JOURNAL OF COMPUTATIONAL PHYSICS 51, 1–27 (1983)

Review Article

Stability Restrictions on Time-Stepsize for Numerical Integration of First-Order Partial Differential Equations*

Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM

National Agency of Environmental Protection, Air Pollution Laboratory, Risø National Laboratory, DK-4000 Roskilde, Denmark

Received November 13, 1981; revised April 22, 1982

Mathematical models described by partial differential equations appear often when different phenomena related to atmospheric transport (for example, advection and diffusion processes) are studied. Truncated Fourier series can be used in the discretization of the space derivatives resulting in a pseudospectral method. In this way the partial differential equation is transformed into a system of ordinary differential equations. This system is normally solved by so-called "step-by-step" integration methods. The stability properties of these methods are studied. Some bounds for the time-stepsize are derived. Several formulae and predictor–corrector schemes with large stability intervals on the imaginary axis are constructed. Numerical experiments with eight time-integration algorithms are carried out. Some recommendations concerning the choice of the time-integration algorithms in three different situations are given. It is emphasized that in a package for the solution of partial differential equations describing transport processes in the atmospheric environment one should have several time-integration algorithms, each of which can be chosen optionally. Finally, the possibility of using variable stepsize, variable formula time-integration algorithms is briefly discussed.

1. THE PSEUDOSPECTRAL (FOURIER) DISCRETIZATION OF THE SPACE DERIVATIVES

Consider

$$\frac{\partial c}{\partial t} = u(x, y, t)\frac{\partial c}{\partial x} + v(x, y, t)\frac{\partial c}{\partial y}, \qquad x \in [0, 2\pi], \quad y \in [0, 2\pi], \quad t \in [0, T], \tag{1.1}$$

$$c(x, y, 0) = f(x, y) \qquad \text{for} \quad \forall x \in [0, \pi] \wedge \forall y \in [0, 2\pi] \qquad (f \text{ being given}), \tag{1.2}$$

$$c(0, y, t) = c(2\pi, y, t) \qquad \text{for} \quad \forall y \in [0, 2\pi] \wedge \forall t \in [0, T], \tag{1.3}$$

$$c(x, 0, t) = c(x, 2\pi, t) \qquad \text{for} \quad \forall x \in [0, 2\pi] \wedge \forall t \in [0, T]. \tag{1.4}$$

* 1980 Mathematics Subject Classification: 65L07, 76C15.

0021-9991/83 $3.00

Introduce grids X_M, Y_N and a matrix $G(t)$ as follows:

$$X_M = \left\{ x_m / x_0 = 0, x_{m+1} = x_m + \Delta x, \Delta x = \frac{2\pi}{2M+1}, m = O(1)\, 2M, x_{2M+1} = 2\pi \right\}, \tag{1.5}$$

$$Y_N = \left\{ y_n / y_0 = 0, y_{n+1} = y_n + \Delta y, \Delta y = \frac{2\pi}{2N+1}, n = O(1)\, 2N, y_{2N+1} = 2\pi \right\}, \tag{1.6}$$

$$G(t) = \begin{bmatrix} c(x_0, y_0, t) & c(x_1, y_0, t) & \cdots & c(x_{2M}, y_0, t) \\ c(x_0, y_1, t) & c(x_1, y_1, t) & \cdots & c(x_{2M}, y_1, t) \\ \vdots & \vdots & & \vdots \\ c(x_0, y_{2N}, t) & c(x_1, y_{2N}, t) & \cdots & c(x_{2M}, y_{2N}, t) \end{bmatrix}. \tag{1.7}$$

The trigonometric polynomials given below can be constructed by the use of X_M, Y_N and the elements of $G(t)$; see, for example, [11]

$$T_{M,n}(x, t) = A_n(t) + \sum_{k=1}^{M} [a_{k,n}(t)\cos(kx) + b_{k,n}(t)\sin(kx)], \qquad n = O(1)\, 2N, \tag{1.8}$$

$$T^*_{N,m}(y, t) = A^*_m(t) + \sum_{k=1}^{N} [a^*_{k,m}(t)\cos(ky) + b^*_{k,m}(t)\sin(ky)], \qquad m = O(1)\, 2M. \tag{1.9}$$

The coefficients of these trigonometric polynomials can be calculated by the use of the following formulae (for $x_m \in X_M \wedge y_n \in Y_N$):

$$A_n(t) = \frac{1}{2M+1} \sum_{m=0}^{2M} c(x_m, y_n, t), \qquad n = O(1)\, 2N, \tag{1.10}$$

$$a_{k,n}(t) = \frac{2}{2M+1} \sum_{m=0}^{2M} c(x_m, y_n, t)\cos(kx_m), \qquad n = O(1)\, 2N, \tag{1.11}$$

$$b_{k,n}(t) = \frac{2}{2M+1} \sum_{m=0}^{2M} c(x_m, y_n, t)\sin(kx_m), \qquad n = O(1)\, 2N, \tag{1.12}$$

$$A^*_m(t) = \frac{1}{2N+1} \sum_{n=0}^{2N} c(x_m, y_n, t), \qquad m = O(1)\, 2M, \tag{1.13}$$

$$a^*_{k,m}(t) = \frac{2}{2N+1} \sum_{n=0}^{2N} c(x_m, y_n, t)\cos(ky_n), \qquad m = O(1)\, 2M, \tag{1.14}$$

$$b^*_{k,m}(t) = \frac{2}{2N+1} \sum_{n=0}^{2N} c(x_m, y_n, t)\sin(ky_n), \qquad m = O(1)\, 2M. \tag{1.15}$$

Let r_i be the vector formed by the elements of the ith row of matrix $G(t)$ and c_j be the vector formed by the elements of the jth column of matrix $G(t)$. This means that

$$r_i = \{c(x_0, y_{i-1}, t), c(x_1, y_{i-1}, t), \ldots, c(x_{2M}, y_{i-1}, t)\}, \qquad i = 1(1)\,2N+1, \tag{1.16}$$

$$c_j = \{c(x_{j-1}, y_0, t), c(x_{j-1}, y_1, t), \ldots, c(x_{j-1}, y_{2N}, t)\}, \qquad j = 1(1)\,2M+1. \tag{1.17}$$

It is clear that some approximations $\hat{r}_i$ and $\hat{c}_i$ to the vectors

$$\left\{\frac{\partial c(x_0, y_{i-1}, t)}{\partial x}, \frac{\partial c(x_1, y_{i-1}, t)}{\partial x}, \ldots, \frac{\partial c(x_{2M}, y_{i-1}, t)}{\partial x}\right\}, \quad i = 1(1)\,2N+1, \tag{1.18}$$

$$\left\{\frac{\partial c(x_{j-1}, y_0, t)}{\partial y}, \frac{\partial c(x_{j-1}, y_1, t)}{\partial y}, \ldots, \frac{\partial c(x_{j-1}, y_{2N}, t)}{\partial y}\right\}, \quad j = 1(1)\,2M+1, \tag{1.19}$$

respectively, can be computed by the use of (i) the elements of matrix $G(t)$ as input information, (ii) equalities (1.8)–(1.17), and (iii) the derivatives of the trigonometric polynomials (1.8), (1.9). Some FFT (fast Fourier transform) algorithm must be applied in order to increase the efficiency of the computations (see [4, 7, 13, 25, 33]).

It is easy to see that there exists a $(2M+1)\times(2M+1)$ matrix S_M and a $(2N+1)\times(2N+1)$ matrix S_N such that

$$\hat{r}_i = S_M r_i \ \ (i = 1(1)\,2M+1) \quad \text{and} \quad \hat{c}_j = S_N c_j \ \ (j = 1(1)\,2N+1). \tag{1.20}$$

Let

$$g(t) = \{r_1, r_2, \ldots, r_{2M+1}\} \qquad \text{and} \qquad \tilde{g}(t) = \{c_1, c_2, \ldots, c_{2N+1}\}. \tag{1.21}$$

Since $\hat{r}_i$ and $\hat{c}_j$ are approximations of $\partial c/\partial x$ and $\partial c/\partial y$ respectively on the points of grid $X_M \times Y_N$, it is clear that the partial differential equation (1.1) can be approximated (on the points of grid $X_M \times Y_N$) by a linear system of ordinary differential equations (ODE's)

$$\frac{dg}{dt} = (US + VP\bar{S}P)\,g, \tag{1.22}$$

where g is an abbreviation of $g(t)$, S and $\bar{S}$ are block diagonal matrices (S contains $2N+1$ diagonal blocks equal to matrix S_M, while $\bar{S}$ contains $2M+1$ diagonal blocks equal to matrix S_N), and U and V are diagonal matrices, the diagonal elements of which are values of functions u and v on the points of grid $X_M \times Y_N$.

