Numerical methods for the three-dimensional shallow water equations on supercomputers

E.D. de Goede

1991 Mathematics Subject Classification: 65M06, 65M12, 65M20.
ISBN 90 6196 417 2
NUGI-code: 811

Preface

The subject of this tract is the development of an accurate and efficient numerical method for the three-dimensional shallow water equations. In order to develop such a method, we started in 1988 the VECPARCOMP-project at the CWI (Centre for Mathematics and Computer Science). This research project was supported by Rijkswaterstaat (Dutch Water Control and Public Works Department) and was carried out in co-operation with the Tidal Waters Division of Rijkswaterstaat, Delft Hydraulics and ICIM (Informatics Centre for Civil Engineering and Environment). The project was supervised by a committee with the following members:

Prof. dr. ir. A.W. Heemink (Rijkswaterstaat, University of Delft)
Prof. dr. P.J. van der Houwen (CWI, University of Amsterdam)
Prof. dr. ir. G.S. Stelling (Delft Hydraulics, University of Delft)
Dr. ir. Th.L. van Stijn (ICIM)
Dr. ir. F.W. Wubs (University of Groningen).

I want to thank all those who contributed in some way to the realization of this tract. I like to mention some of them explicitly. In particular, I thank Prof. dr. P.J. van der Houwen for his guidance and for the stimulating discussions during this project. I am grateful to Dr. ir. Th.L. van Stijn and Prof. dr. ir. A.W. Heemink for their useful advices and for acquainting me with the Rijkswaterstaat. Further, the constructive remarks and the careful reading of the papers by Dr. B.P. Sommeijer have significantly contributed to the successful completion of this research project. Prof. dr. ir. G.S. Stelling and Dr. ir. F.W. Wubs are also thanked. Their assistance and insight in shallow water models has been very valuable.

I like to express my great appreciation to Joke Blom, Walter Lioen, Margreet Louter-Nool, Herman te Riele and Dik Winter for their support during the many numerical experiments. Owing to the rapid changes in hardware and software, their help was indispensable.

Finally, I want to thank the Stichting Mathematisch Centrum for publishing this research as a CWI Tract.

Ilpendam, October 1992 E.D. de Goede

Contents

Notation

C	Chezy coefficient	$(m^{1/2}s^{-1})$
d	undisturbed depth of water	(m)
f	Coriolis coefficient	(s^{-1})
g	acceleration due to gravity	(ms^{-2})
h	total depth $(= d + \zeta)$	(m)
nx, ny	number of grid points in the x- and y-direction, respectively	
ns	number of grid points in the vertical direction	
p	pressure	$(kgm^{-1}s^{-2})$
p_a	atmospheric pressure	$(kgm^{-1}s^{-2})$
R	radius of the Earth	(m)
q	number of smoothing factors	
t	time	(s)
u,v	velocity components in the x- and y-direction, respectively	(ms^{-1})
u_d,v_d	velocity components at some depth near the bottom	(ms^{-1})
w	vertical velocity component in the x-y-z co-ordinate system	(ms^{-1})
W_f	wind stress	$(kgm^{-1}s^{-2})$
x,y	horizontal spatial co-ordinates	(m)
Δx, Δy	mesh sizes in the x- and y-direction, respectively	(m)
z	vertical spatial co-ordinate in the x-y-z co-ordinate system	(m)
γ	relaxation parameter for the SCG method	
λ	eddy viscosity coefficient in the horizontal direction	(m^2s^{-1})
μ	eddy viscosity coefficient in the σ-direction	(m^2s^{-1})
ρ	water density	(kgm^{-3})
σ	vertical spatial co-ordinate in the sigma-transformed system	
$\Delta\sigma$	mesh size in the vertical direction	
τ	time step	(s)
τ_{xx}, τ_{xy},...	components of the stress tensor	$(kgm^{-1}s^{-2})$
ϕ	angle between wind direction and the positive x-axis	(°)
ω	vertical velocity component in the x-y-σ co-ordinate system	(s^{-1})
ω_e	angular speed of the Earth's rotation	(s^{-1})
χ, φ	polar co-ordinates in east longitude and north latitude, respectively	(°)
$\Delta\chi$, $\Delta\varphi$	mesh sizes in the polar co-ordinate system	(°)
ζ	water elevation above undisturbed depth	(m)

Chapter 1

Introduction

1.1. SHALLOW WATER MODELS

The shallow water equations describe a mathematical model for flows in which the length of the free surface waves is significantly larger than the water depth. Examples are flows in rivers, estuaries and shallow seas. From a mathematical point of view, these hydrodynamic models are complex, because they involve effects of e.g., the wind, the earth's rotation and the geometry of the water system. *Numerical models* have become established for predicting water flows. By the advances in numerical mathematics and in computer performance, simulations of water flows can be performed accurately. Nowadays, numerical models are much cheaper and flexible than *scale models*, which have frequently been used in the past.

It is more than 60 years ago that numerical models were introduced for the simulation of water flows. For the purpose of predicting the effect of the closure of the Zuiderzee in the Netherlands, numerical computations were performed in 1926 by the Dutch physicist Lorentz [13]. It appeared that the tidal elevations agreed well with the numerical predictions. A more recent application was made for the storm surge barrier in the mouth of the Oosterschelde (Eastern Scheldt), which is in the south-western part of the Netherlands. A numerical tidal model was developed for the accurate prediction of the water level to ensure that the barrier will be closed in time in case of extremely high water.

In the past, many numerical methods were developed for the two-dimensional shallow water equations. In the Netherlands well-known methods for these (depth-averaged) equations are the ADI-method of Leendertse [11], the ADI-method of Stelling [16], the finite element method of Praagman [15] and the stabilized Runge-Kutta method of Wubs [19]. The main goal of two-dimensional models is the accurate prediction of water levels. Presently, a two-dimensional model of the Continental Shelf is operational at KNMI (Royal Dutch Meteorological Institute). Using wind and atmospheric pressure data from a numerical model of the atmosphere, the water elevations in the North Sea and especially along the Dutch coast are computed four times a day [10].

In the last decennium computing power has increased significantly. As a consequence, more physics could be included in numerical models. There has been a major research effort in developing three-dimensional models, which yield information about the vertical structure of the water. The ability to accurately predict these vertical structures is particularly important in a wide range of pollution problems. With two-dimensional models such information can not be obtained.

The application of three-dimensional models requires a great computational effort, especially when a high resolution is needed. Therefore, it is necessary to construct methods that are able to fully exploit the facilities of fast computers, such as vector and parallel computers. So far, the numerical methods used in the Netherlands for the three-dimensional shallow water equations (see e.g., [12] and [17]), were not

developed with vector and parallel computers in mind. In order to develop a computationally efficient three-dimensional shallow water model on such computers, the VECPARCOMP project was started four years ago. This project is a co-operation between the Rijkswaterstaat (Dutch Water Control and Public Works department) and the CWI (Centre for Mathematics and Computer Science). In this tract the results of the VECPARCOMP project are presented.

This tract deals with the development of numerical methods for the three-dimensional shallow water equations on vector and parallel computers. The three-dimensional shallow water equations and their derivation will be discussed in Chapter 2. The subsequent chapters are devoted to the numerical discretization of these equations by means of the method of lines approach. This approach first discretizes in space, followed by the discretization in time.

1.2. SPACE DISCRETIZATION

For the space discretization we use the staggered grid that is known as the Arakawa C-grid [1]. This is the most commonly used and the most successfully used grid for shallow water models. In the vertical direction the shallow water equations were transformed into (depth-following) sigma co-ordinates to obtain the same vertical resolution in the whole water system [14].

On this staggered grid the spatial derivatives were replaced by finite differences. It is well-known that finite differences can be implemented efficiently on vector and parallel computers. We examined various discretizations for the advective terms. It appears that the finite differences developed by Stelling [16] perform best. Both the special discretization near the boundaries and the introduction of some dissipation by the upwind discretization of the mixed advective terms, which are described in [16], turns out to be essential (see Chapter 7). The discretization of the other terms will be discussed in Chapter 3.

1.3. TIME INTEGRATION

At CWI time integration methods for two-dimensional shallow water models were developed by Wubs in 1983-1987 [19]. Our project may be considered as a follow-up of this research. In three-dimensional models there is a multi-layer approach in the vertical direction, instead of one (depth-averaged) layer in the two-dimensional case. As a first introduction to three-dimensional models we investigated the influence of the vertical diffusion term. For a model without advective terms, we examined time integrators that were explicit, semi-implicit or implicit in the vertical. Chapter 3 is devoted to this topic.

It appears that the vertically implicit methods of Chapter 3 perform best. However, for these methods we are still faced with a CFL condition that depends on the water depth and on the horizontal mesh sizes. This implies that for small values of the horizontal mesh sizes or for very deep water, this time step restriction is more severe than necessary for accuracy considerations. In order to increase the stability we applied so-called right-hand side smoothing. Right-hand side smoothing has originally been developed by Wubs [19]. By this technique, the computation time for our three-dimensional shallow water models reduces considerably while the accuracy remains acceptable (see Chapter 4).

Next, we constructed a two-stage time splitting method that is unconditionally stable (see Chapter 5). For two-dimensional shallow water models, this method is

very similar to the one described in [18], where its feasibility for practical computations has been shown. It appears that the efficiency of our method is even higher for three-dimensional models than for two-dimensional ones.

For the model without advective terms, we compared the unconditionally stable method with the vertically implicit method stabilized by right-hand side smoothing of Chapter 4. The results will be presented in Chapter 6. The unconditionally stable method is the most accurate one. This method is also more efficient, because large time steps can be used.

So far, the advective terms were omitted. We incorporated the advective terms in the aforementioned unconditionally stable method, which will be described in Chapter 7. The discretizations developed by Stelling [16] are applied. The introduction of the advective terms results in a hardly more complicated system of equations.

1.4. Implementation on vector and parallel computers

As mentioned earlier, the application of three-dimensional shallow water models requires the use of fast computers, such as vector and parallel computers. At the end of 1988 an Alliant FX/4 was installed at CWI. The Alliant FX/4 was used to investigate parallel methods for our shallow water equations. The numerical experiments described in Chapters 3-8 were carried out on this mini-supercomputer. Both the vector and the parallel optimization of the Alliant FX/4 were utilized.

During our four-year project, we investigated test problems of an increasing degree of complexity. The most realistic experiments were carried out on CRAY supercomputers. A river problem in which a jetty was situated, was simulated on a CRAY Y-MP4/464 (see Chapter 8). Since December 1990 this supercomputer is operational at the Academic Computing Services Amsterdam (SARA). Chapter 9 deals with the implementation of a northwest European Continental Shelf model on the CRAY Y-MP2E installed at ICIM (Informatics Centre for Civil Engineering and Environment). The CRAY Y-MP2E was recently installed in the Netherlands for the simulation of large scale models of rivers and seas.

In this tract, Chapters 3-9 are based on papers that have been published or have been accepted for publication. In order to obtain a uniform notation, the chapters slightly differ from the papers. Chapter 3 contains parts of the papers [2] and [3]. Chapters 4-8 correspond with the papers in [4, 5, 6, 7 and 8], respectively. The paper described in Chapter 9 has been accepted for publication [9].

References

1. A. Arakawa and V.R. Lamb, Computational design of the basic dynamical processes of the UCLA general circulation model, *Meth. Comp. Phys.*, 16, 173-263 (1977).
2. E.D. de Goede, Finite difference methods for the 3D hydrodynamical equations, *Proceedings of the 1st Int. Conf. on Applications of supercomputers in engineering*, Southampton, 133-144 (1989).
3. E.D. de Goede, A computational model for the three-dimensional shallow water flows on the Alliant FX/4, *Supercomputer*, 32, 43-49 (1988).

4. E.D. DE GOEDE, Stabilization of a time integrator for the 3D shallow water equations by smoothing techniques, *Int. J. Numer. Meth. in Fluids*, 12, 475-490 (1991).
5. E.D. DE GOEDE, A time splitting method for the three-dimensional shallow water equations, *Int. J. Numer. Meth. in Fluids*, 13, 519-534 (1991).
6. E.D. DE GOEDE, Numerical methods for the 3D shallow water equations on vector and parallel computers, *Appl. Numer. Math.*,5, 3-18 (1992).
7. E.D. DE GOEDE, On the numerical treatment of the advective terms in 3D shallow water models, *Proceedings of the 2nd Symposium on High Performance Computing*, Montpellier, 491-502 (1991).
8. E.D. DE GOEDE, 3D shallow water model on the CRAY Y-MP4/464, *Proceedings of the 6th Int. Workshop on the Use of Supercomputers in Theoretical Science*, Antwerp, 107-114 (1991).
9. E.D. DE GOEDE, A numerical model of the northwest European Continental Shelf on the CRAY Y-MP2E, accepted for publication in *IMPACT of Computing in Science and Engineering*.
10. A.W. HEEMINK, A. LANGERAK, Th.L. VAN STIJN, L.P.M. DE VREES AND J.W. DE VRIES, The new Dutch storm surge forecasting system, *Proceedings of the Workshop STORM '91*, Hamburg, 1991.
11. J.J. LEENDERTSE, *Aspects of a computational model for long period water wave propagation*, Memorandum RM-5294-PR, Rand Corp., Santa Monica, California, 1967.
12. J.J. LEENDERTSE, *A new approach to three-dimensional free-surface flow modelling*, Memorandum R-3712-NETH/RC, Rand Corp., Santa Monica, California, 1989.
13. H.A. LORENTZ, *Verslag Staatscommissie Zuiderzee 1918-1926 (Report of the Government Zuiderzee Commission)*, Alg. Landsdrukkerij, The Hague, 1926 (Dutch).
14. N.A. PHILIPS, A coordinate system having some special advantages for numerical forecasting, *J. Meteorol.*, 14, 184-194 (1957).
15. N. PRAAGMAN, *Numerical solution of the shallow water equations by a finite element method*, Ph.D. Thesis, Delft University, 1979.
16. G.S. STELLING, *On the construction of computational methods for shallow water flow problems*, Ph.D. Thesis, Delft University, 1983.
17. TRISULA, A multi-dimensional flow and water-quality simulation system, Delft Hydraulics, The Netherlands, 1989.
18. P. WILDERS, Th.L.VAN STIJN, G.S. STELLING AND G.A. FOKKEMA, A fully implicit splitting method for accurate tidal computations, *Int. J. Numer. Meth. in Eng.*, 26, 2707-2721 (1988).
19. F.W. WUBS, *Numerical solution of the shallow-water equations*, Ph.D. Thesis, University of Amsterdam, 1987.

Chapter 2

The Three-Dimensional Shallow Water Equations

2.1. Mathematical model in Cartesian co-ordinates

In this chapter we will derive a mathematical model for three-dimensional shallow water flows. We will only consider homogeneous flows. The mathematical description of homogeneous water flows consists of a system of differential equations that are physically based on the conservation laws for mass and momentum. These equations are a simplification of the well-known Navier Stokes equations.

The time-dependent, incompressible Navier Stokes equations may be written in Cartesian co-ordinates as (see [2,5])

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - w\frac{\partial u}{\partial z} + f_x + \frac{1}{\rho}\left\{ -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} \right\} \tag{1.1}$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - w\frac{\partial v}{\partial z} - f_y + \frac{1}{\rho}\left\{ -\frac{\partial p}{\partial y} + \frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} \right\} \tag{1.2}$$

$$\frac{\partial w}{\partial t} = -u\frac{\partial w}{\partial x} - v\frac{\partial w}{\partial y} - w\frac{\partial w}{\partial z} - f_z + \frac{1}{\rho}\left\{ -\frac{\partial p}{\partial z} + \frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{zy}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} \right\} - g \tag{1.3}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0\,, \tag{1.4}$$

where the Coriolis term is defined by

$$\begin{aligned} f_x &= 2\,(\,\Omega_3 v - \Omega_2 w\,) \\ f_y &= 2\,(\,\Omega_1 w - \Omega_3 u\,) \\ f_z &= 2\,(\,\Omega_2 u - \Omega_1 v\,)\,, \end{aligned} \tag{1.5}$$

with $\Omega = (\Omega_1, \Omega_2, \Omega_3)$ denoting the earth rotation vector. For latitudes that are not too close to the equator we may simplify equation (1.5) to [3]

$$\begin{aligned} f_x &= 2\,\Omega_3 v \\ f_y &= -2\,\Omega_3 u \\ f_z &= 2\,(\,\Omega_2 u - \Omega_1 v\,)\,. \end{aligned} \tag{1.5'}$$

The equations (1.1)-(1.3) are the equations of motion and (1.4) is the continuity equation. In system (1.1)-(1.4) the curvature effects of the earth are neglected. The earth's rotation is modelled by the Coriolis term.

In shallow water flows the fluid motions are predominantly horizontal. The vertical acceleration of the large scale motion is very small, particularly if compared with the acceleration due to gravity. Therefore, neglecting the vertical acceleration and advection is justified [2]. For the same reason the Coriolis term and the components of the stress tensor may be neglected in (1.3). Then, equation (1.3) reduces to the hydrostatic equation

$$\frac{\partial p}{\partial z} = -\rho g \,. \tag{1.6}$$

The internal pressure distribution in the flow can be derived by vertical integration of equation (1.6), which leads to

$$p = p_a + \rho\, g(\zeta - z) \,.$$

The variations of the atmospheric pressure p_a are small compared to the variations of $\rho g(\zeta - z)$ and are neglected. This yields

$$\frac{\partial p}{\partial x} = \rho\, g \frac{\partial \zeta}{\partial x} \quad \text{and} \quad \frac{\partial p}{\partial y} = \rho\, g \frac{\partial \zeta}{\partial y} \,. \tag{1.7}$$

The components of the stress tensor in (1.1) and (1.2) may be expressed as gradients of Reynolds stresses [2], through the relationships

$$\begin{aligned}
&\tau_{xx} = \rho\lambda \frac{\partial u}{\partial x}, \quad \tau_{xy} = \rho\lambda \frac{\partial u}{\partial y}, \quad \tau_{xz} = \rho\mu \frac{\partial u}{\partial z}, \\
&\tau_{yx} = \rho\lambda \frac{\partial v}{\partial x}, \quad \tau_{yy} = \rho\lambda \frac{\partial v}{\partial y}, \quad \tau_{yz} = \rho\mu \frac{\partial v}{\partial z} \\
&\tau_{zx} = \rho\lambda \frac{\partial w}{\partial x}, \quad \tau_{zy} = \rho\lambda \frac{\partial w}{\partial y}, \quad \tau_{zz} = \rho\mu \frac{\partial w}{\partial z} \,.
\end{aligned} \tag{1.8}$$

By the requirement that at the moving water surface a particle must follow the motion of that surface, we obtain for $z=\zeta(x,y,t)$

$$w = \frac{d\zeta}{dt} = u \frac{\partial \zeta}{\partial x} + v \frac{\partial \zeta}{\partial y} + \frac{\partial \zeta}{\partial t} \,. \tag{1.9}$$

Similarly, at the bottom $z=-d(x,y)$ the boundary condition reads

$$w = u \frac{\partial(-d)}{\partial x} + v \frac{\partial(-d)}{\partial y} \,. \tag{1.10}$$

Integrating the continuity equation (1.4) from the bottom to the surface, yields

$$w(x,y,\zeta,t) - w(x,y,-d,t) = -\int_{-d}^{\zeta} \frac{\partial u}{\partial x} dz - \int_{-d}^{\zeta} \frac{\partial v}{\partial y} dz . \qquad (1.11)$$

Then, applying Leibniz' rule and combination of (1.11) with (1.9) and (1.10), leads to

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(\int_{-d}^{\zeta} u dz \right) - \frac{\partial}{\partial y}\left(\int_{-d}^{\zeta} v dz \right) . \qquad (1.12)$$

The change of the water elevation ζ is related to the vertically integrated flow. Note that equation (1.12) also occurs in two-dimensional shallow water models.

By integrating from the bottom to a certain level $z=h_1$, we obtain the following relationship for the vertical velocity w:

$$w(x,y,h_1,t) = -\frac{\partial}{\partial x}\left(\int_{-d}^{h_1} u dz \right) - \frac{\partial}{\partial y}\left(\int_{-d}^{h_1} v dz \right) + u \frac{\partial (h_1)}{\partial x} + v \frac{\partial (h_1)}{\partial y} . \qquad (1.13)$$

This expression for w allows a non-zero vertical velocity at the bottom (see (1.10)).

Using the relations (1.1), (1.2), (1.5'), (1.7), (1.8) and (1.12) the three-dimensional shallow water equations equations in Cartesian co-ordinates read

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - w\frac{\partial u}{\partial z} + fv - g\frac{\partial \zeta}{\partial x} + \lambda \frac{\partial^2 u}{\partial x^2} + \lambda \frac{\partial^2 u}{\partial y^2} + \frac{\partial}{\partial z}\left(\mu \frac{\partial u}{\partial z} \right) \qquad (1.14)$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - w\frac{\partial v}{\partial z} - fu - g\frac{\partial \zeta}{\partial y} + \lambda \frac{\partial^2 v}{\partial x^2} + \lambda \frac{\partial^2 v}{\partial y^2} + \frac{\partial}{\partial z}\left(\mu \frac{\partial v}{\partial z} \right) \qquad (1.15)$$

$$w = -\frac{\partial}{\partial x}\left(\int_{-d}^{h_1} u dz \right) - \frac{\partial}{\partial y}\left(\int_{-d}^{h_1} v dz \right) \qquad (1.16)$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(\int_{-d}^{\zeta} u dz \right) - \frac{\partial}{\partial y}\left(\int_{-d}^{\zeta} v dz \right) \qquad (1.17)$$

with $-d \le h_1 \le \zeta$.

In (1.14)-(1.15) we have used the oceanographic notation

$$f = 2\,|\Omega|\,\sin(\varphi) ,$$

where φ denotes the earth's latitude.

The boundary conditions have not been specified yet. At the sea surface $z = \zeta$ the boundary conditions are given by

$$\left(\mu \frac{\partial u}{\partial z}\right)_{z=\zeta} = \frac{1}{\rho} W_f \cos(\phi), \qquad \left(\mu \frac{\partial v}{\partial z}\right)_{z=\zeta} = \frac{1}{\rho} W_f \sin(\phi). \tag{1.18}$$

Similarly, at the boundary conditions at the bottom z = –d(x,y) we prescribe

$$\left(\mu \frac{\partial u}{\partial z}\right)_{z=-d} = \frac{k\, g\, u_d}{C^2}, \qquad \left(\mu \frac{\partial v}{\partial z}\right)_{z=-d} = \frac{k\, g\, v_d}{C^2}, \tag{1.19}$$

with k an appropriate coefficient of bottom friction. Equation (1.19) represents a linear law of bottom friction. An alternative to (1.19) is the quadratic law of bottom friction, of the form

$$\left(\mu \frac{\partial u}{\partial z}\right)_{z=-d} = \frac{g\, u_d}{C^2}\sqrt{u_d^2 + v_d^2}, \quad \left(\mu \frac{\partial v}{\partial z}\right)_{z=-d} = \frac{g\, v_d}{C^2}\sqrt{u_d^2 + v_d^2}. \tag{1.20}$$

2.2. MATHEMATICAL MODEL IN SIGMA CO-ORDINATES

In the vertical direction the domain is bounded by the bottom topography and the *time-dependent* water elevation ζ. To ensure that the three-dimensional domain is constant in time, the equations have to be transformed in the vertical direction into depth-following (sigma) co-ordinates. Transforming equations (1.14)-(1.17) from the interval $-d \le z \le \zeta$ into the constant interval $1 \ge \sigma \ge 0$, by the so-called sigma transformation [4]

$$\sigma = \frac{\zeta - z}{d + \zeta},$$

leads to

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \omega\frac{\partial u}{\partial \sigma} + fv - g\frac{\partial \zeta}{\partial x} + \lambda\frac{\partial^2 u}{\partial x^2} + \lambda\frac{\partial^2 u}{\partial y^2} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial u}{\partial \sigma}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - \omega\frac{\partial v}{\partial \sigma} - fu - g\frac{\partial \zeta}{\partial y} + \lambda\frac{\partial^2 v}{\partial x^2} + \lambda\frac{\partial^2 v}{\partial y^2} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial v}{\partial \sigma}\right) \tag{2.2}$$

$$\omega = \frac{1}{h}\left\{-(1-\sigma)\left(\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) + \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right)\right) + \frac{\partial}{\partial x}\left(h\int_\sigma^1 u\,d\sigma\right) + \frac{\partial}{\partial y}\left(h\int_\sigma^1 v\,d\sigma\right)\right\} \tag{2.3}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) - \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right). \tag{2.4}$$

The relation between the new vertical velocity ω and the untransformed (physical) velocity w is given by [1,6]

$$w = -\omega h + \frac{\partial \zeta}{\partial t} - \sigma \frac{\partial h}{\partial t} + u \left(\frac{\partial \zeta}{\partial x} - \sigma \frac{\partial h}{\partial x} \right) + v \left(\frac{\partial \zeta}{\partial y} - \sigma \frac{\partial h}{\partial y} \right) .$$

Transformation of the surface and sea-bed boundary conditions into sigma co-ordinates, yields at the sea surface

$$\left(\mu \frac{\partial u}{\partial \sigma} \right)_{\sigma=0} = -\frac{h}{\rho} W_f \cos(\phi) \quad , \quad \left(\mu \frac{\partial v}{\partial \sigma} \right)_{\sigma=0} = -\frac{h}{\rho} W_f \sin(\phi) . \tag{2.5}$$

At the bottom z = –d(x,y) the quadratic law of bottom friction (cf. (1.20)) leads to

$$\left(\mu \frac{\partial u}{\partial \sigma} \right)_{\sigma=1} = -h \frac{g\, u_d}{C^2} \sqrt{u_d^2 + v_d^2} \quad , \quad \left(\mu \frac{\partial v}{\partial \sigma} \right)_{\sigma=1} = -h \frac{g\, v_d}{C^2} \sqrt{u_d^2 + v_d^2} . \tag{2.6}$$

Furthermore, for the transformed vertical velocity ω we have

$$\omega(x,y,0,t) = 0 \quad \text{and} \quad \omega(x,y,1,t) = 0 .$$

System (2.1)-(2.4) together with its boundary conditions (2.5)-(2.6) will be the starting point for our mathematical shallow water model. In the following chapters we will sometimes use a simplified model. For example, in several chapters the advective terms will be omitted. In Chapter 9 a complete model in polar co-ordinates will be used.

For a detailed description of three-dimensional shallow water equations we refer to [2,5].

REFERENCES

1. A.M. DAVIES, Application of the DuFort-Frankel and Saul'ev methods with time splitting to the formulation of a three-dimensional hydrodynamic sea model, *Int. J. Numer. Meth. in Fluids*, 5, 405-425 (1985).
2. G.J.H. LINDIJER, *Three-dimensional circulation models for shallow lakes and seas*, Delft Hydraulics Report R 900 I, 1976.
3. H. GERRITSEN, *Accurate boundary treatment in shallow water flow computations*, Ph.D. Thesis, Twente University, 1982.
4. N.A. PHILLIPS, A coordinate system having some special advantages for numerical forecasting, *J. Meteorol.*, 14, 184-194 (1957).
5. L.C. VAN RIJN, *Principles of fluid flow and surface waves in rivers, estuaries and seas*, Aqua Publications, Amsterdam, 1990.
6. TRISULA, A multi-dimensional flow and water-quality simulation system, Delft Hydraulics, The Netherlands, 1989.

Chapter 3

Explicit and Semi-Implicit Methods for the Three-Dimensional Shallow Water Equations

E.D. de Goede
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB Amsterdam, The Netherlands

For a linear three-dimensional hydrodynamic sea model the stability and efficiency on vector and parallel computers of various time integrators is compared for a wind induced flow in a rectangular basin. Owing to stability, it appears to be necessary to treat the vertical terms in an implicit way. We show that the so-called vertically implicit methods can be computed efficiently on vector and parallel computers.

3.1. INTRODUCTION

In this paper one-step time integrators for the three-dimensional hydrodynamic equations are developed and compared with each other with respect to stability and efficiency on vector and parallel computers. Section 3.2 provides the simplified hydrodynamic equations in depth-following (sigma) co-ordinates. These equations describe the motion and the elevation of water.

For the numerical discretization of the shallow water equations we follow the method of lines approach. This approach first converts the system of partial differential equations (PDEs) into a system of ordinary differential equations (ODEs) by discretization of the space derivatives. We use second-order finite differences (see Section 3.3). Then, in Section 3.4 various time integrators are developed for this system of ODEs. Application of time integrators for a three-dimensional model requires a great computational effort. Especially for fully implicit methods, this is a severe disadvantage. If an explicit method is used, then besides the CFL stability condition there is also a condition imposed by the vertical diffusion term [1]. In many problems the last condition is more restrictive. To investigate the influence of this stability condition, we examine time integrators that are explicit, semi-implicit or implicit in the vertical direction.

In the numerical experiments, which are described in Section 3.6, a wind induced test model is examined. This model has been used by others [1,3] and is therefore an ideal case for comparing the results. It appears that the time integrators in which the vertical diffusion is treated implicitly perform best in our experiments. Section 3.7 deals with a stability analysis for one of these so-called vertically implicit methods. For this time integrator the stability condition does not depend on the vertical mesh size. Thus, the time step is only limited in terms of the horizontal mesh sizes.

The vertically implicit methods require the solution of a large number of tridiagonal systems. In Section 3.5 we will investigate two algorithms for the solution of these systems. The numerical results will be shown on a vector computer (viz., a 2-pipe CDC CYBER 205) and on a vector-parallel computer (viz., an Alliant FX/4).

3.2. MATHEMATICAL MODEL

In this section a simplified three-dimensional hydrodynamic sea model will be presented. For a detailed description of the hydrodynamic sea model we refer to [2]. The simplified three-dimensional hydrodynamic equations, of continuity and motion, may be written in sigma co-ordinates as [1,2]

$$\frac{\partial u}{\partial t} = fv - g\frac{\partial \zeta}{\partial x} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial u}{\partial \sigma}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -fu - g\frac{\partial \zeta}{\partial y} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial v}{\partial \sigma}\right) \tag{2.2}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) - \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right). \tag{2.3}$$

To ensure that the domain in which the equations (2.1)-(2.3) are solved, is constant in time, the equations have been transformed in the vertical direction into depth-following (sigma) co-ordinates by the so-called sigma transformation [6]

$$\sigma = \frac{\zeta - z}{d + \zeta}.$$

The boundary conditions at the sea surface ($\sigma = 0$) are given by

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \cos(\phi), \quad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \sin(\phi). \tag{2.4}$$

Similarly, the boundary conditions at the bottom ($\sigma = 1$) read

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=1} = \frac{g\,u_d}{C^2}, \qquad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=1} = \frac{g\,v_d}{C^2}. \tag{2.5}$$

3.3. SPACE DISCRETIZATION

Following the method of lines approach, we first replace the spatial operators in (2.1)-(2.3) by finite differences. For the finite differences there are essentially two approaches in the vertical. In a model with Cartesian co-ordinates (see e.g., in [5]), a fixed grid is used in the vertical, through which the fluid is free to move. This can be visualized by considering the fluid in horizontal slices, in which only the upper

layer has a time-variable height. Since this model is fixed in the vertical, the number of grid layers increases as the depth increases, but reduces in shallow regions. This problem of reduced vertical resolution in the shallow regions can be overcome by using the depth-following (sigma) co-ordinates of the previous section. Then, a constant number of grid layers is used in the vertical at each horizontal grid point. Moreover, there are no 'zig-zag boundaries' in the vertical in the case of an irregular bottom. From a computational point of view, it is also advantageous to have a constant number of grid layers, especially on vector and parallel computers. Therefore, we have chosen the sigma co-ordinate model (2.1)-(2.3).

The computational domain is covered by an nx·ny·ns rectangular grid. The notation used for the velocities is $U_{i,j,k}$ and $V_{i,j,k}$, where i,j refers to the horizontal grid point and k to the vertical layer. The surface elevation points are denoted by $Z_{i,j}$ and are computed at the sea surface only. The vertical diffusion coefficient is assumed to vary only through the vertical. Hence, μ_k denotes the coefficient at layer k. In both the horizontal and vertical direction a staggered grid is used. Figure 1 shows the horizontal grid spacing. In the vertical a varying mesh of thickness $\Delta\sigma_k$, where k refers to the k-th grid layer from the surface, is used. Hence, it is possible to increase the resolution near the surface and the bottom. For the structure of the vertical grid spacing we refer to [2].

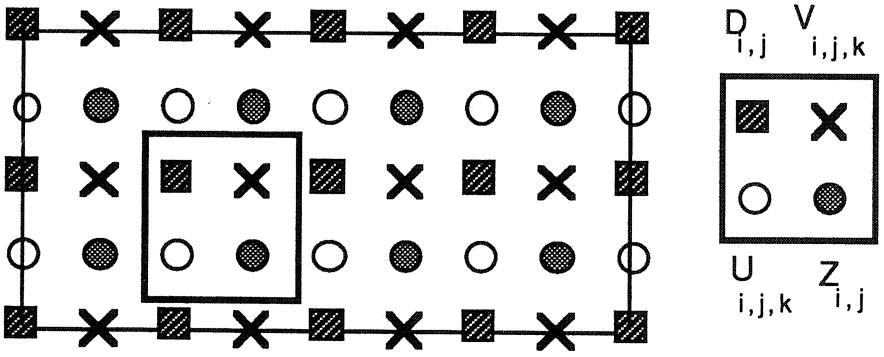

Figure 1. The staggered grid in the (x,y)-plane.

The use of a staggered grid has the following advantages:

a) For the system of equations (2.1)-(2.3) the storage requirements decrease with a factor 4 (the mesh sizes of the staggered grid in Figure 1 are twice the mesh sizes of the unstaggered grid). However, components that are not available in a particular grid point, have to be obtained by averaging.
b) It simplifies the boundary conditions (e.g., in U-boundary points no conditions for the V-velocity have to be prescribed).
c) It reduces the possibility of spurious "2Δx-waves" [7].

For the approximation of the spatial derivatives, second-order central differences are used in both the horizontal and vertical direction. The horizontal mesh sizes are denoted by Δx and Δy. For the equations of motion (2.1) and (2.2), we now obtain

$$\frac{\partial U_{i,j,k}}{\partial t} = f\,\tilde{V}_{i,j,k} - g\left(\frac{Z_{i,j} - Z_{i-1,j}}{\Delta x}\right) + \frac{1}{\bar{H}_{i,j}^2}\frac{1}{\Delta\sigma_k}\left\{\frac{\mu_{k+1}(U_{i,j,k+1} - U_{i,j,k})}{0.5\,(\Delta\sigma_{k+1} + \Delta\sigma_k)} - \frac{\mu_k(U_{i,j,k} - U_{i,j,k-1})}{0.5\,(\Delta\sigma_k + \Delta\sigma_{k-1})}\right\} \quad (3.1)$$

$$\frac{\partial V_{i,j,k}}{\partial t} = -f\,\tilde{U}_{i,j,k} - g\left(\frac{Z_{i,j+1} - Z_{i,j}}{\Delta y}\right) + \frac{1}{\tilde{H}_{i,j}^2}\frac{1}{\Delta\sigma_k}\left\{\frac{\mu_{k+1}(V_{i,j,k+1} - V_{i,j,k})}{0.5\,(\Delta\sigma_{k+1} + \Delta\sigma_k)} - \frac{\mu_k(V_{i,j,k} - V_{i,j,k-1})}{0.5\,(\Delta\sigma_k + \Delta\sigma_{k-1})}\right\}, \quad (3.2)$$

where

$$\tilde{U}_{i,j,k} = 0.25\cdot\left(U_{i,j,k} + U_{i+1,j,k} + U_{i,j+1,k} + U_{i+1,j+1,k}\right),$$

$$\tilde{V}_{i,j,k} = 0.25\cdot\left(V_{i,j,k} + V_{i,j-1,k} + V_{i-1,j,k} + V_{i-1,j-1,k}\right),$$

$$\bar{H}_{i,j} = Z_{i,j} + \frac{1}{2}\left(D_{i,j} + D_{i,j-1}\right) \quad \text{and} \quad \tilde{H}_{i,j} = Z_{i,j} + \frac{1}{2}\left(D_{i,j} + D_{i+1,j}\right).$$

Considering equation (2.3), we have to approximate an integral which ranges from the bottom to the surface. The vertical direction has been divided into a number of grid layers. Let s_k denote the interfaces between the layers, defined by

$$s_k = \sum_{q=1}^{k} \Delta\sigma_q, \quad k=1,\ldots,ns\ .$$

Then, for the integral in equation (2.3) we may write

$$\int_0^1 u(i\Delta x, j\Delta y, \sigma)\,d\sigma = \sum_{k=1}^{ns} \int_{s_{k-1}}^{s_k} u(i\Delta x, j\Delta y, \sigma)\,d\sigma \approx \sum_{k=1}^{ns} (s_k - s_{k-1})U_{i,j,k} = \sum_{k=1}^{ns} \Delta\sigma_k U_{i,j,k},$$

which leads to

$$\frac{\partial Z_{i,j}}{\partial t} = -\frac{1}{\Delta x}\left\{\bar{H}_{i+1,j}\sum_{k=1}^{ns}\Delta\sigma_k U_{i+1,j,k} - \bar{H}_{i,j}\sum_{k=1}^{ns}\Delta\sigma_k U_{i,j,k}\right\} - \frac{1}{\Delta y}\left\{\tilde{H}_{i,j}\sum_{k=1}^{ns}\Delta\sigma_k V_{i,j,k} - \tilde{H}_{i,j-1}\sum_{k=1}^{ns}\Delta\sigma_k V_{i,j-1,k}\right\}. \quad (3.3)$$

Now, the semi-discretized system (3.1)-(3.3) can be written in the form

$$\frac{d}{dt}\begin{pmatrix} U \\ V \\ Z \end{pmatrix} = \begin{pmatrix} \Lambda_{\sigma\sigma} & F & -\Theta_2 g D_x \\ -F & \Lambda_{\sigma\sigma} & -\Theta_2 g E_y \\ -\Theta_1 H E_x & -\Theta_1 H D_y & 0 \end{pmatrix} \begin{pmatrix} U \\ V \\ Z \end{pmatrix}, \tag{3.4}$$

where $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{Z}$ are grid functions approximating the velocities u, v and ζ, respectively. The components $U_{i,j,k}$, $V_{i,j,k}$ and $Z_{i,j}$ are numbered lexicographically. $\Lambda_{\sigma\sigma}$ is a tridiagonal matrix approximating the vertical diffusion term, including the discretization of the term $1/h^2$. Θ_1 is a (nx·ny·ns)·(nx·ny) matrix (a row of ns diagonal matrices with $\Delta\sigma_k$ on the diagonal of the k-th matrix). Θ_2 is a (nx·ny)·(nx·ny·ns) matrix (a column of ns identity matrices). F is a four diagonal matrix (due to the grid staggering) of order nx·ny·ns approximating the Coriolis term. D_x and D_y are lower bidiagonal matrices of order nx·ny approximating the differential operators $\partial/\partial x$ and $\partial/\partial y$, respectively. E_x and E_y are upper bidiagonal matrices with $E_x = -D_x^T$ and $E_y = -D_y^T$. Both D_x and E_x are matrices that approximate the differential operator $\partial/\partial x$. However, the matrices differ slightly because of the grid staggering.

For example, in the case of ns = 4, the structure of system (3.4), in which the four diagonal matrix F is replaced by a diagonal matrix, is given in Figure 2.

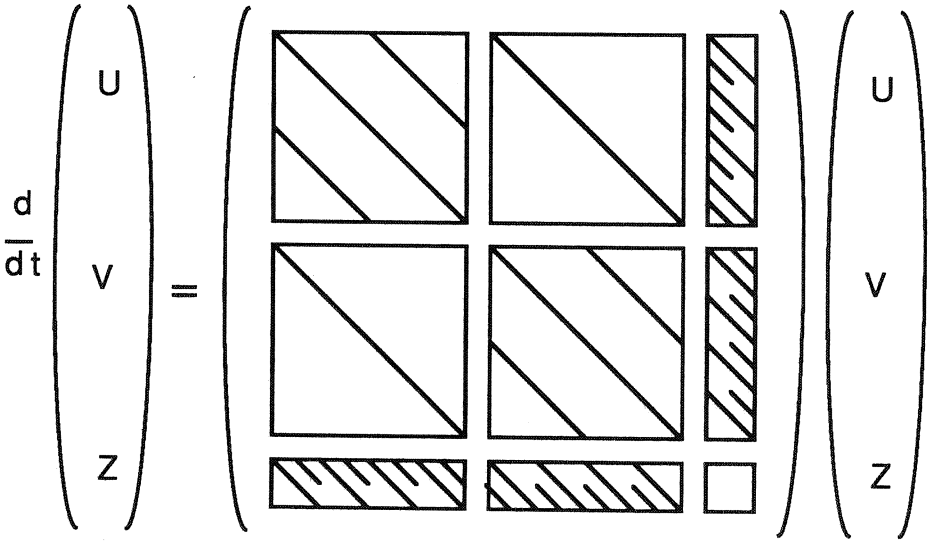

Figure 2. The structure of the semi-discretized system (3.4).

3.4. TIME INTEGRATION

In this section one-step time integrators for the semi-discretized system (3.4) are described. We will introduce time integrators that are explicit, semi-implicit or implicit in the vertical direction.

Considering explicit methods, we do not use the Forward Euler method, because the stability region of this method does not contain any part of the imaginary axis. Therefore, we apply the one-step, explicit, 3-stage, second-order Runge-Kutta method which has an imaginary stability boundary $\beta = 2$. Let the system of ODEs

$$\frac{dS}{dt} = F(t,S(t))$$

represent the semi-discretized system (3.4), with $S = (U,V,Z)^T$ and $F(t,S(t))$ denoting the right-hand side of (3.4). Then, the Runge-Kutta method can be written in the form

$$\begin{aligned} S_1 &= S^n \\ S_2 &= S^n + 0.5\tau\, F(t_n, S_1) \\ S_3 &= S^n + 0.5\tau\, F(t_n+0.5\tau, S_2) \\ S^{n+1} &= S^n + \tau\, F(t_n+0.5\tau, S_3)\ , \end{aligned} \tag{4.1}$$

where n denotes the time level $n\tau$, with τ the time step. It is known that the imaginary stability boundary of method (4.1) is optimal for explicit, 3-stage, second-order Runge-Kutta methods [4].

