INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 6, 641-657 (1986)

STABILIZATION OF EXPLICIT METHODS FOR HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS

F. W. WUBS

Centre for mathematics and Computer Science, Amsterdam

SUMMARY

It is well known that explicit methods are subject to a restriction on the time step. This restriction is a drawback if the variation of the solution in time is so small that accuracy considerations would allow a larger time step. In this case, implicit methods are more appropriate because they do allow large time steps. However, in general, they require more storage and are more difficult to implement than explicit methods. In this paper we propose a technique by which it is possible to stabilize explicit methods for quasi-linear hyperbolic equations. The stabilization turns out to be so effective that explicit methods become a good alternative to unconditionally stable implicit methods.

1. INTRODUCTION

In numerical analysis, we distinguish explicit and implicit time integrators for partial differential equations. It is well known that explicit methods are subject to a restriction on the time step. This restriction is a drawback if the variation in time is so small that accuracy considerations would allow a larger time step. In this case, implicit methods are more appropriate because they do allow large time steps. However, in general, they require more storage and are more difficult to implement than explicit methods. In this paper, we propose a technique by which it is possible to stabilize explicit methods for quasi-linear hyperbolic equations. The stabilization turns out to be so effective that explicit methods become a good alternative to unconditionally stable implicit methods. More precisely, the stabilized explicit methods are competitive with conventional implicit methods with respect to both accuracy and computational costs. In fact, we will show, for some examples, that the technique also inherently appears in implicit methods, which explains the improved stability behaviour of implicit methods. In the fifties, explicit methods were quite popular because of their simplicity. Thereby, they were well suited for hand calculations and small computers. With the coming of more powerful computers in the sixties, having also a larger memory, implicit methods became popular. In the seventies, when the vector computers were introduced, the explicit methods became in scope again, because they allow a high degree of vectorization. Therefore, the stabilization technique given here may be of interest for the efficient use of explicit methods in a large variety of problems. In fact, our attention was focused on explicit methods when we started to construct a shallow-water equation solver for use on the vector computer CYBER 205.

In this paper, we restrict ourselves to hyperbolic problems; however the theory develops in a similar way for parabolic problems. In Section 2 the theory is presented and in Section 3 some numerical illustrations will be given.

Received 2 May 1985
Revised 11 February 1986

2. THEORY

Consider the equation

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t), \quad \mathbf{x} \in R^n, \quad t > 0 \tag{1}$$

where $\mathbf{u} = (u_1, (\mathbf{x}, t), u_2(\mathbf{x}, t), \ldots, u_N(\mathbf{x}, t))^{\mathrm{T}}$, defining a first-order quasi-linear hyperbolic system of N equations.[1] Using explicit methods for (1), the time step is restricted by the Courant–Friedrichs–Lewy (C.F.L.) condition (see (25)). In many problems, this time step restriction is much more severe than the one following from accuracy considerations. For instance, in order to represent an irregular geometry, a fine space mesh is needed. At the same time the variation of the solution in time may be very slow. In that case, one likes to use much larger time steps than the one allowed by the C.F.L. condition. In the following sections, we will show that it is possible to stabilize an explicit method by an appropriate smoothing of the right-hand side. Smoothing of the right-hand side will not give rise to larger errors when $\mathbf{u}_t$ has small space derivatives. We will show for some examples that small time derivatives of $\mathbf{u}$ imply, under certain conditions, small space derivatives of $\mathbf{u}_t$. Moreover, this property is trivial for the limiting stationary case. We emphasize that $\mathbf{u}$ itself may have large space derivatives. For example, we may think of solutions which are close to a steady state and which can be written in the form

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_0(\mathbf{x}) + \mathbf{u}_1(\mathbf{x}, t) \tag{2}$$

where $\mathbf{u}_0(\mathbf{x})$, the stationary solution, has large space derivatives and $\mathbf{u}_1(\mathbf{x}, t)$ is a smooth function in both variables $\mathbf{x}$ and t.

For the stabilization of explicit methods, smoothing is often used before, but then usually the grid function $\mathbf{u}$ is smoothed,[2,3] rather than the right-hand side. This smoothing of $\mathbf{u}$ may only be applied, without danger of loss of accuracy, if $\mathbf{u}$ itself is smooth, i.e. if $\mathbf{u}$ has small derivatives with respect to the space variables, which, in general, is not true. As an example, the famous variant of the Lax–Wendroff scheme proposed by Richtmyer and Morton[4] may be regarded as a two-stage second-order Runge–Kutta method,[5] where, in the first stage, the solution $\mathbf{u}$ is smoothed, in order to obtain a stable method.

In the field of the boundary-value problems the stabilization technique is known by the name residual averaging.[6] In this case, explicit time stepping is used to solve a boundary-value problem. The explicit method is then stabilized by using an implicit smoothing operator (see Section 2.4) in order to accelerate the convergence. Our contribution will be the construction of explicit smoothing operators which are less expensive than the implicit smoothing operators, especially if we want to use a vector computer.

2.1. The smoothness of the right-hand side

The assumption of a smooth right-hand side (or, equivalently, of $\mathbf{u}_t$) is important for the error introduced by smoothing. In hyperbolic equations we may expect in some cases that there is a relation between the variation of the solution in time and in space. For example, we may think of wave-like phenomena moving with some characteristic speed over the field. We will show for two examples that such a relation exists.

Example 1. Consider the one-dimensional system of equations

$$\mathbf{u}_t = \mathbf{A}\frac{\partial}{\partial x}\mathbf{u} + \mathbf{B}\mathbf{u} + \mathbf{g}(x), x \in R \tag{3}$$

where $\mathbf{u} = (u_1(x,t), u_2(x,t), \ldots, u_N(x,t))^{\mathrm{T}}$, $\mathbf{A}$ is a non-singular, constant $N \times N$ matrix, $\mathbf{B}$ a constant $N \times N$ matrix and $\mathbf{g}$ an arbitrary continuous function of x. Differentiating (3) with respect to time yields

$$(\mathbf{u}_t)_t = \mathbf{A}\frac{\partial}{\partial x}\mathbf{u}_t + \mathbf{B}\mathbf{u}_t \tag{4}$$

Using (4) it follows that if all time derivatives of $\mathbf{u}$ up to order n are small, then the $(n$-1)th space derivative of $\mathbf{u}_t$ is small. Hence, the right-hand side in (3) has small space derivatives if $\mathbf{u}$ has small time derivatives.

