Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

F.W. Wubs

Evaluation of time integrators for the shallow-water equations

Department of Numerical Mathematics Report NM-R8406 June

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

EVALUATION OF TIME INTEGRATORS FOR THE SHALLOW-WATER EQUATIONS

F.W. WUBS
Centre for Mathematics and Computer Science, Amsterdam

In this report some well-known time integrators for the shallow-water equations are described and compared with each other with respect to efficiency. In fact, the efficiencies of low-order versus high-order methods and of implicit methods versus explicit methods are considered. Numerical experiments for a problem posed by GRAMMELTVEDT [4] showed a good performance of the classical fourth-order Runge-Kutta method. However, conclusions should be taken with care because the influence of the error due to the space discretization is not taken into account.

1980 MATHEMATICS SUBJECT CLASSIFICATION: Primary: 65M20. Secondary: 76B15.
KEY WORDS & PHRASES: method of lines, linear stability, shallow-water equations, efficiency of time integrators, storage reduction.
NOTE: These investigations were supported (in part) by the Netherlands Foundation for Technical Research (STW), future Technical Science Branch/ Division of the Netherlands Organization for the Advancement of Pure Research (ZWO).

Report NM-R8406
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Contents

1. INTRODUCTION

In this report some well-known time integrators for the shallow-water equations (SWEs) are described and compared with each other with respect to efficiency. Standard software for initial value problems [8] is often not applicable to the SWEs, due to the large system of difference equations (easily over 10,000) occurring after discretization. In the last decades, many methods have been proposed to solve the SWEs. Therefore, there is the need for an evaluation of these and new methods with respect to accuracy, stability, flexibility and computational costs, because, in practice, reliability and the time needed to obtain results are crucial.

Here, we will apply the so-called method of lines approach. In using this approach, the hyperbolic partial-differential equations (PDEs) are converted into a system of ordinary-differential equations (ODEs) by discretizing the space variables, while the time variable is left continuous. As a first step some time integrators have been studied for such a system, where the space discretization for all time integrators is the same.

In fact, we are interested in the efficiency of low-order explicit time integrators versus high-order explicit methods and in the efficiency of implicit methods versus explicit methods. To answer these questions, numerical tests were performed for a problem posed by GRAMMELTVEDT [4]. These tests show that the choice of a method depends largely on the model and the accuracy requirements.

In section 1, the PDEs will be given and the space variables will be discretized, furthermore, a heuristic determination of the eigenvalues will be given. In section 2 the time integrators are described i.e.,

1. an ADI method proposed by FAIRWEATHER & NAVON [1] (1980),
2. the classical fourth-order Runge-Kutta method [10] (1901),
3. a stabilized Runge-Kutta method [7] (1971) and
4. the Leap-Frog method [12] (1922).

The results are presented in section 3 and finally, in section 4, the methods and results are evaluated.

1.2. The differential equations and boundary conditions

Throughout this report we shall adopt de notation used in [1]. We denote by w the vector

$$w = (u,v,\Phi)^T, \tag{1.1}$$

where u,v are the velocity components in the x- and y-direction, respectively, and

$$\Phi = 2(gh)^{\frac{1}{2}}, \tag{1.2}$$

where h is the depth of the fluid and g is the acceleration due to gravity. The shallow-water equations then read [5]

$$\frac{\partial w}{\partial t} = A(w)\,\frac{\partial w}{\partial x} + B(w)\,\frac{\partial w}{\partial y} + C(y)w,$$

$$0 \le x \le L, \quad 0 \le y \le D, \quad t \ge 0. \tag{1.3}$$

In (1.3) A,B, and C are matrices given by

$$A = -\begin{bmatrix} u & 0 & \Phi/2 \\ 0 & u & 0 \\ \Phi/2 & 0 & u \end{bmatrix} \quad B = -\begin{bmatrix} v & 0 & 0 \\ 0 & v & \Phi/2 \\ 0 & \Phi/2 & v \end{bmatrix} \quad C = \begin{bmatrix} 0 & f & 0 \\ -f & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{1.4}$$

where f is the Coriolis factor approximated by

$$f = \hat{f} + \beta(y-D/2) \tag{1.5}$$

with $\hat{f}$ and β constants. Periodic boundary conditions are assumed in the x-direction.

$$w(x,y,t) = w(x+L,y,t), \tag{1.6}$$

while in the y-direction

$$v(x,0,t) = v(x,D,t) = 0. \tag{1.7}$$

Initially

$$w(x,y,0) = \psi(x,y). \tag{1.8}$$

It is easy to see that the sum E of kinetic and potential energy,

$$E = \tfrac{1}{2} \int_0^L \int_0^D (u^2+v^2+\Phi^2/4)\Phi^2/(4g)\,dxdy, \tag{1.9}$$

is independent of time, i.e., the total energy is conserved. Also, the average value of the height of the free surface, $\bar{h}$, is independent of time, i.e.,

$$\bar{h} = \Big(\int_0^L \int_0^D h\,dx\,dy\Big)/\bar{A} \tag{1.10}$$

is conserved, $\bar{A}$ being the area of the spatial domain.

1.2. The space discretization

As said in the introduction, we are interested in the efficiency of time integrators with respect to a fixed space discretization. Therefore, the PDEs (1.3) are transformed into a system of ODEs;

$$\frac{dW}{dt} = F(W), \tag{1.11}$$

where F(W) is the space discretization of the right-hand side of (1.3).

Let N_x and N_y be positive integers and set

$$\Delta x = L/N_x, \tag{1.12}$$

$$\Delta y = D/N_y,$$

then by W_{jk} we mean:

(1.13) $$W_{jk} = W(j\Delta x, k\Delta y, t).$$

Now, the space-discretized form of (1.3) we shall use, is

(1.14) $$\frac{dW_{jk}}{dt} = A(W_{jk})D_x W_{jk} + B(W_{jk})D_y(k)W_{jk} + C(y_k)W_{jk},$$

where $$D_x W_{jk} = (W_{j+1,k} - W_{j-1,k})/(2\Delta x)$$

$$D_y(k)W_{jk} = (W_{j,k+1} - W_{j,k+1})/(2\Delta y),\ k = 1, \ldots, N_y - 1,$$

$$D_y(0)W_{j0} = (W_{j,1} - W_{j,0})/\Delta y, \quad \text{and}$$

$$D_y(N_y)W_{jN_y} = (W_{jN_y} - W_{j,N_y-1})/\Delta y.$$

Hence, in the interior domain central discretizations are used, whereas at the boundaries, y = 0 and y = D, one-sided discretizations are applied in the y-direction. In the x-direction always second-order differences can be used as a result of the periodicity (1.6). This discretization was used before by GUSTAFSSON [5] and FAIRWEATHER and NAVON [1].

We notice that this discretization appeared to be unstable in long-time integrations as was found by the three mentioned authors. DEKKER and VERWER [15] proved that for certain initial conditions the total energy increases locally in time and derived a space discretization for (1.3) which conserves the total energy.

1.3. Determination of eigenvalues

In this subsection, we will calculate the eigenvalues associated with the ODEs, because the magnitude of these eigenvalues do often restrict the time step in the finite-difference equations (FDEs) (the fully discretized PDEs). For convenience, we shall assume an infinite domain (R^2), which in our case appeared to be not a severe restriction, presumably, because the only other operator involved, is the operator defined by the boundary conditions v = 0 at y = 0 and y = D, which is of course stable.

DEFINITION: *A system of* PDEs, ODEs *or* FDEs *is stable in the* L_2 *-norm on* R^2 *iff for any two solutions* w *and* $\tilde{w}$ *holds:*

$$(1.15) \qquad |w(t_1) - \tilde{w}(t_1)|_{L_2} \leq |w(t_0) - \tilde{w}(t_0)|_{L_2}, \; t_1 \geq t_0.$$

Of course, the domain for the ODEs and FDEs is a restriction of R^2 to the used grid and for the FDEs t_1 and t_0 may only assume discreet values in the time direction. We shall use this definition for small perturbations, which leads to a linear stability approach.

