INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 13, 519–534 (1991)

A TIME-SPLITTING METHOD FOR THE THREE-DIMENSIONAL SHALLOW WATER EQUATIONS

E. D. DE GOEDE

Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

SUMMARY

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.

KEY WORDS Three-dimensional shallow water equations Method of lines Time integrators Stability Vector and parallel computers

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 multidimensional 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.[1]

For the two-dimensional shallow water equations several of the existing numerical methods have been based on the ADI method.[2,3] 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.[4] In Reference 5 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 Reference 5. 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.[5–7] In addition to that, in three-dimensional models, where a vertical diffusion term is involved, we have to treat the vertical

0271–2091/91/140519–16$08.00 *Received September 1990*

diffusion term implicitly to avoid the maximally stable time step becoming too small.[8] 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.[9]

At the second stage a linearization process is used to iteratively solve the non-linear 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 Reference 5 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 non-linear 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 relatively 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.

2. MATHEMATICAL MODEL

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

C	Chezy coefficient
d	undisturbed depth of water
f	Coriolis term
F_b	bottom stress in x-direction
F_s	surface stress in x-direction
g	acceleration due to gravity
G_b	bottom stress in y-direction
G_s	surface stress in y-direction
h	total depth $(=d+\zeta)$
N	vertical diffusion coefficient
t	time

u, v	velocity components in x- and y-direction
x, y, σ	a left-handed set of co-ordinates
W_f	wind stress
ρ	density
φ	angle between wind direction and positive x-axis
ζ	elevation above undisturbed depth.

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

$$\frac{\partial u}{\partial t}=fv-g\frac{\partial \zeta}{\partial x}+\frac{1}{\rho h^2}\frac{\partial(N\partial u/\partial\sigma)}{\partial\sigma}, \tag{1}$$

$$\frac{\partial v}{\partial t}=-fu-g\frac{\partial \zeta}{\partial y}+\frac{1}{\rho h^2}\frac{\partial(N\partial v/\partial\sigma)}{\partial\sigma}, \tag{2}$$

$$\frac{\partial \zeta}{\partial t}=-\frac{\partial}{\partial x}\left(h\int_0^1 u\,\mathrm{d}\sigma\right)-\frac{\partial}{\partial y}\left(h\int_0^1 v\,\mathrm{d}\sigma\right), \tag{3}$$

with boundaries

$$0\leqslant x\leqslant L, \qquad 0\leqslant y\leqslant B, \qquad 1\geqslant\sigma\geqslant 0.$$

Thus the domain is a rectangular basin. Owing to the sigma transformation in the vertical the domain is constant in time.[8,10] We have the closed boundary conditions

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

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

$$-\left(N\frac{\partial u}{\partial\sigma}\right)_0=hF_s, \qquad -\left(N\frac{\partial v}{\partial\sigma}\right)_0=hG_s,$$

and at the bottom ($\sigma=1$)

$$-\left(N\frac{\partial u}{\partial\sigma}\right)_1=hF_b, \qquad -\left(N\frac{\partial v}{\partial\sigma}\right)_1=hG_b.$$

The bottom stress is parametrized using a linear law of bottom friction, which is of the form

$$F_b=g\rho u_d/C^2, \qquad G_b=g\rho v_d/C^2,$$

with u_d and v_d the components of the velocity at some depth near the bottom. The surface stresses are expressed as

$$F_s=W_f\cos\varphi, \qquad G_s=W_f\sin\varphi.$$

3. SPACE DISCRETIZATION

For the space discretization of the equations (1)–(3) the computational domain is covered by an $nx\cdot ny\cdot ns$ rectangular staggered grid. Owing to the sigma transformation we have a constant number of grid layers in the vertical direction. In what follows U is a grid function whose