Example 2. As a second example we consider the linearized shallow-water equations given by

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}_x, \mathbf{u}_y) = -\left(\mathbf{A}\frac{\partial}{\partial x} + \mathbf{B}\frac{\partial}{\partial y}\right)\mathbf{u} + \mathbf{h}(x,y) \tag{5}$$

where $\mathbf{u} = (u, v, \zeta)^{\mathrm{T}}$, $\mathbf{h}(x,y)$ is an arbitrary function of x and y, and

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & g \\ 0 & 0 & 0 \\ H & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & g \\ 0 & H & 0 \end{bmatrix}, \quad g, H > 0$$

Differentiation of (5) with respect to t gives

$$(\mathbf{u}_t)_t = -\left(\mathbf{A}\frac{\partial}{\partial x} + \mathbf{B}\frac{\partial}{\partial y}\right)\mathbf{u}_t \tag{6}$$

Hence $\mathbf{u}_t$ is also a solution of the linearized shallow-water equations, in this case, without the forcing term $\mathbf{h}(x,y)$. Let us define by $\xi(x,y), \eta(x,y)$ a normalized orthogonal co-ordinate system. After transformation (in terms of these co-ordinates) (6) becomes

$$(\mathbf{u}_t)_t = -\left[\mathbf{A}\left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right) + \mathbf{B}\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right)\right]\mathbf{u}_t \tag{7}$$

Now we assume that the partial derivatives with respect to η are negligible and that all the time derivatives of $\mathbf{u}$ are small. Furthermore, we assume that

$$\frac{\partial \eta}{\partial x}\frac{\partial u_t}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial v_t}{\partial \xi} \tag{8a}$$

is small. This condition says that the vector $((u_t)_\xi, (v_t)_\xi)^{\mathrm{T}}$ is small in the η-direction, given by the vector $(\eta_x, \eta_y)^{\mathrm{T}}$. Using the special structure of $\mathbf{A}$ and $\mathbf{B}$, it follows from the first and second equations in (6) that $(\zeta_t)_x$ and $(\zeta_t)_y$ are small. Furthermore, from the third equation in (7), it follows that

$$\frac{\partial \xi}{\partial x}\frac{\partial u_t}{\partial \xi} + \frac{\partial \xi}{\partial y}\frac{\partial v_t}{\partial \xi} \tag{8b}$$

is small. Combining (8a) and (8b), we have that $(u_t)_\xi$ and $(v_t)_\xi$ are small. As the derivatives with respect to η are negligible, it follows that $(u_t)_x, (u_t)_y$ and $(v_t)_x, (v_t)_y$ are small. Hence, all first-order space derivatives of $\mathbf{u}_t$ are small. Proceeding in the same way, under similar assumptions, it is possible to show that all higher-order space derivatives of $\mathbf{u}_t$ are small.

These two examples show that, under certain assumptions, we may expect that the right-hand side is smooth in space if the time derivatives of the solution are small. The property of a smooth right-hand side in space can be used effectively to stabilize an explicit time integration method by smoothing the discretized form of $\mathbf{f}(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t)$, which is obtained by the method of lines. In this approach, the space discretization gives rise to a system of ordinary differential equations[5]

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{U} = \mathbf{F}(\mathbf{U}, t), \tag{9}$$

where U is a grid function approximating $\mathbf{u}$, and $\mathbf{F}(., t)$ a vector function approximating $\mathbf{f}(., \mathbf{x}, t)$. Thereafter, an appropriate time integrator is used to solve this equation. Instead of (9), we propose to solve

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{U} = S\mathbf{F}(\mathbf{U}, t), \tag{10}$$

where S is a smoothing operator, with the property $S \to I$, the identity operator, when the mesh size tends to zero.

In fact, many unconditionally stable time integrators, applied to (9), can be written as a conditionally stable (explicit) integrator applied to (10). We will illustrate this for Euler's backward method applied to

$$u_t = f(u_x, x) \tag{11}$$
$$f(u_x, x) = u_x + g(x)$$

where $g(x)$ is an arbitrary function of x. The right-hand side in (11) is discretized, on a grid with mesh size h, with the usual second-order central differences

$$F_j(\mathbf{U}) = (D\mathbf{U})_j + g(x_j),\ x_j = jh \tag{12}$$

where

$$(D\mathbf{U})_j = (U_{j+1} - U_{j-1})/(2h) \tag{13}$$

and U_j approximates $u(x_j)$. When backward Euler is applied to (9), with F given by (12), we find

$$U_j^{n+1} - \Delta t(D\mathbf{U})_j^{n+1} = U_j^n + \Delta t\, g(x_j) \tag{14}$$

where $\mathbf{U}^n$ approximates the exact solution $\mathbf{U}(t)$ of (9) at $t^n = n\Delta t$. This can be rewritten as

$$U_j^{n+1} - \Delta t(D\mathbf{U})_j^{n+1} = U_j^n - \Delta t(D\mathbf{U})_j^n + \Delta t F_j(\mathbf{U}^n) \tag{15}$$

As the operator $(I - \Delta t D)$ is invertible, we find

$$U_j^{n+1} = U_j^n + \Delta t\{(I - \Delta t D)^{-1}\mathbf{F}(\mathbf{U}^n)\}_j \tag{16}$$

which is simply forward Euler applied to (10) with the smoothing operator $S = (I - \Delta t D)^{-1}$. A discussion of this smoothing operator and another example can be found in the Appendix. Here, we mention that the time step appears in the smoothing operator. Because the magnitude of the time step determines the amount of smoothing needed to obtain a stable method, it will also appear in our smoothing operators. Moreover, the time step in the smoothing operator ensures the consistency of (10) with (9).

In the remainder of this section, we will illustrate the theory by the scalar equation (11) and its semi-discretization (12). We are aware of the fact that (11) is simple, but it gives relevant information for less trivial cases (see Section 3.3).

