IMA Journal of Numerical Analysis (1990) **9**, 361–378

Improving the Stability of Predictor–Corrector Methods by Residue Smoothing

P. J. VAN DER HOUWEN AND B. P. SOMMEIJER

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

[Received 18 August 1988 and in revised form 19 January 1990]

Residue smoothing is usually applied in order to accelerate the convergence of iteration processes. Here, we show that residue smoothing can also be used in order to increase the stability region of predictor–corrector methods. We shall concentrate on increasing the real stability boundary. The iteration parameters and the smoothing operators are chosen such that the stability boundary becomes as large as $c(m, q)m^2 4^q$ where m is the number of right-hand side evaluations per step, q the number of smoothing operations applied to each right-hand side evaluation, and $c(m, q)$ a slowly varying function of m and q, of magnitude 1.3 in a typical case. Numerical results show that, for a variety of linear and nonlinear parabolic equations in one and two spatial dimensions, these smoothed predictor–corrector methods are at least competitive with conventional implicit methods.

1. Introduction

CONSIDER THE initial value problem

$$dy/dt = f(t, y), \qquad y(t_0) \text{ prescribed} \quad \text{for } t_0 \leqslant t \leqslant T, \tag{1.1}$$

and apply the implicit linear k-step method defined by the characteristic polynomials

$$\rho(\zeta) = \sum_{i=0}^{k} a_i \zeta^{k-i}, \qquad \sigma(\zeta) = \sum_{i=0}^{k} b_i \zeta^{k-i}, \tag{1.2}$$

with $a_0 = 1$ and $b_0 \neq 0$. Then, in order to obtain the numerical approximation y_{n+1} to $y(t_{n+1})$, we have to solve the equation

$$y - b_0 \tau f(t_{n+1}, y) - \Sigma_n = 0, \tag{1.3a}$$

where $\tau := t_{n+1} - t_n$ and Σ_n denotes the sum of already computed back values, i.e.

$$\Sigma_n := \sum_{i=1}^{k} [-a_i y_{n+1-i} + b_i \tau f(t_{n+1-i}, y_{n+1-i})]. \tag{1.3b}$$

We shall be particularly interested in the case where (1.1) originates from the semidiscretization of parabolic initial boundary-value problems in two or three spatial dimensions. In such cases, the solution of (1.3) is usually rather time consuming. If functional iteration is used (e.g. predictor–corrector iteration), then rather small values of τ are required, not to obtain a sufficiently accurate solution of (1.3), but rather to keep the integration process stable. Therefore, functional iteration may cost a large amount of computational effort to reach the end point.

In van der Houwen & Sommeijer (1983) generalizations of predictor–corrector iteration, which allow for much larger values of τ and yet preserve stability, have been proposed. In the special case of semi-discrete partial differential equations, the efficiency of these *generalized predictor–corrector methods* (GPC methods) can be further improved by employing *residue smoothing*; that is, instead of (1.3), we solve the 'preconditioned' equation

$$S(y - b_0\tau f(t_{n+1}, y) - \Sigma_n) = 0, \tag{1.4}$$

where S is a nonsingular smoothing operator which removes the high frequencies from the vector to which it is applied. Residue smoothing has been used in several papers in order to accelerate the *convergence* of iteration processes (cf., e.g. Lerat, 1979; Jameson, 1983; Turkel, 1985; Jameson & Mavriplis, 1985; van der Houwen *et al.*, 1988).

However, residue smoothing can also be used to improve the *stability* of time integration methods (see, e.g., Swanson & Turkel, 1986). In the present paper, we use residue smoothing in order to improve the stability of predictor–corrector methods. The smoothing operators employed are of explicit type and are related to the smoothing techniques used in Wubs (1986) and in van der Houwen *et al.* (1986). They differ from the smoothing operators investigated by Lerat, Jameson and Turkel in the papers mentioned above. The combination of GPC methods and explicit residue smoothing techniques results in a huge improvement of the stability properties. In fact, as we shall see in the numerical experiments, it is now feasible to take timesteps of the same size as the meshwidth used in the semi-discretization. This is quite remarkable for an explicit method, since the major impediment for most explicit methods is that the timestep should behave as the square of the meshwidth to preserve stability.

In Section 2 an expression for the local error of smoothed GPC methods is derived. From this expression, the order conditions of the method easily follow. Section 3 presents the main part of the paper. It provides expressions for the iteration parameters which generate 'almost' maximal *real stability boundaries* for a class of predictor–corrector pairs. The magnitude of the stability boundary β is of the form

$$\beta = c(m, q)m^2 4^q, \tag{1.5}$$

where $c(m, q)$ is a slowly varying function of (m, q), m is the number of iterations performed by the GPC method, and q is the number of basic matrix–vector multiplications needed to apply the smoothing operator S (here, a basic matrix–vector multiplication is a tridiagonal matrix–vector multiplication in one-dimensional problems and a block-tridiagonal matrix–vector multiplication in two-dimensional problems).

We emphasize that the result (1.5) is derived for the model situation where the smoothing operator is a polynomial expression of the Jacobian matrix of the system of differential equations (1.1). In practice, however, we replace the

Jacobian matrix by a simple difference matrix which is fixed for rather large problem classes (see the discussion in Section 3.3). It turns out that even when this difference matrix is a very crude approximation to the true Jacobian matrix, the predicted stability boundary is quite reliable.

The smoothed GPC method has been applied to a variety of parabolic Dirichlet boundary-value problems, both of linear and nonlinear type and both in one and two spatial dimensions; its efficiency has been compared with the efficiency of more conventional implicit methods. On the basis of computational effort versus accuracy, the conventional methods are competitive for one-dimensional problems, but considerably less efficient in two-dimensional problems (see the tables of results in Section 5). However, in our opinion, the main advantage of the smoothed GPC method is its extremely simple implementation (cf. Section 4).

2. The SGPC method

If a GPC method is applied to equation (1.4) we obtain the computational scheme

$y_{n+1}^{(0)}$ = some initial approximation to the solution of (1.3),

$$y_{n+1}^{(j)} = \sum_{l=1}^{j}\left[\left(\mu_{jl}+\frac{\bar{\mu}_{jl}}{b_0}\right)y_{n+1}^{(l-1)} - \frac{\bar{\mu}_{jl}}{b_0}S(y_{n+1}^{(l-1)} - b_0\tau f_{n+1}^{(l-1)} - \Sigma_n)\right] \quad (j=1,\dots,m), \qquad (2.1\text{a})$$

$$y_{n+1} = y_{n+1}^{(m)},$$

where $f_{n+1}^{(l)} := f(t_{n+1}, y_{n+1}^{(l)})$ and where the parameters $\bar{\mu}_{jl}$ and μ_{jl} satisfy the condition

$$\sum_{l=1}^{j}\left(\mu_{jl}+\frac{\bar{\mu}_{jl}}{b_0}\right) = 1 \quad (j=1,\dots,m). \qquad (2.1\text{b})$$

By virtue of this condition the solution of (1.3) satisfies the scheme (2.1). The smoothed GPC method (SGPC method) defined by (2.1) reduces to the GPC method analysed in van der Houwen & Sommeijer (1983) if we set $S = I$, I denoting the identity matrix. Following the proof of Theorem 3.1 given in this reference, we arrive at the following theorem on the local (truncation) error of (2.1), where we have assumed that the back values are exact.