On the other hand, the (first-order) Backward Euler method for system (3.4) reads

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & -\tau F & \tau\Theta_2 g D_x \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & \tau\Theta_2 g E_y \\ \tau\Theta_1 H E_x & \tau\Theta_1 H D_y & I \end{pmatrix} \begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix}, \tag{4.2}$$

where the approximations at the time level $(n+1)\tau$ are denoted by an asterisk. This method requires extensive computation, since at each time step a linear system of order $nx \cdot ny \cdot (2ns+1)$ has to be solved. In order to reduce the computational complexity, we may uncouple the computation of the Z-component from the computation of the U- and V-components. This leads to

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & -\tau F & \tau\Theta_2 g D_x \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & \tau\Theta_2 g E_y \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -\tau\Theta_1 H E_x & -\tau\Theta_1 H D_y & I \end{pmatrix} \begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix}, \tag{4.3}$$

or to

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & -\tau F & 0 \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & 0 \\ \tau\Theta_1 HE_x & \tau\Theta_1 HD_y & I \end{pmatrix}\begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} I & 0 & -\tau\Theta_2 gD_x \\ 0 & I & -\tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix}. \quad (4.4)$$

Owing to the coupling of the velocity components, we do not use methods (4.3) and (4.4) in our experiments. A further simplification can be made by uncoupling the computation of the U-component from the V-component. By transferring the Coriolis term of the U-component to the right-hand side, we obtain for method (4.3)

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & 0 & \tau\Theta_2 gD_x \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & \tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} I & \tau F & 0 \\ 0 & I & 0 \\ -\tau\Theta_1 HE_x & -\tau\Theta_1 HD_y & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix}, \quad (4.5)$$

and for method (4.4)

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & 0 & 0 \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & 0 \\ \tau\Theta_1 HE_x & \tau\Theta_1 HD_y & I \end{pmatrix}\begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} I & \tau F & -\tau\Theta_2 gD_x \\ 0 & I & -\tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix}. \quad (4.6)$$

The methods can be made more symmetric in various ways. In the experiments we have observed that symmetrization of the Z-component deteriorates the stability considerably. Therefore, we will only symmetrize the velocity components. The symmetrical variant of method (4.6) reads

$$\begin{pmatrix} I-\frac{\tau}{2}\Lambda_{\sigma\sigma} & 0 & 0 \\ \tau F & I-\frac{\tau}{2}\Lambda_{\sigma\sigma} & 0 \\ \tau\Theta_1 HE_x & \tau\Theta_1 HD_y & I \end{pmatrix}\begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} I+\frac{\tau}{2}\Lambda_{\sigma\sigma} & \tau F & -\tau\Theta_2 gD_x \\ 0 & I+\frac{\tau}{2}\Lambda_{\sigma\sigma} & -\tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix}. \quad (4.7)$$

An explicit treatment of the Coriolis term can be achieved by transferring the Coriolis term of the V-component to the right-hand side. For method (4.5) we then obtain

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & 0 & \tau\Theta_2 gD_x \\ 0 & I-\tau\Lambda_{\sigma\sigma} & \tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} I & \tau F & 0 \\ -\tau F & I & 0 \\ -\tau\Theta_1 HE_x & -\tau\Theta_1 HD_y & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix}. \quad (4.8)$$

Splitting of the vertical diffusion term $\Lambda_{\sigma\sigma}$ into $\Lambda_l+\Lambda_u$, with Λ_l a lower bidiagonal matrix and Λ_u an upper bidiagonal matrix, leads to

$$\begin{pmatrix} I-\tau\Lambda_l & 0 & \tau\Theta_2 gD_x \\ 0 & I-\tau\Lambda_l & \tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^* \\ V^* \\ Z^* \end{pmatrix} = \begin{pmatrix} I+\tau\Lambda_u & \tau F & 0 \\ -\tau F & I+\tau\Lambda_u & 0 \\ -\tau\Theta_1 HE_x & -\tau\Theta_1 HD_y & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix} \quad (4.9)$$

At the next time step the matrices Λ_l and Λ_u are interchanged. Thus, the direction of the sweep is alternated every time step to avoid any bias in the method. This method has been developed in [1]. It requires the solution of bidiagonal systems.

Considering methods (4.5)-(4.8), nx·ny tridiagonal linear systems (of order ns) have to be solved at each time step. In the next section we will discuss efficient methods for the solution of the tridiagonal systems.

3.5. Solution of the tridiagonal systems

Here, we consider two methods for the solution of the tridiagonal systems. First we use the Gaussian Elimination (double sweep) method. Since this is a recursive method, it seems to be unattractive on vector and parallel computers. However, in our case we have a *large number* of independent tridiagonal systems. Therefore, the systems can be solved in a vector-parallel mode on e.g., the Alliant FX/4. This means that each processor is executing vector instructions to compute a certain operation of the Gaussian Elimination method for all tridiagonal systems. This method, which we denote by method GE, can be described schematically by

Method GE (Gaussian Elimination method)
for k=1,...,ns do (in **scalar** mode)
  for j=1,...,ny
  for i=1,...,nx do (in **vector-parallel** mode)
  perform some step of the GE method at layer k and point (i,j).

The loops with indices i and j can be collapsed into a single DO-loop to obtain a more efficient code. These iterations are executed in vector-parallel mode. For example, on a 4-pipe CYBER 205 and on a four-processor Alliant FX/4 this results in a comparable form of parallelism. The iterations of DO-loops are distributed across the pipes/processors until the entire DO-loop has been executed.

On vector computers method GE vectorizes well and also requires a minimal number of operations. However, on parallel computers parallelism at a higher level than at the innermost DO-loop level may be preferred. Especially, for short DO-loops the overhead due to vectorization and parallelization may be considerable. In our case, the loop with index k is recursive and is therefore not suited for parallel execution. In the literature several methods have been developed to reduce this recursion problem to smaller recursion problems. Here, we use a variant of Wang's method that has been developed in [8]. We now briefly describe this method.

Let us assume that ns can be factorized as ns = pq. The tridiagonal system is written as a p by p block matrix, in which each block is a q by q matrix. Then, the off-diagonal elements on the p diagonal blocks are eliminated in parallel. This method, which we denote by method WANG, can be considered as a method in which the Gaussian Elimination method is applied in parallel for all p diagonal blocks. This method reads

Method WANG (variant of Wang's method)
for k1=1,...,p do (in **parallel** mode)
  for k2=1,...,q do (in **scalar** mode)
    for j=1,...,ny
    for i=1,...,nx do (in **vector** mode)
     perform some step of the WANG method at layer
     k2+(k1-1)q and point (i,j).

Here, the loop with index k2 is recursive. The parallelism is at a higher level than for method GE. On the other hand, method WANG requires more operations than method GE (about 2.5 times as many).

We now give the results for the wind driven test problem that is described in more detail in the next section. The computations have been performed on a grid with nx=10, ny=18 and ns=24. In this case, 180 tridiagonal systems of dimension 24 have to be solved. In Table 5.1 we list the computation times for the solution of the tridiagonal systems on the Alliant FX/4 for various optimizations (–=no optimization, G=Global, V=Vector and P=Parallel). This mini-supercomputer has four vector processors. For method WANG we have divided the 24 vertical layers into four blocks (i.e., p = 4 and q = 6). Thus, the computations for each block have been performed on a different processor.

	# PROC.	(–)	(G)	(GV)	(GVP)
METHOD GE	1	37.2	9.6	2.47	2.59
	2				1.36
	3				1.00
	4		9.6	2.47	0.87
METHOD WANG	1	87.7	17.3	4.91	4.95
	2				2.64
	3				2.47
	4		17.3	4.91	1.60

Table 5.1. Computation times on the Alliant FX/4 (in s).

Without any optimization, method GE is about a factor 2.4 faster, which is in accordance with the number of operations. Also in vector mode, method GE is more efficient. The vectorization properties of both methods are comparable. Although method WANG requires about 2.5 times as many operations, the number of divisions is equal for both methods. Since divisions are more expensive than additions and multiplications we obtain a gain factor of 1.8 for method GE.

In parallel mode, we expected the smallest computation time for method WANG. Although method WANG requires more operations, we expected the parallelization overhead for method GE to be relatively larger. However, it turns out that even for

our relatively small test problem method GE is faster. On four processors we obtain a speed-up of about three for method GE. It should be noted that method WANG is superior when e.g., one large tridiagonal system has to be solved [8].

We conclude that it is not worthwhile to develop a method that contains parallelism at a higher level (as in method WANG), because a *large number* of independent tridiagonal systems has to be solved. Thus, on both vector and parallel computers method GE is the most efficient method for the solution of a large number of tridiagonal systems. Moreover, method GE requires a minimal number of operations. In the numerical experiments method GE will be used. It should be noted that on vector computers method GE is the only possibility.

3.6. NUMERICAL EXPERIMENTS

To compare the various time integrators we choose a test problem that has been used by others [1,3]. In this experiment the water is initially at rest and the motion in the basin is generated by a constant wind stress. The closed rectangular basin has dimensions representative of the North Sea. Thus, a wind driven circulation is gradually developed and finally reaches a steady state. In this experiment the total depth h in system (2.1)-(2.3) is replaced by d, which leads to a linear system of equations (cf. [1]). The following parameters are used in this experiment:

$L = 400$ km
$\Delta x = 400/9$ km
$B = 800$ km
$\Delta y = 800/17$ km
$f = 1.22\text{e-}4\ s^{-1}$
$g = 9.81\ m/s^2$
$d = 65$ m
$\mu = 0.065\ m^2/s$
$C = 70\ m^{1/2}/s$
$W_f = 1.5\ kg/ms^2$
$\rho = 1025\ kg/m^3$
$\phi = 90°$.

We integrate over a period of 24 hours, with time steps of 3, 10, 20 and 30 minutes. The experiments have been carried out on a (2-pipe) CDC CYBER 205. Tables 6.1 and 6.2 show the water elevation computed at the south-western corner of the basin for two different vertical resolutions, namely $\Delta\sigma = 1/ns$, with ns=5 and ns=25. Overflow is denoted by ***.

The methods used in this experiment are:

the 3-stage Runge-Kutta method (4.1)
the vertically implicit method (4.5)
the vertically implicit method (4.6)
the symmetrized, vertically implicit method (4.7)
the vertically implicit method (4.8)
the vertically semi-implicit method (4.9).

method	Δt (min)	ζ_{max} (cm)	time (hrs)	ζ_{min} (cm)	time (hrs)	comp. time (s)
(4.1)	3	172.6	8.7	45.8	18.3	4.36
	10	172.3	8.7	45.9	18.3	1.31
	20	171.6	8.7	46.6	18.3	0.65
	30	***				
(4.5)	3	173.0	8.8	45.5	18.3	1.26
	10	173.7	8.7	44.8	18.5	0.38
	20	175.0	9.0	44.0	18.7	0.19
	30	***				
(4.6)	3	173.0	8.7	45.5	18.3	1.26
	10	174.1	8.7	44.8	18.3	0.38
	20	176.1	8.7	43.2	18.3	0.19
	30	***				
(4.7)	3	172.9	8.7	45.6	18.3	1.53
	10	173.8	8.7	45.1	18.3	0.46
	20	175.7	8.7	44.0	18.3	0.23
	30	***				
(4.8)	3	172.7	8.7	45.5	18.3	1.26
	10	172.9	8.8	44.8	18.3	0.38
	20	174.0	9.0	42.6	18.3	0.19
	30	***				
(4.9)	3	172.5	8.7	45.8	18.3	1.34
	10	172.5	8.8	45.6	18.3	0.40
	20	173.4	8.7	44.4	18.3	0.20
	30	***				

Table 6.1. Surface elevations with ns=5.

Table 6.1 shows that the results are comparable for all methods when ns=5. However, in the case of 25 layers the methods behave differently (see Table 6.2). The RK3 method becomes unstable for already the smallest time step used in this experiment. Method (4.9), in which bidiagonal systems have to be solved, yields accurate solutions for time steps of maximally five minutes. However, the vertically implicit methods (4.5)-(4.8) behave as in the case of five layers. In both experiments the maximally stable time step is about 20 minutes. It seems that for these methods the maximally stable time step is independent of the vertical mesh size. In the next section we will carry out a stability analysis for one of the vertically implicit methods, viz., method (4.6).

method	Δt (min)	ζ_{max} (cm)	time (hrs)	ζ_{min} (cm)	time (hrs)	comp. time (s)
(4.1)	3	***				
(4.5)	3	173.8	8.8	41.0	18.3	4.31
	10	174.8	8.8	40.3	18.5	1.29
	20	176.9	9.0	38.7	18.7	0.65
	30	***				
(4.6)	3	173.8	8.7	41.0	18.3	4.30
	10	174.8	8.7	40.3	18.3	1.29
	20	176.9	8.7	38.7	18.3	0.65
	30	***				
(4.7)	3	173.7	8.7	41.1	18.3	5.79
	10	174.7	8.7	40.6	18.3	1.74
	20	176.5	8.7	39.4	18.3	0.87
	30	***				
(4.8)	3	173.4	8.8	41.0	18.3	4.36
	10	173.7	8.7	40.2	18.3	1.29
	20	174.7	9.0	37.9	18.3	0.65
	30	***				
(4.9)	3	173.5	8.6	41.1	18.2	5.05
	10	179.4	8.3	34.1	17.8	1.51
	20	212.1	8.0	-3.1	18.3	0.76
	30	***				

Table 6.2. Surface elevations with ns=25.

The numerical results show that the vertically implicit methods are more robust than the other two methods. Thus, an implicit treatment for the vertical diffusion term is beneficial because of stability considerations. For the vertically implicit methods, the **U**-, **V**- and **Z**-component are computed after each other. This is advantageous for the both the stability and the storage requirements.

Concerning computation time, it is evident that the vertically implicit methods (4.5)-(4.8) can be computed efficiently on vector and parallel computers. It is surprising that the vertically implicit methods are slightly more efficient than method (4.9), in which bidiagonal systems have to be solved. This is due to the way of programming. The tridiagonal systems are build up and solved at the same time. This leads to a smaller number of divisions. In general, the efficiency of both methods will be comparable.

We conclude this section with an overview of computation times on various computers, ranging from supercomputers to personal computers. This could easily

be done, because the code was written in the ANSI FORTRAN 77 programming language. We have applied method (4.6) to the test problem with ns=25 and τ=1200 s (see Table 6.2). In Table 6.3 we list the results (* = 32 bits precision, otherwise 64 bits precision).

COMPUTER	OPTIMIZATION (-)	(G)	(GV)	(GVP)
MACINTOSH PLUS *	6110			
VAX-11/780 *	508	178		
ALLIANT FX/4 *	125.9	30.9	8.4	2.82
ALLIANT FX/4				3.67
CDC CYBER 990	53.3	10.6		
CDC CYBER 205	11.9	5.8	0.65	
CRAY X-MP/28			0.236	
NEC SX/2 *			0.088	

Table 6.3. Computation times (in s).

For this test problem the CDC CYBER 205 is about 6 times faster than the Alliant FX/4, and the NEC SX/2 supercomputer is about 70.000 times faster than the Macintosh Plus ! This clearly illustrates the need of supercomputers.

3.7. STABILITY ANALYSIS

In this section the stability of method (4.6) is examined. For that purpose we introduce the eigenvalues $i\delta_x$, $i\delta_y$ and $\gamma_{\sigma\sigma}$ of the difference operators D_x, D_y and $\Lambda_{\sigma\sigma}$, corresponding to the eigenvectors $e^{i(\alpha_1\Delta x+\alpha_2\Delta y+\alpha_3\Delta\sigma)}$. The following assumptions are made:

a) the total depth h is constant

b) $\sum_{k=1}^{ns} \Delta\sigma_k\tau(h_x+hD_x)U_{i,j,k} = \tau(h_x+hD_x)\,U_{i,j}$

c) $\forall k : 1 \le k \le ns : \Delta\sigma_k = 1/ns$.

Although assumption b) reduces the analysis to a two-dimensional stability analysis, the results are in agreement with our three-dimensional experiments. We now construct for method (4.6) the so-called amplification matrix. This amplification matrix may be written in the form

$$G = A^{-1} B ,$$

where

$$A = \begin{pmatrix} I - \tau\gamma_{\sigma\sigma} & 0 & 0 \\ \tau F & I - \tau\gamma_{\sigma\sigma} & 0 \\ i\tau h\delta_x & i\tau h\delta_y & I \end{pmatrix}, \quad B = \begin{pmatrix} I & \tau F & -i\tau g\delta_x \\ 0 & I & -i\tau g\delta_y \\ 0 & 0 & I \end{pmatrix},$$

where

$$\delta_x = \frac{\sin(\alpha_1\, 0.5\Delta x)}{0.5\Delta x}, \quad \delta_y = \frac{\sin(\alpha_2\, 0.5\Delta y)}{0.5\Delta y}, \quad \gamma_{\sigma\sigma} = \frac{-2+\cos(\alpha_3\Delta\sigma)}{(\Delta\sigma)^2}\,\frac{\mu}{h^2} \tag{7.1}$$

and α_1, α_2 and α_3 are constants. Stability in the sense of O'Brien-Hyman-Kaplan [10] is ensured if $\|G\| \le 1$. To study the stability we write

$$\begin{aligned} G &= q\ \ q^{-1}\, A^{-1}\, q\ \ q^{-1}\, B\, q\ \ q^{-1} && \Leftrightarrow \\ G &= q\,(q^{-1}\, A\, q)^{-1}\ (q^{-1}\, B\, q)\ \ q^{-1} && \Leftrightarrow \\ G &= q\ \ \tilde{A}^{-1}\ \ \tilde{B}\ \ q^{-1} \end{aligned}$$

where

$$q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sqrt{\frac{h}{g}} \end{pmatrix},$$

and

$$\tilde{A} = \begin{pmatrix} I - \tau\gamma_{\sigma\sigma} & 0 & 0 \\ \tau F & I - \tau\gamma_{\sigma\sigma} & 0 \\ i\tau\tilde{\delta}_x & i\tau\tilde{\delta}_y & I \end{pmatrix} \text{ and } \tilde{B} = \begin{pmatrix} I & \tau F & -i\tau\tilde{\delta}_x \\ 0 & I & -i\tau\tilde{\delta}_y \\ 0 & 0 & I \end{pmatrix},$$

with

$$\tilde{\delta}_x = \sqrt{gh}\,\frac{\sin(\alpha_1\, 0.5\Delta x)}{0.5\Delta x}, \quad \tilde{\delta}_y = \sqrt{gh}\,\frac{\sin(\alpha_2\, 0.5\Delta y)}{0.5\Delta y}.$$

In the remainder of the section we will omit the tildes. Suppose that λ denotes the eigenvalues of G. Then, the eigenvalues satisfy

$$\det|A^{-1}B - \lambda I| = 0 \qquad \Leftrightarrow$$
$$\det|A^{-1}| \cdot \det|B - \lambda A| = 0 .$$

Assuming that A is invertible, we find

$$\det|B - \lambda A| = \det \begin{vmatrix} I+\lambda(\tau\gamma_{\sigma\sigma}-1) & \tau F & -i\tau\delta_x \\ -\lambda\tau F & I+\lambda(\tau\gamma_{\sigma\sigma}-1) & -i\tau\delta_y \\ -i\lambda\tau\delta_x & -i\lambda\tau\delta_y & I-\lambda \end{vmatrix} = 0 . \qquad (7.2)$$

The eigenvalue equation for G is

$$\lambda^3 + a_1\lambda^2 + a_2\lambda + a_3 = 0 ,$$

where

$$a_1 = -\frac{(\tau\gamma_{\sigma\sigma} - 3 + \tau^2\delta_x^2 + \tau^2\delta_y^2)(\tau\gamma_{\sigma\sigma} - 1) - \tau^2F^2 + \tau^3\delta_x\delta_yF}{(1-\tau\gamma_{\sigma\sigma})^2}$$

$$a_2 = -\frac{2\tau\gamma_{\sigma\sigma} - 3 + \tau^2\delta_y^2 + \tau^2F^2 + \tau^2\delta_x^2 - \tau^3\delta_x\delta_yF}{(1-\tau\gamma_{\sigma\sigma})^2}$$

$$a_3 = -\frac{1}{(1-\tau\gamma_{\sigma\sigma})^2} .$$

Note that the coefficients are real, whereas the matrix in (7.2) is complex. Therefore, we can apply the Hurwitz-criterion [9] to ensure that the eigenvalues of the amplification matrix G are within the unit circle. Thus, we must require

$$1 + a_1 + a_2 + a_3 > 0$$
$$1 - a_1 + a_2 - a_3 > 0$$
$$3 + a_1 - a_2 - 3a_3 > 0$$
$$1 - a_2 + a_1a_3 - a_3^2 > 0 .$$

We then obtain the following inequalities:

(a) $\tau^3\gamma_{\sigma\sigma}(\delta_x^2 + \delta_y^2) < 0$

(b) $\tau^2\gamma_{\sigma\sigma}^2 - 4\tau\gamma_{\sigma\sigma} + 4 + \tau^2(\delta_x^2 + \delta_y^2)(0.5\tau\gamma_{\sigma\sigma} - 1) + \tau^3\delta_x\delta_yF - \tau^2F^2 > 0$

(c) $\tau^2\gamma_{\sigma\sigma}^2 + \tau^2(\delta_x^2 + \delta_y^2)\,(2 - \tau\gamma_{\sigma\sigma}) - 2\tau^3\delta_x\delta_y F + 2\tau^2F^2 > 0$

(d) $\tau^3\gamma_{\sigma\sigma}^3 - 2\tau^2\gamma_{\sigma\sigma}^2 + \tau\gamma_{\sigma\sigma}\,(\tau^2\delta_x^2 + \tau^2\delta_y^2 - \tau^3\delta_x\delta_y F + \tau^2F^2) - 2\tau^2F^2$

$+ 2\tau^3\delta_x\delta_y F - \tau^2\delta_x^2 - \tau^2\delta_y^2 < 0\,.$

From (7.1) it can be easily seen that $\gamma_{\sigma\sigma} < 0$. Thus, inequality (a) can not be satisfied if $\delta_x = \delta_y = 0$. However, in that case the amplification matrix is of the simple form

$$G = \begin{pmatrix} I+\lambda(\tau\gamma_{\sigma\sigma}-1) & \tau F & 0 \\ -\lambda\tau F & I+\lambda(\tau\gamma_{\sigma\sigma}-1) & 0 \\ 0 & 0 & I-\lambda \end{pmatrix},$$

with the eigenvalues

$$\lambda = 1,\ \frac{-\tau^2F^2 - 2(\tau\gamma_{\sigma\sigma} - 1) \pm \tau F\sqrt{\tau^2F^2 + 4(\tau\gamma_{\sigma\sigma} - 1)}}{2(\tau\gamma_{\sigma\sigma} - 1)^2}.$$

Thus, inequality (a) can be rewritten to

$$\text{(a')}\quad \frac{-\tau^2F^2 - 2(\tau\gamma_{\sigma\sigma} - 1) \pm \tau^2F\sqrt{\tau^2F^2 + 4(\tau\gamma_{\sigma\sigma} - 1)}}{2(\tau\gamma_{\sigma\sigma} - 1)^2} \le 1\,.$$

The inequalities (a'), (b), (c) and (d) are too complicated to derive stability conditions. Therefore, we neglect the influence of the Coriolis term. Then, it can easily be verified that inequalities (a'), (c) and (d) are always satisfied. For the second inequality we obtain

$$\tau < \frac{1}{\sqrt{gh}}\,\frac{1}{\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}}. \qquad (7.3)$$

This condition shows that the maximally stable time step is independent of $\Delta\sigma$. For the parameters used in our experiment, stability condition (7.3) yields a maximally stable time step of about 1300 seconds, which is in agreement with the numerical results. Experimentally, we have observed that the maximal stable time step is of the same order when the Coriolis term is not neglected.

3.8. CONCLUSIONS

In this paper we have investigated the stability and efficiency of one-step time integrators for the three-dimensional hydrodynamic equations. From our linear test problem, it appears to be necessary to treat the vertical diffusion term in an implicit or semi-implicit way. The vertically implicit methods (4.5)-(4.8) perform better than the semi-implicit method (4.9). For both the stability and the storage requirements it is advantageous to compute the U-, V- and Z-component sequentially. The vertically implicit methods satisfy this condition.

For these methods a large number of tridiagonal systems have to be solved. We have considered two methods for the solution of these systems. The method in which the tridiagonal systems are solved by the Gaussian Elimination method appears to be the most efficient one. Because of the large number of tridiagonal systems, these systems are solved in a vector-parallel mode, resulting in a high performance on vector and parallel computers. Moreover, this method requires a minimal number of operations.

REFERENCES

1. A.M. DAVIES, Application of the DuFort-Frankel and Saul'ev methods with time splitting to the formulation of a three dimensional hydrodynamic sea model, *Int. J. Numer. Meth. in Fluids*, 5, 405-425 (1985).
2. E.D. DE GOEDE, *Finite difference methods for the three-dimensional hydrodynamic equations*, Report NM-R8813, CWI, Amsterdam, 1988.
3. N.S. HEAPS, On the numerical solution of the three-dimensional hydrodynamic equations for tides and storm surges, *Mem. Soc. Roy. Sci. Liège Ser. 6*, 2, 143-180 (1972).
4. P.J. VAN DER HOUWEN, *Construction of integration formulas for initial-value problems*, North-Holland, Amsterdam, 1977.
5. J.J. LEENDERTSE, R.C. ALEXANDER, AND S.-K. LIU, *A three dimensional model for estuaries and coastal seas : volume I, Principles of computation*, The Rand Corporation, RM-1417-OWRR, 1973.
6. N.A. PHILLIPS, A coordinate system having some special advantages for numerical forecasting, J. *Meteorol.*, 14, 184-194 (1957).
7. G.S. STELLING, *On the construction of computational methods for shallow water flow problems*, Ph.D. Thesis, TU Delft, 1983.
8. H.A. VAN DER VORST AND K. DEKKER, The vectorization of linear recurrence relations, *SIAM J. on Sci. and Stat. Comput.*, 2, 27-35 (1989).
9. J.D. LAMBERT, *Computational methods in ordinary differential equations*, John Wiley & Sons, London, 1973.
10. G.G. O'BRIEN, M.A. HYMAN AND S. KAPLAN, A study of the numerical solution of partial differential equations, *J. Mathematics Phys.*, 29, 223-251 (1950).

Chapter 4

Stabilization of a Time Integrator for the 3D Shallow Water Equations by Smoothing Techniques

E.D. de Goede
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB Amsterdam, The Netherlands

A smoothing technique is applied to improve the stability of a semi-implicit time integrator for the three-dimensional shallow water equations. In this method the terms involving the vertical direction are treated implicitly. The stability condition for the time step only depends on the horizontal mesh sizes. Therefore, in the horizontal direction a smoothing operator is added. Owing to the smoothing, the maximally stable time step increases considerably, while the accuracy is hardly affected. Moreover, it turns out that the smoothing operator is efficient on vector and parallel computers.

4.1. INTRODUCTION

In numerical analysis, we distinguish explicit and implicit time integrators for partial differential equations. It is well-known that implicit methods are in general stable for any time step, but cannot exploit the facilities of vector and parallel computers as well as explicit methods do. On the other hand, explicit methods impose a severe restriction on the time step and therefore the time step is not dictated by accuracy considerations. To improve the stability of explicit methods, we will use smoothing techniques.

Smoothing techniques are frequently applied in numerical methods. Usually, the smoothing technique consists in applying a matrix S to some vector **F**. The aim is to reduce the magnitude of the high frequencies occurring in the Fourier expansion of the vector to be smoothed, without affecting the lower frequencies too much. A simple example of an m × m smoothing matrix S is given by **G** = S**F**, where

$$\begin{aligned} G_1 &= F_1 \\ G_i &= \frac{1}{4}(F_{i-1} + 2F_i + F_{i+1}) , \quad i=2,...,m-1, \\ G_m &= F_m \end{aligned} \tag{1.1}$$

with F_i and G_i denoting the components of the vectors **F** and **G**, respectively.

In this paper our starting point is the semi-implicit time integrator that has been developed for the linearized three-dimensional shallow water equations (SWEs)

in [3]. In this method only the vertical terms are treated implicitly. For this method we are faced with a CFL stability condition that depends on the horizontal mesh sizes Δx and Δy. For small values of Δx and Δy this time step restriction may be more severe than necessary for accuracy considerations. Therefore, we will add a smoothing operator in the horizontal direction to make the stability condition due to the horizontal mesh sizes less restrictive.

The time integrator described in [3] can be considered as a method in which an implicit smoothing operator already appears in the vertical direction. The smoothing in both the horizontal and the vertical direction may be interpreted as a preconditioning of the right-hand side of the semi-discrete shallow water equations. It will be shown that the maximally stable time step increases considerably when the smoothing operator in the horizontal direction is applied. The time step for the stabilized time integrator is now dictated by accuracy considerations, as it applies to implicit methods. Moreover, the stabilized time integrator can be implemented efficiently, as will be shown in the experiments. The efficiency of this method will be tested on various domains. In the experiments we will use a rectangular domain representing the North Sea and an irregular domain representing the IJsselmeer. The IJsselmeer is the largest lake in the Netherlands.

The technique of stabilizing explicit time integrators by right-hand side smoothing has been applied by Wubs for the numerical solution of the two-dimensional shallow water equations [10]. For an overview of various smoothing techniques we refer to [4].

Section 4.2 provides the theory for the smoothing. In Section 4.3 we describe the semi-implicit time integrator for the shallow water equations. In Section 4.4 the smoothing is applied to stabilize this time integrator. Section 4.5 is devoted to the implementation of the smoothing matrices. Finally, in Section 4.6 we show by a number of experiments that application of the smoothing operators leads to a considerable reduction of the computation time, while the accuracy remains acceptable. The numerical solution is compared with a solution computed with a very small time step on the domain used in the experiments. This reference solution may therefore be considered as an almost exact solution on this domain. The reduction of the computation time is more or less independent of the domain. When the solution tends to a steady state, we even obtain a reduction factor of about 10.

4.2. RIGHT-HAND SIDE SMOOTHING

Consider the partial differential equation

$$\frac{\partial w}{\partial t} = Lw(t,x) + c(t,x), \tag{2.1}$$

where L is a *linear* differential operator with respect to the space variable x and c is a given function. This equation, together with its boundary conditions, can be semi-discretized into a system of ordinary differential equations (ODEs) of the form

$$\frac{dW}{dt} = J\,W(t) + C(t), \tag{2.2}$$

with J the Jacobian matrix, C an approximation to c and W an approximation to w at the grid points used for the semi-discretization. We shall always assume that this system is stable in the sense that the eigenvalues of J are in the nonpositive half plane. In Section 4.3 we shall see that the linearized 3D shallow water equations can be semi-discretized into this form.

If the system (2.2) is integrated by an *explicit* time integrator, then its maximally stable time step is limited owing to the usually extremely large magnitude of the spectral radius of J. Therefore, the time step has to be unrealistically small in order to achieve stability. This restriction is a drawback if the variation of the solution in time is so small that accuracy considerations would allow a larger time step. To obtain a better conditioned right-hand side function, we premultiply the right-hand side of the original semi-discretization (2.2), or some part of it, by a *smoothing operator* S. Thus, we replace (2.2) either by

$$\frac{dW}{dt} = S \{ J W(t) + C(t) \}, \tag{2.3a}$$

or by

$$\frac{dW}{dt} = SJ W(t) + C(t). \tag{2.3b}$$

In (2.3b) a part of the right-hand side is smoothed. The semi-discretization (2.3a) is particularly attractive in problems where it is known that the time derivative of the exact solution, i.e., $\partial w/\partial t$ is a smooth function of the space variable x (e.g., in problems where a steady state is to be approximated). In such cases the right-hand side function of the semi-discretization (2.2) is also a 'smooth' grid function, so that it may be premultiplied by the smoothing operator S without much loss of accuracy.

The maximally stable time step may increase considerably when the explicit time integrator is applied to (2.3) instead of to (2.2). To achieve that the condition of SJ is better than that of J, the operator S should strongly damp the high frequencies (stiff components) in the Fourier expansion of the vector JW, so that the spectral radius of SJ is substantially less than that of J. One may consider the equations (2.3) as 'smoothed' or 'preconditioned' semi-discretizations of the original equation (2.1).

We emphasize that, in this paper, the right-hand side function is smoothed, instead of the grid function W(t) itself. The latter type of smoothing is often used. However, it may only be applied, without considerable loss of accuracy, if W(t) itself is a 'smooth' grid function for a fixed value of t. This is in general not the case. An example of this latter type of smoothing is the well-known Lax-Wendroff method [8].

To characterize the effect of right-hand side smoothing on the accuracy of the initial semi-discretization (2.2), we introduce the *order of consistency of smoothing operators*. Let Δ be the mesh size, then the smoothing operator S is said to be consistent of order p if $S=I+O(\Delta^p)$ as Δ tends to zero. Hence, S converges to the identity operator I if the grid is refined.

We remark that the application of right-hand side smoothing is not restricted to linear ODEs. Right-hand side smoothing can also be applied to more general systems of the form

$$\frac{dW}{dt} = F(t,W(t)), \tag{2.2'}$$

by replacing it by the smoothed system

$$\frac{dW}{dt} = S\,F(t,W(t)). \tag{2.3'}$$

Summarizing, the smoothing operator S should satisfy the following requirements:

(A) S is consistent of order $p \geq 1$
(B) the smoothed system is again stable
(C) the spectral radius of SJ is considerably smaller than that of J
(D) the application of the operator S does not require much computational effort.

In the following subsections it will be shown that, instead of looking for highly stable integration methods, one may equally well apply methods in which the right-hand side function of the system of ODEs (2.2) is premultiplied by a smoothing operator S such that the magnitude of the spectral radius associated with the right-hand side function reduces considerably. We distinguish smoothing that is dependent on and smoothing that is largely independent of the right-hand side function. The former type of smoothing is based on operator splitting and will be discussed in Section 4.2.1. Smoothing operators that are to a large degree independent of the right-hand side function will be discussed in Section 4.2.2.

4.2.1. Smoothing operators based on operator splitting

Smoothing operators based on operator splitting are suggested by considering splitting methods developed for the time integration of partial differential equations. Our starting point is the forward Euler method applied to the semi-discretization (2.2), which can be described by

$$W^{n+1} = W^n + \tau \{ JW^n + C^n \}, \tag{2.4}$$

where τ is the time step and W^n and C^n denote approximations to $W(n\tau)$ and $C(n\tau)$, respectively. Let us split the matrix J into

$$J = J_1 + J_2,$$

and let us replace the forward Euler method (2.4) by the splitting method

$$W^{n+1} - \tau J_2 W^{n+1} = W^n + \tau \{ J_1 W^n + C^n \},$$

or, equivalently,

$$W^{n+1} = (I - \tau J_2)^{-1} \{ (I + \tau J_1) W^n + \tau C^n \} . \tag{2.5}$$

This method can be rewritten as

$$W^{n+1} = W^n + \tau S \{ J W^n + C^n \}, \tag{2.6}$$

with

$$S = (I - \tau J_2)^{-1} . \tag{2.7}$$

The splitting method (2.6)-(2.7) may be interpreted as the forward Euler method applied to the system of ODEs (2.3a), which is a 'smoothed' version of the initial semi-discretization (2.2), with a smoothing operator S defined by (2.7). By an appropriate choice of the matrix J_2, this splitting method has much better stability characteristics than the forward Euler method (2.4). For example, the choices J_2=J and J_2=J/2 lead to the A-stable methods of Laasonen (backward Euler) and Crank-Nicolson (trapezoidal rule), respectively. Another possibility is to choose J_2 equal to a lower (or upper) triangular matrix. For the two-dimensional shallow water equations such an approach has been followed by Fischer [2] and Sielecki [9]. In fact, the method developed in [3] for the linearized shallow water equations may be interpreted as a combination of the Crank-Nicolson method and the approach of Sielecki and Fischer. In that paper it was shown that the stability of the resulting numerical method improves considerably, whereas the computations can be performed efficiently.

4.2.2. SMOOTHING OPERATORS FOR GENERAL VECTOR FUNCTIONS

The smoothing operators considered in the previous subsection strongly depend on the specific form of the right-hand side function. In this subsection we summarize the main properties of the family of smoothing operators developed in [4,6]. These operators are largely independent of the particular form of the vector function to which they are applied and therefore we shall present the results for the general equation (2.3'). We will again assume that the eigenvalues of the Jacobian matrix J:=∂F/∂W are in the nonpositive half plane.

The smoothing operator S will be chosen of the form S=P(D), where D is a difference matrix and the smoothing function P(z) is a polynomial or a rational function, yielding explicit or implicit smoothing operators, respectively. First we discuss the choice of the matrix D. In our *theoretical* considerations we assume that D is equal to the Jacobian J, normalized by its spectral radius, i.e.,

$$D = \frac{J}{\rho(J)} . \tag{2.8}$$

We emphasize that in *practice* it is generally not attractive to choose D according to (2.8) and we shall employ some cheap approximation to the normalized Jacobian matrix. If D is defined according to (2.8), then the eigenvalues of SJ=P(D)J are given by $\rho(J)zP(z)$, where z runs through the spectrum of D.

4.2.2.1. EXPLICIT SMOOTHING OPERATORS

In the case of explicit smoothing we are looking for a polynomial P(z) such that the magnitude of zP(z) is sufficiently small with P(0)=1 and z either in [–1,0] or in [–i,i]. Moreover, the polynomial P(z) will be chosen such that zP(z) remains in the nonpositive half plane. It was shown in [4,6] that polynomials of the form

$$P(z) = \frac{U_{2k}\left(\sqrt{1 + z^2}\right)}{2k + 1}, \quad U_{2k}(x) := \frac{\sin((2k+1)\arccos(x))}{\sin(\arccos(x))}, \tag{2.9}$$

minimize the magnitude of zP(z) on the purely imaginary interval [–i,i]. However, if z has negative real parts, then it may happen that Re {zP(z)} > 0 causing unstable behaviour. Since we shall apply smoothing to vector functions whose Jacobian matrices possess eigenvalues with negative real parts (caused by the vertical diffusion and the bottom friction in the SWEs), we require that Re {zP(z)} ≤ 0 for all values with Re {z} ≤ 0 (see condition (B)). For this case the following theorem defines a family of nearly optimum polynomials [4,6].

THEOREM 2.1. *Let* D *be defined by* (2.8), *let* S=P(D) *with* P(z) *defined by*

$$P(z) = \frac{T_k(1+2z^2) - 1}{2k^2 z^2}, \quad T_k(x) = \cos(k \arccos(x)). \tag{2.10}$$

Then the following assertions hold:

(a) *If* Re {z} ≤ 0 *then* Re {zP(z)} ≤ 0.

(b) *If* z *is purely imaginary, then* zP(z) *is again purely imaginary and for sufficiently large* k *its maximum is approximately*

$$\frac{2}{\pi k}. \tag{2.11}$$

PROOF. For a proof of (a) we refer to [4,6].

(b) We have to find the maximum of |zP(z)| on [–i,i], or, equivalently,

$$\max \left| \{T_k(1+2z^2) - 1\} \frac{1}{2k^2 z} \right|. \tag{2.12}$$

The range of $\{1+2z^2\}$ in (2.12) is [–1,1]. On this interval the Chebyshev polynomial $T_k(1+2z^2)$ satisfies the 'so-called' equal ripple property [5], which means that it alternatingly assumes equal maximum and minimum values. Because of the factor $1/(2k^2z)$, let us now assume that the value in (2.12) can be

approximated at the smallest value of |z| for which $T_k(1+2z^2)$ reaches its minimum. Thus, we require that

$$T_k(1+2z^2) = \cos(k \arccos(1+2z^2) = -1 ,$$

for |z| as small as possible, which yields

$$z = \pm i \frac{\sqrt{1-\cos(\pi/k)}}{\sqrt{2}} .$$

For these values of z we obtain that (2.12) is bounded by

$$\frac{\sqrt{2}}{k^2\sqrt{1-\cos(\pi/k)}} \approx \frac{2}{\pi k} , \tag{2.13}$$

for k sufficiently large. For many values of k we verified numerically that the reduction factor is close to $2/\pi k$. Therefore, we conclude that the approximation applied in this theorem is justified. []

An extremely efficient implementation of the smoothing operator of Theorem 2.1 can be obtained by using the following factorization theorem (see also Section 4.5), which justifies the application of these smoothing operators.

THEOREM 2.2. *Let the matrix* D *be defined by* (2.8), *let* S=P(D) *with* P(z) *defined by* (2.10), *let the factor matrices* F_j *be generated by*

$$F_1 = I + D^2, \quad F_{j+1} = (I - 2F_j)^2 , \; j > 0 ,$$

and let $k = 2^q$. *Then,* S *can be factorized by*

$$S = F_q F_{q-1} \cdots F_1 . \tag{2.14}$$

For a proof of Theorem 2.2 we refer to [4,6]. []

4.2.2.2. IMPLICIT SMOOTHING OPERATORS

In this subsection we will discuss implicit smoothing, i.e., if F is the vector to be smoothed, then the smoothed vector G is obtained by solving $G = S^{-1}F$, where S^{-1} is a smoothing operator (see (1.1)). Implicit smoothing has been applied in [7,10].

THEOREM 2.3. *Let* S^{-1}=P(D) *with* D *a difference matrix and* P(z) *defined by*

$$P(z) = \frac{1}{1 - \frac{1}{4}\alpha^2 z^2}, \quad \textit{with} \quad \alpha > 0 . \tag{2.15}$$

Then the following assertion holds:
If z is purely imaginary, then

$$| zP(z) | \leq \frac{1}{\alpha}.$$

PROOF. This follows immediately from elementary analysis. □

As mentioned before, in practice we shall choose D equal to some cheap approximation of the normalized Jacobian which satisfies condition (B). In choosing a difference matrix D the boundary conditions have to be incorporated in D. This is important to preserve conservation of mass. In this paper we shall choose smoothing operators of the form

$$D^2 = \frac{1}{4}\begin{pmatrix} 0 & & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 0 & & & & 0 \end{pmatrix}. \tag{2.16}$$

The implicit smoothing operator described in Theorem 2.3 with D^2 as in (2.16) results in the solution of a tridiagonal system. Therefore, this implicit smoothing operator does not require much computational effort. In practice, the value of α in (2.15) depends on the time step and on the mesh sizes.

Let us now discuss the order of consistency of the smoothing operator S with D^2 defined in (2.16). We assume that D^2 and P(z) satisfy the conditions

$$D^2 = O(\Delta^s) \text{ as } \Delta \to 0, \quad P(z) = 1 + O(z^{2r}) \text{ as } z \to 0, \tag{2.17}$$

where Δ denotes the mesh size, r and s are positive integers. Hence, S is consistent of order p=rs. For example, the smoothing matrix defined in (1.1) can be generated by $P(z)=1+z^2$ with D^2 defined by (2.16) and is second-order consistent (s=2, r=1). When P(z) is defined by (2.10) and D^2 by (2.16), then it can be easily verified that S is also second-order consistent.