components $U_{i,j,k}$ approximate the velocity u. The components $U_{i,j,k}$ are numbered lexicographically. Likewise **V**, **Z**, **D** and **H** are grid functions for v, ζ, d and h respectively. Figure 1 shows the horizontal grid spacing. Note that **D**, **H** and **Z** are only computed at the upper layer. Furthermore, $\mathbf{\Lambda}_{\sigma\sigma}$ is a tridiagonal matrix approximating the vertical diffusion term, $\mathbf{\Theta}_1$ is an $(nx \cdot ny \cdot ns) \times (nx \cdot ny)$ matrix (a row of ns diagonal matrices of order $(nx \cdot ny)^2$ with $\Delta\sigma_k$ on the diagonal of the kth submatrix), $\mathbf{\Theta}_2$ is an $(nx \cdot ny) \times (nx \cdot ny \cdot ns)$ matrix (a column of ns identity matrices of order $(nx \cdot ny)^2$), **F** is a four-diagonal matrix (due to the grid staggering) of order $(nx \cdot ny \cdot ns)^2$ appoximating the Coriolis term, $\mathbf{D}_x$ and $\mathbf{D}_y$ are bidiagonal matrices (one diagonal and one lower diagonal) of order $(nx \cdot ny)^2$ approximating the differential operators $\partial/\partial x$ and $\partial/\partial y$ respectively, and $\mathbf{E}_x$ and $\mathbf{E}_y$ are bidiagonal matrices (one diagonal and one upper diagonal) with $\mathbf{E}_x = -\mathbf{D}_x^{\mathrm{T}}$ and $\mathbf{E}_y = -\mathbf{D}_y^{\mathrm{T}}$. The matrices $\mathbf{D}_x$ and $\mathbf{E}_x$ differ because of the grid staggering. We remark that $\mathbf{\Lambda}_{\sigma\sigma}$ also contains the discretization of the term $1/\rho h^2$.

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

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{W} = \mathbf{F}(\mathbf{W}) = \begin{bmatrix} \mathbf{\Lambda}_{\sigma\sigma} & \mathbf{F} & -\mathbf{\Theta}_2 \mathbf{g} \mathbf{D}_x \\ -\mathbf{F} & \mathbf{\Lambda}_{\sigma\sigma} & -\mathbf{\Theta}_2 \mathbf{g} \mathbf{E}_y \\ -\mathbf{\Theta}_1 \mathbf{H} \mathbf{E}_x & -\mathbf{\Theta}_1 \mathbf{H} \mathbf{D}_y & 0 \end{bmatrix} \mathbf{W} + \begin{bmatrix} \mathbf{F}_u \\ \mathbf{F}_v \\ 0 \end{bmatrix}, \tag{4}$$

where $\mathbf{W} = (\mathbf{U}, \mathbf{V}, \mathbf{Z})^{\mathrm{T}}$ and $(\mathbf{F}_u, \mathbf{F}_v, 0)^{\mathrm{T}}$ contains the components of the wind stress. Note that the integrals in (3) are approximated by $\mathbf{\Theta}_1 \mathbf{U}$ and $\mathbf{\Theta}_1 \mathbf{V}$ respectively.

4. TIME INTEGRATION

In this section we develop a time integration method for the semidiscretized system (4). We apply a two-stage time-splitting method of the form

$$\begin{aligned} \mathbf{W}^{n+1/2} &= \mathbf{W}^n + \tfrac{1}{2}\tau[\mathbf{F}^1(\mathbf{W}^{n+1/2}) + \mathbf{G}^1(\mathbf{W}^n) + \mathbf{C}^{n+1/2}], \\ \mathbf{W}^{n+1} &= \mathbf{W}^{n+1/2} + \tfrac{1}{2}\tau[\mathbf{F}^2(\mathbf{W}^{n+1/2}) + \mathbf{G}^2(\mathbf{W}^{n+1}) + \mathbf{C}^{n+1/2}], \end{aligned} \tag{5}$$

where $\mathbf{C} = (\mathbf{F}_u, \mathbf{F}_v, 0)^{\mathrm{T}}$, τ denotes the time step and $\mathbf{W}^n$ is a numerical approximation to $\mathbf{W}(t)$ of (4) at $t = n\tau$. Several well-known splitting methods, e.g. ADI methods, can be written in this form. In

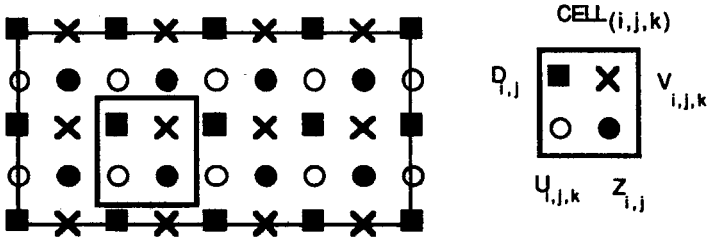

Figure 1. The staggered grid in the (x, y)-plane

this paper we choose

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

$$\mathbf{G}^1(\mathbf{W}^n)=\begin{bmatrix}0 & \mathbf{F} & -\Theta_2\mathbf{g}\mathbf{D}_x\\ 0 & 0 & -\Theta_2\mathbf{g}\mathbf{E}_y\\ -\Theta_1\mathbf{H}^n\mathbf{E}_x & -\Theta_1\mathbf{H}^n\mathbf{D}_y & 0\end{bmatrix}\mathbf{W}^n,$$