THEOREM 2.1 *Let f be sufficiently differentiable, and define the polynomials*

$$P_0(x) = 1, \qquad P_j(x) = \sum_{l=1}^{j}[\mu_{jl} + \bar{\mu}_{jl}x]P_{l-1}(x) \quad (j=1,\dots,m), \qquad (2.2)$$

and the matrices

$$Z := \tau(\partial f/\partial y)(t_{n+1}, \eta), \qquad \hat{Z} := SZ + (I-S)/b_0, \qquad (2.3)$$

where η is the solution of (1.3). *Then, the local error of $y_{n+1}^{(j)}$ in* (2.1) *is given by*

$$y_{n+1}^{(j)} - y(t_{n+1}) = [I - P_j(\hat{Z})][\eta - y(t_{n+1})] + P_j(\hat{Z})[y_{n+1}^{(0)} - y(t_{n+1})]$$
$$+ O(\tau^{3+2\min(p,\bar{p})}) \quad (j=0,\dots,m),$$

where p and $\bar{p}$ respectively denote the orders of accuracy of the corrector (1.3) *and the predictor used to obtain* $y_{n+1}^{(0)}$.

COROLLARY 2.1 *Let the iteration polynomial $P_m(x)$ have a zero at $x = 0$ of multiplicity r and let the smoothing operator S satisfy the condition*

$$S = I + O(\tau^s) \quad \text{as } \tau \to 0, \text{ for } s > 0.$$

Then the SGPC *method has order*

$$p^* := \min\{p, \bar{p} + r, \bar{p} + rs, 2(1 + \min(p, \bar{p}))\}.$$

Thus, if low-order predictors are used, we have to choose $P_m(x)$ such that r is sufficiently large in order to compensate the low value of $\bar{p}$. For example, for given p and $\bar{p}$, we can only achieve $p^* = p$ if $\bar{p} \geqslant \frac{1}{2}(p - 2)$ and $r \geqslant (p - \bar{p}) \max\{1, 1/s\}$. We observe that s is not necessarily an integer and that $P_j(x)$ should always satisfy the condition $P_j(1/b_0) = 1$ in order to fulfil condition (2.1b).

3. Stability

3.1 *The Characteristic Equation*

For the stability analysis we employ the linear test equation

$$\mathrm{d}y/\mathrm{d}t = Jy, \tag{3.1}$$

where J is a constant matrix. Let $y_{n+1}^{(0)}$ be computed by an explicit linear $\bar{k}$-step method defined by polynomials $\bar{\rho}(\zeta)$ and $\bar{\sigma}(\zeta)$, and assume $\bar{a}_0 = 1$. Then, on substitution of (3.1) into (2.1) we are led to the recursion

$$(P_m(\hat{Z}) - I)S(\rho(E) - Z\sigma(E))y_n = (I - b_0\hat{Z})P_m(\hat{Z})(\bar{\rho}(E) - Z\bar{\sigma}(E))y_{n+k-\bar{k}}, \tag{3.2}$$

where we used the notation introduced in (2.2) and (2.3) and where E denotes the forward shift operator. From this recursion we easily deduce the following theorem.

THEOREM 3.1 *Let S and Z share the same eigensystem and let z and $\hat{z}$ denote the eigenvalues of Z and $\hat{Z}$ corresponding to the same eigenvector. Then the characteristic equation of the* SGPC *method in $P(ESC)^mE$ mode is given by*

$$\rho(\zeta) - z\sigma(\zeta) = \frac{(1 - b_0z)P_m(\hat{z})}{P_m(\hat{z}) - 1}[\bar{\rho}(\zeta) - z\bar{\sigma}(\zeta)]\zeta^{k-\bar{k}}. \tag{3.3}$$

Let $z^* := P_m(\hat{z})$, then we define the *stability domain D* by the set of points (z, z^*) in the (z, z^*)-plane where (3.3) has its roots on the unit disk. Under the assumption of Theorem 3.1 we have that $\hat{z}$ is a function of z. This leads us to the stability criterion

$$\big(z, P_m(\hat{z}(z))\big) \in D \quad \text{for all eigenvalues } z \text{ of } Z = \tau J. \tag{3.4}$$

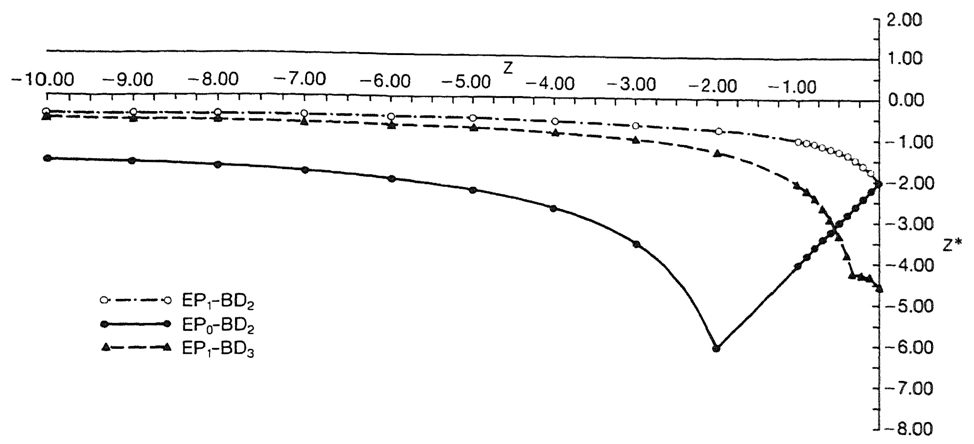

FIG. 3.1. Stability domains of some $EP_{\tilde{p}}$–BD_p methods.

In Figure 3.1 a few stability domains are plotted in the real (z, z^*)-plane for the case where $\{\rho, \sigma\}$ is defined by the pth order *backward differentiation* formula and $\{\tilde{\rho}, \tilde{\sigma}\}$ is defined by the $\tilde{p}$th order *extrapolation* formula, i.e. $\tilde{\rho}(\zeta) = (\zeta - 1)^{\tilde{p}+1}$ and $\tilde{\sigma}(\zeta) \equiv 0$ (the generated SGPC method will be called a *smoothed* $EP_{\tilde{p}}$–BD_p *method*).