2.2. *Stability*

In order to solve the initial value problem (9) we consider explicit m-point single step Runge–

Kutta formulae, i.e. formulae of the type

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \sum_{i=0}^{m-1} \theta_i \mathbf{K}_i \tag{17}$$

$$\mathbf{K}_i = \Delta t \mathbf{F}\left(\mathbf{U}^n + \sum_{l=0}^{i-1} \alpha_{i,l} \mathbf{K}_l, t^n + \mu_i \Delta t \right)$$

$$\mu_0 = 0$$

When (17) is applied to the scalar equation

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda u \tag{18}$$

we obtain

$$u^{n+1} = P_m(\Delta t \lambda) u^n$$

where $P_m(z)$ is a polynomial of the form

$$P_m(z) = \beta_0 + \beta_1 z + \cdots + \beta_m z^m \tag{19}$$

of which the coefficients β_j can be expressed in terms of the Runge–Kutta parameters. The polynomial $P_m(z)$ is the so-called stability polynomial associated with formula (17). The polynomial is compatible with a Runge–Kutta formula of order p, provided that

$$\beta_j = \frac{1}{j!}, \quad j = 0, 1, \ldots, p \tag{20}$$

Furthermore, the region defined by

$$S = \{z \mid \ |P_m(z)| < 1\} \tag{21}$$

will be called the stability region of the Runge–Kutta formula and $P_m(\Delta\lambda)$ is called the amplification factor.[4] In this paper, scheme (17) is said to be stable when the set of points $\Delta t\lambda$, where λ is an eigenvalue of the Jacobian matrix of (9), belongs to the stability region S.

For example, if we apply (17) to (9) with the right-hand side (12) and if we assume periodic boundary conditions, then the eigenfunctions of the Jacobian matrix are Fourier components,

$$V_j = \exp(\mathrm{i}bjh), \quad b \in R \tag{22}$$

and the associated eigenvalues are

$$\lambda = \mathrm{i}\frac{\sin(bh)}{h} \tag{23}$$

Thereby, the amplification factor becomes $P_m(\Delta t \mathrm{i} \sin(bh)/h)$. Furthermore, the method is von Neumann stable if the absolute value of this amplification factor is smaller than one.[4]

The largest constant C, such that $|P_m(\mathrm{i}y)| < 1$ for $y \in [-C, C]$ and $y \in R$, is called the imaginary stability boundary. For the classical Runge–Kutta method $C = 2\sqrt{2}$.[5] If, in general, an explicit method has an imaginary stability boundary C, then when this method is applied to $\{(9), (12)\}$ we have the von Neumann stability condition

$$\Delta t < Ch \tag{24}$$

When smoothing is applied it is of interest to compare the stability condition with the C.F.L.

condition, because the stability condition can never exceed the C.F.L. condition. The C.F.L. condition says that the convex hull of the domain of dependence of the exact solution at a point **x** at time t_1 must be contained in the convex hull of the domain of dependence of the approximating solution at the same point in space and time.[7]

Lemma 1. Let (11) be discretized with central differences involving l points to the left and to the right. If an explicit Runge–Kutta method with stability polynomial $P_m(z)$ is used to solve the resulting system of ODEs then we have the C.F.L. condition

$$\Delta t \leqslant mlh \tag{25}$$

Proof

It is straightforwardly proved that (25) is the C.F.L. condition for the Runge–Kutta method.

2.3. Explicit smoothing operators

2.3.1. Derivation. Consider the smoothing operator S defined by

$$(S_1\mathbf{F})_j := (F_{j+1} + F_{j-1})/2 \tag{26}$$

In order to determine the maximum allowed time step, we now need the eigenvalues of $S_1 D$. These are simply the products of the eigenvalues of S_1 and D, because S_1 and D have the same eigenfunctions (22). The eigenvalues of S_1 are

$$\lambda_{S_1} = \cos(bh) \tag{27}$$

and the products of the eigenvalues of S_1 and D

$$\lambda_{S_1D} = \cos(bh)\mathrm{i}\sin(bh)/h = \mathrm{i}\sin(2bh)/(2h) \tag{28}$$

Hence, compared with (23) the maximum eigenvalues have been reduced by a factor two. However, this may still be very restrictive. Therefore, we repeat the smoothing. Defining a second smoothing operator by

$$(S_2\mathbf{F})_j := (F_{j+2} + F_{j-2})/2 \tag{29}$$

we have, along the same line, that again a factor two is won. In general, we apply the smoothing operator

$$S := \prod_{k=1}^{n} S_k \tag{30}$$

where

$$(S_k\mathbf{F})_j := (F_{j+2^{k-1}} + F_{j-2^{k-1}})/2 \tag{31}$$

The maximum eigenvalue is now reduced by a factor 2^n. This means that the time step can be increased exponentially, whereas the costs grow linearly. Hence, as 2^n time steps are more expensive than one time step with n smoothings, smoothing makes the method much more efficient.

The reader may have noticed that in the case $g \equiv 0$ the smoothing degenerates to a discretization on a coarser grid. This appears quite natural, because of the following reasoning. The solution is of the form

$$u(x,t) = r(x+t) \tag{32}$$

where r is a function depending on the initial and boundary conditions. If in this case the time derivatives are small, then also the space derivatives are small. Hence, if for accuracy reasons the time step may be increased then also the mesh size may be increased. If, however, g is non-zero the discretization differs essentially from the one on a coarser grid. For example, a function $g(jh) = (-1)^j$ cannot be approximated on a coarser grid.

We now will define the smoothing operator more generally by

$$S := \prod_{k=k_0}^{n} S_k \tag{33}$$

where

$$(S_k\mathbf{F})_j := \mu_k F_{j+2^{k-1}} + (1-2\mu_k)F_j + \mu_k F_{j-2^{k-1}} \tag{34}$$

Notice that the smoothing operator in (34) appears to be an identity operator plus a discretized form of a diffusion operator. For $\mu_k = \frac{1}{2}$ for all k and $k_0 = 1$ we have again (30). Another, special smoothing operator following from (33) is the case where $\mu_k = \frac{1}{4}$ for all k and $k_0 = 2$. The eigenvalue of (34) for this value of μ_k is

$$\lambda_{S_k} = \cos^2(2^{k-2}bh) \tag{35}$$

Now, when (33) is applied to (12), again the corresponding eigenvalues may be multiplied and we find

$$\lambda_{SD} = \mathrm{i}\prod_{k=2}^{n} \cos(2^{k-2}bh)\sin(2^{n-1}bh)/(2^{n-1}h) \tag{36}$$

In order to approximate the modulus of the maximum eigenvalue we need the inequality

$$|\cos(x)\sin(2x)| = |2(1-\sin^2(x))\sin(x)| \leqslant \tfrac{4}{9}\sqrt{3} \tag{37}$$

Isolating $\cos(2^{n-2}bh)$ from the product sequence (36) and combining it with $\sin(2^{n-1}bh)$, we can apply inequality (37) to find an upper bound for the maximum modulus of (36). This gives

$$|\lambda_{SD}| < \tfrac{4}{9}\sqrt{3}/(2^{n-1}h) \approx 0{\cdot}77/(2^{n-1}h) \tag{38}$$

In Reference 8 it is shown that (30), (31) defines an optimal smoothing operator for the considered equation. However, for initial-boundary value problems we found that this operator gave unstable results, whereas the operator (34) with $k_0 = 2$ and μ_k slightly smaller than $\frac{1}{4}$ gave stable results (see Section 3.3). This is possibly due to the fact that in the latter case (33) has positive eigenvalues.