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

$$\mathbf{G}^2(\mathbf{W}^{n+1})=\begin{bmatrix}0 & 0 & -\Theta_2\mathbf{g}\mathbf{D}_x\\ 0 & 0 & -\Theta_2\mathbf{g}\mathbf{E}_y\\ -\Theta_1\mathbf{H}^{n+1}\mathbf{E}_x & -\Theta_1\mathbf{H}^{n+1}\mathbf{D}_y & 0\end{bmatrix}\mathbf{W}^{n+1}. \tag{6}$$

Apart from the Coriolis term $\mathbf{F}$, all terms are treated in a symmetrical way. When we neglect the Coriolis term, the time-splitting method (5)–(6) 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

$$\begin{bmatrix}\mathbf{I}-\frac{1}{2}\tau\Lambda_{\sigma\sigma} & 0 & 0\\ \frac{1}{2}\tau\mathbf{F} & \mathbf{I}-\frac{1}{2}\tau\Lambda_{\sigma\sigma} & 0\\ 0 & 0 & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{U}^{n+1/2}\\ \mathbf{V}^{n+1/2}\\ \mathbf{Z}^{n+1/2}\end{bmatrix}=\mathbf{B}^n, \tag{7}$$

where $\mathbf{B}^n$ 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-components the implicit treatment of the vertical diffusion term requires the solution of $nx\cdot ny$ tridiagonal systems of order ns.[9] 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 (7)) prohibits the U- and V-components 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

$$\begin{bmatrix}\mathbf{I} & 0 & \frac{1}{2}\tau\Theta_2\mathbf{g}\mathbf{D}_x\\ 0 & \mathbf{I} & \frac{1}{2}\tau\Theta_2\mathbf{g}\mathbf{E}_y\\ \frac{1}{2}\tau\Theta_1\mathbf{H}^{n+1}\mathbf{E}_x & \frac{1}{2}\tau\Theta_1\mathbf{H}^{n+1}\mathbf{D}_y & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{U}^{n+1}\\ \mathbf{V}^{n+1}\\ \mathbf{Z}^{n+1}\end{bmatrix}=\mathbf{B}^{n+1/2}, \tag{8}$$

where $\mathbf{B}^{n+1/2}$ contains the discretizations at time level $t=(n+1/2)\tau$. The equations for the U- and V-components are linear and are not coupled with each other. They are only coupled with the

equation for the Z-component. Therefore the components $\mathbf{U}^{n+1}$ and $\mathbf{V}^{n+1}$ can easily be eliminated from (8) and a system in the unknown $\mathbf{Z}^{n+1}$ results. Thus at the second stage the continuity equation (3) and the water elevation gradient in the momentum equations (1), (2) are treated implicitly. This approach was originally proposed in Reference 11 and has been applied by many others.[5-7]

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

$$Z_{i,j}^{n+1} - \frac{\tau^2 g}{4(\Delta x)^2}[\bar{H}_{i+1,j}^{n+1}(Z_{i+1,j}^{n+1} - Z_{i,j}^{n+1}) - \bar{H}_{i,j}^{n+1}(Z_{i,j}^{n+1} - Z_{i-1,j}^{n+1})]$$

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

$$= B_{i,j}^{n+1/2} \quad \text{for} \begin{cases} i = 1, \ldots, nx \\ j = 1, \ldots, ny \end{cases} \tag{9}$$

where

$$\bar{H}_{i,j}^{n} = Z_{i,j}^{n} + \tfrac{1}{2}(D_{i,j} + D_{i,j-1}), \qquad \tilde{H}_{i,j}^{n} = Z_{i,j}^{n} + \tfrac{1}{2}(D_{i,j} + D_{i+1,j}).$$

Note that $\bar{H}$ and $\tilde{H}$ differ because of the grid staggering. System (9) is a non-linear equation because $H_{i,j}$ contains the component $Z_{i,j}$. When system (9) has been solved, the values for the components $\mathbf{U}^{n+1}$ and $\mathbf{V}^{n+1}$ can be computed by back substitution.

System (9) can be written in the form

$$\mathbf{A}(\mathbf{Z}^{n+1})\mathbf{Z}^{n+1} = \mathbf{B}_z^{n+1/2} \tag{10}$$

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

$$\mathbf{A}(\mathbf{Z}^{(q)})\mathbf{Z}^{(q+1)} = \mathbf{B}_z^{n+1/2} \tag{11}$$

with $\mathbf{Z}^{(0)} = \mathbf{Z}^{n+1/2}$. In (11) the upper index (q) denotes the iteration index. The matrix $\mathbf{A}(\mathbf{Z}^{(q)})$ is a symmetric and strictly diagonal dominant matrix with positive values on the main diagonal and negative ones elsewhere because we require that $\mathbf{D} + \mathbf{Z}^{(q)} (= \mathbf{H}^{(q)}) > \mathbf{0}$. Thus system (11) is positive definite. In Section 6 we will discuss iteration methods for the solution of system (11).