In order to apply the stability condition (3.4) we need to know the function $\hat{z} = \hat{z}(z)$. This will be discussed in the next subsection in the case where the smoothing operator S is suitable for use in *parabolic* problems. The general effect of these smoothing operators is the reduction of the length of the (real) eigenvalue interval of the matrix $\hat{Z}$ in (2.3) in comparison with the length of the eigenvalue interval of Z. It will be shown that such a reduction leads to increased real stability boundaries of the SGPC method.

3.2 *Smoothing Operators for Parabolic Problems*

For elliptic difference equations with Dirichlet-type boundary conditions, suitable smoothing operators for residue smoothing were derived in van der Houwen *et al.* (1988). If (1.1) originates from a parabolic problem with Dirichlet boundary conditions, i.e. $\partial f/\partial y$ possesses a negative spectrum, then (1.3) can be considered as an elliptic system of difference equations, so that these smoothing operators are expected to be suitable in the case (1.3) too. However, the boundary equations in (1.3) need some attention as we will see below.

3.2.1 *One-Dimensional Problems* Let M be the number of internal grid points used to semi-discretize the parabolic problem. Then, the system (1.1) contains M equations approximating the parabolic equation at these internal grid points. In addition, we assume that the system (1.1) contains two equations representing the Dirichlet boundary conditions. If the boundary conditions are of the form $u(0, t) = a(t)$, $u(1, t) = b(t)$, where $u(x, t)$ denotes the solution of the parabolic problem, then the first and last equations of (1.1) are given by

$$\frac{dy_0}{dt}=\frac{da(t)}{dt}, \qquad \frac{dy_{M+1}}{dt}=\frac{db(t)}{dt}; \tag{3.5}$$

here the subscripts refer to the components of the vector y and not to the time level. Thus, although the components y_0 and y_{M+1} are explicitly given by, respectively, $a(t)$ and $b(t)$, we assume that they are obtained numerically by integrating the equations (3.5) as part of the system (1.1). As a consequence, the system (1.3) also contains $M+2$ equations, whereas the usual approach defines a system of M equations by eliminating y_0 and y_{M+1} by means of the boundary conditions, i.e. the usual approach prescribes zero residues at the boundary points.

The reason for this unconventional approach can be traced back to the fact that we are not actually *solving* the system (1.3), but we stop the iteration process as soon as the last iterate satisfies the stability condition (3.4). Hence, the residue occurring in (2.1a) is not necessarily decreasing during the iteration process. If we ignore this feature of GPC methods, and just introduce zero-residues at boundary points, then we create a residue vector which may have jumps in the magnitude of its components near the boundary points. Obviously, when smoothing operators are applied to such 'unsmooth' residue vectors, we introduce large errors into the scheme.

Using the equations (3.5) leads to the following boundary equations of the system (1.3):

$$y_0-b_0\tau\frac{da}{dt}(t_{n+1})-(\Sigma_n)_0=0, \qquad y_{M+1}-b_0\tau\frac{db}{dt}(t_{n+1})-(\Sigma_n)_{M+1}=0. \tag{3.6}$$

We are now in a position to apply the smoothing operator S. For convenience, we reproduce the definition of the operator below.

Let the grid be uniform, let D be the difference operator

$$D=\tfrac{1}{4}\begin{bmatrix} 0 & & \cdots & & 0 \\ 1 & -2 & 1 & & \\ 0 & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & 1 & -2 & 1 \\ 0 & & \cdots & & 0 \end{bmatrix}, \tag{3.7}$$

and define the matrices F_j by the recursion

$$F_1=I+D, \qquad F_{j+1}=(I-2F_j)^2 \quad \text{for } j\geq 1. \tag{3.8}$$

Then, the smoothing operator S is defined by the matrix

$$S=\prod_{j=1}^{q} F_j \quad \text{for } q\geq 1. \tag{3.9}$$

The matrices F_j are easily precomputed, so that the application of S requires q matrix–vector multiplications. It can easily be shown that the matrices F_j are essentially tridiagonal matrices. Hence, the application of S does not require much computational effort. In fact, this is due to the special form of the matrix D. In this connection, we remark that D is allowed to be any difference matrix, provided that it has its eigenvalues in the interval $[-1, 0]$ and is such that for any smooth grid function y we have $Dy \to 0$ as the grid is refined.

An important property of the smoothing matrix S is the fact that, once the difference matrix D has been chosen, it does not depend on the particular problem to be integrated.

Furthermore, we note that $S = I + O(D)$ as the grid is refined. Hence, if Δ is the mesh size, then

$$S = I + O(\Delta^2) \quad \text{as } \Delta \to 0. \tag{3.10}$$

Finally, we actually computed the matrices F_j based on the difference matrix D as defined in (3.7) and determined the factorized operator S (cf. (3.9)) for arbitrary values of q. For convenience of the reader, we give a FORTRAN 77 subroutine which performs this operator S (the statements between the dashed rules are comment lines).

3.2.2 *Two-Dimensional Problems* In the case of two-dimensional problems we proceed as in the preceding subsection and we define the system (1.3) in such a way that the boundary equations are analogous to (3.6), just by using the time derivatives of the Dirichlet-boundary conditions.

The smoothing matrix S can be defined by (3.8) and (3.9) if D is replaced by the two-dimensional analogue of (3.7). However, the precomputation of the matrices F_j is not as easy as in the one-dimensional case, and more importantly, it depends on the geometry of the domain on which the problem is defined.

A simple modification of the smoothing operator overcomes this difficulty: let the residue occurring in (2.1a) be arranged as a two-dimensional array in the natural way; then we first smooth all the rows of this array by applying the one-dimensional smoothing matrix defined in Section 3.2.1, and next we smooth all the columns of the resulting array again by this one-dimensional smoothing matrix. In this way, the two-dimensional smoothing process is broken down into a sequence of problem-independent one-dimensional smoothing operators.

The analysis given in van der Houwen *et al.* (1988) shows that this modified smoothing process reduces the spectral radius of the matrix $\hat{Z}$ by an extra factor $\approx 0{\cdot}6$ when compared with the unmodified smoothing process. The reason for this is that now each component is smoothed twice, once in one spatial direction and once in the other spatial direction.

