every non-zero divergence leads to production or destruction of mass, an effect that is usually unwanted for transport simulations.

Method B uses the set of discretized continuity equations (2.25) and (2.24) to obtain the vertical velocities at the top faces of all interior and bottom cells, going step by step from $k^{i,j}_{\min}$ to $k^{i,j,n}_{\max-1}$. The velocities at the top of the surface cells are no unknowns of the system, so they do not need to be calculated. By using the discretized continuity equations, a strict mass conservation is preserved in all cells. The errors in the flow field, however, are accumulated from bottom to top, so that the cells near surface cells may contain a much larger error than cells near the bottom, especially, when also accumulation over time is considered.

It is suggested that method B is used whenever strict mass conservation is required, while method A is expected to give better results for pure flow- and wave-simulations. In the computations shown in this thesis approach B is used if not mentioned otherwise, and the stopping criterion of the iterative solver is set small enough so that the convergence error does not have any noticeable effect on the solution.

2.7 Discussion

We have seen in the description of the numerical scheme that the pressure was treated as a whole in all interior and bottom cells and that only a higher-order non-hydrostatic term was neglected in the surface cells. This is a difference to many non-hydrostatic schemes proposed in the literature, which use a pressure splitting throughout the domain. In the following some non-hydrostatic schemes, and the applied splittings, are explained. As a basis we use a time-discretized momentum equation with the pressure terms included using a θ-scheme. For generality, the advective and viscous terms are simplified as functions of velocities at the old an the new time level. This gives

$$\frac{u^{n+1}-u^n}{\Delta t} = -\theta\left(g\frac{\partial \zeta^{n+1}}{\partial x}+\frac{\partial q^{n+1}}{\partial x}\right)-(1-\theta)\left(g\frac{\partial \zeta^{n}}{\partial x}+\frac{\partial q^{n}}{\partial x}\right)+a(u^{n+1})+b(u^n) \qquad (2.47)$$

The schemes of Casulli and Stelling [1998] and Mahadevan et al. [1996b] are based on a hydrostatic assumption in the prediction step. The predicted velocity field is corrected by the non-hydrostatic pressure q, which itself is computed in a way that the final velocity field fulfills the continuity equation. The split equations, based on the decoupling of the hydrostatic and non-hydrostatic pressure components, are given by

$$\begin{aligned}\frac{u^{n+1*}-u^n}{\Delta t} &= -\theta g\frac{\partial \zeta^{n+1}}{\partial x}-(1-\theta)g\frac{\partial \zeta^{n}}{\partial x}+a(u^{n+1*})+b(u^n)\\ \frac{u^{n+1}-u^{n+1*}}{\Delta t} &= -\frac{\partial q^{n+1}}{\partial x}.\end{aligned}$$

The scheme uses $\theta = 1$ for the implementation of the non-hydrostatic pressure q. The surface elevation ζ^{n+1} is calculated from a depth-integrated continuity equation without using information about the non-hydrostatic pressure. The term $a(u^{n+1})$ is equivalent to the vertical viscous term in this particular scheme. We get some idea about the splitting error if we sum the two equations:

$$\frac{u^{n+1}-u^n}{\Delta t} = -\theta g\frac{\partial \zeta^{n+1}}{\partial x}-(1-\theta)g\frac{\partial \zeta^{n}}{\partial x}-\frac{\partial q^{n+1}}{\partial x}+a(u^{n+1*})+b(u^n)$$

We see that the splitting error is depending on the differences $u^{n+1} - u^{n+1*}$. If $u^{n+1} = u^{n+1*}$, then there is no splitting error. But this can only occur when the non-hydrostatic pressure is constant. In fact, the non-hydrostatic pressure has to be zero, because of the pressure boundary condition at the surface.

Casulli [1999] and Busnelli [2001] included the non-hydrostatic pressure of the last time step in the momentum prediction equations. The split equations are given as

$$\begin{aligned}
\frac{u^{n+1*} - u^n}{\Delta t} &= -\theta g \frac{\partial \zeta^{n+1*}}{\partial x} - (1-\theta)\left(g\frac{\partial \zeta^n}{\partial x} + \frac{\partial q^n}{\partial x}\right) + a(u^{n+1*}) + b(u^n) \\
\frac{u^{n+1} - u^{n+1*}}{\Delta t} &= -\theta \frac{\partial q^*}{\partial x}, \qquad q^* = q^{n+1} + g\zeta^{n+1} - g\zeta^{n+1*}.
\end{aligned}$$

The pressure terms are treated completely with a θ-scheme. The unknown q^* in the pressure Poisson equation is the sum of the non-hydrostatic pressure q^{n+1} and a non-hydrostatic update $(g\zeta^{n+1} - g\zeta^{n+1*})$ of the hydrostatic pressure. The water levels are influenced by the non-hydrostatic pressure in this scheme. Still, there is a contribution $a(u^{n+1*})$ of the vertical viscous terms that remains if both equations are summed. Therefore, the splitting error is a still present.

The time discretization of the scheme of Bijvelds [2003] is similar to a θ-scheme with $\theta = 1$. In the first step a simple, merely first-order accurate ADI method is used. In the second step a system of equations is solved for Δq^*, which is the correction $q^{n+1} - q^n$ of the non-hydrostatic pressure plus a non-hydrostatic update $g\zeta^{n+1} - g\zeta^{n+1*}$ of the water level. The scheme is given as

$$\begin{aligned}
\frac{u^{n+1*} - u^n}{\Delta t} &= -g\frac{\partial \zeta^{n+1/2*}}{\partial x} + \frac{\partial q^n}{\partial x} + a(u^{n+1*}) + b(u^n) \\
\frac{u^{n+1} - u^{n+1*}}{\Delta t} &= -\frac{\partial \Delta q^*}{\partial x}, \qquad \Delta q^* = q^{n+1} + g\zeta^{n+1} - q^n - g\zeta^{n+1*}.
\end{aligned}$$

Due to the ADI scheme used in the prediction step, the preliminary surface elevation $\zeta^{n+1/2*}$ is temporally located at a half time step in the u-momentum equation. The non-hydrostatic correction, however, includes the preliminary water level ζ^{n+1*} at the full time step. Therefore, one component of the splitting error is a function of $\zeta^{n+1*} - \zeta^{n+1/2*}$. Some properties of the hydrostatic prediction step are analyzed in appendix C. Since in the scheme both the vertical advective and the vertical viscous terms are discretized implicitly, there is also an error component which is a function of $u^{n+1} - u^{n+1*}$ that grows with the size of the advective and viscous terms. There is an important difference, though, between a pressure-correction method, as used in the Bijvelds scheme, and the fractional step method, as used in Casulli [1999]: In the pressure-correction method the splitting error with respect to the advective and viscous terms is related to

$$u^{n+1} - u^{n+1*} = -\Delta t \frac{\partial \Delta q^*}{\partial x} \approx \Delta t(q^{n+1} - q^n) = O(\Delta t^2),$$

while in the fractional step method we have

$$u^{n+1} - u^{n+1*} = -\Delta t \frac{\partial q^*}{\partial x} \approx \Delta t(q^{n+1}) = O(\Delta t).$$

For this reason it is more advantageous to use a pressure-correction method than a fractional step method. A more thorough investigation about the relation between pressure-correction and fractional step methods, including analysis and test cases, is given in Armfield and Street [2002].

The scheme described in this thesis considers the total pressure, which means that only the

sum of the hydrostatic and non-hydrostatic components appears in the momentum equations:

$$\frac{u^{n+1*} - u^n}{\Delta t} = -g\frac{\partial \zeta^n}{\partial x} - \frac{\partial q^n}{\partial x} + a(u^{n+1*}) + b(u^n)$$
$$\frac{u^{n+1} - u^{n+1*}}{\Delta t} = -\theta\frac{\partial \Delta p}{\partial x}, \qquad \Delta p = q^{n+1} + g\zeta^{n+1} - q^n - g\zeta^n.$$

As in the scheme of Bijvelds, this method includes both vertical advective and vertical viscous terms implicitly and the method is a pressure-correction method. However, the splitting error due to the ADI method is not present anymore. Furthermore, a θ-scheme is used for the pressure terms in the momentum equations and for the continuity equation in cells containing the free surface to obtain up to second order accuracy in time with respect to the pressure terms. A second order spatial discretization of the pressure condition at the free surface is realized. While in all other methods discussed so far the hydrostatic assumption was used in all cells containing the free surface, in the scheme presented in this thesis the pressure is approximated by a linear profile in z-direction.

Chapter 3

Solution of the pressure-correction equation

In the method of Bijvelds [2003] the system of pressure-correction equations was solved with the conjugate gradient (CG) method of Hestenes and Stiefel [1952], using a tridiagonal Jacobi preconditioner for convergence acceleration. If the dynamic boundary condition at the free surface is implemented using a linear interpolation, then the pressure matrix is not symmetric anymore. Since the CG method is not suitable for linear systems with a non-symmetric matrix, a different iterative method had to be chosen. Section 3.1 explains the reasons why the BiCGSTAB method is well suited as an iterative solver in the given framework. To accelerate the iterative solver it is crucial to combine it with a good preconditioner. Section 3.2 relates the Jacobi-based preconditioners already existing in Delft3D-FLOW with preconditioners based on an ILU decomposition. It is explained why a tridiagonal Jacobi preconditioner accelerates the solution of weakly non-hydrostatic problems well and why the ILU decomposition is a more effective preconditioner for strongly non-hydrostatic cases. In section 3.3 possible starting values and stopping criteria for the iterative solution method are discussed. The final implementation of the preconditioned BiCGSTAB method is given in section 3.4.

3.1 Choice of the solution method

A large number of algorithms is available to solve systems of equations with sparse, non-symmetric matrices and, depending on the application, some algorithms perform better than others. Two types of methods are often used in the context of calculating the time-dependent incompressible Navier-Stokes equations: The generalized minimum residual method (GMRES) by Saad and Schultz [1986] and the biconjugate gradient stabilized method (BiCGSTAB) by van der Vorst [1992]. Just as the CG method, they are Krylov subspace methods. These are iterative methods that approximate the solution x of a system $Ax = b$ by a converging sequence of approximations x^i, given some initial guess x^0. The iteration is stopped if a solution is found that satisfies some given error criterion or if a given maximum number of iterations is reached. The x_i are found within a Krylov subspace $K^i(A, r^0) \equiv \text{span}\{r^0, Ar^0, \ldots, A^{i-1}r^0\}$ with the initial residual $r^0 = Ax^0 - b$. In every iteration step i the subspace is extended, and if $x^i - x$ can be minimized in some way within the given $K^i(A, r^0)$, then the approximation x^i comes closer and closer to the true solution. Krylov subspace methods differ in the way they evolve the Krylov subspace and in the way they determine x^i within the subspace.

Krylov methods are not the only way to solve a linear system of equations. There are several alternatives which all have their particular advantages and disadvantages: Direct methods based

on Gaussian elimination solve small systems of equations very efficiently, but because of their work and memory requirements they can not be used for very large systems. Multigrid methods [Hackbusch, 1985, Wesseling, 1992], on the other hand, are well suited for many problems with a very large number of unknowns. Fast Poisson solvers are specialized and very efficient solvers for pressure-correction equations, but their high efficiency is limited to problems with orthogonal coordinates and rectangular domains [Wesseling, 2001]. Basic iterative methods, like the Jacobi and Gauss-Seidel method, have a low convergence rate, so they are not often used directly as a solution method. However, they are easy to implement and they smooth out high-frequency components of the error quickly. This is why they are often used as smoothers within multigrid methods. Solution methods are often combined such that one method serves as an accelerator, preconditioner or smoother of another method.

There is an important advantage of iterative methods over direct methods in the framework of a time-stepping algorithm: Often the solution of the previous time step is a good initial guess for the solution of the next time step. When the solution is smooth and the time step is small, then the method may only need a few iterations until a sufficiently accurate solution is obtained.

An important consideration regarding the method used for the implementation in Delft3D-FLOW was the existence of a CG method that was already tailored to the matrix structure used in the software. It is comparatively simple to change the existing scheme to another Krylov method rather than to a fundamentally different method like multigrid or a direct method.

The most important differences between GMRES and BiCGSTAB are the *convergence* behavior, the amount of *work* per iteration step and the *storage* requirements. Depending on the problem to be solved, one of the methods performs better than the other (see Nachtigal et al. [1992] for a comparison between GMRES and CGS, a predecessor of BiCGSTAB). The algorithms are listed as algorithms 1 and 2.

Algorithm 1 BiCGSTAB method

matrix A and right-hand side b are given
x^0 is an initial guess
$\bar{r}^0 = r^0 = b - Ax^0$
$\rho_{-1} = \alpha_{-1} = \omega_{-1} = 1$
$p^{-1} = v^{-1} = 0$
for $i = 0 \ldots N$ **do**
  $\rho_i = (\bar{r}^0, r^i)$
  $\beta_{i-1} = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$
  $p^i = r^i + \beta_{i-1}(p^{i-1} - \omega_{i-1}v^{i-1})$
  $v^i = Ap^i$
  $\alpha_i = \rho_i/(\bar{r}^0, v^i)$
  $s = r^i - \alpha_i v^i$
  $t = As$
  $\omega_i = (t, s)/(t, t)$
  $x^{i+1} = x^i + \alpha_i p^i + \omega_i s$
  $r^{i+1} = s - \omega_i t$
  if stopping criterion fulfilled **then**
    leave for-loop
  end if
end for

By looking at the algorithm one can see that BiCGSTAB contains two matrix-vector products per iteration step, which are usually the most expensive part of an iteration step. They are

Algorithm 2 GMRES method

1: matrix A and right-hand side b are given
2: x^0 is an initial guess
3: $r^0 = b - Ax^0,\ \beta = ||r^0||_2,\ v^1 = r^0/\beta$
4: create an $(m+1) \times m$ matrix $\bar{H}_m$ with elements $h_{ij} = 0$
5: **for** $j = 1 \ldots m$ **do**
6: $\quad w^j = Av^j$
7: $\quad$ **for** $i = 1 \ldots j$ **do**
8: $\quad\quad h_{ij} = (w^j)^T v^i$
9: $\quad\quad w^j = w^j - h_{ij} v^i$
10: $\quad$ **end for**
11: $\quad h_{j+1,j} = ||w^j||_2$
12: $\quad v^{j+1} = w^j / h_{j+1,j}$
13: **end for**
14: compute y_m that minimizes $||\beta e_1 - \bar{H}_m y||_2$
15: $x_m = x_0 + V_m y_m$

required for the bi-orthogonalization process the method is based on. In the GMRES method only one matrix-vector product is present per iteration step, but there is an inner iteration that loops over all residual vectors computed so far. This part of the solution procedure is a modified Gram-Schmidt process, which orthogonalizes the Krylov subspace at every iteration step using the newly calculated residual. Therefore, all residuals must be stored in memory and j inner products have to be performed at iteration step j, which means that both work and memory requirements increase linearly with the number of iterations.

The convergence behavior of BiCGSTAB and GMRES without preconditioner will be illustrated using a simple model system of equations that is solved using the standard Matlab 7.4 solver routines, with some statements added to allow for timing. We solved a perturbed Poisson equation discretized on a three-dimensional domain consisting of $10 \times 10 \times 10$ cubes, with homogeneous Neumann boundary conditions. The central discretization of a first derivative was added to the central tridiagonal part in order to introduce some asymmetry in the matrix. We chose a standard normally distributed random vector as a right-hand side of the system of equations. In a time-stepping algorithm usually a good initial guess for the solution is available. To simulate this, the initial vector x^0 was composed as $x^0 = x + \sigma n$, where x is a numerical solution of the system, calculated beforehand with a high accuracy, n is a standard normally distributed random vector and σ a small perturbation parameter. The model is *not* intended to represent the characteristics of the actual pressure-correction system introduced in chapter 2, which has a more irregular structure with respect to the asymmetric part of the matrix.

The behavior of the methods is shown in figure 3.1, where the 2-norm of the residual depending on the number of matrix-vector products and depending on time is plotted. It can be observed that the residual decreases monotonically in the case of GMRES, while several spikes appear in the case of BiCGSTAB. Although GMRES reaches the lower limit 10^{-8} after fewer matrix-vector products than BiCGSTAB, it still takes more time. In the first few iterations, however, the GMRES method seems more favorable because of the smoother convergence.

As could be seen in the example, the choice between GMRES and BiCGSTAB with respect to the computational speed depends mainly upon the expected number of iterations. When several iterations are needed to convergence to a solution, then the BiCGSTAB method should be used, otherwise the GMRES method may be a good alternative. There exist several methods to overcome the problem of the increasing work and memory requirements of the GMRES method,

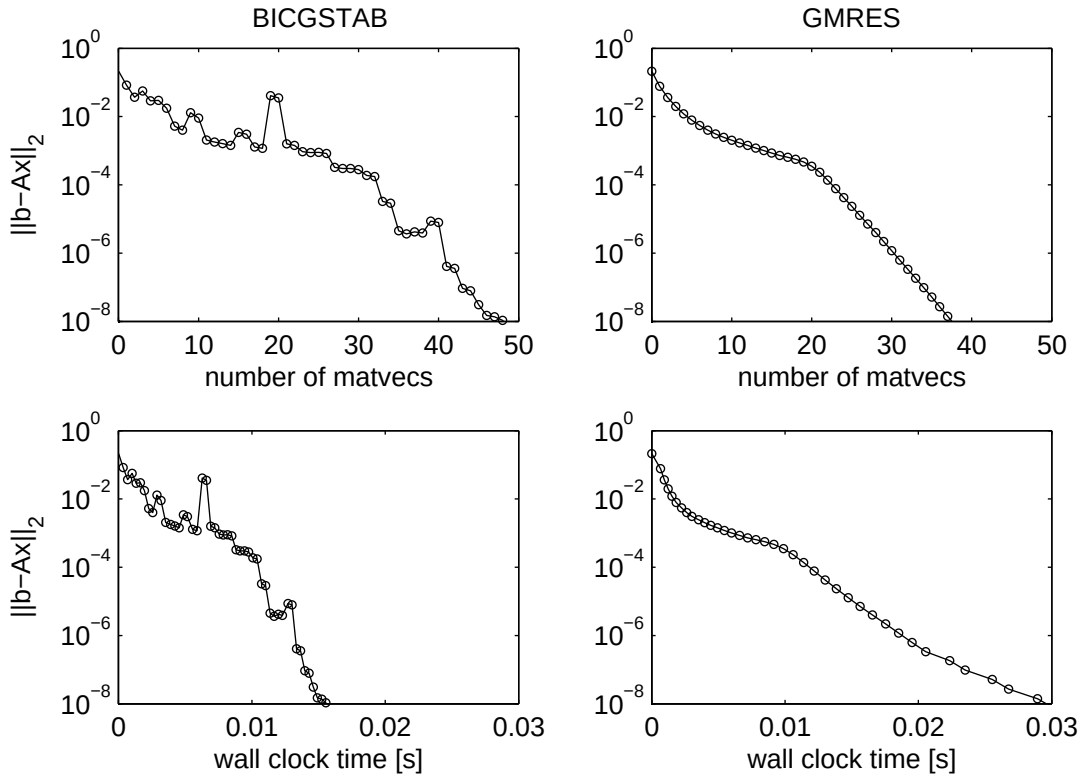

Figure 3.1: Convergence of the BiCGSTAB (left) and the GMRES (right) method on a perturbed Poisson problem. The plots show the 2-norm of the residual depending on the number of matrix vector products (markers, top) and on the wall clock time (bottom). While GMRES has a smoother and a faster convergence in terms of matrix-vector products (top), the BiCGSTAB method reaches a high accuracy in less time because the computational work per iteration step is constant.

based on restarting or truncating the method. These methods, however, usually have a less strong convergence than full GMRES.

In Delft3D-FLOW, the typical number of iterations varies from application to application. A robust method must be implemented that performs well for any typical application. An important consideration in this context is the static memory management of Delft3D-FLOW. Vector memory is allocated only once in the setup of the computation, before the first time step. In this phase a couple of "working arrays" is made available, which can be used for any purpose, but which may later be overwritten by any function. The number of working arrays is sufficient to store all vectors that are needed in the BiCGSTAB method. To get a sufficient number of vectors for the GMRES method, the number of working arrays is too small. Restarting would be necessary after very few iterations. Increasing the number of working arrays is not an option. This is the reason why only the BiCGSTAB method has been implemented as a non-symmetric solver for the pressure-correction equation. In the end, the choice of the preconditioner will be more important for the speed of convergence than the choice of the Krylov method.

3.2 Preconditioners

The convergence of Krylov methods for a system $Ax = b$ depends on the initial guess x_0, the right-hand side b and the eigenvalue spectrum of the system matrix A. Usually the condition number κ of the matrix, which is the ratio between the biggest and the smallest eigenvalue, gives already a good idea about how well the iterative method will perform. An "ill-conditioned" system possesses a system matrix with a large condition number and is characterized by low convergence rates of the iterative solver.

The idea of preconditioning is to replace the linear system $Ax = b$ by the preconditioned system $\tilde{A}\tilde{x} = \tilde{b}$, where $\tilde{A} = B^{-1}AC^{-1}$, $\tilde{x} = Cx$ and $\tilde{b} = B^{-1}b$, which means that the preconditioned system is equivalent to

$$B^{-1}AC^{-1}(Cx) = B^{-1}b.$$

both the preconditioned and the original system have the same solution in terms of x. In order to obtain x form $\tilde{x}$, the system $Cx = \tilde{x}$ has to be solved after the iteration is finished.

The matrix $M = BC$ is called the preconditioner matrix. For now we assume that $C = I$ so that

$$M^{-1}Ax = M^{-1}b,$$

which is called *left* preconditioning. The first requirement for the preconditioner matrix is that $M^{-1}A$ has a nicer spectrum than A. This is the case when M is a good approximation of A in some sense, because then $M^{-1}A$ will be close to the identity matrix, which has the smallest possible value of a condition number, $\kappa = 1$. The second requirement is that $M^{-1}Ax$ should be easily calculated or that, equivalently, $My = Ax$ should be easily solved for y. The third requirement is that the set-up of the preconditioner should not be too expensive.

Setting $B \equiv I$ and $C \equiv M$ yields *right* preconditioning, which leads to systems of equations of the form

$$AM^{-1}(Mx) = b.$$

For methods that require a symmetric system matrix, the preconditioner is applied as a symmetric *split* preconditioner

$$M^{-1/2}AM^{-1/2}(M^{1/2}x) = M^{-1/2}b,$$

The spectrum of the preconditioned matrix is the same, independent of how the preconditioner matrix is split. An aspect affecting the choice of the preconditioner splitting is that common stopping criteria of the iteration are often based on the updated residual $r_i \approx Ax_i - b$ of the original equation system. In case the solution method provides only the updated residual of the preconditioned system $\tilde{r}_i \approx \tilde{A}\tilde{x}_i - \tilde{b}$, a back-transformation of this residual is necessary when left or split preconditioning is used, which is done by computing $r_i = B\tilde{r}_i$ in every iteration step. This is usually more expensive than solving $Cx = \tilde{x}$ for x once at the end of the iteration, as necessary in case of split or right preconditioning.

In the next sections several commonly used preconditioners will be explained and examined with respect to their applicability to the pressure-correction matrix in Delft3D-FLOW. We will show that when leaving the domain of weakly non-hydrostatic flows and approaching the strongly non-hydrostatic domain, also the preconditioner technique ought to be changed.

