Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

F.W. Wubs

Stabilization of explicit methods for hyperbolic initial-value problems

Department of Numerical Mathematics Report NM-R8521 September

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.).

Stabilization of Explicit Methods for Hyperbolic Initial-Value Problems

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

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.

1980 Mathematics subject classification: Primary:65M10. Secondary:65M20.
Key Words and Phrases: Stabilization, hyperbolic equations, method of lines, residual averaging

Note: 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 at the expense of the Centre for Mathematics and Computer Science (CWI).

This report will be submitted for publication elsewhere.

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 precise, 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 and therefore provides an 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.

Report NM-R8521
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

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 together with a numerical illustration for an initial-value problem.

2. THEORY

Consider the equation

$$\mathbf{u}_t = \mathbf{f}(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t), \quad \mathbf{x} \in R^n, \quad t > 0, \tag{2.1}$$

where $\mathbf{u}=(u_1(\mathbf{x},t), u_2(\mathbf{x},t), \ldots, u_N(\mathbf{x},t))$, defining a first-order quasi-linear hyperbolic system with N equations [3]. Using explicit methods for (2.1), the time step is restricted by the Courant-Friedrichs-Levy (C.F.L.) stability condition (see (2.15)). 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, we will make clear that small time derivatives of $\mathbf{u}$ imply, under certain conditions, small space derivatives of the right-hand side function in which the exact solution is substituted. We emphasize, that $\mathbf{u}$ itself may have large space derivatives. This observation is the basis of the technique, in which we will smooth $\mathbf{f}$ in order to stabilize the method.

For the stabilization of explicit methods, smoothing is widely used before but then usually the grid function $\mathbf{u}$ is smoothed [11,12], rather than the right-hand side function $\mathbf{f}$. 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 space variables, which in general is not true. As an example, the famous variant of the Lax-Wendrof scheme proposed by Richtmyer and Morton[10] may be regarded as a two-stage second-order Runge-Kutta method[7], 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 under 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 an implicit smoothing operator (see Section 2.2) in order to accelerate the convergence. Our contribution will be the explicit smoothing operators which are less expensive than the implicit smoothing operators, especially if we want to use a vector computer. However, in this paper we confine ourselves to initial value problems.

We will give here some details about the smoothness of the right-hand side function. Trivially, if the variation in time of the solution $\mathbf{u}$ is zero(the stationary case), then, on substitution of this solution, the variation of the right-hand side function with respect to the space variables (and the time variable) is zero. In the case that $\mathbf{u}$ varies slowly in time, we have that all time derivatives of $\mathbf{u}$ are small. By differentiation of (2.1) with respect to time we find

$$\begin{aligned} \mathbf{u}_{tt} &= f_u(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t)\mathbf{u}_t + \mathbf{f}_t(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t) = \\ &= f_u(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t)\, \mathbf{f}(\mathbf{u}, \mathbf{u}_{x_1} \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t) + \mathbf{f}_t(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t), \end{aligned} \tag{2.2}$$

where $f_u(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t)$ denotes the Jacobian matrix; hence, by substituting the solution $\mathbf{u}$ we conclude that the right-hand side in (2.2) is also small. Now, we make the following assumptions:

1 $\mathbf{f}_t(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \mathbf{u}_{x_n}, \mathbf{x}, t)$ is small with respect to $\mathbf{u}_{tt}$,

2 the Jacobian can be written in the form

$$f_u(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t) = \sum_{i=1}^{n} A_i(\mathbf{u}, \mathbf{x}, t) \frac{\partial}{\partial x_i} + B(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t),$$

where $A_i (i = 1...n)$ and B are matrices, and

3 $\mathbf{f}(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t)$ is locally one dimensional; a coordinate transformation, $\mathbf{x} \to \tilde{\mathbf{x}}$, is made such that $\tilde{x}_1$ is in this main direction.