Let a PDE be defined by

$$(1.16) \qquad \frac{\partial}{\partial t} w = D(w) + f.$$

If w and $\tilde{w}$ are solutions of (1.16), we have

$$(1.17) \qquad \frac{\partial}{\partial t}(w-\tilde{w}) = D(w)-D(\tilde{w}).$$

Our definition of stability demands then that

$$(1.18) \qquad D(w) - D(\tilde{w}) \leq - \varepsilon(w-\tilde{w}), \quad \varepsilon \geq 0.$$

If D(w) is sufficient smooth then, for small values of $w-\tilde{w}$, (1.18) may be replaced by

$$(1.19) \qquad D(w) - D(\tilde{w}) \approx \frac{\partial D}{\partial w} (w-\tilde{w})$$

and

$$(1.20) \qquad \mathrm{Re}(s, \frac{\partial D}{\partial w} s) \leq 0, \; \forall s,$$

where (,) denotes the inner product connected with the L_2 - norm. As $\frac{\partial D}{\partial w}$ is often a matrix, (1.20) is difficult to verify, but if $\frac{\partial \bar{D}}{\partial w}$ is normal, non-positive eigenvalues are sufficient to fulfil (1.20); in other cases this is a neccesary condition.

In our case $\frac{\partial D}{\partial w}$ follows from (1.31) $((\frac{\partial D}{\partial w})_{ij} = \frac{\partial D_i}{\partial w_j}$ $i,j = 1,2,3)$

$$(1.21)\quad \frac{\partial D}{\partial w} = \begin{bmatrix} -u_x - u\frac{\partial}{\partial x} - v\frac{\partial}{\partial y} & -u_y + f & -\phi_x/2 - \phi/2\frac{\partial}{\partial x} \\ -v_x - f & -v_y - v\frac{\partial}{\partial y} - u\frac{\partial}{\partial x} & -\phi_y/2 - \phi/2\frac{\partial}{\partial y} \\ -\phi/2\frac{\partial}{\partial x} - \phi_x & -\phi/2\frac{\partial}{\partial y} - \phi_y & -\frac{1}{2}(u_x+v_y) - u\frac{\partial}{\partial x} - v\frac{\partial}{\partial y} \end{bmatrix}$$

We now freeze the coefficients in $\frac{\partial D}{\partial w}$ and write s as a Fourier series.

$$(1.22)\qquad s = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a(\underline{b})e^{i\underline{b}\cdot\underline{x}}\, db_1 db_2$$

It is sufficient to consider one component of (1.22). Hence, we try to find λ such that

$$(1.23)\qquad \left[\frac{\partial D}{\partial w} - \lambda I\right] \underline{a}e^{ib_1x+ib_2y} = \underline{0}$$

for the accomponying eigenvector $\underline{a}$. It is straight forward to show, as $\frac{\partial}{\partial x} e^{ib_1 x} = ib_1 e^{ib_1 x}$, that this equation can be replaced by

$$(1.24)\qquad \left[\frac{\partial D'}{\partial w}(b_1,b_2) - \lambda I\right] \underline{a}e^{ib_1x+ib_2y} = \underline{0}.$$

where

$$(1.25)\qquad \frac{\partial D'}{\partial w} = \begin{bmatrix} -u_x - ib_1u - ib_2v & -u_y + f & -\phi_x/2 - ib_1\phi/2 \\ -v_x - f & -v_y - ib_1u - ib_2v & -\phi_y/2 - ib_2\phi/2 \\ -ib_1\phi/2 - \phi_x & -\phi/2\, ib_2 - \phi_y & -\frac{1}{2}(u_x+v_y) - ib_1u - ib_2v \end{bmatrix}$$

The third degree characteristic polynomial resulting from (1.24) is difficult to factorize, therefore we make some simplifications beforehand, using that ϕ, related to the depth by (1.2), is relatively large. Thus we assume

$$(1.26)\qquad |u|, |v|, |u_x|, |u_y|, |v_x|, |v_y|, |\phi_x|, |\phi_y| \ll \phi.$$

Hereby, (1.25) reduces to the symmetric (thus normal) matrix

$$(1.27)\qquad \frac{\partial D'}{\partial w} = \begin{bmatrix} 0 & 0 & -ib_1\phi/2 \\ 0 & 0 & -ib_2\phi/2 \\ -ib_1\phi/2 & -ib_2\phi/2 & 0 \end{bmatrix}$$

The characteristic polynomial of $\frac{\partial D'}{\partial w}$ is now

$$(1.28)\qquad \lambda^3 + (b_1^2+b_2^2)\,\frac{\phi^2}{4}\,\lambda = 0.$$

Thus the eigenvalues are

$$(1.29)\qquad \lambda = 0 \quad \text{and} \quad \lambda(b_1,b_2) = \pm i\sqrt{(b_1^2+b_2^2)\phi^2/4}.$$

These eigenvalues show that (1.20) is marginally satisfied and that there is no damping of waves which corresponds to the conservation of total energy (1.9).

For the semi-discretized system (1.14) the eigenvalues in the interior of the computational domain can be obtained along the same lines. However, the central discretization used in (1.14) has a different eigenvalue, which follows from

$$(1.30)\qquad [\frac{\partial}{\partial x}]\, e^{ib_1x+ib_2x} = \frac{1}{2\Delta x}\left(e^{i(b_1(x+\Delta x)+b_2y} - e^{i(b_1(x-\Delta x)+b_2y)}\right)$$
$$= \frac{i\,\sin b_1\Delta x}{\Delta x}\; e^{ib_1x+ib_2x}.$$

By replacing b_1 and b_2 in (1.24)-(1.29) by $\frac{\sin b_1\Delta x}{\Delta x}$ and $\frac{\sin b_2\Delta y}{\Delta y}$, respectively, we find eigenvalues

$$(1.31)\qquad \lambda = 0 \quad \text{and} \quad \lambda(b_1,b_2) = \pm i\sqrt{\left\{\left(\frac{\sin b_1\Delta x}{\Delta x}\right)^2 + \left(\frac{\sin b_2\Delta y}{\Delta y}\right)^2\right\}\phi^2/4}.$$

From these eigenvalues it may be doubted whether (1.14) is stable, because (1.32) means marginally stability, and the neglected terms, due to (1.26), may easily cause an actual positive eigenvalue, as must be the case according to the observed instability of (1.14) for long time integration [1,5,15]

The next step is the time-discretization. Again (1.15) is used, but now t_0, t_1 are restricted to discreet t values. In explicit methods, (1.15) often leads to a restriction of the time step.
As a check of (1.15) for FDEs is very tedious, we shall use the simple test equation

$$\frac{dw'}{dt} = \lambda w', \quad w' = w - \tilde{w}, \tag{1.32}$$

which replaces (1.17) and where λ is the maximum of the above eigenvalues

$$\lambda = \pm i \sqrt{\left\{\left(\frac{1}{\Delta x}\right)^2 + \left(\frac{1}{\Delta y}\right)^2\right\}\phi^2/4}, \quad \phi = 2\sqrt{gh}. \tag{1.33}$$

With the help of the test equation we can investigate whether a time integrator satisfies (1.15), and gives information on the allowed time steps. However, one must be careful in applying (1.33):

1. If $\frac{\partial D}{\partial w}$ is not diagonizable, this will only give an indication. In our case $\frac{\partial D}{\partial w}$ is almost normal.
2. In implicit methods the D operator is split up, this requires a special treatment. An example is found in section 2.1.2.

Note that in this section time and space discretization are handled apart, i.e., we first tried to find conditions for a stable space discretization by examining the ODEs and then a condition for the time integrator is derived. However, an unstable space discretization may be matched by a stable time discretization, or vice versa, leading to a stable system of FDEs [3,13]. As an example we mention the well-known first-order upwind-finite-difference scheme which, when applied to $u_t = -u_x$, produces an exact solution for equal space and time step [3]. The approach we followed, is typical for the Method of Lines [14].