The considered preconditioners are specialized for matrices with a particular structure. The structure of the matrix is depending on the ordering of the pressure unknowns. In the presented scheme the unknowns are ordered lexicographically and the matrix has a structure as given in figure 3.2. The size of the matrix with respect to storage is $k_{\max}i_{\max}j_{\max}$, which is the total number of all cells, regardless of whether they are filled or not. The size of the matrix with respect to matrix-vector operations is usually smaller, because no calculations need to be done

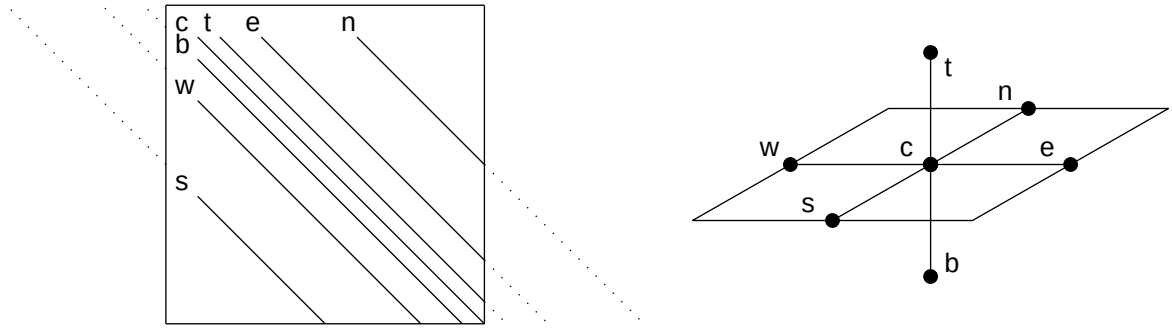

Figure 3.2: Sketch of the Matrix structure (left) and the stencil (right) of the three-dimensional pressure Poisson equation. The letters are abbreviations referring to n(orth), s(outh), e(ast), w(est), t(op), b(ottom) and c(enter).

for empty cells. For instance in vertical direction only cells with vertical index k are present in the computations if $k^{i,j}_{\min} \leq k \leq k^{i,j,n}_{\max}$, where $k^{i,j,n}_{\max} \leq k_{\max}$ and $k^{i,j}_{\min} \geq 1$ (see section 2.2).

3.2.1 Diagonal Jacobi

A simple preconditioner for the system $Ax = b$ is given by a diagonal matrix D with elements $d_{ii} = a_{ii}$. This preconditioner is called diagonal Jacobi preconditioner or diagonal scaling, because it sets the diagonal elements of the preconditioned matrix equal to 1.

Usually this preconditioner is applied directly to the matrix, which means $\tilde{A}$ is computed and stored in the setup phase of the iterative method and multiplications with $\tilde{A}$ are performed directly in every step of the iterative method. Therefore, besides the costs connected with the stopping criterion and the back-scaling of the solution, as described above, the only additional cost of the preconditioned iteration is the matrix scaling in the setup phase.

The preconditioner is computationally cheap and simple, and it improves the convergence especially if the matrix is badly scaled. On the other hand, in the special case that all diagonal elements have the same value, the diagonal Jacobi preconditioner does not have any effect on the spectrum of the matrix. Therefore, it does not improve the convergence considerably in practical situations where (almost) uniform meshes are used.

Diagonal scaling was implemented and documented by Bijvelds [2003] in connection with a CG method. He used the preconditioner in a research version of Delft3D-FLOW.

3.2.2 Tridiagonal Jacobi

An improvement of the Jacobi preconditioner can be obtained by including the subdiagonal and the superdiagonal in the preconditioner, which yields a preconditioner matrix with a bandwidth 1. This tridiagonal Jacobi preconditioner, however, can *not* be easily applied directly to A anymore, because the inverse of M is not sparse enough. In fact, it is a matrix whose non-zeros elements are clustered as blocks with a size $k_{\max} \times k_{\max}$ around the main diagonal, where $k_{\max}$ is the total number of grid layers. Therefore, it is not efficient to store the preconditioned matrix in memory, as done for diagonal Jacobi, or to calculate the inverse of the preconditioner matrix at all. Systems of equations are solved instead. To do this efficiently, the matrix is decomposed in a product of lower and upper triangular parts L_{trid} and U_{trid}. These matrices can be computed by the first step of the Thomas algorithm [Morton and Mayers, 1994]. Using these factors, the

preconditioned matrix $\tilde{A}$ can be given as $U_{\text{trid}}^{-1}L_{\text{trid}}^{-1}A$, $L_{\text{trid}}^{-1}AU_{\text{trid}}^{-1}$ or $AU_{\text{trid}}^{-1}L_{\text{trid}}^{-1}$. Within Krylov algorithms the preconditioner appears only in places where operations $y = \tilde{A}\tilde{x}$ have to be performed. When preconditioning is applied as a (possibly unsymmetric) split preconditioner, in order to compute

$$y = L_{\text{trid}}^{-1}AU_{\text{trid}}^{-1}\tilde{x}$$

we successively solve the systems

$$U_{\text{trid}}z = \tilde{x}, \qquad L_{\text{trid}}y = Az,$$

which is done efficiently by the second part of the Thomas algorithm with consists of forward and backward substitution. Left and right preconditioning can be performed in a similar way:

$$\begin{aligned} y = U_{\text{trid}}^{-1}L_{\text{trid}}^{-1}Ax: &\qquad L_{\text{trid}}z = Ax, &&\qquad U_{\text{trid}}y = z, \\ y = AL_{\text{trid}}^{-1}U_{\text{trid}}^{-1}\tilde{x}: &\qquad U_{\text{trid}}z_1 = \tilde{x}, &&\qquad L_{\text{trid}}z_2 = z_1, \qquad y = Az_2 \end{aligned}$$

The tridiagonal Jacobi preconditioner is especially well suited for matrices where the largest elements are located within the three central diagonals. In the given ordering, these elements in the pressure matrix are related to the vertical discretization. Suppose we are given a Poisson matrix of a three-dimensional problem with Dirichlet boundary conditions on a uniform grid. In this case along each diagonal of the Poisson matrix the elements have the same size. We denote the absolute value of the non-zero off-diagonal elements according to their origin related to the discretization, so that, with respect to figure 3.2,

$$\begin{aligned} d_x &= |e_i| = |w_i| = 1/\Delta x^2, \\ d_y &= |n_i| = |s_i| = 1/\Delta y^2, \\ d_z &= |t_i| = |b_i| = 1/\Delta z^2, \\ d_c &= |c_i| = 2/\Delta x^2 + 2/\Delta y^2 + 2/\Delta z^2. \end{aligned}$$

The relation between the size of any non-zero off-diagonal elements d_α for any $\alpha = x, y, z$ and the main diagonal elements d_c is now given by

$$\frac{d_\alpha}{d_c} = \frac{\Delta x^2\Delta y^2\Delta z^2}{2\Delta\alpha^2(\Delta x^2\Delta y^2 + \Delta x^2\Delta z^2 + \Delta y^2\Delta z^2)}.$$

If $\Delta x = \Delta y = \Delta z$, then the factor is $d_\alpha/d_c = 1/6$. If however, as in most weakly non-hydrostatic simulations, $\Delta z << \Delta x$ and $\Delta z << \Delta y$, then it is easy to see that

$$\frac{d_x}{d_c} \ll \frac{1}{2}, \qquad \frac{d_y}{d_c} \ll \frac{1}{2}, \qquad \frac{d_z}{d_c} \approx \frac{1}{2}.$$

This means that matrices originating from a discretization with flat cells have small "off-tridiagonal" elements, and hence, are close to being tridiagonal matrices. In this case the tridiagonal part of the matrix will be a good preconditioner.

To illustrate the effect of the tridiagonal Jacobi preconditioner on the spectrum of a pressure matrix with respect to the aspect ratio of the cells, we investigate a 27×27 Poisson matrix originating from a problem with homogeneous Neumann boundary conditions, discretized with $3 \times 3 \times 3$ cells . The horizontal spacing is set to $\Delta x = \Delta y = 1$. The relative eigenvalue spectrum for $\Delta z = 1$, $\Delta z = 0.5$ and $\Delta z = 0.2$ with and without the application of the preconditioner is shown in figure 3.3. The eigenvalues λ_m are ordered according to size, with λ_1 being the smallest and λ_{27} being the largest. The condition numbers of all three un-preconditioned matrices are 5.82, the condition numbers of the preconditioned matrices are 4.22, 2.60 and 1.36, respectively.

For Poisson matrices with a discretization of $15 \times 15 \times 15$ cells we computed a reduction of the condition number from 103.1 to 69.1 for $\Delta z = 1.0$, to 35.0 for $\Delta z = 0.5$ and to 8.6 for $\Delta z = 0.2$, using the Matlab `eig` function. This illustrates the statement that for discretizations with flat cells a strong decrease in the condition number can be achieved with this preconditioner.

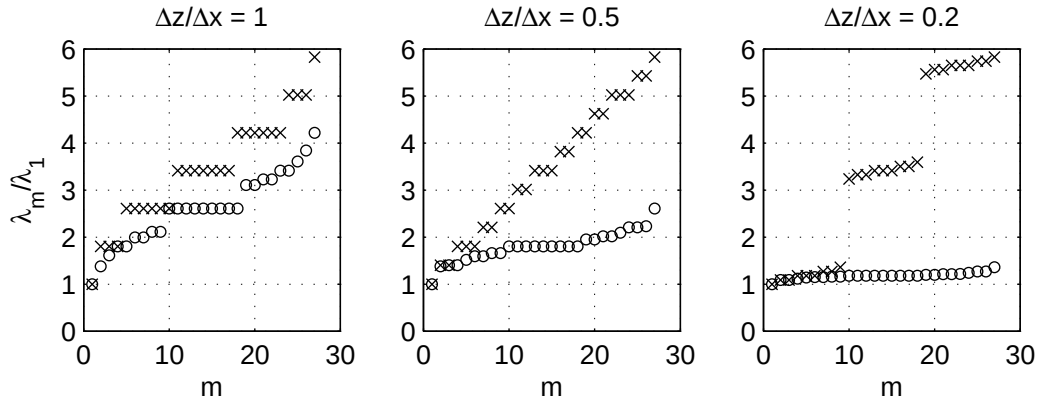

Figure 3.3: Relative eigenvalue spectrum of a 27×27 Poisson matrix originating from uniform spatial discretizations using three different vertical spacings. The crosses mark eigenvalues of the original matrix, the circles mark eigenvalues of the matrix after preconditioning with the tridiagonal Jacobi method.

The tridiagonal Jacobi preconditioner was operational already in the first version of Delft3D-FLOW with non-hydrostatic capability, but was not documented in the manual. Although for many practical cases the preconditioner already gives some performance improvement, it is important that the user is able to choose between different preconditioners according to the problem to be simulated.

3.2.3 ILU(0)

We have seen that the tridiagonal Jacobi preconditioner already reduces the condition number by a large amount, when the grid cells are rather flat. For meshes with grid cells that have similar sizes in the vertical and the horizontal direction, however, it is necessary to include information about *all* non-zero elements. This can be done with a preconditioner based on the incomplete LU-decomposition.

The full LU-decomposition presents a basis for many direct solution methods. It consists of a factorization of the matrix A in a lower triangular matrix L and an upper triangular matrix U such that $A = LU$. It is relatively expensive to compute and store the LU-decomposition because L and U are much less sparse than A. The complete matrix band is filled with non-zero elements. These "fill-in" elements in L and U, however, are relatively small compared to the elements that are located at the non-zero positions of the matrix A.

The incomplete LU-decomposition (ILU) is an approximate factorization that replaces some of the fill-in elements by zero. More specifically, the ILU(0) decomposition replaces *all* fill-in elements by zero, defining the preconditioner matrix $M = LU$ by

$$\begin{aligned} l_{i,j} &= 0 \quad \text{if} \quad j > i \quad \text{or} \quad a_{i,j} = 0 \\ u_{i,j} &= 0 \quad \text{if} \quad i > j \quad \text{or} \quad a_{i,j} = 0 \\ m_{i,j} &= a_{i,j} \quad \text{if} \quad a_{i,j} \neq 0, \end{aligned}$$

where $a_{i,j}$, $l_{i,j}$, $u_{i,j}$ and $m_{i,j}$ are the matrix elements of A, L, U and M, respectively. In this context, i is the row index and j is the column index. In Saad [2003] the ILU(0) preconditioner

for a five-diagonal pressure matrix is derived. By doing the same derivation for a seven-diagonal matrix we find that it is possible to write the ILU(0)-preconditioner as

$$M = (D + L_A)D^{-1}(D + U_A),$$

where L_A is the strictly lower triangular part and U_A is the strictly upper triangular part of A. In this form, M is suitable to be used as a split preconditioner. Once the diagonal matrix D is computed using the ILU(0) algorithm and stored in memory, the preconditioning step can be easily performed using elements of A.

To calculate D we have to deal with indices related to the space discretization and indices related to the rows and columns of the matrix. To avoid confusion we will choose $m = 1 \dots i_{\max}$ and $n = 1 \dots j_{\max}$ as the indices related to the space discretization in x- and y-direction and we use i and j for the position within the matrix. We recall the matrix structure of A as shown in figure 3.2. The matrix is represented by 7 vectors of length $i_{\max}j_{\max}k_{\max}$ in such a way that the vector index is equal to the row-number of the matrix. The algorithm to compute the diagonal elements of D is now given as a recursive expression for $i = nmk = 1 \dots i_{\max}j_{\max}k_{\max}$ over all non-empty cells:

$$d_{nmk} = c_{nmk} - \frac{b_{nmk}t_{nmk^-}}{d_{nmk^-}} - \frac{e_{nmk}w_{n^-mk}}{d_{n^-mk}} - \frac{s_{nmk}n_{nm^-k}}{d_{nm^-k}}, \tag{3.1}$$

where nmk^- denotes the matrix row $nmk - 1$, nm^-k denotes the matrix row $nmk - k_{\max}$ and n^-mk denotes the matrix row $nmk - (k_{\max}m_{\max})$.

There are several ways to split M, but we will use

$$M = \underbrace{\left[(D + L_A)D^{-\frac{1}{2}}\right]}_{L}\underbrace{\left[D^{-\frac{1}{2}}(D + U_A)\right]}_{U}, \tag{3.2}$$

because in the special case of a symmetric matrix A this splitting has the property $U = L^T$, which makes the preconditioner applicable to the CG method as well. In this case the ILU(0) decomposition is equivalent to the Incomplete Cholesky decomposition without fill-in.

If we want to use the preconditioner in the form of (3.2) we either have to store L and U in memory or we have to perform a multiplication with a diagonal matrix every time we apply the preconditioner. Therefore, we can save a considerable amount of memory or work by eliminating the matrix multiplications within the preconditioner. We transform the system $L^{-1}AU^{-1}u = L^{-1}b$ with $u \equiv Ux$ as follows:

$$\begin{aligned}
\left[(D + L_A)D^{-\frac{1}{2}}\right]^{-1} A \left[D^{-\frac{1}{2}}(D + U_A)\right]^{-1} u &= \left[(D + L_A)D^{-\frac{1}{2}}\right]^{-1} b, \\
D^{\frac{1}{2}}(D + L_A)^{-1}A(D + U_A)^{-1}D^{\frac{1}{2}}u &= D^{\frac{1}{2}}(D + L_A)^{-1}b, \\
D^{\frac{1}{2}}(D + L_A)^{-1}D^{\frac{1}{2}}D^{-\frac{1}{2}}AD^{-\frac{1}{2}}D^{\frac{1}{2}}(D + U_A)^{-1}D^{\frac{1}{2}}u &= D^{\frac{1}{2}}(D + L_A)^{-1}D^{\frac{1}{2}}D^{-\frac{1}{2}}b, \\
\underbrace{(I + L_{\tilde{A}})}_{\tilde{L}}{}^{-1}\tilde{A}\underbrace{(I + U_{\tilde{A}})}_{\tilde{U}}{}^{-1}u &= (I + L_{\tilde{A}})^{-1}\tilde{b},
\end{aligned}$$

with

$$\begin{aligned}
\tilde{A} &= D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, \\
\tilde{b} &= D^{-\frac{1}{2}}b, \\
L_{\tilde{A}} &= D^{-\frac{1}{2}}L_AD^{-\frac{1}{2}}, \\
U_{\tilde{A}} &= D^{-\frac{1}{2}}U_AD^{-\frac{1}{2}},
\end{aligned}$$

where $L_{\tilde{A}}$ is equivalent to the strictly lower triangular part of $\tilde{A}$ and $U_{\tilde{A}}$ is equivalent to the strictly upper triangular part of $\tilde{A}$. Note that we can write

$$\begin{aligned} u &= Ux \\ &= D^{-\frac{1}{2}}(D + U_A)x \\ &= D^{-\frac{1}{2}}(D + U_A)D^{-\frac{1}{2}}D^{\frac{1}{2}}x \\ &= (I + U_{\tilde{A}})\tilde{x} \\ &= \tilde{U}\tilde{x} \end{aligned}$$

with $\tilde{x} = D^{\frac{1}{2}}x$. Now we can transform the original system $Ax = b$ to the scaled system $\tilde{A}\tilde{x} = \tilde{b}$ and apply the $\tilde{L}\tilde{U}$-preconditioned BiCGSTAB method to the scaled system. Since A, b and x are no longer needed, we can use their computer memory to store $\tilde{A}$, $\tilde{b}$ and $\tilde{x}$. The matrices $L_{\tilde{A}}$ and $U_{\tilde{A}}$ are directly obtained from the matrix $\tilde{A}$, without any additional storage or matrix multiplication. The only additional work is the diagonal scaling of A, b and x before the iteration and the transformation of $\tilde{x}$ to x after the iteration. Because of this transformation we have to keep D in memory.

The multiplications of vectors with $\tilde{L}^{-1}\tilde{A}\tilde{U}^{-1}$ are the computationally most expensive steps during the BiCGSTAB iteration. Actually one has to solve two triangular systems of equations and multiply a vector with a matrix. It is possible to save some of this work by applying Eisenstat's trick

$$\begin{aligned} \tilde{L}^{-1}\tilde{A}\tilde{U}^{-1}u &= \tilde{L}^{-1}\tilde{b}, \\ \tilde{L}^{-1}(\tilde{L} + \tilde{A} - \tilde{L} - \tilde{U} + \tilde{U})\tilde{U}^{-1}u &= \tilde{L}^{-1}\tilde{b}, \\ \tilde{U}^{-1}u + \tilde{L}^{-1}\left[u + (\mathrm{diag}(\tilde{A}) - 2I)\tilde{U}^{-1}u\right] &= \tilde{L}^{-1}\tilde{b}. \end{aligned}$$

Computing the matrix-vector product with $\tilde{L}^{-1}\tilde{A}\tilde{U}^{-1}$ costs now only two solutions of triangular systems, a product of a vector times a diagonal matrix and a few vector additions, while the matrix-vector product has disappeared. Counting all vector operations for the ILU(0)-preconditioned BiCGSTAB method with a 7-diagonal matrix, the Eisenstat implementation will give a performance increase of about 10% for the iterative solver. Considering the total calculation time and the simplicity of the original algorithm, the gain of using Eisenstats trick is small. Therefore, though mentioning the possibility of this optimization, it is not implemented in the code.

If we perform the experiments from the previous section on a $3 \times 3 \times 3$ grid with the ILU(0) preconditioner, we obtain condition numbers of the preconditioned system of 1.39, 1.34 and 1.13 for $\Delta z = 1.0, 0.5$ and 0.2, respectively, whereas the original system had a condition number of 5.82. The relative eigenvalue spectrum is shown in figure 3.4. The experiments on a $15 \times 15 \times 15$ grid yield 11.3, 9.94, 4.91 versus 103.1 of the original system.

3.2.4 Modified ILU(0)

In the ILU(0)-decomposition we set $m_{i,j} = a_{i,j}$ if $a_{i,j} \neq 0$. A similar decomposition is the MILU decomposition, which is achieved by using row sums in a way that

$$\begin{aligned} l_{i,j} &= 0 \qquad \text{if} \quad j > i \quad \text{or} \quad a_{i,j} = 0 \\ u_{i,j} &= 0 \qquad \text{if} \quad i > j \quad \text{or} \quad a_{i,j} = 0 \\ \sum_i m_{i,j} &= \sum_i a_{i,j} \end{aligned}$$

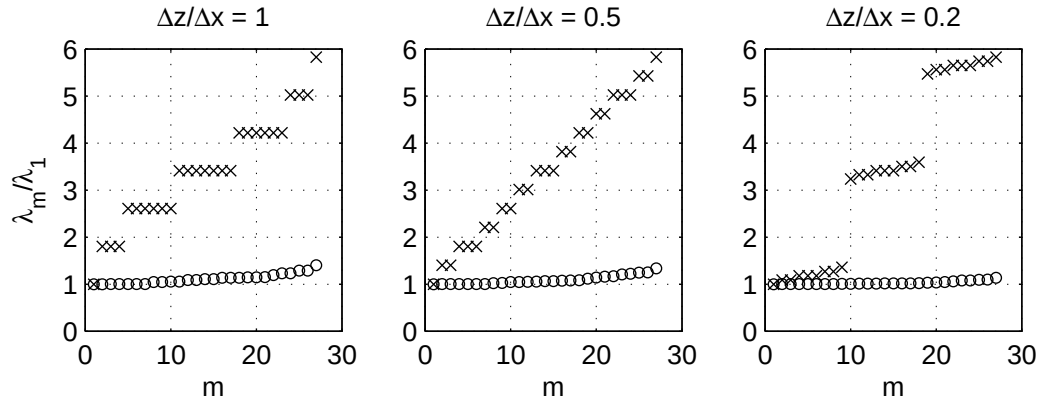

Figure 3.4: Relative eigenvalue spectrum of a 27×27 Poisson matrix originating from uniform spatial discretizations using three different vertical spacings. The crosses mark eigenvalues of the original matrix, the circles mark eigenvalues of the matrix after preconditioning with the ILU(0) method.

which leads to the recursive loop for $i = nmk = 1 \ldots n_{\max} m_{\max} k_{\max}$ over

$$\begin{aligned} d_{nmk} = c_{nmk} &- \frac{b_{nmk}(t_{nmk^-} + w_{nmk^-} + n_{nmk^-})}{d_{nmk^-}} \\ &- \frac{e_{nmk}(t_{n^-mk} + w_{n^-mk} + n_{n^-mk})}{d_{n^-mk}} \\ &- \frac{s_{nmk}(t_{nm^-k} + w_{nm^-k} + n_{nm^-k})}{d_{nm^-k}}. \end{aligned}$$

It is easy to generate an interpolation between the ILU(0) and the MILU preconditioner by including a parameter α_{milu}, so that ILU(0) is obtained when $\alpha_{\text{milu}} = 0$ and MILU is obtained when $\alpha_{\text{milu}} = 1$. We call this method MILU(α_{milu}). The resulting expression is

$$\begin{aligned} d_{nmk} = c_{nmk} &- \frac{b_{nmk}(t_{nmk^-} + \alpha_{\text{milu}}(w_{nmk^-} + n_{nmk^-}))}{d_{nmk^-}} \\ &- \frac{e_{nmk}(w_{n^-mk} + \alpha_{\text{milu}}(t_{n^-mk} + n_{n^-mk}))}{d_{n^-mk}} \\ &- \frac{s_{nmk}(n_{nm^-k} + \alpha_{\text{milu}}(t_{nm^-k} + w_{nm^-k}))}{d_{nm^-k}}. \end{aligned}$$

We calculate the condition number of the preconditioned $15 \times 15 \times 15$ Poisson matrix with MILU(α_{milu}) preconditioning. By doing a few tests we found that $\alpha_{\text{milu}} = 0.95$ is close to the optimum value. We see that there is some gain when MILU(0.95) is used instead of the ILU(0)-factorization in this case: The condition numbers were 5.16, 4.48, 3.53 versus 103.1 of the original system. Since the condition number is only one index of predicting the convergence of the BiCGSTAB method, practice must show whether the application of the MILU-factorization really gives a significant performance improvement. If the few extra operations in the setup phase are neglected, the additional cost per iteration step is zero compared to ILU(0), therefore even a small improvement of the convergence behavior is directly rewarded in overall speed of the computation.

