Applied Numerical Mathematics 10 (1992) 3–18
North-Holland

APNUM 300

Numerical methods for the 3D shallow water equations on vector and parallel computers

Erik D. de Goede

Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, Netherlands

Abstract

De Goede, E.D., Numerical methods for the 3D shallow water equations on vector and parallel computers, Applied Numerical Mathematics 10 (1992) 3–18.

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.

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. Both 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.

2. Mathematical model

In this section we will describe a mathematical model for the three-dimensional shallow water equations. The following symbols are used:

u, v:	velocity components in x- and y-direction,
ζ:	water elevation above undisturbed depth,
x, y, σ:	a left-handed set of co-ordinates,
t:	time,
C:	Chezy coefficient,
L, B:	dimensions of the basin in x- and y-direction, respectively,
d:	undisturbed depth of water,
f:	Coriolis term,
g:	acceleration due to gravity,
h:	total depth $(= d + \zeta)$,
w_f:	wind stress,
μ:	vertical diffusion coefficient,
φ:	angle between wind direction and the positive x-axis,
ρ:	density.

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

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

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

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

Owing to the sigma transformation [13]

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

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

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

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

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

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.

3. Space discretization

For the space discretization of the equations (2.1)–(2.3), the computational domain is covered by an $nx \cdot ny \cdot ns$ rectangular staggered grid (see [3–5]). 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, $\boldsymbol{V}$, $\boldsymbol{Z}$, $\boldsymbol{D}$ and $\boldsymbol{H}$ are grid functions approximating v, ζ, d and h, respectively. Note that $\boldsymbol{D}$, $\boldsymbol{H}$ and $\boldsymbol{Z}$ are only computed at the upper layer. Furthermore, $A_{\sigma\sigma}$ is a tridiagonal matrix approximating the vertical diffusion term, including the discretization of the term $1/h^2$. We remark that $A_{\sigma\sigma}$ 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 $(nx \cdot ny \cdot ns) \cdot (nx \cdot ny)$ matrix (a row of ns diagonal matrices of order $nx \cdot ny$ with $\Delta\sigma_k$ on the diagonal of the kth submatrix). Θ_2 is an $(nx \cdot ny) \cdot (nx \cdot ny \cdot ns)$ matrix (a column of ns identity matrices of order $nx \cdot ny$. F is a four-diagonal matrix (due to the grid staggering) of order $nx \cdot ny \cdot ns$, approximating the Coriolis term. D_x and D_y are bidiagonal matrices (one diagonal and one lower diagonal) of order $nx \cdot ny$, 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 $E_x = -D_x^{\mathrm{T}}$ and $E_y = -D_y^{\mathrm{T}}$. 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 vertical directions. Now, the semi-discretized system can be written in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}W = F(W) = \begin{pmatrix} \Lambda_{\sigma\sigma} & F & -\Theta_2 gD_x \\ -F & \Lambda_{\sigma\sigma} & -\Theta_2 gE_y \\ -\Theta_1 HE_x & -\Theta_1 HD_y & 0 \end{pmatrix} W + \begin{pmatrix} F_u \\ F_v \\ \mathbf{0} \end{pmatrix}, \tag{3.1}$$

where $W = (U, V, Z)^{\mathrm{T}}$ and $(F_u, F_v, \mathbf{0})^{\mathrm{T}}$ 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.

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.

4.1. The conditionally stable method

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

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

where τ denotes the time step and $W^n = (U^n, V^n, Z^n)^{\mathrm{T}}$ 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

$$W^{n+1} = W^n + \tau(I - \tau\{A_1 + A_2\})^{-1}F(W^n), \quad \text{with}$$
$$A_1 = \begin{pmatrix} \Lambda_{\sigma\sigma} & 0 & 0 \\ 0 & \Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} 0 & 0 & 0 \\ -F & 0 & 0 \\ -\Theta_1 HE_x & -\Theta_1 HD_y & 0 \end{pmatrix}. \tag{4.1}$$

The stability condition for method (4.1) is given by

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

where Δx and Δy denote the horizontal mesh sizes. This stability condition is slightly more restrictive than the condition derived in [3].

Method (4.1) is first-order accurate in time. For the U- and V-components, the implicit treatment of the vertical diffusion term requires the solution of $nx \cdot ny$ 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. Therefore, we have increased the stability of method

(4.1) by right-hand side smoothing [4]. The application of right-hand side smoothing in more than one direction is complicated. Therefore, we have applied one-dimensional smoothing 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 $\boldsymbol{U}$- and $\boldsymbol{Z}$-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

$$D_u = \frac{1}{4}\begin{pmatrix} 0 & & & & & 0 \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 \\ 0 & & & & & 0 \end{pmatrix},$$

$$D_z = \frac{1}{4}\begin{pmatrix} -1 & 1 & & & & 0 \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 \\ 0 & & & & -1 & 1 \end{pmatrix}. \tag{4.4}$$

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)

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

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

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

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 $w = (u, v, \zeta)^{\mathrm{T}}$) 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

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

with $\tilde{W}^n = (U^{n-1/2}, V^{n-1/2}, Z^n)^{\mathrm{T}}$ 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 surface waves are now treated second-order accurate in time. The stability condition (4.6) is also valid for method (4.7).