Using these assumptions, we have

$$\mathbf{u}_{tt} = \tilde{A}_1(\mathbf{u}, \tilde{\mathbf{x}}, t) \frac{\partial}{\partial \tilde{x}_1} \mathbf{f}(\mathbf{u}, \mathbf{u}_{\tilde{x}_1}, \mathbf{u}_{\tilde{x}_2}, \ldots, \mathbf{u}_{\tilde{x}_n}, \tilde{\mathbf{x}}, t) + \tag{2.3}$$

$$B(\mathbf{u}, \mathbf{u}_{\tilde{x}_1}, \mathbf{u}_{\tilde{x}_2}, \ldots, \mathbf{u}_{\tilde{x}_n}, \tilde{\mathbf{x}}, t)\, \mathbf{f}(\mathbf{u}, \mathbf{u}_{\tilde{x}_1}, \mathbf{u}_{\tilde{x}_2}, \ldots, \mathbf{u}_{\tilde{x}_n}, \tilde{\mathbf{x}}, t) + \epsilon,$$

where ϵ represents the terms negligible with respect to $\mathbf{u}_{tt}$. It follows from (2.1) that, if $\mathbf{u}_t$ is small, then $\mathbf{f}(\mathbf{u}, \mathbf{u}_{\tilde{x}_1}, \mathbf{u}_{\tilde{x}_2}, \ldots, \mathbf{u}_{\tilde{x}_n}, \tilde{\mathbf{x}}, t)$ is small. Hence, the second term in the right-hand side of (2.3) is insignificant. Now, for non-singular $\tilde{A}_1$, we have that the derivative of $\mathbf{f}$ in the main direction is small if $\mathbf{u}_{tt}$ is small. By assumption 3 we already have that the derivatives of $\mathbf{f}$ in the directions orthogonal to the main direction are small. Repeating the described process, by differentiation of (2.2) with respect to time, will give that all space derivatives of $\mathbf{f}$ are small. For example, we consider the scalar equation

$$u_t = f(u_x, x), \quad x \in R, \tag{2.4}$$

where

$$f(u_x, x) = u_x + g(x) \tag{2.5}$$

and g an arbitrary continuous function of x. Differentiating (2.4) with respect to time gives that the first derivative of f with respect to space is small. Repeating the differentiation with respect to time gives that all derivatives of f with respect to space are small.

The property of a smooth right-hand side function in space can be used effectively to stabilize an explicit time integration method by smoothing of the discretized form of $\mathbf{f}(\mathbf{u}, \mathbf{u}_{x_1}, \mathbf{u}_{x_2}, \ldots, \mathbf{u}_{x_n}, \mathbf{x}, t)$, obtained by the method of lines In this approach, the space discretization gives rise to a system of ordinary differential equations [7],

$$\frac{d}{dt}\mathbf{U} = \mathbf{F}(\mathbf{U}, t), \tag{2.6}$$

where $\mathbf{U}$ 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 (2.6), we propose to solve

$$\frac{d}{dt}\mathbf{U} = S\mathbf{F}(\mathbf{U}, t), \tag{2.7}$$

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

In fact, many stable time integrators, applied to (2.6), can be written as a conditionally stable (explicit) integrator applied to (2.7). We will illustrate this for Eulers backward method applied to the semi-discretization of equation (2.4). The right-hand side function $f(u_x, x)$ in (2.5) is discretized, on a grid with mesh size h, with the usual second-order central differences

$$F_j(\mathbf{U}) = (D\mathbf{U})_j + g(x_j), \quad x_j = jh, \tag{2.8}$$

where

$$(D\mathbf{U})_j = (U_{j+1} - U_{j-1}) / (2h), \tag{2.9}$$

and U_j approximates $u(x)$. When backward Euler is applied to (2.6), with $\mathbf{F}$ given by (2.8), we find

$$U_j^{n+1} - \Delta t (DU)_j^{n+1} = U_j^n + \Delta t\, g(x_j), \tag{2.10}$$

where $\mathbf{U}^n$ approximates the exact solution $\mathbf{U}(t)$ of (2.6) at $t^n = n\Delta t$. This can be rewritten to

$$U_j^{n+1} - \Delta t (DU)_j^{n+1} = U_j^n - \Delta t (DU)_j^n + \Delta t F_j(\mathbf{U}^n). \tag{2.11}$$

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

$$U_j^{n+1} = U_j^n + \Delta t \{ (I - \Delta t D)^{-1} \mathbf{F}(\mathbf{U}^n) \}_j, \tag{2.12}$$

which is simply forward Euler applied to (2.7) with the smoothing operator $S = (I - \Delta t D)^{-1}$. A discussion of this smoothing operator and another example can be found in appendix A. 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 (2.7) with (2.6).