To illustrate the convergence of the presented preconditioners, the Matlab test problem of section 3.1 is considered again, using the same random right hand side and initial values. The

matrix is modified to represent an aspect ratio $\Delta z/\Delta x = \Delta z/\Delta y = 1$ in a first computation and $\Delta z/\Delta x = \Delta z/\Delta y = 0.2$ in a second computation.

Figure 3.5 shows the convergence of the BiCGSTAB method with the diagonal Jacobi, the tridiagonal Jacobi and the ILU-based preconditioners ILU(0), MILU and MILU(0.5), where $\alpha_{\text{milu}} = 0.5$ was found to be close to optimum after doing a few tests. There is no difference between the diagonal Jacobi preconditioned and the unpreconditioned iteration, so the unpreconditioned iteration is not shown explicitly in the plot. For $\Delta z/\Delta x = \Delta z/\Delta x = 0.2$ we can see that the ILU-based preconditioners have a much steeper convergence curve than the Jacobi-based preconditioners, and that the MILU(0.5) is the best preconditioner among the ILU-based ones. When we choose $\Delta z/\Delta x = \Delta z/\Delta x = 0.2$, then we see that the tridiagonal Jacobi preconditioner is almost as good as the ILU-based preconditioners. Considering that the tridiagonal Jacobi preconditioner is relatively cheap per time step, it may be that it outperforms the other preconditioners in terms of wall-clock times. For a comparison of wall-clock times for a realistic test case computed with Delft3D-FLOW, see section 4.3.

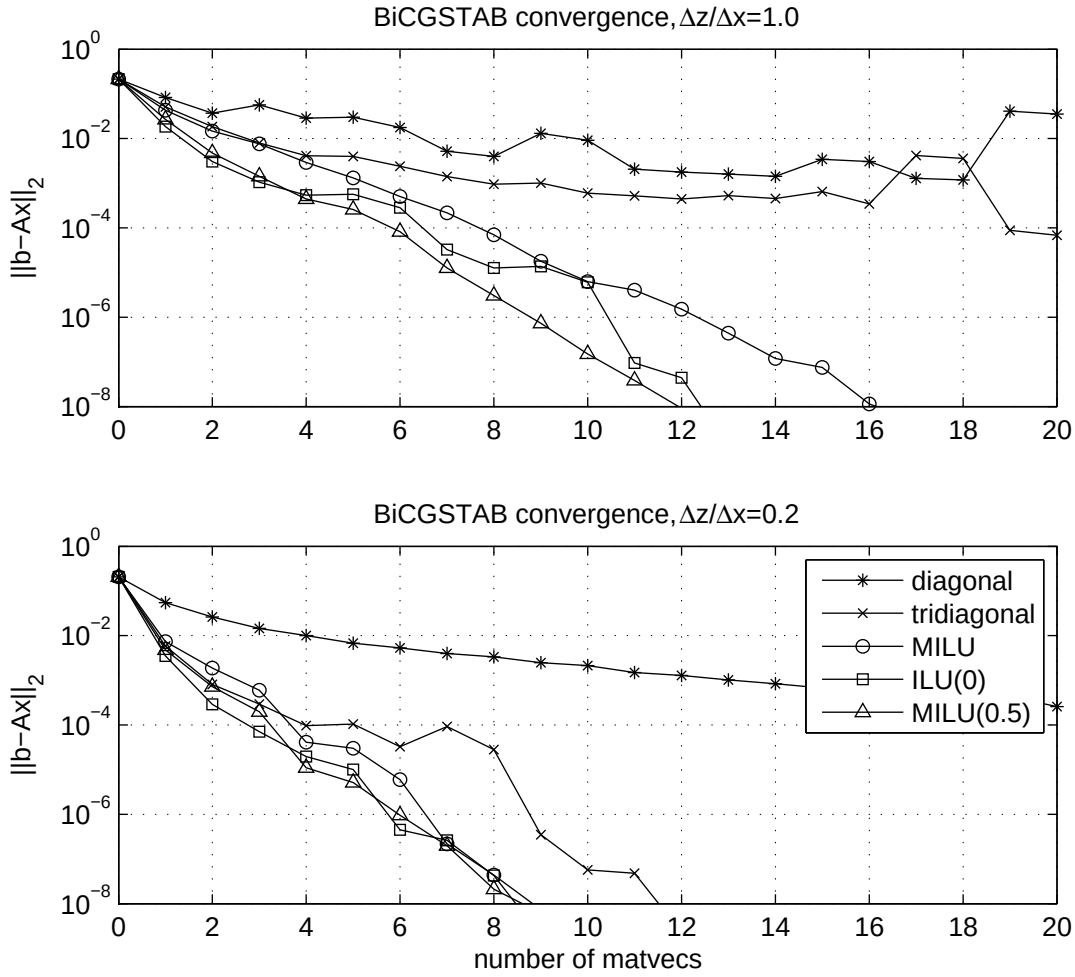

Figure 3.5: Convergence of the BiCGSTAB method with different preconditioners for a perturbed Poisson equation. In the case that $\Delta z/\Delta x = 1$ the combination of ILU(0) and MILU with a parameter $\alpha = 0.5$ gives best results. The difference between the tridiagonal Jacobi preconditioner and the ILU preconditioner is mainly depending on the value of $\Delta z/\Delta x$.

3.3 Starting values and stopping criteria

An iterative method takes an initial guess of the solution and updates this guess every step until a good approximation of the solution is obtained with a sufficient accuracy. It is good to have a starting guess available that is close to the actual solution, so that only a few iterations are needed until the approximation is sufficiently accurate. It is necessary to define what is "sufficient accuracy" based on some problem-related quantity. The discussion about starting values and stopping criteria involves some physical relations. Therefore, we have to switch from the unspecific and general unknown x used in previous sections to the specific unknown Δp of the actual pressure-correction system. To compare unknowns of different time steps we add an index to the pressure-correction. We define the pressure-correction at time step n as $\Delta p^n = p^{n+1} - p^n$ and we denote the initial guess of the iterative method at time step n with Δp_0^n.

Two simple choices of initial values for the pressure-correction at time step n are $\Delta p_0^n = 0$ and $\Delta p_0^n = \Delta p^{n-1}$, where Δp^{n-1} is the solution of the previous time step. One could also think of a linear extrapolation $\Delta p_0^n = \Delta p^{n-1} + (\Delta p^{n-1} - \Delta p^{n-2})$ to get a better estimate. A few runs with a simple test case in the development phase of the solver have showed that $\Delta p_0^n = \Delta p^{n-1}$ gives an improvement over $\Delta p_0^n = 0$, but the extrapolation does not have significant effect on the convergence. Therefore, simply the solution of the previous step is used as an initial value in the BiCGSTAB solver of Delft3D-FLOW. In the very first time step the starting guess is set to the hydrostatic pressure, because no better guess is readily available.

Besides the initial guess, every iterative method to solve a linear system requires a criterion that determines when to step out of the iteration. Finding a good termination criterion is more difficult than finding a good initial guess.

Iterative methods are able to find the solution of a given system with an accuracy of almost the order of magnitude of the machine precision if the system is sufficiently well conditioned. The actual advantage of iterative methods, however, is their capability to produce results with a lower, but still sufficient, accuracy within a much shorter calculation time. The stopping criterion prescribes the compromise between the accuracy of the solution and the time spent to find this solution.

When we want to solve a system of equations originating from a discretized partial differential equation, the stopping criterion should be satisfied if the solution has an accuracy of the size of the discretization error. The stopping criterion could also express some physical condition or the reduction of the initial error by a certain factor. If the solution of a linear system of equations is only a part of another solution process, like a pressure-correction method, then the accuracy of the iterative solution should be compared to the accuracy of the overall process.

One way to formulate a stopping criterion for a system $Ax = b$ is to measure the error $e^i = x - x^i$ in some vector norm. If we knew the error e^i, we could stop iterating if the error norm $||e^i||$ became smaller than some tolerance. But knowing e^i, we would even be able to calculate the true solution directly from $x = e^i + x^i$. Unfortunately, calculating the error is as hard as calculating the true solution, because $Ax = A(x^i + e^i) = b$ and so $Ae^i = b - Ax^i$. Therefore, one is usually satisfied if a bound of the error norm can be measured, for example

$$||e^i|| \leq ||A^{-1}|| \cdot ||b - Ax^i||, \tag{3.3}$$

where $||A^{-1}||$ is a matrix norm of the inverse of A, which is compatible with the vector norm used. The term

$$r^i \equiv b - Ax^i \tag{3.4}$$

is called the residual. From (3.3) it is clear that if $||A^{-1}|| \cdot ||r||$ is smaller than some tolerance then also $||e^i||$ is smaller than this tolerance. The problem yet to be resolved is the efficient estimation of $||A^{-1}||$, which is possible only in special cases (see Barrett et al. [1994]).

Instead of using an error bound, our aim is to find a physical quantity that can be used as a stopping criterion. Therefore, recall the origin of the system to be solved: In section 2.5 the pressure-correction equation is derived from the momentum correction equations, here shown without space discretization,

$$\begin{aligned} u^{n+1} &= -\Delta t\theta \frac{\partial \Delta p}{\partial x} + u^{n+1*} \\ v^{n+1} &= -\Delta t\theta \frac{\partial \Delta p}{\partial y} + v^{n+1*} \\ w^{n+1} &= -\Delta t\theta \frac{\partial \Delta p}{\partial z} + w^{n+1*}. \end{aligned} \tag{3.5}$$

It is required that Δp is computed in such a way that the final velocity satisfies

$$\frac{\partial u^{n+1}}{\partial x} + \frac{\partial v^{n+1}}{\partial y} + \frac{\partial w^{n+1}}{\partial z} = 0.$$

We know that we are not able to calculate the solution of the pressure-correction equation accurately in the presence of numerical errors. This means we approximate Δp by $\Delta \hat{p}$ in a way that

$$\frac{\partial u^{n+1}}{\partial x} + \frac{\partial v^{n+1}}{\partial y} + \frac{\partial w^{n+1}}{\partial z} = e_{\mathrm{c}}, \tag{3.6}$$

where e_{c} is some error. In fact, equation (3.6) expresses that e_{c} is the divergence of the final velocity field. By substituting (3.5) in (3.6) we obtain the pressure-correction equation

$$\Delta t\theta \left(\frac{\partial^2 \Delta \hat{p}}{\partial x^2} + \frac{\partial^2 \Delta \hat{p}}{\partial y^2} + \frac{\partial^2 \Delta \hat{p}}{\partial z^2} \right) = \frac{\partial u^{n+1*}}{\partial x} + \frac{\partial v^{n+1*}}{\partial y} + \frac{\partial w^{n+1*}}{\partial z} + e_{\mathrm{c}}. \tag{3.7}$$

An important observation is that the divergence of the final velocity field is equivalent to the residual in the iterative solution method. To show this, suppose we discretize (3.7) so that we can write it in matrix form

$$A \cdot (\Delta \hat{p}) = b + e_{\mathrm{c}}, \tag{3.8}$$

where A is the matrix resulting from the discretization of the left-hand side and b is the discretized divergence field of the preliminary velocity field. If we stop the iteration at step i, the substitution of (3.8) with $x^i \equiv \Delta \hat{q}^*$ in (3.4) reveals that $e_{\mathrm{c}} = r^i$. So the residual of the iterative method can be used to limit the divergence of the final velocity field.

To measure the size of the residual we need to transform the information contained in the elements of the residual vector to a single number. To keep the formulas simple we say that the residual has k elements and to access one element of the residual we use only one a single index from 1 to k. A way to measure the size of the residual is using a p-norm

$$||r||_p = \Big(\sum_{j=1}^{k} |r_j|^p \Big)^{1/p},$$

where $p \in \mathbb{R}$ and $p \geq 1$. Usual choices of p are 1, 2 and ∞. The infinity norm can be written as $||r||_\infty = \max_j |r_j|$. One has to bear in mind that p-norms with $p < \infty$ scale with $k^{1/p}$. This is unproblematic if the stopping criterion has the form of $||r||_p < \epsilon ||x||_p$ for an arbitrary vector x with k elements, because both sides of the inequality scale with the same number. If we have a criterion in the form of $||r||_p < \epsilon$, however, the choice of ϵ must be based on information about p and k. Therefore, if ϵ is meant to be a limit for the divergence of the final velocity field, should use $p = \infty$.

There exist alternatives to using a p-norm of the residual as a stopping criterion. One could think, for example, of the spatial average of the absolute divergence field,

$$r_{avg} = \frac{\sum_{j=1}^{k} \Delta x_j \Delta y_j \Delta z_j |r_j|}{\sum_{j=1}^{k} \Delta x_j \Delta y_j \Delta z_j}.$$

In case of a uniform grid

$$r_{avg} = \sum_{j=1}^{k} |r_j|/k = ||r||_1/k,$$

which shows the relationship between the average and the 1-norm. In general we can interpret the average as the 1-norm of a weighted residual according to

$$r_{avg} = ||a \cdot r||_1,$$

where the vector a consists of elements

$$a_j = \frac{\Delta x_j \Delta y_j \Delta z_j}{\sum_{i=1}^{k} \Delta x_i \Delta y_i \Delta z_i}.$$

Up to now we are able to define the size of the final divergence in terms of its maximum or its average absolute. To formulate a stopping criterion, we still require an inequality. The most obvious choices are:

- *Bind the divergence:* To limit the divergence directly in terms of a number ϵ, we can formulate the stopping criterion $||r^i|| < \epsilon$. This gives a absolute bound on the error introduced by the iterative method in each time step. The accuracy of the solver is independent on the initial guess. Therefore, it is considered quite a save bound. However, if the error in the preliminary velocity field is very large, then making the *divergence* of the final velocity field very small does not necessarily mean that the *error* of the final velocity field is very small, too.

- *Reduce the divergence of the intermediate velocity field by a factor.* The criterion $||r^i|| < \epsilon||b||$ expresses the fact that the goal of the non-hydrostatic extension is to *improve* the hydrostatic guess. If the guess is not very accurate, we do not need to calculate the pressure-correction very accurately either, which saves computation time. If the divergence of the preliminary velocity field is already very small, however, there might be a lot of iteration steps necessary to make it even smaller. This is the case when a solution approaches a steady state. In this case, making the convergence error very small is not meaningful if other errors of the numerical scheme are much larger.

- *Reduce the initial residual by a factor.* The bound $||r^i|| < \epsilon||r^0||$ leads to an approximately constant work load every time step. If $\Delta p_0^n = 0$ is chosen as an initial guess, then this criterion is equivalent to the previous one. If we use the $\Delta p_0^n = \Delta p^{n-1}$, then we might have a better initial guess and, therefore, a smaller initial residual. Still, the algorithm will try to increase this residual by a constant factor. In case of an almost steady state, the algorithm will still perform a lot of iterations per time step.

Since the last criterion does not reward the use of a good initial guess, it is not as useful as the second criterion, although both criteria will lead to similar numbers of iterations in case that no good initial guess is available. The best stopping criterion for most practical applications is a combination of the first and the second criterion,

$$||r^i|| < \epsilon_1 \ \texttt{or} \ ||r^i|| < \epsilon_2||b||. \tag{3.9}$$

If the preliminary velocity field has already a relatively small divergence, then we only iterate until the divergence reaches ϵ_1, so that the number of iterations goes to zero when the velocity field approaches a steady state. If the preliminary velocity field has a very high divergence, then we decrease that divergence only by the factor ϵ_2, because the error is expected to be large, anyway, even if the divergence is small. By setting either ϵ_1 or ϵ_2 to zero we obtain one of the first two criteria.

3.4 Implementation

A pseudo-code of the final implementation of the BiCGSTAB algorithm with MILU preconditioning is given as algorithm 3. The special form of the preconditioner, as described in sections (3.2.3) and 3.2.4 leads to a more complicated algorithm then when we had stored the matrices L and U directly in memory. However, in the chosen form the algorithm is more efficient with respect to memory requirements. In the beginning of the solution process the matrix and the right-hand side are scaled with $D^{-1/2}$ and the scaled system is used as the actual system to be solved. Only at the end and for the stopping criteria it is required to transform to the unscaled quantities.

Compared to algorithm 1, the iteration loop is shifted, so that the stopping criterion is at the beginning of the iteration. This has the following advantage: If a very good initial guess is available, then it can be directly used as a final solution without actually doing a single iteration. This is may be the case when the time step is set to a very small value or when the solution reaches a steady state. Then only every few time steps a Krylov iteration needs to be done to update the pressure.

After checking for convergence with the updated residual, there is another check with the true residual. This check is computationally expensive, because it involves a matrix-vector product. Still, it is necessary, because it is reported in van der Vorst [1992] that the updated residual may differ by orders of magnitude from the true residual, and convergence in terms of the true residual is what we actually require.

The preconditioner is applied as a right preconditioner to the scaled system, so that the preconditioned and the unpreconditioned updated residual are the same. The additional cost to compute the final solution from the preconditioned solution could be avoided by using the preconditioned search directions in line 25. This means that the scaled solution is directly obtained without an additional system to be solved at the end of the iteration.

There are several possible divisions by zero in the algorithm. Therefore, before the divisions, it is checked, whether the divisor is zero. If this is the case, the algorithm is restarted at line 6. The checks for divisions by zero are not mentioned in the pseudo-code.

Algorithm 3 BiCGSTAB method with MILU preconditioning

take $\tilde{x}^0$ from the previous time step (zero in first step)
generate matrix A and right-hand side b
compute D of the MILU decomposition of A
$\tilde{A} = D^{-1/2} A D^{-1/2}$
$\tilde{b} = D^{-1/2} b$
$p^i = \bar{r}^0 = \tilde{r}^0 = \tilde{b} - \tilde{A}\tilde{x}^0$
$\rho_0 = (\bar{r}^0, \tilde{r}^0)$
for $i = 0 \dots N$ **do**
  $r_{\text{update}} = D^{1/2} \tilde{r}^i$
  if $||r_{\text{update}}||_\infty < \epsilon_1$ or $||r_{\text{update}}||_\infty < \epsilon_2 ||b||$ **then**
    $r_{\text{true}} = D^{1/2}(\tilde{b} - \tilde{A}\tilde{x}^i)$
    if $||r_{\text{true}}||_\infty < \epsilon_1$ or $||r_{\text{true}}||_\infty < \epsilon_2 ||b||$ **then**
      leave for-loop
    else
      restart
    end if
  end if
  $\hat{p} = \tilde{U}^{-1} \tilde{L}^{-1} p^i$ (forward/backward substitution)
  $v = \tilde{A}\hat{p}$
  $\alpha_i = \rho_i / (\bar{r}^0, v)$
  $s = \tilde{r}^i - \alpha_i v$
  $\hat{s} = \tilde{U}^{-1} \tilde{L}^{-1} s$ (forward/backward substitution)
  $t = \tilde{A}\hat{s}$
  $\omega_i = (t, s)/(t, t)$
  $\tilde{x}^{i+1} = \tilde{x}^i + \alpha_i \hat{p} + \omega_i \hat{s}$
  $\tilde{r}^{i+1} = s - \omega_i t$
  $\rho_{i+1} = (\bar{r}^0, \tilde{r}^{i+1})$
  $\beta_i = (\rho_{i+1}/\rho_i)(\alpha_i/\omega_i)$
  $p^{i+1} = \tilde{r}^{i+1} + \beta_i (p^i - \omega_i v)$
end for
compute solution $x = D^{-1/2} \tilde{x}$ using most recent value of $\tilde{x}^i$

Chapter 4

Simulations

In this chapter we want to study the accuracy and the efficiency of the numerical scheme described in chapter 2. In section 4.1 we consider the accuracy with respect to the damping and the phase speed of waves for a simple test problem with a standing wave in a basin. In this case we are able to combine the outcome of analytical considerations with actual simulations. In section 4.2 the wave propagation over a submerged bar is modeled in the x-z-plane. This is a more demanding simulation, because it features non-linear effects and waves with different wave speeds and wave length. In section 4.3 the efficiency of the solver will be tested with a three-dimensional model of a ship lock.

4.1 Standing wave in a basin

A standard test problem to investigate the accuracy of a numerical scheme for wave problems is the two-dimensional vertical (2DV) simulation of a standing surface wave with a small amplitude in a closed water basin. The basin has a rectangular shape with a length L and a mean water level H. Free-slip conditions are prescribed at all boundaries except the free surface. Initially, the water is at rest and the water level is prescribed as

$$\zeta(x,0) = A_{\zeta,0}\cos(kx), \qquad 0\,\mathrm{m} \leq x \leq L.$$

When all non-linear, viscous and turbulent effects are neglected, the solution for the water level is an undamped linear wave, which makes this test case interesting for an analytical study.

To examine the numerical scheme with respect to wave applications a distinction between different types of waves can be made, using the linearized dispersion relation

$$\omega = \pm\sqrt{gk\tanh(kH)}, \tag{4.1}$$

which relates the angular frequency ω with the wave number k and the mean water depth H, provided the amplitude is relatively small compared to k and to H. Other variables of possible interest are the phase velocity $v_\mathrm{p} = \omega/k$, the wave length $\lambda = 2\pi/k$, the wave period $T = 2\pi/\omega$ and the wave frequency $f = \omega/2\pi$. The dispersion describes the dependence of the wave speed with respect to the depth of the water an the length of the wave. However, in certain cases only one of both effects is important. Based on the size of kH we can discriminate between the following types of waves:

- *Long waves:* When the wave length is significantly greater than the water depth, then $kH \ll 1$. By using a Taylor expansion,

$$\tanh(kH) = kH + O((kH)^3)$$

Therefore, the angular frequency can be approximated by $\omega = \pm k\sqrt{gH}$ and the wave speed is $v_\mathrm{p} = \pm\sqrt{gH}$.

- *Short waves:* When the kH is in the order of magnitude of 1, then the full relation (4.1) must be used to describe the dispersion.
- *Very short waves:* If the value of kH is much larger than 1 than another simplification can be made. We know that
$$\lim_{kH\to\infty} \tanh(kH) = 1,$$
so we can approximate $\omega = \pm\sqrt{gk}$. We see that in this case the wave speed $v_\mathrm{p} = \pm\sqrt{g/k}$ is no longer dependent on the depth.

Illustrations of the different types of waves are shown in figure 4.1, where cross sections of standing waves in a basin with different length are displayed.

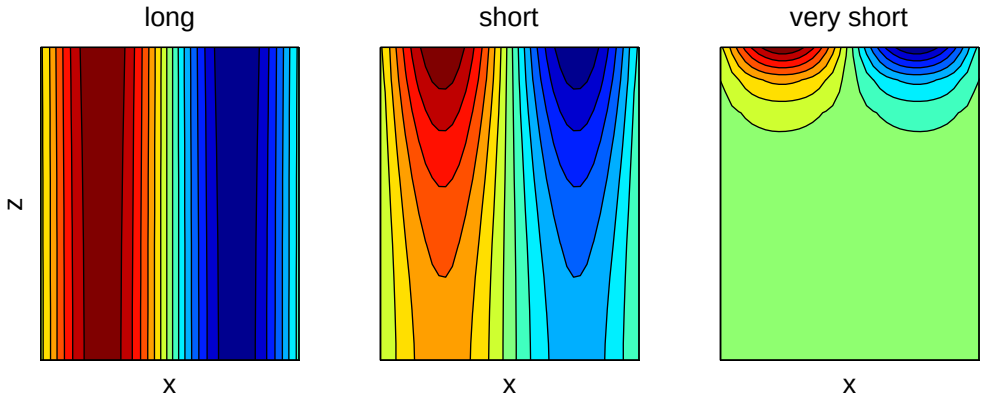

Figure 4.1: Typical depth profiles of waves characterized by the attributes 'long', 'short' and 'very short', obtained with the Delft3D-FLOW code. The plots show a cross section view of the horizontal velocity of a simple standing wave in a basin, with length to height ratios 200 m : 10 m, 40 m : 10 m and 5 m : 10 m, respectively, which means $kH \approx 0.31, 1.6$ and 12.6 from left to right. Because the amplitude was chosen very small, the surface profile is not visible in the plot.