Summarizing, if we choose the matrix D^2 defined by (2.16), then the smoothing matrix S=P(D), with P(z) the polynomial (2.10) or the rational function (2.15), reduces the magnitude of the spectrum associated with the right-hand side function considerably, whereas the spectrum remains in the nonpositive half plane. For this choice of D^2, the smoothing matrix S is independent of the right-hand side function.

In Section 4.4 we shall use both smoothing based on operator splitting and smoothing of general vector functions based on the theorems in Section 4.2.2.

4.3. MATHEMATICAL MODEL

In this section we will describe the mathematical model and the time integrator to which the smoothing will be applied. We will use a three-dimensional model in sigma co-ordinates in which the advective terms have been omitted. This model is described by

$$\frac{\partial u}{\partial t} = fv - g\frac{\partial \zeta}{\partial x} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial u}{\partial \sigma}\right) \tag{3.1}$$

$$\frac{\partial v}{\partial t} = -fu - g\frac{\partial \zeta}{\partial y} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial v}{\partial \sigma}\right) \tag{3.2}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) - \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right), \tag{3.3}$$

with boundaries

$$0 \le x \le L$$
$$0 \le y \le B$$
$$1 \ge \sigma \ge 0 .$$

Thus, the domain is a rectangular basin. Owing to the sigma transformation in the vertical, the domain is constant in time. We have the closed boundary conditions

$$u(0,y,\sigma,t) = 0, \quad u(L,y,\sigma,t) = 0,$$
$$v(x,0,\sigma,t) = 0, \quad v(x,B,\sigma,t) = 0.$$

The boundary conditions at the sea surface ($\sigma = 0$) are given by

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \cos(\phi), \quad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \sin(\phi).$$

At the bottom ($\sigma = 1$) we have a linear law of bottom friction of the form

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g\,u_d}{C^2}, \qquad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g\,v_d}{C^2},$$

with u_d and v_d representing the components of the current at some depth near the bottom.

4.3.1. SPACE DISCRETIZATION

For the space discretization of the equations (3.1)-(3.3) the computational domain is covered by an nx·ny·ns rectangular staggered grid (see [3]). For the approximation of the spatial derivatives, second-order central finite differences are used in both the horizontal and vertical direction.

We use the following notation: **U**, **V** and **Z** are grid functions approximating u, v and ζ, respectively. The **Z**-points are only specified at the sea surface. Furthermore, $A_{\sigma\sigma}$ is a tridiagonal matrix approximating the vertical diffusion term, including the discretization of the term $1/h^2$. Θ_1 is an (nx·ny·ns)·(nx·ny) matrix (a row of ns

diagonal matrices of order nx·ny with $\Delta\sigma_k$ on the diagonal of the k-th submatrix). Θ_2 is an (nx·ny)·(nx·ny·ns) matrix (a column of ns identity matrices of order nx·ny). F is a four-diagonal matrix (due to the grid staggering) of order nx·ny·ns, approximating the Coriolis term. D_x and D_y are bidiagonal matrices (one diagonal and one lower diagonal) of order nx·ny, approximating the differential operators $\partial/\partial x$ and $\partial/\partial y$, respectively. E_x and E_y are bidiagonal matrices (one diagonal and one upper diagonal) with $E_x = -D_x^T$ and $E_y = -D_y^T$. The matrices D_x and E_x differ because of the grid staggering.

Now, the semi-discretized system can be written in the form

$$\frac{d}{dt}W = F(W) = (A + B)\,W + C\,, \quad \text{with } W = \begin{pmatrix} U \\ V \\ Z \end{pmatrix}, \tag{3.4}$$

and

$$A = \begin{pmatrix} \Lambda_{\sigma\sigma} & 0 & 0 \\ -F & \Lambda_{\sigma\sigma} & 0 \\ -\Theta_1 H E_x & -\Theta_1 H D_y & 0 \end{pmatrix}, B = \begin{pmatrix} 0 & F & -\Theta_2 g D_x \\ 0 & 0 & -\Theta_2 g E_y \\ 0 & 0 & 0 \end{pmatrix}, C = \begin{pmatrix} F_u \\ F_v \\ 0 \end{pmatrix}. \tag{3.5}$$

The reason for this splitting will become clear in the next sections. The vector C contains the components of the wind stress. Note that the integrals in (3.3) are approximated by $\Theta_1 U$ and $\Theta_1 V$, respectively.

4.3.2. TIME INTEGRATION

We start with the time integrator for (3.4)-(3.5) developed in [3]:

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & 0 & 0 \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & 0 \\ \tau\Theta_1 H E_x & \tau\Theta_1 H D_y & I \end{pmatrix}\begin{pmatrix} U^{n+1} \\ V^{n+1} \\ Z^{n+1} \end{pmatrix} = \begin{pmatrix} I & \tau F & -\tau\Theta_2 g D_x \\ 0 & I & -\tau\Theta_2 g E_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix} + \tau\begin{pmatrix} F_u^n \\ F_v^n \\ 0 \end{pmatrix},$$

or, equivalently,

$$(I - \tau\,A)\,W^{n+1} = (I + \tau B)\,W^n + \tau\,C^n\,.$$

This method can be written in the form

$$W^{n+1} = W^n + \tau\,(I - \tau A)^{-1}\,F(W^n)\,. \tag{3.6}$$

In terms of method (2.6)-(2.7), we have that $S = (I - \tau A)^{-1}$. Thus, this time integrator can be considered as a method in which the right-hand side function is preconditioned by the implicit smoothing operator $(I - \tau A)^{-1}$. It can easily be seen

that the components are calculated sequentially (first **U**, then **V** and finally **Z**). This is advantageous for both the stability and storage requirements. For the two-dimensional shallow water equations a similar approach has been followed by e.g., Fischer [2] and Sielecki [9]. The time step restriction for method (3.6) is given by

$$\tau < \frac{1}{\sqrt{gh}} \frac{1}{\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}}, \tag{3.7}$$

where Δx and Δy denote the horizontal mesh sizes. This condition is slightly more restrictive than the condition derived in [3]. We remark that the time step in (3.7) does not depend on the vertical mesh size $\Delta\sigma$. However, the condition imposed by the horizontal mesh sizes is still rather restrictive. Therefore, we will add a smoothing operator in the horizontal direction. This smoothing operator will be described in the next section.

4.4. SMOOTHING

In this section the stability of method (3.6) will be improved by a smoothing of general vector functions (see Section 4.2.2).

Method (3.6) can be written in the form

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \tau\,(I - \tau\{A_1+A_2\})^{-1}\,\mathbf{F}(\mathbf{W}^n)\,, \tag{4.1}$$

where $A_1 + A_2 = A$ (see (3.5)) with

$$A_1 = \begin{pmatrix} \Lambda_{\sigma\sigma} & 0 & 0 \\ 0 & \Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } A_2 = \begin{pmatrix} 0 & 0 & 0 \\ -F & 0 & 0 \\ -\Theta_1 HE_x & -\Theta_1 HD_y & 0 \end{pmatrix}.$$

Method (4.1) may therefore be interpreted as the forward Euler method in which the right-hand side function is smoothed by the matrix $(I - \tau\{A_1+A_2\})^{-1}$. The vertical terms are treated implicitly, because the matrix A_1 contains the discretization of the vertical diffusion term. The stability condition for this time integrator does not depend on the vertical mesh size $\Delta\sigma$. However, the condition imposed by the horizontal mesh sizes is still rather restrictive (see (3.7)). The horizontal terms are treated partly implicitly ($A_2\mathbf{W}^{n+1}$ in (4.1)) and partly explicitly ($B\mathbf{W}^n$ in (3.6)). Hence, we add another preconditioning of the right-hand side function, i.e., a smoothing of general vector functions described in Section 4.2.2.

The right-hand side function of the **U**-component only contains derivatives in the x-direction and will therefore be smoothed in the x-direction only. Similarly, the **V**-component is only smoothed in the y-direction. The **Z**-component is smoothed in both directions. However, the smoothing of the right-hand side function in two directions is complicated. The precomputation of the *cheap* factor matrices (see Theorem 2.2) is only feasible in one-dimensional cases. Therefore, we apply one-dimensional smoothing in the x- and y-direction, successively.

In the x-direction the smoothing matrix has the simple structure

$$\begin{pmatrix} S_u & & \\ & 0 & \\ & & S_z \end{pmatrix},$$

where S_u and S_z denote the smoothing matrices for the right-hand side function of the U- and Z-component, respectively. Here, $S_u=P(D_u)$ and $S_z=P(D_z)$ with P(z) defined by (2.10) and

$$D_u^2 = \frac{1}{4}\begin{pmatrix} 0 & & & & 0 \\ 1 & -2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & 1 & -2 & 1 \\ 0 & & & & 0 \end{pmatrix} \quad \text{and} \quad D_z^2 = \frac{1}{4}\begin{pmatrix} -1 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & 1 & -2 & 1 \\ 0 & & & & -1 & 1 \end{pmatrix}. \qquad (4.2)$$

In the y-direction the smoothing matrix has a similar simple structure. Note that D_u and D_z only differ in the first and last row, which is due to the grid staggering and to the boundary conditions. The number of different boundary conditions is very limited (open or closed boundaries, u- or ζ-boundaries). The smoothing matrices, including the values in the first and last row, are therefore computed in advance.

The time integration method in which the smoothing based on general vector function has been added, can be written in the form

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \tau\,(I - \tau\{A_1+SA_2\})^{-1}\,S\,\mathbf{F}(\mathbf{W}^n)\,, \qquad (4.3)$$

with the matrices A_1 and A_2 defined in (4.1) and the smoothing operator S defined in Theorem 2.1. The smoothing operator S appears twice in (4.3). The first operator S is a result of the fact that the components of W are computed sequentially. The second operator S in (4.3) is clearly a smoothing of the right-hand side function. In cases where the solution becomes stationary (thus $\mathbf{F}(\mathbf{W}) = \mathbf{0}$), it is evident that methods (4.1) and (4.3) obtain the same stationary solution.

The stability condition for method (4.3) reads (see (2.13) and (3.7))

$$\tau < \frac{\pi k}{2}\,\frac{1}{\sqrt{gh}}\,\frac{1}{\sqrt{\dfrac{1}{(\Delta x)^2}+\dfrac{1}{(\Delta y)^2}}}. \qquad (4.4)$$

Hence, the gain factor obtained by the smoothing of general vector functions is $\pi k/2$.

4.5. Implementation of the smoothing operators

In this section we discuss the implementation of the smoothing matrices $(I - \tau A)^{-1}$ and S (see (4.3)). For the U- and V-component the smoothing matrix $(I - \tau A)^{-1}$ requires the solution of nx·ny tridiagonal systems of order ns, which can be computed efficiently [3]. The smoothing operator S can be computed in various ways. The most efficient implementation is based on the factorization property presented in Theorem 2.2. If the factor matrices of (2.14) are computed in advance, then the evaluation of P(D) only requires q (= $^2\log(k)$) matrix–vector operations.

For example, applying Theorem 2.2 for matrix D_u^2 (see (4.2)), we find the factor matrices

$$F_1 = \frac{1}{4}\begin{pmatrix} 4 & & & & 0 \\ 1 & 2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & 1 & 2 & 1 \\ 0 & & & & & 4 \end{pmatrix}, \qquad F_2 = \frac{1}{4}\begin{pmatrix} 4 & & & & & 0 \\ 2 & 1 & 0 & 1 & & \\ 1 & 0 & 2 & 0 & 1 & \\ & 1 & 0 & 2 & 0 & 1 \\ & & \cdot & \cdot & \cdot & \cdot \end{pmatrix}, \text{ etc.}$$

Evidently, the matrix–vector multiplications with these essentially tridiagonal factor matrices are extremely cheap, especially on vector computers. For example, on the CDC CYBER 205 the operations can be performed in two linked triad instructions (except near the boundaries).

Since the smoothing operator S is applied in the x- and y-direction, successively, it consists of a sequence of one-dimensional operators. Therefore, the smoothing operator can be implemented on irregular domains too. However, the bandwidth of the factor matrices F_q is 2^q+1. In practice, the value of q is at most five. In experiments with irregular domains it might happen that there are not enough grid points in the x- or y-direction. In this case we apply the implicit smoothing operator defined in Theorem 2.3 with D^2 as in (4.2), instead of the explicit smoothing operator. Thus, the application of the smoothing operator is hardly complicated when the domain is irregular.

The implicit smoothing operator requires the solution of a small tridiagonal system with a dimension of at most 2^q. For the solution of the tridiagonal systems we use the Gaussian Elimination method. Since these systems are small and the implicit smoothing is only applied in narrow regions (where the number of grid points is less than or equal to 2^q), the computation time for the sequential Gaussian Elimination method is very limited, also on vector and parallel computers.

4.6. Numerical experiments

In this section we show for a number of test problems (see [1,3]) the effects of smoothing on the stability and on the accuracy. In the test problems the water is initially at rest and the motion in the basin is generated by a wind stress. Thus, a wind driven circulation is gradually developed. We carry out two experiments with a constant wind stress and one with a time-dependent wind stress. In the experiments with a constant wind stress we use a rectangular basin with dimensions representative of the North Sea and an irregular basin representing the IJsselmeer. The IJsselmeer is the largest lake in the Netherlands.

The following parameter values are used in all experiments:

$f = 1.22e\text{-}4\ s^{-1}$
$g = 9.81\ m/s^2$
$\mu = 0.065\ m^2/s$
$C = 70\ m^{1/2}/s$
$\rho = 1025\ kg/m^3$
$\phi = 45°$.

For the time integration we use method (4.3). In the experiments we vary the number of smoothing factors in the factorized smoothing operator (see (2.14)) to investigate the effects of smoothing on the stability and on the accuracy. The experiments have been carried out on an Alliant FX/4. This mini-supercomputer is equipped with four vector processors. On such a computer we can investigate the effect of the smoothing on both vector-parallel computers and scalar computers.

To represent the results, we use the following notation:

q	: number of smoothing factors
ERROR-ζ	: maximal global error for the water elevation at the end point t=T
$TOT_{VP/S}$	: total computation time
$SMO_{VP/S}$	: computation time for the smoothing operator

The indices VP and S indicate Vector–Parallel optimization and Scalar optimization, respectively. Thus, the experiment is carried out on one processor if only the scalar optimization is used. At the end of the integration process the numerical solution for the ζ-component is compared with a reference solution computed on the same grid with τ=30 s. The reference solution may be considered as an almost exact solution of our semi-discretized system (3.4). Thus, the accuracy results listed in this section represent the error due to the time integration. We experimentally determined the maximally stable time step for each value of q. These time steps are in agreement with (4.4) (see Table 6.2).

In the first two experiments we use a rectangular basin of 400 by 800 km with Δx=10 km, Δy=10 km, $\Delta\sigma$=0.25 and h=65 m. Thus, the computations are performed on a grid with nx=41, ny=81 and ns=4.

In the first experiment we integrate over a period of five days with the constant wind stress

$$W_f = 1.5\ kg/ms^2 . \tag{6.1}$$

At that time, the steady state has already been reached.

In this experiment the maximal value for the water elevation is about 1.07 m. The results show that the time integration can be performed with much larger time steps when the smoothing technique is applied. In this experiment, in which the solution becomes stationary, the accuracy is hardly reduced by the smoothing procedure. Only for large q some errors occur. This is due to the fact that for these values of q the steady state has not been reached yet. If the time integration is performed over a longer period, we obtain the same results for large values of q as

for the case q=0. This is in agreement with the theory that a stationary solution should be independent of the number of smoothing factors (see Section 4.4).

q	τ (s)	ERROR-ζ (m)	TOT_{VP} (s)	SMO_{VP} (s)	TOT_S (s)	SMO_S (s)
0	270	0.001	345.0	0.0	2600.1	0.0
1	800	0.002	173.1	20.5	1463.8	243.1
2	1800	0.008	85.6	17.6	767.3	219.7
3	3600	0.022	46.9	12.8	446.3	169.6
4	7200	0.055	25.9	8.7	255.8	116.8
5	14400	0.152	13.7	5.1	142.8	75.0

Table 6.1. Test problem with a constant wind stress.

In Table 6.2 we list the gain factors of the maximally stable time steps compared with the case q=0 ($\tau_{max} \approx 277$ s), and we compare them with the theoretical gain factors. Moreover, we list the gain factors in computation times.

	q = 1	q = 2	q = 3	q = 4	q = 5
theoretically (= $2^{q-1}\pi$ for q > 0)	3.1	6.3	12.6	25.1	50.3
experimentally (see Table 6.1)	2.9	6.5	13.0	25.9	51.9
in computation time (VP)	2.0	4.0	7.4	13.3	25.2
in computation time (S)	1.8	3.4	5.8	10.2	18.2

Table 6.2. Gain factors.

The theoretical gain factor $2^{q-1}\pi$ (see (4.4) with $k=2^q$) is in agreement with the experimental results. The results show a significant reduction in computation time, especially when the vector and parallel optimization is used. The overhead due to the smoothing operator is less than a factor of two, even for large values of q. In the case of the vector and parallel optimization, the computation time is reduced by about a factor of three due to the vectorization, and by an additional factor of three due to the parallel optimization.

It is interesting to investigate the effect of smoothing when the solution of a test problem does not become stationary. Therefore, in the second experiment we introduce a time-dependent wind stress (cf. (6.1))

$$W_f = 1.5 * (1 + 0.5 * \sin \frac{2\pi t}{24 * 3600}) \text{ kg/ms}^2 . \qquad (6.2)$$

Now, we have a periodically varying wind with a period of 24 hours. We integrate over a period of five days. At that time the solution is almost periodic. In the case without smoothing in the horizontal we obtain the following maximal water elevations at the south-west corner of the basin:

$\zeta = 2.106$ m at $t = 7.3 + iP$ hours

with period P = 24 hours and i a positive integer. When smoothing is applied we have observed that about the same maximal and minimal water elevations are reached as in the case without smoothing in the horizontal. It seems that the smoothing operator hardly introduces a dissipation error. However, some errors in the phase of the periodic solution appear. In Table 6.3 we list the maximal global error in the numerical solution for the water elevation measured at the end point T=120 hours compared with a reference solution computed with τ=30 s.

q	τ (s)	ERROR-ζ (m)	τ (s)	ERROR-ζ (m)
0			270	0.008
1	270	0.008	720	0.014
2	270	0.023	1800	0.052
3	270	0.067	3600	0.139
4	270	0.193	7200	0.463

Table 6.3. Test problem with a time-dependent wind stress.

The results show that the error due to the smoothing operator is even smaller than the error due to the larger time steps. For example, in the case q=2 the error due to the larger time steps (i.e., 0.029 m.) is larger than the error due to the smoothing (i.e., $\leq$0.023 m). Thus, when a fully implicit method had been used, the accuracy would also decrease for large time steps.

In the third experiment we investigate the efficiency of the smoothing operators on an irregular basin, i.e., a geometry of the IJsselmeer. Figure 1 shows the geometry of the IJsselmeer used in this experiment. We choose $\Delta x=\Delta y$=1.0 km and h=6.5 m. The IJsselmeer is represented by about 1100 grid points in the horizontal direction. The vertical representation is made by five layers of the same depth. We integrate over a period of one day with the same constant wind stress as in the first experiment (see (6.1)). At the end point T=24 hours we compare the numerical solution for the ζ-component with a reference solution computed with τ=10 s.

Without smoothing the maximally stable time step is about 87 s. In Table 6.4 we list the results.

q	τ (s)	ERROR-ζ (m)	TOT_{VP} (s)	SMO_{VP} (s)
0	80	0.000	200.3	0.0
1	80	0.001	280.4	22.1
2	80	0.009	305.4	48.9
3	80	0.021	339.4	77.5
1	270	0.001	86.0	6.6
2	600	0.009	44.4	9.4
3	1200	0.030	24.8	6.9

Table 6.4. IJsselmeer problem with a constant wind stress.

In this experiment the maximal value for the water elevation is about 0.79 m. The results in this experiment are comparable with the results on a rectangular domain (see Table 6.1). The accuracy is hardly reduced by the smoothing procedure. Moreover, the overhead due to the smoothing operator is even less than in the experiment with a rectangular domain. This is due to the fact that the smoothing matrix has been computed on the irregular domain representing the IJsselmeer, whereas the computations that do not involve the smoothing, have been performed on a surrounding rectangular domain. On vector and parallel computers this is in general an efficient approach, because direct addressing can be used in most cases. The efficiency depends on the number of dummy grid points in the surrounding rectangle compared with the number of grid points in the irregular (physical) domain. However, regardless of the implementation used, it may be concluded that the smoothing operator can be implemented efficiently on both regular and irregular domains.

4.7. Conclusions

In this paper we have applied right-hand side smoothing to improve the stability of a time integrator for the linearized 3D shallow water equations. We started with the semi-implicit time integrator developed in [3]. It turns out that this method may be considered as a method in which the right-hand side function is premultiplied by an implicit smoothing operator. The vertical terms are treated implicitly. Since the number of points in vertical direction may be very small, explicit smoothing can not be applied. Moreover, the stability condition imposed by the vertical terms is often the most restrictive one. Therefore, we prefer an implicit treatment of the vertical terms.

In the horizontal direction we may choose between explicit and implicit smoothing of vector functions. In this paper we have applied explicit smoothing whenever possible. Only in cases where explicit smoothing can not be applied (i.e., in narrow regions), we have used implicit smoothing. It turns out that this approach is efficient, especially on vector and parallel computers.

Owing to the smoothing in the horizontal direction, the maximally stable time step increases considerably, while the accuracy decreases only slightly. In our wind driven test problems the maximally stable time step increases by a factor of more than 10 (in the case q=3), while the accuracy is still acceptable. In this case the overhead in computation time due to the smoothing is only about 30%. Moreover, the error due to the large time steps is more or less comparable with the error introduced by the smoothing. Thus, also for fully implicit methods the accuracy will decrease for such large time steps.

REFERENCES

1. A.M. DAVIES, Application of the DuFort-Frankel and Saul'ev methods with time splitting to the formulation of a three dimensional hydrodynamic sea model, *Int. J. Numer. Meth. in Fluids*, 5, 405-425 (1985).
2. G. FISCHER, Ein numerisch verfahren zur errechnung von windstau und gezeiten in randmeeren, *Tellus*, 11, 60-76 (1959).
3. E.D. DE GOEDE, *Finite difference methods for the three-dimensional hydrodynamic equations*, Report NM-R8813, CWI, Amsterdam, 1988.
4. P.J. VAN DER HOUWEN, Stabilization of explicit difference schemes by smoothing techniques , *in K. Strehmel (ed.): Numerical Treatment of Differential Equations, (Proc. Fourth Seminar Halle: NUMDIFF-4), Teubner-Texte zur Mathematik 104*, BSB B.G. Teubner Verlaggesellschaft, Leipzig, 205-215 (1987).
5. P.J. VAN DER HOUWEN, *Construction of integration formulas for initial value problems*, North-Holland, Amsterdam, 1977.
6. P.J. VAN DER HOUWEN, C. BOON AND F.W. WUBS, Analysis of smoothing matrices for the preconditioning of elliptic difference equations, *Z. Angew. Math. Mech.*, 68, 3-10 (1988).
7. A. JAMESON, The evolution of computational methods in aerodynamics, *J. Appl Mech.*, 50, 1052-1076 (1983).
8. R.D. RICHTMYER AND K.W. MORTON, *Difference methods for initial value problems*, Interscience Publishers, Wiley, New York, London, 1967.
9. A. SIELECKI, An energy conserving difference scheme for storm surge equations, *Monthly Weather Review*, 96, 150-156 (1968).
10. F.W. WUBS, Stabilization of explicit methods for hyperbolic partial differential equations, *Int. J. Numer. Meth. in Fluids*, 6, 641-657 (1986).

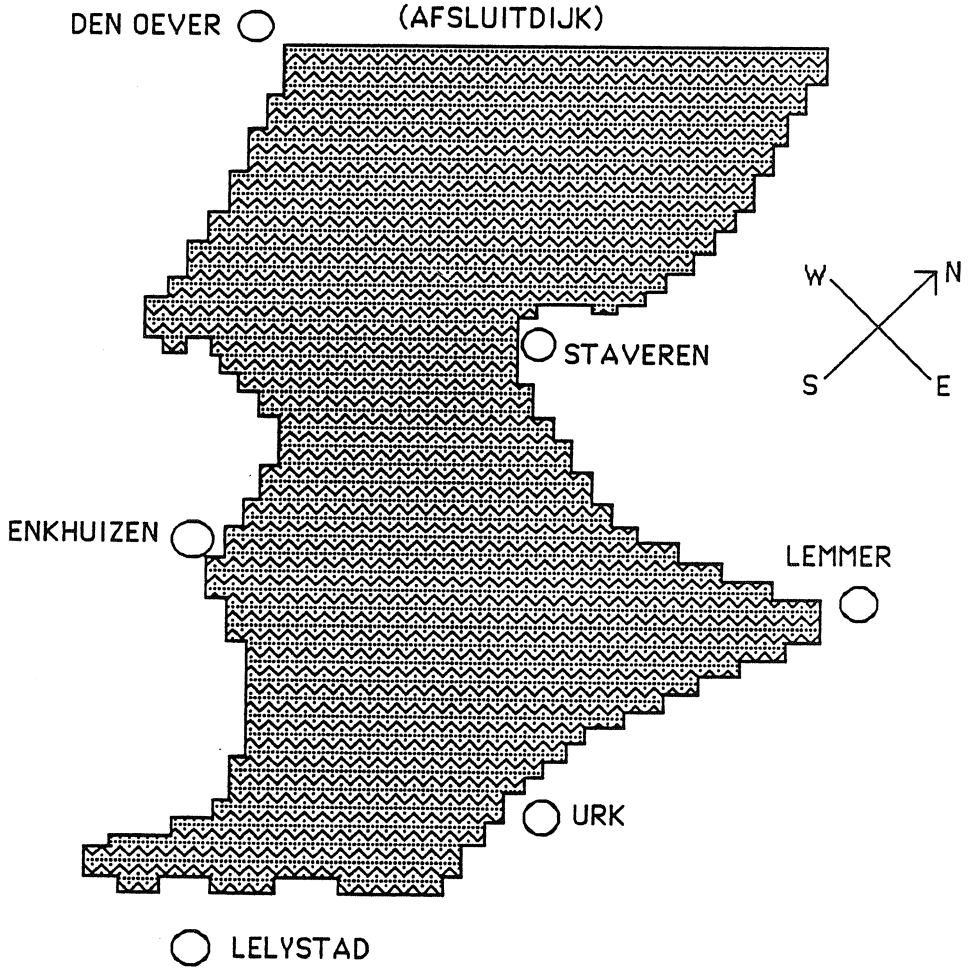

Fig. 1. The geometry of the IJsselmeer.

Chapter 5

A Time Splitting Method for the Three-Dimensional Shallow Water Equations

E.D. de Goede
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB Amsterdam, The Netherlands

In this paper we describe a time splitting method for the three-dimensional shallow water equations. The stability of this method neither depends on the vertical diffusion term nor on the terms describing the propagation of the surface waves. The method consists of two stages and requires the solution of a sequence of linear systems. For the solution of these systems we apply a Jacobi-type iteration method and a conjugate gradient iteration method. The performance of both methods is accelerated by a technique based on smoothing. The resulting method is mass conservative and efficient on vector and parallel computers. The accuracy, stability and computational efficiency of this method are demonstrated for wind induced problems in a rectangular basin.

5.1. INTRODUCTION

In this paper a time splitting method for the three-dimensional shallow water equations (SWEs) will be described. The aim of splitting methods is always to split the solution of a large and complicated system, which arises when applying fully implicit methods to multi-dimensional problems, into a few less complicated systems. Well-known splitting methods are alternating direction implicit (ADI) methods, locally one-dimensional (LOD) methods and Hopscotch methods [6].

For the two-dimensional shallow water equations several of the existing numerical methods have been based on the ADI method (see e.g., [10,12]). These ADI methods are unconditionally stable and therefore allow the use of large time steps. However, for large time steps these methods suffer from inaccuracies when dealing with complex geometries [13]. In [15] a two-stage time splitting method has been developed in which these inaccuracies are absent, even for large time steps.

In this paper we will present a two-stage time splitting method for the three-dimensional shallow water equations which has a strong resemblance to the method in [15]. We will use a model for the shallow water equations in which the advective terms have been omitted. The stability of a numerical method for this model depends on the conditions imposed by the vertical diffusion term and by the terms describing the propagation of the surface waves (the CFL condition). In two-dimensional models many methods are known in which the terms describing the propagation of the surface waves are treated implicitly (see e.g., [1,2,15]). In

addition to that, in three-dimensional models, where a vertical diffusion term is involved, we have to treat this vertical diffusion term implicitly to avoid the maximally stable time step becoming too small [3]. In this paper we will develop a two-stage method in which the vertical diffusion is treated implicitly at the first stage, whereas the terms concerning the propagation of surface waves are treated implicitly at the second stage. It will be shown that the stability of this time splitting method neither depends on the vertical diffusion nor on the propagation of the surface waves. For computational efficiency the Coriolis term will be treated in a semi-implicit way. The Coriolis term hardly affects the stability, which justifies this simplification.

At the first stage our time integration method requires the solution of a large number of tridiagonal systems, all of the same dimension. Since the tridiagonal systems are independent of each other, the solution of these systems can be computed in parallel [4].

At the second stage a linearization process is used to iteratively solve the nonlinear system. The linearization is done in such a way that conservation of mass remains guaranteed. Then at each iteration step a linear, symmetric, positive definite system has to be solved. In the literature a large number of iteration methods have been proposed for such systems. In this paper we apply a Jacobi-type iteration method and a conjugate gradient iteration method for the solution of this system. Both iteration methods will be accelerated by a technique based on smoothing. Application of the smoothing matrices reduces the number of iterations considerably. Moreover, the smoothing matrices are very simple to implement and are highly suited for vector and parallel computers.

In [15] a two-stage time splitting method has been developed for the two-dimensional shallow water equations. It was reported that this time splitting method is feasible for practical computations. For this method a major part of the computation is involved in the nonlinear system at the second stage. Since in our time splitting method the water elevation is the only unknown in the system at the second stage, this system is of the same (two-dimensional) structure and thus of the same computational complexity for both two-dimensional and three-dimensional models. The computation time required by the other parts of our method, i.e., the computation of the three-dimensional velocity components, is proportional to the number of grid layers in the vertical direction. Therefore, the efficiency of the time splitting method developed in this paper is even higher for three-dimensional models than for two-dimensional models.

The accuracy, stability and computational efficiency of our time splitting method will be illustrated in the numerical experiments.

5.2. Mathematical model

In this section we will describe a mathematical model for the three-dimensional shallow water equations. We will use a three-dimensional model in sigma co-ordinates in which the advective terms have been omitted. In this paper we focus on stability conditions imposed by the vertical diffusion term and by the terms describing the propagation of the surface waves. In future we will develop a numerical method for a mathematical model in which the advective terms are present.

The mathematical model used in this paper is described by

$$\frac{\partial u}{\partial t} = fv - g\frac{\partial \zeta}{\partial x} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial u}{\partial \sigma}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -fu - g\frac{\partial \zeta}{\partial y} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial v}{\partial \sigma}\right) \tag{2.2}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(h\int_0^1 u d\sigma\right) - \frac{\partial}{\partial y}\left(h\int_0^1 v d\sigma\right), \tag{2.3}$$

with boundaries

$$0 \le x \le L$$
$$0 \le y \le B$$
$$1 \ge \sigma \ge 0 .$$

Thus, the domain is a rectangular basin. Owing to the sigma transformation in the vertical, the domain is constant in time [3,5]. We have the closed boundary conditions

$$u(0,y,\sigma,t) = 0, \quad u(L,y,\sigma,t) = 0,$$
$$v(x,0,\sigma,t) = 0, \quad v(x,B,\sigma,t) = 0.$$

The boundary conditions at the sea surface ($\sigma = 0$) are given by

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \cos(\phi), \quad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \sin(\phi),$$

and at the bottom ($\sigma = 1$) we have a linear law of bottom friction of the form

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g\, u_d}{C^2}, \qquad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g\, v_d}{C^2},$$

with u_d and v_d the components of the velocity at some depth near the bottom.

5.3. SPACE DISCRETIZATION

For the space discretization of the equations (2.1)-(2.3) the computational domain is covered by an nx·ny·ns rectangular staggered grid. Figure 1 shows the horizontal grid spacing. Owing to the sigma transformation, we have a constant number of grid layers in the vertical direction. In what follows, U(t) is a grid function whose components $U_{i,j,k}(t)$ approximate the velocity u(t). The components $U_{i,j,k}(t)$ are numbered lexicographically. Likewise V, Z, D and H are grid functions for v, ζ, d and h, respectively. Note that D, H and Z are only computed at the upper layer.

Furthermore, $\Lambda_{\sigma\sigma}$ is a tridiagonal matrix approximating the vertical diffusion term, including the discretization of the term $1/h^2$. Θ_1 is an (nx·ny·ns)·(nx·ny) matrix (a row of ns diagonal matrices of order nx·ny with $\Delta\sigma_k$ on the diagonal of the k-th submatrix), Θ_2 is an (nx·ny)·(nx·ny·ns) matrix (a column of ns identity matrices of order nx·ny). F is a four-diagonal matrix (due to the grid staggering) of order nx·ny·ns, approximating the Coriolis term. D_x and D_y are bidiagonal matrices (one diagonal and one lower diagonal) of order nx·ny, approximating the differential operators $\partial/\partial x$ and $\partial/\partial y$, respectively. E_x and E_y are bidiagonal matrices (one diagonal and one upper diagonal) with $E_x = -D_x^T$ and $E_y = -D_y^T$. The matrices D_x and E_x differ because of the grid staggering.

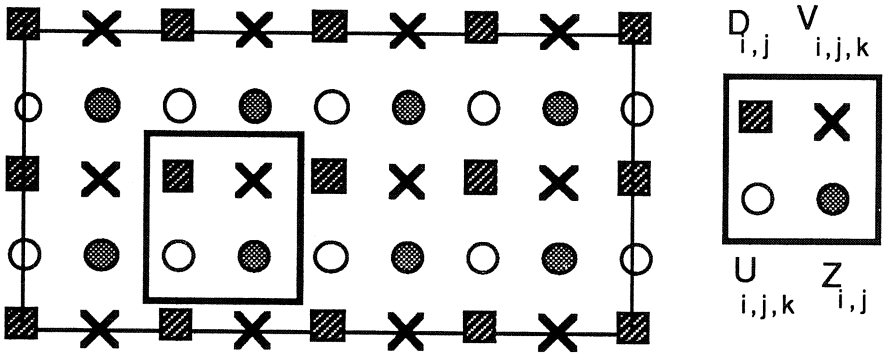

Figure 1. The staggered grid in the (x,y)-plane.

For the approximation of the spatial derivatives, second-order central finite differences are used in both the horizontal and vertical direction. Now, the semi-discretized system can be written in the form

$$\frac{d}{dt}W = F(W) = \begin{pmatrix} \Lambda_{\sigma\sigma} & F & -\Theta_2 g D_x \\ -F & \Lambda_{\sigma\sigma} & -\Theta_2 g E_y \\ -\Theta_1 H E_x & -\Theta_1 H D_y & 0 \end{pmatrix} W + \begin{pmatrix} F_u \\ F_v \\ 0 \end{pmatrix}, \qquad (3.1)$$

where $W=(U,V,Z)^T$ and $(F_u,F_v,0)^T$ contains the components of the wind stress. Note that the integrals in (2.3) are approximated by $\Theta_1 U$ and $\Theta_1 V$, respectively.

5.4. TIME INTEGRATION

In this section we develop a time integration method for the semi-discretized system (3.1). We apply a two-stage time splitting method of the form

$$\begin{aligned} W^{n+1/2} &= W^n + \tfrac{1}{2}\tau \{ F^1(W^{n+1/2}) + G^1(W^n) + C^{n+1/2} \} \\ W^{n+1} &= W^{n+1/2} + \tfrac{1}{2}\tau \{ F^2(W^{n+1/2}) + G^2(W^{n+1}) + C^{n+1/2} \}, \end{aligned} \qquad (4.1)$$

where $C=(F_u,F_v,0)^T$, τ denotes the time step and W^n is a numerical approximation to $W(t)$ of (3.1) at $t=n\tau$. Several well-known splitting methods, e.g., ADI methods, can be written in this form. In this paper we choose

$$F^1(W^{n+1/2}) = \begin{pmatrix} \Lambda_{\sigma\sigma} & 0 & 0 \\ -F & \Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & 0 \end{pmatrix} W^{n+1/2}$$

$$G^1(W^n) = \begin{pmatrix} 0 & F & -\Theta_2 g D_x \\ 0 & 0 & -\Theta_2 g E_y \\ -\Theta_1 H^n E_x & -\Theta_1 H^n D_y & 0 \end{pmatrix} W^n , \qquad (4.2)$$

$$F^2(W^{n+1/2}) = \begin{pmatrix} \Lambda_{\sigma\sigma} & F & 0 \\ -F & \Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & 0 \end{pmatrix} W^{n+1/2} ,$$

$$G^2(W^{n+1}) = \begin{pmatrix} 0 & 0 & -\Theta_2 g D_x \\ 0 & 0 & -\Theta_2 g E_y \\ -\Theta_1 H^{n+1} E_x & -\Theta_1 H^{n+1} D_y & 0 \end{pmatrix} W^{n+1} .$$

Apart from the Coriolis term F, all terms are treated in a symmetrical way. When we neglect the Coriolis term, the time splitting method (4.1)-(4.2) is second-order accurate in time.

The structure of the resulting systems at both stages determines the efficiency of this time splitting method. At the first stage we have to solve the system

$$\begin{pmatrix} I - \frac{1}{2}\tau\Lambda_{\sigma\sigma} & 0 & 0 \\ \frac{1}{2}\tau F & I - \frac{1}{2}\tau\Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} U^{n+1/2} \\ V^{n+1/2} \\ Z^{n+1/2} \end{pmatrix} = B^n , \qquad (4.3)$$

where B^n contains the discretizations at time level $t=n\tau$. It is evident that the Z-component can be computed straightforwardly. For the U- and V-component the implicit treatment of the vertical diffusion term requires the solution of $nx \cdot ny$ tridiagonal systems of order ns [4]. For computational efficiency the Coriolis term is treated in a semi-implicit way. Although an implicit treatment of the Coriolis term for the V-component (see (4.3)) prohibits the U- and V-component from being computed in parallel, we prefer this choice because of accuracy considerations. The results are more accurate than in the case of a fully explicit treatment of the Coriolis term, especially when large time steps are used.

At the second stage the terms describing the propagation of the surface waves are treated implicitly. This system reads

$$\begin{pmatrix} I & 0 & \frac{1}{2}\tau\Theta_2 g D_x \\ 0 & I & \frac{1}{2}\tau\Theta_2 g E_y \\ \frac{1}{2}\tau\Theta_1 H^{n+1} E_x & \frac{1}{2}\tau\Theta_1 H^{n+1} D_y & I \end{pmatrix} \begin{pmatrix} U^{n+1} \\ V^{n+1} \\ Z^{n+1} \end{pmatrix} = B^{n+1/2}, \quad (4.4)$$

where $B^{n+1/2}$ contains the discretizations at time level $t=(n+1/2)\tau$. The equations for the U- and V-component are linear and are not coupled with each other. They are only coupled with the equation for the Z-component. Therefore, the components U^{n+1} and V^{n+1} can easily be eliminated from (4.4) and a system merely in the unknown Z^{n+1} results. Thus, at the second stage the continuity equation (2.3) and the water elevation gradient in the momentum equations (2.1)-(2.2) are treated implicitly. This approach was originally proposed in [8] and has been applied by many others (e.g., [1,2,15]).

We now describe this system for each cell (i,j) of component Z. The grid sizes in the x- and y-direction are denoted by Δx and Δy, respectively. Then, the system for $Z_{i,j}$ reads

$$\begin{aligned} & Z_{i,j}^{n+1} - \frac{\tau^2 g}{4(\Delta x)^2} \left\{ \bar{H}_{i+1,j}^{n+1} \left(Z_{i+1,j}^{n+1} - Z_{i,j}^{n+1} \right) - \bar{H}_{i,j}^{n+1} \left(Z_{i,j}^{n+1} - Z_{i-1,j}^{n+1} \right) \right\} \\ & - \frac{\tau^2 g}{4(\Delta y)^2} \left\{ \tilde{H}_{i,j}^{n+1} \left(Z_{i,j+1}^{n+1} - Z_{i,j}^{n+1} \right) - \tilde{H}_{i,j-1}^{n+1} \left(Z_{i,j}^{n+1} - Z_{i,j-1}^{n+1} \right) \right\} \quad (4.5) \\ & = B_{i,j}^{n+1/2}, \quad \text{for} \quad \begin{matrix} i=1,\ldots,nx \\ j=1,\ldots,ny \end{matrix}, \end{aligned}$$

where

$$\bar{H}_{i,j}^{n} = Z_{i,j}^{n} + \frac{1}{2}\left(D_{i,j} + D_{i,j-1} \right) \quad \text{and} \quad \tilde{H}_{i,j}^{n} = Z_{i,j}^{n} + \frac{1}{2}\left(D_{i,j} + D_{i+1,j} \right).$$

Note that $\bar{H}$ and $\tilde{H}$ differ because of the grid staggering. System (4.5) is a nonlinear equation, because $H_{i,j}$ contains the component $Z_{i,j}$. When system (4.5) has been solved, the values for the components U^{n+1} and V^{n+1} can be computed by back substitution.

System (4.5) can be written in the form

$$A(Z^{n+1})\, Z^{n+1} = B_Z^{n+1/2}, \quad (4.6)$$

where $B_Z^{n+1/2}$ contains the discretizations at $t=(n+1/2)\tau$ for the Z-component. For its linearization we introduce the process

$$A(Z^{(m)})\, Z^{(m+1)} = B_Z^{n+1/2}, \quad (4.7)$$

with $Z^{(0)}=Z^{n+1/2}$. In (4.7) the upper index (m) denotes the iteration index. The matrix $A(Z^{(m)})$ is a symmetric and strictly diagonal dominant matrix with positive

values on the main diagonal and negative ones elsewhere because we require that $D+Z^{(m)}$ $(=H^{(m)})$ >0. Thus, system (4.7) is positive definite. In Section 5.6 we will discuss iteration methods for the solution of system (4.7).