2. THE FOUR TIME INTEGRATORS

In this section, four time integrators are described. Also their stability, storage requirement and implementation will be considered.

2.1. The Fairweather-Navon ADI method.

2.1.1. Description

The time integration of (1.14) can be performed by using an ADI method. Thereby, the right-hand side operator is split into two operators, i.e. one for the x-direction and one for the y-direction. Based on a Crank-Nicolson-type time integrator, an ADI method is obtained with unconditional stability for non-positive eigenvalues. However, a straight forward application of this technique gives rise to a set of nonlinear algebraic equations to be solved (GUSTAFSSON [5]). This can of course be handled by a Newton process, but Fairweather and Navon linearized the algebraic equations in such a way that second-order consistency in time was preserved.

The split operators in the ADI method are defined by

$$F^1_{jk}(W_{jk}) = A(W_{jk})D_x W_{jk} + C^1_k W_{jk} \text{ and}$$

(2.1)

$$F^2_{jk}(W_{jk}) = B(W_{jk})D_y(k)W_{jk} + C^2_k W_{jk},$$

where

$$C^1_k = \begin{bmatrix} 0 & 0 & 0 \\ f_k & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } C^2_k = \begin{bmatrix} 0 & f_k & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} .$$

It will be clear that (compare (1.14))

$$\frac{dW_{jk}}{dt} = F^1_{jk}(W_{jk}) + F^2_{jk}(W_{jk}). \tag{2.2}$$

A non-linear ADI method [5] can be defined by

$$W^*_{jk} = W^n_{jk} + \frac{\Delta t}{2}\left(F^1_{jk}(W^*_{jk}) + F^2_{jk}(W^n_{jk})\right) \text{ and}$$

$$W^{n+1}_{jk} = W^*_{jk} + \frac{\Delta t}{2}\left(F^1_{jk}(W^*_{jk}) + F^2_{jk}(W^{n+1}_{jk})\right), \tag{2.3}$$

where

$$W^n_{jk} = W(j\Delta x, k\Delta y, n\Delta t).$$

The non-linearity is due to the matrices A and B in (2.1). Therefore, Fairweather and Navon slightly changed (2.1) to

$$F^1_{jk}(W_{jk}) = A(\hat{W}_{jk})D_x W_{jk} + C^1_k W_{jk} \quad \text{and}$$

$$F^2_{jk}(W_{jk}) = B(\hat{W}_{jk})D_y(k) W_{jk} + C^2_k W_{jk}, \tag{2.4}$$

where

$$\hat{W}^n_{jk} = \tfrac{1}{2}(3W^n_{jk} - W^{n-1}_{jk}) \quad \text{for } n \geq 1 \quad \text{and}$$
$$\hat{W}^0_{jk} = W^0_{jk} + \frac{\Delta t}{2} F_{jk}(W^0_{jk}).$$

It is straight forward to show that this method is locally second order in time and space except for the boundaries, where the space discretization is only first order (see (1.14)). In the remainder, we shall call this scheme FN-method.

2.1.2. Stability

Unfortunately, the test equation may not be used directly, which is caused by the operator splitting. Therefore, we try to find a form for (2.3), where it is possible to use the eigenvalues of the operator $\frac{\partial D}{\partial W}$. This can be achieved by taking the first variation of (2.3), which is equivalent to (1.19), assuming that the linearization (2.4) has not yet been performed. We find

$$\delta W^* = \delta W^n + \frac{\Delta t}{2}\frac{\partial F^1}{\partial W}\delta W^* + \frac{\Delta t}{2}\frac{\partial F^2}{\partial W}\delta W^n \quad \text{and} \tag{2.5}$$

$$\delta W^{n+1} = \delta W^* + \frac{\Delta t}{2}\frac{\partial F^1}{\partial W}\delta W^* + \frac{\Delta t}{2}\frac{\partial F^2}{\partial W}\delta W^{n+1}.$$

After some algebraic manipulations it follows that

$$\left(1 - \frac{\Delta t}{2}\left(\frac{\partial F^1}{\partial W} + \frac{\partial F^2}{\partial W}\right) + \frac{\Delta t^2}{4}\frac{\partial F^1}{\partial W}\frac{\partial F^2}{\partial W}\right)\delta W^{n+1} = \tag{2.6}$$

$$\left(1 + \frac{\Delta t}{2}\left(\frac{\partial F^1}{\partial W} + \frac{\partial F^2}{\partial W}\right) + \frac{\Delta t^2}{4}\frac{\partial F^1}{\partial W}\frac{\partial F^2}{\partial W}\right)\delta W^n.$$

We may now replace the operators by eigenvalues if they are normal and have the same system of eigenvectors, i.e. if they commute. This is in general not the case but for small Δt we may replace (2.6) by

$$\left(1 - \frac{\Delta t}{2}\frac{\partial D}{\partial W}\right)\delta W^{n+1} = \left(1 + \frac{\Delta t}{2}\frac{\partial D}{\partial W}\right)\delta W^n. \tag{2.7}$$

These operators do commute and are normal under assumption (1.26) so we may replace (2.7) by

$$(2.8) \qquad (1-\Delta t\lambda)\delta W^{n+1} = (1+\Delta t\lambda)\delta W$$

Inequality (1.15) requires that

$$(2.9) \qquad \left|\frac{1+\Delta t\lambda}{1-\Delta t\lambda}\right| \leq 1,$$

which, after some manipulations, gives

$$(2.10) \qquad \lambda + \bar{\lambda} \leq 0.$$

Since $\lambda + \bar{\lambda} = 0$ in our case, the only restriction is that Δt should be so small that (2.7) may be written. In practice, Δt may be chosen relatively large and is often more restricted by accuracy- than by stability requirements.

2.1.3. Algorithm

Due to the form of the matrices A and B, no more than two variables are coupled to each other at a certain time level. This leads to the following steps per timestep:

1 calculate $\hat{W} = \frac{1}{2}(3W^n - W^{n-1})$,
2 calculate $Z = W^n + \frac{\Delta t}{2} F^2(W^n)$,
3 solve for each horizontal row the coupled variables $\{U^*_{jk}, \phi^*_{jk}\}$,
4 solve for each horizontal row $\{V^*_{jk}\}$ except for the boundaries $y = 0, y = D$,
5 calculate $Z^* = 2*W^* - Z$ (i.e. $W^* + \frac{\Delta t}{2} F^1(W^*)$),
6 solve for each vertical row the coupled variables $\{W^{n+1}_{jk}, \phi^{n+1}_{jk}\}$,
7 solve for each vertical row $\{U^{n+1}_{jk}\}$,
8 store W^n into W^{n-1} and W^{n+1} into W^n.

In steps 3 and 6 a block-tridiagonal matrix, each block being (2×2), occurs. The solution procedure of equations with such a coefficient matrix is essentially the same as in case of a scalar tridiagonal matrix, which occurs in steps 4 and 7. Each operation on an element in the scalar case should

be performed in a generalized way on a (2×2) block in the block-tridiagonal case.

On horizontal lines (step 3 and 4) the periodic boundary conditions give rise to cyclic coefficient matrices. This leads to two extra right-hand sides in step 3 and one extra right-hand side in step 4 (see [11]). From the above timestep, we conclude that quite a lot of storage is needed. In fact we need the following arrays of length $(3\times N_x\times N_y)$: V,V^*,W,W^*,W^{n+1}, W^n and W^{n-1}, but from step 1 we see that $\hat{W}$ can be stored into W^{n-1}, and from step 5 that W^{n+1} can be stored in W^*. In actual calculation V^* is calculated per vertical line. This can also be done for V, but then step 5, which is faster than the form between brackets, cannot be performed. For the coefficient matrices, occurring in step 3,4, 6 and 7, at most an array of $9*2*N_x$ length is needed.

Summarising, four arrays of length $3*N_x*N_y$ and one of $9*2*N_x$ length are needed.

2.2. The Runge-Kutta 4(RK4) method

This fourth order method, the most famous of all the Runge Kutta methods, is of great practical importance for the SWEs, because it is an explicit method and has a stability region including a relatively large part of the imaginary axis.