4.1.1 Damping effect of the time discretization

The solution of the water level in the closed basin can be reasonably well approximated by an undamped standing harmonic wave, which will serve as an "exact solution" in the following. After transforming it to the complex domain it is given as

$$\zeta(x,t) = A_{\zeta,0} e^{i(kx-\omega t)}, \tag{4.2}$$

the real part of which is the actual wave solution.

The solution of the simulated problem should be close to equation (4.2). However, the discretization in space and time may introduce artificial damping or amplification, a phase shift or a deformation of the wave. Here, we concentrate on the artificial damping. To investigate this effect we consider the solution of the water level at one of the boundaries ($x = 0$ or $x = L$), where $\cos(kx) = 1$. This gives water levels and respective time derivatives of

$$\zeta(0,t) = A_{\zeta,0} e^{-i\omega t} \qquad \text{and} \qquad \frac{\mathrm{d}\zeta(0,t)}{\mathrm{d}t} = -i\omega\zeta(0,t).$$

We can interpret the second equation as an ordinary differential equation. Discretization of the time derivative with a θ-scheme yields

$$\frac{\zeta^{n+1} - \zeta^n}{\Delta t} = -i\omega(\theta\zeta^{n+1} + (1-\theta)\zeta^n). \tag{4.3}$$

Actually the considered non-hydrostatic free-surface model can not be completely expressed as a θ-scheme, because several terms in the momentum equations are treated fully explicitly or fully implicitly. Only in case of a small amplitude these terms are small compared to the θ-discretized pressure terms and the simplification of above is valid.

Equation 4.3 can be rewritten as

$$\zeta^{n+1} = c_\mathrm{a}\zeta^n$$

with

$$\begin{aligned} c_\mathrm{a} &= \frac{1 - i\omega\Delta t(1-\theta)}{1 + i\omega\Delta t\theta} \\ &= \frac{\big(1 - i\omega\Delta t(1-\theta)\big)\big(1 - i\omega\Delta t\theta\big)}{1 + (\omega\Delta t\theta)^2}, \end{aligned}$$

where c_a is called the amplification factor. The absolute value of the amplification factor denotes the damping of the solution per time step. It is given as

$$|c_\mathrm{a}| = \sqrt{\frac{1 + (\omega\Delta t(1-\theta))^2}{1 + (\omega\Delta t\theta)^2}}. \tag{4.4}$$

The exact solution contains no damping. Therefore, $|c_\mathrm{a}|$ should be equal or close to 1. Looking at (4.4), however, we see that depending on θ the solution may be amplified or damped. In the case of $\theta = 0$, which yields the forward Euler-scheme, the wave is amplified because $\sqrt{1 + (\omega\Delta t)^2} > 1$. In the case of $\theta = 1$, which yields the backward Euler-scheme, the wave is damped because $1/\sqrt{1 + (\omega\Delta t)^2} < 1$. For the Crank-Nicolson method, with $\theta = 1/2$, neither damping nor amplification is present.

Equation (4.4) gives rise to an amplitude function A^n_ζ in discrete time, which is obtained using

$$A^n_\zeta = |\zeta^n| = |(c_\mathrm{a})^n\zeta^0| = |c_\mathrm{a}|^n A_{\zeta,0},$$

which is equivalent to the time-continuous function

$$A_\zeta(t) = A_{\zeta,0}(t) \cdot \exp\left\{\frac{t}{\Delta t}\ln|c_\mathrm{a}|\right\}. \tag{4.5}$$

Note that the logarithm gives rise to a negative exponent for $\theta > 1/2$ and a positive exponent for $\theta < 1/2$. In the plots we will also show the negative amplitude to make the visible comparison of the numerical and the predicted solution easier.

The expected numerical solution, including the numerical damping effect due to the time discretization, is given by

$$\zeta(x,t) = A_\zeta(t)\cos(\omega t), \tag{4.6}$$

which will be a measure to evaluate the numerical experiments shown below.

By setting t equal to the wave period $T = \omega/2\pi$ and dividing by the amplitude, we obtain the relative damping per wave period,

$$c_\mathrm{per} \approx \exp\left\{\frac{2\pi}{\omega\Delta t}\ln|c_\mathrm{a}|\right\},$$

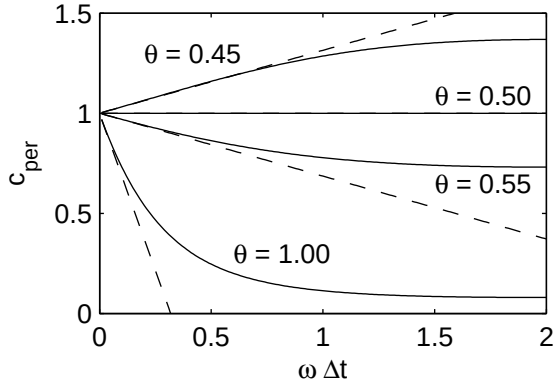

Figure 4.2: Damping of the θ-scheme per wave period for different values of θ plotted as solid lines. The slashed lines refer to the linearization around $\Delta t = 0$. For $\omega\Delta t \gg 1$ the curves approach 1. This is not shown in the plot since only small $\omega\Delta t$ are relevant for studying the accuracy of wave solutions.

which is a function of $\omega\Delta t$ and θ. A Taylor expansion around $\Delta t = 0$ gives

$$c_{\text{per}} = 1 + (1 - 2\theta)\pi\omega\Delta t + O(\Delta t^2),$$

which shows that the damping per wave period is of order Δt for any wave frequency when $\theta \neq 1/2$.

In figure 4.2 the damping behavior is plotted for some values of θ. Because the damping factor is linear in Δt around $\Delta t = 0\,\text{s}$, the time step has to be chosen small in order to reduce damping, or θ has to be chosen close to 0.5. The figure also illustrates the instability of the scheme when $\theta < 0.5$, because then for any Δt all waves are amplified.

We want to test if the numerical damping rate of the solution of the closed basin test case behaves like the analytical prediction. We consider a basin with a length of $L = 20\,\text{m}$ and a depth of $H = 10\,\text{m}$. We set the initial water level amplitude to $A_{\zeta,0} = 0.1\,\text{m}$, which is reasonably small to prevent noticeable nonlinear effects for a sufficiently long simulation time. We set the wave length equal to the length of the basin, $\lambda = L = 20\,\text{m}$, which leads to a wave number of $k \approx 0.314\,\text{m}^{-1}$. The wave number is relatively small, so the wave can be categorized as a very short wave. According to the linear dispersion relation (4.1), using a gravity $g \approx 9.81\,\text{m/s}^2$ we obtain the angular frequency $\omega \approx \pm 1.75\,\text{s}^{-1}$. The experiment is set up in Delft3D-FLOW with 11 horizontal layers, located between $z = -10.5\,\text{m}$ and $z = 0.5\,\text{m}$. To obtain a 2DV model with the three-dimensional code, only a single grid layer is used in the y-direction. Further, we use 40 cells to discretize in x-direction, so that $\Delta x = 0.5\,\text{m}$ and we set $\Delta t = 0.12\,\text{s}$. The CFL number with respect to the wave propagation is given as $(\omega/k) \cdot \Delta t/\Delta x \approx 1.34$.

The result of a simulation using $\theta = 1$ in Delft3D-FLOW with the method described in chapter 2 is shown in figure 4.3, together with the envelope according to (4.5) and the damped solution according to (4.6). The solution shows that the prediction of the damping rate was accurate and, thus, the damping related to the time discretization is indeed the most prominent error source. The plot also shows that the numerical solution does not exhibit any significant phase shift with respect to the analytical phase shift.

Now we decrease θ in the time integration of the surface equation and the momentum equations to 0.5. As explained above, the damping due to the time discretization should be completely eliminated. The result of the computations is shown in figure 4.4 together with the predicted solution (4.6), which reduces to the exact solution (4.2) for $x = 0$ or $x = L$. Indeed,

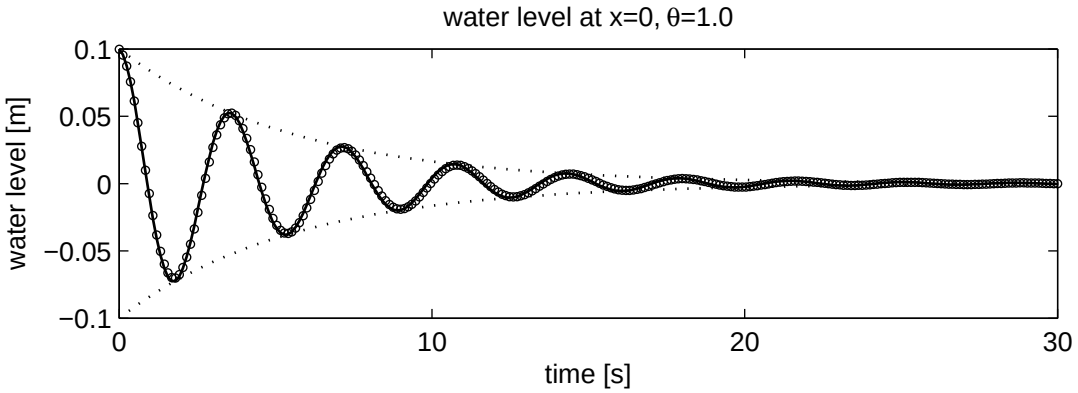

Figure 4.3: Water level at $x = 0$ with $\theta = 1$. The numerical solution is marked with circles, the predicted solution and amplitude are plotted with solid and dotted lines, respectively. The predicted solution is equivalent to the exact solution multiplied with the amplitude function (4.5).

the damping is not visible anymore and an accurate solution is obtained. The conclusion is that, in order to eliminate wave damping, θ should be chosen 0.5. The scheme considered in this thesis contains explicit terms ($\theta = 0$) in the momentum prediction equations. Therefore, in some cases it may be necessary to set θ slightly larger than 0.5 in order to achieve stability. In the basin test case considered here, it turned out that choosing $\theta = 0.5$ was stable.

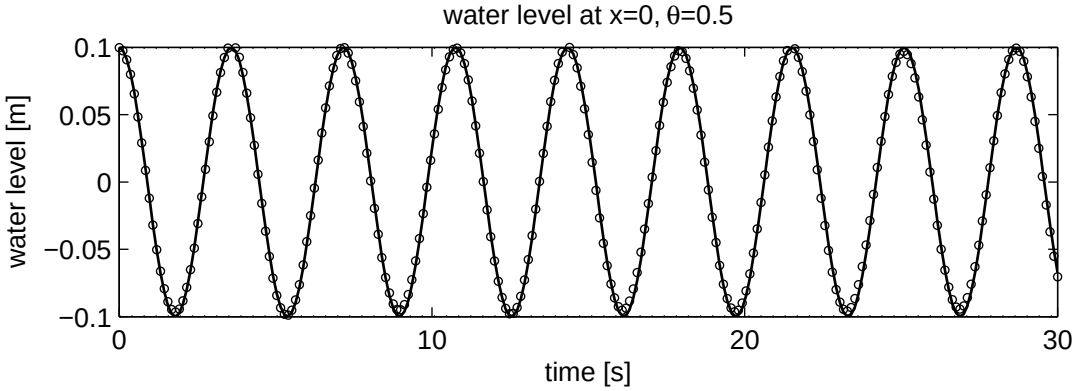

Figure 4.4: Water level at $x = 0$ with $\theta = 0.5$. The numerical solution is marked with circles and the exact solution is shown as a solid line. The close match shows that the numerical damping in figure 4.3 was mainly caused by the time discretization.

We are interested if the damping behavior of the Bijvelds scheme is similar to a θ scheme with $\theta = 1$. Therefore, we perform further numerical experiments, comparing both schemes to the analytical predictions used in the previous studies. We extend the model to take different wave lengths into account: We choose $L = 200\,\mathrm{m}, 20\,\mathrm{m}, 5\,\mathrm{m}$ and set the wave lengths equal to L. The first case, $L = 200\,\mathrm{m}$ almost allows for a hydrostatic assumption, because the angular frequency obtained form the hydrostatic dispersion relation $\omega = \pm k\sqrt{gh}$ differs only about 2 % from the agular frequence obtained by the general linear dispersion relation (4.1). The third case is a clear example of a very short wave which can be described almost perfectly with the deep water relation $\omega = \pm\sqrt{gk}$.

We decrease the initial amplitude to $A_{\zeta,0} = 0.01\,\mathrm{m}$ in order reduce nonlinear effects which

could come into play as $L = 5\,\mathrm{m}$. For the discretization, we again use 40 grid cells in x-direction, so that $\Delta x = 5.0\,\mathrm{m}, 0.5\,\mathrm{m}, 0.125\,\mathrm{m}$. Some adjustment had to be made in order to compare the schemes successfully: For the case $L = 5\,\mathrm{m}$ the waves were already very short, so that most of the information was contained in a thin layer below the surface. In order to capture this layer, the grid had to be refined in this region. Still using 11 layers, the relative layer thickness was chosen $\Delta x/H = 0.01$, 0.01, 0.01, 0.02, 0.03, 0.04, 0.05, 0.09, 0.14, 0.20, 0.40 from top to bottom. To achieve wave damping rates similar to those of the experiments done above, Δt had to be chosen such that $\omega\Delta t$ is similar in all three cases. Still, the time steps had to be easily expressible in minutes to comply with the Delft3D-FLOW input file format. Choosing $\Delta t = 0.3\,\mathrm{s}, 0.06\,\mathrm{s}, 0.03\,\mathrm{s}$ met all these requirements.

Figure 4.5 shows the results of the computations. It turns out that for all kH-values the Bijvelds scheme exhibits less damping than the present scheme. The damping per wave period for the present scheme matches well with the analytical predictions. For the Bijvelds scheme the damping is less. In both schemes the damping per wave period did not show any dependency on kH. However, the Bijvelds scheme exhibits a phase error that seems to depend on the wave length. Especially in the cases of $L = 20\,\mathrm{m}$ and $L = 5\,\mathrm{m}$ the phase shift is clearly visible in the plot.

The results give rise to two questions: Why does the Bijvelds scheme exhibit significantly less damping than the present scheme with $\theta = 1$? And why does it have a large phase error when short waves are simulated?

The answer of the first question will be given below. For the second question a first idea is to check if the improved implementation of the dynamic surface condition is the reason. If so, setting $\alpha = 0$ would result in similar phase errors for both schemes. Numerical experiments, however, showed that there was no significant dependency on the choice of α. A case where the implementation of the dynamic surface condition *is* important will be shown in section 4.1.3. It is suspected that the phase error is a result of the particular splitting of the equations. However, further studies, involving a deeper analysis of the Bijvelds scheme, will be required to gain better understanding of the phase error.

The ADI scheme in the prediction step is one of the main differences between the method featured in this thesis and the method of Bijvelds, and the ADI scheme is also the cause for the difference in the damping behavior that was found above. A thorough analysis of the ADI prediction scheme is presented in appendix C, where the analysis of the previous section is extended to a system of three variables in two dimensions. Instead of an amplification factor, an amplification matrix is found. Inspection of the eigenvalues of this matrix reveals the damping and phase error of the scheme. We find the non-trivial eigenvalues

$$\lambda_{2/3} = \frac{1 - \frac{1}{4}(\omega\Delta t)^2 \pm \sqrt{-(\omega\Delta t)^2 + \frac{1}{16}(\omega\Delta t)^4}}{1 + \frac{1}{2}(\omega\Delta t)^2}. \tag{4.7}$$

To be consistent with the notation of the previous section we denote the common absolute value of both eigenvalues $\lambda_{2/3}$ of the ADI scheme as $|c_\mathrm{a}|$. Ignoring the higher order term in the square root, the damping per time step is given by

$$|c_\mathrm{a}| = \frac{1 + \frac{1}{4}(\omega\Delta t)^2}{1 + \frac{1}{2}(\omega\Delta t)^2}. \tag{4.8}$$

As done in the previous section, we can compute the envelope function and the damping per wave period. Adding the envelope function to the plot, see figure 4.6, we see that the damping behavior of the Bijvelds scheme is predicted correctly.

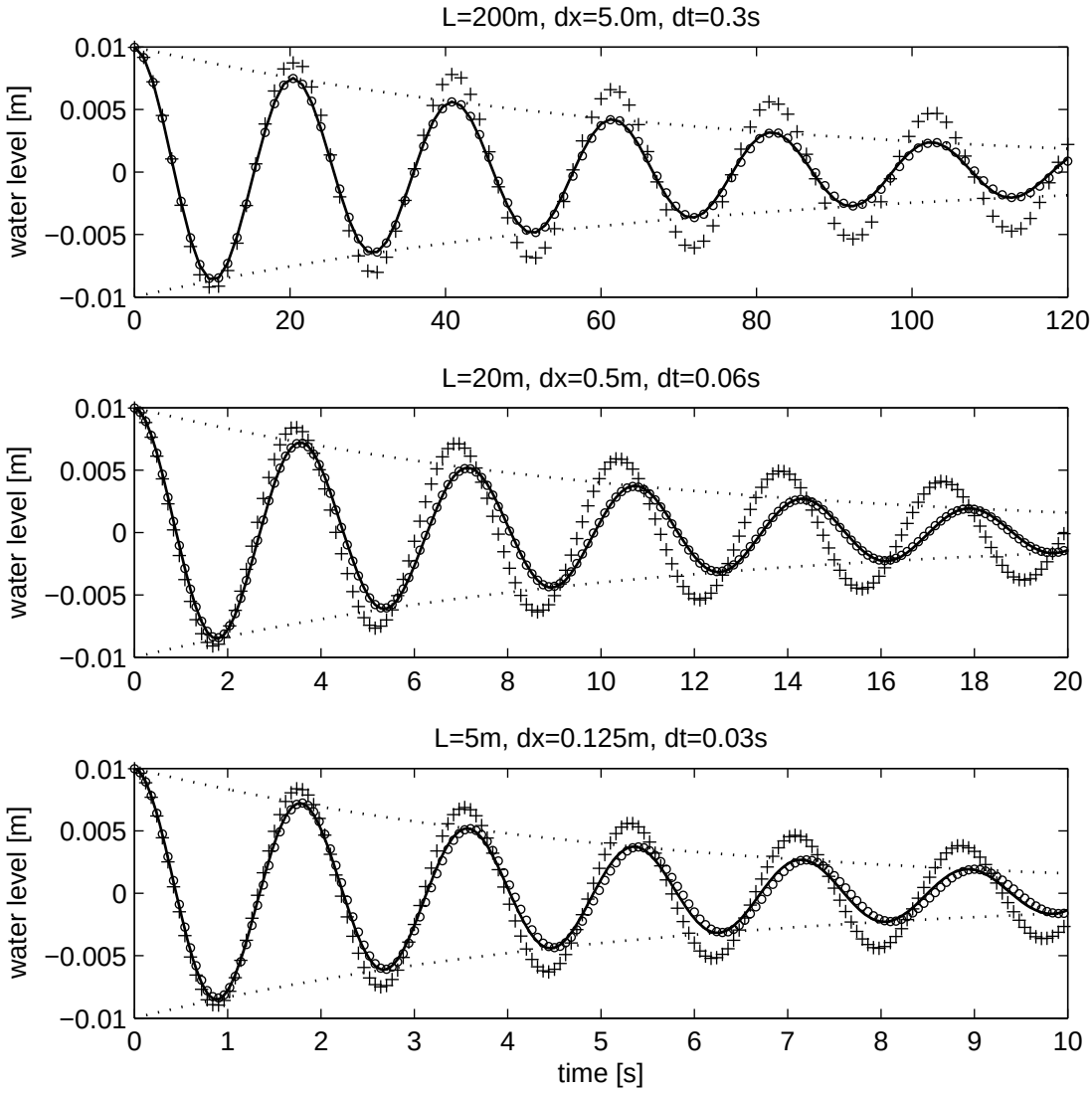

Figure 4.5: Solution of basin test case with different wave lengths, $\theta = 1$ (crosses: pressure split, circles: no pressure split, solid and dotted lines: prediction for un-split scheme). The characteristic dimensionless numbers are $kH \approx 0.31, 3.1, 12.6$ (top to bottom) and $\omega\Delta t \approx 0.092, 0.11, 0.11$, respectively.

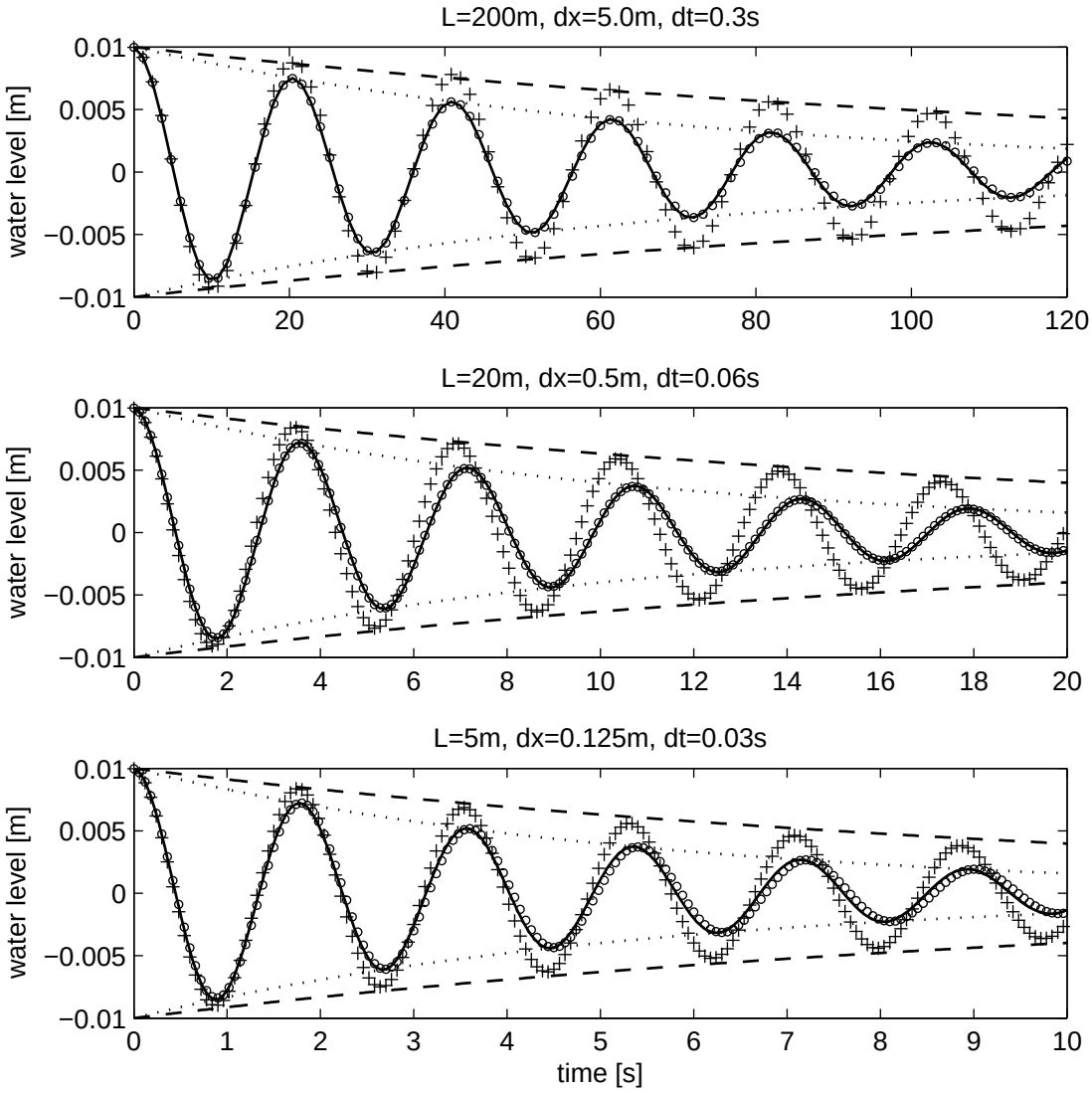

Figure 4.6: Solution of basin test case with different wave lengths, $\theta = 1$ (crosses: pressure split, circles: no pressure split, solid and dotted lines: prediction for un-split scheme). The slashed line shows that the predicted damping behavior of the Bijvelds method after analyzing the ADI prediction scheme.