3.2.3 *Non-Dirichlet Conditions* In the preceding discussions we have explicitly stipulated that the boundary conditions are of Dirichlet type. In the case of non-Dirichlet conditions, the time derivatives of the boundary values are not explicitly available. However, if we are able to generate these time derivatives in a stable way, then the SGPC method so far described can be applied without any change. The generation of stable time derivatives of boundary values in the case of Neumann-type boundary conditions is the subject of future investigation by the present authors.

```
      SUBROUTINE SMOOTH(M,Q,QAPPL,U,V)
      REAL U(Ø:M+1),V(1:M)
      INTEGER M,Q,QAPPL,QMAX
-----------------------------------------------------------------------
    THIS ROUTINE PERFORMS THE SMOOTHING OPERATOR S IN ITS FACTORIZED
    FORM (CF. (3.7),(3.8) AND (3.9)).

    M - THE NUMBER OF INTERNAL GRIDPOINTS.
    U - VECTOR OF LENGTH (Ø:M+1), I.E. INCLUDING THE BOUNDARY POINTS.
        ON INPUT, U SHOULD CONTAIN THE VECTOR TO BE SMOOTHED;
        ON RETURN, U CONTAINS THE SMOOTHED RESULT VECTOR.
        ALTHOUGH THE BOUNDARY VALUES U(Ø) AND U(M+1) ARE UNCHANGED
        ON EXIT, THEY ARE USED IN THE SMOOTHING OPERATOR (CF. THE
        MATRIX D IN (3.7)). THIS CORRESPONDS TO THE CASE OF
        DIRICHLET BOUNDARY CONDITIONS.
    V - AN AUXILIARY VECTOR OF LENGTH M TO STORE INTERMEDIATE RESULTS.
    Q - THE NUMBER OF FACTORS IN THE OPERATOR S, I.E. THE REQUIRED
        NUMBER OF SMOOTHING FACTORS.
  QAPPL - OUTPUT PARAMETER; QAPPL IS THE NUMBER OF SMOOTHING
        FACTORS ACTUALLY APPLIED. SINCE THIS NUMBER SHOULD SATISFY
        2**QAPPL .LE. M+1, IT IS SET TO MIN(Q,QMAX), WITH
        QMAX=LOG(M+1)/LOG(2). HENCE, IN SPECIFYING A VALUE OF Q
        WHICH IS TOO LARGE WITH RESPECT TO THE NUMBER OF GRIDPOINTS,
        QAPPL WILL BE SMALLER THAN Q.
        IN CASE THE INPUT PARAMETERS Q AND M ARE SUCH THAT THE
        RELATION 2**Q=M+1 IS EXACTLY SATISFIED, THEN QMAX=Q AND A
        PROVISION IS MADE TO PREVENT THAT THE INTEGER QMAX IS GIVEN
        THE VALUE Q-1, DUE TO (MACHINE DEPENDENT) INACCURACIES IN THE
        LOGARITHMIC FUNCTION.

    RDOFF IS SET TO THE MACHINE ROUNDOFF AND MAY REQUIRE AMENDMENT ON
    DIFFERENT MACHINES.
-----------------------------------------------------------------------
      DATA RDOFF/Ø.71E-14/
      QMAX=LOG(M+1.Ø)/LOG(2.Ø)+1.ØE3*RDOFF
      QAPPL=MIN(Q,QMAX)

      DO 7Ø J=1,QAPPL
         L=2**(J-1)-1
         DO 1Ø I=1,M
   1Ø       V(I)=2.Ø*U(I)
         DO 2Ø I=L+1,M
   2Ø       V(I)=V(I)+U(I-L-1)
         DO 3Ø I=1,M-L
   3Ø       V(I)=V(I)+U(I+L+1)
         DO 4Ø I=1,L
   4Ø       V(I)=V(I)+2.Ø*U(Ø)-U(L+1-I)
         DO 5Ø I=M+1-L,M
   5Ø       V(I)=V(I)+2.Ø*U(M+1)-U(2*M+1-L-I)
         DO 6Ø I=1,M
   6Ø       U(I)=V(I)/4.Ø
   7Ø    CONTINUE
      RETURN
      END
```

3.3 *The Real Stability Boundary of* SGPC *Methods*

We shall derive an approximation to the real stability boundary of SGPC methods for the model situation where the difference matrix D is not defined by (3.7), but, instead, by

$$D := \frac{1}{R}\frac{\partial f}{\partial y}, \tag{3.7'}$$

where R denotes the spectral radius of $\partial f/\partial y$. Notice that this matrix has its eigenvalues in the interval $[-1, 0]$ (recall that $\partial f/\partial y$ was assumed to have negative eigenvalues); furthermore, $Dy \to 0$ as the grid is refined for any smooth grid function y, while (3.10) is satisfied. We emphasize that the definition of the difference matrix D by (3.7′) is just to obtain theoretical results. In actual applications, however, a (repeated) calculation of the Jacobian matrix $\partial f/\partial y$ is rather expensive and therefore we shall always use the matrix D as defined in (3.7). As a consequence, the resulting smoothing operator becomes problem independent and its factors F_j (cf. (3.9)) have a nice structure allowing for an efficient implementation (see the subroutine SMOOTH). A special advantage of this approach is offered when we are dealing with *systems* of parabolic equations because then (3.7) can be applied to each (discretized) equation individually.

However, the crucial point is of course, whether the model situation (3.7′) is of any value for the practical situation where we use (3.7). If the system (1.1) represents the standard symmetric three-point discretization of the diffusion equation $u_t = u_{xx}$, then both matrices coincide. However, for general nonlinear problems the matrix D defined by (3.7) may be quite a poor approximation to the matrix $\partial f/(R\,\partial y)$. In Section 5, we shall show by means of both linear and nonlinear examples that the results obtained for the model situation apply surprisingly well to nonmodel problems.

The main tool in deriving the real stability boundary is the following theorem (a proof can be given along the lines of the proof of Lemma 3.2 in van der Houwen *et al.* (1986)):

THEOREM 3.2 *Let $k = 2^q - 1$, then the matrix S defined by* (3.8) *and* (3.9) *is given by*

$$S := \frac{T_{k+1}(I + 2D) - I}{2(k+1)^2 D}, \qquad T_l(x) := \cos(l \arccos x).$$

We observe that the numerator of this expression contains the factor D so that S is actually a polynomial of degree k in D.

Consider the test equation (3.1), i.e. $\partial f/\partial y = J$ and $D = J/R$. Substitution of the resulting matrix S into (2.3) expresses the matrices Z and $\hat{Z}$ in terms of the fixed matrix J. From this the following relation between the eigenvalues z and $\hat{z}$ of Z and $\hat{Z}$ is immediate:

$$\hat{z} = \hat{z}(z) := \frac{1}{b_0}\left\{1 + \frac{\tau R}{2(k+1)^2}\left(b_0 - \frac{1}{z}\right)\left[T_{k+1}\left(1 + \frac{2z}{\tau R}\right) - 1\right]\right\}. \tag{3.11}$$

By means of this relation we can proceed with the stability criterion (3.4). Suppose that we know the range of values assumed by the function $P_m(\hat{z}(z))$ for $-\tau R \leq z \leq 0$, i.e. the values of

$$z^*_{\min} := \min_{\mathcal{J}} P_m(\hat{z}), \qquad z^*_{\max} := \max_{\mathcal{J}} P_m(\hat{z}), \tag{3.12a}$$

where $\mathcal{J}$ is the interval of $\hat{z}$-values assumed by $\hat{z}(z)$ on the interval $[-\tau R, 0]$.