In the remainder of this section, we will continue to illustrate the theory by using the scalar equation (2.4) and its semi-discretization (2.8). We will use linear stability theory[10] to determine the maximum allowed time step for a particular explicit method in the case without smoothing and with smoothing, respectively. Therefore, we have to know the maximum eigenvalue of D for eigenfunctions $\mathbf{V}$ with property $|V_j| = 1$. Clearly, these eigenfunctions of D are

$$V_j = \exp(ibjh) \tag{2.13}$$

for all $b \in R$. The corresponding eigenvalue is now found to be

$$\lambda_D = i \sin(bh) / h, \tag{2.14}$$

which is maximal for $bh = \pm\pi / 2$. This means, that the stability region of the time integrator should contain a non-zero part of the imaginary axis in order to perform a stable integration. For an explicit method, we then find the C.F.L. condition

$$\max(|\lambda_D|)\Delta t < C \;\rightarrow \Delta t < C h, \tag{2.15}$$

where C is a positive constant depending on the time integrator used.

2.1 Explicit smoothing operators

2.1.1 Derivation

Consider the smoothing operator S defined by

$$(S_1 \mathbf{F})_j := (F_{j+1} + F_{j-1}) / 2. \tag{2.16}$$

In order to determine the maximum time step, we now need the eigenvalues of $S_1 D$. These are simply the product of the eigenvalues of S_1 and D, because S_1 and D have the same eigenfunctions (2.13). The eigenvalues of S_1 are

$$\lambda_{S_1} = \cos(bh), \tag{2.17}$$

and the product of the eigenvalues of S and D

$$\lambda_{S_1 D} = \cos(bh) i \sin(bh) / h = i \sin(2bh) / (2h). \tag{2.18}$$

Hence, compared with (2.14) the maximum eigenvalues have been reduced by a factor two. However, this may still be very restrictive. Therefore, we will repeat the smoothing. Defining a second smoothing operation by

$$(S_2 \mathbf{F})_j := (F_{j+2} + F_{j-2}) / 2, \tag{2.19}$$

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

$$S:=\prod_{k=1}^{n} S_k \tag{2.20}$$

where

$$(S_k\mathbf{F})_j:=(F_{j+2^{k-1}}+F_{j-2^{k-1}})/2. \tag{2.21}$$

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 to a discretization on a coarser grid. This appears quite natural, because of the following reasoning. The solution is of the form

$$u(x,t)=r(x+t), \tag{2.22}$$

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

$$S:=\prod_{k=k_0}^{n} S_k, \tag{2.23}$$

where

$$(S_k\mathbf{F})_j:=\mu_k F_{j+2^{k-1}}+(1-2\mu_k)F_j+\mu_k F_{j-2^{k-1}}. \tag{2.24}$$

Notice that the smoothing operator in (2.24) 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 (2.20). Another, special smoothing operator following from (2.23) is the case where where $\mu_k=\frac{1}{4}$ for all k and $k_0=2$. The eigenvalue of (2.24) for this value of μ_k is

$$\lambda_{S_k}=\cos^2(2^{k-2}bh). \tag{2.25}$$

Now, when (2.23) is applied to (2.8), again the corresponding eigenvalues may be multiplied and we find

$$\lambda_{SD}=i\prod_{k=2}^{n}\cos(2^{k-2}bh)\sin(2^{n-1}bh)/(2^{n-1}h). \tag{2.26}$$

In order to approximate the maximum eigenvalue we need the inequality

$$|\cos(x).\sin(2x)|=|2(1-\sin(x))\sin(x)|<\tfrac{4}{9}\sqrt{3}. \tag{2.27}$$

Isolating $\cos(2^{n-2}bh)$ from the product sequence (2.26) and combining it with $\sin(2^{n-1}bh)$, we can apply inequality (2.27) to find an expression for the maximum modulus of (2.26). This gives

$$|\lambda_{SD}|<\tfrac{4}{9}\sqrt{3}/(2^{n-1}h)\approx .77/(2^{n-1}h). \tag{2.28}$$