2.2.1. Description [10]

The time integrator assumes the following form

$$W^{n+1} = W^n + \frac{\Delta t}{6}(K_1+2K_2+2K_3+K_4), \tag{2.11}$$

where

$$K_1 = F(W^n),$$

$$K_2 = F(W^n + \frac{\Delta t}{2} K_1),$$

$$K_3 = F(W^n + \frac{\Delta t}{2} K_2),$$

$$K_4 = F(W^n + \Delta t K_3).$$

2.2.2 Stability

The explicitness of the RK4 method makes it possible to use the model equation (1.32) straight forwardly. By substitution of (1.32) into (2.11) we find:

$$W^{n+1} = (1+\Delta t\lambda + \frac{(\Delta t\lambda)^2}{2} + \frac{(\Delta t\lambda)^3}{6} + \frac{(\Delta t\lambda)^4}{24})\, W^n. \tag{2.12}$$

The boundary of the stability region is determined by

$$1 + \Delta t\lambda + \frac{(\Delta t\lambda)^2}{2} + (\frac{(\Delta t\lambda)^3}{6}) + \frac{(\Delta t\lambda)^4}{24} = e^{i\theta} \tag{2.13}$$

A zero on the imaginary axes can be found by substituting $\Delta t\lambda = iy$ into (2.13) which results in two equations

$$y - \frac{1}{6} y^3 = \sin\theta \text{ and} \tag{2.14}$$

$$1 - \frac{y^2}{2} + \frac{y^4}{24} = \cos\theta.$$

Using $\sin^2\theta + \cos^2\theta = 1$ we find

$$y^6(8-y^2) = 0. \tag{2.15}$$

Thus the zero's are

$$y = 0, \quad \theta = 0, 2\pi \tag{2.16}$$

and

$$y = \pm 2\sqrt{2}, \quad \sin\theta = \pm\frac{2}{3}\sqrt{2} \quad \cos\theta = -\frac{1}{3}.$$

Hence, using (1.32), the following restriction on the stepsize Δt is found

$$\Delta t < \frac{2\sqrt{2}}{\sqrt{gh(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2})}} \tag{2.17}$$

2.2.3. Algorithm

In this case the following steps are performed during one timestep:

1 calculate $R = F(W^n)$,

2 $S_{in} = W^n + \frac{\Delta t}{2} R$,

3 calculate $S_{out} = F(S_{in})$,

4 $R = R + 2*S_{out}$,

5 $S_{in} = W^n + \frac{\Delta t}{2} S_{out}$,

6 calculate $S_{out} = F(S_{in})$,

7 $R = R + 2*S_{out}$,

8 $S_{in} = W^n + \Delta t\ S_{out}$,

9 calculate $S_{out} = F(S_{in})$,

10 $W^{n+1} = W^n + \frac{\Delta t}{6} (R+S_{out})$,

11 store W^{n+1} in W^n.

The four F evaluations, in steps 1,3,6 and 9, are far more expensive than the simple addings in the other steps. Thus the number of F evaluations determines the costs of the method. Storage of size $3*N_x*N_y$ is needed for R, S_{in}, S_{out}, W^n, and W^{n+1}. Of course, step 11 is superfluous because W^{n+1} can directly be stored in W^n during step 10, eliminating the need for an array W^{n+1}. Due to the special form of the SWE it is possible to store the result of an F evaluation in S_{in} on the costs of a work space of length $3*N_y$. So three arrays of length $3*N_x*N_y$ plus a work space of length $3*N_y$ are sufficient for this method.

2.3. A stabilized Runge Kutta (RKS) method

2.3.1. Description

In section (1.3) it was shown that the eigenvalues of the right-hand side of the SWEs have imaginary values. Therefore, special Runge Kutta schemes were constructed [7] with a stability region containing an as large as possible part of the imaginary axis. Here a second-order scheme is used defined by

$$W^{n+1} = W^n + \Delta t K_3, \tag{2.18}$$

with

$$K_1 = F(W^n),$$
$$K_2 = F(W^n + \tfrac{1}{2}\Delta t K_1),$$
$$K_3 = F(W^n + \tfrac{1}{2}\Delta t K_2).$$

Thus three F evaluations per time step are needed.

2.3.2. Stability

By applying (2.18) to the model equation (1.32), we find the equation

$$W^{n+1} = (1+\lambda\Delta t+\tfrac{1}{2}(\lambda\Delta t)^2 + \frac{1}{4}(\lambda\Delta t)^3)W^n \tag{2.19}$$

which leads to the following equation for the boundary of the stability region

$$1 + \lambda\Delta t + \tfrac{1}{2}(\lambda\Delta t)^2 + \frac{1}{4}(\lambda\Delta t)^3 = e^{i\theta}. \tag{2.20}$$

In the same way as in section 2.2.2 we find zero's

$$\lambda\Delta t = \pm 2i, \tag{2.21}$$

which are the boundaries of this region on the imaginary axes.
Thus, the time step is restricted by

$$(2.22) \qquad \Delta t = \frac{2}{\sqrt{gh\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}}.$$

2.3.3. Algorithm

During one time step the following steps have to be performed:

1 calculate $S = F(W^n)$,

2 $S = W^n + \Delta t/2 * S$,

3 calculate $S = F(S)$,

4 $S = W^n + \Delta t/2 * S$,

5 calculate $S = F(S)$,

6 $W^n = W^n + \Delta t * S$.

As in the previous section, the storage requirements can be kept small by using S for in- and output for the F evaluation. In this case no array is needed to store the intermediate results, because for updating W^n only the last calculated S(or K_3) is needed. So only two arrays of length $3*N_x*N_y$ plus a work space of length $3*N_y$ are required.

2.4. The Leap-Frog (LF) method

This method was first used by RICHARDSON [12]. HARRIS and JELESNIANSKI [6] applied it to the SWEs. The method is fast because only one F evaluation per timestep is needed, but when dissipation is added in the SWEs the method will become unstable, as will be shown.

2.4.1. Description

The method is defined by the simple formula

$$(2.23) \qquad W^{n+1} = W^{n-1} + 2*\Delta t*F(W^n).$$

It is easy to see that the method is second order consistent in time.

2.4.2. Stability

By applying (2.23) to the model equation we find

$$W^{n+1} = W^{n-1} + 2\lambda\Delta t W^{n}. \tag{2.24}$$

The boundary of the stability region can be found by substituting

$$W^{k+1} = W^{k} e^{i\phi}, \tag{2.25}$$

which leads to the equation

$$\lambda\Delta t = \frac{e^{i\phi} - e^{-i\phi}}{2} = i\sin\phi. \tag{2.26}$$

So the stability region, coinciding with its boundary, is only the interval

$$(-i,i). \tag{2.27}$$

From this result we conclude that the method is useful in our case, but unstable in case where dissipation is added, because then the eigenvalue is certainly in the left half plane. However, when the operator F(W) is split over the various time stages stable methods can be made [9].

2.4.3. Algorithm

One time step is by this method described by:

1 calculate $S = F(W^{n})$

2 $S = W^{n-1} + 2*\Delta t*S$

3 $W^{n-1} = W^{n}$

4 $W^{n} = S$.

Storage is needed for S, W^{n-1} and W^{n}. Thus again a workspace and three

arrays are required.

3. RESULTS

In order to compare the four time integrators we have chosen the test model described by GRAMMELTVEDT [4]. As in shallow water calculations one is often interested in short-time integration, we have restricted the time integration to 48 hours fysically. The solutions of the four time integrators are compared with a very accurate solution calculated by a Runge-Kutta-Fehlberg method [2] with local-error tolerance 10^{-7}. Only solutions with the same mesh width in space are compared with each other. Therefore, also the efficiencies of the methods, determined by accuracy and computational costs, are compared for a fixed mesh width in space. However, we performed this procedure for three different mesh sizes in space. In the following, the test model is described and the efficiency calculations are presented in tables and plots.