It should be noted that the water elevation is the only unknown in system (11). 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 (5)–(6), because for two-dimensional problems a major part of the computation is required for the solution of this system.

The linearization process (11) was first used by Leendertse.[2] Conservation of mass remains guaranteed by this process. A slightly different linearization process has been introduced in Reference 5. In our numerical experiments (see Section 7) we obtained comparable results for both linearization processes. The linearization process will be a topic of further research.

5. STABILITY

We now analyse the stability of method (5)–(6) with the matrix method. In this section we omit the Coriolis force and the inhomogeneous term. It is well known that the Coriolis force hardly affects the stability. In this section we make plausible that the simplified method is un-

conditionally stable. The stability analysis used here is similar to the one described in Reference 12. 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 non-linear ones.

Since we have omitted the Coriolis force and the inhomogeneous term, we have that (cf. (6))

$$\mathbf{F}^1(\mathbf{W}^n)=\mathbf{F}^2(\mathbf{W}^n)=\mathbf{A}^n\mathbf{W}^n, \qquad \mathbf{G}^1(\mathbf{W}^n)=\mathbf{G}^2(\mathbf{W}^n)=\mathbf{B}^n\mathbf{W}^n,$$

where

$$\mathbf{A}^n=\begin{bmatrix} \Lambda_{\sigma\sigma} & 0 & 0 \\ 0 & \Lambda_{\sigma\sigma} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{B}^n=\begin{bmatrix} 0 & 0 & -\Theta_2 g\mathbf{D}_x \\ 0 & 0 & -\Theta_2 g\mathbf{E}_y \\ -\Theta_1\mathbf{H}^n\mathbf{E}_x & -\Theta_1\mathbf{H}^n\mathbf{D}_y & 0 \end{bmatrix}$$

Then method (5)–(6) can be written in the form

$$\mathbf{W}^{n+1}=(\mathbf{I}-\tfrac{1}{2}\tau\mathbf{B}^{n+1})^{-1}(\mathbf{I}+\tfrac{1}{2}\tau\mathbf{A}^{n+1/2})(\mathbf{I}-\tfrac{1}{2}\tau\mathbf{A}^{n+1/2})^{-1}(\mathbf{I}+\tfrac{1}{2}\tau\mathbf{B}^n)\mathbf{W}^n. \tag{12}$$

Let us now define the amplification matrix

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

In order to guarantee that method (12) is stable, we have to require $\|\prod_{i=0}^{n-1}\mathbf{C}^i\|$ to remain uniformly bounded for all values of n and τ such that

$$\left\|\prod_{i=0}^{n-1}\mathbf{C}^i\right\|<K \quad \text{for } 0<n\tau<T \text{ and } K \text{ constant.}^{13} \tag{13}$$

Let us now verify this condition for the numerical method (12). We have that

$$\left\|\prod_{i=0}^{n-1}\mathbf{C}^i\right\|\leqslant\|(\mathbf{I}-\tfrac{1}{2}\tau\mathbf{B}^n)^{-1}\|\,\|(\mathbf{I}+\tfrac{1}{2}\tau\mathbf{A}^{n-1/2})(\mathbf{I}-\tfrac{1}{2}\tau\mathbf{A}^{n-1/2})^{-1}\|$$

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

It can be verifed that both the matrix $\mathbf{A}^n$ and the matrix $\mathbf{B}^n$ have their eigenvalues in the left half-plane. The eigenvalues of the matrix $\mathbf{A}^n$ are even real non-positive. Therefore we have that

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

$$\text{for } i=0,\ldots,n-1.$$

Using these relations, we obtain that

$$\left\|\prod_{i=0}^{n-1}\right\|\leqslant\|(\mathbf{I}-\tfrac{1}{2}\tau\mathbf{B}^n)^{-1}(\mathbf{I}+\tfrac{1}{2}\tau\mathbf{B}^0)\|.$$