Observe that vector $g(t)$ is formed by the elements of matrix $G(t)$ when these are ordered by rows, while $\tilde{g}(t)$ is formed by the elements of matrix $G(t)$ when these are ordered by columns. This shows that there exists a permutation matrix P such that

$$\tilde{g}(t) = Pg(t) \qquad \text{and} \qquad P^{-1} = P. \tag{1.23}$$

Let $\tilde{z}(t) = \bar{S}\tilde{g}(t)$. By (1.23) it follows that we have to use vector

$$z(t) = P^{-1}\tilde{z}(t) = P\bar{S}\tilde{g}(t) \tag{1.24}$$

in order to get correspondence with the coordinates of vector Sg.

Relations (1.23) and (1.24) are also used in the derivation of (1.22).

The method used in the transformation of the partial differential equation (1.1) into the system of ODE,s (1.22) is well known as the pseudospectral (Fourier) method ([3, 6, 13, 18, 25, 33, 36]). Three remarks are necessary in connection with this transformation.

Remark 1.1. The assumption $x \in [0, 2\pi]$ is not a restriction. If x belongs to an arbitrary interval $[a, b] \in \mathbb{R}$, then the interval $[a, b]$ can be transformed to $[0, 2\pi]$ by the substitution $\xi = 2\pi(x-a)/(b-a)$. The same is true for the interval of variable y. ∎

Remark 1.2. If the equation

$$\frac{\partial c}{\partial t} = u(x, y, t)\frac{\partial c}{\partial x} + v(x, y, t)\frac{\partial c}{\partial y} + Q(x, y, t) \tag{1.25}$$

is to be solved instead of (1.1), then an extra term (a vector whose elements are the values of $Q(x, y, t)$ on the points of grid $X_M \times Y_N$) has to be added in the right-hand side of the system of ODE,s (1.22). ∎

Remark 1.3. Denote $R^* = (US + VP\bar{S}P)g$. In the actual computations with the Fourier method, one normally uses vector R^*. Thus, the matrices S, P, and $\bar{S}$ are neither calculated explicitly nor stored in the calculations. However, these matrices are needed in the stability analysis. ∎

Assume that (1.1) is transformed into (1.22). Some time-integration algorithm has to be applied in order to obtain an approximate solution of (1.22). *The problem of choosing the time-integration algorithm will be discussed in the next sections.*

2. Stability Restrictions on the Time Stepsize

Consider the test-equation

$$\frac{\partial c}{\partial t} = u\frac{\partial c}{\partial x}, \qquad u \text{ being a constant.} \tag{2.1}$$

In this special case (when u is a constant), the system of ODE's (1.22) can be rewritten as

$$\frac{dg}{dt} = uSg. \tag{2.2}$$

Consider the grid

$$T_K = \{t_k/t_0 = 0, t_{k+1} = t_k + \Delta t, \Delta t = T/K, k = 0(1)K - 1, K \in \mathbb{N}\}. \tag{2.3}$$

The time-increment Δt will be called time-stepsize (or shorter, stepsize).

Numerical time-integration algorithms of *explicit* type are normally used in order to calculate some approximations g_k $(k = 1(1)K)$ to the exact solution vectors $g(t_k)$ of system (2.2). Assume that all approximations up to some g_{k-1} have already been found. Then an explicit forward time-integration algorithm for calculating the next approximation g_k is given by a formula of the following type:

$$g_k = \sum_{i=1}^{s} \alpha_i g_{k-i} + \Delta t \sum_{i=1}^{s} \beta_i f_{k-i}, \qquad k = s(1)K, \quad f_j = uSg_j \ (j = 0(1)K). \tag{2.4}$$

If $s = 1$, then the algorithm is self-starting; otherwise some starting procedure is needed in the calculation of $g_1, g_2, \ldots, g_{s-1}$. Sometimes the starting procedure is used in the calculation of more than the $s - 1$ starting vectors. Moreover, the starting procedure *may* be used (and in fact *is* often used) with some $\Delta t_1 \neq \Delta t$.

Formula (2.4) is called the *general linear multistep method* (see e.g., [11, 14, 19, 27, 40]). Very often the following three polynomials:

$$\rho(\xi) = \xi^s - \sum_{i=1}^{s} \alpha_i \xi^{s-i}, \qquad \sigma(\xi) = \sum_{i=1}^{s} \beta_i \xi^{s-i}, \qquad \pi(\xi, \bar{h}) = \rho(\xi) + \bar{h}\sigma(\xi), \tag{2.5}$$

are associated with the general linear multistep method (2.4).

The following three definitions will be useful in the further considerations.

DEFINITION 2.1. It is said that an arbitrary polynomial $P(z)$ satisfies the *root criterion* if all roots of $P(z)$ lie on the unit disk (i.e., $|z| \leqslant 1 \wedge z \in \mathbb{C}$) and that if there are roots on the unit circle ($|z| = 1 \wedge z \in \mathbb{C}$), then these roots are simple. ∎

DEFINITION 2.2. Let $\bar{h}$ be an arbitrary complex number whose real part is nonpositive (i.e., $\bar{h} \in \mathbb{C} \wedge \mathrm{Re}(\bar{h}) \leqslant 0$). The linear multistep method (2.4) is said to be *absolutely stable for* $\bar{h}$ if the third of its associated polynomials $\pi(\xi, \bar{h})$ satisfies the root criterion. ∎

DEFINITION 2.3. The set S^* of all $\bar{h}$ with $\bar{h} \in \mathbb{C} \wedge \mathrm{Re}(\bar{h}) \leqslant 0$ for which (2.4) is absolutely stable is called the *absolute stability region* for the linear multistep method. ∎

The concept of absolute stability of a numerical method for integration of systems of ODE's has been introduced by Dahlquist [10]. The importance of this concept can

be illustrated as follows. Let $u = -1$. Denote by λ_i $(i = 1(1)\,2M + 1)$ the eigenvalues of matrix S in (2.2). Consider

$$\lambda = \max(|\lambda_i|) \quad (\text{for } i = 1(1)\,2M + 1,\ \mathrm{Re}(\lambda_i) \leqslant 0) \qquad \text{and} \qquad \bar{h} = \lambda\,\Delta t. \tag{2.6}$$

The linear multistep method will produce stable results when it is applied in the solution of system (2.2) with $u = -1$ only if Δt is chosen so that $\bar{h}$ is inside the absolute stability region S^*.

The above analysis shows that both the eigenvalues of matrix S (these depend on the space discretization chosen) and the absolute stability region (it depends on the linear multistep method which is selected) impose some restrictions on the choice of the time-stepsize Δt. Our purpose in this paper is: *to use some specific properties of the special problem under consideration (system* (2.2)*) to derive some stability bounds for Δt when explicit time-integration algorithms are applied in the solution of the system of* ODE's. First, some information about the eigenvalues of matrix S is needed. The following result is a simple modification for our needs of a result obtained by Kreiss and Oliger [25]; see also Fornberg [13].

LEMMA 2.1. *Matrix S is antisymmetric and its eigenvalues are the elements of the set* $\Lambda^* = \{\lambda_j = (-M + j)i / j = 0(1)\,2M,\ i^2 = -1\}$. ∎

Lemma 2.2 (given below) is a simple corollary from Theorem 2.10.2 in Lancaster [28].

LEMMA 2.2. *An antisymmetric matrix S is orthogonally similar to the diagonal matrix of its eigenvalues, i.e., there exists an orthogonal matrix H such that $S = H\Lambda H^T$, where Λ is a diagonal matrix whose diagonal elements are the eigenvalues of matrix S.* ∎

A stability bound for the time-stepsize Δt in the numerical integration of (2.2) can be obtained using Lemmas 2.1 and 2.2.

THEOREM 2.1. *Let* (i) h_{imag} *be the length of the interval on the positive part of the imaginary axis (containing the origin) which belongs to the absolute stability region S^* of the linear multistep method* (2.4) *and* (ii) *the number of points in the space grid* (1.5) *be* $2M + 2$. *Then the application of* (2.4) *in the solution of* (2.2) *with any constant $u \neq 0$ leads to stable computations if Δt is chosen so that*

$$|u|\,M\,\Delta t < h_{\mathrm{imag}}. \tag{2.7}$$

Proof. Use (i) the fact that the eigenvalues of S are given by Lemma 2.1, (ii) the equality $S = H\Lambda H^T$, and (iii) the substitution $H^T g = z$ to split the system (2.2) into $2M + 1$ independent ODE's

$$\frac{dz_\mu}{dt} = iu(-M + \mu)\,z_\mu, \qquad \mu = O(1)\,2M, \quad i^2 = -1. \tag{2.8}$$

The computations in the solution of any equation (2.8) by (2.4) will be stable [10] if $|u|\,|-M+\mu|\,\Delta t < h_{\text{imag}}$. Global stability for the whole set (2.8) will obviously be achieved by choosing Δt so that (2.7) is satisfied because $\lambda = \max(|-M+\mu|) = M$ (for $\mu = O(1)\,2M$). ∎

COROLLARY 2.1. *Let*

(i) *x belong to an arbitrary interval* $[a, b]$ *(instead of* $[0, 2\pi]$*),*

(ii) *the number of points in the space grid* (1.5) *be* $2M + 2$,

(iii) h_{imag} *be as in Theorem* 2.1.