4.1.2 Damping effect of upwind discretization

The test case in section 4.1.1 revealed that the error due to the implicit time integration method can cause numerical damping. If a higher order scheme is used for the *time* discretization, or if a sufficiently small time step is used, then the error due to the *space* discretization becomes important for the overall accuracy of the solution. It is known that a first order upwind scheme used for discretizing an advection term introduces an error of order $O(\Delta x)$, which acts like an artificial viscosity term. As mentioned at the end of section 2.3, in the non-hydrostatic module of Delft3D-FLOW the horizontal advection terms are discretized with a multidimensional first order upwind scheme, which simplifies to a standard first order upwind scheme in case of 2DV simulations. In cases where the spatial step size or the velocity is large, the error due to the space discretization can no longer be ignored. In this section the effect of this error is studied analytically and numerically.

The practical motivation for the investigation of the error related to the upwind discretization comes from simulations done with the basin test case with $L = 5\,\mathrm{m}$ and $H = 10\,\mathrm{m}$. The wave has a wave number of $k = 2\pi/(5\,\mathrm{m}) \approx 1.26\,\mathrm{m}^{-1}$ and a water level amplitude of 0.1 m. We chose $\Delta t = 0.012\,\mathrm{s}$ and $\theta = 0.5$ to prevent visible time discretization effects. The spatial step was set to $\Delta x = 0.25\,\mathrm{m}$. The resulting horizontal velocity in the top layer is given in the figure 4.7 on the left. Over a longer period of time there is a prominent wave damping, even when $\theta = 0.5$ is used. The damping does not decrease with a smaller time step. Furthermore, the damping is not exactly exponential, as in the case of the damping due to the time discretization scheme. However, the damping is less for longer waves or waves with a smaller amplitude. These findings indicate that the error has its origin in a spatial discretization. More specifically, the first order multi-directional upwind discretization of the horizontal advection terms is thought to be the main reason, since the viscosity was set to a negligibly small value and the vertical advection terms are discretizeted with a second order accurate method.

One way to verify that the upwind terms are really the cause of the damping is to see what happens if these terms are set to zero in the scheme. The result can be seen in figure 4.7 on the right. Now the damping is much less, which means a better numerical approximation of these terms is required to get more accurate results. It must be mentioned,however, that the computations became instable after about 150 s of simulation time and that important physical effects like the steepening of waves and different flow phenomena require the inclusion of the horizontal advection terms in the equations.

To get some more insight in the behavior of the numerical solution, we want to estimate the damping effect of the first order upwind discretized advection term analytically. For simplicity we focus on the term $u\,\partial u/\partial x$, which is included explicitly in the u-momentum equations. For pure wave problems both u and $\partial u/\partial x$ are oscillating around zero and there seems no obvious way to linearize one or the other. However, under certain assumptions it is possible to perform a non-linear analysis. In order to keep the analysis relatively simple we do not take the full numerical scheme into account, but merely consider the one-dimensional equation

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} = f(u(x,t),x,t), \tag{4.9}$$

because we are mainly interested in a qualitative study of the damping introduced by the discretization of the advective terms. Since we do not have $\partial u/\partial x$ in the discretized state space, we have to discretize it with the help of neighboring velocity points, which introduces a space discretization error. To get an estimation of that error we assume that a velocity $u(x)$ is given at a certain fixed time t_0. We assume that the function $f(x,t)$ has such a form that a wave-solution exists and the velocity may be written as

$$u(x) = A(t)e^{i(kx-\omega t)}.$$

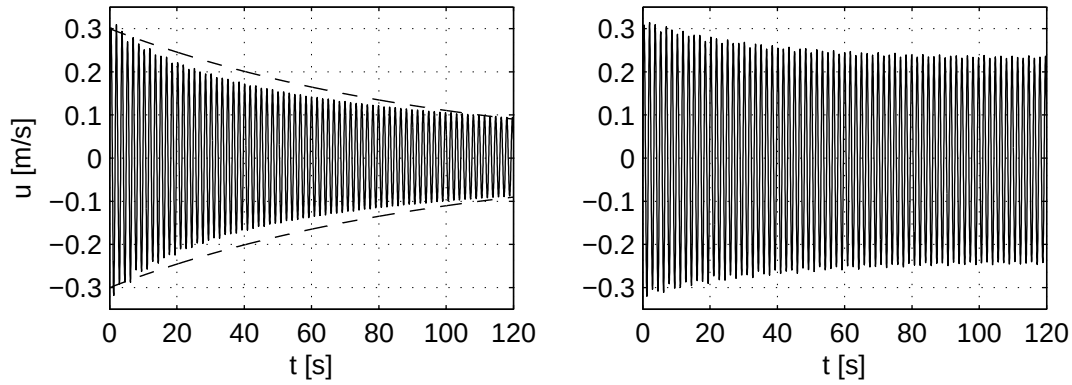

Figure 4.7: Damping effect of the upwind discretization of the horizontal advection terms. The plots shows $u(t)$ in the top layer at a horizontal position of $x = 1.25\,\mathrm{m}$. A wave with a water level amplitude of 0.1 m and a wave length of 5 m in a 10 m deep basin was simulated using 11 layers, $\Delta t = 0.012\,\mathrm{s}$ and $\Delta x = 0.25\,\mathrm{m}$. The plot on the left shows the results obtained with the Borsboom model, in the picture on the right the horizontal advection terms were left out in the equations. The slashed line indicates a simple exponential function, which can not be fitted well to the envelope of the solution.

The exact value of $u\,\partial u/\partial x$ is then given as

$$u(x)\frac{\partial u(x)}{\partial x} = ikA(t)^2 e^{2i(kx-\omega t)} = iku(x)^2.$$

Now we will compute the value of $u\,\partial u/\partial x$ when a first order upwind method is used. In analogy to the expression of the exact value we introduce the notations

$$u(x)\frac{\partial u(x-\frac{\Delta x}{2})}{\partial x} \qquad \text{and} \qquad u(x)\frac{\partial u(x+\frac{\Delta x}{2})}{\partial x}$$

for the values of the upwind discretized terms, depending on the sign of $u(x)$. First, we consider the case of $u > 0$:

$$\begin{aligned} u(x)\frac{\partial u(x-\frac{\Delta x}{2})}{\partial x} &= u(x)\frac{u(x)-u(x-\Delta x)}{\Delta x} \\ &= A(t)e^{i(kx-\omega t)}\frac{A(t)e^{i(kx-\omega t)} - A(t)e^{i(kx-\Delta x-\omega t)}}{\Delta x} \\ &= \frac{A(t)^2}{\Delta x}e^{2i(kx-\omega t)}(1-e^{-ik\Delta x}) \end{aligned}$$

and using a Taylor approximation around $\Delta x = 0$,

$$e^{-ik\Delta x} = 1 - \Delta x ik + \frac{\Delta x^2 i^2 k^2}{2}, \tag{4.10}$$

we can write

$$\begin{aligned} u(x)\frac{\partial u(x-\frac{\Delta x}{2})}{\partial x} &= \frac{A(t)^2}{\Delta x}e^{2i(kx-\omega t)}(1-1+\Delta x ik - \frac{\Delta x^2 i^2 k^2}{2}) \\ &= ikA(t)^2 e^{2i(kx-\omega t)}(1-\frac{\Delta x ik}{2}) \\ &= u(x)\frac{\partial u(x)}{\partial x}(1-\frac{\Delta x ik}{2}). \end{aligned}$$

In case of $u < 0$ the same analysis leads to

$$u(x)\frac{\partial u(x+\Delta x)}{\partial x} = u(x)\frac{\partial u(x)}{\partial x}(1+\frac{\Delta x i k}{2}).$$

The local discretization error is the difference between the exact operator applied to $u(x)$ and the upwind operator applied to $u(x)$. It is given as

$$\begin{aligned}\epsilon &= u(x)\frac{\partial u(x)}{\partial x} - u(x)\frac{\partial u(x\pm\Delta x)}{\partial x}\\ &= \pm u(x)\frac{\partial u(x)}{\partial x}\frac{\Delta x i k}{2}\\ &= \pm i k u(x)^2\frac{\Delta x i k}{2}\\ &= \mp\frac{k^2 u(x)^2\Delta x}{2},\end{aligned}$$

where the right hand side in the last line has reversed signs, so it is negative when $u < 0$. We can also write ϵ for any direction of u as

$$\epsilon = \frac{k^2|u|\Delta x}{2}\cdot u.$$

The relative error with respect to the velocity has the dimension 1/s and is given by

$$\epsilon_{\text{rel}} = \frac{\epsilon}{u} = \frac{k^2|u|\Delta x}{2}.$$

We want to estimate the value of ϵ_{rel} to get an idea of the damping introduced by the upwinding. It is unhandy that ϵ_{rel} is depending on u. By averaging over a period and a wave length of $|u|$ we want to obtain an ϵ_{avg} which is not dependent on time and space anymore. Let the exact solution may be given as $A(t)\sin\omega t\sin kx$. We want to average ϵ_{rel} over $[0, T/2]$ and $[0, \pi/k]$. This means we have to solve the integral

$$\begin{aligned}\epsilon_{\text{avg}} &= \frac{k^2\Delta x}{2}\frac{2}{T}\frac{k}{\pi}\int_0^{\frac{T}{2}}\int_0^{\frac{\pi}{k}}|u(t)|\,dx\,dt\\ &= \frac{k^3\Delta x}{\pi T}\int_0^{\frac{T}{2}}\int_0^{\frac{\pi}{k}}A(t)\sin\omega t\sin kx\,dx\,dt\end{aligned}$$

This integral can not be computed, because we do not know $A(t)$. We can only use the assumption that the damping effect is relatively small within the time interval $[0, T/2]$, which means that the damping takes place on a much larger time scale than the oscillation of the solution. Then the factor $A(t)$ can be put out of the definite integral. We obtain an averaged relative error per second of

$$\begin{aligned}\epsilon_{\text{avg}} &\approx \frac{A(t)k^3\Delta x}{\pi T}\int_0^{\frac{T}{2}}\int_0^{\frac{\pi}{k}}\sin\omega t\sin kx\,dx\\ &\approx \frac{A(t)k^3\Delta x}{\pi T}\frac{2T}{\pi k}\\ &\approx \frac{2A(t)k^2\Delta x}{\pi^2}.\end{aligned}$$

We want to approximate the envelope of the numerical solution using the knowledge about ϵ_{avg}. For this purpose we can say that per time step the upwind scheme introduces a damping error of $\epsilon_{\text{avg}} \cdot \Delta t$. We discretize the velocity amplitude $A(t)$ in time to get A^n, given by the relation

$$A^{n+1} = A^n(1 - \epsilon_{\text{avg}}\Delta t) = A^n \left(1 - \frac{2A^n k^2 \Delta x}{\pi^2}\Delta t\right). \tag{4.11}$$

Starting with some initial amplitude, it is straightforward to solve this equation numerically to get an approximation of $A(t)$.

Note that we have used several assumptions to study the effect of the upwind scheme: We assumed that the damping happens on a sufficiently longer time scale than the oscillation, which is reasonable in the considered test case. Furthermore, we assumed that u is a function of x and t only, in other words, the depth profile is assumed to be uniform. In reality, the velocity decays with the depth, and so does the relative damping, because it is a function of u. However, it would be difficult to include this nonlinear effect in the analysis. As a third assumption we neglected the interdependency of the equations and terms and considered only a single simplified equation. Still, having found a relation for the damping of the simplified example, we should expect that at least qualitatively the results carry over to the complete scheme.

Figure 4.8 compares the approximation of $A(t)$, obtained using equation (4.11), with the numerical solution of the basin test problem as described above. It can be seen that the damping behavior is predicted quite well, even with the analysis of the simplified scheme. This indicates that, indeed, the damping comes from the upwind discretization.

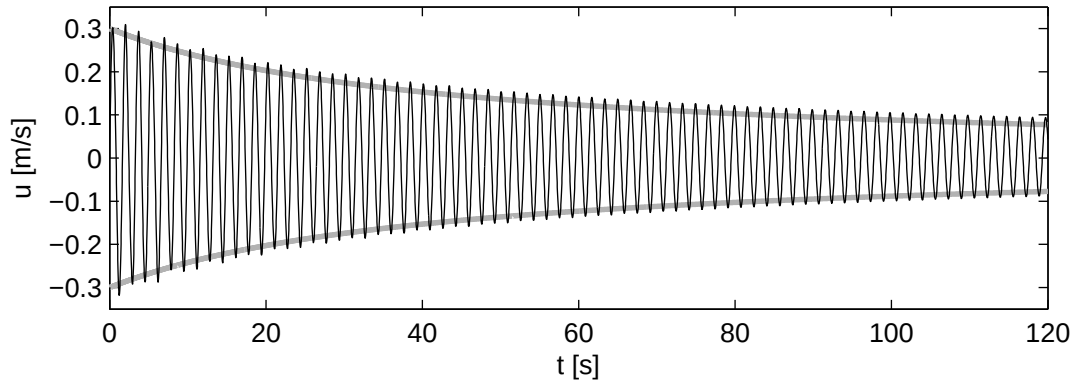

Figure 4.8: Result of the simulation (thin black line) with the analytically predicted velocity amplitude (bold gray line). The result

To eliminate the damping due to the spatial discretization of the advective terms in the horizontal direction, higher order schemes should be considered. A second order central scheme is no possible candidate, because it introduces numerical wiggles when the velocity is large. Higher order upwind schemes or flux-limiter schemes are known to exhibit much less damping than first order upwind. However, after Godunov's theorem higher order scheme are not total variation diminishing (TVD) and, therefore, not completely free of numerical wiggles. This means that the first order upwind scheme is more robust than any higher order alternative, but, as we have seen, not always accurate.

4.1.3 Phase error

To reduce the error introduced by the dynamic condition at the surface, an interpolation factor α was introduced in section 2.5, equations (2.42) and (2.43). When this factor is chosen 0, then the surface pressure discretization used by Bijvelds [2003] is obtained, where the pressure in the center of the surface cell is set to zero. If the water level is everywhere situated close to the cell centers, then the error will still be small in the Bijvelds approach. If, however, the water level is situated close to the upper or lower cell faces, then the error will be large. Furthermore, the vertical cell size has to be small in order to reduce the error.

We consider the basin test case described above, where we investigate the solutions obtained with the new and with the old approach using different displacements of the grid relative to the physical domain, as sketched in figure 4.9. The vertical grid with 11 layers is located between $z_{\text{bot}} = -10.5\,\text{m} + z_{\text{offset}}$ and $z_{\text{top}} = 0.5\,\text{m} + z_{\text{offset}}$. The displacement is given as $z_{\text{offset}} = -0.3\,\text{m}$, $0\,\text{m}$, $0.3\,\text{m}$. With an amplitude of $A_{\zeta,0} = 0.1\,\text{m}$, the water level is still contained completely in one grid layer, so that any effect connected with the movement of the surface through different grid layers is avoided.

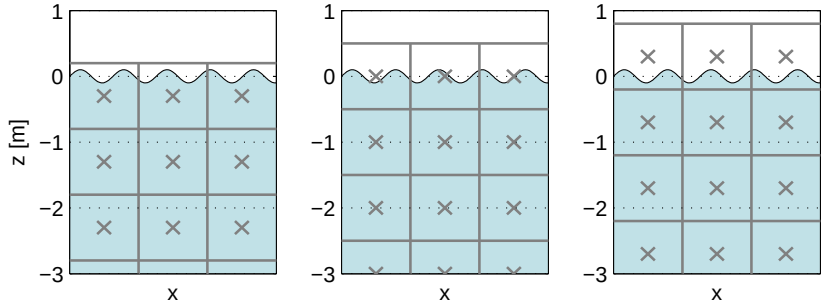

Figure 4.9: Displacement of the grid (gray lines) relative to the physical domain (blue area). In the top cells the distance between the pressure points (crosses) and the water surface (curly line) determines the phase error, when the hydrostatic approximation is used in these cells. From left to right: $z_{\text{top}} = 0.2\,\text{m}, 0.5\,\text{m}, 0.8\,\text{m}$

The results of the computations, which were done using $\theta = 0.5$, $\Delta t = 0.12\,\text{s}$, $\Delta x = 0.5\,\text{m}$ and $L = 20\,\text{m}$, are shown in figure 4.10. For $z_{\text{top}} = 0.5\,\text{m}$ both surface representations lead to results that comply with the analytical result. When the grid is displaced, then the results differ. The hydrostatic approximation in the surface cells leads to a phase error with the following properties:

- When the zero-pressure point is situated *below* the surface ($z_{\text{top}} = 0.2\,\text{m}$), then the wave moves *faster* than the analytical phase speed.
- When the zero-pressure point is situated *above* the surface($z_{\text{top}} = 0.8\,\text{m}$), then the wave moves *slower* than the analytical phase speed.

These statements are true as well for the improved surface condition, as can be seen in the plot, but the phase error is much smaller in this case.

The results show that the grid should always be located such that the pressure points are near the water surface. However, using the improved boundary condition strongly reduces the effect of the grid placement. In many applications with strong surface dynamics it is not possible to keep the distance between surface and pressure points small, therefore it is recommended to use the improved surface representation in cases, where a correct phase speed is important.

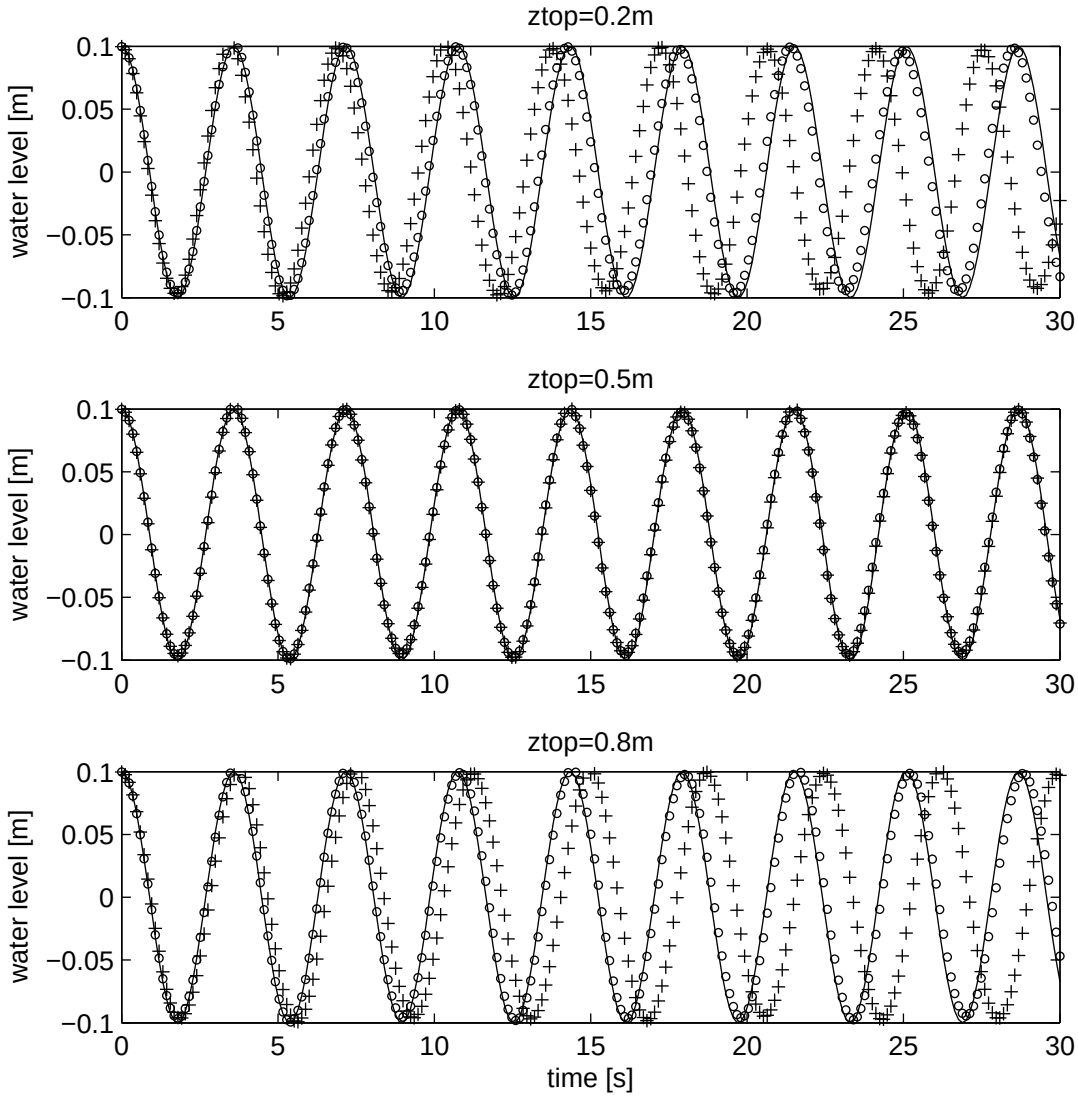

Figure 4.10: Water level $\zeta(t)$ at the left boundary of a closed basin with initial cosine profile. The solid line shows the exact solution, the crosses show the numerical solution using a hydrostatic approximation in the top layer and the circles show the solution obtained with an interpolated surface condition. The grid is placed in such a way that the discretized pressure points are below the free surface (upper plot), near the free surface (middle plot) and above the free surface (lower plot).

4.2 Wave propagation over a submerged bar

After having tested the behavior of the numerical scheme with respect to a short harmonic wave with a fixed wave length, we now want to investigate a case that includes waves with multiple wave lengths and speeds. When waves travel through a domain with a varying depth, higher harmonics may be generated, which travel with a smaller velocity according to the dispersion relation.

A well-known test case to study the accuracy of a numerical scheme with respect to wave dispersion are the experiments performed by Beji and Battjes [1993]. They introduced harmonic water waves at one side of a 30 m long flume with a still water depth of 0.4 m. The waves traveled over a submerged bar, as sketched in figure 4.11. On the other side of the flume an artificial beach was placed to hinder wave reflections. The measurements show that, as the wave travels over the bar, the amplitude steepens and higher harmonics are introduced. At the falling slope of the bar these harmonics become free as they start traveling at different speeds.

The boundary conditions we will compare our simulations with comply with the test case A described by Dingemans [1994]. Sine-shaped surface waves with an amplitude of $A_\zeta = 0.01\,\mathrm{m}$ and a period of $T = 2.02\,\mathrm{s}$ are imposed at the left boundary.