With respect to the C.F.L. condition (25) we remark that an application of (33) gives rise to a differencing in which $l = 1 + \sum_{k=k_0}^{n} 2^{k-1} = 2^n - 2^{k_0-1} + 1$. In the case without smoothing $l = 1$. Application of Lemma 1 in both cases shows that with the resulting method after smoothing an increase of the time step with a factor $2^n - 2^{k_0-1} + 1$ is possible. This factor is obtained for $k_0 = 1$ and almost obtained for $k_0 = 2$, as can be seen from the reduction of the magnitude of the spectral radius of the Jacobian matrix.

2.3.2. *The smoothing error.*

Here we will give an approximation of the error due to the smoothing operation (33), for μ_k independent of k. This is achieved by comparing the smoothed and non-smoothed right-hand sides. We will see that the smoothness of the original right-hand side and the time step determine the magnitude of the error. If, in the following, the subscript h is used in connection with a continuous function, then this denotes the restriction of that function to the grid.

Lemma 2. Let $\mathbf{A}(\xi_h)$ be a discretization of $a(\xi(x), \xi_x(x), x)$. If $\mathbf{A}(\xi_h)$ and ξ_h satisfy the condition

$$A_j(\xi_h) = a(\xi(x_j), \xi_x(x_j), x_j) + C_j h^2 + O(h^4) \tag{39}$$

$$C_{j\pm 1} = C_j \pm D_j h + O(h^2) \tag{40}$$

and, moreover, $a(\xi(x), \xi_x(x), x) \in C^4$, then the error due to the smoothing operator (33) is, with $\mu_k = \mu$,

$$(SA(\xi_h))_j - A_j(\xi_h) = \mu h^2 \frac{2^{2n} - 2^{2k_0-2}}{3} \frac{\partial^2}{\partial x^2} a(\xi(x_j), \xi_x(x_j), x_j) + O(h^4) \tag{41}$$

Proof

Let $\phi(x) = a(\xi(x), \xi_x(x), x)$. Using Taylor expansions, we find by substitution of $\phi(x)$ into (34)

$$(S_k \phi_h)_j = \left(1 + \mu(2^{k-1}h)^2 \frac{\partial^2}{\partial x^2}\right)\phi(x_j) + O(h^4) \tag{42}$$

Hence, we have the following error due to the smoothing (33) for $\phi(x_j)$:

$$\begin{aligned}(S\phi_h)_j - \phi(x_j) &= \prod_{k=k_0}^{n}\left(1 + \mu(2^{k-1})^2 \frac{\partial^2}{\partial x^2}\right)\phi(x_j) + O(h^4) - \phi(x_j) \qquad (43)\\ &= \mu h^2\left(\sum_{k=k_0}^{n} (2^{k-1})^2\right)\frac{\partial^2}{\partial x^2}\phi(x_j) + O(h^4)\\ &= \mu h^2 \frac{2^{2n} - 2^{2k_0-2}}{3} \frac{\partial^2}{\partial x^2}\phi(x_j) + O(h^4)\end{aligned}$$

With (39) it follows that

$$(SA(\xi_h))_j - A_j(\xi_h) = (Sa_h)_j - a(\xi(x_j), \xi_x(x_j), x_j) + h^2((SC)_j - C_j) + O(h^4) \tag{44}$$

It follows from (40) that $(SC)_j - C_j$ is of $O(h^2)$. Furthermore, by assumption $\phi(x) = a(\xi(x), \xi_x(x), x)$, hence, the lemma follows by substitution of (43) into (44).

Corrollary. The error due to the smoothing operator (2.33) is of $O(h^2)$.

Theorem 1. Let the conditions of Lemma 2 be satisfied. Let **F** be defined by (12). Let C be the imaginary stability boundary of an explicit method (see (24)). Then the error due to the smoothing operator (33) is, when the maximum allowed time step is used, for the special case $\mu_k = \frac{1}{2}$, $k_0 = 1$,

$$(S\mathbf{F}(u_h))_j - F_j(u_h) = \frac{(\Delta t/C)^2 - h^2}{6} \frac{\partial^2}{\partial x^2} f(u_x, x_j) + O(h^4) \tag{45}$$

and for the special case $\mu_k = \frac{1}{4}$, $k_0 = 2$

$$(S\mathbf{F}(u_h))_j - F_j(u_h) = \frac{\frac{32}{27}(\Delta t/C)^2 - 2h^2}{6} \frac{\partial^2}{\partial x^2} f(u_x, x_j) + O(h^4) \tag{46}$$

Proof

First we prove (45). Denote by Δt_0 the maximum time step without smoothing. Hence, from (24) $\Delta t_0/h = C$. In Section 2.3.1 we have found that the time step can be increased by a factor 2^n. This gives $(\Delta t/\Delta t_0) = 2^n$. Substituting this into (41) and setting $\xi(x) = u(x, t)$ for some time t, we arrive at (45). The proof of (46) follows the same line, except that from (38), the time step can now be increased by a factor $\frac{3}{4}\sqrt{32^{n-1}}$.

2.4. *An implicit smoothing operator*

Another smoothing operator, we want to introduce, is an implicit one. This smoothing operator is implicitly defined by

$$-\mu(SF)_{j+1}+(1+2\mu)(SF)_j-\mu(SF)_{j-1}=F_j \tag{47}$$

For the eigenfunctions (22), the eigenvalues of this system are

$$\lambda_S=1/[1+4\mu\sin^2(bh/2)] \tag{48}$$

The reduction factor is found by the multiplication of the eigenvalues of S and D, giving

$$\lambda_{SD}=\mathrm{i}\sin(bh)/\{h[1+4\mu\sin^2(bh/2)]\} \tag{49}$$

$$=2\mathrm{i}\sin(bh/2)\cos(bh/2)/\{h[1+4\mu\sin^2(bh/2)]\}$$

Omiting $\cos(bh/2)$, which is less than one, and writing $x=\sin(bh/2)$ we find

$$|\lambda_{SD}|<2x/[h(1+4\mu x^2)],\quad 0<x<1. \tag{50}$$

By differentiation with respect to x we find a maximum of the right-hand side for

$$x=1/\sqrt{(4\mu)},\quad \mu>\tfrac{1}{4}, \tag{51}$$

$$x=1,\quad 0<\mu<\tfrac{1}{4}.$$

Substitution in (50) gives

$$\max|\lambda_{SD}|<1/(2h\sqrt{\mu}) \tag{52}$$

Hence, increasing μ by a factor of four will decrease the maximal eigenvalue by a factor of two.
Notice that from the stability condition (24) and from (52) it follows that

$$\mu\geqslant\tfrac{1}{4}\Delta t^2/(C^2h^2) \tag{53}$$

If μ satisfies this condition, then we have constructed an unconditionally stable method. Compared to the usual implicit time integrators, this method is simpler to implement. Especially if the right-hand side (see (1)) becomes non-linear and complicated.