It is evident that only the explicit part $(\mathbf{I}+\frac{1}{2}\tau\mathbf{B}^0)$ may cause problems. In general, it is not possible to find an upper bound for $\|(\mathbf{I}+\frac{1}{2}\tau\mathbf{B}^0)\|$. In such a situation we may stabilize our integration method by computing the first approximation $\mathbf{W}^1$ by the backward Euler–LOD method and apply method (12) for $n\geqslant 1$. This technique has been proposed in Reference 12. We thus consider the method with first time step

$$\mathbf{W}^1=(\mathbf{I}-\tfrac{1}{2}\tau\mathbf{B}^1)^{-1}(\mathbf{I}-\tfrac{1}{2}\tau\mathbf{A}^{1/2})^{-1}\mathbf{W}^0, \tag{14a}$$

and for $n \geqslant 1$,

$$\mathbf{W}^{n+1} = (\mathbf{I} - \tfrac{1}{2}\tau\mathbf{B}^{n+1})^{-1}(\mathbf{I} + \tfrac{1}{2}\tau\mathbf{A}^{n+1/2})(\mathbf{I} - \tfrac{1}{2}\tau\mathbf{A}^{n+1/2})^{-1}(\mathbf{I} + \tfrac{1}{2}\tau\mathbf{B}^{n})\mathbf{W}^{n}. \tag{14b}$$

On a fixed space grid the LOD method (14a) is only first-order-accurate in time, but since we only perform one LOD step, method (14) is still second-order-accurate on fixed space grids. Using method (14), we obtain that

$$\left\| \prod_{i=0}^{n-1} \mathbf{C}^{i} \right\| \leqslant \| (\mathbf{I} - \tfrac{1}{2}\tau\mathbf{B}^{n})^{-1} \| \, \| (\mathbf{I} + \tfrac{1}{2}\tau\mathbf{A}^{1/2})^{-1} \|.$$

Now condition (13) is satisfied. We have no practical experience with method (14). In the numerical experiments no large errors were found in the original method (12) even for very large time steps. Thus there was no need for stabilization. In Reference 12 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.

6. SOLVING THE LINEAR SYSTEMS

In this section we describe how the linear systems at both stages, i.e. system (7) and system (11), 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 seems to be unattractive 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 Reference 9 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 (11). In the literature a large number of iterative 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 $\mathbf{S}$ such that the system

$$\mathbf{SAZ}^{q+1} = \mathbf{SB}$$

has a much smaller condition number than the original system $\mathbf{AZ}^{q+1} = \mathbf{B}$. For the preconditioning of system (11) we will use a smoothing matrix $\mathbf{S}$ of the form $\mathbf{S} = P(\mathbf{D})$, where $P(z)$ is a polynomial and $\mathbf{D}$ is some matrix. The matrix $\mathbf{D}$ will be a difference matrix of which the eigenvalues are assumed to be in the interval $[-1, 0]$.[14] The polynomial $P(z)$ will be chosen such that $P(0) = 1$ and the eigenvalues of $\mathbf{S}$ are in the interval $[0, 1]$. First we discuss the choice of the matrix $\mathbf{D}$. In our *theoretical* considerations we assume that $\mathbf{D}$ is equal to the normalized matrix $\mathbf{A}$, i.e.

$$\mathbf{D} = \mathbf{A}/\rho(\mathbf{A}), \tag{15}$$

where $\rho(\,)$ denotes the spectral radius. We emphasize that in *practice* it is generally not attractive to choose $\mathbf{D}$ according to (15) and we shall employ some cheap approximation to the normalized matrix. If $\mathbf{D}$ is defined according to (15), then the eigenvalues of $\mathbf{SA} = P(\mathbf{D})\mathbf{A}$ are given by $\rho(\mathbf{A})zP(z)$, where z runs through the spectrum of $\mathbf{D}$. 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[15]

$$P_{2^q-1}(z) = \prod_{k=1}^{q} \left(1 + \mu \frac{T_{2^{k-1}}(1+2z) - 1}{2} \right), \qquad T_p(x) = \cos[p \cos^{-1}(x)]. \tag{16}$$

It has been proved that the spectral radius of the matrix $\mathbf{SA}$ is minimized by the polynomial (16) when the matrix $\mathbf{A}$ has real non-positive eigenvalues and $\mu=1$.[14] For $\mu=1$ we have that (16) is equal to

$$P_{2q-1}(z)=\frac{T_{2q}(1+2z)-1}{2z}\frac{1}{4^q}. \tag{17}$$

The factorization in (16) makes it possible to implement the smoothing operator $\mathbf{S}$ in a very efficient way. This is stated in the following theorem.