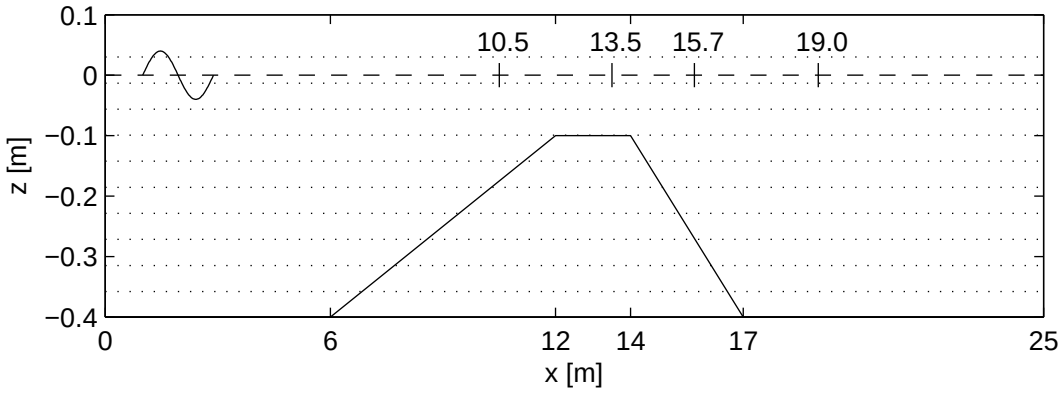

Figure 4.11: Sketch of the geometry used in the Beji-Battjes simulation, including the positions of the considered measurements at $x = 10.5\,\mathrm{m}, 13.5\,\mathrm{m}, 15.7\,\mathrm{m}$ and $19.0\,\mathrm{m}$ and the layer distribution (dotted lines). At the left boundary surface waves with a sine shape are imposed, at the right boundary a sponge layer is implemented.

To simulate the Baji-Battjes test problem A, the domain was discretized with 10 equidistant layers in vertical direction, located between $z_{\mathrm{bot}} = -0.40\,\mathrm{m}$ and $z_{\mathrm{top}} = -0.03\,\mathrm{m}$. With this equidistant layer distribution only three layers remained active on top of the bar. In x-direction the domain between $x = 0\,\mathrm{m}$ and $x = 25\,\mathrm{m}$ was discretized with 2000 grid cells of uniform step size $\Delta x = 0.0125\,\mathrm{m}$. This small step size was found to be necessary to prevent artificial damping of higher harmonics. A sponge layer was used to prevent wave reflection at the right boundary.

For the time discretization a step size of $0.012\,\mathrm{s}$ was used. A criterion for the choice of the time step is given in Delft3D-FLOW user manual [Del, 2006], where it is recommended to choose $\Delta t \leq T/40$. Considering that the period of the highest dominant harmonics found in the experiment is $T \approx 0.5\,\mathrm{s}$, this criterion is just fulfilled. Another constraint is the CFL-number related to the wave speed, which is important when θ is chosen close to 0.5. The fastest waves are the longest waves in the deepest water. Using $\omega \approx \pi\,\mathrm{s}^{-1}$ of the imposed wave in connection with equation (4.1), the wave number of the fastest waves is approximately $k \approx 1.7\,\mathrm{m}^{-1}$. We obtain a CFL-number for the imposed wave of $(\omega/k) \cdot (\Delta t/\Delta x) \approx 1.8$. This gives a clear indication that Δt should not be increased much more. In order to prevent possible stability problems we

used a minimum θ of 0.55.

First, we apply the scheme of Bijvelds [2003] to the model problem. The results shown in figure 4.12 reveal that the wave is damped too much. Most detail with respect to higher harmonics is lost. The method presented in this thesis, using $\theta = 1$, exhibits even more damping, as is illustrated in figure 4.13. This conforms with the findings of section 4.1.1 for the basin test case. If, however, the time discretization scheme is close to the second order Crank Nicolson scheme, the damping can be greatly reduced. Choosing $\theta = 0.55$, as shown in figure 4.14, eliminates most of the damping and the results of the simulation match well with the measurements. In the simulations with the present scheme the improved surface condition was used. Additional experiments using a hydrostatic approximation in the surface cells, setting $\alpha = 0$, resulted in a small but visible effect on the accuracy of the solution, as shown in figure 4.15. Finally, to verify that it is not possible to use a much smaller number of vertical layers with the present scheme, we present the result of a computation with merely 5 layers in figure 4.16.

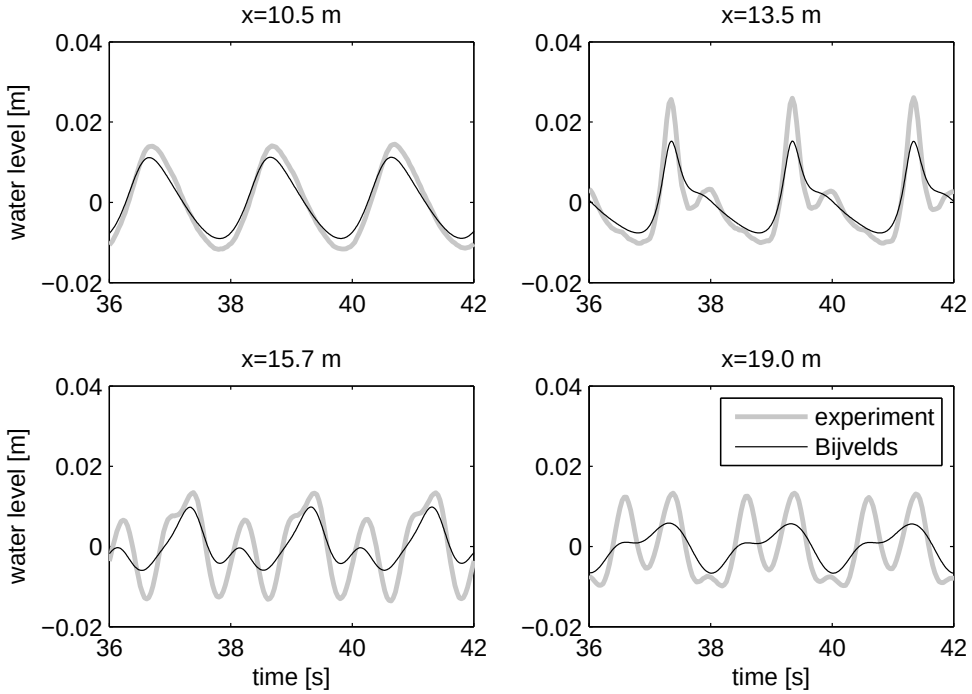

Figure 4.12: Results of a simulation of the Beji-Battjes experiment obtained with the Bijvelds-scheme. The numerical damping is mainly caused by the time discretization and the ADI scheme.

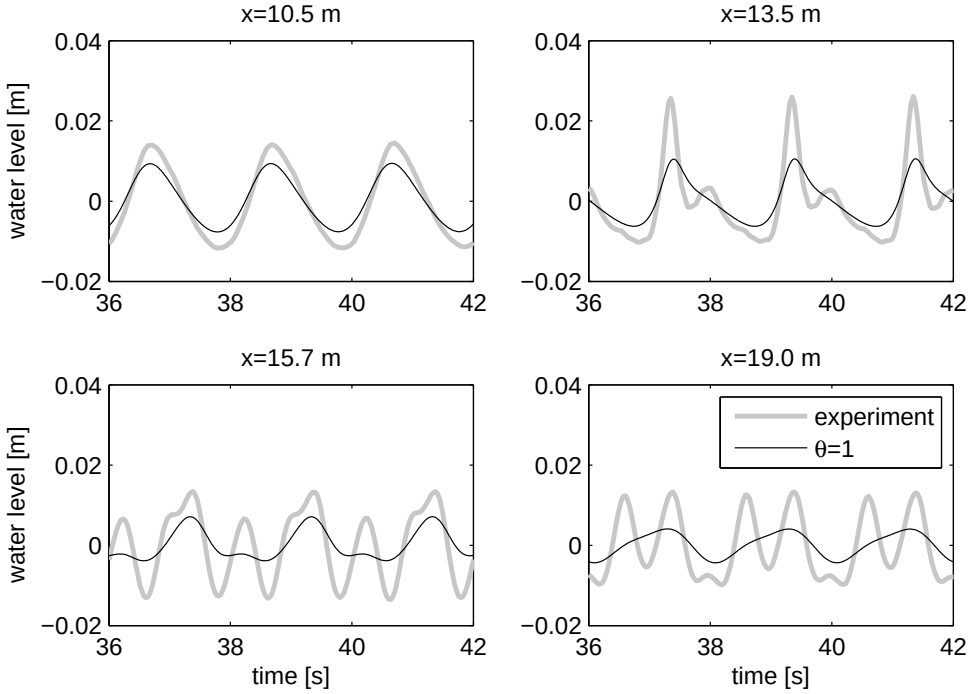

Figure 4.13: Results of a simulation of the Beji-Battjes experiment using the scheme presented in this thesis with $\theta = 1$. Even more damping than in the Bijvelds-scheme is present.

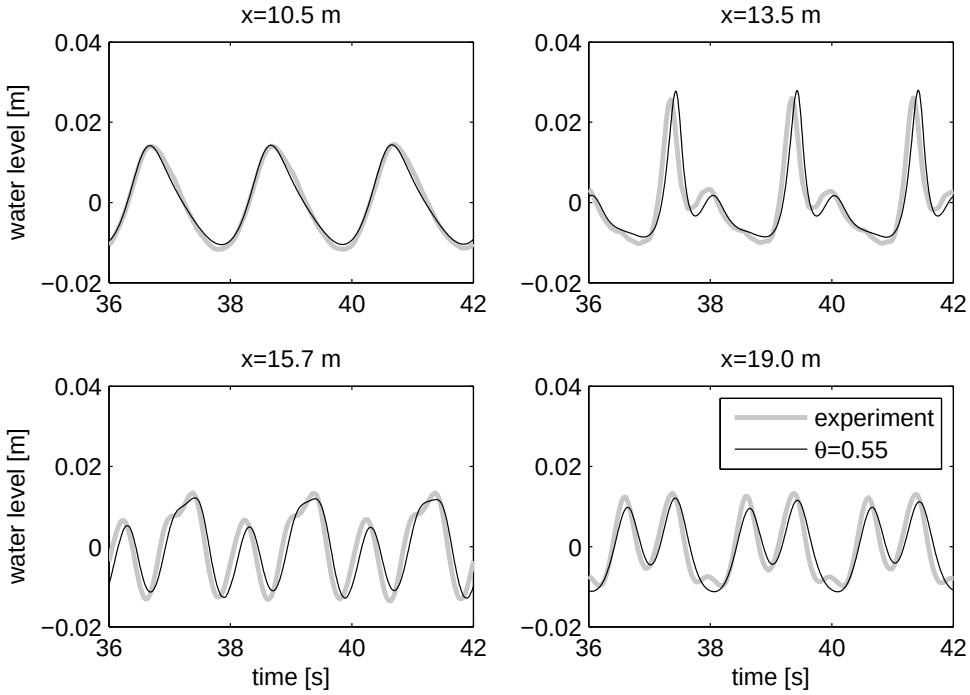

Figure 4.14: Results of a simulation of the Beji-Battjes experiment using the scheme presented in this thesis with $\theta = 0.55$. The almost second-order time discretization scheme eliminates most of the numerical damping.

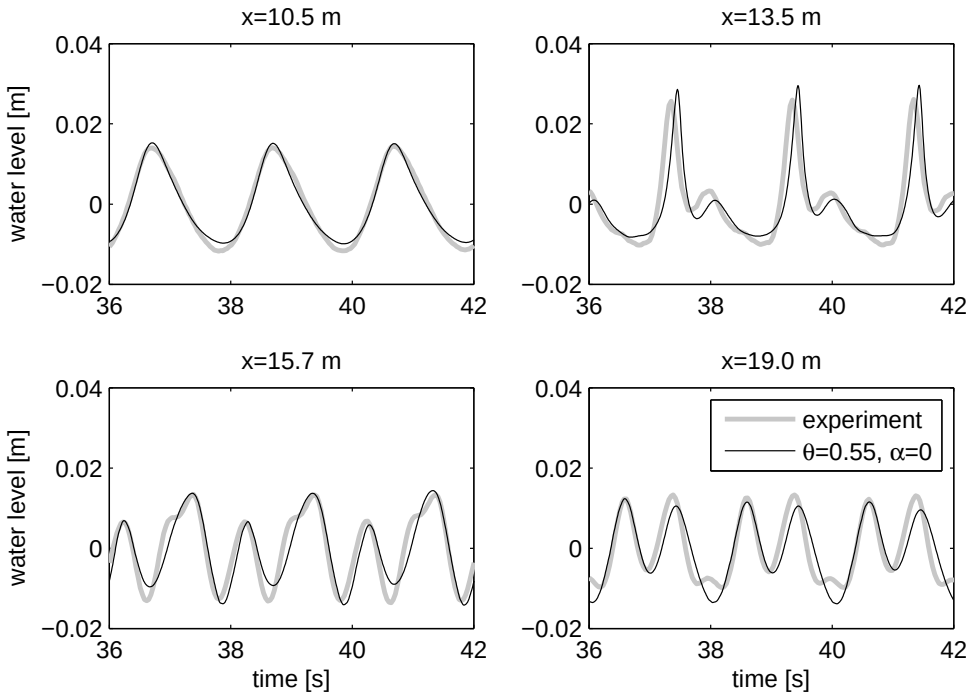

Figure 4.15: Results of a simulation of the Beji-Battjes experiment using the scheme presented in this thesis with $\theta = 0.55$, but with $\alpha = 0$ so the pressure is hydrostatic in the surface cells. In comparison to figure 4.14 the difference between simulation and experiment is slightly larger.

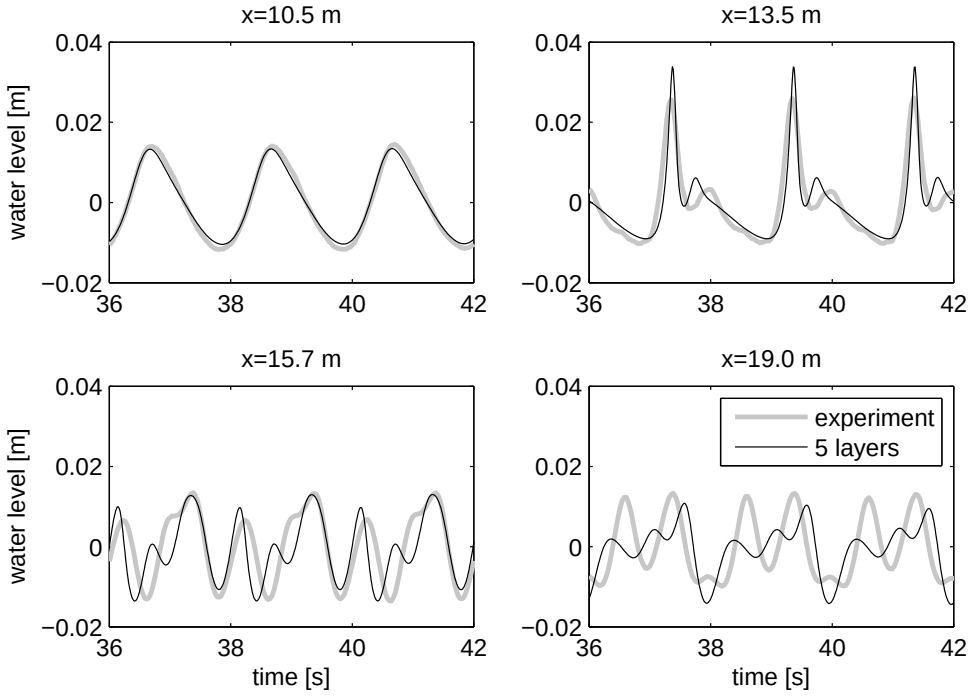

Figure 4.16: Results of a simulation of the Beji-Battjes experiment using the scheme presented in this thesis with $\theta = 0.55$, but discretized with only 5 layers. The scheme fails to compute the correct wave patterns at distances $x = 15.7\,\mathrm{m}$ and $x = 19.0\,\mathrm{m}$

4.3 Density driven flow in a ship lock

In chapter 3 the preconditioned BiCGSTAB solver for the pressure-correction system was described. In this section we analyze the performance of the new solver with a three-dimensional flow test case related to an engineering problem. Apart from that we briefly discuss some physical aspects of the simulation.

4.3.1 Model setup

We consider a 75 m long section of a 61 m wide ship lock that has several inlets in the lower area of the side walls. The 10 inlets in the computational domain have square sections of $2\,\mathrm{m} \times 1.5\,\mathrm{m}$ (width times height) and are located at distances of 15 m apart from each other. Initially the ship lock is filled with 20 m deep water at rest. The water constant initial a salinity of 10 ppt (parts per thousand). At $t = 0\,\mathrm{s}$ the inlets are opened and water with a salinity of 3 ppt streams in with a velocity of 1.08 m/s perpendicular to the side walls. A sketch of the problem is given in figure 4.17. We are interested in the development of the flow and the salinity field as well as the surface elevation during 15 min simulation time. We can compute the expected rise of the mean water level at the end of the simulation, given the size and the number of inlets, the inflow velocity, the area of the ship lock section. We find that a mean water level difference of $H(900\,\mathrm{s}) - H(0\,\mathrm{s}) \approx 6.37\,\mathrm{m}$.

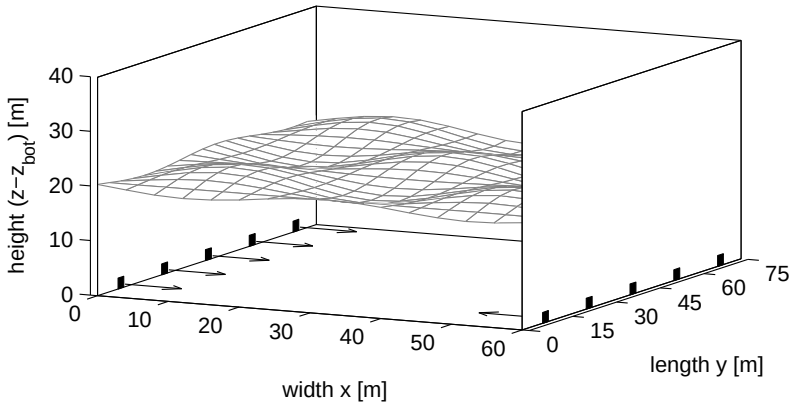

Figure 4.17: Sketch of the ship lock test case.

The domain is discretized with 40 grid layers between $z_{\mathrm{bot}} = -0.5\,\mathrm{m}$ and $z_{\mathrm{top}} = 39.5\,\mathrm{m}$, with a size of the grid cells of $1\,\mathrm{m} \times 1\,\mathrm{m} \times 1\,\mathrm{m}$. The computational domain consists of 96 075 discretization cells at the beginning of the simulation and 123 525 discretization cells at the end of the simulation. The time step is set to 0.3 s, so 3000 time steps will be performed in total. The CFL number with respect to flow was expected to be less than 0.4, based on the inflow velocity. The CFL number with respect to the longest possible waves will be approximately 5, which is rather large and indicates that these waves can not be captured accurately in the model. In order to avoid any possible stability problems with respect to the wave propagation we set $\theta = 1$, which will lead to additional wave damping, as we have seen in previous sections. However, since in this test case our focus is the flow field, we do not need to simulate the wave motion very accurately, as long as it is not expected to influence the flow field significantly.

The test case is run using the BiCGSTAB method in connection with the improved surface condition given by (2.42) and (2.43). A MILU preconditioner with $\alpha_{\mathrm{milu}} = 0.95$ is used to speed up the convergence, as described in section 3.2. The value of α_{milu} was found by performing

several short test runs with different values and observing the number of iterations. Choosing 0.95 yielded heuristically a good convergence rate, but the method was not very sensitive to the choice as long as α_{milu} was chosen within the range $0 \leq \alpha_{\text{milu}} \leq 0.99$. As a stopping criterion (see equation (3.9)) we used

$$||r^i||_\infty < 10^{-3} \quad \texttt{or} \quad ||r^i||_\infty < 10^{-10}||b||_\infty.$$

The small value of 10^{-10} for the second statement was chosen in order to actually let the first statement $||r^i||_\infty < 10^{-3}$ determine the number of iterations.

In order to make the experiment traceable, some other parameters have to be mentioned. The particular choice of the parameters are disputable, but since we are not interested in quantitative results with respect to the flow and salinity field, we only mention them here: The standard k-ϵ turbulence model of Delft3D-FLOW was applied, with background horizontal viscosity and diffusivity of $10^{-2}\,\text{m}^2/\text{s}$ and $10^{-4}\,\text{m}^2/\text{s}$ and background vertical viscosity and diffusivity of $10^{-6}\,\text{m}^2/\text{s}$ and $10^{-7}\,\text{m}^2/\text{s}$ and an Ozmidov length scale of 1 m.

4.3.2 Results of the simulation

The results of the simulations obtained with the present scheme are shown in figures 4.18 to 4.22. In figure 4.18 a cross-section parallel to the x-z-plane is shown, at $y = 38\,\text{m}$, where the middle pair of inlets is located. Since the inflowing water has a lower salt concentration than the water that is already in the lock, rising plumes are formed near the inlets. The velocity in x-direction of the inflowing water is quite large. Therefore, the plumes do not rise straight up, but tilt towards the center of the lock. The water in the rising plume is mixed quickly with the surrounding saltier water. In the beginning the two opposite plumes appear symmetric to each other. Later the symmetry disappears with increasing time, and the flow pattern becomes irregular. At $t = 900\,\text{s}$ we see that the water level has risen about 6 m and the water in the lock is diluted by the inflowing water.

The three-dimensional salinity field at $t = 900\,\text{s}$ is shown as cross sections in different directions in the figures 4.19 to 4.21. Large-scale turbulence can be observed and the rising plumes are tilted randomly also in y-direction, although the inflow velocity is perpendicular to the y-axis.

Pronounced surface patterns are formed as soon as the plumes hit the free surface. This is shown in figure 4.22. The white spots are uprisings from the surrounding water level due to the positive vertical velocity of the velocity in the center of the plume. They are surrounded by regions with a lower surface level, where the velocity is directed downward. With increasing time also the surface patterns become more and more irregular.

4.3.3 Numerical aspects

The initial conditions and the boundary conditions are not fully compatible, because the initial velocity field does not match with the prescribed velocity at the inlets. A few time steps are needed where the numerical scheme has to deal with the resulting transient effect. This is reflected in the number of iterations necessary to solve for the pressure-correction in the beginning of the solution process. In the beginning more than ten iteration were necessary, but within about 20 time steps the number of iterations decreased to merely 2 or 3. The convergence error bound was set to (10^{-4}), which was found to be sufficient for qualitatively good results. A smaller number yielded more iterations, for example when using $||r^i||_\infty < 10^{-7}$ about 6 iterations were required for convergence.