It should be noted that the water elevation is the only unknown in system (4.7). Thus, this system is of the same (two-dimensional) structure and computational complexity for both two-dimensional and three-dimensional models. This is an important feature of the time integration method (4.1)-(4.2), because for two-dimensional problems a major part of the computation is required for the solution of this system.

The linearization process (4.7) was first used by Leendertse [10]. Conservation of mass remains guaranteed by this process. A slightly different linearization process was introduced in [15]. In our numerical experiments (see Section 7) we obtained comparable results for both linearization processes.

5.5. Stability

We will now analyze the stability of method (4.1)-(4.2) with the matrix method. In this section we will omit the Coriolis force and the inhomogeneous term. It is well-known that the Coriolis force hardly affects the stability. We will make plausible that the simplified method is unconditionally stable. The stability analysis used here is similar to the one described in [14]. That paper was devoted to a study of the stability and convergence properties of the Peaceman-Rachford ADI method when applied to initial-boundary value problems, including nonlinear ones.

Since we have omitted the Coriolis force and the inhomogeneous term, we have that (cf. (4.2))

$$F^1(W^n) = F^2(W^n) = A^n W^n \quad \text{and} \quad G^1(W^n) = G^2(W^n) = B^n W^n ,$$

with

$$A^n = \begin{pmatrix} \Lambda_{\infty} & 0 & 0 \\ 0 & \Lambda_{\infty} & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } B^n = \begin{pmatrix} 0 & 0 & -\Theta_2 g D_x \\ 0 & 0 & -\Theta_2 g E_y \\ -\Theta_1 H^n E_x & -\Theta_1 H^n D_y & 0 \end{pmatrix}.$$

Then, method (4.1)-(4.2) can be written in the form

$$W^{n+1} = (I - \tfrac{1}{2}\tau B^{n+1})^{-1} (I + \tfrac{1}{2}\tau A^{n+1/2}) (I - \tfrac{1}{2}\tau A^{n+1/2})^{-1} (I + \tfrac{1}{2}\tau B^n) W^n. \quad (5.1)$$

Let us now define the amplification matrix

$$C^n = (I - \tfrac{1}{2}\tau B^{n+1})^{-1} (I + \tfrac{1}{2}\tau A^{n+1/2}) (I - \tfrac{1}{2}\tau A^{n+1/2})^{-1} (I + \tfrac{1}{2}\tau B^n) .$$

In order to guarantee that method (5.1) is stable we have to require $\| \prod_{i=0}^{n-1} C^i \|$ to remain uniformly bounded for all values of n and τ, such that

$$\| \prod_{i=0}^{n-1} C^i \| < K \,, \quad \text{for } 0 < n\tau < T \text{ and } K \text{ constant [9].} \tag{5.2}$$

Let us now verify this condition for the numerical method (5.1). Then,

$$\| \prod_{i=0}^{n-1} C^i \| \le \| (I - \tfrac{1}{2}\tau B^n)^{-1} \| \; \| (I + \tfrac{1}{2}\tau A^{n-1/2}) (I - \tfrac{1}{2}\tau A^{n-1/2})^{-1} \| \prod_{i=0}^{n-2} \| (I + \tfrac{1}{2}\tau B^{i+1}) (I - \tfrac{1}{2}\tau B^{i+1})^{-1} (I + \tfrac{1}{2}\tau A^{i+1/2}) (I - \tfrac{1}{2}\tau A^{i+1/2})^{-1} \| \| I + \tfrac{1}{2}\tau B^0 \| .$$

It can be verified that both the matrix A^n and the matrix B^n have their eigenvalues in the left half plane. The eigenvalues of the matrix A^n are even real nonpositive. Therefore, we have that

$$\| (I + \tfrac{1}{2}\tau B^{i+1}) (I - \tfrac{1}{2}\tau B^{i+1})^{-1} \| \le 1 \quad \text{and}$$

$$\| (I + \tfrac{1}{2}\tau A^{i+1/2}) (I - \tfrac{1}{2}\tau A^{i+1/2})^{-1} \| \le 1 \,, \quad \text{for } i=0,\ldots,n-1 \,.$$

Using these relations, we obtain

$$\| \prod_{i=0}^{n-1} C^i \| \le \| (I - \tfrac{1}{2}\tau B^n)^{-1} (I + \tfrac{1}{2}\tau B^0) \| .$$

It is evident that only the explicit part $(I + \frac{1}{2}\tau B^0)$ may cause problems. In general, it is not possible to find an upper bound for $\| (I + \frac{1}{2}\tau B^0) \|$. In such a situation we may stabilize our integration method by computing the first approximation W^1 by the backward Euler-LOD method, and apply method (5.1) for $n \ge 1$. This technique has been proposed in [14]. We thus consider the method with for the first time step

$$W^1 = (I - \tfrac{1}{2}\tau B^1)^{-1} (I - \tfrac{1}{2}\tau A^{1/2})^{-1} W^0 \,, \tag{5.3a}$$

and for $n \ge 1$

$$W^{n+1} = (I - \tfrac{1}{2}\tau B^{n+1})^{-1} (I + \tfrac{1}{2}\tau A^{n+1/2}) (I - \tfrac{1}{2}\tau A^{n+1/2})^{-1} (I + \tfrac{1}{2}\tau B^n) W^n . \tag{5.3b}$$

On a fixed space grid the LOD method (5.3a) is only first-order accurate in time, but since we only perform one LOD step, method (5.3) is still second-order accurate on fixed space grids. Using method (5.3), we obtain

$$\| \prod_{i=0}^{n-1} C^i \| \le \| (I - \tfrac{1}{2}\tau B^n)^{-1} \| \; \| (I + \tfrac{1}{2}\tau A^{1/2})^{-1} \| .$$

Now, condition (5.2) is satisfied. We have no practical experience with method (5.3). In the numerical experiments no large errors were found for the

original method (5.1), even for very large time steps. Thus, there was no need for stabilization. In [14] the authors advise the use of one or more LOD steps in situations where the initial values contain large errors. This might occur when experimental data with significant errors are used as initial values.

5.6. SOLVING THE LINEAR SYSTEMS

In this section we describe how the linear systems at both stages, i.e., system (4.3) and system (4.7), are solved. At the first stage we apply the Gaussian Elimination (double sweep) method for the solution of the tridiagonal systems. Since this is a recursive method, it is an unattractive method on vector and parallel computers. However, we make use of the fact that a large number of tridiagonal systems of the same dimension have to be solved. In [4] the computational efficiency of this approach has been demonstrated on vector and parallel computers. Moreover, this method requires a minimal number of operations.

At the second stage we have to solve the linear, symmetric system (4.7). In the literature a large number of iteration methods have been proposed for such systems. Here, we will apply a Jacobi-type method and a conjugate gradient (CG) method. Both methods will be accelerated by a preconditioning technique. Before discussing the iteration methods, we first consider the preconditioning.

The essence of preconditioning is the determination of a matrix S such that the system

$$SA\,Z^{(m+1)} = SB\,,$$

has a much smaller condition number than the original system $AZ^{(m+1)}=B$. For the preconditioning of system (4.7) we will use a smoothing matrix S of the form S=P(D) where P(z) is a polynomial and D is some matrix. The matrix D will be a difference matrix of which the eigenvalues are assumed to be in the interval [–1,0] (see [7]). The polynomial P(z) will be chosen such that P(0)=1 and the eigenvalues of S are in the interval [0,1]. First we discuss the choice of the matrix D. In our *theoretical* considerations we assume that D is equal to the normalized matrix A, i.e.,

$$D = \frac{A}{\rho(A)}\,, \tag{6.1}$$

where $\rho(\cdot)$ denotes the spectral radius. We emphasize that in *practice* it is generally not attractive to choose D according to (6.1) and we shall employ some cheap approximation to the normalized matrix. If D is defined according to (6.1), then the eigenvalues of SA=P(D)A are given by $\rho(A)zP(z)$, where z runs through the spectrum of D. Now, we are looking for a polynomial such that the magnitude of zP(z) on [–1,0] is sufficiently small. In this paper we choose the polynomial [16]

$$P_{2^q-1}(z) = \prod_{k=1}^{q} \left(1 + \gamma \frac{T_{2k-1}(1+2z) - 1}{2} \right)\,,\quad T_p(x) = \cos(p \arccos(x))\,. \tag{6.2}$$

In [7] it has been proved that the spectral radius of the matrix SA is minimized by the polynomial (6.2) when the matrix A has real nonpositive eigenvalues and $\gamma=1$. For $\gamma=1$ we have that (6.2) is equal to

$$P_{2^q-1}(z) = \frac{T_{2^q}(1+2z) - 1}{2z} \frac{1}{4^q}. \tag{6.3}$$

The factorization in (6.2) makes it possible to implement the smoothing operator S in a very efficient way. This is stated in the following theorem.

THEOREM 6.1. *Let* $S=P(D)$ *with* $P(z)$ *defined by* (6.2) *and let the factor matrices* F_j *be defined by*

$$F_j = I + \gamma D_j, \text{ where } D_{j+1} = 4 D_j (I + D_j) \text{ with } D_1 = D \text{ and } j \geq 1. \tag{6.4}$$

Then, $S = P_{2^q-1}(D)$ *can be factorized according to*

$$S = F_q F_{q-1} \cdots F_1 \cdot \square \tag{6.5}$$

Thus, the smoothing matrix S consists of q factor matrices. For a proof of Theorem 6.1 we refer to [16].

The most efficient implementation of S is based on the factorization property in Theorem 6.1. However, in two or more dimensions the precomputation of the factor matrices F_j defined by (6.4) is not attractive. Therefore, we consider an alternative smoothing matrix S which only consists of one-dimensional operators. For our two-dimensional problem (4.7) we apply one-dimensional smoothing in the x- and y-direction, successively. An extra advantage of the splitting in one-dimensional operators is that the application of the smoothing operator S is now hardly complicated when the domain is irregular [5].

As mentioned before, in practice we shall choose D equal to some cheap approximation of (6.1). Since the smoothing matrix S consists of one-dimensional operators, we choose

$$D = \frac{1}{4} \begin{pmatrix} -1 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & 1 & -2 & 1 \\ 0 & & & & 1 & -1 \end{pmatrix}. \tag{6.6}$$

If the factor matrices of (6.5) are computed in advance, then the evaluation of P(D) only requires q matrix–vector operations. Moreover, the factor matrices exhibit a regular pattern which can be exploited for an efficient implementation. For example, applying Theorem 6.1 for matrix D in (6.6) yields the factor matrices

$$F_1 = \frac{1}{4}\begin{pmatrix} 3 & 1 & & & 0 \\ 1 & 2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & & 1 & 2 & 1 \\ 0 & & & & 1 & 3 \end{pmatrix}, \quad F_2 = \frac{1}{4}\begin{pmatrix} 2 & 1 & 1 & & & 0 \\ 1 & 2 & 0 & 1 & & \\ 1 & 0 & 2 & 0 & 1 & \\ & 1 & 0 & 2 & 0 & 1 \\ & & \cdot & \cdot & \cdot & \cdot \end{pmatrix}, \text{ etc.} \qquad (6.7)$$

Evidently, the matrix–vector multiplications with these essentially tridiagonal factor matrices are extremely cheap, especially on vector computers.

We shall now discuss the application of the Jacobi-type iteration method and the CG iteration method to system (4.7).

5.6.1. THE SMOOTHED JACOBI METHOD

For the solution of system (4.7) we apply the smoothed Jacobi method [7]

$$\mathbf{Z}_{k+1} = \mathbf{Z}_k + \omega S\{\ \mathbf{B} - A\,\mathbf{Z}_k\ \}\ , \qquad k=1,2,3,..., \qquad (6.8)$$

where $\mathbf{Z}_k$ is the k-th iterate, ω is a relaxation parameter and S is the smoothing matrix described in Theorem 6.1 with D as in (6.6). For the smoothed Jacobi method we choose $\gamma=1$ (see (6.4)). As mentioned in the previous section, the smoothing matrix S consists of q smoothing factors. In [7] it has been demonstrated that one should not iterate with a fixed value of q. Therefore, we choose the number of smoothing factors at the k-th iteration step equal to k modulo(q+1), which yields the cyclic sequence of 1,2,...,q,0,1,2,...,q,0,1,2,...,q,.. smoothing factors.

Let us now examine how the relaxation parameter ω should be chosen. For the spectral radius of A we have

$$\rho(A) \approx 1 + gH_{max}\left\{\frac{\tau^2}{(\Delta x)^2} + \frac{\tau^2}{(\Delta y)^2}\right\}\ , \quad \text{with } H_{max} = \max_{\substack{1\le i\le nx \\ 1\le j\le ny}} \{H_{i,j}\}\ .$$

Similarly, for the spectral radius of SA we have that

$$\rho(SA) \approx 1 + \frac{1}{4^q} gH_{max}\left\{\frac{\tau^2}{(\Delta x)^2} + \frac{\tau^2}{(\Delta y)^2}\right\}\ .$$

Following the analysis in [7], we obtain

$$\omega = \frac{2}{\rho(SA)}\ . \qquad (6.9)$$

However, in our case, we do not choose ω fixed for each component $Z_{i,j}$. We make ω dependent on the local depth, i.e.,

$$\omega_{i,j} = \frac{2}{1 + \frac{1}{4^q} gH_{i,j}\left\{\frac{\tau^2}{(\Delta x)^2} + \frac{\tau^2}{(\Delta y)^2}\right\}}\ . \qquad (6.10)$$

In the case of a fixed relaxation parameter, we have observed in our experiments that (6.9) is the optimum relaxation parameter. However, we obtain much better results with the relaxation parameter in (6.10) when an irregular bottom topography is used.

5.6.2. THE SMOOTHED CG METHOD

The second iteration method that we applied for the solution of system (4.7) is a preconditioned CG method. The preconditioned CG method can be formulated as follows:

Let $\mathbf{Z}_0$ be an initial guess for $\mathbf{Z}^{(m+1)}$ and

$$\mathbf{R}_0 = \mathbf{B} - A\mathbf{Z}_0 \,, \quad \mathbf{P}_0 = S\mathbf{R}_0$$

For k=0,1,2,...., until convergence

$$\begin{aligned}
\alpha_k &= \frac{\mathbf{R}_k{}^T(S\mathbf{R}_k)}{\mathbf{P}_k{}^T(A\mathbf{P}_k)} \\
\mathbf{Z}_{k+1} &= \mathbf{Z}_k + \alpha_k \mathbf{P}_k \\
\mathbf{R}_{k+1} &= \mathbf{R}_k - \alpha_k A\mathbf{P}_k \\
\beta_k &= \frac{\mathbf{R}_{k+1}{}^T(S\mathbf{R}_{k+1})}{\mathbf{R}_k{}^T(S\mathbf{R}_k)} \\
\mathbf{P}_{k+1} &= S\,\mathbf{R}_{k+1} + \beta_k \mathbf{P}_k \,.
\end{aligned} \tag{6.11}$$

In (6.11) the matrix S denotes the preconditioning matrix. It is well-known that the unpreconditioned CG method can be implemented efficiently on vector and parallel computers, but in general the preconditioned version is much more troublesome. In the literature various techniques for the construction of a suitable preconditioning matrix have been proposed (see [11] for a survey). Here, we choose a positive definite matrix S of the form S=P(D), where D is the difference matrix in (6.6) and P(z) the polynomial (6.2). By choosing $\gamma \in [0,1)$ we obtain that S is positive definite. This preconditioning matrix can be implemented efficiently on vector and parallel computers, because only matrix–vector operations are involved. The convergence is improved by this preconditioning matrix since the condition number of SA is much smaller than that of A. It should be noted that this preconditioning matrix S is independent of A, whereas in general the preconditioning matrix S is some approximation to the inverse of A.

5.7. NUMERICAL EXPERIMENTS

In this section we illustrate for a number of test problems the accuracy and the computational aspects of the time integration method (4.1)-(4.2). In the test problems the water is initially at rest and the motion in the closed basin is generated by a periodic wind stress. Thus, a wind driven circulation is gradually developed.

The following parameter values are used in all experiments:

$f = 1.22e\text{-}4\ s^{-1}$
$g = 9.81\ m/s^2$
$\mu = 0.065\ m^2/s$
$C = 70\ m^{1/2}/s$
$\rho = 1025\ kg/m^3$
$\phi = 45°$.

The experiments have been carried out on the Alliant FX/4. This mini-supercomputer has four vector processors. In all experiments we have used both the vector optimization and the parallel optimization.

At the end of the integration process the numerical solution has been compared with a reference solution computed on the same grid with τ=60 s. The reference solution may be considered as an almost exact solution of our semi-discretized system (3.1). Thus, the accuracy results listed in this section represent the error due to the time integration.

In the experiments we have used a rectangular basin of 400 by 800 km with different bottom topographies. For the grid sizes we have chosen Δx=10 km, Δy=10 km and $\Delta\sigma$=0.2. Thus, the computations have been performed on a grid with nx=41, ny=81 and ns=5. We have integrated over a period of five days with the time-dependent wind stress

$$W_f = 1.5 * (1 + 0.5 * \sin \frac{2\pi t}{24 * 3600}) \quad kg/ms^2. \tag{7.1}$$

Thus, we have a periodically varying wind with a period of 24 hours. To measure the obtained accuracy, we define

ERR-· : maximal global error of either u, v or ζ at T = 120 hours. (7.2)

In the first experiment we have a plane bottom with a depth of 45 m, except for a deeper channel in a diagonal direction (depth 65 m). This is shown in Figure 2. In the second experiment we use a basin with an inclined bottom of a depth of 20 m at one end and 340 m at the other end (see Figure 3).

In Table 7.1 we list the maximal global errors for the test problem with a channel in a diagonal direction. In this experiment the maximal values for u, v and ζ are about 0.4 m/s, 1.1 m/s and 2.6 m, respectively. We have observed that after a few days the solution becomes periodic with a period of 24 hours for any time step τ. For the largest time steps the accuracy results seem to be unacceptable. However, a careful examination of the integration process shows that, even in the case of large time steps, the maximal and minimal values of the periodic numerical solution are very close to the extreme values of the reference solution. The differences are in the

size of a few centimetres. Thus, our integration method hardly introduces a dissipation error. However, for the large time steps, errors in the phase of the periodic solution appear. For example, in the case of τ=4800 s the phase error is about one hour. When we compare the numerical solution computed with τ=4800 s at T=121.33 hours with the reference solution at T=120 hours, the maximal global errors for the three components are 0.050 m/s, 0.054 m/s and 0.124 m, respectively. This is significantly less than in Table 7.1.

τ (s)	ERR-u (m/s)	ERR-v (m/s)	ERR-ζ (m)
600	0.006	0.018	0.038
1200	0.014	0.041	0.087
2400	0.035	0.103	0.212
4800	0.088	0.269	0.549

Table 7.1. Test problem with a channel in a diagonal direction.

We now discuss the computational efficiency of the time integration method (4.1)-(4.2). To represent the results, we use the following notation:

q	: number of smoothing factors (see (6.5))
γ	: smoothing coefficient (see (6.2))
TOTAL	: total computation time
ITER	: computation time for the iteration process
PREC	: computation time for the preconditioning
#ITER	: number of iterations averaged over the integration steps
CONV	: convergence factor averaged over the integration steps.

At each integration step the convergence factor is defined by $(r(k))^{1/k}$, where k is the smallest value for which the residue (cf. (4.6))

$$r(k) = \| B_z^{n+1/2} - AZ_k \|_\infty$$

drops below a certain tolerance. In the experiments we require that $r(k) \leq 10^{-3}$.

In Table 7.2 we list the computation times and the convergence results for the time integration method (4.1)-(4.2) in which either the smoothed Jacobi (SJAC) method or the smoothed CG (SCG) method has been applied. For both iteration methods we vary the number of smoothing factors. The case q=0 corresponds to the unpreconditioned case. For the parameter γ in the preconditioning matrix of the SCG method we have experimentally derived an optimum value. As mentioned in Section 6.1, for the SJAC method we have γ=1.

τ (s)	method	q	μ	TOTAL (s)	ITER (s)	PREC (s)	#ITER	CONV
600	SCG	0		395.6	70.3	0.	3.0	0.23
	SJAC	1		473.0	148.8	21.5	8.4	0.56
1200	SCG	0		233.5	69.8	0.	9.2	0.60
	SJAC	2		318.2	154.7	47.7	17.0	0.72
2400	SCG	0		169.5	87.5	0.	26.0	0.81
		1	0.9	156.8	75.1	25.4	14.1	0.68
		2	0.75	161.3	79.5	37.9	11.4	0.61
	SJAC	3		220.1	137.7	50.9	26.3	0.81
4800	SCG	0		155.3	114.7	0.	75.4	0.91
		1	0.925	118.9	77.9	28.2	31.2	0.80
		2	0.85	109.9	69.5	35.8	21.9	0.74
	SJAC	0		827.1	786.7	0.	869.6	0.99
		1		517.3	477.3	100.2	297.2	0.98
		2		279.0	238.2	75.0	115.8	0.94
		3		193.1	151.8	64.1	55.0	0.88
		4		184.2	143.8	62.1	48.6	0.86

Table 7.2. Computation times for the channel problem.

In the case of a time step of 4800 s, we have listed the results for various values of q. When no preconditioning is applied, the Jacobi method converges extremely slow in this case. However, by applying four smoothing factors, the number of iterations is reduced by a factor of 18, whereas the computation time for the iteration process is reduced by a factor of 5.5.

When no preconditioning is applied, the CG method has a much better convergence behaviour than the Jacobi method. For the CG method it is even better to apply no preconditioning in the case of small time steps, since the number of iterations is already very limited. However, for large time steps both the number of iterations and the computation time are reduced when the preconditioning is applied.

In Table 7.2 we list the optimum values for γ. For values in the neighbourhood of the optimum value the number of iterations hardly increases. Thus, the choice of the parameter γ in the preconditioning matrix S of the SCG method is not critical. In this experiment the SCG method requires less computation time than the SJAC method.

In Table 7.3 we list the maximal global errors for the test problem with the inclined bottom (see Figure 3). In this experiment the maximal values for u, v and ζ are about 0.7 m/s, 1.4 m/s and 1.2 m, respectively. The results are comparable with the results in the first experiment. After a few days the numerical solution also becomes periodic with a period of 24 hours for any time step τ. However, in this experiment the phase errors are much smaller.

τ (s)	ERR-u (m/s)	ERR-v (m/s)	ERR-ζ (m)
600	0.005	0.003	0.016
1200	0.008	0.005	0.018
1800	0.012	0.009	0.035
3600	0.032	0.034	0.122

Table 7.3. Test problem with an inclined bottom.

The computational results in this experiment, which are listed in Table 7.4, are also comparable with the results of the first experiment. Both the number of iterations and the computation time for the iteration process are reduced when the preconditioning is applied. As in the first experiment, the SCG method requires less computation time than the SJAC method.

τ (s)	method	q	μ	TOTAL (s)	ITER (s)	PREC (s)	#ITER	CONV
600	SCG	0		432.2	108.1	0.	7.4	0.49
	SJAC	1		570.4	241.2	41.4	15.8	0.68
1200	SCG	0		326.6	163.9	0.	24.4	0.75
	SJAC	2		413.5	248.3	71.9	28.2	0.77
1800	SCG	0		271.5	160.0	0.	39.7	0.84
		1	0.85	253.7	143.9	48.1	22.7	0.73
	SJAC	3		357.5	247.9	100.3	38.4	0.81
3600	SCG	0		276.1	221.3	0.	109.9	0.92
		1	0.9	193.8	141.2	51.6	47.3	0.82
		2	0.8	240.2	186.4	91.9	45.0	0.82
	SJAC	4		257.5	203.0	91.4	48.6	0.82

Table 7.4. Computation times for the problem with an inclined bottom.

In the experiments we have used both the vector and the parallel optimization of the Alliant FX/4. For both iteration methods the computation time is reduced by about a factor of three by vectorization, and by an additional factor of three by the parallel optimization. However, not only the computation time for both iteration methods, but also the computation time for our integration method (4.1)-(4.2) is reduced by the above-mentioned factors. This shows that our integration method (4.1)-(4.2), in which either the SCG method or the SJAC method has been applied, can be implemented efficiently on vector and parallel computers.

ns	TOTAL (s)	ITER (s)	PREC (s)	#ITER	CONV
1	84.7	76.5	26.0	28.5	0.79
2	94.2	77.1	27.2	30.2	0.80
5	118.9	77.9	28.2	31.2	0.80
10	160.1	80.5	28.5	31.6	0.80
25	278.5	81.3	29.0	31.9	0.80

Table 7.5. Computation times for different numbers of vertical layers.

We now carry out an experiment in which we vary the number of layers in the vertical direction. Our aim is to illustrate the efficiency of the time integration method (4.1)-(4.2) for three-dimensional shallow water problems. We have chosen the bottom topography of the first experiment (i.e., a plane bottom with a deeper diagonal channel) and a time step of 4800 s. The SCG method is used with q=1 and γ=0.925 (see Table 7.2). Table 7.5 presents the computation times and the convergence results for different numbers of grid layers in the vertical direction. The number of vertical grid layers is denoted by ns.

Since the system that we have to solve at the second stage, is of the same computational complexity for both two-dimensional and three-dimensional problems, the results in the last four columns are more or less constant. Thus, the computation time required for the solution of this system is independent of the number of vertical grid layers. In the two-dimensional case (i.e., ns=1) a major part of the computation time is required for the solution of the system at the second stage (about 90%). However, for three-dimensional experiments the computation time for the solution of the system at the second stage becomes relatively less. For example, in the case of ns=10, about half the computation time is required for the solution of this system. This percentage depends on the time step used. In this experiment we have used a rather large time step. For smaller time steps the percentage of computation time required for the solution of the system is significantly less. In conclusion, the time integration method (4.1)-(4.2) is very suited for three-dimensional problems, especially when large time steps are used.

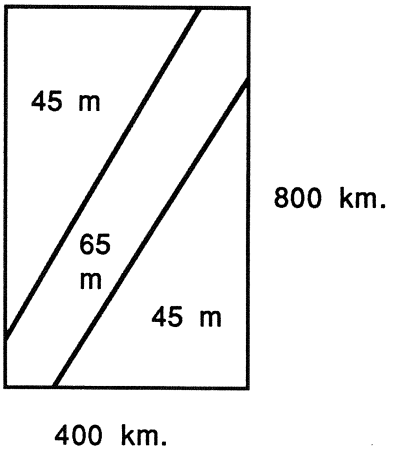

Figure 2.
The plane bottom with a diagonal channel.

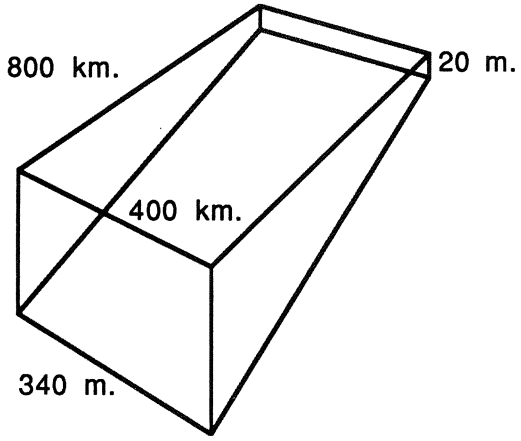

Figure 3.
The inclined bottom.

5.8. Conclusions

In this paper we have presented a two-stage time splitting method for the three-dimensional shallow water equations. The method has been developed in such a way that its stability neither depends on the vertical diffusion term nor on the terms describing the propagation of the surface waves. At the first stage a large number of tridiagonal systems of the same dimension have to be solved. At the second stage the system to be solved is symmetric, five-diagonal and positive definite. For the solution of the latter system we have developed a smoothed Jacobi (SJAC) method and a smoothed CG (SCG) method. Both methods have been accelerated by a technique based on smoothing. The smoothing matrices have been chosen in such a way that the number of iterations is moderate in all cases. Moreover, the smoothing matrices can be implemented efficiently on vector and parallel computers, because only matrix–vector operations are involved. It should be noted that the smoothing matrices for the CG method are independent of the system to be solved. In the experiments the SCG method requires less computation time than the SJAC method.

It has been shown that the time integration method presented in this paper is suited for three-dimensional problems. When we apply our method to two-dimensional problems, the system to be solved at the second stage is the most time consuming part. In three-dimensional models the same amount of computation time is required, because this system is independent of the number of grid layers in the vertical direction. The computation time for the other parts of the method is proportional to the number of vertical grid layers. Therefore, the time splitting method is more efficient for three-dimensional problems than for two-dimensional problems. In [15] it was reported that a time splitting method of this form is already feasible for practical computations of two-dimensional problems.

Finally, the method is mass conservative and can be implemented efficiently on vector and parallel computers.

REFERENCES

1. J.O. BACKHAUS, A semi-implicit scheme for the shallow water equations for application to shelf sea modelling, *Continental Shelf Research*, 2, 243-254 (1983).
2. V. CASULLI, Semi-implicit finite difference methods for the two dimensional shallow water equations, *J. Comp. Phys.*, 86, 56-74 (1990).
3. A.M. DAVIES, Application of the DuFort-Frankel and Saul'ev methods with time splitting to the formulation of a three dimensional hydrodynamic sea model, *Int. J. Numer. Meth. in Fluids*, 5, 405-425 (1985).
4. E.D. DE GOEDE, A computational model for the three-dimensional shallow water flows on the ALLIANT FX/4, *Supercomputer*, 32, 43-49 (1988).
5. E.D. DE GOEDE, Stabilization of a time integrator for the 3D shallow water equations by smoothing techniques, *Int. J. Numer. Meth. in Fluids*, 12, 475-490 (1991).
6. A.R. GOURLAY, *Splitting methods for time dependent partial differential equations, The State of the Art in Numerical Analysis*, D. Jacobs (ed.), Academic press, London - New York - San Francisco, 757-791 (1977).
7. P.J. VAN DER HOUWEN, C. BOON and F.W. WUBS, Analysis of smoothing matrices for the preconditioning of elliptic difference equations, *Z. Angew. Math. Mech.*, 68, 3-10 (1988).
8. Y. KURIHARA, On the use of implicit and iterative methods for the time integration of the wave equation, *Month. Weather Rev.*, 93, 33-46 (1965).
9. P.D. LAX and R.D. RICHTMYER, Survey of the stability of linear finite difference equations, *Comm. Pure Appl. Math.*, 17, 267-293 (1956).
10. J.J. LEENDERTSE, *Aspects of a computational model for long period water wave propagation*, Memorandum RM-5294-PR, Rand Corp., Santa Monica, California, 1967.
11. J. ORTEGA and R. VOIGT, Solution of partial differential equations on vector and parallel computers, *SIAM Review*, 27, 149-240 (1985).
12. G.S. STELLING, *On the construction of computational methods for shallow water flow problems*, Ph.D. Thesis, Delft University, 1983.
13. G.S. STELLING, A.K. WIERSMA and J.B.T.M. WILLEMSE, Practical aspects of accurate tidal computations, *J. Hydr. Eng., ASCE*, 112, 802-817 (1986).
14. J. VERWER and W.H. HUNDSDORFER, Stability and convergence of the Peaceman-Rachford ADI method for initial-boundary problems, *Math. Comp.*, 53, 81-101 (1989).
15. P. WILDERS, Th.L.VAN STIJN, G.S. STELLING and G.A. FOKKEMA, A fully implicit splitting method for accurate tidal computations, *Int. J. Numer. Meth. in Eng.*, 26, 2707-2721 (1988).
16. F.W. WUBS, *Numerical solution of the shallow water equations*, Ph.D. Thesis, University of Amsterdam, Amsterdam, 1987.

Chapter 6

Numerical Methods for the 3D Shallow Water Equations on Vector and Parallel Computers

E.D. de Goede
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB Amsterdam, The Netherlands

In this paper numerical methods for the three-dimensional shallow water equations are examined. Since three-dimensional models require a great computational effort, it is important to construct methods that are not only accurate, but also efficient on vector and parallel computers. We compare the accuracy and efficiency of a conditionally stable and an unconditionally stable method on the Alliant FX/4.
The unconditionally stable method consists of two stages and requires the solution of a sequence of linear systems. For the solution of these systems, we apply a Jacobi-type iteration method and a conjugate gradient iteration method. The performance of both iteration methods is accelerated by a technique based on smoothing. Both *explicit* and *implicit* smoothing is examined. It turns out that the unconditionally stable method is more efficient than the conditionally stable method.

6.1. INTRODUCTION

In numerical analysis, we distinguish explicit and implicit time integrators for partial differential equations. It is well-known that implicit methods are in general unconditionally stable, but cannot exploit the facilities of vector and parallel computers as well as explicit methods do. On the other hand, explicit methods impose a severe restriction on the time step and therefore the time step is not dictated by accuracy considerations.

In this paper we will compare the efficiency and accuracy of a conditionally stable and an unconditionally stable method for the three-dimensional shallow water equations. These methods have been described in [4] and [5], respectively. Since three-dimensional models require a great computational effort, we will pay attention to the efficiency of these numerical methods on vector and parallel computers. The experiments will be carried out on the Alliant FX/4 (a mini-supercomputer with four vector processors).

A mathematical model for the three-dimensional shallow water equations will be used in which the advective terms have been omitted. We will focus on the stability conditions imposed by the vertical diffusion term and by the terms describing the propagation of the surface waves. In three-dimensional models the vertical diffusion term has to be treated implicitly to avoid the maximally stable time step becoming too small. (see e.g., [1,5]). Therefore, the numerical methods described in this paper

treat this term in an implicit way. This requires the solution of a large number of tridiagonal systems, all of the same dimension. Since the tridiagonal systems are independent of each other, the solution of these systems can be computed efficiently in a vector-parallel mode [3].

The terms concerning the propagation of surface waves are integrated differently. For the conditionally stable method these terms are treated partly explicitly, which results in a CFL stability condition that depends on the water depth and on the horizontal mesh sizes Δx and Δy.

The unconditionally stable method consists of two stages. At the first stage the vertical diffusion term is treated implicitly, whereas at the second stage the terms concerning the propagation of the surface waves are treated implicitly. At the second stage a linearization process is used to iteratively solve the nonlinear system. The linearization is chosen in such a way that conservation of mass is guaranteed. Then, at each iteration step, a linear, symmetric, positive definite system has to be solved. In the literature a large number of iteration methods have been proposed for such a system (see e.g., [17]). In this paper we will apply a Jacobi-type iteration method and a conjugate gradient iteration method for the solution of this system. The iteration methods will be accelerated by a technique based on smoothing. Both *explicit* and *implicit* smoothing will be examined. It appears that especially explicit smoothing is suitable on vector and parallel computers.

6.2. MATHEMATICAL MODEL

In this section we will describe a mathematical model for the three-dimensional shallow water equations. We will use a three-dimensional model in sigma co-ordinates in which the advective terms have been omitted. The mathematical model used in this paper is described by

$$\frac{\partial u}{\partial t} = fv - g\frac{\partial \zeta}{\partial x} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial u}{\partial \sigma}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -fu - g\frac{\partial \zeta}{\partial y} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu \frac{\partial v}{\partial \sigma}\right) \tag{2.2}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) - \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right). \tag{2.3}$$

Owing to the sigma transformation [13]

$$\sigma = \frac{\zeta - z}{d + \zeta}, \quad \text{where } -d \le z \le \zeta \text{ and } 1 \ge \sigma \ge 0,$$

the domain is constant in time. We have the closed boundary conditions

$$u(0,y,\sigma,t) = 0, \quad u(L,y,\sigma,t) = 0,$$
$$v(x,0,\sigma,t) = 0, \quad v(x,B,\sigma,t) = 0.$$

The boundary conditions at the sea surface ($\sigma = 0$) are given by

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \cos(\phi), \qquad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho} W_f \sin(\phi), \tag{2.4}$$

Similarly, at the bottom ($\sigma = 1$) we have a linear law of bottom friction of the form

$$\left(\mu \frac{\partial u}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g\, u_d}{C^2}, \qquad \left(\mu \frac{\partial v}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g\, v_d}{C^2},$$

where u_d and v_d represent components of the velocity at some depth near the bottom.

6.3. Space discretization

For the space discretization of the equations (2.1)-(2.3), the computational domain is covered by an nx·ny·ns rectangular staggered grid (see [3,4,5]). Owing to the sigma transformation, we have a constant number of grid layers in the vertical direction. In what follows, $\mathbf{U}(t)$ is a grid function whose components $U_{i,j,k}(t)$ approximate the velocity $u(t)$. The components $U_{i,j,k}(t)$ are numbered lexicographically. Likewise, $\mathbf{V}$, $\mathbf{Z}$, $\mathbf{D}$ and $\mathbf{H}$ are grid functions approximating v, ζ, d and h, respectively. Note that $\mathbf{D}$, $\mathbf{H}$ and $\mathbf{Z}$ are only computed at the upper layer. Furthermore, $\Lambda_{\sigma\sigma}$ is a tridiagonal matrix approximating the vertical diffusion term, including the discretization of the term $1/h^2$. We remark that $\Lambda_{\sigma\sigma}$ does not contain the discretization of the wind stress, because this term is independent of the velocity components (see (2.4)). Θ_1 is an (nx·ny·ns)·(nx·ny) matrix (a row of ns diagonal matrices of order nx·ny with $\Delta\sigma_k$ on the diagonal of the k-th submatrix). Θ_2 is an (nx·ny)·(nx·ny·ns) matrix (a column of ns identity matrices of order nx·ny. F is a four-diagonal matrix (due to the grid staggering) of order nx·ny·ns, approximating the Coriolis term. D_x and D_y are bidiagonal matrices (one diagonal and one lower diagonal) of order nx·ny, approximating the differential operators $\partial/\partial x$ and $\partial/\partial y$, respectively. E_x and E_y are bidiagonal matrices (one diagonal and one upper diagonal) with $E_x = -D_x^T$ and $E_y = -D_y^T$. The matrices D_x and E_x differ because of the grid staggering.

For the approximation of the spatial derivatives, second-order central finite differences are used in both the horizontal and the vertical direction. Now, the semi-discretized system can be written in the form

$$\frac{d}{dt}\mathbf{W} = \mathbf{F}(\mathbf{W}) = \begin{pmatrix} \Lambda_{\sigma\sigma} & F & -\Theta_2 g D_x \\ -F & \Lambda_{\sigma\sigma} & -\Theta_2 g E_y \\ -\Theta_1 H E_x & -\Theta_1 H D_y & 0 \end{pmatrix} \mathbf{W} + \begin{pmatrix} \mathbf{F}_u \\ \mathbf{F}_v \\ \mathbf{0} \end{pmatrix}, \tag{3.1}$$

where $\mathbf{W}=(\mathbf{U},\mathbf{V},\mathbf{Z})^T$ and $(\mathbf{F}_u,\mathbf{F}_v,\mathbf{0})^T$ contains the components of the wind stress. Note that the integrals in (2.3) are approximated by $\Theta_1\mathbf{U}$ and $\Theta_1\mathbf{V}$, respectively.

6.4. TIME INTEGRATION

In this section we will describe time integration methods for the semi-discretized system (3.1). Both a conditionally stable and an unconditionally stable method will be discussed.

6.4.1. THE CONDITIONALLY STABLE METHOD

First we consider the conditionally stable method that has been described in [3]. This method reads

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & 0 & 0 \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & 0 \\ \tau\Theta_1 HE_x & \tau\Theta_1 HD_y & I \end{pmatrix}\begin{pmatrix} U^{n+1} \\ V^{n+1} \\ Z^{n+1} \end{pmatrix} = \begin{pmatrix} I & \tau F & -\tau\Theta_2 gD_x \\ 0 & I & -\tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^n \\ V^n \\ Z^n \end{pmatrix} + \tau\begin{pmatrix} F_u^n \\ F_v^n \\ 0 \end{pmatrix},$$

where τ denotes the time step and $W^n=(U^n,V^n,Z^n)^T$ is a numerical approximation to the solution $W(t)$ of (3.1) at $t=n\tau$. This Vertically Implicit Method (VIM) can be written in the form

$$W^{n+1} = W^n + \tau\,(I - \tau\{A_1+A_2\})^{-1}\,F(W^n)\,, \tag{4.1}$$

with

$$A_1 = \begin{pmatrix} \Lambda_{\sigma\sigma} & 0 & 0 \\ 0 & \Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} 0 & 0 & 0 \\ -F & 0 & 0 \\ -\Theta_1 HE_x & -\Theta_1 HD_y & 0 \end{pmatrix}.$$

The stability condition for method (4.1) is given by

$$\tau < \frac{1}{\sqrt{gh}}\,\frac{1}{\sqrt{\dfrac{1}{(\Delta x)^2}+\dfrac{1}{(\Delta y)^2}}}\,, \tag{4.2}$$

where Δx and Δy denote the horizontal mesh sizes.

Method (4.1) is first-order accurate in time. For the U- and V-component, the implicit treatment of the vertical diffusion term requires the solution of nx·ny tridiagonal systems of dimension ns [3,4]. For large values of h (i.e., very deep water) or for small values of the horizontal mesh sizes, the time step restriction for method (4.1) may be more severe than necessary for accuracy considerations. In order to increase the stability of method (4.1), right-hand side smoothing has been applied in [4]. The application of right-hand side smoothing in more than one direction is complicated. Therefore, we have constructed one-dimensional smoothing matrices in x- and y-direction, successively. In the x-direction the smoothing matrix has the structure

$$\begin{pmatrix} S_u & & \\ & 0 & \\ & & S_z \end{pmatrix},$$

where S_u and S_z denote the smoothing matrices for the right-hand side function of the U- and Z-component, respectively. These smoothing matrices are of the form $S_u=P(D_u)$ and $S_z=P(D_z)$ with P(z) defined by [7,8]

$$P(z) = \frac{T_{2^q}(1+2z) - 1}{2z} \frac{1}{4^q}, \quad T_k(x) = \cos(k \arccos(x)) \tag{4.3}$$

and

$$D_u = \frac{1}{4}\begin{pmatrix} 0 & & & & 0 \\ 1 & -2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & 1 & -2 & 1 \\ 0 & & & & 0 \end{pmatrix}, \quad D_z = \frac{1}{4}\begin{pmatrix} -1 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & 1 & -2 & 1 \\ 0 & & & -1 & 1 \end{pmatrix}. \tag{4.4}$$

In the y-direction the smoothing matrix has a similar structure (see [4]). Note that D_u and D_z only differ in the first and last row, which is due to the grid staggering and to the boundary conditions. The number of different boundary conditions is very limited (open or closed boundaries, u- or ζ-boundaries). The smoothing matrices, including the values in the first and last row, are therefore computed in advance.