Theorem 1

Let $\mathbf{S}=P(\mathbf{D})$ with $P(z)$ defined by (16) and let the factor matrices $\mathbf{F}_j$ be defined by

$$\mathbf{F}_j=\mathbf{I}+\mu\mathbf{D}_j, \quad \text{where} \quad \mathbf{D}_{j+1}=4\mathbf{D}_j(\mathbf{I}+\mathbf{D}_j) \text{ with } \mathbf{D}_1=\mathbf{D} \text{ and } j\geqslant 1. \tag{18}$$

Then $\mathbf{S}=P_{2q-1}(\mathbf{D})$ can be factorized according to

$$\mathbf{S}=\mathbf{F}_q\mathbf{F}_{q-1}\ldots\mathbf{F}_1. \qquad \square \tag{19}$$

Thus the smoothing matrix $\mathbf{S}$ consists of q factor matrices. For a proof of Theorem 1 we refer to Reference 15.

The most efficient implementation of $\mathbf{S}$ is based on the factorization property in Theorem 1. However, in two or more dimensions the precomputation of the factor matrices $\mathbf{F}_j$ defined by (18) is not attractive. Therefore we consider an alternative smoothing matrix $\mathbf{S}$ which consists only of one-dimensional operators. For our two-dimensional problem (11) we apply one-dimensional smoothing in the x- and y-directions successively. An extra advantage of the splitting in one-dimensional operators is that the application of the smoothing operator $\mathbf{S}$ is now hardly complicated when the domain is irregular.[10]

As mentioned before in practice we shall choose $\mathbf{D}$ equal to some cheap approximation of (15). Since the smoothing matrix $\mathbf{S}$ consists of one-dimensional operators, we choose

$$\mathbf{D}=(1/4)\begin{bmatrix} -1 & 1 & & & & 0 \\ 1 & -2 & 1 & & & \\ & \cdot & \cdot & & & \\ & & \cdot & \cdot & & \\ & & & 1 & -2 & 1 \\ 0 & & & & 1 & -1 \end{bmatrix}. \tag{20}$$

If the factor matrices of (19) are computed in advance, then the evaluation of $P(\mathbf{D})$ 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 1 for matrix $\mathbf{D}$ in (20), we find the factor matrices

$$\mathbf{F}_1=(1/4)\begin{bmatrix} 3 & 1 & & & & 0 \\ 1 & 2 & 1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & 1 & 2 & 1 \\ 0 & & & & 1 & 3 \end{bmatrix}, \quad \mathbf{F}_2=(1/4)\begin{bmatrix} 2 & 1 & 1 & & & & 0 \\ 1 & 2 & 0 & 1 & & & \\ 1 & 0 & 2 & 0 & 1 & & \\ & 1 & 0 & 2 & 0 & 1 & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \quad \text{etc.} \tag{21}$$

Evidently the matrix–vector multiplications with these essentially three-diagonal 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 (11).

6.1. *The smoothed Jacobi method*

For the solution of system (11) we apply the smoothed Jacobi method[14]

$$\mathbf{Z}_{k+1}=\mathbf{Z}_k+\omega\mathbf{S}(\mathbf{B}-\mathbf{A}\mathbf{Z}_k), \quad k=1, 2, 3, \ldots, \tag{22}$$

where $\mathbf{Z}_k$ is the kth iterate, ω is a relaxation parameter and $\mathbf{S}$ is the smoothing matrix described in Theorem 1 with $\mathbf{D}$ as in (20). For the smoothed Jacobi method we choose $\mu=1$ (see (18)). As mentioned in the previous section, the smoothing matrix $\mathbf{S}$ consists of q smoothing factors. It has been demonstrated that one should not iterate with a fixed value of q.[14] Therefore we choose the number of smoothing factors at the kth iteration step equal to $k \bmod (q+1)$, which yields the cyclic sequence of $1, 2, \ldots, q, 0, 1, 2, \ldots, q, 0, 1, 2, \ldots, q, \ldots$ smoothing factors.

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

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

Similarly, for the spectral radius of $\mathbf{SA}$ we have that

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

Following the analysis in Reference 14, we obtain

$$\omega=2/\rho(\mathbf{SA}). \tag{23}$$

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.

$$\omega_{i,j}=2\Big/\left[1+\frac{1}{4^q}gH_{i,j}\left(\frac{\tau^2}{(\Delta x)^2}+\frac{\tau^2}{(\Delta y)^2}\right)\right]. \tag{24}$$

In the case of a fixed relaxation parameter we observed in our experiments that (23) was the optimum relaxation parameter. However, we obtained much better results with the relaxation parameter in (24) when an irregular bottom topography was used.

6.2. *The smoothed CG method*

The second iteration method that we applied for the solution of system (11) is a preconditioned CG method. The preconditioned CG method can be formulated as follows.