Theorem 2. Let the conditions of Lemma 2 (see Section 2.3.2) be satisfies. Let $\mathbf{F}$ be given by (12). Let C be the imaginary stability boundary of an explicit method (see (24)). Assume periodic boundary conditions and equality in (53). Then the error due to the implicit smoothing operator (47) is given by

$$(S\mathbf{F}(u_h))_j-F_j(u_h)=\frac{1}{4}\frac{\Delta t^2}{C^2}\frac{\partial^2}{\partial x^2}f(u_x,x_j)+O(h^4) \tag{54}$$

Proof

From (47) we obtain

$$(S^{-1}\mathbf{F})_j=-\mu F_{j+1}+(1+2\mu)F_j-\mu F_{j-1} \tag{55}$$

Using Gerschgorin's theorem,[9] we have that the minimum eigenvalue of S^{-1} is greater than or equal to 1. Hence the spectral radius of S is smaller than or equal to 1 and thereby S is bounded in any matrix norm. Let S' be defined by

$$(S'\mathbf{F})_j=\mu F_{j+1}+(1-2\mu)F_j+\mu F_{j-1} \qquad (56$$

then for a test function $\phi(x)\in C^2$ we have

$$\begin{aligned}(S\phi_h - S'\phi_h)_j &= \{S(\phi_h - S^{-1}S'\phi_h)\}_j \\ &= \left\{S\left[\phi_h - S^{-1}\left(1+\mu h^2\frac{\partial^2}{\partial x^2}\right)\phi_h + O(h^3)\right]\right\}_j \\ &= \{S(\phi_h - S^{-1}\phi_h)\}_j + O(h^2) \\ &= \left\{S\left[\phi_h - \left(1+\mu h^2\frac{\partial^2}{\partial x^2}\right)\phi_h\right]\right\}_j + O(h^2) \\ &= \left\{S\left(-\mu h^2\frac{\partial^2}{\partial x^2}\phi_h\right)\right\}_j + O(h^2) = O(h^2) \qquad (57)\end{aligned}$$

From (39), (56) and (57) it follows that

$$\begin{aligned}(S\mathrm{F}(u_h))_j - F_j(u_h) &= \{S[\mathrm{F}(u_h) - S^{-1}\mathrm{F}(u_h)]\}_j \\ &= \{S[f_h - S^{-1}f_h + h^2(\mathrm{C} - S^{-1}\mathrm{C})]\}_j + O(h^4) \\ &= \left\{S\left[\mu h^2\frac{\partial^2}{\partial x^2}f(u_x,x)\right]_h\right\}_j + O(h^4) \\ &= \left\{S'\left[\mu h^2\frac{\partial^2}{\partial x^2}f(u_x,x)\right]_h\right\}_j + O(h^4) \\ &= \mu h^2\frac{\partial^2}{\partial x^2}f(u_x,x_j) + O(h^4) \qquad (58)\end{aligned}$$

From (58) and equality in (53), we obtain (54).

2.5. *Systems*

In the case of systems, the same smoothing operators can be used again. This proceeds as follows. First the terms on the right-hand side which play an important role with respect to stability have to be determined. These terms contain, in general, a derivative in one of the space directions. Then the right-hand side of the equation containing such a term has to be smoothed in the same direction as that of the derivative. If this equation contains important derivatives with respect to stability in more directions, then this equation is successively smoothed in all these directions. We will clarify this by an example.

Example 3. The shallow-water equations can be written in the form

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_{xx}, \mathbf{u}_{yy}, x, y, t) \qquad (59)$$

where $\mathbf{u} = (u, v, \zeta)^{\mathrm{T}}$ and

$$\begin{aligned}f^u(\cdot) &= -uu_x - vu_y - g\zeta_x + \nu\Delta u - C_z\sqrt{(u^2+v^2)}u/H \\ f^v(\cdot) &= -uv_x - vv_y - g\zeta_y + \nu\Delta v - C_z\sqrt{(u^2+v^2)}v/H \\ f^\zeta(\cdot) &= -(Hu)_x - (Hv)_y \\ H(x,y,t) &= h(x,y) + \zeta(x,y,t)\end{aligned}$$

where g, ν and C_z are positive constants. These equations are again discretized by standard central differences (13). In many applications it suffices to consider for stability a reduced Jacobian of the space discretized form of (59), i.e.

$$\mathbf{J}_r = -\begin{bmatrix} 0 & 0 & gD_x \\ 0 & 0 & gD_y \\ \tilde{H}D_x & \tilde{H}D_y & 0 \end{bmatrix} \tag{60}$$

where D_x and D_y are the discretizations of $\partial/\partial x$ and $\partial/\partial y$, respectively, and $\tilde{H}$ is a constant approximating $H(x, y, t)$. If this Jacobian is applied to a Fourier component $\exp[i(b_1 x + b_2 y)]$, then we obtain the eigenvalues 0, $\pm\sqrt{[g\tilde{H}(\lambda_x^2 + \lambda_y^2)]}$, where λ_x and λ_y are the eigenvalues of D_x and D_y for the Fourier component, respectively. According to this outcome, it seems appropriate to smooth the right-hand sides of the successive equations in the x-, y- and x, y-directions, respectively. Using the same one-dimensional smoothing operators as in the previous sections, the third right-hand side has to be smoothed in x- and y-directions, successively. After smoothing, the Jacobian matrix becomes

$$\tilde{\mathbf{J}}_r = -\begin{bmatrix} 0 & 0 & gS_xD_x \\ 0 & 0 & gS_yD_y \\ \tilde{H}S_yS_xD_x & \tilde{H}S_xS_yD_y & 0 \end{bmatrix} \tag{61}$$

where S_x and S_y denote the smoothing operators in the x- and y-directions, respectively. The eigenvalues of the Jacobian after smoothing are 0, $\pm\sqrt{\{g\tilde{H}[\lambda_{S_y}(\lambda_{S_x}\lambda_x)^2 + \lambda_{S_x}(\lambda_{S_y}\lambda_y)^2]\}}$, where λ_{S_x} snd λ_{S_y} denote the eigenvalues of S_x and S_y, respectively. Hence, as long as the neglected terms in (59) do not become important with respect to stability, it is possible to reduce the modulus of the maximum eigenvalue to any desired magnitude.