The application of right-hand side smoothing to method (4.1) leads to the Stabilized Vertically Implicit Method (SVIM)

$$W^{n+1} = W^n + \tau\,(I - \tau\{A_1+SA_2\})^{-1}\, S\, F(W^n), \tag{4.5}$$

with the matrices A_1 and A_2 as in (4.1). The stability condition for method (4.5) is given by

$$\tau < \pi\, 2^{q-1}\, \frac{1}{\sqrt{gh}} \frac{1}{\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}}, \tag{4.6}$$

where the gain factor obtained by right-hand side smoothing is $\pi 2^{q-1}$ (cf. (4.2)).

Right-hand side smoothing is particularly attractive in problems where it is known that the time derivative of the exact solution (in our case, $\partial w/\partial t$ with $w=(u,v,\zeta)^T$) is a smooth function of the space variable. For example, this occurs in problems where the solution is close to a steady state. In such cases, the right-hand side function of the semi-discretized system (see e.g., in (3.1)) is also a smooth grid function. Thus, it can be multiplied by the smoothing operator S without much loss of accuracy.

In order to prevent large errors, it is therefore important to smooth the complete right-hand side function. In [1] a fractional step method has been developed which

has a comparable accuracy and computational efficiency as method (4.1). However, for the method in [1] right-hand side smoothing is less attractive, because it can only be applied to a part of the right-hand side function.

We emphasize again that in method (4.5) the right-hand side function is smoothed, instead of the grid function W(t) itself. Both types of smoothing may be considered as a technique in which horizontal diffusion is added. The latter type of smoothing is often used (e.g., in the well-known Lax-Wendroff method [14]). However, it may only be applied, without considerable loss of accuracy, if W(t) itself is a smooth grid function for a fixed value of t. This is, in general, not the case. In [15] very high-order smoothing operators have been developed to restrict the decrease in accuracy.

Method (4.5) can be made more accurate by applying a technique in which the water elevation and the velocity components are computed at different time levels. This technique has been introduced in [6]. For method (4.5) this yields

$$\tilde{W}^{n+1} = \tilde{W}^{n} + \tau\,(I - \tau\{A_1 + SA_2\})^{-1}\, S\, F(\tilde{W}^{n}), \tag{4.7}$$

with $\tilde{W}^{n} = (U^{n-1/2}, V^{n-1/2}, Z^{n})^{T}$ and S the smoothing operator in (4.5). The Coriolis term and the vertical diffusion term are still treated first-order accurate in time. However, the terms describing the propagation of the surfaces waves are now treated second-order accurate in time. The stability condition (4.6) is also valid for method (4.7).

6.4.2. THE UNCONDITIONALLY STABLE METHOD

In [5] the two-stage Time Splitting Method (TSM)

$$\begin{aligned} W^{n+1/2} &= W^{n} + \tfrac{1}{2}\tau\,\{\, F^{1}(W^{n+1/2}) + G^{1}(W^{n}) + C^{n+1/2}\,\} \\ W^{n+1} &= W^{n+1/2} + \tfrac{1}{2}\tau\,\{\, F^{2}(W^{n+1/2}) + G^{2}(W^{n+1}) + C^{n+1/2}\,\}, \end{aligned} \tag{4.8}$$

with

$$\begin{aligned} F^{1}(W^{n+1/2}) &= \begin{pmatrix} \Lambda_{\infty} & 0 & 0 \\ -F & \Lambda_{\infty} & 0 \\ 0 & 0 & 0 \end{pmatrix} W^{n+1/2}, \\ G^{1}(W^{n}) &= \begin{pmatrix} 0 & F & -\Theta_2 g D_x \\ 0 & 0 & -\Theta_2 g E_y \\ -\Theta_1 H^{n} E_x & -\Theta_1 H^{n} D_y & 0 \end{pmatrix} W^{n}, \\ F^{2}(W^{n+1/2}) &= \begin{pmatrix} \Lambda_{\infty} & F & 0 \\ -F & \Lambda_{\infty} & 0 \\ 0 & 0 & 0 \end{pmatrix} W^{n+1/2}, \end{aligned} \tag{4.9}$$

$$G^2(W^{n+1}) = \begin{pmatrix} 0 & 0 & -\Theta_2 g D_x \\ 0 & 0 & -\Theta_2 g E_y \\ -\Theta_1 H^{n+1} E_x & -\Theta_1 H^{n+1} D_y & 0 \end{pmatrix} W^{n+1},$$

and $C=(F_u,F_v,0)^T$ has been developed. When we neglect the Coriolis term, this method is second-order accurate in time. For two-dimensional problems, this time splitting method is very similar to the method described in [16]. In [4] it has been shown that this method is unconditionally stable.

At both stages a system of equations has to be solved. The structure of these systems determines the efficiency of method (4.8)-(4.9). At the first stage we have to solve

$$\begin{pmatrix} I - \frac{1}{2}\tau\Lambda_{\sigma\sigma} & 0 & 0 \\ \frac{1}{2}\tau F & I - \frac{1}{2}\tau\Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} U^{n+1/2} \\ V^{n+1/2} \\ Z^{n+1/2} \end{pmatrix} = B^n, \tag{4.10}$$

where B^n contains the discretizations at time level $t=n\tau$. This system is very similar to the system that has to be solved for the SVIM method (cf. (4.1)).

At the second stage the terms describing the propagation of the surface waves are treated implicitly. This system reads

$$\begin{pmatrix} I & 0 & \frac{1}{2}\tau\Theta_2 g D_x \\ 0 & I & \frac{1}{2}\tau\Theta_2 g E_y \\ \frac{1}{2}\tau\Theta_1 H^{n+1} E_x & \frac{1}{2}\tau\Theta_1 H^{n+1} D_y & I \end{pmatrix} \begin{pmatrix} U^{n+1} \\ V^{n+1} \\ Z^{n+1} \end{pmatrix} = B^{n+1/2}, \tag{4.11}$$

where $B^{n+1/2}$ contains the discretizations at time level $t=(n+1/2)\tau$. The equations for the U- and V-component are linear and are not coupled with each other. They are only coupled with the equation for the Z-component. Therefore, the components U^{n+1} and V^{n+1} can easily be eliminated from (4.11) and a nonlinear system in the unknown Z^{n+1} results. A linearization process is used to iteratively solve this nonlinear system. Then, at each iteration step, we obtain a linear, symmetric, positive definite system of the form

$$A(Z^{(m)})\, Z^{(m+1)} = B_z^{n+1/2}, \tag{4.12}$$

where $Z^{(0)}=Z^{n+1/2}$, $B_z^{n+1/2}$ contains the discretizations at $t=(n+1/2)\tau$ for the Z-component and (m) denotes the iteration index. For a detailed description of (4.12) we refer to [5].

We emphasize that the water elevation is the only unknown in system (4.12). Thus, this system is of the same (two-dimensional) structure and thus of the same computational complexity for both two-dimensional and three-dimensional test problems. The computation time for the other parts of method (4.8)-(4.9) is proportional to the number of vertical grid layers. Therefore, this time splitting

method is more efficient for three-dimensional than for two-dimensional problems. In [16] it was reported that a time splitting method of this form is already feasible for two-dimensional problems.

6.5. SOLVING THE LINEAR SYSTEMS

In this section we will describe how the linear systems (4.10) and (4.12) are solved. For system (4.10), which requires for both velocity components the solution of nx·ny tridiagonal systems of dimension ns, we apply the Gaussian Elimination (double sweep) method. Since this method is recursive, it is an unattractive method on vector and parallel computers. However, we make use of the fact that a *large number* of tridiagonal systems of the same dimension has to be solved. Therefore, the systems can be solved efficiently in a vector-parallel mode [3]. Moreover, this method requires a minimal number of operations.

In the literature a large number of iteration methods have been proposed for linear, symmetric systems such as system (4.12). Here, we will apply a Jacobi-type method and a conjugate gradient (CG) method.

6.5.1. THE SMOOTHED JACOBI METHOD

For the solution of system (4.12), written as $AZ=B$, we apply the smoothed Jacobi method [8]

$$Z_{k+1} = Z_k + \omega S\{B - AZ_k\}\ , \qquad k=1,2,3,..., \tag{5.1}$$

where Z_k denotes the k-th iterate, ω is a relaxation parameter and S is a smoothing operator. We only consider smoothing operators S that consist of one-dimensional operators in x- and y-direction, successively. This will be explained later. The one-dimensional smoothing operators are chosen of the form P(D), where D is a difference matrix and the smoothing function P(z) is a polynomial or rational function, yielding *explicit* or *implicit* smoothing, respectively. Here, we choose

$$S = \begin{cases} P(D) & (5.2a) \\ (I - \alpha D)^{-1}\ , & (5.2b) \end{cases}$$

with P(z) as defined in (4.3), D as in (4.4) and α some parameter. The implicit smoothing operator (5.2b) requires the solution of a tridiagonal system. On the other hand, the explicit smoothing operator (5.2a) requires q (tridiagonal) matrix–vector operations, because

$$P(z) = \frac{T_{2^q}(1+2z) - 1}{2z} \frac{1}{4^q} = \prod_{i=1}^{q} \left(1 + \frac{T_{2^{i-1}}(1+2z) - 1}{2} \right). \tag{5.3}$$

For this choice of P(z) with D as in (4.4), the q factor matrices of the explicit operator exhibit a regular pattern, which has been exploited for an efficient implementation [4,5]. The precomputation of these factor matrices is only feasible in one-dimensional cases. Therefore, we apply one-dimensional smoothing in the x-

and y-direction, successively. This enables an efficient implementation of the smoothing operator on both regular and irregular domains [4].

For the explicit smoothing operator (5.2a), a good choice of the relaxation parameter ω in (5.1) has been derived in [5]. In the case of implicit smoothing, the smoothed Jacobi method reads

$$\mathbf{Z}_{k+1} = \mathbf{Z}_k + \omega\,(I-\alpha D_z^{(x)})^{-1}\,(I-\alpha D_z^{(y)})^{-1}\{\mathbf{B} - A\,\mathbf{Z}_k\}\,, \qquad k=1,2,3,..., \tag{5.4}$$

where $D_z^{(x)}$ and $D_z^{(y)}$ denote the matrix D_z in (4.4) applied in the x- and y-direction, respectively. If we choose

$$\omega = -\frac{2\alpha\beta+\alpha^2}{\beta^2}, \quad \text{with the constant } \beta = \frac{\tau^2}{\Delta^2}\,gH_{max}\,, H_{max} = \max_{\substack{1\le i\le nx\\ 1\le j\le ny}} \{H_{i,j}\}$$

and $\Delta = \Delta x = \Delta y$, then method (5.4) may be written in the form

$$\mathbf{Z}_{k+1} = \mathbf{Z}_k - (2\alpha_2 + 1)\,(\beta D_z^{(x)} - \alpha_2 I)^{-1}\,(\beta D_z^{(y)} - \alpha_2 I)^{-1}\,(\mathbf{B} - A\,\mathbf{Z}_k)\,, \tag{5.4'}$$

where $\alpha_2 = \beta / \alpha$. Using the relation $A=I+\beta D_z^{(x)}+\beta D_z^{(y)}$, it can be verified that (5.4') is equivalent to

$$\begin{aligned}(\beta D_z^{(x)} - \alpha_2 I)\,\tilde{\mathbf{Z}} &= (\beta D_z^{(x)} - \alpha_2 I)\,\mathbf{Z}_k + \{\mathbf{B} - A\mathbf{Z}_k\} \qquad (5.5)\\ (\beta D_z^{(y)} - \alpha_2 I)\,\mathbf{Z}_{k+1} &= (\beta D_z^{(y)} - \alpha_2 I)\,\tilde{\mathbf{Z}} + \{\mathbf{B} - A\tilde{\mathbf{Z}}\}\,.\end{aligned}$$

Method (5.5) may be considered as an ADI iteration method written in residual form. For such methods the derivation of parameter values has been described extensively in [17]. In our case this results in an optimum value of $\alpha_2 = \pi\sqrt{\beta}$, which yields $\alpha = \sqrt{\beta} / \pi$. We emphasize, however, that this only applies if we compute the solution sufficiently accurate. For moderately accurate computations, this α-value may not be the best possible. In our numerical experiments the value of α is determined experimentally.

6.5.2. The smoothed CG method

The second iteration method that we apply for the solution of system (4.12), is a preconditioned CG method. The preconditioned CG method can be formulated as follows (see e.g., [10]):

Let $\mathbf{Z}_0$ be an initial guess for $\mathbf{Z}^{(m+1)}$ and

$$\mathbf{R}_0 = \mathbf{B} - A\mathbf{Z}_0 \,, \quad \mathbf{P}_0 = S\mathbf{R}_0$$

For k=0,1,2,...., until convergence

$$\begin{aligned}
\alpha_k &= \frac{\mathbf{R}_k{}^T(S\mathbf{R}_k)}{\mathbf{P}_k{}^T(A\mathbf{P}_k)} \\
\mathbf{Z}_{k+1} &= \mathbf{Z}_k + \alpha_k \mathbf{P}_k \\
\mathbf{R}_{k+1} &= \mathbf{R}_k - \alpha_k A\mathbf{P}_k \\
\beta_k &= \frac{\mathbf{R}_{k+1}{}^T(S\mathbf{R}_{k+1})}{\mathbf{R}_k{}^T(S\mathbf{R}_k)} \\
\mathbf{P}_{k+1} &= S\,\mathbf{R}_{k+1} + \beta_k \mathbf{P}_k \,,
\end{aligned} \tag{5.6}$$

where $\mathbf{R}_k$ denotes the k-th residual vector and $\mathbf{P}_k$ the k-th search direction. In (5.6) the matrix S denotes the preconditioning matrix. It is well-known that the unpreconditioned CG method can be implemented efficiently on vector and parallel computers, but in general the preconditioned version is much more troublesome. In the literature various techniques for the construction of a suitable preconditioning matrix have been proposed (see [12] for a survey). Here, we again use an *explicit* and an *implicit* smoothing operator. In the explicit case we choose a positive definite matrix S of the form S=P(D), where D is the difference matrix in (4.4) and

$$P(z) = \prod_{i=1}^{q} \left(1 + \gamma \frac{T_{2^{i-1}}(1+2z) - 1}{2} \right) , \tag{5.7}$$

where we have to choose $\gamma \in [0,1)$ in order to obtain a positive definite matrix S. If γ=1, then the polynomial P(z) in (5.7) is identical to the polynomial in (4.3). This smoothing operator can be implemented efficiently on vector and parallel computers, because only matrix–vector operations are involved [5]. In the case of implicit preconditioning we apply the incomplete Cholesky factorization [11]. This leads to the well-known ICCG method.

6.6. Numerical experiments

In this section we compare the accuracy and computational efficiency of the conditionally stable methods (4.5) and (4.7) and the unconditionally stable method (4.8)-(4.9). The experiments have been carried out on the Alliant FX/4, which is a mini-supercomputer with four vector processors. In all experiments we have used both the vector and the parallel optimization of the Alliant FX/4.

The water is initially at rest and the motion in the closed basin is generated by a periodic wind stress. Thus, a wind driven circulation is gradually developed. The following parameter values have been used in all experiments:

$C = 70\ m^{1/2}/s$
$f = 1.22e\text{-}4\ s^{-1}$
$g = 9.81\ m/s^2$
$\mu = 0.065\ m^2/s$
$\phi = 90°$
$\rho = 1025\ kg/m^3$.

We have used a rectangular basin of 400 by 800 km with different bottom topographies. For the horizontal grid sizes we have chosen Δx=10 km and Δy=10 km. The computations have been performed on a grid with nx=41, ny=81 and ns=5. We have integrated over a period of five days with a periodically varying wind stress of

$$1.5 + 0.75 * \sin \frac{2\pi\ t}{24 * 3600}\ kg/ms^2 .$$

The following numerical methods have been used:

SVIM : the Stabilized Vertically Implicit Method (4.5)
SVIM2 : the Hansen-type Stabilized Vertically Implicit Method (4.7) (6.1)
TSM : the unconditionally stable Time Splitting Method (4.8)-(4.9).

At the end of the integration process the numerical solution has been compared with a reference solution computed on the same grid with τ=30 s. The reference solution may be considered as an almost exact solution of our semi-discretized system (3.1). Thus, the accuracy results listed in this section represent the error due to the time integration.

To represent the results we define:

q : number of smoothing factors (see (4.3))
ERR·· : maximal global error of either u, v or ζ at the end point T = 5 days
COMP : computation time on the Alliant FX/4.

For the TSM method we require that the residue $\| B_z^{n+1/2} - AZ_k \|_\infty$ drops below the tolerance 10^{-3} (see (4.12)). This value is a good compromise between the accuracy and the computational costs.

In the first experiment we choose a plane bottom of 45 m with a deeper channel in a diagonal direction (depth 65 m). This is shown in Figure 1. For this test problem the results are presented in Table 6.1.

In this experiment the maximal values for u, v and ζ are about 0.4 m/s, 1.1 m/s and 2.6 m, respectively. We have observed that after a few days the solution becomes periodic with a period of 24 hours for any time step. As expected, the SVIM2 method is more accurate than the SVIM method. The results for the SVIM-type methods clearly show that one should not apply more smoothing factors than needed. In the case τ=1800 s and q=3, the results are much less accurate than for q=2, which is sufficient for stability. For a fixed time step τ, the TSM method roughly requires twice as much computation time as the SVIM methods. However, when we consider the accuracy, the TSM method is more accurate.

method	τ (s)	q	ERR-u (m/s)	ERR-v (m/s)	ERR-ζ (m)	COMP (s)
SVIM	270	0	0.005	0.012	0.015	442.3
	800	1	0.031	0.044	0.063	222.5
	1800	2	0.090	0.132	0.194	111.5
	1800	3	0.134	0.250	0.344	121.1
	3600	3	0.154	0.308	0.457	60.0
SVIM2	270	0	0.004	0.005	0.015	444.4
	800	1	0.027	0.023	0.063	225.4
	1800	2	0.071	0.087	0.194	112.2
	1800	3	0.119	0.184	0.349	123.6
	3600	3	0.142	0.233	0.457	61.0
TSM	270		0.002	0.005	0.015	817.3
	800		0.008	0.024	0.054	319.3
	1800		0.024	0.070	0.146	183.6
	3600		0.061	0.180	0.367	160.4

Table 6.1. Test problem with a channel in a diagonal direction.

In the second experiment we use a basin with an inclined bottom of a depth of 20 m at the one end and 340 m at the other end (see Figure 2). The results are listed in Table 6.2a. Here, the maximal values for the three components are about 0.7 m/s, 1.4 m/s and 1.2 m, respectively. For the SVIM2 method, large errors for the velocity components occur if smoothing is applied. On the other hand, the accuracy of the TSM method is very satisfactory, even for large time steps. The experiment with the diagonal channel is the only one in which some inaccuracies occur for the TSM method. This is possibly due to the discontinuous bottom topography. In all other experiments (see also [5]), the errors for the TSM method are very small. The results show that the TSM method is a more suitable method for the three-dimensional shallow water equations than the SVIM-type methods.

In the experiments the SVIM-type methods perform less satisfactory for three-dimensional test problems than for the corresponding two-dimensional ones. As an illustration, for the second test problem we list in Table 6.2b the results for the SVIM2 method when only one grid layer in the vertical direction is used (i.e., ns=1). In this experiment the errors for the velocity components are about ten times smaller, whereas the maximal values for the velocities are only about four times smaller (see Table 6.2a). The TSM method does in general not encounter any accuracy problems for both two- and three-dimensional test problems.

In the three-dimensional experiments the large errors occur near the boundaries. Since we apply smoothing of the right-hand side function, its smoothness plays an important role. We have observed that in the two-dimensional experiments the right-hand side function is smoother near the boundaries than in the three-

method	τ (s)	q	ERR-u (m/s)	ERR-v (m/s)	ERR-ζ (m)	COMP (s)
SVIM2	100	0	0.001	0.001	0.001	1182.1
	300	1	0.055	0.061	0.003	541.4
	600	2	0.288	0.319	0.019	330.3
	1200	3	0.485	0.563	0.067	180.3
TSM	100		0.001	0.001	0.001	2181.6
	300		0.002	0.002	0.004	773.4
	600		0.005	0.003	0.016	437.1
	1200		0.008	0.005	0.018	315.7
	2400		0.018	0.014	0.046	237.3

Table 6.2a. Test problem with an inclined bottom.

dimensional experiments. This results in a smaller decrease of the accuracy when smoothing is applied. Thus, we conclude that the smoothing technique is more suitable for two-dimensional experiments than for three-dimensional ones. In three-dimensional experiments, one should be more careful with the application of right-hand side smoothing.

method	τ (s)	q	ERR-u (m/s)	ERR-v (m/s)	ERR-ζ (m)	COMP (s)
SVIM2	100	0	0.001	0.001	0.001	196.4
	300	1	0.005	0.008	0.003	109.4
	600	2	0.031	0.045	0.015	68.2
	1200	3	0.053	0.082	0.060	37.5

Table 6.2b. Test problem with an inclined bottom and ns=1.

In the literature various numerical methods have been constructed that are implicit in the vertical direction and explicit in the horizontal direction (see e.g., [1,9]). These methods yield an accuracy and efficiency which is more or less similar to the SVIM method without smoothing (i.e., SVIM with q=0). When right-hand side smoothing is used, we can in general apply two or three smoothing factors while the accuracy remains acceptable. In these cases, the SVIM method is about a factor of five more efficient that the aforementioned methods (see also [4]). However, the

TSM method yields more accurate results and in many cases also more efficient results. Therefore, we conclude that the TSM method is a very suitable method for the three-dimensional shallow water equations.

In the experiments we have used both the vector and the parallel optimization of the Alliant FX/4. For all numerical methods described in this paper (see (6.1)), the computation time reduces by about a factor of three due to the vectorization and by an additional factor of three due to the parallel optimization. This shows that these methods can be implemented efficiently on vector and parallel computers.

Below, we will discuss the performance of the iteration methods used for the solution of system (4.12). To represent the results we use the following notation:

q : number of smoothing factors (see (4.3))
γ : smoothing coefficient (see (5.7))
ITER : computation time for the iteration process
PREC : computation time for the preconditioning (PREC is a part of ITER)
#ITER : number of iterations averaged over the integration steps.

In Table 6.3 we list the results for the smoothed Jacobi method. Here the bottom topography with the diagonal channel has been used. In this experiment only twenty-five time steps have been performed. We have varied the number of smoothing factors q. The case q=0 corresponds to the unpreconditioned case, whereas implicit smoothing is denoted by IMP.

	τ = 800 s			τ = 3600 s		
q	ITER (s)	PREC (s)	#ITER	ITER (s)	PREC (s)	#ITER
0	41.3	0.0	110	368.2	0.0	982
1	26.2	3.6	42	279.3	43.9	438
2	10.0	2.8	14	104.0	29.2	146
3	7.4	2.6	9	41.0	14.7	49
4	8.2	3.4	10	12.1	4.8	15
IMP	20.5	14.1 (α=0.8)	12	75.6	51.8 (α=20.6)	45

Table 6.3. Smoothed Jacobi method for the problem with a diagonal channel.

When no preconditioning is applied, the Jacobi method converges extremely slow. When we apply explicit smoothing, both the number of iterations and the computation time are reduced considerably. For example, in the case τ=3600 s and q=4, the computation time for the iteration process is even reduced by a factor of 30.

For the best choice of q, the explicit smoothing operator requires less iterations than the implicit smoothing operator. Moreover, since the implicit smoothing can not be implemented as efficiently as the explicit smoothing, the reduction in computation time is less for implicit smoothing.

In Table 6.4 we list the results for the smoothed CG method. The value of α is in the neighbourhood of the optimum theoretical value given in Section 6.5.1. Moreover, this value is not critical. Here, we use the basin with an inclined bottom. For the implicit preconditioner, viz., the incomplete Cholesky factorization, we only list the number of iterations, because it has not been implemented in an efficient way. An efficient implementation of the Cholesky factorization has been described in [2].

For the parameter γ in the explicit smoothing operator, we have derived experimentally an optimum value. In Table 6.4 these optimum values are presented. For γ-values in the neighbourhood of the optimum value, the number of iterations hardly increases. Thus, the choice of the parameter γ in the preconditioning matrix S of the SCG method is not critical.

In the case of the smallest time step of 800 s, it is better to apply no preconditioning, because the number of iterations is already very limited. For larger time steps both the number of iterations and the computation time reduces when the explicit smoothing operator is applied. The results show that the number of iterations for the ICCG method is slightly less compared with the explicit preconditioning. We expect that the explicit smoothing operators can be implemented more efficiently than the implicit ones, especially on irregular domains. Therefore, the explicit smoothing operators seem to be a good choice for our shallow water problems.

τ (s)	q	γ	ITER (s)	PREC (s)	#ITER
800	0		123.4	0.0	11
1800	0		160.0	0.0	40
	1	0.85	143.9	48.1	23
	ICCG				17
3600	0		221.3	0.0	110
	1	0.9	141.2	51.6	47
	2	0.8	186.4	91.9	45
	ICCG				34

Table 6.4. Results for the CG method.

6.7. CONCLUSIONS

In this paper we have compared the accuracy and computational efficiency of numerical methods for the three-dimensional shallow water equations. Both a conditionally stable and an unconditionally stable method have been examined. The experiments show that both methods can be implemented efficiently on vector and parallel computers. In [4] the stability of the conditionally stable method has been increased by right-hand side smoothing. In general, the application of right-hand side smoothing results in a reduction of the computation time of about a factor of five, while the accuracy is still acceptable [4].

This smoothing technique performs relatively better for two-dimensional problems. In three-dimensional cases, we encounter in some cases large errors for the velocity components. On the other hand, the unconditionally stable method yields very accurate results, even for large time steps. Since three-dimensional models are applied to test problems where the vertical structure of the velocities is needed, especially the accuracy for the velocity components should be emphasized. For the largest time step with an acceptable accuracy, the unconditionally stable method requires in many cases less computation time than the smoothed conditionally stable method. Therefore, we conclude that the unconditionally stable method is a suitable method for the three-dimensional shallow water models.

For the unconditionally stable method a symmetric, pentadiagonal and positive definite system has to be solved. We have examined a Jacobi-type iteration method and a CG iteration method for the solution of this system.These iteration methods have been accelerated by both an *explicit* and an *implicit* preconditioning operator. For our shallow water problems the explicit preconditioner seems to be more efficient. In [5] it has been shown that the smoothed CG method requires less computation time than the smoothed Jacobi-type method.

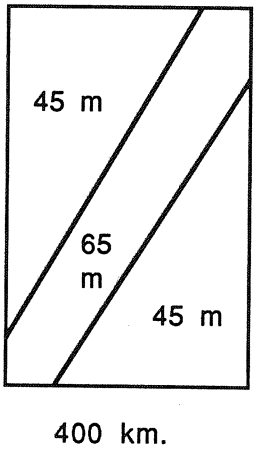

Figure 1.
The plane bottom with a diagonal channel.

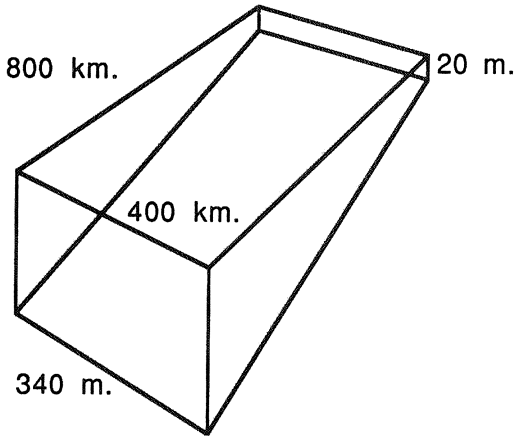

Figure 2.
The inclined bottom.

REFERENCES

1. A.M. DAVIES, Application of the DuFort-Frankel and Saul'ev methods with time splitting to the formulation of a three dimensional hydrodynamic sea model, *Int. J. Numer. Meth. in Fluids*, 5, 405-425 (1985).
2. S.C. EISENSTAT, Efficient implementation of a class of preconditioned conjugate gradient methods, *SIAM J. Sci. Comput.*, 2, 1-4 (1981).
3. E.D. DE GOEDE, A computational model for the three-dimensional shallow water flows on the Alliant FX/4, *Supercomputer*, 32, 43-49 (1988).
4. E.D. DE GOEDE, Stabilization of a time integrator for the 3D shallow water equations by smoothing techniques, *Int. J. Numer. Meth. in Fluids*, 12, 475-490 (1991).
5. E.D. DE GOEDE, A time splitting method for the three-dimensional shallow water equations, *Int. J. Numer. Meth. in Fluids*, 13, 519-534 (1991).
6. W. HANSEN, Hydrodynamical methods applied to oceanographic problems, *Proceedings of the symposium on mathematical hydrodynamical methods of physical oceanography*, Institut für Meereskunde der Universität Hamburg, 25-34 (1961).
7. P.J. VAN DER HOUWEN, Stabilization of explicit difference schemes by smoothing techniques , *in K. Strehmel (ed.): Numerical Treatment of Differential Equations, (Proc. Fourth Seminar Halle 1987: NUMDIFF-4), Teubner-Texte zur Mathematik 104*, BSB B.G. Teubner Verlaggesellschaft, Leipzig, 205-215 (1987).
8. P.J. VAN DER HOUWEN, C. BOON AND F.W. WUBS Analysis of smoothing matrices for the preconditioning of elliptic difference equations, *Z. Angew. Math. Mech.*, 68, 3-10 (1988).
9. J.J. LEENDERTSE, *Aspects of a computational model for long period water wave propagation*, Memorandum RM-5294-PR, Rand Corp., Santa Monica, California, 1967.
10. A.R. MITCHELL AND D.F. GRIFFITHS, *The finite difference method in partial differential equations*, Wiley, Chichester (1980).
11. J.M. MEIJERINK AND H.A. VAN DER VORST, An iterative solution method for linear systems of which the coefficient is an M-matrix, *Math. Comp.*, 31, 148-162 (1977).
12. J. ORTEGA AND R. VOIGT, Solution of partial differential equations on vector and parallel computers, *SIAM Review*, 27, 149-240 (1985).
13. N.A. PHILLIPS, A coordinate system having some special advantages for numerical forecasting, *J. Meteorol.*, 14, 184-194 (1957).
14. R.D. RICHTMYER AND K.W. MORTON, *Difference methods for initial value problems*, Interscience Publishers, Wiley, New York, London (1967).
15. R. SHAPIRO, Smoothing, filtering and boundary effects, *Rev. Geophys. and Space Phys.*, 8, 359-387 (1970).
16. P. WILDERS, Th.L.VAN STIJN, G.S. STELLING AND G.A. FOKKEMA, A fully implicit splitting method for accurate tidal computations, *Int. J. Numer. Meth. in Eng.*, 26, 2707-2721 (1988).
17. D.M. YOUNG, *Iterative solution of large linear systems*, Academic Press, New York (1971).

Chapter 7

On the Numerical Treatment of the Advective Terms in 3D Shallow Water Models

E.D. de Goede
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB Amsterdam, The Netherlands

In this paper we present a numerical method for the three-dimensional shallow water equations. These equations describe flows in e.g., shallow seas, rivers and estuaries. Since three-dimensional models require a great computational effort, the method is constructed in such a way that it fully exploits the facilities of vector and parallel computers. This two-stage method, which is unconditionally stable, is applied to a problem involving the development of a circulation in a rectangular basin and to a river problem in which a jetty has been situated.

7.1. INTRODUCTION

In the past, many numerical methods for the simulation of water flows were based on the so-called two-dimensional shallow water equations. These equations can be obtained from the three-dimensional equations by averaging over the vertical co-ordinate. They only yield the water elevation and the depth-averaged velocities. However, there are many cases in which the vertical structure of the flow is required. For example, when the dispersion of a pollutant is desired. In this paper we will develop a numerical method for the three-dimensional shallow water equations. The three-dimensional models are an order of magnitude more expensive than the two-dimensional models and it is therefore important to construct methods that are not only robust and accurate, but also able to fully exploit the facilities of vector and parallel computers.

In [3] we developed an unconditionally stable two-stage method for the three-dimensional shallow water equations in which the advective terms were omitted. We focussed on the stability conditions imposed by the vertical diffusion and by the terms describing the propagation of the surface waves. In this paper we will incorporate the advective terms into this method.

For the model without advective terms, the unconditionally stable method has been compared with conditionally stable methods [4]. The unconditionally stable method yields the most accurate results. The method is also more efficient, because large time steps can be used. For two-dimensional models, this approach is very similar to the one described in [11], where its feasibility for practical computations has been shown. For three-dimensional models the efficiency of our method is even higher than for two-dimensional models [3].

For the numerical treatment of the advective terms, we will follow the approach developed in [9]. The introduction of the advective terms results in a hardly more complicated system. At the first stage the implicit treatment of the advective terms yields a large, non-symmetric, linear system. For its solution we will develop a Jacobi-type iteration method. When only one iteration is performed, this corresponds with an explicit treatment of the advective terms. Thus, the method offers the facility of both an explicit and an implicit treatment of the advective terms.

At the second stage the method is comparable with the method developed for the model without the advective terms [3]. Again, a sequence of linear systems has to be solved at this stage. This system is solved by a conjugate gradient method in which a preconditioner based on smoothing is used [3]. It appears that this iteration method is highly suitable for vector and parallel computers.

In order to facilitate the comparison with existing numerical methods, we will apply our method to test problems from the literature. In the first experiment we will examine the development of a circulation in a rectangular basin with dimensions representative of the North Sea [1,6]. In the second test problem we will study a river flow past a jetty [9].

7.2. MATHEMATICAL MODEL

In this section we will describe a mathematical model for the three-dimensional shallow water equations. The three-dimensional model in sigma co-ordinates is given by [1,10]

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \omega\frac{\partial u}{\partial \sigma} + fv - g\frac{\partial \zeta}{\partial x} + \lambda\frac{\partial^2 u}{\partial x^2} + \lambda\frac{\partial^2 u}{\partial y^2} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial u}{\partial \sigma}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - \omega\frac{\partial v}{\partial \sigma} - fu - g\frac{\partial \zeta}{\partial y} + \lambda\frac{\partial^2 v}{\partial x^2} + \lambda\frac{\partial^2 v}{\partial y^2} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial v}{\partial \sigma}\right) \tag{2.2}$$

$$\omega = \frac{1}{h}\left\{-(1-\sigma)\left(\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) + \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right)\right) + \frac{\partial}{\partial x}\left(h\int_\sigma^1 u\,d\sigma\right) + \frac{\partial}{\partial y}\left(h\int_\sigma^1 v\,d\sigma\right)\right\} \tag{2.3}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) - \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right). \tag{2.4}$$

The equations (2.1)-(2.3) are the momentum equations and (2.4) denotes the continuity equation. In our model the accelerations in the vertical direction have been neglected, because they are very small, particularly when compared with the acceleration due to gravity. This is known as the so-called shallow water approximation.

In the vertical, the domain is bounded by the bottom topography and the time-dependent water elevation. To ensure that the three-dimensional domain is constant in the vertical direction, system (2.1)-(2.4) has been transformed into the constant interval [0,1] by the sigma transformation [8]

$$\sigma = \frac{\zeta - z}{d + \zeta}, \quad \text{where } -d \le z \le \zeta \text{ and } 1 \ge \sigma \ge 0 . \tag{2.5}$$

The relation between the untransformed (physical) vertical velocity w and the transformed velocity ω is given by [1,10]

$$w = -\omega h + \frac{\partial \zeta}{\partial t} - \sigma \frac{\partial h}{\partial t} + u \left(\frac{\partial \zeta}{\partial x} - \sigma \frac{\partial h}{\partial x} \right) + v \left(\frac{\partial \zeta}{\partial y} - \sigma \frac{\partial h}{\partial y} \right) .$$

The domain is defined by

$$0 \le x \le L , \quad 0 \le y \le B \quad \text{and} \quad 1 \ge \sigma \ge 0 ,$$

i.e., a rectangular basin. Owing to the sigma transformation in the vertical, the domain is constant in time. The boundary conditions at the sea surface ($\sigma = 0$) are given by [1]

$$\left(\mu \frac{\partial u}{\partial \sigma} \right)_{\sigma=0} = -\frac{h}{\rho} W_f \cos(\phi) , \left(\mu \frac{\partial v}{\partial \sigma} \right)_{\sigma=0} = -\frac{h}{\rho} W_f \sin(\phi) \quad \text{and} \quad \omega(x,y,0,t) = 0 .$$

Similarly, at the bottom ($\sigma = 1$) we have

$$\left(\mu \frac{\partial u}{\partial \sigma} \right)_{\sigma=1} = -h \frac{g\, u_d}{C^2} \sqrt{u_d^2 + v_d^2} , \quad \left(\mu \frac{\partial v}{\partial \sigma} \right)_{\sigma=1} = -h \frac{g\, v_d}{C^2} \sqrt{u_d^2 + v_d^2}$$

and $\omega(x,y,1,t) = 0$,

where the first two conditions represent a quadratic law of bottom friction.

7.3. NUMERICAL DISCRETIZATION

To discretize system (2.1)-(2.4), we first apply a finite difference space discretization on a spatial grid that is staggered in both the horizontal and the vertical direction (see [2,3]). Figure 1 shows the horizontal grid spacing. The computational domain is covered by an nx·ny·ns rectangular grid. On this grid the spatial derivatives are replaced by second-order finite differences, which results into a semi-discretized system of dimension nx·ny·(3ns+1). Owing to the sigma transformation (2.5), we have a constant number of grid layers in the vertical direction. In what follows, **U**(t) is a grid function whose components $U_{i,j,k}(t)$ approximate the velocity u(t). The components $U_{i,j,k}(t)$ are numbered lexicographically. Likewise, **V**, Ω, **Z**, **D** and **H** are grid functions approximating v, ω, ζ, d and h, respectively. Note that **Z**, **D** and **H** are two-dimensional unknowns. The grid sizes in x- and y-direction are denoted by Δx and Δy, respectively. In the vertical direction, we choose a varying grid of thickness $\Delta\sigma_k$, where k refers to the k-th grid layer from the surface. Hence, it is possible to increase the resolution near the surface and the bottom.

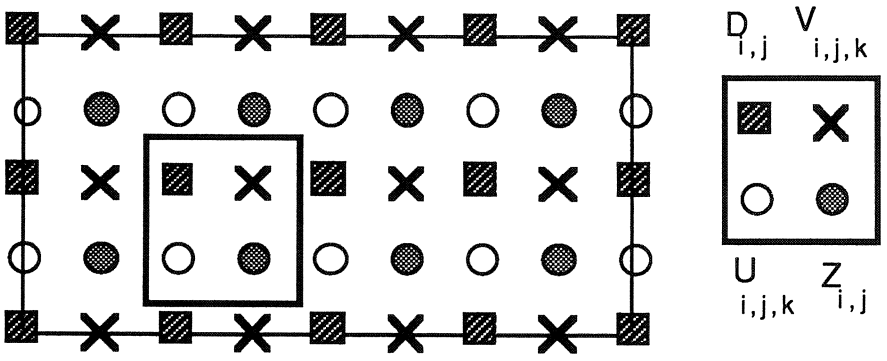

Figure 1. The staggered grid in the horizontal direction.

We now describe the time integrator for the semi-discretized system. Our method consists of two stages. At the first stage we compute intermediate approximations, indicated by an asterisk. The upper indices denote the time level, whereas the lower indices refer to the discretization in space. The first stage reads

$$\frac{U^* - U^n}{1/2\tau} = -U^n U_x^* - V^n U_{1y}^* - \Omega^n U_\sigma^* + fV^n - gZ_{0x}^n + \lambda U_{xx}^* + \lambda U_{yy}^* + \frac{\mu}{H^2} U_{\sigma\sigma}^*$$

$$\frac{V^* - V^n}{1/2\tau} = -U^* V_{1x}^* - V^n V_y^* - \Omega^n V_\sigma^* - fU^* - gZ_{0y}^n + \lambda V_{xx}^* + \lambda V_{yy}^* + \frac{\mu}{H^2} V_{\sigma\sigma}^*$$

$$\Omega^* = \frac{1}{H^*}\{ -(1-\sigma)\left((H^* \int_0^1 U^* d\sigma)_{0x} + (H^* \int_0^1 V^* d\sigma)_{0y} \right)$$

$$+ (H^* \int_\sigma^1 U^* d\sigma)_{0x} + (H^* \int_\sigma^1 V^* d\sigma)_{0y} \} \qquad (3.1a)$$

$$\frac{Z^* - Z^n}{1/2\tau} = -(H^n \int_0^1 U^n d\sigma)_{0x} - (H^n \int_0^1 V^n d\sigma)_{0y},$$

where τ denotes the time step. Next, we perform the second stage to arrive at the new time level.

$$\frac{U^{n+1}-U^*}{1/2\tau} = -U^{n+1}U_x^* - V^*U_y^* - \Omega^*U_\sigma^* + fV^* - gZ_{0x}^{n+1} + \lambda U_{xx}^* + \lambda U_{yy}^* + \frac{\mu}{H^2}U_{\sigma\sigma}^*$$

$$\frac{V^{n+1}-V^*}{1/2\tau} = -U^*V_x^* - V^{n+1}V_y^* - \Omega^*V_\sigma^* - fU^* - gZ_{0y}^{n+1} + \lambda V_{xx}^* + \lambda V_{yy}^* + \frac{\mu}{H^2}V_{\sigma\sigma}^*$$

$$\Omega^{n+1} = \frac{1}{H^{n+1}}\{-(1-\sigma)\left((H^{n+1}\int_0^1 U^{n+1}d\sigma)_{0x} + (H^{n+1}\int_0^1 V^{n+1}d\sigma)_{0y}\right)$$

$$+ (H^{n+1}\int_\sigma^1 U^{n+1}d\sigma)_{0x} + (H^{n+1}\int_\sigma^1 V^{n+1}d\sigma)_{0y}\} \qquad (3.1b)$$

$$\frac{Z^{n+1}-Z^*}{1/2\tau} = -(H^{n+1}\int_0^1 U^{n+1}d\sigma)_{0x} - (H^{n+1}\int_0^1 V^{n+1}d\sigma)_{0y}.$$

In the Appendix a detailed description of the space discretization is given. When applied to two-dimensional problems, this method is very similar to the method developed in [11].

In [3] it has been proved that method (3.1) is unconditionally stable for a model in which the advective terms and the Coriolis term have been omitted. For a model including the Coriolis term and the advective terms, method (3.1) has the same good stability properties.