2.1.2 The smoothing error

Here, we will give an approximation of the error due to the smoothing operation (2.23), for μ_k independent of k. This is achieved by comparing the smoothed and non-smoothed right-hand side function. We will see that the smoothness of the original right-hand side function and the time step, we want to use, 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 1. 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

$$A_j(\xi_h)=a(\xi(x_j),\xi_x(x_j),x_j)+C_jh^2+O(h^4), \tag{2.29}$$

$$C_{j\pm1}=C_j\pm D_jh+O(h^2), \tag{2.30}$$

and, moreover, $a(\xi(x),\xi_x(x),x)\in C^4$, then the error due to the smoothing operator (2.23) is, with $\mu_k=\mu$,

$$(S\mathbf{A}(\xi_h))_j-A_j(\xi_h)=\mu h^2\frac{2^{2n}-2^{2k_0-2}}{3}\frac{\partial^2}{\partial x^2}a(\xi(x_j),\xi_x(x_j),x_j)+O(h^4). \tag{2.31}$$

PROOF Let $\phi(x)=a(\xi(x),\xi_x(x),x)$. Using Taylor expansions, we find by substitution of $\phi(x)$ into (2.24)

$$(S_k\phi_h)_j=(1+\mu(2^{k-1}h)^2\frac{\partial^2}{\partial x^2})\phi(x_j)+O(h^4). \tag{2.32}$$

Hence, we have the following error due to the smoothing (2.23) for $\phi(x_j)$

$$\begin{aligned}(S\phi_h)_j-(x_j)&=\prod_{k=k_0}^{n}(1+\mu(2^{k-1})^2\frac{\partial^2}{\partial x^2})\phi(x_j)+O(h^4)-\phi(x_j)= \\ &=\mu h^2(\sum_{k=k_0}^{n}(2^{k-1})^2)\frac{\partial^2}{\partial x^2}\phi(x_j)+O(h^4)= \\ &=\mu h^2\frac{2^{2n}-2^{2k_0-2}}{3}\frac{\partial^2}{\partial x^2}\phi(x_j)+O(h^4).\end{aligned} \tag{2.33}$$

With (2.29) it follows that

$$(S\mathbf{A}(\xi_h))_j-A_j(\xi_h)=(Sa_h)_j-a(\xi(x_j),\xi_x(x_j),x_j)+h^2((S\mathbf{C})_j-\mathbf{C}_j)+O(h^4). \tag{2.34}$$

It follows from (2.30) that $(S\mathbf{C})_j-C_j$ is of $O(h^2)$. Furthermore, by assumption $\phi(x)=a(\xi(x),\xi_x(x),x)$, hence, the lemma follows by substitution of (2.33) into (2.34).

COROLLARY The error due to the smoothing operator (2.23) is of $O(h^2)$.

THEOREM 1. Let the conditions of Lemma 1 be satisfied. Let $\mathbf{F}$ be defined by (2.8). Let C be the imaginary stability boundary of an explicit method (see (2.15)). Then the error due to the smoothing operator (2.23) is, for the special case $\mu_k=\frac{1}{2}$, $k_0=1$,

$$(S\mathbf{F}(u_h))_j-F_j(u_h)=\frac{(\Delta t/C)^2-h^2}{6}\frac{\partial^2}{\partial x^2}f(u_x,x_j)+O(h^4), \tag{2.35}$$

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

$$(S\mathbf{F}(u_h))-F_j(u_h)=\frac{\frac{32}{27}(\Delta t/C)^2-2h^2}{6}\frac{\partial^2}{\partial x^2}f(u_x,x_j)+O(h^4). \tag{2.36}$$

PROOF First we prove (2.35). Denote by Δt_0 the maximum time step without smoothing. Hence, from (2.15) $\Delta t_0 / h = C$. In Section 2.1.1, 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 (2.31) and setting $\xi(x) = u(x,t)$ for some time t, we have (2.35). The proof of (2.36) follows the same line, except that from (2.28), the time step can now be increased by a factor $\frac{3}{4}\sqrt{3}\, 2^{n-1}$.