The time step of 0.3 s was chosen with respect to stability. A larger time step lead to eventual break-downs of the simulation. With respect to the accuracy of the flow field a smaller time step

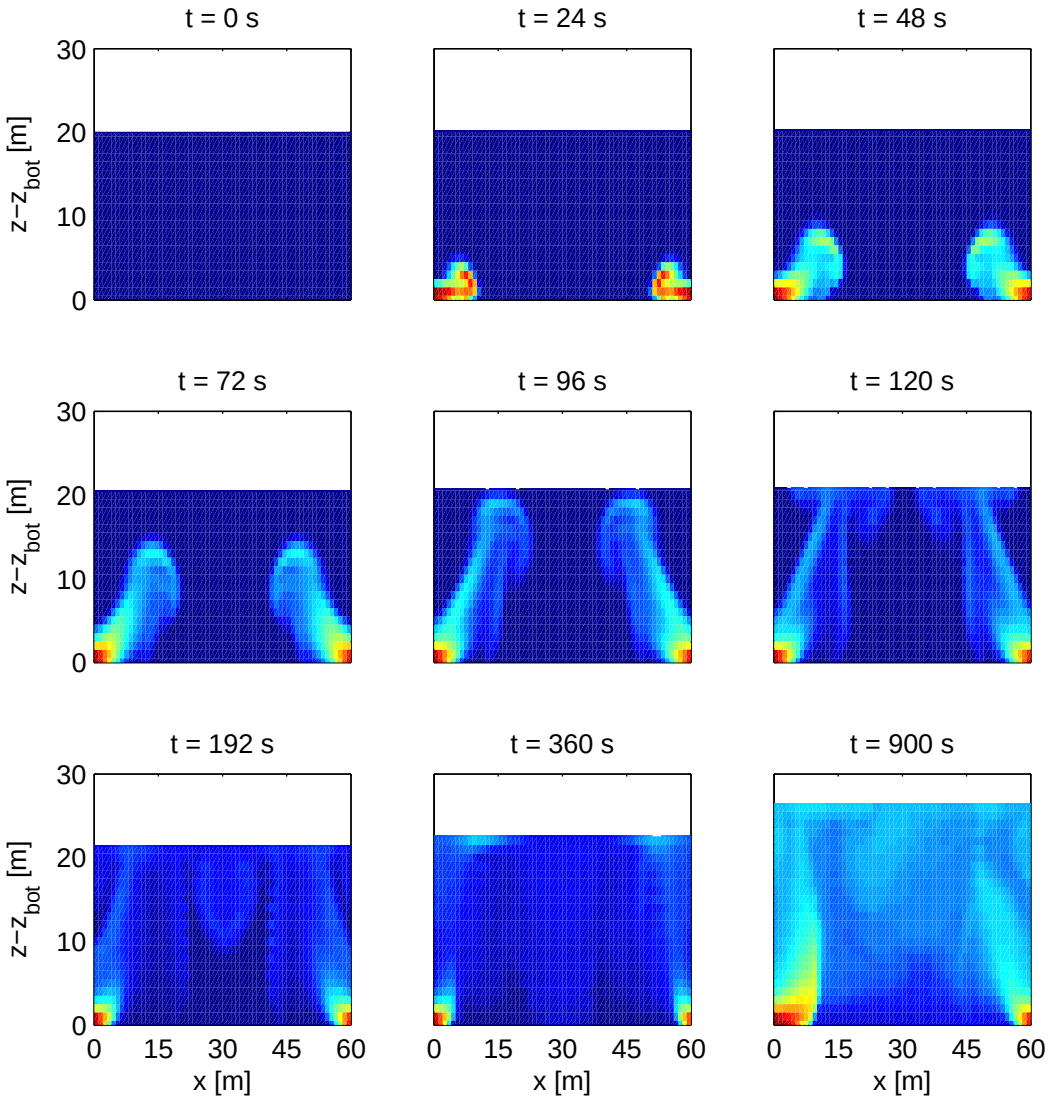

Figure 4.18: A vertical cross-section of the lock at a location of $y = 38\,\mathrm{m}$ at different simulation times. The color scale denotes salinity values between 3 ppm (red) and 10 ppm (blue).

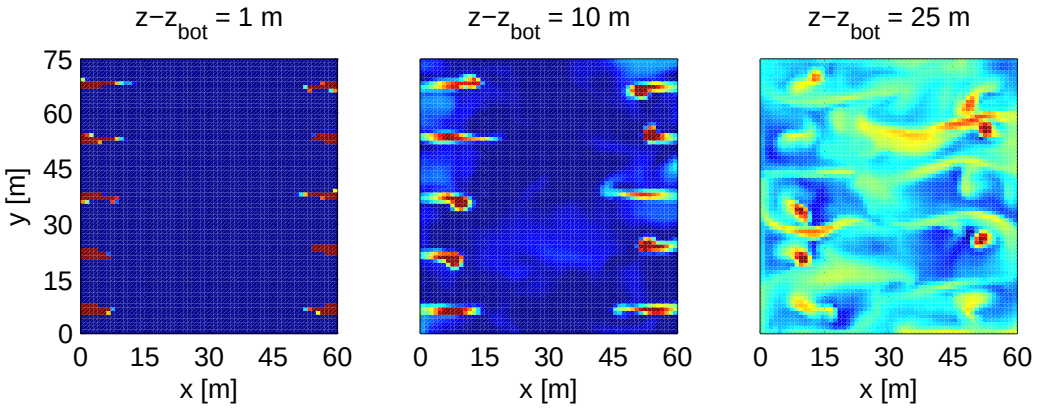

Figure 4.19: Horizontal sections of the lock at different distances from the bottom at a simulation time of $t = 900\,\mathrm{s}$. The color scale denotes salinity values between 7 ppm (red) and 8.5 ppm (blue).

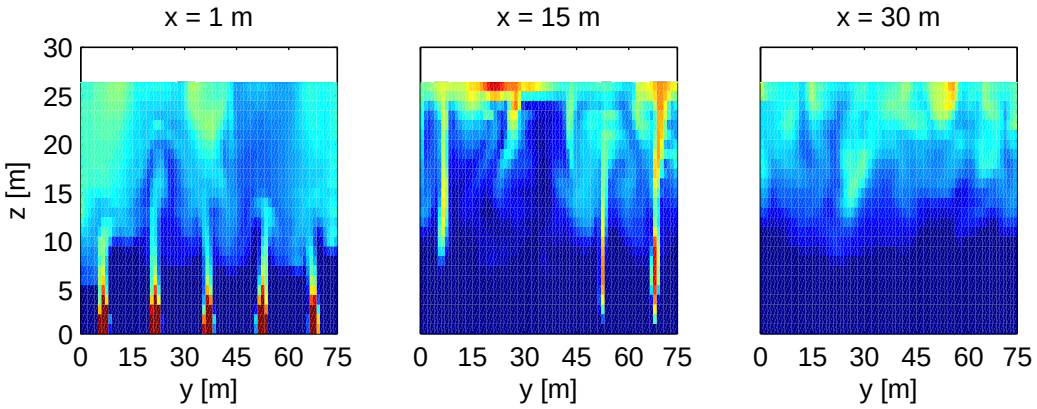

Figure 4.20: Vertical sections along the lock at different x-locations at a simulation time of $t = 900\,\mathrm{s}$. The color scale denotes salinity values between 7 ppm (red) and 8.5 ppm (blue).

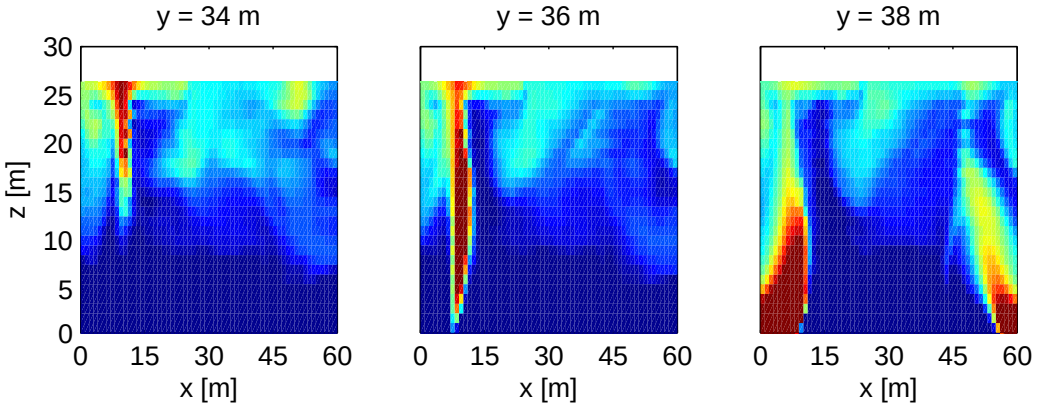

Figure 4.21: Vertical cross sections of the lock at different y-locations near the central pair of jets at a simulation time of $t = 900\,\mathrm{s}$. The color scale denotes salinity values between 7 ppm (red) and 8.5 ppm (blue).

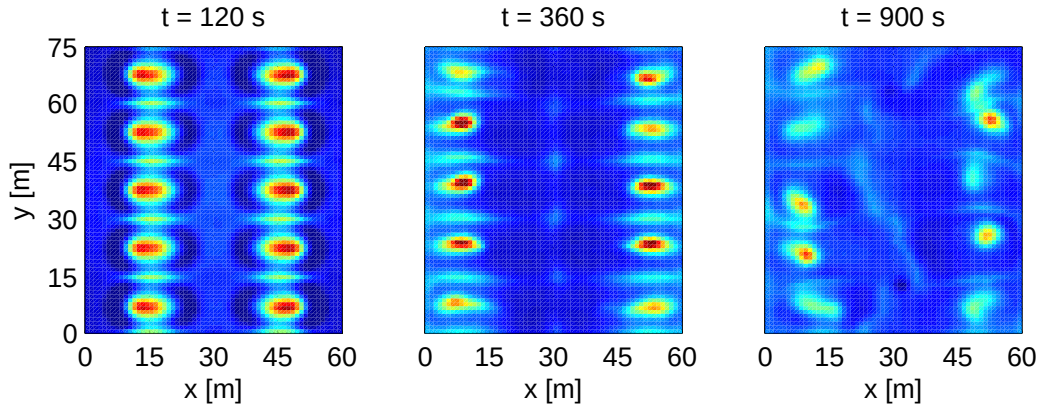

Figure 4.22: Relative water level at different simulation times. The color scale denotes a deviation from the mean water level from -0.5 cm (blue) to 2 cm (red). The difference between the maximum and minimum water level was 2.6 cm, 2.9 cm and 2.0 cm at times 120 s, 360 s and 900 s, respectively.

seems not necessary. Having a smaller time step, however, would result in even less iterations per time step.

It is interesting that an important aspect of the spacial discretization of the momentum equations can be seen in the plots. In figure 4.18, at time $t = 192\,\mathrm{s}$ we see wiggles at the interfaces between sweeter and saltier water at $x \approx 20\,\mathrm{m}$ and $x \approx 40\,\mathrm{m}$. These wiggles are obviously caused by the vertical discretization of the advective terms, which is done with a central scheme. The results show that for strongly non-hydrostatic applications this is not a good choice.

4.3.4 Performance of the solvers

We now want to compare the effect of the solution time of the pressure-correction equation on the total solution time. The pressure solver is an expensive part of the overall solution process in terms of wall-clock time, but the time spent in other parts of the code may still overweight the cost of the pressure solver. In practice, improving the solver only makes sense, if it has a noticeable effect on the *overall* solution process.

The simulation described above will serve as a reference to other combinations of solution methods with preconditioners. We now want to compare the performance of the following two discretizations of the free surface and respective iterative methods:

- BiCGSTAB method with the improved surface discretization, which leads to an unsymmetric system matrix,
- CG method with the hydrostatic assumption in the surface cells, obtained by setting $\alpha = 0$, which leads to a symmetric matrix.

With the comparison between the two approaches, we can estimate how much it costs, in terms of wall clock time, to use the improved surface representation, and if the use of a non-symmetric solver adds a lot of overhead to the total computation time.

The comparison of the solvers and respective surface discretizations is made using also different types of preconditioning:

- no preconditioning,

- diagonal Jacobi preconditioning,
- tridiagonal Jacobi preconditioning,
- MILU(0.95) preconditioning.

The computational speed that is gained through a better preconditioner might compensate for the additional cost related to the non-symmetric solver. Moreover, also the CG method will benefit from a better preconditioner, which means that computations done with the Bijvelds approach could be done more efficiently, too.

Combining the two iterative methods with the four alternative variants of preconditioning lead to a total number of 8 simulations, which were done on a Windows-PC with an AMD Athlon-64 4000+ processor with a clock speed of 2.41 GHz and 2 GB RAM. The simulations of all eight simulations give qualitatively the same result in terms of the flow and salinity field, independent of the methods and preconditioners used. Small quantitative differences can be explained with the onset of turbulence and the respective chaotic behavior of the flow field.

If one and the same simulation is run twice on the same hardware, it will most probably not take the same wall clock time, but there will be some fluctuation. Therefore, all runs were done several times and the simulation time of individual runs were shortened. Doing 9 runs of all eight simulations with a simulation time of 5 min was achievable within reasonable total wall clock time on the hardware used. We additionally performed 9 runs of the standard hydrostatic model of Delft3D-FLOW to estimate the computational overhead that is produced by the non-hydrostatic module.

The mean wall clock times are shown in figure 4.23. There is not much performance difference between the tridiagonal Jacobi preconditioner, the diagonal Jacobi preconditioner and the unpreconditioned iteration. In these three cases the CG method solving the symmetric system is faster than the BiCGSTAB method solving the unsymmetric system. The MILU preconditioner speeds up the computation of both iterative methods. Now the two methods need approximately the same computation times. The BiCGSTAB method seems a bit faster, but the difference is marginal.

It is important to recognize that the speed-up of the non-hydrostatic part of the code is much larger than the speedup of the whole program. The comparison to the hydro*static* code, which does not include the solution of a pressure-correction equation, shows that not much more can be gained by trying to speed up the non-hydrostatic part of the computations even further.

4.3.5 Influence of the aspect ratio

In section 3.2.2 we explained why the tridiagonal Jacobi preconditioner is expected to be a good preconditioner if $\Delta z \ll \Delta x$ and $\Delta z \ll \Delta y$. With a modification of the ship lock test case described above we want to investgate how the MILU and tridiagonal Jacobi preconditioner perform for discretizations with flat cells. We have seen in section 3.2.3 that the MILU preconditioner is able to reduce the conditioner number of the considered matrix more than tridiagonal Jacobi preconditioner. However, the MILU preconditioner is at the same time more expensive than the diagonal Jacobi preconditioner if the work per iteration step or the work in the setup phase is considered. Therefore, it is expected that there is a point where the reduction of iteration numbers of the MILU preconditioner and the efficiency of the tridiagonal Jacobi preconditioner balance each other.

To compare the dependencies of the two preconditioners on the aspect ratio with each other, we modify the test case shown above by compressing the grid and the geometry in vertical direction by factors 2, 4 and 8, respectively. This means that the height of the grid cells, the height of the inlets and the initial water level were decreased by the same factor. Again, we did

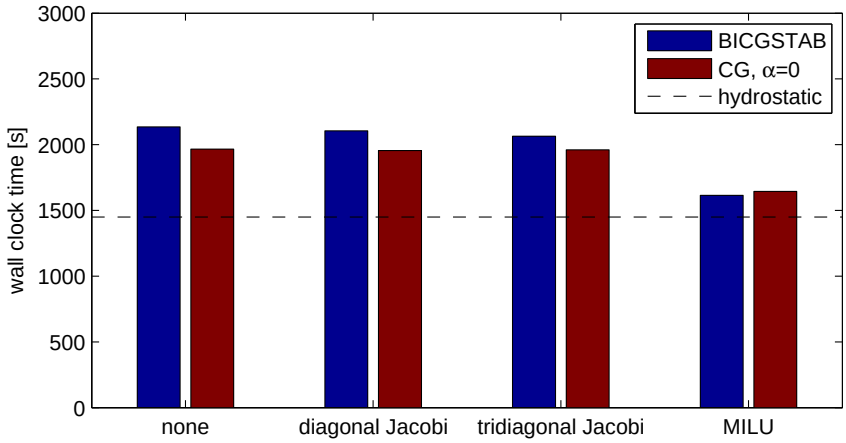

Figure 4.23: Comparison of wall clock times of the ship lock test case with different preconditioners. The simulation time was 5 min. The red bars show the calculation times when the improved surface discretization is used and the pressure-correction is solved with the BiCGSTAB method. The blue bars show the calculation times when the hydrostatic assumption is used in the surface cells ($\alpha = 0$). The slashed line denotes the wall clock time of the hydrostatic computation.

9 runs of each simulation and measured the wall clock times. The respective wall clock timings needed for 5 min simulation time and the mean numbers of iterations are presented in figure 4.24. What we can see is that indeed the tridiagonal Jacobi becomes more efficient than the MILU(0.95) as $\Delta x/\Delta z$ and $\Delta y/\Delta z$ become larger. Although the MILU(0.95) preconditioner reduces the mean number of iterations more than the tridiagonal Jacobi preconditioner for the considered aspect ratios, the computational overhead of the MILU(0.95) preconditioner makes it more inefficient when flat geometries with flat discretization cells are considered.

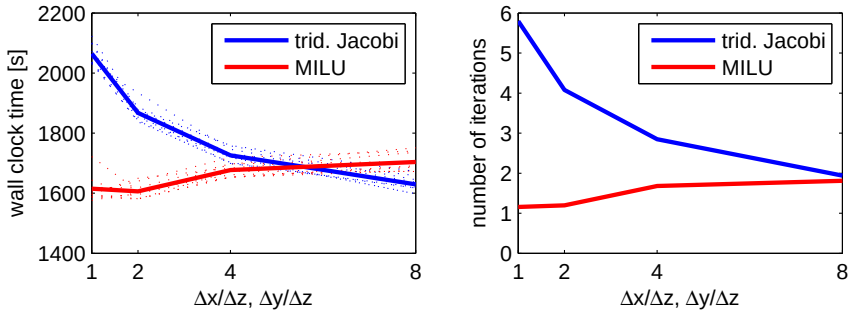

Figure 4.24: Wall clock times and iteration numbers of the ship lock test case solved with the BiCGSTAB method over a simulation time of 5 min. The efficiency of the tridiagonal Jacobi and the MILU preconditioners was tested for different aspect ratios of the problem and the grid. In the left plot the mean wall clock time over 9 runs is shown with a solid line and the individual runs are shown with a dotted line.

4.3.6 Constancy condition

Another aspect of the numerical scheme is demonstrated on basis of the ship lock model. In section 2.6 two methods were presented to update the final velocity field. One possibility was

the use of the momentum correction equations and the other one was the use of the continuity equation. As the main reason for using the continuity equations the mass conservation property was mentioned. In order to test the mass conservation of the non-hydrostatic model in dependency of the method used to update the vertical velocity, we changed the initial salinity of the original ship lock model to a constant field of 3 ppt. If the model is mass conserving then the salinity field should remain unchanged up to machine precision. In section 2.6 we predicted that the maximum divergence will be of the size of the convergence error of the iterative solver. This error will carry over to the salinity field, if the momentum equation are used for the update of the vertical velocity. In the computations we used a stopping criterion with a divergence bound much higher than the machine precision. Therefore, the deviations of the salinity field are expected to be considerable larger than the machine precision. In the computations we use a convergence bound for the iterative method of 10^{-2} instead of 10^{-4}. Figure 4.25 shows the result of the computation with the different update methods. It becomes clear, why the continuity equation should be used for transport problems. The result also verifies that the numerical scheme is strictly mass conservative.

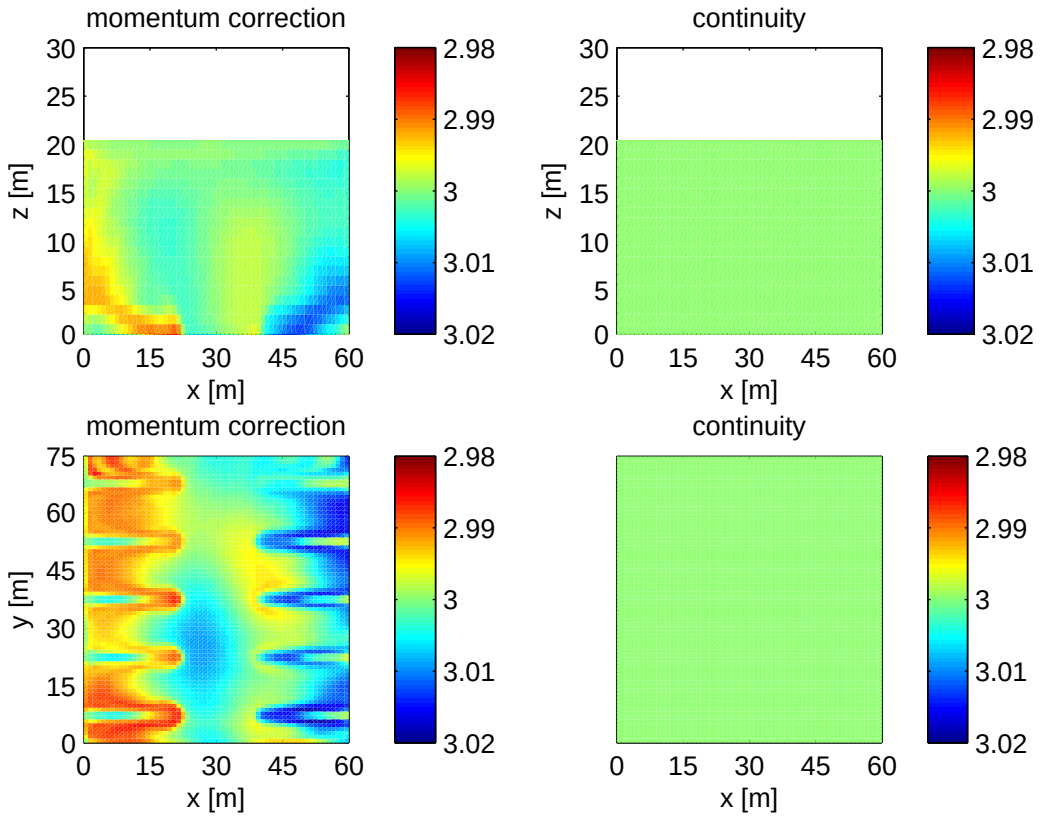

Figure 4.25: Salinity of the $y = 38$ m cross section (top) and the $z = 1$ m horizontal section (bottom) of the ship lock after a simulation time of $t = 60$ s. To test the mass conservation the initial salinity and the inflow salinity were set to 3 ppt. For the update of the vertical velocity the momentum correction equation (left) and the continuity equation (right) were used. When the momentum equation was used, the extremal values throughout the whole domain were 2.987 ppt and 3.013 ppt and the mean was 3.0004 ppt. When the continuity equation was used, then the values differed from 3 ppt with little more than machine precision.

Chapter 5

Conclusions and recommendations

This thesis describes a three-dimensional model to simulate non-hydrostatic free-surface flows. In the presented approach the pressure is not split in its hydrostatic and non-hydrostatic components and the pressure in the surface cells is not assumed to be hydrostatic. The scheme is found to have the following properties:

- The improved implementation of the pressure boundary condition at the free surface leads to a more accurate representation of the phase speed of traveling waves, almost independent of the location of the free surface relative to the grid points.
- The θ-scheme for the time discretization is able to reduce the damping of waves when $\theta \approx 0.5$. Thus, the presented method is much more accurate than the Bijvelds scheme for applications involving wave propagation.
- The first-order multi-dimensional upwind scheme leads to substantial damping when short waves with a significant amplitude are considered.
- The numerical solution of the wave propagation over a bar (Beji-Battjes test case) matches well with the experiment when 10 equidistantly distributed layers are used in the calculation.
- Using the interpolated surface condition gives visible improvements on the result of the Beji-Battjes test case. The choice of the time integration parameter θ, however, is more important than the surface discretization.

The improved surface representation leads to asymmetry in the matrix of the pressure-correction equation. The CG-method, which was already available in Delft3D-FLOW, could not be applied anymore and a non-symmetric solver had to be implemented. The BiCGSTAB method with MILU preconditioning is found to be a good replacement for the CG method:

- When MILU preconditioning is used, the improved surface discretization in combination with the BiCGSTAB method does not add any significant computational cost compared to the previous surface discretization combined with the CG method.
- The tridiagonal Jacobi preconditioner, which was already available in Delft3D-FLOW, is well suited for weakly non-hydrostatic applications with discretization cells that are rather flat. By using the MILU preconditioner for strongly non-hydrostatic flow applications with similar horizontal and vertical grid sizes, the wall clock time of the overall solution process is decreased.

We recall that the main difference between the Bijvelds model and the model presented in this thesis is the pressure handling. Other features, like the discretization of the advective and diffusive terms, are the same in both methods. This leads to the following recommendations:

- Since in strongly non-hydrostatic environments the grid spacing and the velocity components are often of similar size in all directions, the discretization of the vertical advection terms should be reconsidered. For strongly non-hydrostatic applications there is no reason anymore to treat horizontal and vertical terms differently.
- To overcome the damping due to the first order multi-dimensional upwind scheme a more accurate scheme, such as a higher-order upwind or a flux-limiter scheme, should be applied.
- Further research should be done on the applicability of the surface boundary condition with respect to the tangential stress. In the presented test cases the surface gradient was relatively small so that zero stress in horizontal direction was a good approximation. For strongly non-hydrostatic applications with large surface gradients this may no longer be the case.