3.1. The test problem

The test problem described below was first used by GRAMMELTVEDT [4]. The boundary conditions were already given in (1.6) and (1.7). The initial condition is:

$$(3.1) \qquad h(x,y) = H_0 + H_1 \tanh\left(\frac{9(D/2-y)}{2D}\right) + H_2 \operatorname{sech}^2\left(\frac{9(D/2-y)}{D}\right) \sin\left(\frac{2\pi x}{L}\right).$$

The initial velocity fields are derived from the initial height field via the geostrophic relationship

$$(3.2) \qquad u = \left(\frac{-g}{f}\right) \frac{\partial h}{\partial y}, \quad v = \left(\frac{g}{f}\right) \frac{\partial h}{\partial x}.$$

The constants used are

$$L = 6000 \text{ km}, \qquad g = 10 \text{ msec}^{-2},$$
$$D = 4400 \text{ km}, \qquad H_0 = 2000 \text{ m},$$
$$f = 10^{-4} \text{sec}^{-1}, \qquad H_1 = 220 \text{ m},$$
$$\beta = 1.5\times10^{-11} \text{sec}^{-1}\text{m}^{-1}, \qquad H_2 = 133 \text{ m}.$$

A plot of the height lines of this initial condition is given in figure 3.1 This initial condition describes a wave of length L in the x-idrection. The initial condition is made consistent with the boundary conditions at y = 0 and y = D by setting v to zero at these boundaries. The three different meshsizes are defined by $N_x \times N_y$ is (15×11), (30×22) and (60×44), respectively. The solutions obtained with a Runge-Kutta-Fehlberg method (tolerance 10^{-7}) after two days are displayed in figure 3.2, 3.3 and 3.4. These figures show the height lines of the solution. The dots occurring in fig.3.2 mean that more than one height line can be drawn, which shows that unphysical height lines enter in the solution on this course grid.
From this plots we see that the solution is largely dependent on the space-discretization, which is for our purpose not important because we consider only the error due to the time discretization. These solutions will be used as a reference in the following.

3.2. Efficiency

Using the reference solutions, we are able to calculate errors. In our case, several errors are calculated.
Define $a_i = w_i - \bar{w}_{iref}$, where $\bar{w}_{iref} = \frac{1}{\bar{A}} \int_{\bar{A}} w_{iref} d\bar{A}$, i.e., the mean value of the reference solution on the spatial domain $\bar{A}$, $w_1 = u$, $w_2 = v$, $w_3 = h$.
Define next a relative error by

$$(3.3) \qquad rel_i = \frac{|a_i - a_{iref}|}{\max|a_{iref}|},$$

where a_{iref} is derived from the Runge-Kutta-Fehlberg solution. The $L_{\bar{1}}, L_{\bar{2}}$ and $L_{\bar{\infty}}$ error are now defined by

$$(3.4) \qquad E_1 = \frac{1}{\bar{A}} \int_{\bar{A}} rel_i dA, \; E_2 = \frac{1}{\bar{A}} \int_{\bar{A}} (rel_i)^2 d\bar{A}, \; E_\infty = \max_{\bar{A}} rel_i,$$

respectively.
Efficiency is defined by the number of significant digits, i.e., the $-^{10}\log(E_{1,2,\infty})$, divided by the computational costs to obtain them.

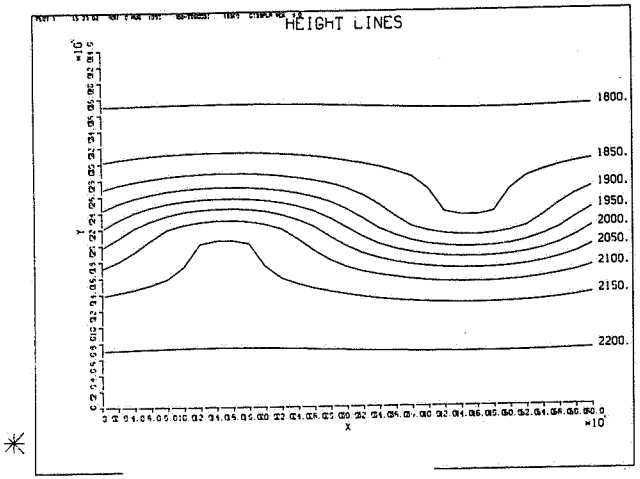


Fig.3.1 initial height lines

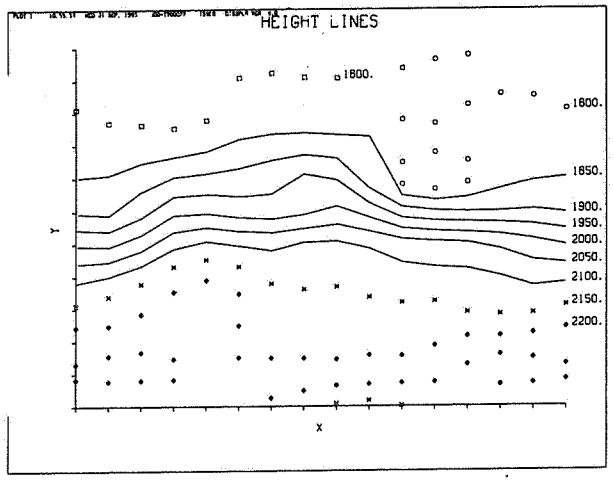


Fig.3.2 height lines on coarse grid after two days

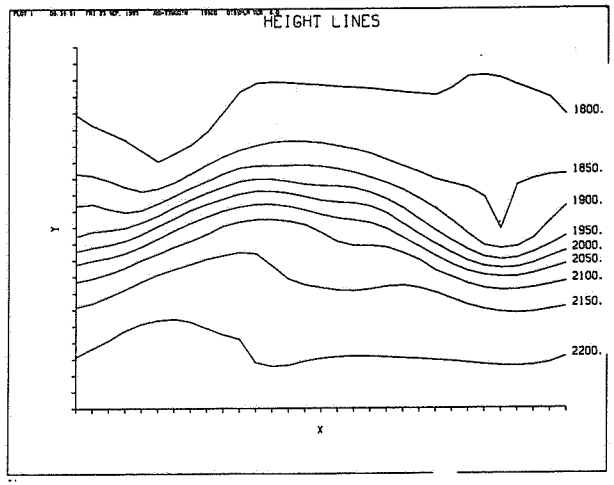


Fig.3.3 height lines on medium grid after two days

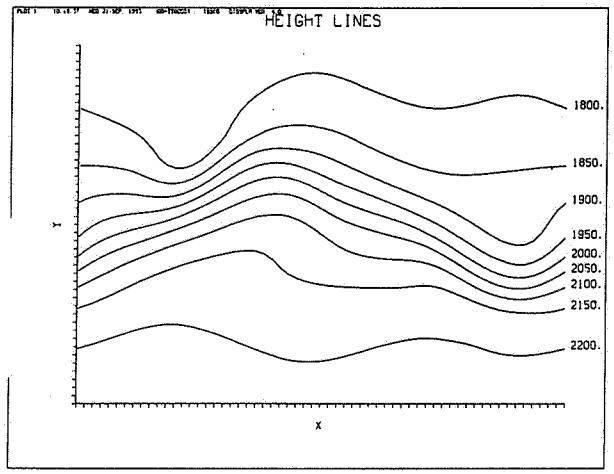


Fig.3.4 height lines on fine grid after two days

In the tables 3.1, 3.2 and 3.3 the significant digits of h are given together with the used time step and the needed computation time (in seconds on a CYBER 170-750). Of course, Δtx(computation time)is almost constant. In the first column, the name of the method and the field length, i.e., the memory requirements expressed in words (octal), is written. The second field length is calculated by hand in case the storage is reduced by the technique described in Appendix A.
The tables show a second-order convergence for the FN, RKS and LF method, whereas the RK^4 method converges fourth order. This can be seen from the fact that halving the time step implies

$$^{10}\log(\text{error}(h/2)) = {}^{10}\log(C(h/2)^q) =$$

$$^{10}\log(Ch^q) + {}^{10}\log((1/2)^q) =$$

$$^{10}\log(\text{error}(h)) - q\ {}^{10}\log 2 \approx \log^{10}\log(\text{error } h)) - qx.30$$