Then the stability bound (2.7) *has to be modified to*

$$\Delta t = \frac{h_{\text{imag}}}{|u|\,M}\,\frac{(b-a)}{2\pi}. \quad ∎ \tag{2.9}$$

The number of points in the space grid (1.5) can be odd; i.e., $x_m = (m-1)\,\Delta x$, $m = O(1)\,2M$. If this is so, then [13] the set of eigenvalues is $\Lambda^* = \{\lambda_j = (-M+1+j)i/(j = O(1)\,2M-2, i^2 = -1), \lambda_{2M-1} = 0\}$, i.e., 0 is a double eigenvalue. In this situation the following corollary of Theorem 2.1 holds.

COROLLARY 2.2. *Let*

(i) $x \in [a, b]$,

(ii) *the number of points in the space grid* (1.5) *be* $2M + 1$ $(M > 1)$, *and*

(iii) h_{imag} *be as in Theorem* 2.1.

Then the stability bound for Δt *is*

$$\Delta t < \frac{h_{\text{imag}}}{|u|\,(M-1)}\,\frac{b-a}{2\pi}. \quad ∎ \tag{2.10}$$

The time-stepsize Δt is used in the numerical integration of (2.2) by some formula of type (2.4). However, (2.2) is found after some discretization of the space derivative in (2.1) on the points of some grid (1.5). Therefore, it is necessary to relate the time-stepsize Δt to the space increment Δx. In the case where the space grid contains $2M + 1$ points, the relation $\Delta x = (b-a)/2M$ holds and the following stability bound is easily obtainable.

COROLLARY 2.3. *The stability bound for* Δt *is related to the choice of* Δx *by*

$$\Delta t < \left[\frac{h_{\text{imag}}}{|u|\,\pi}\,\frac{M}{M-1}\right]\Delta x \tag{2.11}$$

when conditions (i)–(iii) *in Corollary* 2.2 *are satisfied.* ∎

The use of a single formula of type (2.4) has been assumed until now. The so-called predictor–corrector schemes can also be used. In the next part of this section we shall describe how the above results can be extended to a *special class* of predictor–corrector schemes, the PECE schemes. A PECE scheme is defined by the following formulae:

$$g_k^* = \sum_{i=1}^{s} \alpha_i^* g_{k-i} + \Delta t \sum_{i=1}^{s} \beta_i^* f_{k-i} \qquad (\textit{prediction}), \tag{2.12}$$

$$f_k^* = uSg_k^* \qquad (\textit{evaluation}), \tag{2.13}$$

$$g_k = \sum_{i=1}^{s} \alpha_i g_{k-i} + \Delta t\, \beta_0 f_k^* + \Delta t \sum_{i=1}^{s} \beta_i f_{k-i} \qquad (\textit{correction}), \tag{2.14}$$

$$f_k = uSg_k \qquad (\textit{evaluation}). \tag{2.15}$$

The following five polynomials are associated with any PECE scheme:

$$\rho^*(\xi) = \xi^s - \sum_{i=1}^{s} \alpha_i^* \xi^{s-i}, \qquad \sigma^*(\xi) = \sum_{i=1}^{s} \beta_i^* \xi^{s-i}, \tag{2.16}$$

$$\rho(\xi) = \xi^s - \sum_{i=1}^{s} \alpha_i \xi^{s-i}, \qquad \sigma(\xi) = \sum_{i=0}^{s} \beta_i \xi^{s-i}, \tag{2.17}$$

$$\pi_{\mathrm{PECE}}(\xi, \bar{h}) = \rho(\xi) - \bar{h}\sigma(\xi) + \bar{h}\beta_0[\rho^*(\xi) - \bar{h}\sigma^*(\xi)]. \tag{2.18}$$

It is clear now that all results from the first part of this section (which hold in the case where a single formula (2.4) is used) can be extended for the PECE schemes if the absolute stability region of polynomial (2.18) is considered instead of the absolute stability region of polynomial $\pi(\xi, \bar{h})$ from (2.5).

Consider now the partial differential equation (t as in (1.1))

$$\frac{\partial c}{\partial t} = u(x, y, t)\frac{\partial c}{\partial x} + v(x, y, t)\frac{\partial c}{\partial y} + Q(x, y, t), \qquad x \in [a_1, b_1], \quad y \in [a_2, b_2]. \tag{2.19}$$

Assume that the numbers of grid-points are odd in both the x and y directions ($2M + 1$ and $2N + 1$, respectively). Denote

$$u^* = \max_{x \in [a_1, b_1], y \in [a_2, b_2], t \in [0, T]} (|u(x, y, t)|),$$

$$v^* = \max_{x \in [a_1, b_1], y \in [a_2, b_2], t \in [0, T]} (|v(x, y, t)|). \tag{2.20}$$

Then one *should expect* the computations to be stable if

$$\Delta t < \frac{h_{\text{imag}}}{\pi}\left[\frac{u^*(M-1)}{M\,\Delta x}+\frac{v^*(N-1)}{N\,\Delta y}\right]^{-1}. \tag{2.21}$$

If $M = N$ and $\Delta x = \Delta y$, then (2.21) reduces to

$$\Delta t < \left[\frac{h_{\text{imag}}}{(u^*+v^*)\pi}\,\frac{M}{M-1}\right]\Delta x. \tag{2.22}$$

Remark 2.1. The ratio $\Delta t/\Delta x$ is often used in the stability criteria. To the authors' knowledge, the first stability criterion containing $\Delta t/\Delta x$ was proposed by Courant, Friedrichs, and Lewy [8]. If the stability bound has to be extended to two-dimensional space regions, then it may be better to express the stability criterion only as a bound for Δt; see (2.21). ∎

It is seen that the stability bounds given in this section depend on three factors:

(i) on the problem that is to be solved (through u^* and v^*),

(ii) on the discretization of the space derivatives (through M and N), and

(iii) on the time-integration algorithm chosen (through h_{imag}).

The dependence of the stability bounds on the particular time-integration algorithms will be studied in Sections 3 and 4.

3. On the Length of the Stability Interval for Linear Multistep Formulae

Several results concerning the maximal value of h_{imag} for the time-integration algorithms of type (2.4) will be given in this section.

THEOREM 3.1. *Let F be any formula of type* (2.4). *The stability interval h_{imag} of formula F satisfies the following inequality*:

$$0 \leqslant h_{\text{imag}} \leqslant 1. \tag{3.1}$$

Proof. The assertion of Theorem 3.1 can easily be deduced from a more general result (Theorem 5.1 in Jeltsch and Nevanlinna [24, p. 82]). ∎

Remark 3.1. Both bounds in (3.1) can be reached. For the Euler formula (which can be found from (2.4) by choosing $s = 1$, $\alpha_1 = 1$, $\beta_1 = 1$) we have $h_{\text{imag}} = 0$. For the leapfrog (midpoint) rule (which can be found from (2.4) by choosing $s = 2$, $\alpha_1 = 0$, $\alpha_2 = 1$, $\beta_1 = 1$, $\beta_2 = 0$) we have $h_{\text{imag}} = 1$. Therefore the leapfrog rule is *one* of the best formulae (among the formulae of class (2.4)) with regard to the stability requirements. However, the order of this formula is only 2 (see, e.g., [27] for some details concerning the order of a formula (2.4)). If the accuracy requirements dominate over the stability requirements, it may be more profitable to use formulae of

higher order even if $h_{\text{imag}} < 1$ (for example, a third order formula with $h_{\text{imag}} = 0.72$ has been used in [36]). We shall consider the relationship between the accuracy requirements and the stability requirements in the next sections. ∎

Remark 3.2. The result obtained by Theorem 3.1 is entirely different from some results obtained for systems (2.2) which arise from the discretization of parabolic differential equations. In the latter case matrix S is symmetric and has real eigenvalues. This means that the length of the interval on the negative real axis h_{real}, which belongs to S^*, is of interest. Let γ be an arbitrary positive real number. Then a formula of class (2.4), for which $h_{\text{real}} = \gamma$, can be constructed ([21, 24, 30, 46]). It should be emphasized that if γ is very large, then the formulae are very inaccurate [23]. Nevertheless, the above relationship is much less restrictive than the corresponding relationship (3.1) for h_{imag}. ∎

The following two theorems (which are obvious modifications of theorems proved in [24]) give some information about the possibility of constructing a formula of a given order and with a given stability interval h_{imag}.