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

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

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

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

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

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

ʟt the second stage the terms describing the propagation of the surface waves are treated licitly. This system reads

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

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

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

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

/e emphasize that the water elevation is the only unknown in system (4.12). Thus, this ɜm is of the same (two-dimensional) structure and thus of the same computational plexity for both two-dimensional and three-dimensional test problems. The computation : for the other parts of method (4.8)–(4.9) is proportional to the number of vertical grid rs. Therefore, this time splitting method is more efficient for three-dimensional than for dimensional problems. In [16] it was reported that a time splitting method of this form is ɜdy feasible for two-dimensional problems.

ɔlving the linear systems

ı this section, we will describe how the linear systems (4.10) and (4.12) are solved. For ɜm (4.10), which requires for both velocity components the solution of $nx \cdot ny$ tridiagonal ɜms of dimension ns, we apply the Gaussian Elimination (double sweep) method. Since this ɔod 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.

5.1. The smoothed Jacobi method

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

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

where Z_k denotes the kth 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 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

$$Z_{k+1} = Z_k + \omega\left(I - \alpha D_z^{(x)}\right)^{-1}\left(I - \alpha D_z^{(y)}\right)^{-1}\{\boldsymbol{B} - AZ_k\}, \quad k = 1, 2, 3, \ldots, \tag{5.4}$$

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

$$\omega = -\frac{2\alpha\beta + \alpha^2}{\beta^2},$$

with the constant

$$\beta = \frac{\tau^2}{\Delta^2} g H_{\max}, \qquad H_{\max} = \max_{\substack{1 \leqslant i \leqslant nx \\ 1 \leqslant j \leqslant ny}} \{H_{i,j}\} \quad \text{and} \quad \Delta = \Delta x = \Delta y,$$

then method (5.4) may be written in the form

$$Z_{k+1} = Z_k - (2\alpha_2 + 1)(\beta D_z^{(x)} - \alpha_2 I)^{-1} (\beta D_z^{(y)} - \alpha_2 I)^{-1} (B - AZ_k), \tag{5.4'}$$

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

$$\begin{aligned} (\beta D_z^{(x)} - \alpha_2 I)\tilde{Z} &= (\beta D_z^{(x)} - \alpha_2 I) Z_k + \{B - AZ_k\}, \\ (\beta D_z^{(y)} - \alpha_2 I) Z_{k+1} &= (\beta D_z^{(y)} - \alpha_2 I)\tilde{Z} + \{B - A\tilde{Z}\}. \end{aligned} \tag{5.5}$$

Method (5.5) may be considered as an ADI iteration method written in residual form. For such methods the derivation of parameter values is extensively described 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.

5.2. *The smoothed CG method*

The second iteration method that we applied 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 Z_0 be an initial guess for $Z^{(q+1)}$ and

$$R_0 = B - AZ_0, \qquad P_0 = SR_0.$$

For $k = 0, 1, 2, \ldots,$ until convergence

$$\begin{aligned} \alpha_k &= \frac{R_k^{\mathrm{T}}(SR_k)}{P_k^{\mathrm{T}}(AP_k)}, \\ Z_{k+1} &= Z_k + \alpha_k P_k, \\ R_{k+1} &= R_k - \alpha_k AP_k, \\ \beta_k &= \frac{R_{k+1}^{\mathrm{T}}(SR_{k+1})}{R_k^{\mathrm{T}}(SR_k)}, \\ P_{k+1} &= SR_{k+1} + \beta_k P_k, \end{aligned} \tag{5.6}$$

where R_k denotes the kth residual vector and P_k the kth 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 $\gamma = 1$, 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. 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 an Alliant FX/4, which is a mini-supercomputer having 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:

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

We have used a rectangular basin of 400 by 800 km with different bottom topographies. For the horizontal grid sizes we have chosen $\Delta x = 10$ km and $\Delta y = 10$ 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 the periodically varying northeastern wind stress of

$$1.5 + 0.75 * \sin \frac{2\pi t}{24 * 3600} \text{kgm/s}^2.$$

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 $\tau = 30$ 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-($\cdot$):	maximal global error of either u, v or ζ at the end point $T = 5$ days,
COMP:	computation time on the Alliant FX/4.