Hence, for second-order schemes the number of significant digits must be incremented by about .60 when the time step is halved, whereas for the RK4 scheme an increment of 1.20 should be observed. For large time steps, however, we find a smaller convergence factor due to the influence of high-order terms in the truncation error.
For the explicit methods the maximum time step in the most right column is the largest possible time step. Larger time steps showed instability. These maximum time steps can be estimated beforehand by relations like (2.17).
In the figures 3.5, 3.6 and 3.7 we show the plots resulting from these tables. In each figure, the three plots refer to the coarse, medium and fine grid, respectively. The first figure shows the result when only the computation time is taken into account. As the computation time increases with a factor four when the grid steps are halved in both directions, we have adapted the scaling of the axis to this. The second figure shows the results when the first field length is taken into account. The costs are calculated by

$$costs = (1117. + 125*FL/1000)*CPS/3600,$$

where FL is the field length and CPS the computation time. This formula approximates the costs as accounted by the SARA computer centre in Amsterdam. It shows that costs decrease when a faster method is used or when storage-use is decreased. The figure shows a great influence of the required storage.

Figure 3.7 makes clear the effect of storage reduction in the way described in appendix A. This technique is only applicable for explicit methods.

From these figures we may conclude that high-order explicit methods perform better than low-order explicit methods. The second question concerning the comparison of explicit methods with implicit methods is more difficult. In this case, it is quite clear that the explicit methods perform better than the implicit ADI method. This may be the result of the chosen test problem. To be more precise. The measured global error is built up of local erros. This local truncation error consists of derivatives of F. For problems where these derivatives are small, larger time steps are allowed for accuracy reasons. In this case, the derivatives are quite large as can be seen from the initial condition, therefore, we cannot make use of the ability of implicit methods to use large time steps and as the truncation error of such a method is not better than that of an explicit method, explicit methods become more efficient because less work per time step has to be done. In practical situations the ability of large possible time steps is an advantage, because the gradients are usual not as large as in the used test problem.

Note that only the error due to the time discretization is calculated. When also the error due to the space discretization is taken into account, errors may match. As an example we quote again the first-order upwind-difference scheme. We expect the same effect for the Leap-Frog method, because also here time and space discretization are treated in the same way, i.e. central differences are used.

	Comp. time	Errors L_1	L_2	L_∞	Δt
FN	1.41	1.21	1.09	.49	3600
52700	4.81	1.76	1.65	1.16	900
	18.4	2.95	2.85	2.34	225
	53.2	3.91	3.80	3.30	75
RK4	1.05	1.42	1.30	.81	4800
43400	3.43	3.00	2.89	2.34	1200
42000	13.1	5.39	5.28	4.79	300
	50.4	7.04	6.93	6.40	75
RKS	1.05	1.31	1.21	.63	3600
42500	3.40	1.70	1.60	1.12	900
42000	12.9	2.81	2.70	2.22	225
	38.3	3.75	3.65	3.16	75
LF	.886	.77	.66	.18	1800
43600	.881	1.17	1.05	.53	1728
42600	.926	1.27	1.18	.77	1600
	.985	1.15	1.05	.59	1440
	.995	1.13	1.02	.56	1350
	2.51	1.88	1.77	1.29	450
	14.0	3.45	3.34	2.86	75
	39.7	4.41	4.30	3.81	25

Table 3.1. Signifiant digits of h on the coarse grid

	Comp. time	Errors L_1	L_2	L_∞	Δt
FN	2.68	1.16	1.00	.37	7200
64000	4.70	1.38	1.24	.54	3600
	9.00	1.59	1.49	1.01	1800
	17.6	1.73	1.64	1.14	900
	34.8	2.20	2.09	1.57	450
RK4	6.36	1.98	1.88	1.36	2400
54300	12.2	2.50	2.39	1.81	1200
45200	23.8	3.63	3.52	2.95	600
	47.3	4.83	4.72	4.18	300
RKS	6.54	1.77	1.69	1.24	1728
50500	8.24	1.82	1.73	1.24	1350
45200	12.1	1.92	1.82	1.27	900
	23.6	2.20	2.09	1.50	450
	46.6	2.73	2.62	2.07	225
LF	5.46	1.72	1.63	1.17	800
54000	6.11	1.82	1.71	1.09	675
50000	8.79	1.86	1.76	1.18	450
	17.3	2.40	2.30	1.83	225
	50.9	3.36	3.26	2.80	75

Table 3.2. Significant digits of h on the medium grid

	Comp. time	Errors L_1	L_2	L_∞	Δt
FN	18.2	1.33	1.16	.54	3600
153300	35.1	1.60	1.44	.84	1800
	68.8	1.93	1.82	1.20	900
RK4	47.7	2.73	2.64	2.14	1200
127200	93.9	3.59	3.49	2.90	600
71100	186.	4.75	4.65	4.09	300
RKS	56.1	2.20	2.12	1.66	768
106700	71.2	2.29	2.20	1.70	600
70000	93.9	2.43	2.34	1.83	450
	186.	2.89	2.79	2.22	225
LF	40.6	2.17	2.08	1.54	400
127200	44.6	2.21	2.13	1.67	360
107200	53.1	2.32	2.24	1.81	300
	70.9	2.54	2.46	1.94	225
	210.	3.49	3.40	2.82	75

Table 3.3. Significant digits of h on the fine grid.