3. NUMERICAL ILLUSTRATIONS

To illustrate the foregoing theory, we will give examples of the stabilization for linear and non-linear problems. Furthermore, an application to the shallow-water equations will be shown.

3.1. A linear problem

The linear problem is defined by

$$\begin{aligned} u_t &= u_x - 16\pi/L\cos(32\pi x/L), \quad 0 < t < T, \quad 0 < x < L \\ u(x, 0) &= 0{\cdot}5\sin(2\pi x/L) + 0{\cdot}5\sin(32\pi x/L) \\ u(0, t) &= u(L, t) \end{aligned} \tag{62}$$

where $L = 100$. The exact solution of this problem is

$$u(x, t) = 0{\cdot}5\sin(2\pi(x + t)/L) + 0{\cdot}5\sin(32\pi x/L) \tag{63}$$

Hence, the solution consists of a non-stationary part, which is slowly varying both in the time and in the space variable, and a stationary part which varies rapidly in the space variable only.

Therefore, the numerical approximation of the stationary part needs a finer space mesh than the non-stationary part. This fine space mesh does, when no smoothing is used, severely restrict the time step. Here, we will give the accuracy results for five methods which all have the same semi-discretization (12). The basic time integrator we use is the classical fourth order Runge–Kutta method.[5] This method, which is used by various others,[6,10,11] is conditionally stable for hyperbolic partial differential equations. The imaginary stability boundary of this method is $C = 2\sqrt{2}$. The methods are:

Table I. Numerical results using smoothing operators with $T = 2{\cdot}8 \times 128$, $h = L/N$

			Correct digits, $N = 384$			Correct digits on coarser grids	
Δt	RK4	RK4E1	RK4E2	RK4I	CN	RK4	N
0·7	2·0	2·0(0)	2·0(0)	2·0(0)	1·9	2·0	384
1·4	-	2·1(1)		2·0(1)	1·7	1·6	192
1·866	-		2·0(1)				
2.8	-	1·9(2)		1·7(1)	1·4	0·9	96
3·733	-		1·7(2)				
5·6	-	1·4(3)		1·3(1)	0·9	0·3	48
7·466	-		1·2(3)				
11·2	-	0·8(4)		0·7(1)	0·4	−0·1	24
14·933	-		0·6(4)				

RK4	the classical Runge–Kutta method without smoothing
RK4E1	the classical Runge–Kutta method with smoothing operator (33), where $\mu_k = \frac{1}{2}$ and $k_0 = 1$, $n = [1 + \log_2(\Delta t/(2\sqrt{2}h))]$
RK4E2	the classical Runge–Kutta method with smoothing operator (33), where $\mu_k = \frac{1}{4}$ and $k_0 = 2$, $n = [2 + \log_2(\Delta t/(2\sqrt{2}h))]$ for $1 < \Delta t/(2\sqrt{2}h) < \frac{3}{2}$ and $n = [2 + \log_2(\frac{4}{9}\sqrt{3}\Delta t/(2\sqrt{2}h))]$ for $\Delta t/(2\sqrt{2}h) > \frac{3}{2}$
RK4I	the classical Runge–Kutta method with the implicit smoothing operator (47), where $\mu = \frac{1}{4}\Delta t^2/(2\sqrt{2}h)^2$ for $\Delta t/(2\sqrt{2}h) > 1$, and
CN	the Crank–Nicolson method.

The brackets, [ ], in the expressions for the determination of n denote the entier function. Furthermore, no smoothing is performed for $\Delta t/(2\sqrt{2}h) < 1$ in RK4E1, RK4E2 and RK4I. In Table I we give the number of correct digits produced by these integration methods, i.e. the $-\log_{10}(|\textit{maximum error}|)$, and in parentheses the number of smoothings.

The main part of the table presents the results on a grid with 384 grid points. For reference, we also added results of the RK4 method on coarser grids using the corresponding maximum allowed time steps. These time steps are given in the first column. The hyphens in the column of RK4 denote that the method is unstable for the corresponding time step. The results of RK4E1 and RK4E2 are given for time steps Δt which are the maximum allowed for the corresponding number of smoothings. For RK4E1, this results in a doubling of the allowed time step, each time a new operator is applied. If in RK4E2 the first operator of the product sequence is applied, a factor $\frac{3}{2}\sqrt{3}$ is gained (see (36) and (38)). Thereafter, as with RK4E1, a factor two is gained each time a new smoothing operator of the sequence is applied. RK4I and CN were applied using the same step sizes as RK4E1. Because n is an integer, the increase of the maximum allowed time step proceeds in a discreet way. However, for accuracy reasons it may be desirable to have a smooth increase of the time step as the right-hand side is smoothed more and more. Without going into details, we mention that this can be established by varying the coefficient μ_k of the last smoothing operator in the product sequence (34).

The results on the fine grid ($N = 384$) develop in the same way for all methods when the time step increases: at first, the number of correct digits changes slightly; then, when the time step becomes larger than about 3·5, the number of correct digits decreases rapidly. This can be understood by the following reasoning. The error due to the stationary part of the solution is independent of the time step. For this problem, this error is rather large because of the large space derivatives of the stationary part of the solution. Of course, the error due to the non-stationary part is dependent on the time step. Hence, for a certain time step the error due to the non-stationary part becomes larger than that due to the stationary part of the solution. This time step is about 3·5 for this problem.

The results on coarser grids clearly show the need for a calculation on the fine grid, because the number of correct digits rapidly decreases on coarser grids. This error is due to the stationary part of the solution.

3.2. *A non-linear problem*

In this section, we will use the stabilization technique for a non-linear equation. The problem is given by

$$u_t = uu_x + g(x,t), \quad 0 < t < T, \quad 0 < x < L \tag{64}$$

where $L = 100$. The forcing function g is chosen such that we have a solution consisting of a part which is slowly varying in both the time and the space variables, and a part which varies relatively rapidly in the space variable only. The solution is given by

$$u(x,t) = 0{\cdot}5\sin(2\pi(x+t)/L) + 0{\cdot}5\sin(8\pi x/L) \tag{65}$$

Hence, the function g follows to be

$$\begin{aligned} g(x,t) = 2\pi/L\{&0{\cdot}5\cos(2\pi(x+t)/L) - [0{\cdot}5\sin(2\pi(x+t)/L) + 0{\cdot}5\sin(8\pi x/L)] \\ &\times [0{\cdot}5\cos(2\pi(x+t)/L) + 2\cos(8\pi x/L)]\} \end{aligned} \tag{66}$$

The initial condition is taken from the exact solution (65).