Method (3.1) is first-order accurate in time. To obtain second-order accuracy, the Coriolis term, the mixed advective terms and the bottom friction term should be adjusted. However, a second-order treatment of these terms would decrease the computational efficiency of our time splitting method dramatically. This will be explained in the next section. It should be noted that the diffusion terms and the terms describing the propagation of the surface waves are second-order accurate in time.

Almost all spatial derivatives are discretized in a symmetric and therefore non-dissipative way. However, at the first stage, the mixed advective terms $v\partial u/\partial y$ and $u\partial v/\partial x$ are approximated by an upwind discretization. This approach has been developed in [9]. In the numerical experiments the resulting dissipation is just enough to suppress spurious oscillations. The dissipation, which is of fourth-order magnitude, does not lead to an undesirable damping of the solution.

At the first stage our time splitting method requires the successive solution of two large, non-symmetric, linear systems (first a system for the U-component, next a system for the V-component). At the second stage a nonlinear system has to be solved. These systems and the iteration methods for its solution will be discussed in the next section.

7.4. SOLVING THE SYSTEMS

The structure of the systems at both stages determines the efficiency of our unconditionally stable method. At the first stage the Ω- and Z-component can be

computed straightforwardly. For the U- and V-component, the method requires the successive solution of two non-symmetric, linear systems of dimension nx·ny·ns. The system for the U-component may be written in the form

$$U^* + \tfrac{1}{2}\tau \left\{ U^n U_x^* + V^n U_y^* + \Omega^n U_\sigma^* - \lambda U_{xx}^* - \lambda U_{yy}^* - \frac{\mu}{H^2} U_{\sigma\sigma}^* \right\} = B^n , \quad (4.1)$$

where B^n contains the terms at $t=n\tau$. For the V-component, we have a similar system. The matrix at the left-hand side of system (4.1) contains nine non-zero diagonals. Seven diagonals are due to the discretizations in the horizontal direction and three are due to the vertical derivatives, with one overlapping (main) diagonal. Some of these diagonals contain zero elements. System (4.1) may be written as

$$(I + \tfrac{1}{2}\tau D_h + \tfrac{1}{2}\tau D_v) U^* = B^n , \quad (4.1')$$

where the matrix D_v represents the discretizations in the vertical direction and the contribution on the main diagonal of the discretizations in the horizontal direction. The remaining (six) diagonals resulting from the horizontal derivatives are represented by the matrix D_h.

For the solution of system (4.1'), we apply the preconditioned Jacobi-type method

$$U_{k+1} = U_k + \alpha (I + \tfrac{1}{2}\tau D_v)^{-1} \left\{ B_u^n - (I + \tfrac{1}{2}\tau D_h + \tfrac{1}{2}\tau D_v) U_k \right\} , \; k=1,2,.. \quad (4.2)$$

where α is a relaxation parameter and U_k denotes the k-th iterate with $U_0=U^n$. The term $\alpha (I + \frac{1}{2}\tau D_v)^{-1}$ represents the preconditioning. Method (4.2) can be written in the more efficient form

$$U_{k+1} = U_k + \alpha \left\{ (I + \tfrac{1}{2}\tau D_v)^{-1} (B_u^n - \tfrac{1}{2}\tau D_h U_k) - U_k \right\} . \quad (4.2')$$

At each iteration step the implicit operator $(I + \frac{1}{2}\tau D_v)^{-1}$ requires the solution of nx·ny tridiagonal systems of dimension ns. For its solution we apply the Gaussian Elimination (double sweep) method, which requires a minimal number of operations. Since this is a recursive method, it is an unattractive method on vector and parallel computers. However, we make use of the fact that a large number of tridiagonal systems of the same dimension has to be solved. Therefore, the systems can be solved efficiently in a vector-parallel mode [2].

For the relaxation parameter α we choose either

$$\alpha = 1 \quad \text{or} \quad \alpha = \frac{1}{\rho(I + \frac{1}{2}\tau D_h)} ,$$

where $\rho(\cdot)$ denotes the spectral radius. The Jacobi-type method (4.2') starts with $\alpha=1$. As soon as the residue increases, we switch to the second choice. For this choice the iteration process always converges as opposed to the choice $\alpha=1$. If the method converges for $\alpha=1$, then this convergence is faster than for $\alpha=1/(\rho(I + \frac{1}{2}\tau D_h))$.

At the second stage the equations for the U- and V-component are linear and are not coupled with each other. They are only coupled with the equation for the Ω- and Z-component. If U^{n+1}, V^{n+1} and Z^{n+1} have been computed, then the values for the Ω-component are computed. Owing to our choice of the time splitting, the components U^{n+1} and V^{n+1} can be eliminated and a system merely in the unknown Z^{n+1} results. In order to accomplish such an elimination, the Coriolis term, the mixed advective terms and the bottom friction term have been treated first-order accurate in time. A second-order treatment of these terms would have resulted in a much more complicated system. In that case, the U- and V-component can not be eliminated easily.

We will now describe the system for each cell (i,j) of component Z. This system reads

$$Z_{i,j}^{n+1} - \frac{\tau^2 g}{4(\Delta x)^2}\left\{ \bar{R}_{i+1,j}^{n+1}\left(Z_{i+1,j}^{n+1} - Z_{i,j}^{n+1} \right) - \bar{R}_{i,j}^{n+1}\left(Z_{i,j}^{n+1} - Z_{i-1,j}^{n+1} \right) \right\} \qquad (4.3)$$

$$- \frac{\tau^2 g}{4(\Delta y)^2}\left\{ \tilde{R}_{i,j}^{n+1}\left(Z_{i,j+1}^{n+1} - Z_{i,j}^{n+1} \right) - \tilde{R}_{i,j-1}^{n+1}\left(Z_{i,j}^{n+1} - Z_{i,j-1}^{n+1} \right) \right\}$$

$$= B_{i,j}^{n+1/2}, \quad \text{for} \quad \begin{matrix} i=1,\ldots,nx \\ j=1,\ldots,ny \end{matrix},$$

where

$$\bar{R}_{i,j}^{n+1} = \left\{ \tfrac{1}{2}\left(Z_{i,j}^{n+1} + Z_{i-1,j}^{n+1} \right) + \tfrac{1}{2}\left(D_{i,j} + D_{i,j-1} \right) \right\}$$

$$* \sum_{k=1}^{ns} \Delta\sigma_k \Big/ \left(1 + \tfrac{1}{2}\tau \left(U_x^{n+1/2} \right)_{i,j,k} \right),$$

$$\tilde{R}_{i,j}^{n+1} = \left\{ \tfrac{1}{2}\left(Z_{i,j}^{n+1} + Z_{i,j+1}^{n+1} \right) + \tfrac{1}{2}\left(D_{i,j} + D_{i+1,j} \right) \right\}$$

$$* \sum_{k=1}^{ns} \Delta\sigma_k \Big/ \left(1 + \tfrac{1}{2}\tau \left(U_y^{n+1/2} \right)_{i,j,k} \right).$$

System (4.3) is a nonlinear equation, because $R_{i,j}$ contains the component $Z_{i,j}$. This system may be written in the form

$$A(Z^{n+1})\, Z^{n+1} = B^{n+1/2},$$

where $B^{n+1/2}$ contains the terms at $t=(n+1/2)\tau$. For its linearization we introduce the iteration process

$$A(Z^{(m)})\, Z^{(m+1)} = B^{n+1/2}, \qquad (4.4)$$

where $\mathbf{Z}^{(0)}=\mathbf{Z}^{n+1/2}$ and the upper index (m) denotes the iteration index. The matrix $A(\mathbf{Z}^{(m)})$ is symmetric. For the solution of system (4.4) we developed a conjugate gradient method in which a preconditioner based on smoothing is used [3]. When this system has been solved, the values for the U- and V-component can be computed straightforwardly.

It should be noted that the water elevation is the only unknown in system (4.4). This system is for both two-dimensional and three-dimensional models of the same (two-dimensional) structure and thus of the same computational complexity. Therefore, the time integration method (3.1) is relatively more efficient for three-dimensional problems than for two-dimensional ones.

7.5. NUMERICAL EXPERIMENTS

In order to examine its accuracy and computational efficiency, method (3.1) has been applied to a problem with a closed basin and on a river problem in which a jetty has been situated. In the first experiment we have used a closed rectangular basin of 400 km by 800 km with a constant depth of 65 m. The other parameters are $f=1.22e\text{-}4\ s^{-1}$, $g=9.81\ m/s^2$, $\rho=1025\ kg/m^3$, $\phi=90°$, $C=70\ m^{1/2}/s$, $\lambda=0.0\ m^2/s$ and $\mu=0.065\ m^2/s$. This experiment has also been carried out in [1,6]. The water is initially at rest and the motion in the closed basin is generated by a constant wind stress of $1.5\ kg/ms^2$. The computations have been performed on a grid with nx=10, ny=18 and ns=11, which implies horizontal mesh sizes of 400/9 km and 800/17 km, respectively. In the vertical direction we have chosen for every k: $\Delta\sigma_k=1/ns$. We have integrated over a period of 100 hours.

At the end of the integration process the numerical solution has been compared with a reference solution computed on the same grid with $\tau=30$ s. The reference solution may be considered as an almost exact solution of our semi-discretized system. Thus, the accuracy results listed in this section represent the error due to the time integration.

The experiments have been carried out on an Alliant FX/4, which is a mini-supercomputer having four vector processors. In all experiments we have used both the vector optimization and the parallel optimization.

To represent the results we use the following notation:

ERR-· : maximal global error of either u, v or ζ at the end point T = 100 hours
COMP : computation time on the Alliant FX/4.

We require that the residue for the two iteration methods (viz., the Jacobi-type method (4.2') and the CG method) drops below the tolerance 10^{-3}. By choosing this value, we obtain a good compromise between the accuracy and the computational costs. For this test problem the results are listed in Table 5.1.

In this experiment the maximal values for u, v and ζ are about 0.2 m/s, 0.4 m/s and 1.0 m, respectively. In Table 5.1 no accuracy results have been listed for the vertical velocity w, because these velocities are are small. In this experiment the influence of the advective terms is very limited. Therefore, an explicit treatment of these terms (i.e., only one iteration of the Jacobi-type method (4.2')) is sufficient. It should be noted that one iteration of this method is sufficient to obtain an implicit treatment of the vertical diffusion term. Figures 2a-b show the vertical profiles of

the U- and V-component at the centre of the rectangular basin at T=100 h. At that time the steady state has been reached for moderate values of the time step. Table 5.1 clearly shows that this is not the case for τ=7200 s at T=100 h. If we integrate over a longer period with a time step of 7200 s, then the solution becomes stationary too. The numerical results are in agreement with the results in [1,6].

τ (s)	ERR-u (m/s)	ERR-v (m/s)	ERR-ζ (m)	COMP (s)
360	0.004	0.002	0.005	147.3
1800	0.006	0.005	0.009	30.6
3600	0.009	0.006	0.013	15.5
6000	0.011	0.022	0.035	10.0
7200	0.029	0.031	0.135	8.7

Table 5.1. Test problem with a rectangular basin.

For several numerical methods developed for the 3D shallow water equations, the time step is restricted by the CFL condition $\tau < \Delta / \sqrt{2gh}$, where Δ=min(Δx,Δy). Examples of such conditionally stable methods are described in [1, 2, 6 and 7]. In our experiment this results in a maximally stable time step of about 1245 s. However, in [1,6] the time step was significantly below the CFL condition. In [6] a time step of 360 s was used. The method in [1] was carried out with τ=720 s for the advective terms and with τ=180 s for the terms describing the propagation of the surface waves. For method (3.1) much larger time steps are possible. Moreover, even for a large time step of 3600 s, the relative errors are still very small.

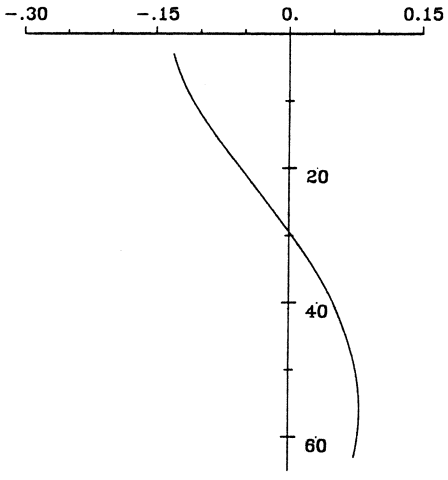

Figure 2a. The vertical U-profile.

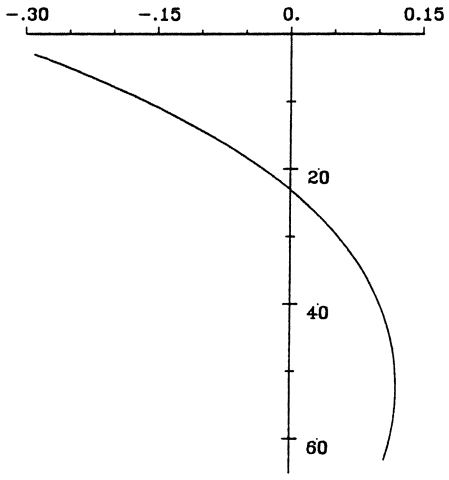

Figure 2b. The vertical V-profile.

As mentioned earlier, the advective terms do not play an important role in this experiment. However, in the second experiment these terms play a crucial role. We examine a flow past a jetty [9]. A rectangular domain with a horizontal dimension of 1500 m by 300 m and a constant depth of 25 m has been used. At the left open boundary, we have prescribed an inflow condition of u=0.5 m/s and at the right boundary a uniform water level ζ=0 m has been given. For a detailed description of the initial conditions and boundary conditions we refer to [9]. The horizontal mesh sizes are 25 m.

Near the boundaries, the discretization of the advective terms have been chosen in a special way. Especially at inflow, the discretization of the advective terms may cause instabilities. To obtain a stable boundary treatment, we have applied the discretization developed in [9]. At the open boundaries, a stabilizing effect is often experienced when Riemann invariants are prescribed. In our experiment this would have resulted into the inflow condition

$$u + 2\sqrt{gh} = 0.5 + 2\sqrt{gh_o} ,$$

where the boundary value for h is denoted by h_o. Since the Riemann invariants (i.e., $u \pm 2\sqrt{gh}$) are in general not known, we have used the following variant [9]:

$$u + \varepsilon \frac{\partial}{\partial t}(u + 2\sqrt{gh}) = 0.5 , \qquad (5.1)$$

with ε some parameter. The time derivative of the Riemann invariant in (5.1) has been discretized in a straightforward manner. For sufficiently small values of ε, the boundary condition (5.1) is only a small perturbation of the original inflow condition u=0.5 m/s. The time-derivative of the Riemann invariant in (5.1) has been introduced to obtain a weakly reflective boundary condition for the short wave components. These components mainly originate from the initial values and the eigenfrequencies of the model. Without Riemann invariants, these components may disturb the solution for a long time, because there is, in general, little dissipation in the model.

In this experiment we have varied the number of grid layers in the vertical direction. At first, we have only used one vertical grid layer. In this case, a comparison with the results in [9] is possible. Various flow patterns are shown in Figure 3. One clearly sees the development of eddies past the jetty. After three hours, the solution is almost stationary. The numerical results are in agreement with the results in [9].

The discretization of the advective terms, which was developed in [9], is very important in this experiment. Both the special discretization near the boundaries and the introduction of some dissipation by the upwind discretization of some advective terms are necessary in order to obtain stable results when a large number of time steps is performed. For example, a central discretization of the advective terms yields instabilities.

We have also computed three-dimensional velocity profiles (e.g., with ns=5). The results are again in agreement with the reference solution. For realistic time steps the Jacobi-type method (4.2') requires less than ten iterations. We have observed that

the number of iterations hardly depends on the number of grid layers in the vertical direction and also hardly depends on the value of the vertical diffusion coefficient.

In the experiments we have used both the vector and the parallel optimization of the Alliant FX/4. For our integration method (3.1) the computation time reduces by about a factor of three due to the vectorization, and by an additional factor of three due to the parallel optimization. This shows that this method can be implemented efficiently on vector and parallel computers. In [5] the computational efficiency of this method was demonstrated on the CRAY Y-MP4/464. On four processors we obtained a performance of more than 500 Mflops.

REFERENCES

1. A.M. DAVIES, Application of the Galerkin methods to the formulation of a three dimensional hydrodynamic nonlinear hydrodynamic sea model, *Appl. Math. Modelling*, 4, 245-256 (1980).
2. E.D. DE GOEDE, A computational model for the three-dimensional shallow water flows on the ALLIANT FX/4, *Supercomputer*, 32, 43-49 (1988).
3. E.D. DE GOEDE, A time splitting method for the three-dimensional shallow water equations, *Int. J. Numer. Meth. in Fluids*, 13, 519-534 (1991).
4. E.D. DE GOEDE, Numerical methods for the 3D shallow water equations on vector and parallel computers, *Appl. Numer. Math.*, 10, 3-18 (1992).
5. E.D. DE GOEDE, 3D shallow water model on the CRAY Y-MP4/464, *Proceedings of the 6th Int. Workshop on the Use of Supercomputers in Theoretical Science, Antwerp*, 107-114 (1991).
6. R.W. LARDNER and H.M. CEKIRGE, A new algorithm for three-dimensional tidal and storm surge computations, *Appl. Math. Modelling*, 12, 471-481 (1988).
7. J.J. LEENDERTSE, *Aspects of a computational model for long period water wave propagation*, Memorandum RM-5294-PR, Rand Corp., Santa Monica, California, 1967.
8. N.A. PHILLIPS, A coordinate system having some special advantages for numerical forecasting, *J. Meteorol.*, 14, 184-194 (1957).
9. G.S. STELLING, *On the construction of computational methods for shallow water flow problems*, Ph.D. Thesis, Delft University, 1983.
10. TRISULA, A multi-dimensional flow and water-quality simulation system, Delft Hydraulics, The Netherlands, 1989.
11. P. WILDERS, Th.L.VAN STIJN, G.S. STELLING and G.A. FOKKEMA, A fully implicit splitting method for accurate tidal computations, *Int. J. Numer. Meth. in Eng.*, 26, 2707-2721 (1988).

APPENDIX: FINITE DIFFERENCES

The operators used in this paper are of the following form:

$$\{ U U_x \}_{i,j,k} = U_{i,j,k} \frac{U_{i+1,j,k} - U_{i-1,j,k}}{2\Delta x}$$

$$\{ Z_{0x} \}_{i,j} = \frac{Z_{i,j} - Z_{i-1,j}}{\Delta x}$$

$$\{ (H\int_0^1 U d\sigma)_{0x} \}_{i,j} = (\tilde{H}_{i+1,j} \sum_{k=1}^{ns} \Delta\sigma_k U_{i+1,j,k} - \tilde{H}_{i,j} \sum_{k=1}^{ns} \Delta\sigma_k U_{i,j,k}) / \Delta x$$

$$\{ U V_{1x} \}_{i,j,k} =$$

$$\begin{cases} \tilde{U}_{i,j,k} (3V_{i,j,k} - 4V_{i-1,j,k} + V_{i-2,j,k}) / (2\Delta x) & \text{if } \tilde{U}_{i,j,k} > 0 \\ \tilde{U}_{i,j,k} (-3V_{i,j,k} + 4V_{i+1,j,k} - V_{i+2,j,k}) / (2\Delta x) & \text{if } \tilde{U}_{i,j,k} \le 0 \end{cases}$$

$$\{ U V_x \}_{i,j,k} = \tilde{U}_{i,j,k} \frac{V_{i+1,j,k} - V_{i-1,j,k}}{2\Delta x}$$

$$\{ U_{xx} \}_{i,j,k} = \frac{U_{i+1,j,k} - 2U_{i,j,k} + U_{i-1,j,k}}{(\Delta x)^2},$$

where

$$\tilde{U}_{i,j,k} = 0.25 (U_{i,j,k} + U_{i+1,j,k} + U_{i,j+1,k} + U_{i+1,j+1,k})$$
$$\tilde{H}_{i,j} = 0.5 (H_{i,j} + H_{i,j-1}) .$$

The operators in the y-direction are defined similarly. For the discretizations in the vertical direction we define

$$\{ \Omega U_\sigma \}_{i,j,k} = \frac{1}{\Delta\sigma_k} \left\{ \tilde{\Omega}_{i,j,k+1} \frac{\Delta\sigma_k U_{i,j,k+1} + \Delta\sigma_{k+1} U_{i,j,k}}{\Delta\sigma_{k+1} + \Delta\sigma_k} - \tilde{\Omega}_{i,j,k} \frac{\Delta\sigma_{k-1} U_{i,j,k} + \Delta\sigma_k U_{i,j,k-1}}{\Delta\sigma_k + \Delta\sigma_{k-1}} \right\}$$

$$\{ U_{\sigma\sigma} \}_{i,j,k} = \frac{8}{(\Delta\sigma_{k-1} + 2\Delta\sigma_k + \Delta\sigma_{k+1})} \left\{ \frac{\mu_{k+1} (U_{i,j,k+1} - U_{i,j,k})}{\Delta\sigma_{k+1} + \Delta\sigma_k} - \frac{\mu_k (U_{i,j,k} - U_{i,j,k-1})}{\Delta\sigma_k + \Delta\sigma_{k-1}} \right\},$$

where

$$\tilde{\Omega}_{i,j} = 0.5 (\Omega_{i,j} + \Omega_{i-1,j}) .$$

Similarly, we define the operators for the V-component.

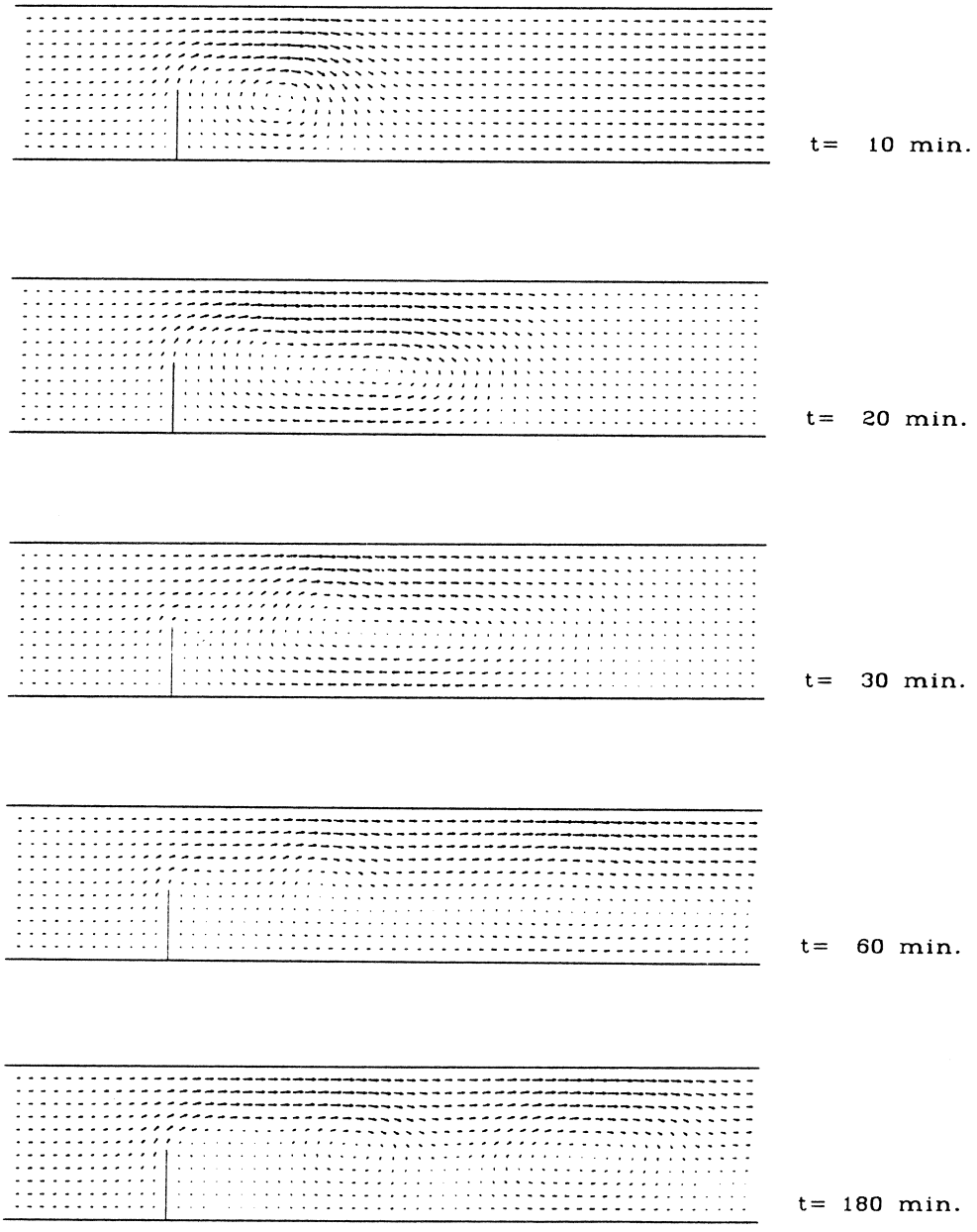

Figure 3. The flow patterns for the river problem with ns=1.

Chapter 8

A Three-Dimensional Shallow Water Model on the CRAY Y-MP4/464

E.D. de Goede
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB Amsterdam, The Netherlands

Simulation of three-dimensional shallow water flows requires the use of fast computers, along with numerical methods that fully exploit the potential of vector and parallel facilities of these computers. In this paper we discuss the implementation of such a numerical method on the CRAY Y-MP4/464, which has recently been installed at the Academic Computing Services Amsterdam (SARA).

8.1. INTRODUCTION

In recent years, numerical modeling has become an important tool for computing shallow water models. For example, flows in rivers, estuaries and seas can be described by these numerical models. Whereas in the past it was necessary to use scale models, it is now possible to solve the three-dimensional shallow water equations using computers. The application of three-dimensional models requires a great computational effort, especially when a high resolution is needed. Therefore, it is necessary to construct numerical methods that are not only robust and accurate, but also efficient on vector and parallel computers. To obtain a computationally efficient method, the VECPARCOMP project has been started. This is a joint project of Rijkswaterstaat (Dutch Water Control and Public Works department) and CWI (Centre for Mathematics and Computer Science).

In [3] we described an unconditionally stable method for the three-dimensional shallow water equations. In that paper it was reported that this method can be implemented efficiently on an Alliant FX/4 (a shared memory mini-supercomputer with four vector processors). In this paper we will describe the implementation of this method on a CRAY Y-MP. In December 1990 a CRAY Y-MP4/464 has been installed at the Academic Computing Services Amsterdam (SARA). Like the Alliant FX/4, this computer is a shared memory system with four vector processors. However, the CRAY Y-MP has a much smaller clock cycle time (6 ns versus 170 ns for the Alliant FX/4). The CRAY Y-MP4/464 replaces a (Control Data Corporation) CYBER 205 installed in 1983.

Both vectorization and parallelism will be examined on the CRAY Y-MP4/464. We will implement several types of parallelism, viz., macrotasking, microtasking and autotasking. A comparison will be made with results on the Alliant FX/4. Since the code was written in the ANSI FORTRAN 77 programming language, the same code was used for both computers.

8.2. MATHEMATICAL MODEL

A mathematical model for the three-dimensional shallow water equations is described by

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \omega\frac{\partial u}{\partial \sigma} + fv - g\frac{\partial \zeta}{\partial x} + \lambda\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial u}{\partial \sigma}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - \omega\frac{\partial v}{\partial \sigma} - fu - g\frac{\partial \zeta}{\partial y} + \lambda\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial v}{\partial \sigma}\right) \tag{2.2}$$

$$\omega = \frac{1}{h}\left\{-(1-\sigma)\left(\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) + \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right)\right) + \frac{\partial}{\partial x}\left(h\int_\sigma^1 u\,d\sigma\right) + \frac{\partial}{\partial y}\left(h\int_\sigma^1 v\,d\sigma\right)\right\} \tag{2.3}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}\left(h\int_0^1 u\,d\sigma\right) - \frac{\partial}{\partial y}\left(h\int_0^1 v\,d\sigma\right). \tag{2.4}$$

The equations (2.1)-(2.3) are the momentum equations and (2.4) denotes the continuity equation. In the vertical, the domain is bounded by the bottom topography and the time-dependent water elevation. To ensure that the three-dimensional domain is also constant in the vertical direction, system (2.1)-(2.4) has been transformed in the vertical into depth-following (sigma) co-ordinates.

In the experiments we will examine a three-dimensional flow past a jetty. The geometry and a velocity pattern are shown in Figure 1. The rectangular basin has a horizontal dimension of 1500 m by 300 m and a constant depth of 25 m. At the left boundary, we prescribe an inflow condition of u=0.5 m/s and at the right boundary a uniform water level ζ=0 m is given. For a detailed description of the initial conditions and boundary conditions we refer to [3,6].

To obtain a discrete system representing (2.1)-(2.4), the equations are discretized in space and time. First we apply a finite difference space discretization on a spatial grid that is staggered in both the horizontal and the vertical direction. The computational domain is covered by an nx·ny·ns rectangular grid. On this grid the spatial derivatives are replaced by second-order finite differences, which leads to a semi-discretized system of dimension nx·ny·(3ns+1).

The time discretization of this semi-discrete system is performed by the numerical method developed in [3]. Our time splitting method is unconditionally stable and consists of two stages. At both stages a system of equations has to be solved. The structure of these systems determines the efficiency of the numerical method. To solve these systems, a Jacobi-type iteration method is used at the first stage, whereas at the second stage a conjugate gradient method is used. The time splitting has been chosen in such a way that in the horizontal direction the computations are independent of each other. For example, the Jacobi-type iteration method requires the solution of nx·ny independent tridiagonal systems, all of dimension ns. In [3] it is reported that this method can be implemented efficiently on an Alliant FX/4.

8.3. IMPLEMENTATION

We now discuss the implementation of this time splitting method on the CRAY Y-MP4/464. A DO-loop in our code may be of the form

```
      DO 100 K=2,NS
        DO 100 J=1,NY
          DO 100 I=1,NX
100         A(I,J,K) = A(I,J,K) + A(I,J,K-1) * B(K)
```

In this example, the I- and J-loop do not show any dependencies, but the K-loop does. The loops with indices I and J are collapsed into a single DO-loop to obtain a longer vector length. This yields

```
      DO 100 K=2,NS
        J=1
        DO 100 I=1,NX·NY                                    (3.1)
100     A(I,J,K) = A(I,J,K) + A(I,J,K-1) * B(K)
```

On the CRAY Y-MP4/464, loop collapsing of simple operations is performed automatically by the compiler. However, for more complicated DO-loops this is not the case. Therefore, the code was collapsed by hand, instead of relying on the compiler.

	ALLIANT FX/4		CRAY Y-MP4/64	
	Mflops	speed-up	Mflops	speed-up
no opt.	0.2		3.2	
		> 3		> 5
scalar opt.	0.7		15.0	
		> 3		> 10
vector opt.	2.0		148.9	

Table 4.1. Mflops rates and speed-ups.

8.4. SCALAR AND VECTOR PERFORMANCE

In Table 4.1 we list the scalar and vector performance of our code on the CRAY Y-MP4/464 and on the Alliant FX/4. The speed-up due to scalar optimization and vectorization is satisfactory on both computers. Both compilers have no problems with vectorizing the DO-loops. It should be noted that Table 4.1 contains results for the complete code. The code not only consists of vectorizable instructions, but there are also non-vectorizable instructions like subroutine calls,

which decrease the performance. For the vectorized code, the computation time on the CRAY Y-MP4/464 is seventy-five times smaller than for the Alliant FX/4. Notice that the ratio of the clock cycle times is only twenty-eight. On the Alliant FX/4 we obtain 15% of the peak performance, whereas on CRAY Y-MP4/464 we obtain 40%. Taking into account the performance of elementary operations on both computers (see [4,5]), we are satisfied with the performance of our code. In [4] it is also reported that the Alliant FX/4 performs considerably below its maximum level (≤40% of the theoretical maximum only). This is due to the insufficient bandwidth from memory to the functional units and/or from cache to the functional units.

8.5. PARALLELISM

On the CRAY Y-MP4/464, parallel processing can be accomplished by three techniques: macrotasking, microtasking and autotasking. Macrotasking allows a program to be partitioned into several *tasks* at the subroutine level. The programmer inserts library routine calls into the code to initiate and synchronize tasks that can be executed in parallel. Macrotasking works best when the amount of work to be partitioned over the processors is large. Microtasking, however, allows parallelism at the DO-loop level. The programmer identifies parallel regions in the code and inserts compiler directives accordingly. Then, at execution, the DO-loop iterations are spread over the processors, which is called strip-mining. Autotasking is a technique that shares the advantages of microtasking, while adding several new advantages. It identifies and exploits parallel regions in a program through the use of a preprocessor. This process may be completely automatic. However, the help of the programmer may be useful since not all types of parallelism can be detected by the compiler.

When applying macrotasking to an existing code, a significant amount of code restructuring may be necessary, which can introduce new errors. It requires a careful partitioning in equal-sized parallel tasks to obtain a good load balancing. On the other hand, it is relatively easy to implement microtasking, which yields an automatic load balancing. The success of the microtasking strategy depends on the synchronization overhead.

As mentioned earlier, the DO-loops in our code have been collapsed explicitly (see (3.1)). This maximizes the vector efficiency while it allows parallelism by the strip-mining technique. In our first experiment with parallelism on the CRAY Y-MP4/464 we implemented autotasking. We hoped that strip-mining of the innermost DO-loops was exploited automatically. However, for our code this is not the case. Thus, we did not obtain any speed-up by autotasking. It should be mentioned that on the Alliant FX/4 the collapsed DO-loops are automatically performed in a vector-parallel mode.

In the second experiment we applied the microtasking technique. By inserting the CMIC$ DO GLOBAL LONG VECTOR directive the innermost DO-loops were strip-mined at execution. However, we obtained incorrect numerical results. Using the microtasking directives, it appears that parallel processing always starts at the first executable statement of the subroutine and always ends at the last executable statement of the subroutine. Consequently, not only the microtasked DO-loop, but the complete subroutine was performed in parallel. To circumvent this unpleasant

microtasking property, we implemented autotasking directives instead of microtasking directives. Thus, we used autotasking but not in an automatic way. With the CMIC$ DO ALL VECTOR AUTOSCOPE autotasking directive we obtained a strip-mining of the innermost DO-loops, while the numerical results remained correct.

8.6. NUMERICAL RESULTS

In this section we present the results for two different test problems. In the first experiment the computations have been performed on a grid with nx=61, ny=13 and ns=10. In this case, we have a vector length of about 800. In the second experiment we have used a grid with 121, 49 and 10 grid points in the three spatial directions, respectively, yielding a vector length of about 6000. The main reason for the second experiment is to investigate the performance on the CRAY Y-MP4/464 for other vector lengths. In order to prevent many more iterations required by the iteration methods, we have increased in the latter experiment the dimensions of the geometry with a factor four. Consequently, in both experiments we obtain the same mesh sizes. In Table 6.1 we list the results for the experiments in which we introduced parallelism by strip-mining of the innermost DO-loops.

problem size	*61·13·10*	*121·49·10*
vector code	17.6 s	137.4 s
parallel code	11.6 s	53.0 s
speed-up	1.5	2.6

Table 6.1. Computation times for the CRAY Y-MP4 by strip-mining.

For the smallest test problem, we obtain a speed-up of 1.5 on four processors, which is disappointing. For the second and largest test problem, the speed-up of 2.6 is still small despite of its large vector length. Thus, especially for small vector lengths the synchronization overhead on the CRAY Y-MP4/464 appears to be considerable.

To obtain a higher efficiency on the CRAY Y-MP4/464, we have implemented a domain decomposition technique. In the x-direction our domain has been split into four subdomains. With the CMIC$ DO PARALLEL autotasking directive the computations for the four subdomains have been performed in parallel. We have applied autotasking instead of macrotasking, because it requires less programming effort while the computation times are comparable.

A decomposition in the x-direction has been chosen to minimize the restructuring of the code. A slightly more complicated decomposition would have been a decomposition of both the x- and y-direction into two parts, as used in [1].

Furthermore, we mention that for a spectral method a domain decomposition in the vertical direction has been applied in [2].

In our opinion, a domain decomposition in the horizontal direction is a good choice for the numerical method used in this paper, because of the many independent computations in the horizontal direction. In time-consuming 3D shallow water models, the value of nx·ny is very large, whereas the value of ns may still be rather small. Moreover, the water elevation and the vertical velocity ω are computed from vertical integrals (see (2.3)-(2.4)) and implicit relations, arising from an implicit treatment of the vertical diffusion term, have to be solved [3]. Therefore, it seems not a good idea to apply a domain decomposition in the vertical direction. This would lead to a lot of communication between the processors.

In Table 6.2 we list for our two test problems the results for the domain decomposition approach. We now have vector lengths of 200 and 1500, respectively, which is one fourth of the vector lengths used in the former experiment. The overall speed-up is defined in the following way:

$$\frac{\text{time for the original vectorized code}}{\text{time for the parallel code with domain decomposition}}$$

The computation time for the original vectorized code is listed in Table 6.1, whereas computation time for the domain decomposition technique is presented in Table 6.2.

problem size	*61·13·10*	*121·49·10*
vector code	20.7 s	141.2 s
parallel code	5.6 s	38.1 s
decomposition speed-up	3.7	3.7
overall speed-up	3.1	3.6

Table 6.2. Computation times for the CRAY Y-MP4 by domain decomposition.

For both test problems, we now obtain a speed-up of 3.7 on four processors. Because of the synchronization overhead we do not obtain an optimum speed-up of four. The difference in overall and domain decomposition speed-up is due to the fact that vector lengths are now a factor of four smaller. In the case of very large vector lengths, this hardly yields a decrease in performance. For the smallest test problem, the reduction is about 20%. The results in Table 6.2 clearly show that the domain decomposition approach gives rise to larger speed-ups than the strip-mining technique.

8.7. Conclusions

In this paper we have investigated the performance of a numerical method for the three-dimensional shallow water equations on the CRAY Y-MP4/464. On one processor the highly vectorizable code yields a performance of about 150 Mflops, which is no fewer than seventy-five times faster than on the Alliant FX/4. To exploit parallelism, we have applied a domain decomposition approach, since in general the strip-mining of the innermost DO loops does not give satisfactory speed-ups, even in the case of large vector lengths.

The experiments show that the CRAY Y-MP4/464 is very suitable for the time-consuming simulation of three-dimensional shallow water flows. With such supercomputers it is possible to simulate realistic models, e.g., including equations for salinity, temperature and turbulence, with a fine resolution. The vector performance is very impressive. When the CRAY Y-MP4/464 is not heavily loaded, an acceptable speed-up due to parallelism can be obtained.

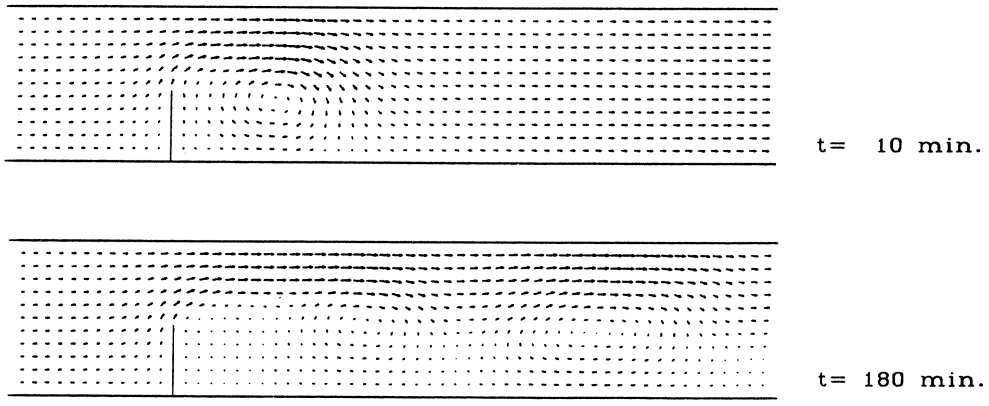

Figure 1. Flow past a jetty.

References

1. P. Andrich, G. Madec and D. L'Hostis, Performance evaluation for an ocean general circulation model: vectorization and multitasking, *Proceedings of the International Conference on Supercomputing*, St. Malo, 295-303 (1988).
2. A.M. Davies, R.B. Grzonka and M. O'Neill, Development and application of 3D shallow sea models on the CRAY computer, *Proceedings of the 5th International Symposium of Science and Engineering on Supercomputers*, London, 323-334 (1990).
3. E.D. de Goede, A time splitting method for the three-dimensional shallow water equations, *Int. J. Numer. Meth. in Fluids*, 13, 519-534 (1991).
4. J.M. van Kats and A.J. van der Steen, *Benchmarktests on an Alliant FX/4, a Convex C-1, an FPS 64/60 and an SCS-40*, report TR-24, ACCU, Utrecht, 1987.

5. A.J. VAN DER STEEN AND R.J. VAN DER PAS, *A family portrait: Benchmarktests on a CRAY Y-MP and a CRAY-2S*, report TR-30, ACCU, Utrecht, 1989.
6. G.S. STELLING, *On the construction of computational methods for shallow water flow problems*, Ph.D. Thesis, Delft University, 1983.

Chapter 9

A Numerical Model of the Northwest European Continental Shelf on the CRAY Y-MP2E

E.D. de Goede
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009AB Amsterdam, The Netherlands

A numerical model for the shallow water equations in polar co-ordinates is applied to the northwest European Continental Shelf. A three-dimensional model of the Continental Shelf is presented. A simulation is carried out for the period of 9 to 12 February 1989. The same Continental Shelf problem has been examined by Leendertse. The numerical results are in good agreement with each other. It appears that the amplitude and phase errors are small. We also describe a comparison with a two-dimensional model of the Continental Shelf.
The time splitting method used in this paper consists of two stages and is unconditionally stable. Moreover, it can fully exploit vector and parallel facilities of supercomputers. Three-dimensional shallow water models require a great computational effort and it is therefore necessary to use such fast computers.
The numerical experiments are carried out on the one-processor CRAY Y-MP2E of ICIM (Informatics Centre for Civil Engineering and Environment). This supercomputer has recently been installed in the Netherlands for the simulation of large scale models of rivers and seas.

9.1. INTRODUCTION

In the past, various two-dimensional numerical models have been developed for the northwest European Continental Shelf (e.g., in [3, 6, 12, 19]). Their main goal is the accurate prediction of the water levels. Presently, a two-dimensional model of the European Continental Shelf is being used for storm surge predictions along the Dutch coast. This model is operational at KNMI (Royal Dutch Meteorological Institute). Using wind and atmospheric pressure data from a numerical model of the atmosphere, the water elevations in the North Sea and especially along the Dutch coast are computed four times a day. Since these models are two-dimensional, they cannot be used to examine the vertical structure of the tidal currents.