THEOREM 3.2. *For any* $\gamma \in [0, 1]$ *and for any* $s \in \{2, 3, 4\}$ *an explicit linear multistep formula* (2.4) *of order* $p = s$ *and with* $h_{\text{imag}} = \gamma$ *can be constructed.* ∎

THEOREM 3.3. *If* $\gamma > 0$ *and* $s = 1 \pmod 4$, *then no explicit linear multistep formula* (2.4) *of order* $p = s$ *and with* $h_{\text{imag}} = \gamma$ *exists.* ∎

These theorems tell us that if the accuracy of the leapfrog rule is not sufficient for our problem, then it is possible to construct a formula of higher order (3 or 4 but not 5; see Theorem 3.3) which has nearly the same stability properties as the leapfrog rule. Some special formulae of order 3 are derived in Zlatev and Østerby [50].

The following theorem shows that the use of *implicit* linear multistep formulae will not lead to a large increase of h_{imag} when the order of the formula is larger than 2.

THEOREM 3.4 (Dekker [12]). *The inequality* $0 \leqslant h_{\text{imag}} \leqslant \sqrt{3}$ *holds for any implicit linear multistep formula if its order is higher than* 2. ∎

Remark 3.3. The upper bound in the above inequality is obtainable. An example is the classical Milne–Simpson formula (which is of order 4; see [27]). ∎

Remark 3.4. If the restriction on the order of the formula is removed, then even methods which are stable on the whole imaginary axis (i.e., with $h_{\text{imag}} = \infty$) can be constructed. An example is the well-known trapezoidal rule which is of order 2 and for which $h_{\text{imag}} = \infty$. ∎

Remark 3.5. It is well known that the use of implicit formulae is justified only in the case where this use allows us to specify large time-stepsizes. Theorem 3.4 indicates that if the accuracy requirements are stringent (so that a formula of order higher than 2 must be used), then *explicit formulae have to be used.* Indeed, in this situation the use of implicit formulae leads to a great increase of the computational

work per step. This increase cannot be compensated by the decrease of the number of steps because the small stability bound ($\sqrt{3}$) does not allow us to increase the time-stepsize very much. Therefore, only explicit formulae are discussed in this paper. ∎

4. On the Length of the Stability Intervals for Some Special PECE Schemes

Consider a special PECE scheme defined by the following formulae:

$$g_k^* = \alpha_s^* g_{k-1} + (1-\alpha_s^*)\, g_{k-2} + \Delta t \sum_{i=1}^{s} \beta_{is}^* f_{k-i}, \qquad f_k^* = uSg_k^*, \tag{4.1}$$

$$g_k = \alpha_s\, g_{k-1} + (1-\alpha_s)\, g_{k-2} + \Delta t\, \beta_{0s} f_k^* + \Delta t \sum_{i=1}^{s} \beta_{is} f_{k-i}, \qquad f_k = uSg_k. \tag{4.2}$$

Definition 4.1. Assume that:

(i) The computations are carried out by formulae (4.1), (4.2).

(ii) The parameters α_s^* and α_s can freely be chosen so that $\alpha_s^* \in \mathbb{R}$ and $\alpha_s \in [0, 2)$ for each $s > 1$.

(iii) The coefficients $\{\beta_{is}^*\}$, $i = 1(1)s$, are determined so that (4.1) is an explicit linear multistep formula of order s.

(iv) The coefficients $\{\beta_{is}\}$, $i = O(1)s$, are determined so that (4.2) is an implicit linear multistep formula of order $s+1$.

Then the scheme is called an $(\alpha^*, \alpha)\, \mathrm{P}_s\mathrm{EC}_{s+1}\mathrm{E}$ scheme. ∎

Predictor–corrector schemes similar to those given by Definition 4.1 have been considered in [41–43, 47, 48]. In connection with the stability interval, h_{imag}, we have to solve

Problem 4.1. Let s be an integer larger than 1. Find two numbers α_s^* and α_s which generate an $(\alpha^*, \alpha)\, \mathrm{P}_s\mathrm{EC}_{s+1}\mathrm{E}$ scheme with as large an h_{imag} as possible. ∎

Problem 4.1 was solved numerically for $s \in \{2, 3, 4\}$ using the optimization algorithm from [41]. Let ε be a positive number which can be made arbitrarily small. In the optimization process, $(\alpha^*, \alpha)\, \mathrm{P}_s\mathrm{EC}_{s+1}\mathrm{E}$ schemes with $h_{\mathrm{imag}} = 2 - \varepsilon$ (for $s = 2$) and $h_{\mathrm{imag}} = 1.73 - \varepsilon$ (for $s = 3$) have been obtained. For $s = 4$ no $(\alpha^*, \alpha)\, \mathrm{P}_4\mathrm{EC}_5\mathrm{E}$ scheme with a stability interval *containing the origin* has been found. However, $(\alpha^*, \alpha)\, \mathrm{P}_4\mathrm{EC}_5\mathrm{E}$ schemes, which are stable in the interval $[0.16, 1.28 - \varepsilon]$, can be constructed. We do not know whether it is possible to apply such schemes in practice.

Remark 4.1. A comparison between the best (with regard to the stability requirements) single formulae (2.4) and $(\alpha^*, \alpha)\, \mathrm{P}_s\mathrm{EC}_{s+1}\mathrm{E}$ schemes indicates that one

can expect that the best $(\alpha^*, \alpha)\,P_sEC_{s+1}E$ schemes can be applied with a stepsize which is nearly twice as large as the optimal stepsize for the best linear multistep formulae (2.4). This means that the number of time-steps for the best $(\alpha^*, \alpha)\,P_sEC_{s+1}E$ schemes will be nearly twice as small as the number of time-steps for the best linear multistep formulae (2.4). Unfortunately, it does not follow from this fact that the computational work will be twice as small when the best $(\alpha^*, \alpha)\,P_sEC_{s+1}E$ schemes are used. The problem is that the computational work per step is greater for the $(\alpha^*, \alpha)\,P_sEC_{s+1}E$ schemes (because two formulae are used). Nevertheless, *the use of* $(\alpha^*, \alpha)\,P_sEC_{s+1}E$ *schemes has at least two advantages.*

(i) The values of functions $u(x, y, t)$, $v(x, y, t)$, and $Q(x, y, t)$ on the grid points have to be calculated once per step both when a formula of type (2.4) is used and when an $(\alpha^*, \alpha)\,P_sEC_{s+1}E$ scheme is applied.

(ii) A very cheap estimation of the order of accuracy of the solution of the system of ODE's can be calculated when a predictor–corrector scheme is used. This gives a possibility of building in an automatic check of the accuracy of the results. ∎

5. Choice of Test-Examples

Some numerical experiments will be carried out in Section 7 in order to illustrate the influence of the time-integration algorithm on

(i) the stability of the results,

(ii) the accuracy of the results, and

(iii) the computing time needed to obtain the results.

Such experiments have to be carried out with text-examples which are both simple and representative. A very reliable test-example (which satisfies both requirements) has been proposed by Molenkamp [32] and Crowley [9]. This example (sometimes slightly modified) has been used in many papers; e.g., [1, 5, 6, 13, 29, 33, 34, 35]. In the notation used in this paper, the Molenkamp–Crowley test-example can be formulated as follows.

Problem 5.1. Find some acceptable approximation to the solution of the partial differential equation given by

$$\frac{\partial c}{\partial t} = (1 - y)\frac{\partial c}{\partial x} + (x - 1)\frac{\partial c}{\partial y}, \qquad x \in [0, 2], \quad y \in [0, 2], \quad t \in [0, 2\pi], \tag{5.1}$$

$$x_0 = 0.5, \qquad y_0 = 1.0, \qquad r = 0.25, \qquad \bar{x} = \sqrt{(x - x_0)^2 + (y - y_0)^2}, \tag{5.2}$$

$$\begin{aligned} c(x, y, 0) &= 100(1 - \bar{x}/r) \qquad \text{for} \quad \bar{x} < r, \\ &= 0 \qquad\qquad\qquad\quad \text{for} \quad \bar{x} \geqslant r. \end{aligned} \tag{5.3}$$

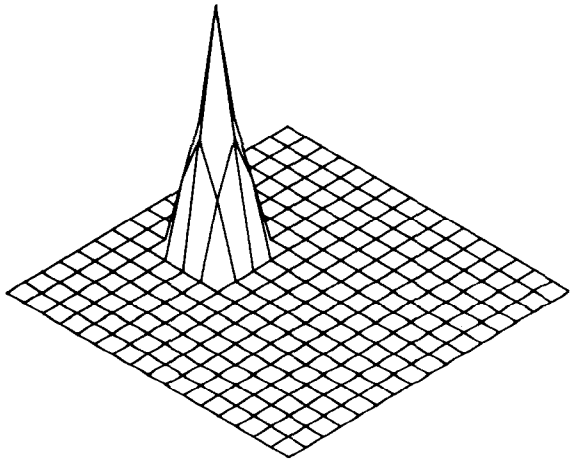

FIG. 1. The solution of Problem 5.1 on a 16×16 grid at $t = 0$ by the $(-0.85, 1.80)$ P_3EC_4E scheme (see Section 6); this means that this is the initial distribution.