2.2 An implicit smoothing operator

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

$$-\mu(S\mathbf{F})_{j+1} + (1+2\mu)(S\mathbf{F})_j - \mu(S\mathbf{F})_{j-1} = F_j, \tag{2.37}$$

For the eigenfunctions (2.13), the eigenvalues of this system are

$$\lambda_S = 1 / (1 + 4\mu \sin^2(bh / 2)). \tag{2.38}$$

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

$$\lambda_{SD} = i \sin(bh) / (h(1 + 4\mu \sin^2(bh / 2))) = \tag{2.39}$$

$$= 2i \sin(bh / 2) \cos(bh / 2) / (h(1 + 4\mu \sin^2(bh / 2))).$$

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

$$|\lambda_{SD}| < 2x / (h(1 + 4\mu x^2)), \quad 0 < x < 1. \tag{2.40}$$

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

$$x = 1 / \sqrt{4\mu}, \ \mu > \tfrac{1}{4}, \tag{2.41}$$

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

Substitution in (2.40) gives

$$\max|\lambda_{SD}| < 1 / (2h\sqrt{\mu}). \tag{2.42}$$

Hence, increasing μ by a factor four will decrease the maximal eigenvalue by a factor two.

Notice that from the C.F.L. condition (2.15) and from (2.42) it follows that

$$\mu \geqslant \tfrac{1}{4} \Delta t^2 / (C^2 h^2). \tag{2.43}$$

If μ satisfies this condition, then we have constructed an unconditionally stable method. Comparing with usual implicit time integrators, this method is simpler to implement. Especially if the right-hand side function (see(2.1)) becomes non-linear and complicated.

THEOREM 2. Let the conditions of Lemma 1 (see Section 2.1.2) be satisfied. Let F be given by (2.8). Let C be the imaginary stability boundary of an explicit method (see (2.15)). Assume periodic boundary conditions and equality in (2.43). Then the error due to the implicit smoothing operator is given by

$$(S\mathbf{F}(u_h))_j - F_j(u_h) = \frac{1}{4} \frac{\Delta t^2}{C^2} \frac{\partial^2}{\partial x^2} f(u_x, x_j) + O(h^4). \tag{2.44}$$

PROOF Let $\phi(x) = f(u_x, x)$ for some fixed time t. Furthermore, we define $s\phi(x_j) = (S\phi_h)_j$. Then $s\phi \in C^4$, if S has bounded eigenvalues. Using the Gerschgorin theorem [8], it can easily be established that the minimum eigenvalue of S^{-1} (see (2.37)) is at least 1. Hence, the maximum eigenvalue of S is at most 1. The Gerschgorin theorem can only be applied to a finite matrix, which we have obtained

by the assumption of periodic boundary conditions.
Using Taylor expansions, it follows from (2.37) that

$$(1-\mu h^2\frac{\partial^2}{\partial x^2})s\phi(x_j)=\phi(x_j)+O(h^4). \tag{2.45}$$

On substitution into (2.45) it can be shown that the linear operator s is of the form

$$s\phi(x)=(1+\mu h^2\frac{\partial^2}{\partial x^2})\phi(x)+O(h^4). \tag{2.46}$$

Hence,

$$(S\phi_h)_j-\phi(x_j)=s\phi(x_j)-\phi(x_j)=\mu h^2\frac{\partial^2}{\partial x^2}\phi(x_j)+O(h^4). \tag{2.47}$$

>From (2.28) and (2.30) it follows that

$$(S\mathbf{F}(u_h))_j-F_j(u_h)=(Sf_h)_j-f(u_x,x_j)+h^2((S\mathbf{C})_j-C_j)+O(h^4). \tag{2.48}$$

As S is symmetric and $S\mathbf{e}=\mathbf{e}$, where $\mathbf{e}=[1,1,..,1]^T$, we have that $(S\mathbf{C})_j-C_j$ is of $O(h^2)$. Using (2.47) and equality in (2.43), we have (2.44).

2.3 Numerical illustration
To illustrate the foregoing theory, we will give an example of the stabilization for a linear and a non-linear problem.