In recent years, there has been a shift towards three-dimensional models to obtain more detailed information about the tidal currents. Three-dimensional models for the Continental Shelf can be found in e.g., [1,4,16]. With such three-dimensional models the interaction between wind and tides can be computed more accurately.

In this paper the fully nonlinear three-dimensional shallow water equations in polar co-ordinates are used. These hydrodynamical equations are solved by the numerical method described in [9]. This two-stage time splitting method is unconditionally stable. For the discretization in space, finite differences are used in both the horizontal and the vertical direction.

In [10] it was reported that this time splitting method can be implemented efficiently on a CRAY Y-MP4/464. In that paper a rectangular basin was investigated. Here, we examine the accuracy and computational efficiency of our method on an irregular domain. We use the geometry of the northwest European Continental Shelf with mesh sizes of about 16 km (the so-called CSM16 model). The input data (geometry, boundary conditions, depth values and Chezy coefficients) were supplied by the Tidal Waters Division of Rijkswaterstaat (Dutch Water Control and Public Works Department). The meteorological input (i.e., wind and atmospheric pressure) is neglected. Both a two-dimensional and a three-dimensional model are examined. A simulation is carried out for the period of 9 to 12 February 1989. This test problem has been examined in [15] and therefore we can compare the numerical results.

The numerical experiments are carried out on the CRAY Y-MP2E installed at ICIM (Informatics Centre for Civil Engineering and Environment). Since May 1991 this supercomputer has been used in the Netherlands for the simulation of estuarine, river and sea models. This CRAY Y-MP2E has one (vector-)processor and a clock cycle time of 6 ns. On this supercomputer the performance of our highly vectorizable numerical method is about 140 Mflops (millions of floating point operations per second), which is very satisfying. The four day simulation of the three-dimensional model requires about 157 seconds. In [15] a similar experiment on a PC with a 80386 chip and a mathematical co-processor requires a computation time of about 12 hours.

9.2. MATHEMATICAL MODEL

For the northwest European Continental Shelf model the three-dimensional shallow water equations are written in the form [4]

$$\frac{\partial u}{\partial t} = -\frac{u}{R\cos\varphi}\frac{\partial u}{\partial \chi} - \frac{v}{R}\frac{\partial u}{\partial \varphi} - \omega\frac{\partial u}{\partial \sigma} + 2\omega_e \sin\varphi\, v + \frac{uv\tan\varphi}{R} - \frac{g}{R\cos\varphi}\frac{\partial \zeta}{\partial \chi} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial u}{\partial \sigma}\right) \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -\frac{u}{R\cos\varphi}\frac{\partial v}{\partial \chi} - \frac{v}{R}\frac{\partial v}{\partial \varphi} - \omega\frac{\partial v}{\partial \sigma} - 2\omega_e \sin\varphi\, u - \frac{u^2\tan\varphi}{R} - \frac{g}{R}\frac{\partial \zeta}{\partial \varphi} + \frac{1}{h^2}\frac{\partial}{\partial \sigma}\left(\mu\frac{\partial v}{\partial \sigma}\right) \tag{2.2}$$

$$\omega = \frac{1}{h}\left\{ \frac{\partial \zeta}{\partial t}(1-\sigma) + \frac{1}{R\cos\varphi}\frac{\partial}{\partial \chi}\left(h\int_{\sigma}^{1} u\,d\sigma\right) + \frac{1}{R\cos\varphi}\frac{\partial}{\partial \varphi}\left(h\int_{\sigma}^{1} v\,d\sigma\cos\varphi\right)\right\} \tag{2.3}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{1}{R\cos\varphi}\frac{\partial}{\partial \chi}\left(h\int_0^1 u\,d\sigma\right) - \frac{1}{R\cos\varphi}\frac{\partial}{\partial \varphi}\left(h\int_0^1 v\,d\sigma\cos\varphi\right). \qquad (2.4)$$

Since the Continental Shelf covers a wide area (see Figure 1), the equations have been transformed from Cartesian into polar co-ordinates. The equations (2.1)-(2.3) are the momentum equations and (2.4) denotes the continuity equation. In our model the accelerations in the vertical direction have been neglected, because they are very small, particularly when compared with the acceleration due to gravity. This is known as the so-called shallow water approximation.

In the vertical, the domain is bounded by the bottom topography and the time-dependent water elevation. To ensure that the three-dimensional domain is constant in the vertical direction, system (2.1)-(2.4) has been transformed into the constant interval [0,1] by the sigma transformation [17]

$$\sigma = \frac{\zeta - z}{d + \zeta}, \quad \text{where } -d \le z \le \zeta \text{ and } 1 \ge \sigma \ge 0. \qquad (2.5)$$

The relation between the untransformed (physical) vertical velocity w and the transformed velocity ω is given by [4,20]

$$w = -\omega h + \frac{\partial \zeta}{\partial t} - \sigma\frac{\partial h}{\partial t} + \frac{u}{R\cos\varphi}\left(\frac{\partial \zeta}{\partial \chi} - \sigma\frac{\partial h}{\partial \chi}\right) + \frac{v}{R}\left(\frac{\partial \zeta}{\partial \varphi} - \sigma\frac{\partial h}{\partial \varphi}\right),$$

which leads to [4]

$$w = -\frac{1}{R\cos\varphi}\frac{\partial}{\partial \chi}\left(h\int_\sigma^1 u\,d\sigma\right) - \frac{1}{R}\frac{\partial}{\partial \varphi}\left(h\int_\sigma^1 v\,d\sigma\right) + \frac{h\tan\varphi}{R}\int_\sigma^1 v\,d\sigma + \frac{u}{R\cos\varphi}\left(\frac{\partial \zeta}{\partial \chi} - \sigma\frac{\partial h}{\partial \chi}\right) + \frac{v}{R}\left(\frac{\partial \zeta}{\partial \varphi} - \sigma\frac{\partial h}{\partial \varphi}\right).$$

The boundary conditions at the sea surface ($\sigma = 0$) are given by

$$\omega(\chi,\varphi,0,t) = 0, \quad \left(\mu\frac{\partial u}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho}W_f\cos(\phi) \quad \text{and} \quad \left(\mu\frac{\partial v}{\partial \sigma}\right)_{\sigma=0} = -\frac{h}{\rho}W_f\sin(\phi).$$

Similarly, at the bottom ($\sigma = 1$) we have

$$\omega(\chi,\varphi,1,t) = 0, \left(\mu\frac{\partial u}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g u_d}{C^2}\sqrt{u_d^2+v_d^2}, \left(\mu\frac{\partial v}{\partial \sigma}\right)_{\sigma=1} = -h\frac{g v_d}{C^2}\sqrt{u_d^2+v_d^2},$$

where u_d and v_d represent components of the velocity at some depth near the bottom.

9.3. NUMERICAL DISCRETIZATION

To discretize system (2.1)-(2.4), we first apply a finite difference space discretization on a spatial grid that is staggered in both the horizontal and the vertical direction. Figure 2 shows the structure of the horizontal grid. The computational domain is covered by an nx·ny·ns rectangular grid. On this grid the spatial derivatives are replaced by second-order finite differences, which results into a semi-discretized system of dimension nx·ny·(3ns+1). Owing to the sigma transformation (2.5), we have a constant number of grid layers in the vertical direction.

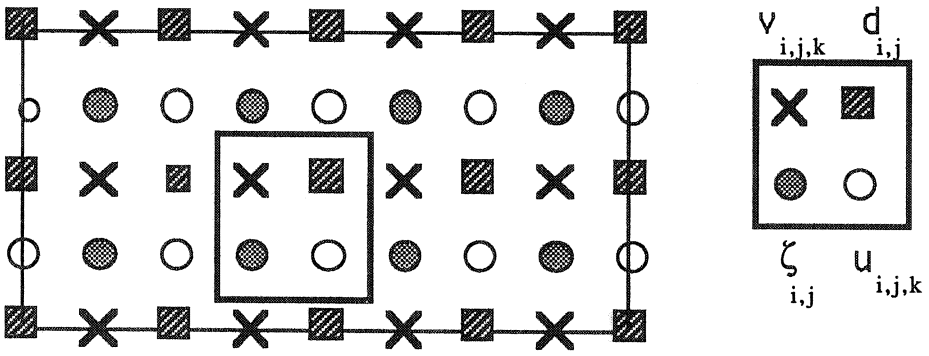

Figure 2. The staggered grid in the horizontal direction.

We now briefly describe the time integration method for the semi-discrete system. For a detailed description we refer to [9]. The first stage of the two-stage time splitting method requires the successive solution of two non-symmetric, linear systems of dimension nx·ny·ns (for the u- and v-component, respectively). This system is solved by a Jacobi-type method, which offers the facility of both an explicit and an implicit treatment of the advective terms. At the second stage a nonlinear system has to be solved. A linearization process is introduced and the resulting linear systems are solved by a preconditioned conjugate gradient method.

In [8] it has been proved that this method is unconditionally stable for a model in which the advective terms and the Coriolis term have been omitted. For a model including these terms, our method appears to have the same good stability properties [9].

The time integration method is first-order accurate in time. It is possible to obtain second-order accuracy by adjusting the discretization of the Coriolis term, the mixed advective terms and the bottom friction term. However, such a second-order treatment would decrease the computational efficiency dramatically. For the sake of efficiency we therefore decided to only use a first-order discretization. It should be noted that the diffusion terms and the terms describing the propagation of the surface waves are second-order accurate in time.

Almost all spatial derivatives are discretized in a symmetric and therefore non-dissipative way. However, following the approach of Stelling [18], at the first stage the mixed advective terms $v\partial u/(R\partial\varphi)$ and $u\partial v/(R\cos\varphi\,\partial\chi)$ are approximated by an

upwind discretization. The resulting dissipation is of fourth-order magnitude and does on the one hand not lead to an undesirable damping of the solution and is on the other hand just enough to suppress spurious oscillations.

9.4. Implementation on vector computers

It is well-known that long vectors are crucial for an efficient use of vector computers. The time splitting method discussed in the previous section has been constructed in such a way that in the horizontal direction the computations are independent of each other [9]. For example, at the first stage the Jacobi-type method requires the solution of nx·ny independent tridiagonal systems, all of dimension ns. Thus, a long vector length of nx·ny is obtained by solving the tridiagonal systems at the same time. In [10] the computational efficiency of this method has been demonstrated on a CRAY Y-MP4/464 for a *rectangular* domain.

We now discuss the implementation on vector computers for *irregular* domains. On such domains, we may perform the computations on a surrounding rectangular domain, which contains both sea and land regions. At the end of each time step, the values in the land regions should be neglected. Then, direct addressing can be used which again leads to vector operations over the whole domain. On the other hand, one may strip out the sea regions and only solve the shallow water equations in these regions (i.e., indirect addressing). Although this leads to shorter vectors, no additional operations are required in the land regions.

We have chosen the direct addressing approach, because on many computers the performance for direct addressing is significantly higher. Obviously, as the land to sea ratio increases, then the indirect addressing technique will become more attractive. On such domains we propose a domain decomposition approach in the horizontal to obtain a better ratio of sea to land regions.

9.5. Application

In this section we examine both a two-dimensional and a three-dimensional model of the northwest European Continental Shelf. This model covers the same computational grid as the CSM16 model (average mesh size of about 16 km) of Rijkswaterstaat. The input data (geometry, boundary conditions, depth values and Chezy coefficients) were supplied by the Tidal Waters Division of Rijkswaterstaat. The boundaries of the model are parallel to the geographical co-ordinates 48° N, 62°20' N, 12° W and 13° E. Along the north, west and south (open) boundaries, water elevations are prescribed (see Figure 1). A simulation is carried out for the period of 9 to 12 February 1989.

For our model in polar co-ordinates (see (2.1)-(2.4)), we choose $\Delta\chi=1/4°$ and $\Delta\varphi=1/6°$, where $\Delta\chi$ and $\Delta\varphi$ denote the grid sizes in χ- and φ-direction, respectively. This leads to a mesh size of 18553 m in the north-south direction and to mesh sizes ranging from 12922 m to 18621 m in the east-west direction. The other parameters are $g=9.81$ m/s^2, $W_f=0$ kg/ms^2 (thus, no wind), $\rho=1025$ kg/m^3, $\omega_e=7.27e\text{-}5$ s^{-1} and $R=6378000$ m. The water is initially at rest and the motion on the Continental Shelf is generated by the water elevations prescribed along the open boundaries. These (weakly reflective) boundary conditions allow disturbances from the interior of the model to propagate outwards.

The computations are performed on a grid with nx=100 and ny=87. In the vertical direction we use various values for ns, ranging from 1 (thus, a two-dimensional experiment) to 25. In the two-dimensional case, the number of active grid cells, which represent sea regions, is about 5500.

In the three-dimensional case, it is necessary to specify how the vertical diffusion coefficient μ varies with χ, φ, σ and t. The horizontal variation of μ over the North Sea has been investigated in [13]. Using these results, an appropriate parametrization of the vertical eddy viscosity is

$$\mu = c \left\{ \left(\int_0^1 u d\sigma \right)^2 + \left(\int_0^1 v d\sigma \right)^2 \right\},$$

where c is some parameter. In our experiments we choose c=0.4 s, which is twice the value used in [4].

The numerical experiments are carried out on the one-processor CRAY Y-MP2E installed at ICIM. Since May 1991 this supercomputer has been operational for the simulation of large scale water models.

Our unconditionally stable time splitting method offers the facility of both an explicit and implicit treatment of the advective terms. In this Continental Shelf test problem the influence of the advective terms is limited. An explicit treatment of these terms is sufficient. Therefore, in this paper we will only show results for the case in which an explicit treatment is used.

We also compare the numerical solution with a reference solution computed on the same grid with a very small time step of 30 s. Thus, the difference between the reference solution and numerical solution computed with a larger time step represents the error due to the time integration. The spatial accuracy can not be examined, because the input data are only available for one fixed grid.

The numerical results are examined in the following stations: Wick, Aberdeen, Cromer, Innerdowsing, Dover, Le Havre, Dieppe, Boulogne, Calais, Oostende, Zeebrugge, Hoek van Holland, IJmuiden, Den Helder and Borkum. These stations have been used in [15] too.

To represent the results we define

ERROR	: absolute amplitude error for the water elevation ζ averaged over the time and over the stations
COMP	: computation time on the CRAY Y-MP2E.

In Table 5.1 we list the amplitude errors and computation times for various values of the time step τ. Moreover, the computation time is subdivided into the time required by the (three-dimensional) velocities and the time required by the (two-dimensional) water elevation ζ.

Even for a large time step of 1200 s the amplitude errors are small. In the three-dimensional experiment the errors are about twice as large. The numerical results show that the phase errors for our time integration method are also small.

The computational complexity for the water elevation is the same for both two-dimensional and three-dimensional models [8]. Therefore, the computation time for the water elevation in Table 5.1 (i.e., column COMP ζ) is more or less independent

of the number of layers in the vertical direction. The computational costs per time step increase when τ increases, because of the larger number of iterations required by the conjugate gradient method (cf. Section 9.3). However, the larger the time step, the smaller the number of time steps to be carried out. As a result, the computation time for the water elevation hardly varies in Table 5.1. The computation time for the velocities (i.e., column COMP u,v,ω) is proportional to the number of vertical grid layers and is inversely proportional to the time step.

τ	**ns=10** (3D)				**ns=1** (2D)			
	ERROR (m)	COMP total (s)	COMP u,v,ω (s)	COMP ζ (s)	ERROR (m)	COMP total (s)	COMP u,v,ω (s)	COMP ζ (s)
300	0.01	282.3	245.5	36.8	0.005	57.3	24.2	33.1
600	0.02	157.2	121.4	35.8	0.01	44.6	12.2	32.4
900	0.03	116.6	80.1	36.5	0.015	41.9	8.1	33.8
1200	0.04	101.9	61.8	40.1	0.02	42.8	6.1	36.7

Table 5.1. Amplitude errors and computation times.

In Figures 3.1 to 3.15 we show the water elevations for the aforementioned stations during the fourth day of the simulation. Both two-dimensional (i.e., ns=1) and three-dimensional results (with ns=10) are given. For the two-dimensional case, the same experiment has been carried out in [15]. The numerical results are in good agreement. In [15] the three-dimensional mathematical model contains an additional equation for the turbulent energy. Owing to this different parametrization of the vertical eddy viscosity, the numerical results can not be compared for the three-dimensional case.

Some oscillations have been observed at the station Wick (see Figure 3.1). These oscillations are due to the choice of the vertical diffusion coefficient. In the case of a constant value for the vertical diffusion we have not observed such oscillations. If we integrate over a longer period than four days, then these oscillations disappear.

Figures 3.1 to 3.15 show small differences between the two-dimensional and the three-dimensional results. In the vertical direction we have observed only a small variation of the tidal currents. At the bottom the velocities are approximately ten percent smaller than at the water surface. For such test problems one may equally well apply two-dimensional models instead of three-dimensional ones in order to estimate the water elevations. However, in order to take into account more detailed physics (e.g., near the sea bottom) three-dimensional models are essential [5,16].

In the experiments we have used both the scalar and the vector optimization of the CRAY Y-MP2E. For our integration method the computation time reduces by about a factor of five due to the scalar optimization, and by an additional factor of

nine due to the vectorization. This again shows this method can be implemented efficiently on vector computers. The gain factor is more or less independent of the number of layers in the vertical direction. The performance of our time splitting method is approximately 140 Mflops. Owing to the direct addressing approach (see Section 9.4), which introduces operations in land regions, the Mflops rate is somewhat misleading. Therefore, we will consider computation times in the next paragraphs.

The WAQUA system, which is a widely applied package in the Netherlands for the simulation of (two-dimensional) river and sea flows, has been implemented on vector–parallel computers too (see e.g., [19]). On the CRAY Y-MP2E a similar (two-dimensional) Continental Shelf experiment as in this paper (with a time step of 600 s) requires about 46 s and yields a performance of about 40 Mflops. Thus, our method has a higher performance on the CRAY Y-MP2E of about a factor of 3.5. However, for this two-dimensional problem the computation times are comparable (46 s vs. 44.6 s), because our time splitting method requires about 3 to 5 times more operations [21]. It should be noted that our numerical method has been developed for three-dimensional models and is relatively more efficient for three-dimensional than for two-dimensional models (see e.g., Table 5.1 and [8]).

For our three-dimensional Continental Shelf experiment we could not find any computation times on vector-parallel computers in the literature. For example, in [15] the three-dimensional Continental Shelf experiment, which has been carried out on a PC with a 80386 chip, requires a computation time of about 12 hours. To illustrate the computational efficiency of our numerical method we therefore make a comparison with the conditionally stable method described in [7]. This method is representative for various numerical methods described in the literature (see e.g., [2,14]). We remark that the first stage of the (unconditionally stable) method used in this paper is very similar to the method in [7]. In Table 5.2 the computation times are presented for the unconditionally stable method in [9] and the conditionally stable method in [7].

	TIME SPLITTING METHOD (s)	CONDITIONALLY STABLE METHOD (s)
ns=5	99.8 (τ=600)	425.5 (τ=50)
ns=10	1532.0 (τ=50)	
ns=10	157.2 (τ=600)	836.2 (τ=50)
ns=10	101.9 (τ=1200)	
ns=25	342.8 (τ=600)	2056.0 (τ=50)

Table 5.2. Computation times (in s).

In this experiment the accuracy of both methods is comparable. The computation times clearly show that the unconditionally stable method is much more efficient, because larger time steps can be used. For example in the case of ns=10 with τ=1200 s, the ratio in computation time is about a factor of eight.

For the conditionally stable method the advective terms require a great computational effort, which is due to the time step restriction. For time steps larger than 50 s we already encounter stability problems for the conditionally stable method. Instabilities occur in the relatively deep inlet near Glasgow.

In conclusion, the unconditionally stable time splitting method used in this paper turns out to be a robust and efficient method for the three-dimensional shallow water equations (see also [10,11]).

REFERENCES

1. J.O. BACKHAUS AND D. HAINBUCHER, A finite difference general circulation model for shelf seas and its application to low frequency variability on the north European Shelf, *Proceedings of three-dimensional models of marine and estuarine dynamics (ed. J.C.J. Nihoul and B.M. Jamart)*, 221-244 (1987).
2. A.M. DAVIES, Application of the DuFort-Frankel and Saul'ev methods with time splitting to the formulation of a three dimensional hydrodynamic sea model, *Int. J. Numer. Meth. in Fluids*, 5, 405-425 (1985).
3. A.M. DAVIES, A numerical model of the North Sea and its use in choosing locations for the deployment of off-shore tide gauges in the JONSDAP '76 Oceanographic experiment, *Dtsch. Hydrogr. Z.*, 29, 11-24 (1976).
4. A.M. DAVIES, A three-dimensional model of the northwest European Continental Shelf, with application to the M_4 tide, *J. Phys. Oceanogr.*, 16, 797-813 (1986).
5. A.M. DAVIES, On the numerical solution of the turbulence energy equations for wave and tidal flows, *Int. J. Numer. Meth. in Fluids*, 12, 17-41 (1991).
6. R.A. FLATCHER, A tidal model of the northwest European Continental Shelf, *Mem. Soc. R. Sci. Liège*, 10, 141-164 (1976).
7. E.D. DE GOEDE, A computational model for the three-dimensional shallow water flows on the Alliant FX/4, *Supercomputer*, 32, 43-49 (1988).
8. E.D. DE GOEDE, A time splitting method for the three-dimensional shallow water equations, *Int. J. Numer. Meth. in Fluids*, 13, 519-534 (1991).
9. E.D. DE GOEDE, On the numerical treatment of the advective terms in 3D shallow water models, *Proceedings of the 2nd symposium on High Performance Computing*, Montpellier, 491-502 (1991).
10. E.D. DE GOEDE, 3D shallow water model on the CRAY Y-MP4/464, *Proceedings of the Sixth International workshop on the Use of Supercomputers in Theoretical Science, Antwerp*, 107-114 (1991).
11. E.D. DE GOEDE, Numerical methods for the three-dimensional shallow water equations, *Appl. Numer. Maths.*, 10, 3-18 (1992).
12. B.M. JAMART and J. OZER, Some results and comments on the tidal flow forum exercise, *Adv. Water Resources*, 12, 211-220 (1989).
13. V.K. KRAAV, Computation of the semidiurnal tide and turbulence parameters in the North Sea, *Oceanology*, 9, 332-341 (1969).

14. J.J. LEENDERTSE, *Aspects of a computational model for long period water wave propagation*, Memorandum RM-5294-PR, Rand Corp., Santa Monica, California, 1967.
15. J.J. LEENDERTSE, *Development of a three-dimensional CSM model*, progress report DGW-OPWA2, 1990.
16. J.J. LEENDERTSE, *A three-dimensional model of the European Continental Shelf*, TRI-FLOW COMP report, California, 1990.
17. N.A. PHILLIPS, A coordinate system having some special advantages for numerical forecasting, *J. Meteorol.*, 14, 184-194 (1957).
18. G.S. STELLING, *On the construction of computational methods for shallow water problems*, Ph.D. Thesis, Delft University, 1983.
19. Th.L. VAN STIJN, *Vectorization of the Continental Shelf Model*, DIV/SWA report 89 005 (1989).
20. TRISULA, A multi-dimensional flow and water-quality simulation system, Delft Hydraulics, The Netherlands, 1989.
21. P. WILDERS, Th.L.VAN STIJN, G.S. STELLING AND G.A. FOKKEMA, A fully implicit splitting method for accurate tidal computations, *Int. J. Numer. Meth. in Eng.*, 26, 2707-2721 (1988).

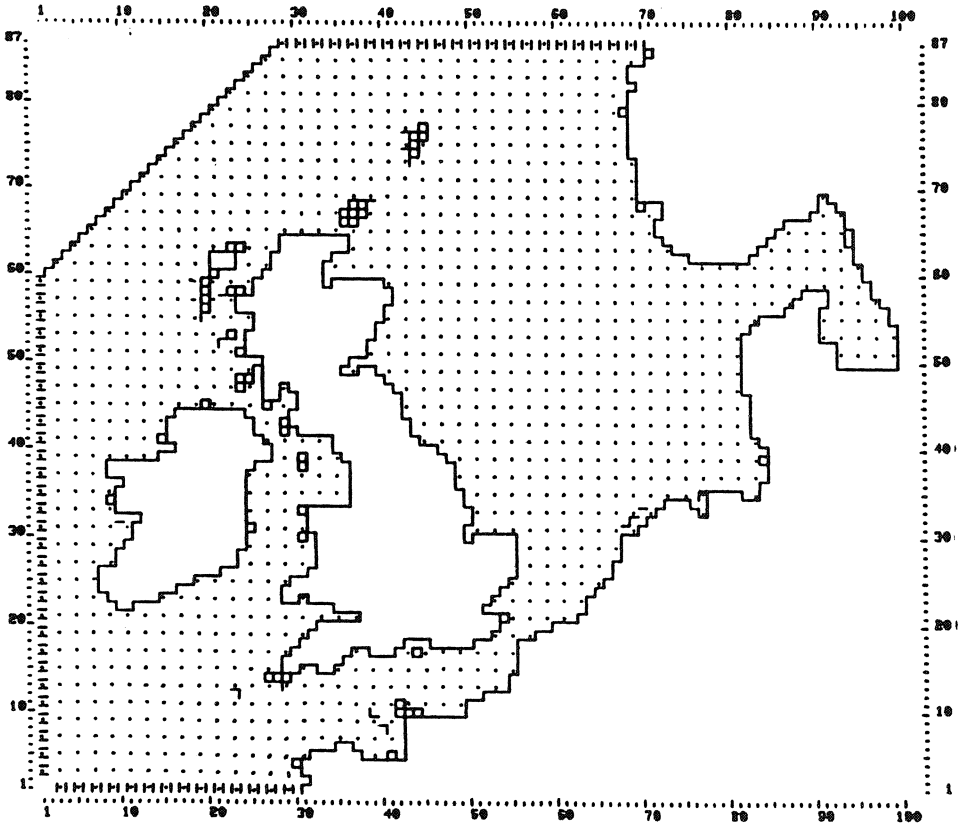

Figure 1. The finite difference grid of the Continental Shelf model.

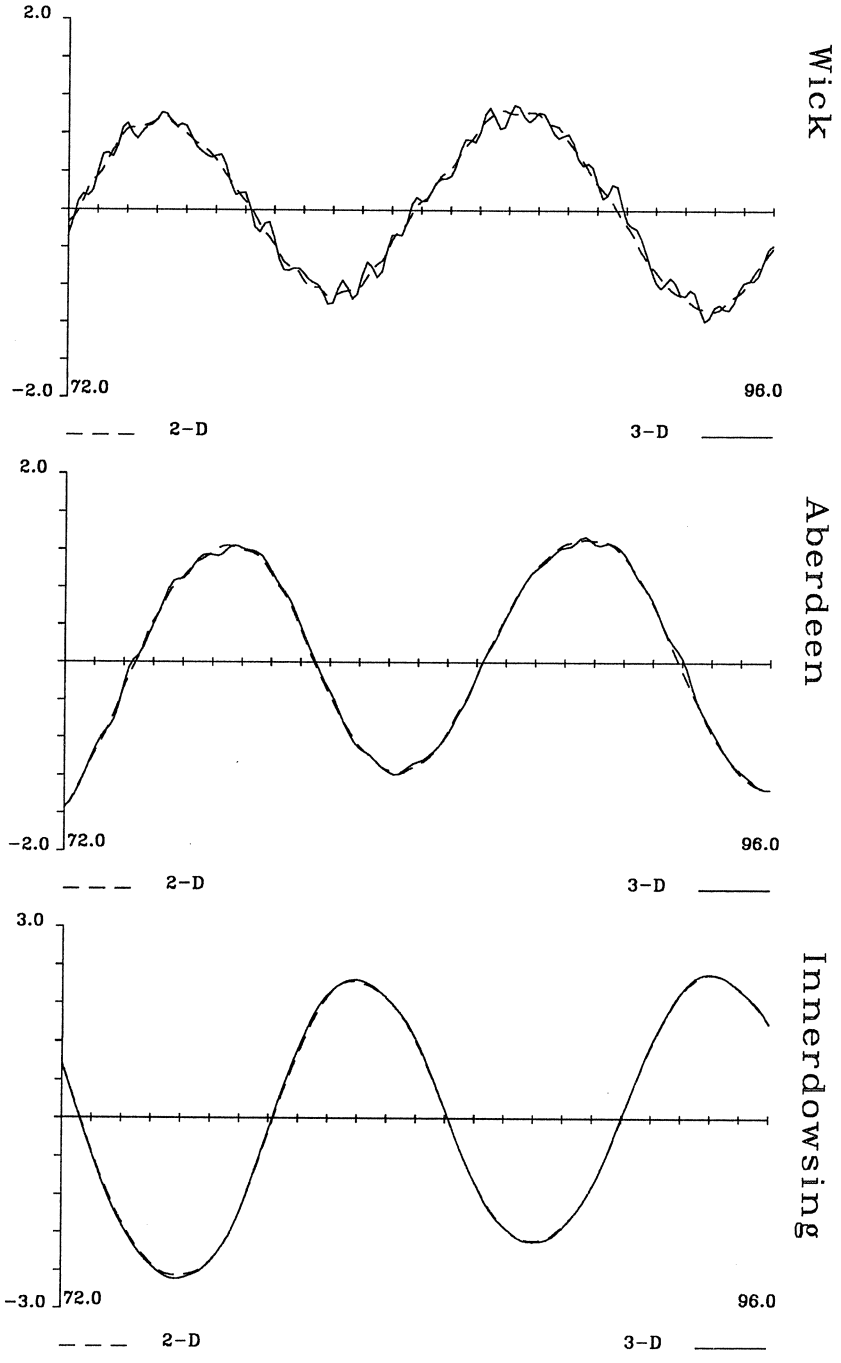
2.0
Wick
-2.0
72.0
96.0
2-D
3-D
2.0
Aberdeen
-2.0
72.0
96.0
2-D
3-D
3.0
Innerdowsing
-3.0
72.0
96.0
2-D
3-D

Figure 3.1–3.3.

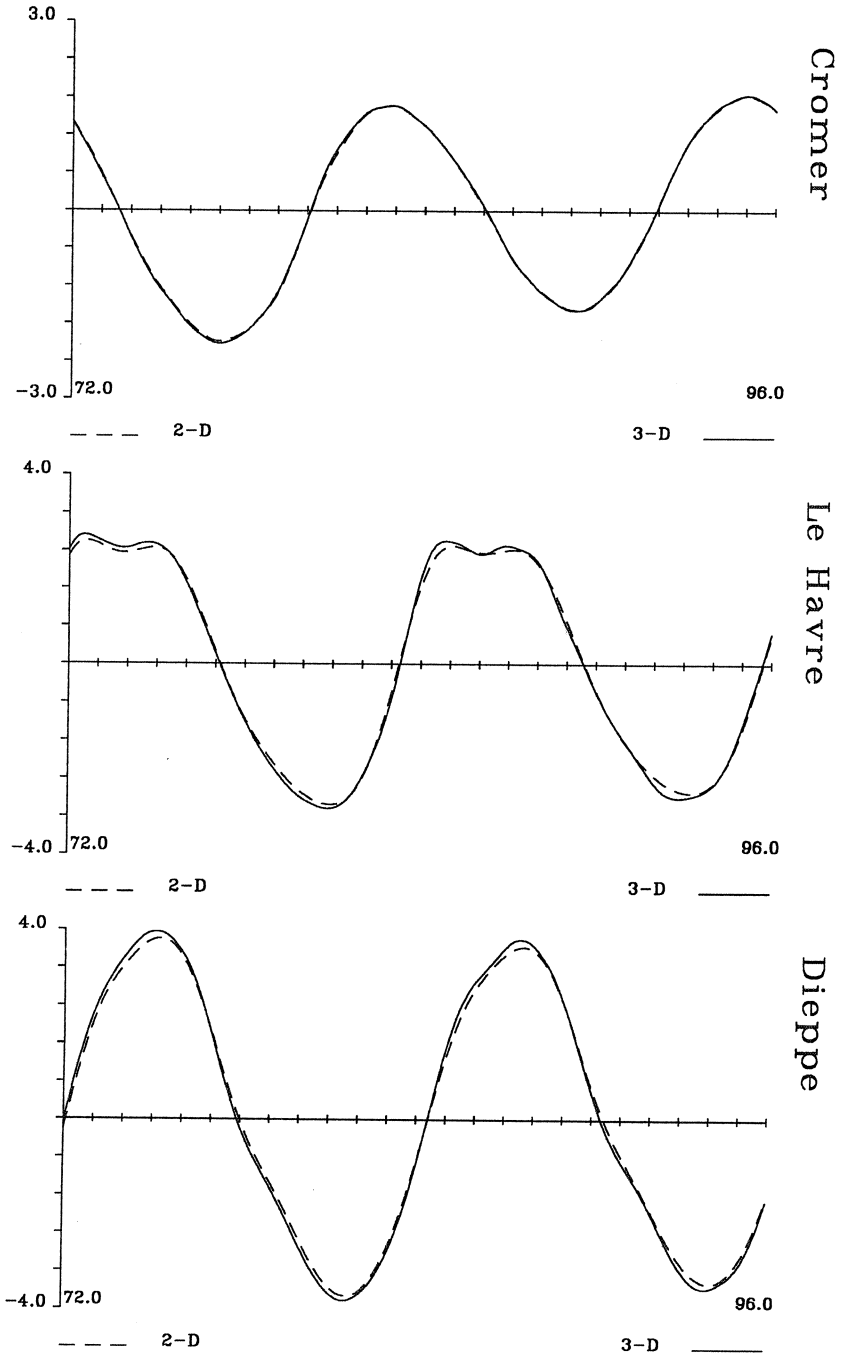
3.0
-3.0
72.0
96.0
2-D
3-D
Cromer
4.0
-4.0
72.0
96.0
2-D
3-D
Le Havre
4.0
-4.0
72.0
96.0
2-D
3-D
Dieppe

Figure 3.4–3.6.

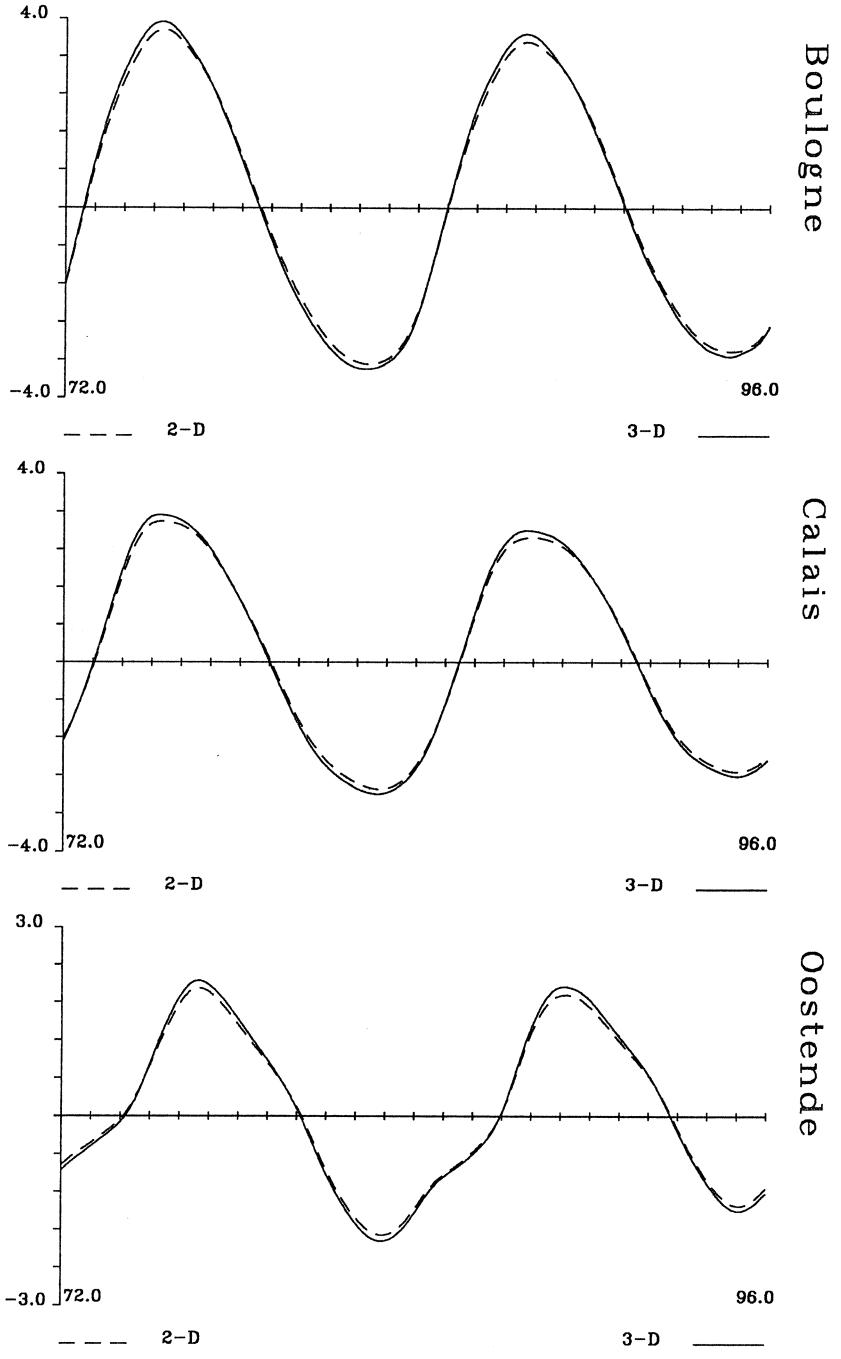
4.0
-4.0
72.0
96.0
2-D
3-D
Boulogne
4.0
-4.0
72.0
96.0
2-D
3-D
Calais
3.0
-3.0
72.0
96.0
2-D
3-D
Oostende

Figure 3.7–3.9.

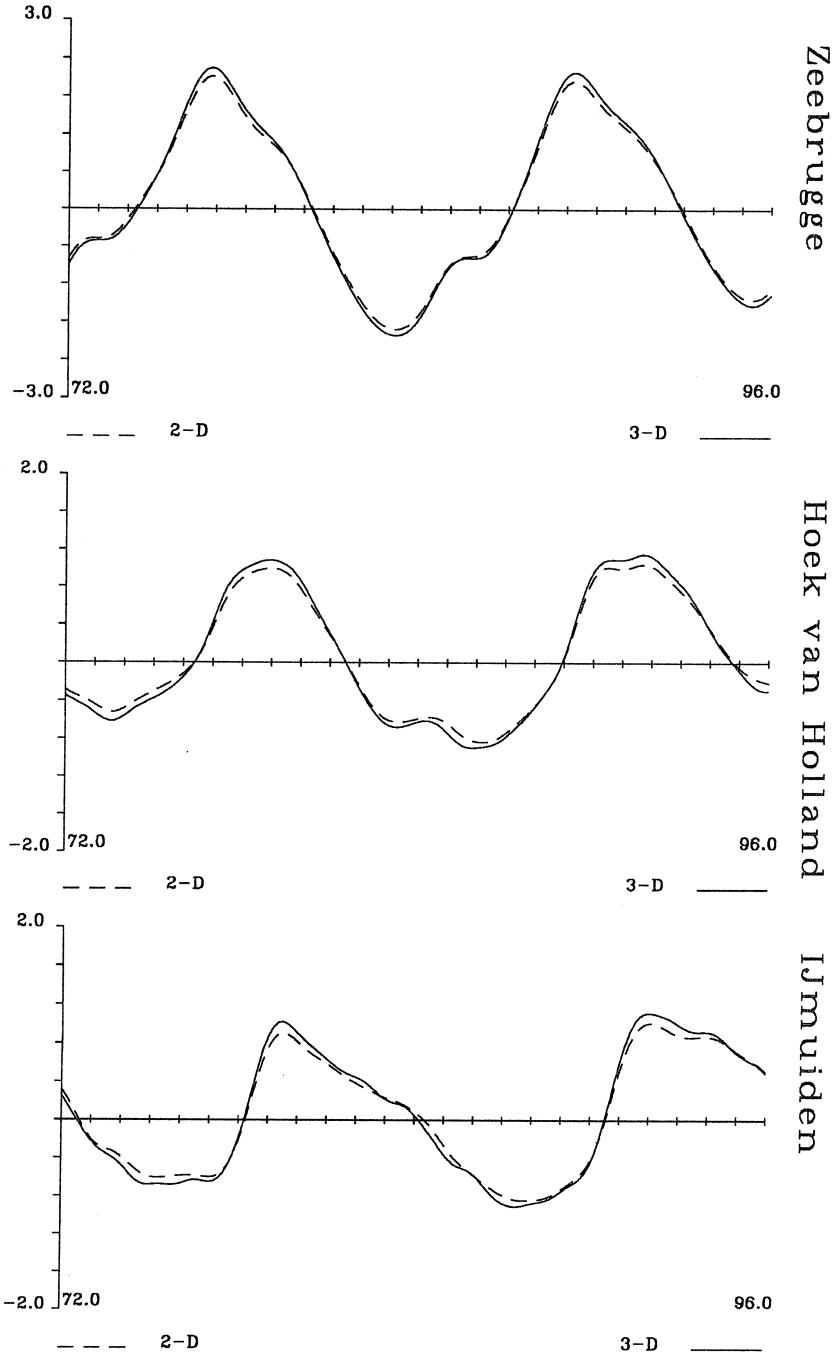

Figure 3.10–3.12.