Then the stability condition (3.4) is certainly satisfied if

$$(z, z^*_{\min}), (z, z^*_{\max}) \in \mathcal{D} \quad \text{for } -\tau R \leqslant z \leqslant 0. \tag{3.12b}$$

In order to find the interval $\mathcal{I}$ we first observe that $\hat{z}(z) \leqslant 1/b_0$ for all $z \in [-\tau R, 0]$. Furthermore, the minimal value of $\hat{z}(z)$ will be assumed at a point in the neighbourhood of $z = 0$ where $T_{k+1}(1 + 2z/\tau R)$ is negative. It is easily shown that this point lies in the interval $[z_0, 0]$, where

$$z_0 := \tfrac{1}{2}\tau R\left[\cos\left(\frac{\pi}{k+1}\right) - 1\right]. \tag{3.12c}$$

Thus,

$$\mathcal{I} = \left[\min_{z_0 \leqslant z \leqslant 0} \hat{z}(z), \frac{1}{b_0}\right]. \tag{3.12d}$$

Before proceeding with the stability criterion (3.12), it should be observed that $P_m(1/b_0) = 1$, so that $z^*_{\max} \geqslant 1$ and consequently, the stability domain should at least contain the line segment $\{(z, 1)\colon -\tau R \leqslant z \leqslant 0\}$ in order to let (3.12) be true. In the following we will concentrate on cases where $\mathcal{D}$ contains this line segment (we remark that the domains shown in Fig. 3.1 satisfy this assumption).

We will now derive explicit expressions for the maximum value of τR for which the SGPC method is stable in the sense of (3.12). This value is called the *real stability boundary* of the method.

THEOREM 3.3 *Let the predictor–corrector pair be such that $\mathcal{D}$ contains the rectangle $[-\tau R, 0] \times [0, 1]$, and let $P_m(x)$ be given by*

$$P_m(x) = P_1(x) = 1 + a\left(x - \frac{1}{b_0}\right) \quad \text{for } 0 \leqslant a \leqslant b_0.$$

Then the SGPC method possesses a real stability boundary

$$\beta(k) \geqslant \frac{k(k+2)}{3b_0} = \frac{4^q - 1}{3b_0} \tag{3.13}$$

for all $a \in [0, b_0]$.

Proof. First we observe that for small values of z the function $\hat{z}(z)$ behaves as

$$\hat{z}(z) \approx \left(1 - \frac{k(k+2)}{3b_0\tau R}\right)z + O(z^2),$$

so that $\hat{z}(z)$ is positive in a left neighbourhood of $z = 0$ if

$$\tau R < \frac{k(k+2)}{3b_0}. \tag{3.14}$$

It can be shown that $\hat{z}(z)$ is positive for all $z \in (-\tau R, 0)$ if this inequality is satisfied. Hence, the interval $\mathcal{I}$ defined in (3.12d) is contained in $[0, 1/b_0]$. From the definition of $P_1(z)$ and from (3.12a) it then follows that $z^*_{\min} \geqslant 0$ and $z^*_{\max} = 1$, so that the condition of the theorem on $\mathcal{D}$ implies that (3.12b) is satisfied. Thus, (3.13) is a sufficient condition for stability. This leads us to the given lower bound on β. □

There seems to be no advantage in choosing a other than b_0 which leads us to the smoothed version of the classical predictor–corrector method in PECE mode:

$$y_{n+1} = y^{(0)}_{n+1} - S(y^{(0)}_{n+1} - b_0 \tau f^{(0)}_{n+1} - \Sigma_n). \tag{3.15}$$

Its order follows from Corollary 2.1 with $r = 1$ and, since $\tau = O(\Delta^2)$, $s = 1$.

Although the stability boundary of (3.15) can be made arbitrarily large by increasing q (cf. (3.13)), we lose accuracy if q becomes too large with respect to the grid. In fact, one should never choose q greater than $\log_2 M$ where M is the number of internal grid points along a row or column of the grid (see the tables of results in Section 5).

In order to preserve stability and accuracy we have to perform more than a single iteration. Adopting the iteration polynomials derived in van der Houwen & Sommeijer (1983), we arrive at the following theorem:

THEOREM 3.4 *Let the predictor–corrector pair be such that $\mathcal{D}$ contains the rectangle $[-\tau R, 0] \times [-d, 1]$, $d > 0$ and let $P_m(x)$ be given by*

$$P_m(x) = \tfrac{1}{2}\left[1 - d + (1 + d)T_m\left(w_0 + \frac{w_0 + 1}{\beta_m}x\right)\right], \tag{3.16}$$

where

$$w_0 := \cos\left(\frac{1}{m}\arccos\frac{d-1}{d+1}\right), \qquad \beta_m := \frac{1}{b_0}\frac{1 + w_0}{1 - w_0}.$$

Furthermore, let

$$\hat{z}_{\min}(k, \tau R) := \min_{z_0 \le z \le 0} \hat{z}(z). \tag{3.17a}$$

Then the SGPC *method possesses a real stability boundary $\beta_m(k)$ which satisfies the inequality*

$$\hat{z}_{\min}(k, \beta_m(k)) > -\beta_m. \tag{3.17b}$$

Proof. The polynomial $P_m(x)$ is chosen such that it is bounded by $-d$ and 1 in the interval $[-\beta_m, 0]$ and by 0 and 1 in the interval $[0, 1/b_0]$. Thus, $P_m(\hat{z})$ assumes values in the interval $[-d, 1]$ if $\hat{z} \in [-\beta_m, 1/b_0]$. Hence, condition (3.12b) is satisfied if $\mathcal{J} \subset [-\beta_m, 1/b_0]$. From (3.12d) it then follows that (3.17b) should be satisfied. □

In Table 3.1 we have listed the stability boundaries $\beta_m(k) = \beta_m(2^q - 1)$ for the $EP_1 - BD_2$ method (cf. Fig. 3.1).