2.3.1 A linear problem
The linear problem is defined by

$$\begin{aligned} &u_t=u_x-16\pi/L\cos(32\pi x/L),\quad 0<t<T,\quad 0<x<L,\\ &u(x,0)=5\sin(2\pi x/L)+.5\sin(32\pi x/L),\\ &u(0,t)=u(L,t), \end{aligned} \tag{2.49}$$

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

$$u(x,t)=.5\sin(2\pi(x+t)/L)+.5\sin(32\pi x/L). \tag{2.50}$$

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 (2.8). The basic time integrator we use, is the classical fourth order Runge-Kutta method [7]. This method, which is used by various others [5,9,6], is conditionally stable for hyperbolic partial differential equations. The imaginary stability boundary of this method is $C=2\sqrt{2}$. The methods are:

RK4 the classical Runge-Kutta method without smoothing

RK4E1 the classical Runge-Kutta method with smoothing operator (2.23), 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 (2.23), 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 (2.37), where $\mu = \frac{1}{4}\Delta t^2 / (2\sqrt{2}h)^2$ for $\Delta t / (2\sqrt{2}h) > 1$, and

CN the Cranck-Nickolson 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 2.1, 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.

Δt	Correct digits, N=384					Correct digits on coarser grids	
	RK4	RK4E1	RK4E2	RK4I	CN	RK4	N
.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	.9	96
3.733	-		1.7(2)				
5.6	-	1.4(3)		1.3(1)	.9	.3	48
7.466	-		1.2(3)				
11.2	-	.8(4)		.7(1)	.4	-.1	24
14.933	-		.6(4)				

Table 2.1: Numerical results using smoothing operators with $T = 2.8 \times 128$, $h = L / N$.

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 hyphons 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 integer n. 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 (2.26) and (2.28)). 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 function is smoothed more and more. Without going into details, this can be established by varying the coefficient μ_k of the last smoothing operator in the product sequence (2.24).

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 the one 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.

2.3.2 A non-linear problem

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

$$u_t = uu_x + g(x,t), \quad 0<t<T, \quad 0<x<L, \tag{2.51}$$

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

$$u(x,t) = .5\sin(2\pi(x+t)/L) + .5\sin(8\pi x/L). \tag{2.52}$$

Hence, the function g follows to be

$$\begin{aligned} g(x,t) = 2\pi/L\{&.5\cos(2\pi(x+t)/L) - [.5\sin(2\pi(x+t)/L) + .5\sin(8\pi x/L)] \\ &[.5\cos(2\pi(x+t)/L) + 2\cos(8\pi x/L)]\} \end{aligned} \tag{2.53}$$

The initial condition is taken from the exact solution (2.52).
We discretized the non-linear term uu_x by

$$\{(u_{j+1} + 2u_j + u_{j-1})/4\}.(u_{j+1} - u_{j-1})/(2h) = \{(u_{j+1} + u_j)^2 - (u_j + u_{j-1})^2\}/(8h). \tag{2.54}$$

Due to the non-linear nature of equation (2.51), almost any time integration will become unstable after a certain time period. In our experiments, this discretization (2.54) performed quite well. For more details on discretizations for non-linear problems we refer to [4], [13], [1] and [2]. For the time discretization, we applied the same time integrators as in Section 2.3.1, except for the *CN* method. This method is modified to

$$\begin{aligned} u_j^{n+1} = u_j^n &+ \tfrac{1}{2}\Delta t\,[((u_{j+1}^n + 2u_j^n + u_{j-1}^n)/4)\cdot(u_{j+1}^{n+1} - u_{j-1}^{n+1})/(2h)] \\ &+ \tfrac{1}{2}\Delta t\,[((u_{j+1}^{n+1} + 2u_j^{n+1} + u_{j-1}^{n+1})/4)\cdot(u_{j+1}^n - u_{j-1}^n)/(2h)] \\ &+ \Delta t\, g(x_j, t + \Delta t/2). \end{aligned} \tag{2.55}$$

This modification is linearly implicit and still second order in time. In the following this method is called MCN. We now give the results in the same form as in Table 2.1

Δt	Correct digits, N = 384					Correct digits on coarser grids	
	RK4	RK4E1	RK4E2	RK41	MCN	RK4	N
.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)	.2	1.0	96
4.2	-		2.1(2)				
6.4	-	1.8(3)		1.6(1)		.6	48
8.4	-		1.6(3)				
12.8	-	1.3(4)		.9(1)		.0	24
16.8	-		1.3(4)				

Table 2.2: Numerical results using smoothing operators with $T = 128 \times 8$, $h = L / N$.