We discretized the non-linear term uu_x by

$$\{(u_{j+1} + 2u_j + u_{j-1})/4\}(u_{j+1} - u_{j-1})/(2h) \tag{67}$$

Owing to the non-linear nature of equation (3), almost any time integration will become unstable after a certain time period. In our experiments, this discretization (67) performed quite well. For more details on discretizations for non-linear problems we refer to References 12–15. For the time discretization, we applied the same time integrators as in Section 3.1, except for the CN method. This method is modified to

$$\begin{aligned} u_j^{n+1} = u_j^n &+ \tfrac{1}{2}\Delta t[((u_{j+1}^n + 2u_j^n + u_{j-1}^n)/4)(u_{j+1}^{n+1} - u_{j-1}^{n+1})/(2h)] \\ &+ \tfrac{1}{2}\Delta t[((u_{j+1}^{n+1} + 2u_j^{n+1} + u_{j-1}^{n+1})/4)(u_{j+1}^n - u_{j-1}^n)/(2h)] \\ &+ \Delta t g(x_j, t + \Delta t/2) \end{aligned} \tag{68}$$

This modification is linearly implicit and still second order in time. In the following this method is called MCN. Table II gives the results in the same form as in Table I

Globally, we observe the same effect for the explicit methods as in the previous section: at first the error of the time stepping is negligible with respect to that of the space discretization; then, when the time step becomes larger than about 5, the error due to the time stepping becomes significant. Furthermore, we find that the application of the smoothing operators gives at first a slight increase of the number of correct digits. This is possibly due to an annihilation of errors. The MCN method performs relatively poorly for this problem, which is caused by the larger error constants of its time discretization.

The implicit smoothing operator is of course more expensive than one explicit smoothing operator of the product sequence. However, as the time step increases, we need more and more applications of the explicit smoothing operators, whereas the implicit smoothing operator needs to be applied only once. Hence, after a certain number of applications of explicit smoothing operators,

Table II. Numerical results using smoothing operators with $T = 128 \times 8$, $h = L/N$

		Correct digits, $N = 384$				Correct digits on coarser grids	
Δt	RK4	RK4E1	RK4E2	RK4I	MCN	RK4	N
0·8	2·0	2·0(0)	2·0(0)	2·0(0)	1·8	2·0	384
1·6	-	2·6(1)		2·4(1)	1·2	1·4	192
2·1	-		2·4(1)				
3·2	-	2·3(2)		2·1(1)	0·2	1·0	96
4·2	-		2·1(2)				
6·4	-	1·8(3)		1·6(1)		0·6	48
8·4	-		1·6(3)				
12·8	-	1·3(4)		0·9(1)		0·0	24
16·8	-		1·3(4)				

explicit smoothing becomes more expensive than implicit smoothing. On a vector computer this number is of course much larger, because the explicit smoothing operators vectorize very well, which is not the case for the implicit smoothing operator.

3.3. *The shallow-water equations*

In this section, we give results of a computation with the shallow-water equations where smoothing is used. These computations are performed on the CYBER 205. On such a vector computer, explicit methods are to be preferred, because they allow a high degree of vectorization. The application of the smoothing operators is such as described in Section 2.5. In this computation, the operator (34) is applied with coefficients depending on k and n, i.e.

$$\mu_k = \tfrac{1}{4}(1 - 2^{-(n+1-k)}) \tag{69}$$

For this choice the eigenvalues of (38) are positive, which appeared important for initial-boundary value problems in order to have a stable integration. With this choice, we found in experiments, that the factor 0·77 in (38) can be replaced by 1. In this computation again the classical Runge–Kutta method is used. For further details we refer to Reference 8. These results will show the relevance of smoothing for flow computations.

The test problem is a square basin, which has sides of length 5 km (see Figure 1). In the middle of the basin is a bump.

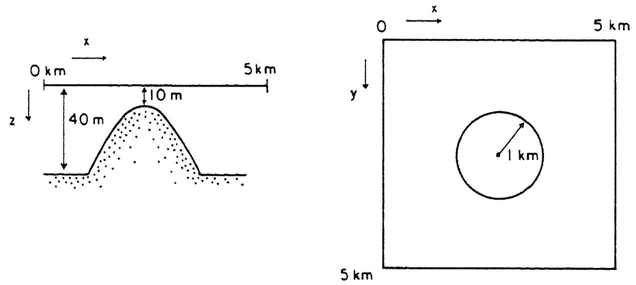

Figure 1. Geometry of a square basin with bump.

The bottom profile $h(x, y)$ is given by

$$h(x, y) = \begin{cases} 40 - 30\cos(\pi r/2)m & \text{for } r < 1 \\ 40m, & \text{elsewhere} \end{cases} \tag{70}$$

where

$$r = \sqrt{[(x - 2{\cdot}5 \times 10^3)^2 + (y - 2{\cdot}5 \times 10^3)^2]}/1000$$

At the left and right boundaries, the elevation ζ is prescribed

$$\begin{aligned} \zeta(0, y, t) &= -\sin(\omega t), \\ \zeta(5{\cdot}0 \times 10^3, y, t) &= -\sin(\omega t - \phi), \end{aligned} \tag{71}$$

where $0 < y < 5{\cdot}0 \times 10^3$ m and

$$\begin{aligned} \omega &= 2\pi/(12 \times 3600)\,\mathrm{s}^{-1} \\ \phi &= 2\pi 5/600 \end{aligned} \tag{72}$$

At the upper and lower boundaries, at $y = 0$ and $y = 5{\cdot}0 \times 10^3$ m, respectively, the normal velocity component v is zero. Furthermore, the constants in equation (59) are chosen to be

$$\nu = 10\,\mathrm{m}^2/\mathrm{s}, \quad C_z = 4 \times 10^{-3}$$

We integrated 15 hours physically with various time steps (see Table III). The calculations were performed on a 24 × 24 grid. At the end of each integration the solution was compared with a reference solution computed on a finer grid (96 × 96). The results are given in significant digits of the v-component. In this case, a root-mean-square error is used, defined by

$$\mathrm{Sd}_2 = -\log_{10}(|v - v_{\mathrm{ref}}|_2/|v_{\mathrm{ref}} - \bar{v}_{\mathrm{ref}}|_2)$$

where

$$\begin{aligned} \bar{v}_{\mathrm{ref}} &= (\textstyle\sum_i v_i)/(24 \times 24) \\ |v|_2 &= \sqrt{(\textstyle\sum_i v_i^2)/(24 \times 24)} \end{aligned} \tag{73}$$

in which the summation is over all grid points. The results are given in Table III.