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

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

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

$$\begin{aligned} \alpha_k &= \frac{\mathbf{R}_k^{\mathrm{T}}(\mathbf{SR}_k)}{\mathbf{P}_k^{\mathrm{T}}(\mathbf{AP}_k)}, \\ \mathbf{Z}_{k+1} &= \mathbf{Z}_k + \alpha_k \mathbf{P}_k, \\ \mathbf{R}_{k+1} &= \mathbf{R}_k - \alpha_k \mathbf{AP}_k, \\ \beta_k &= \frac{\mathbf{R}_{k+1}^{\mathrm{T}}(\mathbf{SR}_{k+1})}{\mathbf{R}_k^{\mathrm{T}}(\mathbf{SR}_k)}, \\ \mathbf{P}_{k+1} &= \mathbf{SR}_{k+1} + \beta_k \mathbf{P}_k. \end{aligned} \tag{25}$$

In (25) 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.[16] Here we choose a positive definite matrix **S** of the form $\mathbf{S}=P(\mathbf{D})$, where **D** is the difference matrix in (20) and $P(z)$ is the polynomial (16). By choosing $\mu \in [0, 1)$ 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**.

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 (5)–(6). 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:

$f=0{\cdot}44/3600=1{\cdot}22\times10^{-4}$

$g=9{\cdot}81\ \mathrm{m\,s^{-2}}$

$N=0{\cdot}065/\rho$

$\rho=1025\ \mathrm{kg\,m^{-3}}$

$\varphi=45^{\circ}$ (north-eastern wind).

The experiments have been carried out on an ALLIANT FX/4. This mini-supercomputer consists of 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 was compared with a reference solution computed on the same grid with $\tau=60$ s. The reference solution may be considered as an almost exact solution of our semidiscretized system (4). Thus the accuracy results listed in this section represent the error due to the time integration.

In the experiments we used a rectangular basin of 400 km by 800 km with different bottom topographies. For the grid sizes we have chosen $\Delta x=10$ km, $\Delta y=10$ km and $\Delta\sigma=0{\cdot}2$ m. Thus the computations have been performed on a grid with $nx=41$, $ny=81$ and $ns=5$. We integrated over

a period of 5 days with the time-dependent wind stress

$$W_f = 1{\cdot}5\left[1 + 0{\cdot}5\sin\left(\frac{2\pi t}{24 \times 3600}\right)\right]. \tag{26}$$

Thus we have a periodically varying (north-eastern) wind with a period of 24 h. To measure the obtained accuracy, we define

ERR-x: maximal global error for component x, with $x = \zeta$, u or v, compared with the reference solution at the end point $T = 120$ h. (27)

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 I we list the maximal global errors for the test problem with a channel in a diagonal direction. In this experiment the maximal values for ζ, u and v are about 2·6 m, 0·4 m s^{-1} and 1·1 m s^{-1} respectively. We observed that after a few days the solution became periodic with a period of 24 h 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 of the order 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 $\tau = 4800$ s the phase error is about 1 h. When we compare the numerical solution computed with $\tau = 4800$ s at $T = 121{\cdot}33$ h with the reference solution at $T = 120$ h, the maximal global errors for

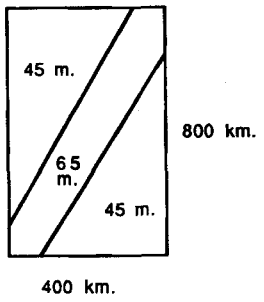

Figure 2. Plane bottom with a channel

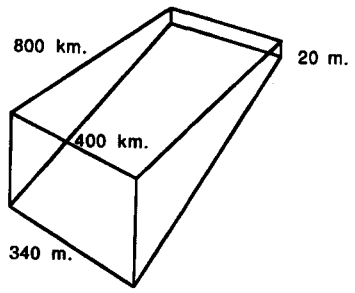

Figure 3. Inclined bottom

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

τ (s)	ERR-ζ (m)	ERR-u (m s^{-1})	ERR-v (m s^{-1})
600	0·038	0·006	0·018
1200	0·087	0·014	0·041
2400	0·212	0·035	0·103
4800	0·549	0·088	0·269

the three components are 0·124 m, 0·050 m s^{-1} and 0·054 m s^{-1} respectively. This is significantly less than in Table I.

We now discuss the computational efficiency of the time integration method (5), (6). To represent the results we use the following notation:

q	number of smoothing factors (see (19))
μ	smoothing coefficient (see (16))
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 (see (10))

$$r(k) = \| \mathbf{B}_z^{n+1/2} - \mathbf{A}\mathbf{Z}_k \|_\infty$$

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

In Table II we list the computation times and the convergence results for the time integration method (5)–(6) 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 μ in the preconditioning matrix of the SCG method we experimentally derived an optimum value. As mentioned in Section 6.1, for the SJAC method we have $\mu=1$.