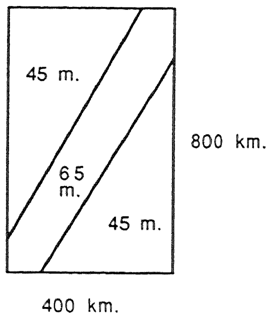

Fig. 1. Plane bottom with a channel.

For the TSM method we have required that the residue $\| B_{\zeta}^{n+1/2} - AZ_k \|_{\infty}$ 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 Fig. 1. For this test problem the results are listed in Table 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 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 $\tau = 1800$ 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

Table 1
Test problem with a channel in a diagonal direction

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

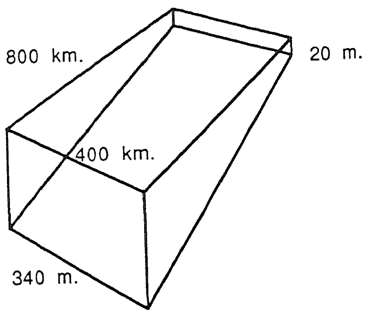

Fig. 2. Inclined bottom.

method roughly requires twice as much computation time as the SVIM methods. However, when we consider the accuracy, the TSM method is more accurate.

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 Fig. 2). The results are listed in Table 2. Here, the maximal values for the three components are about 0.7 m/s and 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 3 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

Table 2
Test problem with an inclined bottom

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 3
Test problem with an inclined bottom and $ns = 1$

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

about four times smaller (see Table 2). 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-dimensional experiments. This results in a smaller decrease of the accuracy when smoothing was 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.

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 than 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 4 we list the results for the smoothed Jacobi method. Here the bottom topography with the diagonal channel is used. In this experiment, only twenty-five time steps have been

Table 4
Smoothed Jacobi method for the problem with a diagonal channel

q	$\tau = 800$ s			$\tau = 3600$ s		
	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 ($\alpha = 0.8$)	12	75.6	51.8 ($\alpha = 20.6$)	45

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.

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 $\tau = 3600$ 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 cannot be implemented as efficiently as the explicit smoothing, the reduction in computation time is less for implicit smoothing.

In Table 5 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 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 5 we have listed these optimum values. For γ-values in the

Table 5
Results for the CG method

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

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, since the number of iterations is already very limited. For the 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.

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 (see also [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 very suitable method for the three-dimensional shallow water models.

For the unconditionally stable method a symmetric, five-diagonal 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.

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, *Internat. J. Numer. Methods Fluids* 5 (1985) 405–425.

[2] S.C. Eisenstat, Efficient implementation of a class of preconditioned conjugate gradient methods, *SIAM J. Sci. Comput.* 2 (1981) 1–4.

[3] E.D. de Goede, A computational model for the three-dimensional shallow water flows on the Alliant FX/4, *Supercomputer* 32 (1988) 43–49.

[4] E.D. de Goede, Stabilization of a time integrator of the 3D shallow water equations by smoothing techniques, *Internat. J. Numer. Methods Fluids* 12 (1991) 475–490.

[5] E.D. de Goede, A time splitting method for the three-dimensional shallow water equations, *Internat. J. Numer. Methods Fluids* 13 (1991) 519–534.
[6] W. Hansen, Hydrodynamical methods applied to oceanographic problems, in: *Proceedings of the Symposium on Mathematical Hydrodynamical Methods of Physical Oceanography,* Institut für Meereskunde der Universität Hamburg (1961).
[7] P.J. van der Houwen, Stabilization of explicit difference schemes by smoothing techniques, in: K. Strehmel, ed., *Numerical Treatment of Differential Equations, Proceedings Fourth Seminar Halle 1987: NUMDIFF-4*, Teubner-Texte zur Mathematik 104 (Teubner, Leipzig, 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 (1988) 3–10.
[9] J.J. Leendertse, Aspects of a computational model for long period water wave propagation. Memorandum RM-5294-PR, Rand Corp., Santa Monica, CA (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 (1977) 148–162.
[12] J. Ortega and R. Voigt, Solution of partial differential equations on vector and parallel computers, *SIAM Rev.* 27 (1985) 149–240.
[13] N.A. Phillips, A coordinate system having some special advantages for numerical forecasting, *J. Meteorol.* 14 (1957) 184–194.
[14] R.D. Richtmyer and K.W. Morton, *Difference Methods for Initial Value Problems* (Interscience/Wiley, New York, 1967).
[15] R. Shapiro, Smoothing, filtering and boundary effects, *Rev. Geophys. Space Phys.* 8 (1970) 359–387.
[16] P. Wilders, T.L. van Stijn, G.S. Stelling and G.A. Fokkema, A fully implicit splitting method for accurate tidal computations, *Internat. J. Numer. Methods Engrg.* 26 (1988) 2707–2721.
[17] D.M. Young, *Iterative Solution of Large Linear Systems Methods* (Academic Press, New York, 1971).