It is possible to give a fairly accurate approximation to $\beta_m(k)$ directly in terms of m and k. This approximation is based on the estimate

$$\hat{z}_{\min} \approx \hat{z}(z_0)$$

instead of (3.17a). Requiring $\hat{z}(z_0) > -\beta_m$, we find

$$\tau R < \tilde{\beta}_m(k) := \left(\beta_m + \frac{1}{b_0}\right)(k+1)^2 - 2\Big/\left[b_0\left(1 - \cos\frac{\pi}{k+1}\right)\right]. \tag{3.18}$$

Thus, the true stability boundary $\beta_m(k)$ is approximated by $\tilde{\beta}_m(k)$. Notice that $\beta_m(0) = \tilde{\beta}_m(0) = \beta_m$ which is precisely the stability boundary of the GPC method

TABLE 3.1
True stability boundaries $\beta_m(k)$ $(=\beta_m(2^q-1))$ of the EP_1–BD_2 *method*

m	$q=0$	$q=1$	$q=2$	$q=3$	$q=4$	$q=5$	$q=6$
1	0·5	4·5	19·7	80·1	322·1	1289·7	5160·5
2	4·5	20·9	85·3	342·8	1372·5	5491·7	21968·3
3	11·3	48·2	194·7	780·5	3123·4	12495·0	49981·5
4	20·9	86·5	347·9	1393·5	5574·5	22299·6	89200·1
5	33·2	135·8	544·9	2181·1	8726·0	34905·6	139623·9
6	48·2	196·0	785·6	3144·1	12577·9	50312·9	201253·2
7	66·0	267·1	1070·1	4282·1	17130·0	68521·6	274088·1
8	86·0	349·2	1398·4	5595·3	22382·5	89531·6	358128·0
9	109·8	442·2	1770·5	7083·4	28335·3	113342·8	453372·1
10	135·8	546·1	2182·3	8746·7	34988·5	139955·4	559823·1
20	546·1	2187·6	8752·0	35009·4	140039·1	560157·9	2240633·2
50	3418·6	13677·6	54711·8	218848·4	875395·0	3501581·3	14006326·6
100	13677·4	54713·3	218853·9	875416·3	3501666·2	14006665·7	56026663·5
$m\to\infty$	$1{\cdot}37m^2$	$5{\cdot}47m^2$	$21{\cdot}8m^2$	$87{\cdot}5m^2$	$350m^2$	$1400m^2$	$5600m^2$

without smoothing. For $k>0$ the approximation is fairly accurate, especially for large values of m. In the case of the EP_1–BD_2 method, this can be easily verified from the true stability boundaries $\beta_m(k)$ listed in Table 3.1. Moreover, given a value of R (determined by the parabolic problem) and a value of τ (determined by accuracy considerations only), we can immediately find from this table a suitable pair (m, q) such that $\tau R<\beta_m(k)$. For small m-values, however, (3.18) slightly overestimates the true boundaries. By taking the integer part of β_m, instead of β_m itself, we found that (3.18) yields a safe value, for all k and m.

We conclude our discussion of stability boundaries of SGPC methods with the observation that for large m and k we have

$$\beta_m(k)\approx\beta_m 4^q\approx\frac{4m^2 4^q}{b_0\{\arccos[(d-1)/(d+1)]\}^2},$$

where $k+1=2^q$. In the case of the EP_1–BD_2 method ($d=\frac{1}{3}$, $b_0=\frac{2}{3}$), this yields

$$\beta_m(k)\approx 1{\cdot}37m^2 4^q, \qquad k+1=2^q.$$

From the numbers listed in Table 3.1 we observe that the true stability boundary is indeed of the form $c(m, q)m^2 4^q$ (cf. (1.5)), where $c(m, q)$ quickly approaches the value 1·37 as m and q increase.

4. The smoothed EP_1–BD_2 method

In this section we give a detailed specification of the SGPC method based on the EP_1 predictor $\{\tilde\rho, \tilde\sigma\}=\{(\zeta-1)^2, 0\}$, the BD_2 corrector $\{\rho, \sigma\}=\{\frac{1}{3}(3\zeta^2-4\zeta+1), \frac{2}{3}\zeta^2\}$, and the iteration polynomial (3.16), where $d=\frac{1}{3}$ and $b_0=\frac{2}{3}$. Following the implementational details given in van der Houwen & Sommeijer

(1983), we obtain the following scheme:

$$\begin{gathered}
y_{n+1}^{(0)} = 2y_n - y_{n-1}; \\
\text{if} \quad m = 1 \quad \text{then} \quad y_{n+1} = y_{n+1}^{(0)} - SR_{n+1}^{(0)}; \\
\text{if} \quad m \geqslant 2 \quad \text{then} \quad y_{n+1}^{(1)} = y_{n+1}^{(0)} - (1 - w_0)SR_{n+1}^{(0)}, \\
y_{n+1}^{(j)} = 2y_{n+1}^{(j-1)} - y_{n+1}^{(j-2)} - 2(1 - w_0)SR_{n+1}^{(j-1)} \quad (j = 2, \dots, m-1), \\
y_{n+1} = \tfrac{1}{3}y_{n+1}^{(0)} - \tfrac{2}{3}y_{n+1}^{(m-2)} + \tfrac{4}{3}y_{n+1}^{(m-1)} - \tfrac{4}{3}(1 - w_0)SR_{n+1}^{(m-1)}.
\end{gathered} \tag{4.1}$$

Here,

$$w_0 := \cos\left(\frac{1}{m}\arccos\left(-\tfrac{1}{2}\right)\right), \qquad R_{n+1}^{(j)} := y_{n+1}^{(j)} - \tfrac{2}{3}\tau f(t_{n+1}, y_{n+1}^{(j)}) - \tfrac{4}{3}y_n + \tfrac{1}{3}y_{n-1}. \tag{4.2}$$

The matrix S is discussed in Section 3.2.

The smoothed EP_1–BD_2 method is, according to Corollary 2.1, second-order accurate in time, because $p = 2$, $\tilde{p} = 1$, $r = 1$, and $s = 1$. A sufficient condition for stability is given by

$$\tau R < \left(\tfrac{3}{2} + \left[\frac{3(1 + w_0)}{2(1 - w_0)}\right]\right)(k + 1)^2 - 3 \Big/ \left[1 - \cos\left(\frac{\pi}{k + 1}\right)\right]. \tag{4.3}$$

We recall that the scheme defined by (4.1) and (4.2) is designed for parabolic problems and will not be suitable for integrating partial differential equations containing hyperbolic terms. In fact, its stability region has the form of a long narrow strip along the negative axis.

5. Numerical experiments

In this section we present a number of 1-D and 2-D initial-boundary value problems by which the smoothed predictor–corrector method will be compared with standard methods. A specification of these methods will be given in the next subsections.

The 1-D problems are defined on the unit interval and the 2-D problems on the unit square; both types are semidiscretized on a uniform spacegrid with meshes of size Δx, using symmetric second-order differences. The Dirichlet boundary conditions are treated as described in Section 3.2.1.