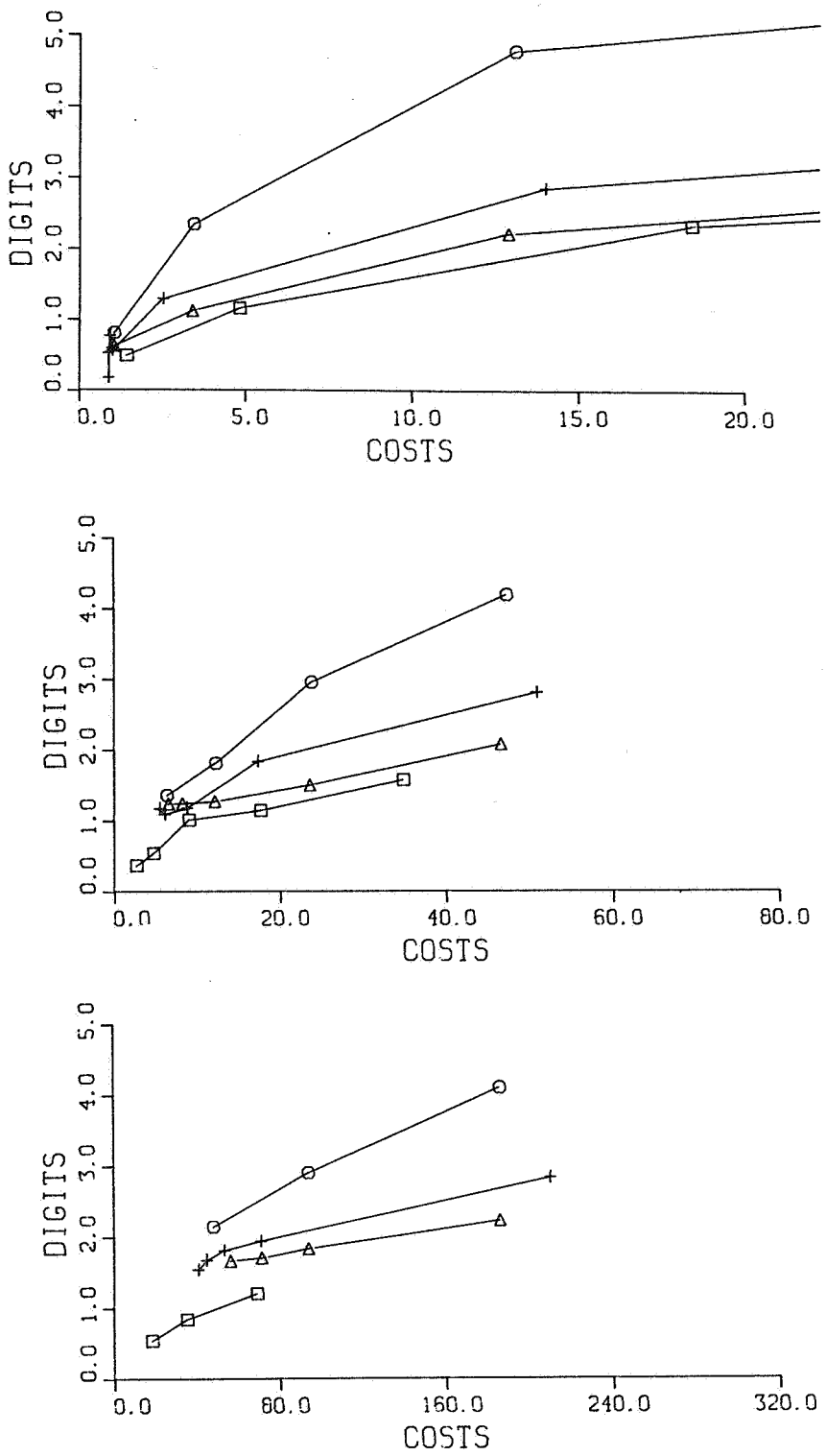


Fig. 3.5. Costs measured in computation time

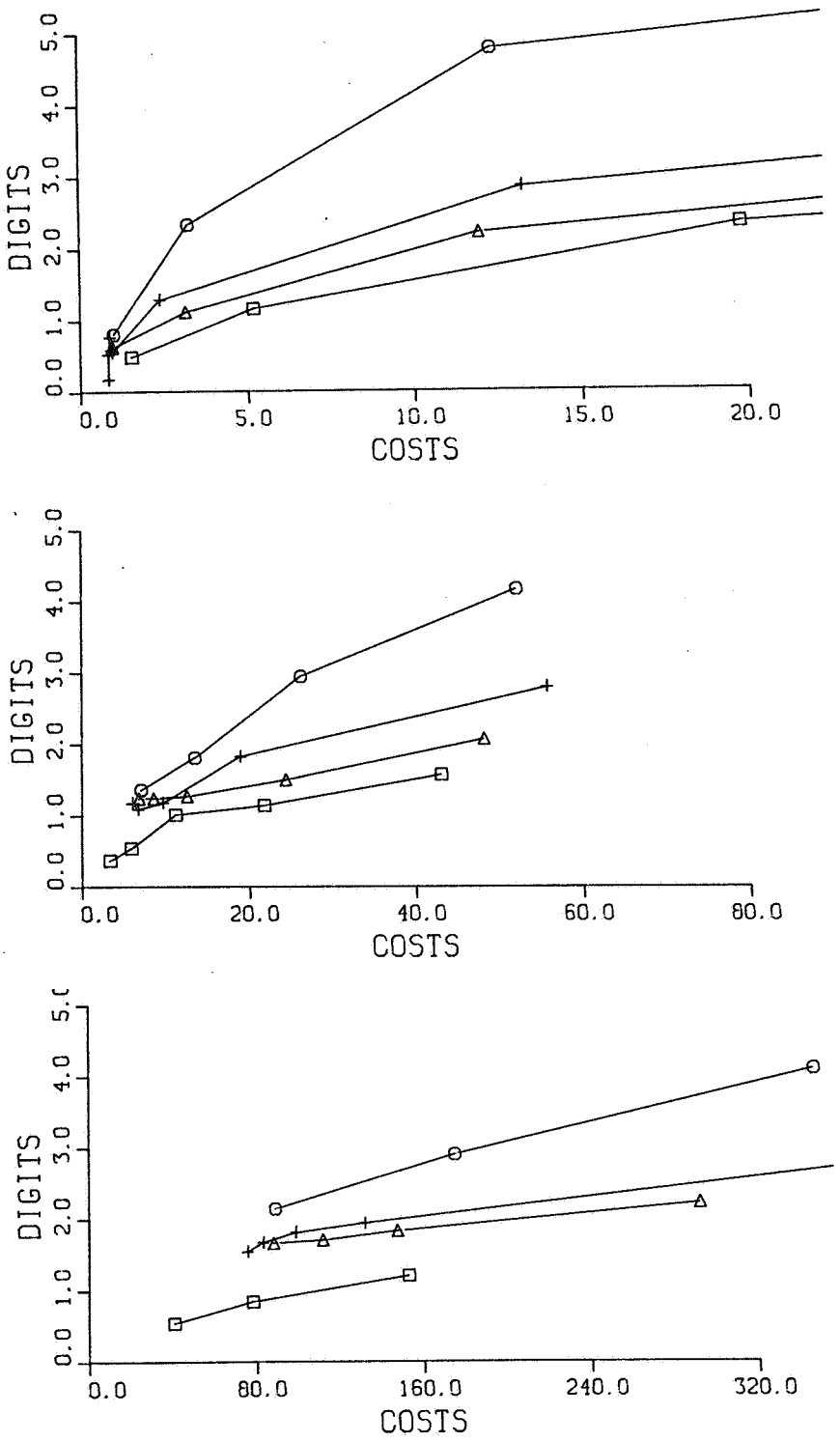
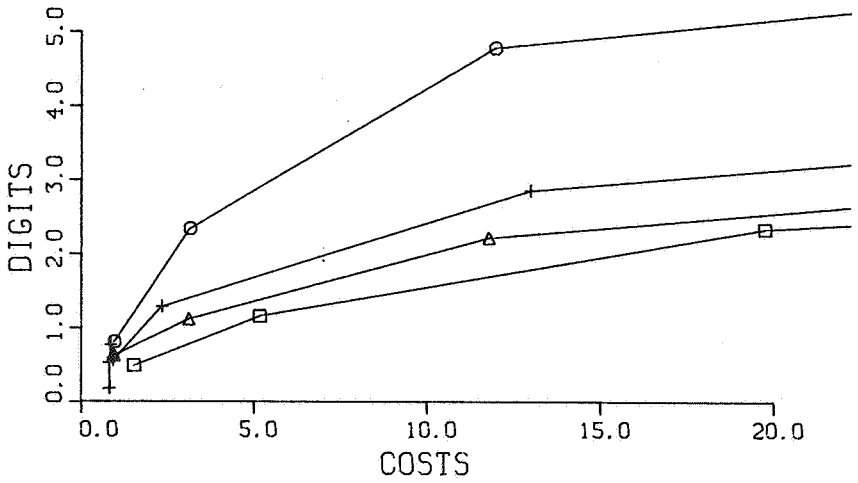
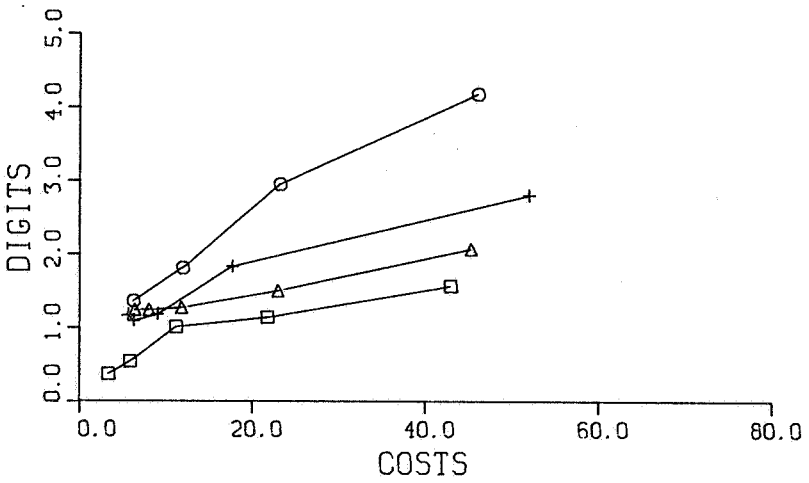
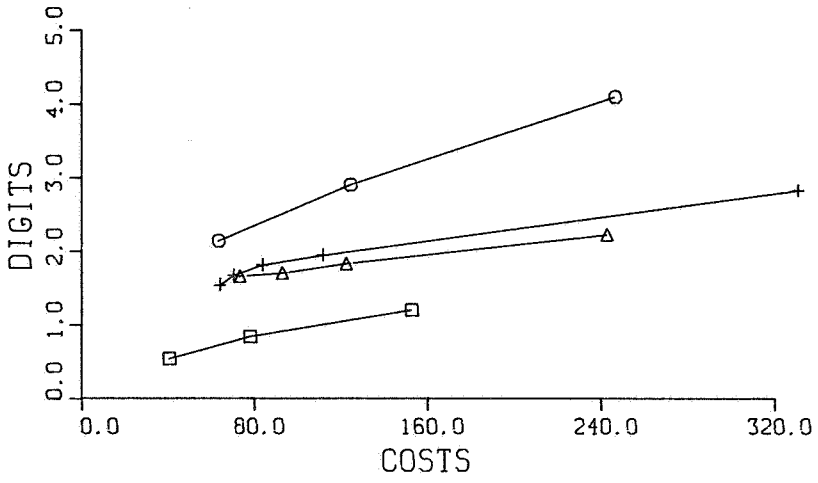