Sometimes Problem 5.1 is used in a slightly modified form as follows.

PROBLEM 5.2. Consider Problem 5.1 with (5.3) replaced by

$$\begin{aligned} c(x, y, 0) &= 50[1 + \cos(\pi\bar{x}/r)] \quad \text{for} \quad \bar{x} < r, \\ &= 0 \qquad\qquad\qquad\quad \text{for} \quad \bar{x} \geqslant r. \end{aligned} \tag{5.4}$$

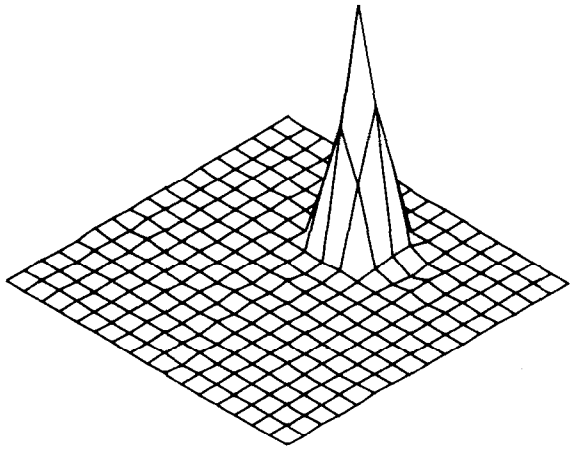

FIG. 2. The computed solution of Problem 5.1 on a 16×16 grid at $t = \pi/2$ by the $(-0.85, 1.80)$ P_3EC_4E scheme (see Section 6); this means that the calculated solution after one quarter of a revolution is given in this figure.

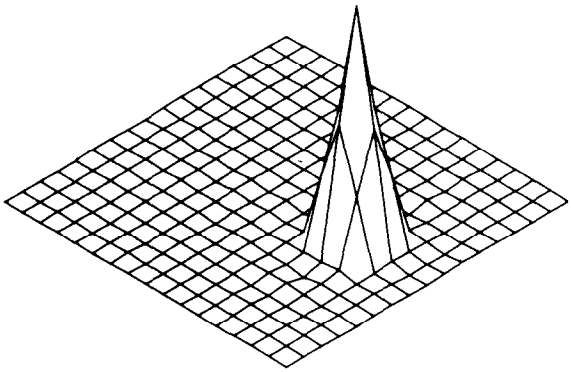

FIG. 3. The computed solution of Problem 5.1 on a 16×16 grid at $t = \pi$ by the (−0.85, 1.80) P_3EC_4E scheme (see Section 6); this means that the calculated solution after one half of a revolution is given in this figure.

We shall show that the original Molenkamp–Crowley test-example (Problem 5.1) is more stringent with regard to the accuracy requirements than its modified form (Problem 5.2, which is often called the cosine hill; see [29]).

It is easy to check the accuracy of the approximations at the endpoint ($t = 2\pi$) of the time-interval because, both for the original test-example (Problem 5.1) and for its modified form (Problem 5.2), the following relationship holds.

$$c(x, y, 2\pi) = c(x, y, 0) \qquad \text{for} \quad \forall x \in [0, 2] \quad \text{and for} \quad \forall x \in [0, 2]. \tag{5.5}$$

More generally, it can easily be seen that Eq. (5.1) describes a rotation of the initial distribution $c(x, y, 0)$ with a constant angular velocity around the axis perpen-

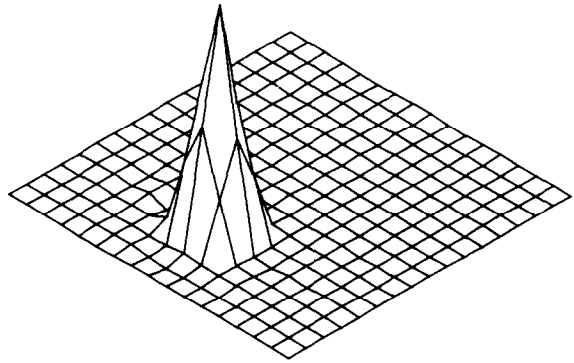

FIG. 4. The computed solution of Problem 5.1 on a 16×16 grid at $t = 3\pi/2$ by the (−0.85, 1.80) P_3EC_4E scheme (see Section 6); this means that the calculated solution after three quarters of a revolution is given in this figure.

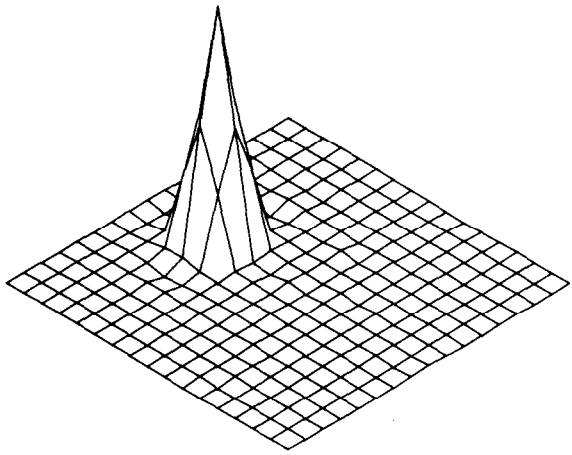

FIG. 5. The computed solution of Problem 5.1 on a 16×16 grid at $t = 2\pi$ by the $(-0.85, 1.80)$ P_3EC_4E scheme (see Section 6); this means that the calculated solution after a revolution is given in this figure; 118 time steps are needed for a full revolution.

dicular to the (x, y) plane and cutting this plane at the point (1.0, 1.0). This is demonstrated in Figs. 1–5.

A third test-example, which is more stringent with regard to accuracy than Problem 5.1, is also needed in our experiments. As such a test-example we will use the following problem.

PROBLEM 5.3. Find some acceptable approximations to the solution of the partial differential equation given by

$$\frac{\partial c}{\partial t} = -\frac{\partial c}{\partial x}, \qquad x \in [0, 2], \quad t \in [0, 1],$$

$$x_0 = 0.5, \quad \sigma = 0.01, \quad c(x, 0) = \frac{1}{\sigma} e^{-(x-x_0)^2/2\sigma^2}. \quad \blacksquare \tag{5.6}$$

It is clear that the exact solution of Problem 5.3 is

$$c(x, t) = \frac{1}{\sigma} e^{-(x-x_0-t)^2/2\sigma^2}. \tag{5.7}$$

Problem 5.3 has been solved with different space discretizations, using grids with $2M = 2^k$ $(k = 2(1)7)$. Some of the numerical results will be given in Section 7.

Some acceptability criteria have to be used when the problems are solved approximately. The acceptability of the results depends on the answer to the following question: Where will the results be used? This means that the acceptability of the results depends on the user's needs and two different users may have different accep-

tability requirements (if the results are to be used for different purposes). However, some acceptability requirements (though sometimes in a very vague form) are always stated and the approximate solution is acceptable if the user has good reasons to expect that these requirements are satisied. The acceptability criterion used in our experiments is defined as follows. Let ($\|\cdot\|$ being the uniform norm of the vector under consideration)

$$T = \frac{\|c_{ij} - c(x_i, y_j, 2\pi)\|}{\|c(x_i, y_j, 2\pi)\|}, \tag{5.8}$$

where c_{ij} and $c(x_i, y_j, 2\pi)$ are, respectively, the computed solution and the exact solution of Problem 5.1 or Problem 5.2 at the grid point (x_i, y_j) $(i = O(1)\, 2M$, $j = O(1)\, 2N)$ for $t = 2\pi$.

In the case where Problem 5.3 is solved let

$$T = \frac{\|c_i - c(x_i, 1)\|}{\|c(x_i, 1)\|}, \tag{5.9}$$

where c_i and $c(x_i, 1)$ are, respectively, the computed solution and the exact solution at the grid-point x_i $(i = O(1)\, 2M)$ for $t = 1.0$.

Acceptability Criterion

The approximate solution obtained by the use of any time-integration algorithm is acceptable if $T \leqslant 0.1$. ∎

It should be noted that the acceptability criterion defined above is easily computable for all test-examples proposed in this section. The requirement for 10% relative accuracy of the approximate solution is often used in different fields in science and engineering.

6. Time-Integration Algorithms Used in the Numerical Experiments

Eight time-integration algorithms have been selected and attached to our package for solving partial differential equations describing different atmospheric phenomena [44, 45]. These time-integration algorithms will briefly be discussed in this section.