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 bad for this problem, which is mainly caused by the linearization of the Cranck-Nickolson method.

The computational costs of the explicit smoothing operators are relatively low with respect to the right-hand side evaluation. This is due to the expensive sine and cosine evaluations in (2.49) and (2.51). But even in the case where the right-hand side function is as cheap as the smoothing operation, it pays to use smoothing as is made clear in Section 2.1.1.
The implicit smoothing operator is of course more expensive than one explicit smoothing operator. 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, 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. Conclusions

In the previous section, we have set up the theory for the stabilization of explicit methods for purely initial-value problems. An analogous theory can be developed for initial-boundary-value problems [14]. This will be subject of a future paper.
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.

References

[1] Arakawa, A., Computational Design for Long-Term Numerical Integration of the Equations of Fluid Motion: 1. Two-Dimensional Incompressible Flow, Journal of Computational Physics, Vol. 1, No. 1, Academic Press, New York, 1966.

[2] ARAKAWA, A., AND V.R. LAMB, The Ucla General Circulation Model, Methods in Computational Physics, Vol. 17, 1977.
[3] COURANT, R. AND D. HILBERT, Methods of Mathematical Physics, Interscience Publishers, 1962.
[4] GRAMMELTVEDT, A., A Survey of Finite-Difference Schemes for the Primitive Equations for a Barotropic Fluid, Monthly Weather Review, Vol. 97., 1969.
[5] HOUWEN, P.J. VAN DER, Construction of Integration Formulas for Initial Value Problems, North-Holland Publishing Company, Amsterdam, 1977.
[6] JAMESON, A., 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] LAMBERT, J.D., Computational Methods in Ordinary Differential Equations, Wiley, London-New York, 1973.
[8] LANCASTER, P., Theory of Matrices, Academic Press, New York and London, 1969.
[9] PRAAGMAN, N., Numerical Solution of the Shallow-Water Equations by a Finite Element Method, Thesis, TH Delft, 1979.
[10] RICHTMYER, R.D. AND K.W. MORTON, Difference Methods for Initial Value Problems, Interscience Publishers, Wiley, New York, London, 1967.
[11] ROSINGER, E.E., 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.
[12] SHUMAN, F., Numerical Methods in Weather Prediction: II, Smoothing and Filtering, Monthly Weather Review, Vol. 85, pp 357-361, 1957.
[13] VERWER, J.G., AND K. DEKKER, Step-by-Step Stability in the Numerical Solution of Partial Differential Equations, Report NW 161/83, Mathematical Centre, Amsterdam, 1983.
[14] WUBS, F.W., Smoothing Techniques for Initial-Boundary-Value Problems, in preparation.

Appendix A: Smoothing operators occurring in other time integrators

In Section 2, 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 {(2.6),(2.8)},as a two-stage first-order Runge-Kutta scheme where an implicit smoother of the form described in Section 2.3 occurs. Furthermore, applying the well-known Crank-Nickolson method to problem {(2.6),(2.8)}, this method appears to be a second-order two-stage Runge-Kutta scheme, where the same implicit smoothing operator occurs.
We rewrite (2.12) to

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

The term $(I-\Delta t^2 D^2)$ is an implicit smoothing operator similar to the one described in Section 2.3. Furthermore, if this implicit smoothing operator is omitted from (A.1), then there remains a two-stage first-order Runge-Kutta scheme, applied to the linear problem{(2.6),(2.8)}. Proceeding in the same way for an application of Crank-Nickolson to {(2.6),(2.8)} we have

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

Here, again the implicit smoothing operator occurs. Omitting this operator, a two-stage second-order Runge-Kutta method, applied to {(2.6),(2.8)} 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.