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, while the computation time for the iteration process is reduced by a factor of 5·5.

Table II. Computation times for the channel problem

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

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 II we have listed the optimum values for μ. For values in the neighbourhood of the optimum value the number of iterations hardly increases. Thus the choice of the parameter μ 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 III we list the maximal global errors for the test problem with an inclined bottom (see Figure 3). In this experiment the maximal values for ζ, u and v are about 1·2 m, 0·7 m s^{-1} and 1·4 m s^{-1} respectively. The results are comparable with the results in the first experiment. After a few days the numerical solution also became periodic with a period of 24 h for any time step τ. However, in this experiment the phase errors are much smaller.

The computational results in this experiment, which are listed in Table IV, 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.

In the experiments we used both the vector and the parallel optimization of the ALLIANT FX/4. For both iteration methods the computation time was reduced by about a factor of three

Table III. Test problem with an inclined bottom

τ (s)	ERR-ζ (m)	ERR-u (m s^{-1})	ERR-v (m s^{-1})
600	0·016	0·005	0·003
1200	0·018	0·008	0·005
1800	0·035	0·012	0·009
3600	0·122	0·032	0·034

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

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 V. Computation times for different numbers of vertical layers

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

by the vectorization and also by a 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 (5)–(6) was reduced by the above-mentioned factors. This shows that our integration method (5)–(6), in which either the SCG method or the SJAC method has been applied, can be implemented efficiently on vector and parallel computers.

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 (5)–(6) 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 $\mu=0{\cdot}925$ (see Table II). Table V 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 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 (5)–(6) is very suited for three-dimensional problems, especially when large time steps are used.

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 was 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 required less computation time than the SJAC method.

It has been shown that the time integration method presented in this paper is very 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 relatively more efficient for three-dimensional problems than for two-dimensional problems. It was reported that a time-splitting method of this form is already feasible for practical computations of two-dimensional problems.[5]

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

REFERENCES

1. A. R. Gourlay, 'Splitting methods for time dependent partial differential equations', in D. Jacobs (ed.), *The State of the Art in Numerical Analysis*, Academic Press, London/New York/San Francisco, 1977, pp. 757–791.
2. J. J. Leendertse, 'Aspects of a computational model for long period water wave propagation', *Memorandum RM-5294-PR*, Rand Corp., Santa Monica, CA, 1967.
3. G. S. Stelling, 'On the construction of computational methods for shallow water flow problems', *Ph.D. Thesis*, Delft University, 1983.
4. G. S. Stelling, A. K. Wiersma and J. B. T. M. Willemse, 'Practical aspects of accurate tidal computations', *J. Hydraul. Eng., ASCE*, **112**, 802–817 (1986).
5. 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. methods eng.*, **26**, 2707–2721 (1988).
6. J. O. Backhaus, 'A semi-implicit scheme for the shallow water equations for application to shelf sea modelling', *Continental Shelf Res.*, **2**, 243–254 (1983).
7. V. Casulli, 'Semi-implicit finite difference methods for the two dimensional shallow water equations', *J. Comput. Phys.*, **86**, 56–74 (1990).
8. 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. methods fluids*, **5**, 405–425 (1985).
9. E. D. de Goede, 'A computational model for the three-dimensional shallow water flows on the ALLIANT FX/4', *Supercomputer*, **32**, 43–49 (1988).
10. E. D. de Goede, 'Stabilization of a time integrator for the 3D shallow water equations by smoothing techniques', *Int. j. numer. methods fluids*, to appear.
11. Y. Kurihara, 'On the use of implicit and iterative methods for the time integration of the wave equation', *Mon. Weather Rev.*, **93**, 33–46 (1965).
12. J. Verwer and W. H. Hundsdorfer, 'Stability and convergence of the Peaceman–Rachford ADI method for initial-boundary problems', *Math. Comput.*, **53**, 81–101 (1989).
13. P. D. Lax and R. D. Richtmyer, 'Survey of the stability of linear finite difference equations', *Commun. Pure Appl. Math.*, **17**, 267–293 (1956).
14. 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).
15. F. W. Wubs, 'Numerical solution of the shallow-water equations', *Ph.D. Thesis*, University of Amsterdam, 1987.
16. J. Ortega and R. Voigt, 'Solution of partial differential equations on vector and parallel computers', *SIAM Rev.*, **27**, 149–240 (1985).