Appendix A

Splitting error

We are given the numerical scheme

$$f^{n+1} = f^n - \Delta t(Af^n + (B_1 + B_2)f^{n+1}),$$

which is identical to

$$f^{n+1} = (I + \Delta tB_1 + \Delta tB_2)^{-1}(I - \Delta tA)f^n. \tag{A.1}$$

We also consider the split scheme given as

$$\begin{aligned} f^{n+1*} &= f^n - \Delta t(Af^n + B_1 f^{n+1*}) \\ \tilde{f}^{n+1} &= f^{n+1*} - \Delta tB_2\tilde{f}^{n+1}, \end{aligned}$$

or equivalently,

$$\begin{aligned} (I + \Delta tB_1)f^{n+1*} &= (I - \Delta tA)f^n \\ (I + \Delta tB_2)\tilde{f}^{n+1} &= f^{n+1*}, \end{aligned}$$

which can be reduced to

$$(I + \Delta tB_1)(I + \Delta tB_2)\tilde{f}^{n+1} = (I - \Delta tA)f^n,$$

$$(I + \Delta tB_1 + \Delta tB_2 + \Delta t^2 B_1B_2)\tilde{f}^{n+1} = (I - \Delta tA)f^n,$$

and finally

$$\tilde{f}^{n+1} = (I + \Delta tB_1 + \Delta tB_2)^{-1}(I - \Delta tA)f^n - (I + \Delta tB_1 + \Delta tB_2)^{-1}\Delta t^2 B_1B_2\tilde{f}^{n+1}.$$

Using a Taylor series expansion around $\Delta t = 0$ we get

$$\tilde{f}^{n+1} = (I + \Delta tB_1 + \Delta tB_2)^{-1}(I - \Delta tA)f^n - \Delta t^2 B_1B_2\tilde{f}^{n+1} + O(\Delta t^3). \tag{A.2}$$

The splitting error is given as

$$\epsilon = \tilde{f}^{n+1} - f^{n+1}.$$

Comparing (A.1) and (A.2) we see that

$$\epsilon = \Delta t^2 B_1B_2\tilde{f}^{n+1} + O(\Delta t^3).$$

Appendix B

Alternative surface pressure-correction equation

In section 2.5 we have derived the pressure correction equations. The equations for the surface cells contained the variable q at the old time level. It also possible to formulate the equations without using q, so that it is not necessary to compute the non-hydrostatic pressure at all.

We start by linearizing the pressure condition at the new time level around the water level at the old time level, which yields

$$0 = p^{n+1}|_{z=\zeta^{n+1}} \approx p^{n+1}|_{z=\zeta^n} + (\zeta^{n+1} - \zeta^n)\frac{\partial p^n}{\partial z}|_{z=\zeta^n} = 0.$$

We have pressure variables only at discrete locations along the z-axis. Therefore we realize the pressure at the free surface with a linear inter-/extrapolation using the pressure of the cell below the surface cell,

$$p^{n+1}|_{z=\zeta^n} \approx p^{n+1}|_{z=z_k} + \frac{z_k - \zeta^n}{z_k - z_{k-1}}\left(p^{n+1}|_{z=z_{k-1}} - p_k^{n+1}|_{z=z_k}\right).$$

For the second term we use

$$(\zeta^{n+1} - \zeta^n)\frac{\partial p^n}{\partial z}|_{z=\zeta^n} = (\zeta^{n+1} - \zeta^n)\Big(- g + \frac{\partial q^{n+1}}{\partial z}|_{z=\zeta^n}\Big) \approx -g(\zeta^{n+1} - \zeta^n),$$

to get the following linearized pressure condition at the surface:

$$0 = p^{n+1}|_{z=z_k} + \frac{z_k - \zeta^n}{z_k - z_{k-1}}\left(p^{n+1}|_{z=z_{k-1}} - p_k^{n+1}|_{z=z_k}\right) - g(\zeta^{n+1} - \zeta^n),$$

which can be written for discretized variables as

$$\alpha_{i,j}^n p_{i,j,k-1}^{n+1} + (1 - \alpha_{i,j}^n)p_{i,j,k}^{n+1} - (\zeta^{n+1} - \zeta^n)g = 0, \qquad k_{\min}^{i,j} < k = k_{\max}^{i,j,n} \tag{B.1}$$

with

$$\alpha_{i,j}^n = \frac{z_k - \zeta_{i,j}^n}{z_k - z_{k-1}} = \frac{z_{k+1/2} + z_{k-1/2} - 2\zeta_{i,j}^n}{z_{k+1/2} - z_{k-3/2}}, \qquad k_{\min}^{i,j} < k = k_{\max}^{i,j,n}. \tag{B.2}$$

It is easy to verify that by choosing $\alpha_{i,j}^n = 0$ we do *not* obtain a formula equivalent to (2.39).

The aim of using the pressure condition was to eliminate the water level term in the discretized momentum correction equation (2.36), where we have the unknown Δp instead of p^{n+1}. Therefore, we rewrite equation (B.1) as

$$\zeta^{n+1} - \zeta^n = \frac{\alpha_{i,j}^n}{g}(\Delta p_{i,j,k-1} + p_{i,j,k-1}^{n+1}) + \frac{(1 - \alpha_{i,j}^n)}{g}(\Delta p_{i,j,k} + p_{i,j,k}^{n+1})$$

Now we are able to eliminate the water levels from (2.36) to obtain the *pressure-correction equation for surface cells*

$$
\begin{aligned}
&- \frac{\alpha^n_{i,j}\Delta p_{i,j,k-1} + (1-\alpha^n_{i,j})\Delta p_{i,j,k}}{g\Delta t} \\
&\qquad + \theta_\zeta \Delta t \theta \Big(D_{xx}(\Delta p) + D_{yy}(\Delta p) - \frac{\Delta p_{i,j,k} - \Delta p_{i,j,k-1}}{\frac{1}{2}(\Delta z^n_{i,j,k-1} + \Delta z^n_{i,j,k})} \Big) \\
&= \theta_\zeta \Big(D_x(u^{n+1*}) + D_y(v^{n+1*}) + w^{n+1*}_{i,j,k-1/2} \Big) \\
&\qquad + (1-\theta_\zeta) \Big(D_x(u^n) + D_y(v^n) + w^n_{i,j,k-1/2} \Big) \\
&\qquad + \frac{\alpha^n_{i,j} p^n_{i,j,k-1} + (1-\alpha^n_{i,j}) p^n_{i,j,k}}{g\Delta t}, \qquad k^{i,j}_{\min} < k = k^{i,j,n}_{\max}.
\end{aligned}
$$

Appendix C

Analysis of the ADI scheme

In the following the schemes that are important for understanding the prediction step in Delft3D-FLOW are investigated by calculating the eigenvalues of the amplification matrix, which give useful information about the damping and the phase error of the numerical solution. In the analysis only the time dependency of the schemes is studied.

The Bijvelds scheme features an ADI scheme to spatially uncouple the unknowns in the prediction step, while for example Casulli [1999] solves the coupled system iteratively with a conjugate gradient method. In a hydrostatic environment ($q = 0$) and assuming the advective and viscous terms are small the ADI scheme can be written in a simplified, linearized form as

$$\begin{aligned}
\frac{u^{n+1*} - u^n}{\Delta t} &= -g\frac{\partial \zeta^{n+1/2*}}{\partial x} \\
\frac{\zeta^{n+1/2*} - \zeta^n}{\Delta t} &= -\frac{H}{2}\left(\frac{\partial u^{n+1*}}{\partial x} + \frac{\partial v^n}{\partial y}\right) \\
\frac{v^{n+1*} - v^n}{\Delta t} &= -g\frac{\partial \zeta^{n+1*}}{\partial y} \\
\frac{\zeta^{n+1*} - \zeta^{n+1/2*}}{\Delta t} &= -\frac{H}{2}\left(\frac{\partial u^{n+1*}}{\partial x} + \frac{\partial v^{n+1*}}{\partial y}\right),
\end{aligned}$$

where H is the mean water level. It is assumed that the amplitude of ζ is small compared to the total water depth. For the analysis of the scheme we assume that the variables u, v and ζ are harmonic waves, given by

$$u = u_0\, e^{i(k_x x + k_y y - \omega t)}, \qquad v = v_0\, e^{i(k_x x + k_y y - \omega t)}, \qquad \zeta = \zeta_0\, e^{i(k_x x + k_y y - \omega t)}.$$

Putting these solutions in the scheme, we obtain

$$\begin{aligned}
u^{n+1*} &= u^n - \Delta t g i k_x \zeta^{n+1/2*} \\
\zeta^{n+1/2*} &= \zeta^n - \Delta t \frac{H}{2} i(k_x u^{n+1*} + k_y v^n) \\
v^{n+1*} &= v^n - \Delta t g i k_y \zeta^{n+1*} \\
\zeta^{n+1*} &= \zeta^{n+1/2*} - \Delta t \frac{H}{2} i(k_x u^{n+1*} + k_y v^{n+1}),
\end{aligned}$$

By substituting the second equation in the first and fourth equation, $\zeta^{n+1/2*}$ can be eliminated. The resulting three equations can be written in form of a system $U^{n+1*} = AU^n$ with $U^{n+1*} = (u^{n+1*}, v^{n+1*}, \zeta^{n+1*})^T$ and $U^n = (u^n, v^n, \zeta^n)^T$. The amplification matrix A has three

eigenvalues. One eigenvalue is given by $\lambda_1 = 1$, the other two are given by

$$\lambda_{2/3} = \frac{1 - \frac{1}{4}(\omega_x^2 + \omega_y^2)\Delta t^2 \pm \sqrt{-(\omega_x^2 + \omega_y^2)\Delta t^2 + \frac{1}{16}(\omega_x^2 - \omega_y^2)^2 \Delta t^4}}{(1 + \frac{1}{2}\omega_y^2 \Delta t^2)(1 + \frac{1}{2}\omega_x^2 \Delta t^2)}, \tag{C.1}$$

where $\omega_x = k_x\sqrt{gH}$ and $\omega_y = k_y\sqrt{gH}$ are the angular frequencies in the respective directions.

The eigenvalue $\lambda_1 = 1$ is related to the eigenvector $\vec{v}_1 = (-\omega/k_x, \omega/k_y, 0)$, which is perpendicular to the direction of wave propagation. When waves traveling along one of the coordinate axes are considered, one of the momentum equations can be left out and, as a result, only the eigenvectors λ_2 and λ_3 remain. These are related to the propagation of the velocity and the surface elevation through space. They contain both real and imaginary components for small values of Δt. The eigenvalues depend on the direction of the wave. To clarify this, we assume an absolute angular frequency ω that satisfies $\omega^2 = \omega_x^2 + \omega_y^2$. For example, a wave that travels exactly in x-direction is obtained by substituting $\omega_x = \omega$ and $\omega_y = 0$ in (C.1), which leads to

$$\lambda_{2/3} = \frac{1 - \frac{1}{4}(\omega\Delta t)^2 \pm \sqrt{-(\omega\Delta t)^2 + \frac{1}{16}(\omega\Delta t)^4}}{1 + \frac{1}{2}(\omega\Delta t)^2}.$$

A wave that travels in a 45-degree angle to the x-axis is characterized by $\omega_x = \omega/\sqrt{2}$ and $\omega_y = \omega/\sqrt{2}$. Now (C.1) simplifies to

$$\lambda_{2/3} = \frac{1 - \frac{1}{4}(\omega\Delta t)^2 \pm \sqrt{-(\omega\Delta t)^2}}{1 + \frac{1}{2}(\omega\Delta t)^2 + \frac{1}{16}(\omega\Delta t)^4}.$$

It can be shown that for sufficiently small Δt the directional anisotropy becomes smaller than other errors. A Taylor expansion of (C.1) reveals that the directional dependency decreases third order in time:

$$\lambda_{2/3} = 1 \pm i\omega\Delta t - \frac{3}{4}\omega^2\Delta t^2 \pm \frac{17\omega_x^4 + 30\omega_x^2\omega_y^2 + 17\omega_y^4}{32\omega^2} i\omega\Delta t^3 + O(\Delta t^4).$$

After having studied the ADI scheme, we now consider a θ-scheme for the prediction step in a hydrostatic environment. The linearized, simplified scheme is given as

$$\begin{aligned}
\frac{u^{n+1*} - u^n}{\Delta t} &= -\theta g \frac{\partial \zeta^{n+1*}}{\partial x} - (1-\theta) g \frac{\partial \zeta^n}{\partial x} \\
\frac{v^{n+1*} - v^n}{\Delta t} &= -\theta g \frac{\partial \zeta^{n+1*}}{\partial y} - (1-\theta) g \frac{\partial \zeta^n}{\partial y} \\
\frac{\zeta^{n+1*} - \zeta^n}{\Delta t} &= -\theta H \left(\frac{\partial u^{n+1*}}{\partial x} + \frac{\partial v^{n+1*}}{\partial y} \right) - (1-\theta) H \left(\frac{\partial u^n}{\partial x} + \frac{\partial v^n}{\partial y} \right).
\end{aligned}$$

Using the same assumptions as in the previous analysis, the scheme is written as

$$\begin{aligned}
u^{n+1*} &= u^n - \Delta t g i k_x (\theta \zeta^{n+1*} + (1-\theta)\zeta^n) \\
v^{n+1*} &= v^n - \Delta t g i k_y (\theta \zeta^{n+1*} + (1-\theta)\zeta^n) \\
\zeta^{n+1*} &= \zeta^n - \Delta t H i k_x (\theta u^{n+1*} + (1-\theta)u^n) - \Delta t H i k_y (\theta v^{n+1*} + (1-\theta)v^n)
\end{aligned}$$

The eigenvalues of the amplification matrix are $\lambda_1 = 1$ and

$$\lambda_{2/3} = \frac{1 - \theta(1-\theta)(\omega\Delta t)^2 \pm \sqrt{-(\omega\Delta t)^2}}{1 + \theta^2(\omega\Delta t)^2},$$

with $\omega^2 = \omega_x^2 + \omega_y^2$. This shows that the θ-scheme without ADI splitting does not introduce a directional anisotropy.

Now we want to investigate the damping and the phase error of both the ADI and the θ-scheme. By investigating the damping we are not only able to assess the accuracy of a numerical scheme, but also its stability. A scheme is only stable if none of the possible wave modes is amplified. The phase error is important to describe the capabilities of a numerical scheme to obtain the correct speed of propagating waves.

The damping per time step is given by the absolute value of the complex eigenvalues. Figure C.1 shows the damping factors per time step of the considered prediction schemes. For the ADI scheme the damping in two different wave directions is plotted and for the θ-scheme the damping for $\theta = 1$ is shown. It can be seen in the plots that the directional dependency of the ADI scheme is small compared to the overall damping error, especially for small $\omega\Delta t$. In practical applications the time step with respect to the angular frequency is chosen such that $\omega\Delta t$ is much smaller than 1, because much more than $2\pi \approx 6$ time steps are needed to resolve a complete wave period accurately. Therefore, with respect to accuracy only the plot on the right side of figure C.1 is important. With respect to stability all modes must be stable. Therefore, the full scale of $\omega\Delta t$ is important. Both bifurcation branches stay below 1 even as $\omega\Delta t \to \infty$, so the scheme is stable.

The main reason for the study of the prediction schemes was that the damping of the Bijvelds scheme was different from a θ-scheme with $\theta = 1$. The plots show that the particular ADI scheme is the reason. It is noted that by choosing $\theta = 1/2$ in the θ scheme, we are able to get zero damping. This flexibility is the reason that the θ-scheme is preferred over the fixed implementation of the prediction scheme of Bijvelds.

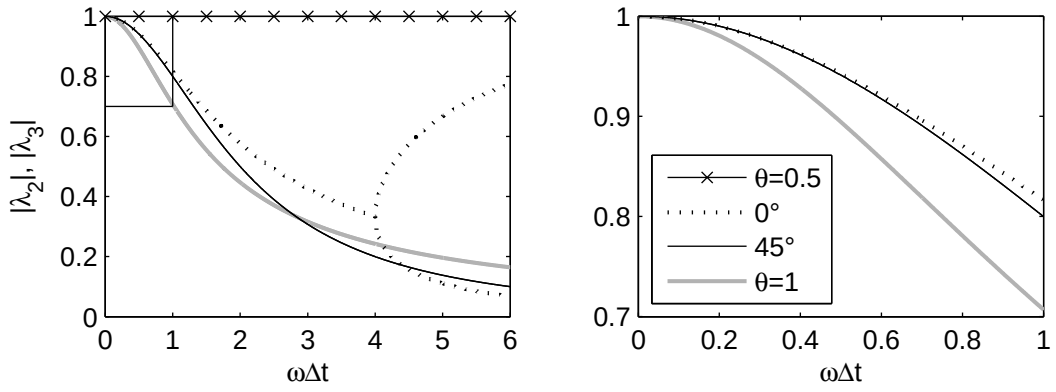

Figure C.1: Damping behavior of the considered ADI scheme with respect to the angle between the wave direction and the x-axis(0, 45). For comparison, the θ-scheme with $\theta = 0.5$ and $\theta = 1$ is shown. The plot on the right shows a magnification of the box shown in the upper-left corner of the plot on the left.

So far we have analyzed the damping behavior of the prediction scheme. To analyze the phase shift, we consider again the change of the solution after a single time step. The phase shift of the *numerical* solution is $\arg(\lambda_{2/3})$ and the phase shift of the *true* solution is $\pm\omega\Delta t$. Figure C.2 shows $\arg(\lambda_{2/3})/(\omega\Delta t)$, which is the relative phase shift of the numerical solution with respect to the true solution. The deviation from 1 or -1, respectively, is the phase error. In the figure also the phase shift of the Crank-Nicolson scheme ($\theta = 0.5$) is shown. Although we have seen above that the damping error was zero, we observe that the scheme has a non-zero phase error. It is interesting that the phase shift of the Crank-Nicolson scheme is the same as

the phase shift of the ADI scheme with an 45-degree angle between the wave and the coordinate axes. The angular dependency of the phase error, however, is small compared to the total phase error. The figure also shows that the implicit Euler scheme exhibits a large phase error.

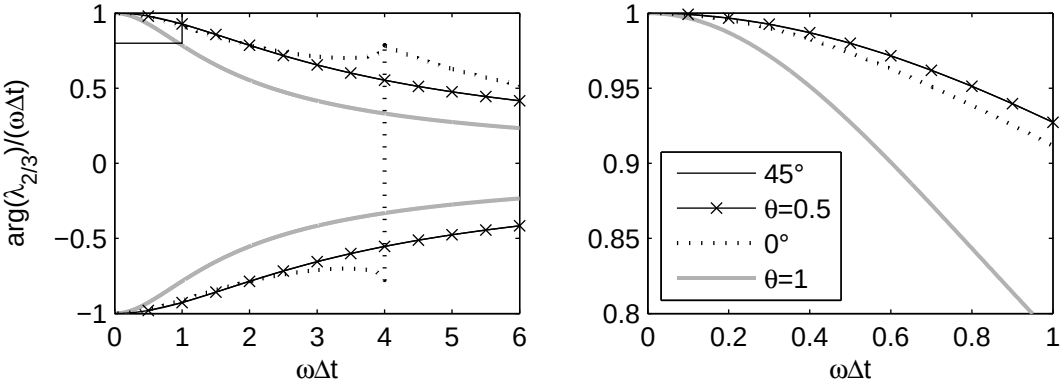

Figure C.2: Relative phase error of the considered ADI scheme with respect to the angle between the wave direction and the x-axis. For comparison, the θ-schemes with $\theta = 0.5$ and $\theta = 1$ are shown. The plot on the right shows a magnification of the box shown in the upper-left corner of the plot on the left.

Bibliography

A. Arakawa and V.R. Lamb. Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in Computational Physics*, 17:173–265, 1977.

S. Armfield and R. Street. An analysis and comparison of the time accuracy of fractional-step methods for the Navier-Stokes equations on staggered grids. *International Journal for Numerical Methods in Fluids*, 38(3):255–282, 2002.

R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

S. Beji and J.A. Battjes. Experimental investigation of wave propagation over a bar. *Coastal Engineering*, 19(1-2):151–162, 1993.

M. Bijvelds. *Numerical modelling of estuarine flow over steep topography*. PhD thesis, TU Delft, 2001.

M. Bijvelds. A non-hydrostatic module for Delft3D-Flow. Technical report, WL|Delft Hydraulics, 2003.

M. Borsboom. Discretization of physical domain and equations. Memo, October 2007.

M.M. Busnelli. *Numerical Simulation of Free Surface Flows with Steep Gradients*. PhD thesis, TU Delft, 2001.

V. Casulli. A semi-implicit finite difference method for non-hydrostatic, free-surface flows. *International Journal for Numerical Methods in Fluids*, 30:425–440, June 1999.

V. Casulli and G.S. Stelling. Numerical simulation of 3D quasi-hydrostatic, free-surface flows. *Journal of Hydraulic Engineering*, 124(7):678–686, 1998.

V. Casulli and P. Zanolli. Semi-implicit numerical modeling of nonhydrostatic free-surface flows for environmental problems. *Mathematical and Computer Modelling*, 36(9-10):1131–1149, December 2002.

Delft3D-FLOW User Manual. Deltares, November 2006. Version 3.13.

M.W. Dingemans. *Water wave propagation over uneven bottoms*. PhD thesis, TU Delft, 1994.

O.B. Fringer, M. Gerritsen, and R.L. Street. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. *Ocean Modelling*, 14(3-4):139–173, 2006.

W. Hackbusch. *Multi-grid methods and applications*. Springer, 1985.

M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur Stand.*, 49(6):409–436, 12 1952.

M.B. Koyigit, R.A. Falconer, and B. Lin. Three-dimensional numerical modelling of free surface flows with non-hydrostatic pressure. *International Journal for Numerical Methods in Fluids*, 40(9):1145–1162, 2002.

A. Mahadevan, J. Oliger, and R. Street. A nonhydrostatic mesoscale ocean model. part i: Well-posedness and scaling. *Journal of Physical Oceanography*, 26(9):1868–1880, September 1996a.

A. Mahadevan, J. Oliger, and R. Street. A nonhydrostatic mesoscale ocean model. part ii: Numerical implementation. *Journal of Physical Oceanography*, 26(9):1881–1900, September 1996b.

K.W. Morton and D.F. Mayers. *Numerical solution of partial Differential Equations*. Cambridge University Press, Cambridge, 1994.

N.M. Nachtigal, S.C. Reddy, and L.N. Trefethen. How fast are nonsymmetric matrix iterations. *SIAM J. Matrix Anal. Appl.*, 13(3):778–795, 1992.

Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2 edition, 2003.

Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3): 856–869, 1986.

G. Stelling and M. Zijlema. An accurate and efficient finite-difference algorithm for non-hydrostatic free-surface flow with application to wave propagation. *International Journal for Numerical Methods in Fluids*, 43(1):1–23, 2003.

H.A. van der Vorst. BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.

C.B. Vreugdenhil. *Numerical methods for shallow-water flow*. Kluwer, Dordrecht, 1994.

P. Wesseling. *An introduction to multigrid methods*. Wiley, Chichester, 1992.

P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer, 2001.

M. Zijlema and G.S. Stelling. Further experiences with computing non-hydrostatic free-surface flows involving water waves. *International Journal for Numerical Methods in Fluids*, 48(2): 169–197, 2005.

M. Zijlema and G.S. Stelling. Efficient computation of surf zone waves using the nonlinear shallow water equations with non-hydrostatic pressure. *Coastal Engineering*, In Press, 2008.