For the time-integration, we used a timestep equal to the meshwidth, i.e. $\tau = \Delta x$. The integration interval is $[0, 1]$ in all experiments. The initial conditions, as well as the starting values needed in a multistep method, are taken from the exact solution.

To measure the accuracy obtained by the various schemes, we define

$$cd := -\log_{10} \|\text{global error in the endpoint } t = 1\|_\infty, \tag{5.1}$$

where the global error is the difference between the numerical solution of the ODE (1.1) and the exact solution of the initial boundary-value problem restricted to the gridpoints. The value of cd can be considered as the number of correct digits in the numerical solution.

5.1 *One-Dimensional Problems*

To start with, we will test four one-dimensional problems. The specification of these problems, as well as the results of the various methods, can be found in Tables 5.1–5.4. To these problems we applied the EP_1–$BD_2(q)$ method for several values of q. In Section 4, this family of methods is completely defined.

TABLE 5.1

N/cd-values for $u_t = e^u u_{xx} + u(9e^u - 1)$ *with exact solution* $u(t, x) = e^{-t}\sin(3x)$ *and* $R_{\max} = e[4(\Delta x)^{-2} - 27]$

Method	$\Delta x = \frac{1}{8}$	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$	$\Delta x = \frac{1}{64}$
EP_1–$BD_2(0)$	50/1·5	149/2·1	429/2·7	1218/3·3
(1)	27/1·5	79/2·1	222/2·7	625/3·3
(2)	14/1·6	45/2·1	120/2·7	332/3·3
(3)	8/1·7	30/2·2	63/2·7	189/3·3
(4)		15/1·7	33/3·2	126/3·4
(5)			31/1·9	63/3·1
(6)				63/2·1
BD_2	7/1·5	15/2·1	31/2·7	63/3·3

TABLE 5.2

N/cd-values for $u_t = u_{xx} + 3xt^2(x^2 - 2t)$ *with exact solution* $u(t, x) = 1 + x^3t^3$ *and* $R_{\max} = 4(\Delta x)^{-2}$

Method	$\Delta x = \frac{1}{8}$	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$	$\Delta x = \frac{1}{64}$
EP_1–$BD_2(0)$	35/1·5	105/2·1	310/2·6	882/3·2
(1)	21/1·6	60/2·1	155/2·6	441/3·2
(2)	14/1·6	30/2·2	93/2·7	252/3·3
(3)	7/1·1	15/1·9	62/2·6	126/3·3
(4)		15/1·2	31/2·1	63/2·9
(5)			31/1·2	63/2·2
(6)				63/1·3
BD_2	7/1·6	15/2·2	31/2·7	63/3·3

TABLE 5.3

N/cd-values for $u_t = u^4u_{xx} - u - 20x^3e^{-t}u^4$ *with exact solution* $u(t, x) = x^5e^{-t}$ *and* $R_{\max} = 4(\Delta x)^{-2} + 1$

Method	$\Delta x = \frac{1}{8}$	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$	$\Delta x = \frac{1}{64}$
EP_1–$BD_2(0)$	22/2·6	55/3·1	147/3·7	409/4·3
(1)	12/2·3	30/3·1	81/3·7	223/4·3
(2)	8/1·6	20/2·5	49/3·2	125/4·0
(3)	7/1·1	15/1·7	34/2·6	81/3·4
(4)		15/1·2	31/1·8	63/2·7
(5)			31/1·2	63/2·0
(6)				63/1·3
BD_2	7/2·6	15/3·1	31/3·7	63/4·3

TABLE 5.4
N/cd-values for $u_t = e^u u_{xx} + u(x - t^2 e^u)$ with exact solution $(u, x) = e^{tx}$ and $R_{\max} = e^e[4(\Delta x)^{-2} + 1]$

Method	$\Delta x = \frac{1}{8}$	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$	$\Delta x = \frac{1}{64}$
EP_1–BD_2(0)	87/1·9	256/1·9	744/2·5	2129/3·1
(1)	46/2·0	132/2·0	380/2·4	1084/3·1
(2)	25/1·5	70/2·2	199/2·4	556/3·2
(3)	15/1·6	38/2·5	110/3·0	296/3·2
(4)		23/1·6	66/2·5	161/3·4
(5)			36/1·6	96/2·5
(6)				63/1·6
BD_2	7/2·1	15/2·7	31/3·3	63/3·9

The only free parameter is m, the number of stages. This parameter is chosen such that the stability condition (3.17) (or (4.3)) is satisfied. Based on Gerschgorin's theorem, we provided an analytical expression yielding a safe upper bound for the spectral radius R. In the nonlinear problems, this expression has been evaluated in each integration step. To the specification of the various problems we have added the maximal value of R that can be expected during the integration process. This maximum value is denoted by $R_{\max}$.

In the tables of results, we only list the number of f-evaluations N (summed over all timesteps), as this is the major part of the computational work; this number is followed by the value of cd (cf. (5.1)).

To judge the merits of this EP–BD method, we also implemented the fully implicit BDF_2, i.e. we directly solved the corrector of the above SGPC method, using Newton's method. In the Tables 5.1–5.4 this method is denoted by BD_2. Again, we list the values of N/cd, where now N denotes the number of Newton iterations (performed in the whole integration process). For our test examples, it turned out that the accuracy furnished by the BD_2 method could not be improved by taking more than one Newton iteration; therefore, the given value of N corresponds to one Newton iteration per step.

Comparing both types of method, we see that the BD_2 method is slightly more accurate than the EP_1–BD_2 method for the same value of N. However, taking into account that one Newton iteration in the BD_2 method involves an f-evaluation, an evaluation and decomposition of the Jacobian matrix and the solution of a tridiagonal system, we think both types of method are at least competitive for one-dimensional problems.

Finally, we observe that taking q as large as allowed by the grid causes a drop in the accuracy. In this connection, it should be remarked that if one were only interested in the steady state and not in time-accurate solutions, then using large q-values would result in a fast convergence to the steady state. This is just the technique investigated in van der Houwen *et al.* (1988) for solving nonlinear elliptic difference equations by smoothed Jacobi-type methods.

5.2 *Two-Dimensional Problems*

Next, we will test some two-dimensional problems. Increasing the dimension of the initial boundary-value problem has hardly any consequences for the application of the SGPC method. Only the smoothing operator S has to be adapted, which can be performed in a straightforward way (see the discussion in Section 3.2.2).