Algorithm 1. This algorithm is based on the use of the classical *leapfrog* (midpoint) rule, which is one of the frequently used explicit methods; see, e.g., [31, 39]. This formula can be found from the general formula (2.4) by setting $s = 2$, $\alpha_1 = 0$, $\alpha_2 = 1$, $\beta_1 = 2$, $\beta_2 = 0$. It is easily seen that the formula is of order $p = 2$ and that its stability interval on the imaginary axis is $h_{\text{imag}} = 1$. ∎

Algorithm 2. This algorithm is based on the explicit third order Adams–Bashforth formula [2]. It can be found from the general formula (2.4) by setting

$s = 3$, $\alpha_1 = 1$, $\alpha_2 = 0$, $\alpha_3 = 0$, $\beta_1 = 23/12$, $\beta_2 = -16/12$, $\beta_3 = 5/12$. The order of the formula is $p = 3$ and the length of its stability interval on the imaginary axis is $h_{\text{imag}} = 0.72$. ∎

Algorithm 3. This algorithm is based on the two-parameter family of formulae discussed in [50]. A member of this family can be found from the general formula (2.4) by setting $s = 3$, $\alpha_3 = 1 - \alpha_1 - \alpha_2$, $\beta_1 = 9/4 - \alpha_1/3 - \alpha_2/12$, $\beta_2 = 9/2 - 2\alpha_1 - \alpha_2/2 + 2\beta_1$, $\beta_3 = -3/2 - \alpha_1/2 + \beta_1$. We shall refer to any formula of this family as (α_1, α_2)-formula. The order of any (α_1, α_2)-formula is $p = 3$. The free parameters α_1 and α_2 have been chosen so that the length of the stability interval of the formula is larger than the length of the stability interval of the third order Adams–Bashforth formula. In our experiments we shall use the formula found by $\alpha_1 = 1.0$ and $\alpha_2 = 0.4$. The length of the stability interval for the (1.0, 0.4)-formula is $h_{\text{imag}} = 0.86$. ∎

Algorithm 4. This algorithm is based on the well-known Adams P_2EC_3E scheme. The Adams predictor–corrector schemes can be found from the general class (4.1), (4.2) by setting $\alpha_s^* = \alpha_s = 1$ ($s = 2, 3, \ldots$). These schemes are implemented in many codes (see, e.g., [15, 16, 22, 26, 37]). The length of the stability interval on the imaginary axis for the Adams P_2EC_3E scheme is $h_{\text{imag}} = 1.20$. ∎

Algorithm 5. This algorithm is based on the well-known Adams P_3EC_4E scheme. This means that its coefficients can be computed from (4.1), (4.2) by setting $\alpha_s^* = \alpha_s = 1$ ($s = 3$). The length of the stability interval on the imaginary axis for the Adams P_3EC_4E scheme is $h_{\text{imag}} = 1.18$. ∎

Algorithm 6. This algorithm is based on the (−0.15, 1.00) P_2EC_3E scheme (this means that it can be found from (4.1), (4.2) by setting $\alpha_s^* = -0.15$ and $\alpha_s = 1.00$, $s = 2$). The length of the stability interval on the imaginary axis for the (−0.15, 1.00) P_2EC_3E scheme is $h_{\text{imag}} = 1.61$. ∎

Algorithm 7. This algorithm is based on the (−0.05, 1.85) P_2EC_3E scheme (this means that it can be found from (4.1), (4.2) by setting $\alpha_s^* = -0.05$ and $\alpha_s = 1.85$, $s = 2$). The length of the stability interval on the imaginary axis for the (−0.05, 1.85) P_2EC_3E scheme is $h_{\text{imag}} = 1.95$. ∎

Algorithm 8. This algorithm is based on the (−0.85, 1.80) P_3EC_4E scheme (this means that it can be found from (4.1), (4.2) by setting $\alpha_s^* = -0.85$ and $\alpha_s = 1.80$, $s = 3$). The length of the stability interval on the imaginary axis for the (−0.85, 1.80) P_3EC_4E scheme is $h_{\text{imag}} = 1.70$. ∎

All eight of these time-integration algorithms have been tested in the solution of many test-examples. Some of the results will be presented in Section 7. It is also necessary to emphasize that, while Algorithms 1, 2, 4, and 5 are well known, Algorithms 3, 6, 7, and 8 have been developed by us for systems of ODE's arising after the space discretization of some partial differential equations.

7. Numerical Results

Some results obtained in the solution of Problems 5.1–5.3, using our package which implements the Fourier method in order to discretize the space derivatives and the eight time-integration algorithms given in the previous section, will be discussed below. However, before beginning this discussion it is necessary to mention that:

(i) All experiments have been carried out on an IBM 3033 installation at NEUCC (Northern Europe University Computing Centre, Lyngby, Denmark) with the FORTH compiler (OPT = 2).

(ii) All computing times are measured with the subroutine TIME from the standard library at NEUCC and are given in seconds.

(iii) The explicit first-order Euler method has been used to compute the starting approximations in all time-integration algorithms (the time-integration algorithms are based on formulae which are not self-starting; see Section 2).

The space derivatives in Problem 5.1 have been discretized by the grids $x_i = (i-1)\Delta x$, $y_j = (j-1)\Delta y$, $\Delta x = \Delta y = \frac{1}{8}$, $i = 1(1)17$, $j = 1(1)17$. Then all eight time-integration algorithms have been used to calculate approximations to the solution of Problem 5.1 at $t = 2\pi$. Similar rules have been used in the space discretization of the other test-problems. The experiments show that the following conclusions can be drawn.

(A) For Problem 5.1 with $2M = 2N = 16$ the bound (2.22) is very pessimistic. This should be expected because it is well known that "this example to some extent behaves like a constant coefficient example" ([13, p. 524]; see also the transformation of the coordinates in [32]). However, note that the theoretical limits for Δt (which are computed with $u^* = v^* = 1.0$) *will ensure* stable computations for all eight time-integration algorithms. Moreover, both the theoretical limits and the actual limits (see

TABLE I

Time-Integration Algorithm	Theoretical Limit for Δt	Actual Limit for Δt
Leapfrog (midpoint) rule	0.02274	0.02454
Third order Adams–Bashforth	0.01637	0.02662
(1.0, 0.4)-formula	0.01955	0.02856
Adams P_2EC_3E scheme	0.02728	0.04620
Adams P_3EC_4E scheme	0.02683	0.04245
(−0.15, 1.00) P_2EC_3E scheme	0.03661	0.05712
(−0.05, 1.85) P_2EC_3E scheme	0.04434	0.05417
(−0.85, 1.80) P_3EC_4E scheme	0.03865	0.05325

Note. The theoretical limits are computed by the use of bound (2.22) for Problem 5.1 with $2M = 2N = 16$ and $u^* = v^* = 1.0$. The actual limits are experimentally found taking into account the acceptability criterion in Section 5.

Table I) for the last three schemes are much larger than the corresponding limits for the leapfrog algorithm which is traditionally used in the numerical solution of partial differential equations. ∎

(B) The accuracy requirements limited the stepsize when the leapfrog rule and the $(-0.05, 1.85)\,P_2EC_3E$ scheme were used. In all of the other six time-integration algorithms the stability requirements were the reason for the actual limit of the stepsize in Table I. For example, the $(-0.05, 1.85)\,P_2EC_3E$ scheme can be used with $\Delta t = 0.06042$, but this causes a relative error of 12% at the point $x = 0.5$, $y = 1.125$ (i.e., the point just over the point where the top of the cone is located when $t = 2\pi$). ∎

(C) The fact that the predictor–corrector schemes have been used with larger values of Δt does not mean automatically that the computing times for the predictor–corrector schemes are smaller than those in the cases where single formulae (one of the first three algorithms) are used. The problem is that the computational work per step for the predictor–corrector schemes is larger. Nevertheless, it is seen from Table II that the $(-0.15, 1.00)\,P_2EC_3E$ scheme is the most economical with regard to the computing time used (i.e., this scheme can be applied with so large a Δt that the extra computing time used at each step is fully compensated by the reduction of the number of time-steps). Note too that if the accuracy requirements are relaxed by 2% (i.e., if the relative error tolerance is increased to 12%), then the $(-0.05, 1.85)\,P_2EC_3E$ scheme will be the best one with regard to the computing time. In the latter case the integration process will be completed in 104 steps after 5.09 sec by the $(-0.05, 1.85)\,P_2EC_3E$ scheme. Finally, it should be mentioned that only the predictor–corrector schemes which we have developed for our package (the last three)