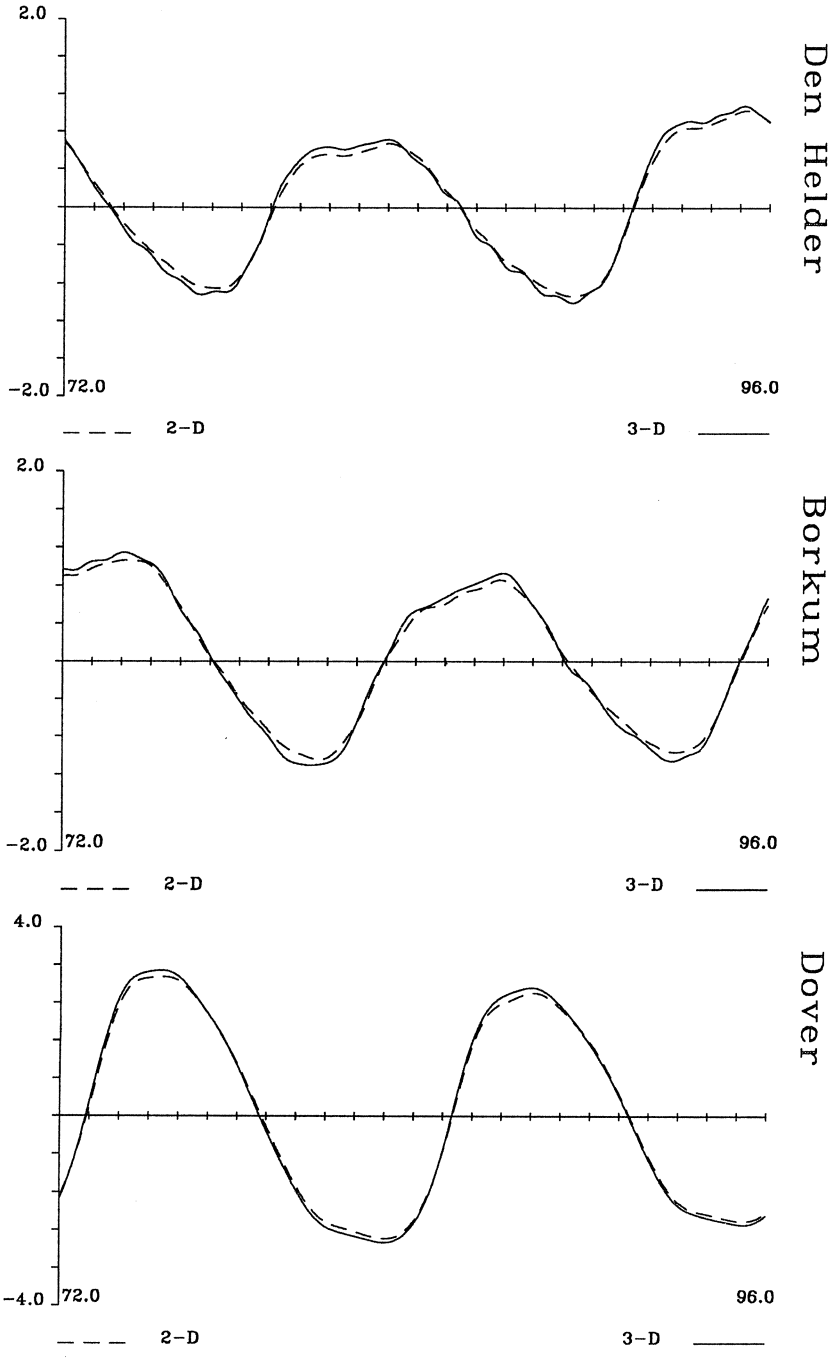

Figure 3.13–3.15.

Chapter 10

Overview and Conclusions

10.1. CONDITIONALLY STABLE METHODS

This thesis deals with the development of numerical methods for the three-dimensional shallow water equations on vector and parallel computers. At first, we investigated the numerical discretization of the vertical diffusion term. If an explicit method is used for a three-dimensional model, then besides the CFL stability condition there is also a condition imposed by the vertical diffusion term [1]. In many problems the latter condition is more restrictive. To investigate the influence of this stability condition, we examined in Chapter 3 time integrators that were explicit, semi-implicit or implicit in the vertical. It appears to be necessary to treat the vertical diffusion term in an implicit way. The vertically implicit (VIM) method (i.e., method (4.6) on page 16)

$$\begin{pmatrix} I-\tau\Lambda_{\sigma\sigma} & 0 & 0 \\ \tau F & I-\tau\Lambda_{\sigma\sigma} & 0 \\ \tau\Theta_1 hE_x & \tau\Theta_1 hD_y & I \end{pmatrix}\begin{pmatrix} U^{n+1} \\ V^{n+1} \\ Z^{n+1} \end{pmatrix} = \begin{pmatrix} I & \tau F & -\tau\Theta_2 gD_x \\ 0 & I & -\tau\Theta_2 gE_y \\ 0 & 0 & I \end{pmatrix}\begin{pmatrix} U^{n} \\ V^{n} \\ Z^{n} \end{pmatrix}. \qquad (10.1)$$

satisfies this condition. The U-, V- and Z-components are computed sequentially in (10.1). This is advantageous for both the stability and the storage requirements. The stability condition for the VIM method reads

$$\tau < \frac{1}{\sqrt{gh}} \frac{1}{\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}}, \qquad (10.2)$$

which shows that the maximally stable time step is independent of the vertical mesh size $\Delta\sigma$.

Method (10.1) may be written in the form

$$\begin{pmatrix} I-\alpha\tau\Lambda_{\sigma\sigma} & 0 & 0 \\ \tau F & I-\alpha\tau\Lambda_{\sigma\sigma} & 0 \\ \beta\tau\Theta_1 hE_x & \beta\tau\Theta_1 hD_y & I \end{pmatrix}\begin{pmatrix} U^{n+1} \\ V^{n+1} \\ Z^{n+1} \end{pmatrix} =$$

$$\begin{pmatrix} I+(1-\alpha)\tau\Lambda_{\sigma\sigma} & \tau F & -\tau\Theta_2 gD_x \\ 0 & I+(1-\alpha)\tau\Lambda_{\sigma\sigma} & -\tau\Theta_2 gE_y \\ (\beta-1)\tau\Theta_1 hD_y & (\beta-1)\tau\Theta_1 hD_y & I \end{pmatrix}\begin{pmatrix} U^{n} \\ V^{n} \\ Z^{n} \end{pmatrix}, \qquad (10.1')$$

where $\alpha=1$ and $\beta=1$. In order to obtain an as large as possible stability region, we varied the values of α and β. A similar stability analysis as in Chapter 3 (cf. pages 22-25) yielded the following conditions:

$$\begin{aligned} &\alpha \geq \frac{1}{2} \\ &\beta \geq 1 \\ &\tau < \frac{1}{\sqrt{2\beta-1}} \frac{1}{\sqrt{gh}} \frac{1}{\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}}. \end{aligned} \tag{10.2'}$$

Thus, $\beta=1$ is the optimum value. For $\beta=1$, the choices $\alpha=1$ and $\alpha=1/2$ lead to the vertically implicit methods (4.6) and (4.7), respectively, which are described on page 16. Both methods may be considered as a combination of the trapezoidal rule and the approach of Fischer [2] and Sielecki [8]. This has been explained in more detail in Chapter 4.

The vertically implicit methods require the solution of tridiagonal systems. The tridiagonal system are solved by the well-known Gaussian Elimination method. We make use of the fact that a large number of independent tridiagonal systems of the same dimension have to be solved. Therefore, these systems can be solved in a vector-parallel mode. Moreover, this method requires a minimal number of operations. Its efficiency has been shown on a CDC CYBER 205 and on an Alliant FX/4.

For large values of h (i.e., deep water) or for small values of the horizontal mesh sizes Δx and Δy, the time step restriction (10.2) or (10.2') may be more severe than necessary for accuracy considerations. To increase the stability we applied in Chapter 4 right-hand side smoothing. This technique was developed by Wubs [13]. The smoothing operators were constructed in such a way that they could be implemented efficiently on vector–parallel computers.

Right-hand side smoothing is particularly attractive in problems where the time derivative of the exact solution is a smooth function of the space variable. This is the case for the shallow water equations. In our experiments the application of right-hand side smoothing results in a reduction of the computation time of about a factor of five, while the accuracy remains acceptable.

The resulting stabilized vertically implicit (SVIM) method (i.e., method (4.3) on page 38) was made more accurate by applying the technique in which the water elevation and the velocity components are computed at different time levels. This technique, which was developed by Hansen [4], leads to method (4.7) on page 70.

10.2. UNCONDITIONALLY STABLE METHODS

We not only considered conditionally stable methods, but also numerical methods from the other end of the spectrum, viz., unconditionally stable methods. In Chapter 5 we constructed a two-stage time splitting (TSM) method. It was proved that this method is unconditionally stable. We remark that the first stage of this method is very similar to the vertically implicit method (10.1). It also requires the solution of a large number of tridiagonal systems. At the second stage the equations for the U- and V-component can easily be eliminated and yield a relatively small

system in which the water elevation Z is the only unknown. A conjugate gradient method was constructed in which a preconditioner based on the smoothing operator of Chapter 4 was used. It appears that the preconditioned CG method is highly suitable for vector and parallel computers.

The TSM method is first-order accurate in time. It is possible to obtain second-order accuracy by adjusting the discretization of the Coriolis term, the mixed advective terms and the bottom friction term. However, such a second-order treatment would decrease the computational efficiency dramatically. For the sake of efficiency we therefore decided to only construct a first-order accurate method. It should be noted that the propagation of the surface waves and the vertical diffusion are second-order accurate in time.

In Chapter 6 we compared the conditionally stable SVIM method with the unconditionally stable TSM method. Both methods can be implemented efficiently on vector-parallel computers. When compared with various methods from the literature, the SVIM method is an efficient method. However, the TSM method requires even less computation time than the SVIM method. Concerning accuracy, we encountered in some cases large errors for the velocity components when the SVIM method was used. The TSM method yields accurate results in all experiments. Since three-dimensional models are applied to test problems where the vertical structure of the velocity is needed, the accuracy for the velocity is very important. Considering the accuracy, stability and computational efficiency, we concluded that our unconditionally stable method is a very suitable method for three-dimensional shallow water models.

So far, the advective terms were omitted. In Chapter 7 we incorporated the advective terms into the TSM method. The discretization of the advective terms appears to be crucial and therefore we have followed the approach of Stelling [9]. Both the special discretization near the boundaries and the upwind discretization of the mixed advective terms as developed in [9] are necessary for stability. For example, central discretizations yielded instabilities in realistic test problems.

The introduction of the advective terms results in a hardly more complicated system. At the first stage two non-symmetric linear systems (for the u- and v-component, respectively) have to be solved. For its solution we developed a Jacobi-type iteration method. This method offers the facility of both an explicit and an implicit treatment of the advective terms. The choice between an explicit or implicit treatment depends on the test problem. At the second stage the system to be solved is very similar to the system without the advective terms (see Chapter 5). Therefore, the same preconditioned CG method is used.

In Chapter 8 the computational efficiency of the TSM method was demonstrated on a CRAY Y-MP4/464 for a rectangular domain. To exploit parallelism on this four-processor supercomputer, we had to apply a domain decomposition approach in the horizontal direction. For the implementation on an irregular domain, we used the geometry of the northwest European Continental Shelf. This model covers the same computational grid as the CSM16 model of Rijkswaterstaat. In Chapter 9 its efficiency was shown on the CRAY Y-MP2E of ICIM (Informatics Centre for Civil Engineering and Environment). This supercomputer was recently installed in the Netherlands for the simulation of large scale water models of rivers and seas.

So far, we only examined the stability and accuracy behaviour of various methods. However, other properties such as the dissipation and dispersion of numerical

methods are also important. To investigate these properties we carried out a similar analysis as described by Leendertse [6]. For the linear (one-dimensional) system

$$\begin{aligned} \frac{\partial u}{\partial t} &= -g\frac{\partial \zeta}{\partial x} - \gamma u \\ \frac{\partial \zeta}{\partial t} &= -d\frac{\partial u}{\partial x} \end{aligned} \tag{10.3}$$

where γ denotes the bottom friction, we obtained for our TSM method that the modulus of the wave propagation reads

$$\frac{\left\{\left(1+\frac{\tau\gamma}{2}\right)^2 + A\left(1+\frac{\tau\gamma}{2}\right) - \frac{\tau^2\gamma^2}{16}\right\}^{\frac{B}{2}} \left\{\left(1-\frac{\tau\gamma}{2}\right)^2 + A\left(1-\frac{\tau\gamma}{2}\right) - \frac{\tau^2\gamma^2}{16}\right\}^{\frac{B}{2}}}{\left\{\left(1+\frac{\tau\gamma}{2}\right)(1+A)\exp\left(-\frac{1}{2}\tau\gamma\right)\right\}^{B}}$$

and that the phase angle of the wave propagation is given by

$$\frac{\tan^{-1}\left(\frac{\sqrt{A\left(1+\frac{\tau\gamma}{2}\right) - \frac{\tau^2\gamma^2}{16}}}{1+0.5\tau\gamma}\right) + \tan^{-1}\left(\frac{\sqrt{A\left(1-\frac{\tau\gamma}{2}\right) - \frac{\tau^2\gamma^2}{16}}}{1+A}\right)}{C}$$

where

$$A = \frac{1}{4}\frac{\tau^2}{(\Delta x)^2} gd \sin^2\left(\frac{2\pi\Delta x}{L}\right), \quad B = \frac{L}{\tau\sqrt{gd - \left(\frac{\gamma\Delta x}{2\pi}\right)^2}} \quad \text{and} \quad C = \frac{2\pi}{B},$$

with L denoting the wave length. For the time splitting methods examined in [6], the moduli of the amplitudes are larger than one, which is due to the presence of the bottom friction term. For unconditional stability the eigenvalues should be less than one. It was concluded that the approximation of the bottom friction term should be made with care to avoid instabilities. For our TSM method the modulus of the amplitudes is always smaller than unity. Moreover, the amplitude errors are about 7.5 times smaller than for the methods in [6]. However, the phase errors for the TSM method are slightly larger.

10.3. Overview of time splitting methods

In the literature various time splitting methods have been developed for the shallow water equations. We now give an overview for some of these methods. For the two-dimensional shallow water equations, well-known methods are the ADI methods of Leendertse [6] and Stelling [9]. The latter one is an improved version of Leendertse's

method. The methods in [7] and [11] may be considered as three-dimensional extensions of the methods in [6] and [9], respectively. The vertical diffusion is treated in an implicit way.

For large time steps all these methods suffer from inaccuracies when dealing with complex geometries. This is the so-called ADI-effect [10]. In [12] a two-stage time splitting method has been developed in which these inaccuracies are absent, even for large time steps. Our TSM method, which has been developed for three-dimensional models, does also not suffer from the ADI-effect. When applied to two-dimensional problems, this method is very similar to the method in [12]. In Table 1 an overview of the various methods is given.

	ADI-effect	no ADI-effect
2D models	ADI-Stelling [9] ADI-Leendertse [6]	Fully implicit splitting method [12]
3D models	ADI-TRISULA [11] ADI-Leendertse [7]	TSM method [3]

Table 1. Time splitting methods for the shallow water equations.

It was reported that the method in [12] is more than acceptable for practical computations. Owing to the special treatment of the terms concerning the propagation of the surface waves, the method in [12] is about three to five times more expensive than the method in [9]. This ratio depends on the time step. For the methods in [3] and [12] a pentadiagonal system, which has a two-dimensional structure, has to be solved at the second stage. In [9] much smaller (tridiagonal) systems have to be solved.

For two-dimensional models the solution of the pentadiagonal system involves a major part of the computation time. This system is of the same (two-dimensional) structure and thus of the same computational complexity for both two-dimensional and three-dimensional models. The computation time required by the other parts of our TSM method, i.e., the computation of the three-dimensional velocity components, is proportional to the number of grid layers in the vertical direction. The efficiency of our time splitting method is therefore higher for three-dimensional models than for two-dimensional ones.

In conclusion, the unconditionally stable time splitting (TSM) method is an accurate and efficient method for three-dimensional shallow water models. This was illustrated both theoretically and experimentally. By comparing the numerical results with accurate reference solutions, it was possible to obtain detailed information about the accuracy of the TSM method. For realistic test problems such as for the IJsselmeer and the Continental Shelf, its computational efficiency was demonstrated on an Alliant FX/4 and on CRAY supercomputers.

REFERENCES

1. A.M. DAVIES, Application of the DuFort-Frankel and Saul'ev methods with time splitting to the formulation of a three dimensional hydrodynamic sea model, *Int. J. Numer. Meth. in Fluids*, 5, 405-425 (1985).
2. G. FISCHER, Ein numerisch verfahren zur errechnung von windstau und gezeiten in randmeeren, *Tellus*, 11, 60-76 (1959).
3. E.D. DE GOEDE, On the numerical treatment of the advective terms in 3D shallow water models, *Proceedings of the 2nd symposium on High Performance Computing*, Montpellier, 491-502 (1991).
4. W. HANSEN, Hydrodynamical methods applied to oceanographic problems, *Proceedings of the symposium on mathematical hydrodynamical methods of physical oceanography*, Institut für Meereskunde der Universität Hamburg, 25-34 (1961).
5. P.J. VAN DER HOUWEN, *Construction of integration formulas for initial-value problems*, North-Holland, Amsterdam, 1977.
6 J.J. LEENDERTSE, *Aspects of a computational model for long period water wave propagation*, Memorandum RM-5294-PR, Rand Corporation, Santa Monica, California, 1967.
7. J.J. LEENDERTSE, *A new approach to three-dimensional free-surface flow modelling*, Memorandum RM-5294-PR, Rand Corporation, Santa Monica, California, 1989.
8. A. SIELECKI, An energy conserving difference scheme for storm surge equations, *Monthly Weather Review*, 96, 150-156 (1968).
9. G.S. STELLING, *On the construction of computational methods for shallow water flow problems*, Ph.D. Thesis, TU Delft, 1983.
10. G.S. STELLING, A.K. WIERSMA and J.B.T.M. WILLEMSE, Practical aspects of accurate tidal computations, *J. Hydr. Eng., ASCE*, 112, 802-817 (1986).
11. TRISULA, A multi-dimensional flow and water-quality simulation system, Delft Hydraulics, The Netherlands, 1989.
12. P. WILDERS, Th.L.VAN STIJN, G.S. STELLING and G.A. FOKKEMA, A fully implicit splitting method for accurate tidal computations, *Int. J. Numer. Meth. in Eng.*, 26, 2707-2721 (1988).
13. F.W. WUBS, *Numerical solution of the shallow-water equations*, Ph.D. Thesis, University of Amsterdam, Amsterdam, 1987.

Index

CWI TRACTS

1 D.H.J. Epema. *Surfaces with canonical hyperplane sections.* 1984.

2 J.J. Dijkstra. *Fake topological Hilbert spaces and characterizations of dimension in terms of negligibility.* 1984.

3 A.J. van der Schaft. *System theoretic descriptions of physical systems.* 1984.

4 J. Koene. *Minimal cost flow in processing networks, a primal approach.* 1984.

5 B. Hoogenboom. *Intertwining functions on compact Lie groups.* 1984.

6 A.P.W. Böhm. *Dataflow computation.* 1984.

7 A. Blokhuis. *Few-distance sets.* 1984.

8 M.H. van Hoorn. *Algorithms and approximations for queueing systems.* 1984.

9 C.P.J. Koymans. *Models of the lambda calculus.* 1984.

10 C.G. van der Laan, N.M. Temme. *Calculation of special functions: the gamma function, the exponential integrals and error-like functions.* 1984.

11 N.M. van Dijk. *Controlled Markov processes; time-discretization.* 1984.

12 W.H. Hundsdorfer. *The numerical solution of nonlinear stiff initial value problems: an analysis of one step methods.* 1985.

13 D. Grune. *On the design of ALEPH.* 1985.

14 J.G.F. Thiemann. *Analytic spaces and dynamic programming: a measure theoretic approach.* 1985.

15 F.J. van der Linden. *Euclidean rings with two infinite primes.* 1985.

16 R.J.P. Groothuizen. *Mixed elliptic-hyperbolic partial differential operators: a case-study in Fourier integral operators.* 1985.

17 H.M.M. ten Eikelder. *Symmetries for dynamical and Hamiltonian systems.* 1985.

18 A.D.M. Kester. *Some large deviation results in statistics.* 1985.

19 T.M.V. Janssen. *Foundations and applications of Montague grammar, part 1: Philosophy, framework, computer science.* 1986.

20 B.F. Schriever. *Order dependence.* 1986.

21 D.P. van der Vecht. *Inequalities for stopped Brownian motion.* 1986.

22 J.C.S.P. van der Woude. *Topological dynamix.* 1986.

23 A.F. Monna. *Methods, concepts and ideas in mathematics: aspects of an evolution.* 1986.

24 J.C.M. Baeten. *Filters and ultrafilters over definable subsets of admissible ordinals.* 1986.

25 A.W.J. Kolen. *Tree network and planar rectilinear location theory.* 1986.

26 A.H. Veen. *The misconstrued semicolon: Reconciling imperative languages and dataflow machines.* 1986.

27 A.J.M. van Engelen. *Homogeneous zero-dimensional absolute Borel sets.* 1986.

28 T.M.V. Janssen. *Foundations and applications of Montague grammar, part 2: Applications to natural language.* 1986.

29 H.L. Trentelman. *Almost invariant subspaces and high gain feedback.* 1986.

30 A.G. de Kok. *Production-inventory control models: approximations and algorithms.* 1987.

31 E.E.M. van Berkum. *Optimal paired comparison designs for factorial experiments.* 1987.

32 J.H.J. Einmahl. *Multivariate empirical processes.* 1987.

33 O.J. Vrieze. *Stochastic games with finite state and action spaces.* 1987.

34 P.H.M. Kersten. *Infinitesimal symmetries: a computational approach.* 1987.

35 M.L. Eaton. *Lectures on topics in probability inequalities.* 1987.

36 A.H.P. van der Burgh, R.M.M. Mattheij (eds.). *Proceedings of the first international conference on industrial and applied mathematics (ICIAM 87).* 1987.

37 L. Stougie. *Design and analysis of algorithms for stochastic integer programming.* 1987.

38 J.B.G. Frenk. *On Banach algebras, renewal measures and regenerative processes.* 1987.

39 H.J.M. Peters, O.J. Vrieze (eds.). *Surveys in game theory and related topics.* 1987.

40 J.L. Geluk, L. de Haan. *Regular variation, extensions and Tauberian theorems.* 1987.

41 Sape J. Mullender (ed.). *The Amoeba distributed operating system: Selected papers 1984-1987.* 1987.

42 P.R.J. Asveld, A. Nijholt (eds.). *Essays on concepts, formalisms, and tools.* 1987.

43 H.L. Bodlaender. *Distributed computing: structure and complexity.* 1987.

44 A.W. van der Vaart. *Statistical estimation in large parameter spaces.* 1988.

45 S.A. van de Geer. *Regression analysis and empirical processes.* 1988.

46 S.P. Spekreijse. *Multigrid solution of the steady Euler equations.* 1988.

47 J.B. Dijkstra. *Analysis of means in some non-standard situations.* 1988.

48 F.C. Drost. *Asymptotics for generalized chi-square goodness-of-fit tests.* 1988.

49 F.W. Wubs. *Numerical solution of the shallow-water equations.* 1988.

50 F. de Kerf. *Asymptotic analysis of a class of perturbed Korteweg-de Vries initial value problems.* 1988.

51 P.J.M. van Laarhoven. *Theoretical and computational aspects of simulated annealing.* 1988.

52 P.M. van Loon. *Continuous decoupling transformations for linear boundary value problems.* 1988.

53 K.C.P. Machielsen. *Numerical solution of optimal control problems with state constraints by sequential quadratic programming in function space.* 1988.

54 L.C.R.J. Willenborg. *Computational aspects of survey data p ocessing.* 1988.

55 G.J. van der Steen. *A program generator for recognition, parsing and transduction with syntactic patterns.* 1988.

56 J.C. Ebergen. *Translating programs into delay-insensitive circuits.* 1989.

57 S.M. Verduyn Lunel. *Exponential type calculus for linear delay equations.* 1989.

58 M.C.M. de Gunst. *A random model for plant cell population growth.* 1989.

59 D. van Dulst. *Characterizations of Banach spaces not containing l^1.* 1989.

60 H.E. de Swart. *Vacillation and predictability properties of low-order atmospheric spectral models.* 1989.

61 P. de Jong. *Central limit theorems for generalized multilinear forms.* 1989.

62 V.J. de Jong. *A specification system for statistical software.* 1989.

63 B. Hanzon. *Identifiability, recursive identification and spaces of linear dynamical systems, part I.* 1989.

64 B. Hanzon. *Identifiability, recursive identification and spaces of linear dynamical systems, part II.* 1989.

65 B.M.M. de Weger. *Algorithms for diophantine equations.* 1989.

66 A. Jung. *Cartesian closed categories of domains.* 1989.

67 J.W. Polderman. *Adaptive control & identification: Conflict or conflux?.* 1989.

68 H.J. Woerdeman. *Matrix and operator extensions.* 1989.

69 B.G. Hansen. *Monotonicity properties of infinitely divisible distributions.* 1989.

70 J.K. Lenstra, H.C. Tijms, A. Volgenant (eds.). *Twenty-five years of operations research in the Netherlands: Papers dedicated to Gijs de Leve.* 1990.

71 P.J.C. Spreij. *Counting process systems. Identification and stochastic realization.* 1990.

72 J.F. Kaashoek. *Modeling one dimensional pattern formation by anti-diffusion.* 1990.

73 A.M.H. Gerards. *Graphs and polyhedra. Binary spaces and cutting planes.* 1990.

74 B. Koren. *Multigrid and defect correction for the steady Navier-Stokes equations. Application to aerodynamics.* 1991.

75 M.W.P. Savelsbergh. *Computer aided routing.* 1992.

76 O.E. Flippo. *Stability, duality and decomposition in general mathematical programming.* 1991.

77 A.J. van Es. *Aspects of nonparametric density estimation.* 1991.

78 G.A.P. Kindervater. *Exercises in parallel combinatorial computing.* 1992.

79 J.J. Lodder. *Towards a symmetrical theory of generalized functions.* 1991.

80 S.A. Smulders. *Control of freeway traffic flow.* 1993.

81 P.H.M. America, J.J.M.M. Rutten. *A parallel object-oriented language: design and semantic foundations.* 1992.

82 F. Thuijsman. *Optimality and equilibria in stochastic games.* 1992.

83 R.J. Kooman. *Convergence properties of recurrence sequences.* 1992.

84 A.M. Cohen (ed.). *Computational aspects of Lie group representations and related topics. Proceedings of the 1990 Computational Algebra Seminar at CWI, Amsterdam.* 1991.

85 V. de Valk. *One-dependent processes.* 1993.

86 J.A. Baars, J.A.M. de Groot. *On topological and linear equivalence of certain function spaces.* 1992.

87 A.F. Monna. *The way of mathematics and mathematicians.* 1992.

88 E.D. de Goede. *Numerical methods for the three-dimensional shallow water equations.* 1993.

89 M. Zwaan. *Moment problems in Hilbert space with applications to magnetic resonance imaging.* 1993.

90 C. Vuik. *The solution of a one-dimensional Stefan problem.* 1993.

91. E.R. Verheul. *Multimedians in metric and normed spaces.* 1993.

MATHEMATICAL CENTRE TRACTS

1 T. van der Walt. *Fixed and almost fixed points.* 1963.

2 A.R. Bloemena. *Sampling from a graph.* 1964.

3 G. de Leve. *Generalized Markovian decision processes, part I: model and method.* 1964.

4 G. de Leve. *Generalized Markovian decision processes, part II: probabilistic background.* 1964.

5 G. de Leve, H.C. Tijms, P.J. Weeda. *Generalized Markovian decision processes, applications.* 1970.

6 M.A. Maurice. *Compact ordered spaces.* 1964.

7 W.R. van Zwet. *Convex transformations of random variables.* 1964.

8 J.A. Zonneveld. *Automatic numerical integration.* 1964.

9 P.C. Baayen. *Universal morphisms.* 1964.

10 E.M. de Jager. *Applications of distributions in mathematical physics.* 1964.

11 A.B. Paalman-de Miranda. *Topological semigroups.* 1964.

12 J.A.Th.M. van Berckel, H. Brandt Corstius, R.J. Mokken, A. van Wijngaarden. *Formal properties of newspaper Dutch.* 1965.

13 H.A. Lauwerier. *Asymptotic expansions.* 1966, out of print; replaced by MCT 54.

14 H.A. Lauwerier. *Calculus of variations in mathematical physics.* 1966.

15 R. Doornbos. *Slippage tests.* 1966.

16 J.W. de Bakker. *Formal definition of programming languages with an application to the definition of ALGOL 60.* 1967.

17 R.P. van de Riet. *Formula manipulation in ALGOL 60, part 1.* 1968.

18 R.P. van de Riet. *Formula manipulation in ALGOL 60, part 2.* 1968.

19 J. van der Slot. *Some properties related to compactness.* 1968.

20 P.J. van der Houwen. *Finite difference methods for solving partial differential equations.* 1968.

21 E. Wattel. *The compactness operator in set theory and topology.* 1968.

22 T.J. Dekker. *ALGOL 60 procedures in numerical algebra, part 1.* 1968.

23 T.J. Dekker, W. Hoffmann. *ALGOL 60 procedures in numerical algebra, part 2.* 1968.

24 J.W. de Bakker. *Recursive procedures.* 1971.

25 E.R. Paërl. *Representations of the Lorentz group and projective geometry.* 1969.

26 European Meeting 1968. *Selected statistical papers, part I.* 1968.

27 European Meeting 1968. *Selected statistical papers, part II.* 1968.

28 J. Oosterhoff. *Combination of one-sided statistical tests.* 1969.

29 J. Verhoeff. *Error detecting decimal codes.* 1969.

30 H. Brandt Corstius. *Exercises in computational linguistics.* 1970.

31 W. Molenaar. *Approximations to the Poisson, binomial and hypergeometric distribution functions.* 1970.

32 L. de Haan. *On regular variation and its application to the weak convergence of sample extremes.* 1970.

33 F.W. Steutel. *Preservation of infinite divisibility under mixing and related topics.* 1970.

34 I. Juhász, A. Verbeek, N.S. Kroonenberg. *Cardinal functions in topology.* 1971.

35 M.H. van Emden. *An analysis of complexity.* 1971.

36 J. Grasman. *On the birth of boundary layers.* 1971.

37 J.W. de Bakker, G.A. Blaauw, A.J.W. Duijvestijn, E.W. Dijkstra, P.J. van der Houwen, G.A.M. Kamsteeg-Kemper, F.E.J. Kruseman Aretz, W.L. van der Poel, J.P. Schaap-Kruseman, M.V. Wilkes, G. Zoutendijk. *MC-25 Informatica Symposium.* 1971.

38 W.A. Verloren van Themaat. *Automatic analysis of Dutch compound words.* 1972.

39 H. Bavinck. *Jacobi series and approximation.* 1972.

40 H.C. Tijms. *Analysis of (s,S) inventory models.* 1972.

41 A. Verbeek. *Superextensions of topological spaces.* 1972.

42 W. Vervaat. *Success epochs in Bernoulli trials (with applications in number theory).* 1972.

43 F.H. Ruymgaart. *Asymptotic theory of rank tests for independence.* 1973.

44 H. Bart. *Meromorphic operator valued functions.* 1973.

45 A.A. Balkema. *Monotone transformations and limit laws.* 1973.

46 R.P. van de Riet. *ABC ALGOL, a portable language for formula manipulation systems, part 1: the language.* 1973.

47 R.P. van de Riet. *ABC ALGOL, a portable language for formula manipulation systems, part 2: the compiler.* 1973.

48 F.E.J. Kruseman Aretz, P.J.W. ten Hagen, H.L. Oudshoorn. *An ALGOL 60 compiler in ALGOL 60, text of the MC-compiler for the EL-X8.* 1973.

49 H. Kok. *Connected orderable spaces.* 1974.

50 A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck, C.H.A. Koster, M. Sintzoff, C.H. Lindsey, L.G.L.T. Meertens, R.G. Fisker (eds.). *Revised report on the algorithmic language ALGOL 68.* 1976.

51 A. Hordijk. *Dynamic programming and Markov potential theory.* 1974.

52 P.C. Baayen (ed.). *Topological structures.* 1974.

53 M.J. Faber. *Metrizability in generalized ordered spaces.* 1974.

54 H.A. Lauwerier. *Asymptotic analysis, part 1.* 1974.

55 M. Hall, Jr., J.H. van Lint (eds.). *Combinatorics, part 1: theory of designs, finite geometry and coding theory.* 1974.

56 M. Hall, Jr., J.H. van Lint (eds.). *Combinatorics, part 2: graph theory, foundations, partitions and combinatorial geometry.* 1974.

57 M. Hall, Jr., J.H. van Lint (eds.). *Combinatorics, part 3: combinatorial group theory.* 1974.

58 W. Albers. *Asymptotic expansions and the deficiency concept in statistics.* 1975.

59 J.L. Mijnheer. *Sample path properties of stable processes.* 1975.

60 F. Göbel. *Queueing models involving buffers.* 1975.

63 J.W. de Bakker (ed.). *Foundations of computer science.* 1975.

64 W.J. de Schipper. *Symmetric closed categories.* 1975.

65 J. de Vries. *Topological transformation groups, 1: a categorical approach.* 1975.

66 H.G.J. Pijls. *Logically convex algebras in spectral theory and eigenfunction expansions.* 1976.

68 P.P.N. de Groen. *Singularly perturbed differential operators of second order.* 1976.

69 J.K. Lenstra. *Sequencing by enumerative methods.* 1977.

70 W.P. de Roever, Jr. *Recursive program schemes: semantics and proof theory.* 1976.

71 J.A.E.E. van Nunen. *Contracting Markov decision processes.* 1976.

72 J.K.M. Jansen. *Simple periodic and non-periodic Lamé functions and their applications in the theory of conical waveguides.* 1977.

73 D.M.R. Leivant. *Absoluteness of intuitionistic logic.* 1979.

74 H.J.J. te Riele. *A theoretical and computational study of generalized aliquot sequences.* 1976.

75 A.E. Brouwer. *Treelike spaces and related connected topological spaces.* 1977.

76 M. Rem. *Associons and the closure statement.* 1976.

77 W.C.M. Kallenberg. *Asymptotic optimality of likelihood ratio tests in exponential families.* 1978.

78 E. de Jonge, A.C.M. van Rooij. *Introduction to Riesz spaces.* 1977.

79 M.C.A. van Zuijlen. *Emperical distributions and rank statistics.* 1977.

80 P.W. Hemker. *A numerical study of stiff two-point boundary problems.* 1977.

81 K.R. Apt, J.W. de Bakker (eds.). *Foundations of computer science II, part 1.* 1976.

82 K.R. Apt, J.W. de Bakker (eds.). *Foundations of computer science II, part 2.* 1976.

83 L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the AUTOMATH system.* 1979.

84 H.L.L. Busard. *The translation of the elements of Euclid from the Arabic into Latin by Hermann of Carinthia (?), books vii-xii.* 1977.

85 J. van Mill. *Supercompactness and Wallman spaces.* 1977.

86 S.G. van der Meulen, M. Veldhorst. *Torrix I, a programming system for operations on vectors and matrices over arbitrary fields and of variable size.* 1978.

88 A. Schrijver. *Matroids and linking systems.* 1977.

89 J.W. de Roever. *Complex Fourier transformation and analytic functionals with unbounded carriers.* 1978.

90 L.P.J. Groenewegen. *Characterization of optimal strategies in dynamic games.* 1981.

91 J.M. Geysel. *Transcendence in fields of positive characteristic.* 1979.

92 P.J. Weeda. *Finite generalized Markov programming.* 1979.

93 H.C. Tijms, J. Wessels (eds.). *Markov decision theory.* 1977.

94 A. Bijlsma. *Simultaneous approximations in transcendental number theory.* 1978.

95 K.M. van Hee. *Bayesian control of Markov chains.* 1978.

96 P.M.B. Vitányi. *Lindenmayer systems: structure, languages, and growth functions.* 1980.

97 A. Federgruen. *Markovian control problems; functional equations and algorithms.* 1984.

98 R. Geel. *Singular perturbations of hyperbolic type.* 1978.

99 J.K. Lenstra, A.H.G. Rinnooy Kan, P. van Emde Boas (eds.). *Interfaces between computer science and operations research.* 1978.

100 P.C. Baayen, D. van Dulst, J. Oosterhoff (eds.). *Proceedings bicentennial congress of the Wiskundig Genootschap, part 1.* 1979.

101 P.C. Baayen, D. van Dulst, J. Oosterhoff (eds.). *Proceedings bicentennial congress of the Wiskundig Genootschap, part 2.* 1979.

102 D. van Dulst. *Reflexive and superreflexive Banach spaces.* 1978.

103 K. van Harn. *Classifying infinitely divisible distributions by functional equations.* 1978.

104 J.M. van Wouwe. *Go-spaces and generalizations of metrizability.* 1979.

105 R. Helmers. *Edgeworth expansions for linear combinations of order statistics.* 1982.

106 A. Schrijver (ed.). *Packing and covering in combinatorics.* 1979.

107 C. den Heijer. *The numerical solution of nonlinear operator equations by imbedding methods.* 1979.

108 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science III, part 1.* 1979.

109 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science III, part 2.* 1979.

110 J.C. van Vliet. *ALGOL 68 transput, part I: historical review and discussion of the implementation model.* 1979.

111 J.C. van Vliet. *ALGOL 68 transput, part II: an implementation model.* 1979.

112 H.C.P. Berbee. *Random walks with stationary increments and renewal theory.* 1979.

113 T.A.B. Snijders. *Asymptotic optimality theory for testing problems with restricted alternatives.* 1979.

114 A.J.E.M. Janssen. *Application of the Wigner distribution to harmonic analysis of generalized stochastic processes.* 1979.

115 P.C. Baayen, J. van Mill (eds.). *Topological structures II, part 1.* 1979.

116 P.C. Baayen, J. van Mill (eds.). *Topological structures II, part 2.* 1979.

117 P.J.M. Kallenberg. *Branching processes with continuous state space.* 1979.

118 P. Groeneboom. *Large deviations and asymptotic efficiencies.* 1980.

119 F.J. Peters. *Sparse matrices and substructures, with a novel implementation of finite element algorithms.* 1980.

120 W.P.M. de Ruyter. *On the asymptotic analysis of large-scale ocean circulation.* 1980.

121 W.H. Haemers. *Eigenvalue techniques in design and graph theory.* 1980.

122 J.C.P. Bus. *Numerical solution of systems of nonlinear equations.* 1980.

123 I. Yuhász. *Cardinal functions in topology - ten years later.* 1980.

124 R.D. Gill. *Censoring and stochastic integrals.* 1980.

125 R. Eising. *2-D systems, an algebraic approach.* 1980.

126 G. van der Hoek. *Reduction methods in nonlinear programming.* 1980.

127 J.W. Klop. *Combinatory reduction systems.* 1980.

128 A.J.J. Talman. *Variable dimension fixed point algorithms and triangulations.* 1980.

129 G. van der Laan. *Simplicial fixed point algorithms.* 1980.

130 P.J.W. ten Hagen, T. Hagen, P. Klint, H. Noot, H.J. Sint, A.H. Veen. *ILP: intermediate language for pictures.* 1980.

131 R.J.R. Back. *Correctness preserving program refinements: proof theory and applications.* 1980.

132 H.M. Mulder. *The interval function of a graph.* 1980.

133 C.A.J. Klaassen. *Statistical performance of location estimators.* 1981.

134 J.C. van Vliet, H. Wupper (eds.). *Proceedings international conference on ALGOL 68.* 1981.

135 J.A.G. Groenendijk, T.M.V. Janssen, M.J.B. Stokhof (eds.). *Formal methods in the study of language, part I.* 1981.

136 J.A.G. Groenendijk, T.M.V. Janssen, M.J.B. Stokhof (eds.). *Formal methods in the study of language, part II.* 1981.

137 J. Telgen. *Redundancy and linear programs.* 1981.

138 H.A. Lauwerier. *Mathematical models of epidemics.* 1981.

139 J. van der Wal. *Stochastic dynamic programming, successive approximations and nearly optimal strategies for Markov decision processes and Markov games.* 1981.

140 J.H. van Geldrop. *A mathematical theory of pure exchange economies without the no-critical-point hypothesis.* 1981.

141 G.E. Welters. *Abel-Jacobi isogenies for certain types of Fano threefolds.* 1981.

142 H.R. Bennett, D.J. Lutzer (eds.). *Topology and order structures, part 1.* 1981.

143 J.M. Schumacher. *Dynamic feedback in finite- and infinite-dimensional linear systems.* 1981.

144 P. Eijgenraam. *The solution of initial value problems using interval arithmetic; formulation and analysis of an algorithm.* 1981.

145 A.J. Brentjes. *Multi-dimensional continued fraction algorithms.* 1981.

146 C.V.M. van der Mee. *Semigroup and factorization methods in transport theory.* 1981.

147 H.H. Tigelaar. *Identification and informative sample size.* 1982.

148 L.C.M. Kallenberg. *Linear programming and finite Markovian control problems.* 1983.

149 C.B. Huijsmans, M.A. Kaashoek, W.A.J. Luxemburg, W.K. Vietsch (eds.). *From A to Z, proceedings of a symposium in honour of A.C. Zaanen.* 1982.

150 M. Veldhorst. *An analysis of sparse matrix storage schemes.* 1982.

151 R.J.M.M. Does. *Higher order asymptotics for simple linear rank statistics.* 1982.

152 G.F. van der Hoeven. *Projections of lawless sequences.* 1982.

153 J.P.C. Blanc. *Application of the theory of boundary value problems in the analysis of a queueing model with paired services.* 1982.

154 H.W. Lenstra, Jr., R. Tijdeman (eds.). *Computational methods in number theory, part I.* 1982.

155 H.W. Lenstra, Jr., R. Tijdeman (eds.). *Computational methods in number theory, part II.* 1982.

156 P.M.G. Apers. *Query processing and data allocation in distributed database systems.* 1983.

157 H.A.W.M. Kneppers. *The covariant classification of two-dimensional smooth commutative formal groups over an algebraically closed field of positive characteristic.* 1983.

158 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science IV, distributed systems, part 1.* 1983.

159 J.W. de Bakker, J. van Leeuwen (eds.). *Foundations of computer science IV, distributed systems, part 2.* 1983.

160 A. Rezus. *Abstract AUTOMATH.* 1983.

161 G.F. Helminck. *Eisenstein series on the metaplectic group, an algebraic approach.* 1983.

162 J.J. Dik. *Tests for preference.* 1983.

163 H. Schippers. *Multiple grid methods for equations of the second kind with applications in fluid mechanics.* 1983.

164 F.A. van der Duyn Schouten. *Markov decision processes with continuous time parameter.* 1983.

165 P.C.T. van der Hoeven. *On point processes.* 1983.

166 H.B.M. Jonkers. *Abstraction, specification and implementation techniques, with an application to garbage collection.* 1983.

167 W.H.M. Zijm. *Nonnegative matrices in dynamic programming.* 1983.

168 J.H. Evertse. *Upper bounds for the numbers of solutions of diophantine equations.* 1983.

169 H.R. Bennett, D.J. Lutzer (eds.). *Topology and order structures, part 2.* 1983.