In this table, the number in parentheses denotes the number of applied smoothing operators. In the first column again the time step is given; it increases downwards with a factor two. In the second column the computation times are given and in the last column the numbers of significant digits are given with, in parentheses, the number of smoothings. The time step $\Delta t = 8$ is the maximum time step without smoothing. The general picture is comparable with the previous cases. At first, the number of significant digits remains constant as the time step increases and then, when the time step becomes greater than 32 s, the error due to the time step becomes dominant.

The computation times show a significant reduction when smoothing is used. Furthermore, they

Table III. Significant digits for the shallow-water equations

	RK4E2	
Δt	Computation time	Sd_2
8	45	2·1(0)
16	26	2·1(1)
32	15	2·1(2)
64	9	1·6(3)

contain information about the overhead of the smoothing, because without smoothing the computation time should decrease by a factor two downwards. The overhead of one smoothing is in this case about $\frac{1}{6}$ of one right-hand side evaluation.

4. CONCLUSIONS

In Section 2, we have set up the theory for the stabilization of explicit methods for purely initial-value problems. For some numerical examples it was shown that the predicted reduction of the spectral radius is correct, even in non-linear partial differential equations. Furthermore, a significant decrease of the computation time was found (see Table III).

Our experiences are that the described stabilization is easy to implement. In fact, by its simplicity, it can be added easily to an existing program. Moreover, we think that the technique can be applied to a large variety of problems, even to problems with non-smooth solutions.[6]

APPENDIX: SMOOTHING OPERATORS OCCURRING IN OTHER TIME INTEGRATORS

In Section 2.1 we have rewritten the implicit backward Euler integrator to an explicit method in which a smoothing operator occurs. We will now show that the backward Euler method also can be considered, for problem $\{(9), (12)\}$, as a two-stage first-order Runge–Kutta scheme where an implicit smoother of the form described in Section 2.5 occurs. Furthermore, applying the well-known Crank–Nicolson method to problem $\{(9), (12)\}$, this method appears to be a second-order two-stage Runge–Kutta scheme, where the same implicit smoothing operator occurs. We rewrite (16) as

$$U_j^{n+1} = U_j^n + \Delta t\{(I - \Delta t^2 D^2)^{-1}(I + \Delta t D)\mathbf{F}(\mathbf{U}^n)\}_j \tag{74}$$

The term $(I - \Delta t^2 D^2)^{-1}$ is an implicit smoothing operator similar to the one described in Section 2.5. Furthermore, if this implicit smoothing operator is omitted from (74), then there remains a two-stage first-order Runge–Kutta scheme, applied to the linear problem $\{(9), (12)\}$. Proceeding in the same way for an application of Crank–Nicolson to $\xi\{(9), (12)\}$ we have

$$U_j^{n+1} = U_j^n + \Delta t\{(I - \Delta t^2 D^2/4)^{-1}(I + \Delta t D/2)\mathbf{F}(\mathbf{U}^n)\}_j \tag{75}$$

Here, again the implicit smoothing operator occurs. Omitting this operator, a two-stage second-order Runge–Kutta method, applied to $\{(9), (12)\}$ remains. However, this scheme, without smoothing operator, is unstable for hyperbolic problems. Hence, by smoothing it is possible to stabilize a method that otherwise would be unstable for all Δt.

ACKNOWLEDGEMENTS

These investigations were supported by the Netherlands Foundation for Technical Research (STW), future Technical Science Branch Division of the Netherlands Organization for the Advancement of Pure Research (ZWO). The experiments were done on a CYBER 750 and CYBER 205 at the expense of the Centre for Mathematics and Computer Science (CWI) and the Control Data Corporation (CDC).

REFERENCES

1. R. Courant, and D. Hilbert, *Methods of Mathematical Physics*, Interscience Publishers, 1962.
2. E. E. Rosinger, 'Nonlinear equivalence, reduction of PDEs to ODEs and fast convergent numerical methods', *Research Notes in Mathematics* 77, Pitman Advanced Publishing Program, Boston-London-Melbourne, 1982.

3. F. Shuman, 'Numerical methods in weather prediction: II, smoothing and filtering', *Monthly Weather Review*, **85**, 357–361 (1957).
4. R. D. Richtmyer, and K. W. Morton, *Difference Methods for Initial Value Problems*, Interscience Publishers, Wiley, New York, 1967.
5. J. D. Lambert, *Computational Methods in Ordinary Differential Equations*, Wiley, New York, 1973.
6. A. Jameson and D. Mavriplis, 'Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh', *AIAA 23rd Aerospace Sciences Meeting*, AIAA-85-0435, Nevada, 1985.
7. J. C. Wilson, 'Stability of Richtmyer type difference schemes in any finite number of space variables and their comparison with multistep strange schemes', *J. Inst. Maths Applics*, **10**, 238–257 (1972).
8. F. W. Wubs, P. J. van der Houwen and B. P. Sommeijer, 'On the construction of optimal smoothing operators for stabilizing explicit time integrators in PDE's', *Report NM86*, Centre for mathematics and Computer Science, Amsterdam, 1986.
9. P. Lancaster, *Theory of Matrices*, Academic Press, New York and London, 1969.
10. P. J. van der Houwen, *Construction of Integration Formulas for Initial Value Problems*, North-Holland Publishing Company, Amsterdam, 1977.
11. N. Praagman, 'Numerical solution of the shallow-water equations by a finite element method', *Thesis*, TH Delft, 1979.
12. A. Grammeltvedt, 'A survey of finite-difference schemes for the Primitive equations for a barotropic fluid', *Monthly Weather Review*, **97**, (1969).
13. J. G. Verwer and K. Dekker, 'Step-by-step stability in the numerical solution of partial differential equations', *Report NW 161/83*, Mathematical Centre, Amsterdam, 1983.
14. A. Arakawa, 'Computational design for long-term numerical integration of the equations of fluid motion: 1. Two-dimensional incompressible flow', *Journal of Computational Physics*, **1**, (1), (1966).
15. A. Arakawa, and V. R. Lamb, 'The UCLA general circulation model', *Methods in Computational Physics*, **17**, (1977).