TABLE II

Time-Integration Algorithm	Number of Steps	Computing Time	Largest error (percentage)	Largest Value of the Solution at $t = 2\pi$	Smallest Value of the Solution at $t = 2\pi$
Leapfrog (midpoint) rule	256	6.23	9	97	−4
Third order Adams–Bashforth	236	5.83	4	96	−2
(1.0, 0.4)-formula	220	5.60	8	92	−2
Adams P_2EC_3E scheme	136	6.74	7	93	−3
Adams P_3EC_4E scheme	148	7.34	3	97	−2
$(-0.15, 1.00)\,P_2EC_3E$ scheme	110	5.35	8	92	−4
$(-0.05, 1.85)\,P_2EC_3E$ scheme	116	5.76	10	95	−5
$(-0.85, 1.80)\,P_3EC_4E$ scheme	118	5.85	4	96	−2

Note. Some characteristics concerning the approximate solutions of Problem 5.1 (at the end of the time-interval, $t = 2\pi$) found by the eight time-integration algorithms. The acceptability criterion from Section 5 is used.

are competitive with the algorithms based on the use of single formulae (the first three). The classical Adams predictor–corrector schemes (which are commonly used in the solution of ODE's) are clearly more time consuming. ∎

In the solution of Problem 5.1 with $2M = 2N = 16$ the stability requirements are dominant over the accuracy requirements (imposed by the acceptability criterion in Section 5) for six of the time-integration algorithms. For the other two algorithms (leapfrog and the $(-0.05, 1.85)\,P_2EC_3E$ scheme) the accuracy requirements are slightly dominant. However, the general conclusion is that for Problem 5.1 with $2M = 2N = 16$ the stability and the accuracy requirements are balanced in some degree. By this we mean that the actual limit of Δt is close to the maximal value allowed by the stability properties of the time-integration algorithm (except perhaps for the leapfrog whose order is only two). We believe that this is a typical situation in the atmospheric environment where the concentration of the pollutants can change sharply in different parts of the domain, but this is compensated to some degree by the fact that the error tolerance is also normally large. Nevertheless, we have carried out many experiments where no balance between the accuracy requirements and the stability requirements is presented. Some of these experiments will be described below.

Let us begin with the situation where the accuracy requirements are dominant over the stability requirements. Problem 5.3 was used as a text-example where this is true. Some results obtained with $2M = 32$ are given in Tables III and IV. The experiments (in which some other values of M have also been used) allow us to draw the following conclusions.

(D) The actual limits for the stepsize Δt are much smaller than the theoretical limits obtained by the use of stability bound (2.22). The advantage of using more accurate formulae (of order four, i.e., the Adams P_2EC_4E scheme and the $(-0.85, 1.80)\,P_2EC_4E$ scheme) is obvious. It should be mentioned here that some

TABLE III

Time-Integration Algorithm	Theoretical Limit for Δt	Actual Limit for Δt
Leapfrog (midpoint) rule	0.02122	0.0045
Third order Adams–Bashforth	0.01528	0.0056
(1.0, 0.4)-formula	0.01825	0.0049
Adams P_2EC_3E scheme	0.02546	0.0091
Adams P_3EC_4E scheme	0.02504	0.0125
$(-0.15, 1.00)\,P_2EC_3E$ scheme	0.03417	0.0111
$(-0.05, 1.85)\,P_2EC_3E$ scheme	0.04138	0.0101
$(-0.85, 1.80)\,P_3EC_4E$ scheme	0.03608	0.0125

Note. The theoretical limits (for Problem 5.3) are computed by the use of bound (2.22) with $2M = 32$, $u^* = 1$, $v^* = 0$, and $\Delta x = \frac{1}{16}$. The actual limits are experimentally found taking into account the acceptability criterion in Section 5.

TABLE IV

Time-Integration Algorithm	Number of Steps	Computing Time	Largest error (percentage)	Largest Value of the Solution at $t = 1.0$	Smallest Value of the Solution at $t = 1.0$
Leapfrog (midpoint) rule	224	0.31	9.3	95.8	−9.31
Third order Adams–Bashforth	179	0.28	9.2	90.8	−5.86
(1.0, 0.4)-formula	204	0.30	8.9	91.1	−7.24
Adams P_2EC_3E scheme	110	0.25	8.8	91.2	−4.52
Adams P_3EC_4E scheme	80	0.19	8.3	93.1	−2.03
(−0.15, 1.00) P_2EC_3E scheme	90	0.21	9.2	90.8	−5.57
(−0.05, 1.85) P_2EC_3E scheme	99	0.23	9.1	96.0	−7.87
(−0.85, 1.80) P_3EC_4E scheme	80	0.19	6.3	93.7	−2.87

Note. Some characteristics concerning the approximate solutions of Problem 5.3 (at the end of the time interval, $t = 1.0$) found by the eight time-integration algorithms. The acceptability criterion from Section 5 is used.

time-integration schemes are very sensitive to the accuracy requirements. This is especially true for the leapfrog rule. Problem 5.1 has been solved with an accuracy requirement $T \leqslant 0.02$ (which is much more stringent than the acceptability criterion given in Section 5). This reqirement can be satisfied if 1024 steps with the leapfrog rule are carried out (24.73 sec); compare these results with Table II. Both the third order Adams–Bashforth formula and the Adams P_3EC_4E scheme are not very sensitive to the accuracy requirements. In the solution of Problem 5.1 with $T \leqslant 0.02$, 360 (8.97 sec) and 170 (8.37 sec) steps were needed for the third order Adams–Bashforth formula and for the Adams P_3EC_4E scheme respectively; compare these results with those for the leapfrog rule above and with the results in Table II.

Note too that in the general case both the time-step and the distances between the grid-points (i.e., the quantities Δx and/or Δy) have normally to be reduced when the degree of the required accuracy is increased. In all test-examples the use of the Fourier method leads to a high degree of accuracy of the space discretization even if very small values of M and/or N are specified. This is not necessarily true for arbitrary problems but it is convenient for our study because we are only interested in the performance of the time-integration algorithms. ∎

Now let us consider the case where the stability requirements dominate over the accuracy requirements. We shall consider the behaviour of the eight time-integration algorithms in the solution of Problem 5.1 with $2M = 2N = 32$. Some results are given in Tables V and VI. The following conclusions can be drawn from these tables.

(E) The accuracy requirements limited the stepsize only for the leapfrog rule. Nevertheless, the leapfrog rule is the best algorithm with regard to the computing

TABLE V

Time-Integration Algorithm	Theoretical Limit for Δt	Actual Limit for Δt
Leapfrog (midpoint) rule	0.01061	0.01428
Third order Adams–Bashforth	0.007639	0.01122
(1.0, 0.4)-formula	0.009125	0.01298
Adams P_2EC_3E scheme	0.01273	0.01916
Adams P_3EC_4E scheme	0.01252	0.01806
(−0.15, 1.00) P_2EC_3E scheme	0.01708	0.02454
(−0.05, 1.85) P_2EC_3E scheme	0.02069	0.02662
(−0.85, 1.80) P_3EC_4E scheme	0.01804	0.02344

Note. The theoretical limits for Problem 5.1 are computed by the use of bound (2.22) with $2M = 2N = 32$, $u^* = v^* = 1.0$, $\Delta x = \Delta y = \frac{1}{16}$. The actual limits are experimentally found taking into account the acceptability criterion in Section 5.

time used for this problem. The performance of the leapfrog rule (with regard to the computing time used) will be slightly improved if the acceptability criterion $T \leqslant 0.1$ is replaced by $T \leqslant 0.12$. The computations will be completed in 420 step after 34.04 sec in this situation. If Problem 5.2 is run instead of Problem 5.1, then the results obtained with the use of 420 steps (after 34.21 sec) will be acceptable also when a very stringent error tolerance ($T \leqslant 0.02$) is used. Some characteristics obtained in such a run are given in Table VII. The results show clearly that the accuracy requirements are considerably weakened when the initial value condition (5.3) in the original Molenkamp–Crowley test-example (Problem 5.1) is replaced by the initial value condition (5.4) in the modified test-example (Problem 5.2). ∎

TABLE VI

Time-Integration Algorithm	Number of Steps	Computing Time	Largest error (percentage)	Largest Value of the Solution at $t = 2\pi$	Smallest Value of the Solution at $t = 2\pi$
Leapfrog (midpoint) rule	440	37.25	10	93	−5
Third order Adams–Bashforth	560	48.16	4	96	−1
(1.0, 0.4)-formula	484	42.98	6	94	−1
Adams P_2EC_3E scheme	328	54.18	5	95	−1
Adams P_3EC_4E scheme	348	58.01	2	98	0*
(−0.15, 1.00) P_2EC_3E scheme	256	42.97	6	94	−1
(−0.05, 1.85) P_2EC_3E scheme	236	39.31	7	93	−4
(−0.85, 1.80) P_3EC_4E scheme	268	45.04	4	96	0*

Note. Some characteristics concerning the approximate solutions of Problem 5.1 (at the end of the time-interval, $t = 2\pi$) found with $2M = 2N = 32$ and by the eight time-integration algorithms. The acceptability criterion from Section 5 is used. The smallest value of the calculated solution is given as 0 when it is larger than −0.5.