If, on the other hand, a fully implicit scheme is applied (e.g. the BDF$_2$) we are faced with a huge algebraic problem, since now the Jacobian matrix has no longer a tridiagonal structure. Therefore, as an alternative to the BDF$_2$, we selected, as a reference method, the second-order ADI method which is defined by

$$\begin{aligned} y^* &= y_n + \tfrac{1}{2}\tau f_1(t_n + \tfrac{1}{2}\tau, y^*) + \tfrac{1}{2}\tau f_2(t_n, y_n), \\ y_{n+1} &= y^* + \tfrac{1}{2}\tau f_1(t_n + \tfrac{1}{2}\tau, y^*) + \tfrac{1}{2}\tau f_2(t_{n+1}, y_{n+1}). \end{aligned} \tag{5.2}$$

Here, f_1 and f_2 contain the discretizations of the spatial derivatives in the x_1- and x_2-direction respectively. The inhomogeneous term in the initial boundary-value problem is equally distributed over both implicit relations in (5.2). We remark that (5.2) may be considered as a 'standard' version of the ADI method for nonlinear problems. We did not consider special variants which may improve the efficiency of the ADI method in special situations (see, for example, Beam & Warming and Briley & McDonald).

In the tables of results, this method is abbreviated as ADI(m), where m denotes the number of Newton iterations used in each of the implicit relations.

We applied the EP$_1$–BD$_2$(q) method and the ADI(m) method to three two-dimensional problems. Tables 5.5–5.7 contain the resulting N/cd-values. Note that, for the ADI method, N means the total number of Newton iterations summed over all steps and both stages in (5.2). For the nonlinear examples, it turned out that two Newton iterations are sufficient to solve the implicit relations in (5.2).

Comparison of both types of method in terms of CPU time in an actual implementation reveals that the ADI method is superior to the EP$_1$–BD$_2$ method in linear situations, but considerably less efficient for nonlinear problems. As in

TABLE 5.5
N/cd-values for $u_t = u_{x_1x_1} + u_{x_2x_2} + 3t^2[x_1^3 + x_2^3 - 2t(x_1 + x_2)]$ with exact solution $u(t, x_1, x_2) = 1 + t^3(x_1^3 + x_2^3)$ and $R_{\max} = 8(\Delta x)^{-2}$

Method	$\Delta x = \frac{1}{8}$	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$
EP$_1$–BD$_2$(0)	49/1·2	150/1·8	434/2·3
(1)	28/1·3	75/1·7	217/2·3
(2)	14/1·3	45/1·9	124/2·4
(3)	7/0·8	30/1·6	62/2·3
(4)		15/0·9	31/1·7
(5)			31/1·1
ADI(1)	14/1·9	30/2·3	62/2·8

TABLE 5.6
N/cd-values for $u_t = e^u(u_{x_1x_1} + u_{x_2x_2}) + u(9e^u - 1)$ *with exact solution* $u(t, x_1, x_2) = e^{-t}[\sin(3x_1) + \sin(3x_2)]$ *and* $R_{\max} = e[8(\Delta x)^{-2} - 27]$

Method	$\Delta x = \frac{1}{8}$	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$
EP_1–BD_2(0)	95/2·4	286/2·9	826/3·7
(1)	50/2·4	147/3·0	420/3·7
(2)	26/2·5	76/3·1	220/3·7
(3)	15/1·8	42/2·8	116/3·6
(4)		27/1·9	67/2·9
(5)			37/2·0
ADI(1)	14/1·9	unstable	unstable
ADI(2)	28/1·9	60/2·5	124/3·1

TABLE 5.7
N/cd-values for $u_t = u^3_{x_1x_1} + u^3_{x_2x_2} + x_1x_2u - 9t^2(x_1^2 + x_2^2)u^3$ *with exact solution* $u(t, x_1, x_2) = e^{tx_1x_2}$ *and* $R_{\max} = 3e^2[8(\Delta x)^{-2} + 18]$

Method	$\Delta x = \frac{1}{8}$	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$
EP_1–BD_2(0)	144/1·1	436/1·6	1274/1·9
(1)	73/1·2	221/1·4	645/1·8
(2)	38/1·7	115/1·6	330/1·7
(3)	21/1·2	62/1·9	173/2·3
(4)		35/1·1	93/1·8
(5)			54/1·1
ADI(1)	14/1·2	unstable	unstable
ADI(2)	28/1·4	60/1·6	124/2·0

the case of the one-dimensional problems, we see again an abrupt decrease of the accuracy if the largest possible value of q is used.

6. Conclusion

Explicit algorithms have been described for the efficient solution of parabolic initial boundary-value problems with Dirichlet boundary conditions. These methods are based on predictor–corrector type schemes and extended with residue smoothing. For a set of test problems, this technique turns out to be at least competitive with implicit methods.

A decisive advantage of the new methods is their extremely simple implementation.

REFERENCES

BEAM, R. M., & WARMING, R. F. 1976 An implicit finite-difference algorithm for hyperbolic systems in conservation law form. *J. Comp. Phys.* **22**, 87–110.

BRILEY, W. R., & MCDONALD, H. 1975 Solution of the three-dimensional compressible Navier–Stokes equations by an implicit technique. In: *Proceedings of the 4th Intern.*

Conf. on Numer. Math. in Fluid Dynamics, Lecture Notes in Physics **35,** Springer-Verlag, 105–110.

van der Houwen, P. J., & Sommeijer, B. P. 1983 Predictor–corrector methods with improved absolute stability regions. *IMA J. Num. Anal.* **3,** 417–437.

van der Houwen, P. J., Boon, C., & Wubs, F. W. 1988 Analysis of smoothing matrices for the preconditioning of elliptic difference equations. *ZAMM* **68,** 3–10.

van der Houwen, P. J., Sommeijer, B. P., & Wubs, F. W. 1986 Analysis of smoothing operators in the solution of partial differential equations by explicit difference schemes. Report NM-R8617, Centrum voor Wiskunde en Informatica, Amsterdam (to appear in *Appl. Numer. Math.*).

Jameson, A. 1983 The evolution of computational methods in aerodynamics. *J. Appl. Mech.* **50,** 1052–1076.

Jameson, A. & Mavriplis, D. 1985 Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh. *AIAA paper* 85-0435.

Lerat, A. 1979 Une class de schémas aux differences implicites pour les systèmes hyperboliques de lois de conservation. *C. R. Acad. Sc. Paris,* t. 288, Série A, 1033–1036.

Swanson, R. C. & Turkel, E. 1986 Pseudo-time algorithms for the Navier–Stokes equations. *Appl. Numer. Math.* **2,** 321–333.

Turkel, E. 1985 Acceleration to a steady state for the Euler equations. In: *Numerical methods for the Euler equations of fluid dynamics* (F. Angrand, A. Dervieux, J. A. Desideri and R. Glowinski, eds). SIAM, Philadelphia, 281–311.

Wubs, F. W. 1986 Stabilization of explicit methods for hyperbolic partial differential equations. *Int. J. Numer. Methods Fluids* **6,** 641–657.