Fig. 3.6. Costs determined by storage plus computation time







3.7. Costs determined by reduced storage plus computation time

4. EVALUATION

In this report we have examined some well known time integrators for the SWEs. The purpose was to compare the efficiency of low-order methods with that of high-order methods and the efficiency of implicit methods versus explicit methods.

Considering the first question we found that the high-order Runge Kutta 4 method is more efficient than low order methods like Leap-Frog and the stabilized Runge-Kutta method. This is due to the fact that the extra F evaluations are compensated by a large stability region and fourth-order accuracy. The fourth-order accuracy is essential to remain accurate with respect to second-order methods, when the largest possible timestep is used.

The second question is more difficult, due to the limited stability region of explicit methods. When for accuracy reasons, the timestep is inside the stability region, the Runge-Kutta four method is more efficient. In the other case, we have no choise, then the implicit method is more efficient. This is especially the case if the solutions are almost stationary. Conclusions should be taken with care, because in these experiments we have not taken into account the influence of the error due to space discretization, which may partly offset the shown error.

REFERENCES

[1] FAIRWEATHER, G. & I.M. NAVON, *A linear ADI method for the shallow water equations*, J. of Comp. Physics 37, 1 - 18, 1980.

[2] FEHLBERG, E., *Low order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems*, NASA Techn. Rep. TRR-315, George C. Marshall Space Flight Center, Marshall, Ala, 1969.

[3] FORSYTHE, G.E. & W. WARSOW, *Finite-difference methods for partial differential equations*, John Wiley & Sons, London, New York, 1960.

[4] GRAMMELTVEDT, A., *A survey of finite-difference schemes for the primitive equations for a barotropic fluid*, Mon. Weather Rev. 97, 384-404, 1969.

[5] GUSTAFSSON, B.G., *An alternating direction implicit method for solving the shallow water equations*, J. of Comp. Physics 7, 239-254, 1971.

[6] HARRIS, D.I. & C.P. JELESNIANSKI, *Some problems in the numerical solution of tydal hydraulics equations*, Monthly Weather Review, 92, 1964.

[7] HOUWEN, P.J. VAN DER, *Stabilized Runge-Kutta methods with limited storage requirements*, Report TW 124, Mathematical Centre, Amsterdam, 1971.

[8] IMSL, Inc., *Twodepep*, Houston.

[9] KREISS, H.O., B. GUSTAFSSON and A. SUNDSFRÖM, *Stability theory of difference approximations for mixed initial boundary value problems*, II, Math. of Comp., V.26, 1972, pp. 649-686.

[10] LAMBERT, J.D., *Computational methods in ordinary differential equations*, Wiley, London-New York, 1973.

[11] NAVON, I.M., *Algorithms for solving scalar and block-cyclic tridiagonal systems of linear algebraic equations*, WISK 265, 1977.

[12] RICHARDSON, L.F., *Weather prediction by numerical process*, p.151, Cambridge University Press, London, 1922. Reprinted 1965 by Dover Publications, Inc., New York.

[13] ROACHE, P.J., *Computational fluid dynamics*, Hermosa Publishers, Albuquerque, N.M., 1972.

[14] VERWER, J.G. & K. DEKKER, *Stability in the methods of lines*, in Proceedings of the seminar "Numerische Behandlung von Differential gleichungen", ed. K. Strehmel, Martin Luther Universität Halle, to appear.

[15] VERWER, J.G. & K. DEKKER, *Step-by-step stability in the numerical solution of partial differential equations*, Report NW 161/83, Mathematical Centre, Amsterdam, 1983.

APPENDIX A

ON STORAGE REDUCTION FOR EXPLICIT METHODS

In explicit methods, like RK methods, often a number of F evaluations have to be performed. This is normally done one after another over the whole field, but this assumes that a certain F_{jk} is dependent on all components of W, which is not the case for discretizations of PDEs. In fact F_{jk} is only dependent on a few components of W in the neighbourhood of W_{jk}. Therefore, we propose to perform all stages at once, which avoids the need for arrays of the size of the whole field. Only a limited number, depending on the number of stages and the discretization, of line arrays are needed. We will show our idea by means of the RK4 method of section 2.2. In that case F_{jk} is dependent on five points, i.e. $W_{j-1,k}$, W_{jk}, $W_{j+1,k}$, $W_{j,k-1}$, and $W_{j,k+1}$. Looking in the x,t-plane we may picture the dependence domain of W_{jk}^{n+1} in one timestep by

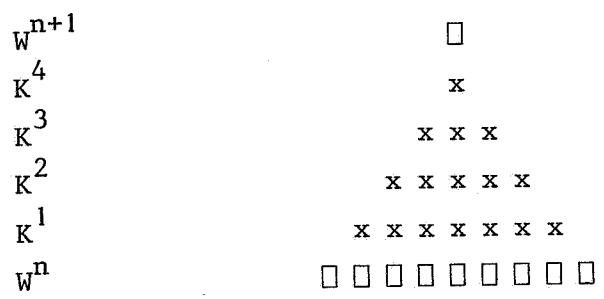


Fig. (A.1).

Likewise in the y,t plane.
Coming from a boundary, say x = 0, we depict the necessary K_{jk} (K denoted by an x) by

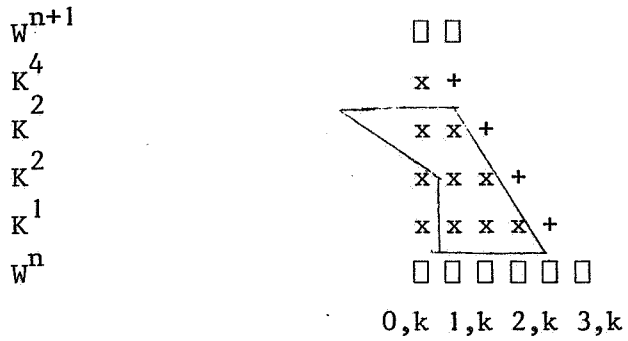


Fig. (A.2)

The next W_{1k}^{n+1} can be calculated by the K's denoted by a + and the K's denoted by an x inside the drawn lines plus the same set at y-index k-1 and k+1. Thus we have to store the set of K's inside the drawn lines over a whole length in y direction. This means seven arrays of length Ny*3 (W_{jk} consists of three elements). The K's denoted by a + have to be stored also, but the drawn lines can be shifted to the right when the calculation is at point 1,k+1 and the K's at the + places are calculated. Thus, three dummy variables are enough.

This last step, however, makes the algorithm recursive, which must be avoided in case where vectorcomputers are used. By adding three arrays of length Ny*3 the algorithm can be reordered in such a way that vector-operations are possible.