TABLE VII

Number of steps	420
Length of the time-stepsize Δt	0.01496
Computing time	34.21
Largest error at $t = 2\pi$	2%
Largest value of the solution at $t = 2\pi$	99
Smallest value of the solution at $t = 2\pi$	−2

Note. Some characteristics obtained in the solution of Problem 5.2 with $2M = 2N = 32$ with the leapfrog (midpoint) rule.

(F) The third order Adams–Bashforth formula and the Adams predictor–corrector schemes perform very poorly in this situation. This is especialy true for the Adams predictor–corrector methods. However, note that these methods are very accurate. Nevertheless, this example explains why the classical Adams predictor–corrector methods are not popular in the numerical solution of partial differential equations describing advection phenomena (though they are very often used in the numerical solution of systems of ODE's; see, for example, [15, 16, 20, 22, 26, 37, 38, 48]). ∎

(G) The special predictor–corrector methods even in this situation (where the stability requirements are clearly dominant over the accuracy requirements) perform sufficiently well. This is especially true for the $(-0.05, 1.85)\,P_2EC_3E$ scheme. For large problems, where some extra work has to be carried out at each step (reading data, interpolation of values of some functions, etc.), the fact that these methods use considerably less steps will give some extra advantages. We have mentioned this also in Remark 4.1. ∎

8. Some Recommendations about the Choice of a Time-Integration Algorithm

Our experiments, some of which are described in Section 7, show that the following recommendations concerning the choice of a time-integration algorithm can be given.

(i) If the stability requirements are clearly dominant over the accuracy requirements, then the use of the leapfrog rule is the most advantageous (with regard to the computing time needed; Table VI as an illustration of this fact).

(ii) If the accuracy requirements are clearly dominant over the stability requirements, then the time-integration algorithms of high order (in our set of time-integration algorithms, the Adams P_3EC_4E scheme and the $(-0.85, 1.80)\,P_3EC_4E$ scheme) will be the best algorithms with regard to the computing time needed; see Table IV as an illustration of this statement.

(iii) If the accuracy requirements and the stability requirements are in balance, then the time-integration algorithms which are both accurate and with large stability intervals h_{imag} will be the best algorithms. Several such algorithms have been developed and tested in the present study with problems of this type. As an illustration for the good performance of these algorithms, see Table II.

The numerical results show also that it is difficult to propose a general-purpose time-integration algorithm which performs well in cases (i)–(iii). Therefore we have included all eight algorithms in our package. The user of the package is able to choose (optionally) each of these algorithms according to the problem which is to be solved. Of course, there is another alternative: one can try to develop a code where the requirement for constant stepsize Δt is dropped and where the code will automatically change both the stepsize and the time-integration formula in order to achieve both the stability and the accuracy requirements. We shall briefly discuss this possibility in Section 9. Now we shall conclude this section with several remarks.

Remark 8.1. The stability bounds derived in Section 2 cannot be directly applied in the calculation of the stepsize Δt which has to be used in the actual computations. Indeed, if the accuracy requirements are dominant over the stability requirements, then the stepsizes which have to be used in order to get acceptable results are much smaller than the stability bounds derived in Section 2. On the other hand, if the stability requirements are dominant over the accuracy requirements and if the functions $u(x, y, t)$ and $v(x, y, t)$ vary rapidly in x, y, and/or t, then the stability bounds derived in Section 2 will ensure stable calculations. However, these bounds may be rather pessimistic (because u^* and v^* have to be used in the stability restrictions for Δt). Nevertheless, these bounds are very useful because they tell us that time-integration algorithms, which are sufficiently accurate and for which the bounds derived in Section 2 are large, will perform well. ∎

Remark 8.2. We must emphasize once again that one should be very careful when the time-integration algorithms with large stability intervals are constructed. It is not sufficient that the algorithm has a large stability interval; it must also be sufficiently accurate (the lower accuracy of some algorithms with large stability intervals is discussed in [23]). ∎

Remark 8.3. If the storage requirements are an important factor, then the leapfrog rule can be implemented in the most economical way. The Adams predictor–corrector schemes are slightly more economical (with regard to the storage requirements) than the predictor–corrector schemes developed by us (the last

9. Development of Variable Stepsize Variable Formula Methods

All time-integration algorithms are used as *constant stepsize constant formula methods* in this paper. This fact puts some requirements to the user. The user must determine a good (or, if possible, the best) formula and a large (or, if possible, the

largest) time-stepsize by which acceptable results will be calculated. Normally, the user knows his problems sufficiently well to be able to determine both the formula and the stepsize which have to be used in the constant stepsize constant formula time-integration algorithm. In such a case, he can hope to solve his problem in an efficient way (see, e.g., [49]). Nevertheless, the situation might be much better if a reliable rule for automatic determination of a good stepsize and a good formula at each step were incorporated in the code. This idea leads to the so-called VSVFM's (variable stepsize variable formula methods); see [42, 43, 48].

The first code where such a rule has been incorporated was the code developed by Krogh [26]; see also [17]. Many other codes have been developed after 1969, [15, 16, 20, 37, 48]. This idea can also be *attractive* in the atmospheric environments. In order to show this let us consider the case where the functions $u(x, y, t)$ and $v(x, y, t)$ are quickly varying in t. In this case, u^* and v^* computed by (2.20) may impose a very severe (and unnecessary) restriction in the time-stepsize bound (2.22). Therefore it is desirable to use

$$\begin{aligned} \bar{u} &= \max_{x\in[a_1,b_1],\, y\in[a_2,b_2],\, t\in T_k^*} (|u(x, y, t)|), \\ \bar{v} &= \max_{x\in[a_1,b_1],\, y\in[a_2,b_2],\, t\in T_k^*} (|v(x, y, t)|), \end{aligned} \tag{9.1}$$

(T_k^* being a neighbourhood of the current integration point t_k) in the choice of the stepsize and/or formula at the time-step $t = t_k$ because $\bar{u}$ and/or $\bar{v}$ can be much smaller than u^* and/or v^*, respectively. If the latter situation occurs, then the code will possibly select a large stepsize Δt (or that formula which allows the use of as large as possible Δt). Thus, the use of a variable stepsize variable formula method will normally lead to some reduction of the time-steps, and therefore, to some reduction in the total computational work. Moreover, the implementation of a variable stepsize variable formula method will give, as a secondary effect, a rather reliable evaluation of the accuracy of the results.

One should be careful, however. The codes mentioned above will clearly be inefficient in the solution of the special systems of ODE's obtained after the discretization of the space derivatives in (1.1). These codes are general-purpose codes and the formulae used are selected so that the absolute stability regions S^* (see Definition 2.3) of the formulae involved are as large as possible. We are not interested in large stability regions. The stability intervals on the imaginary axis are important for us. Therefore, special formulae (or predictor–corrector schemes) with large stability intervals have to be constructed and used as basic formulae (or basic predictor–corrector schemes) in the variable stepsize variable formula methods which will be suitable for our special systems of ODE's. Note that the Adams predictor–corrector schemes are commonly used in the general-purpose codes. It is obvious that these schemes will be very inefficient just in the case where efficiency is most desired (when the stability requirements are dominant over the accuracy requirements; see the numerical results given in Table VI).

The above analysis shows that the idea of using variable stepsize variable formula methods is attractive, but it is not very easy to implement such an idea in a practical code. One should not just select one of the subroutines already developed because the basic formulae in all known packages are used in order to get as large regions of absolute stability as possible, while we are obviously interested only in the stability intervals on the imaginary axis. Special formulae (or special predictor–corrector schemes) have to be developed and implemented for our special situation. Some such methods have been presented in this paper (Algorithms 3, 6–8). We plan to develop a variable stepsize variable formula time-integration algorithm capable of treating efficiently some advection–diffusion processes arising in the atmospheric environment.

REFERENCES

1. D. ANDERSON AND B. FATTAHI, *J. Atmos. Sci.* **31** (1974), 1500.
2. F. BASHFORTH AND J. C. ADAMS, "Theories of Capillary Actions," Cambridge Univ. Press, London/New York, 1883.
3. R. BERKOWICZ AND L. P. PRAHM, *Atmos. Environ.* **12** (1977), 379.
4. E. O. BRIGHAM, "The Fast Fourier Transform," Prentice–Hall, Englewood Cliffs, N. J., 1974.
5. S. Z. BURNSTEIN AND A. A. MIRIN, *J. Comput. Phys.* **5** (1970), 547.
6. O. CHRISTENSEN AND L. P. PRAHM, *J. Appl. Meteorol.* **15** (1976), 1284.
7. J. W. COOLEY AND J. W. TUKEY, *Math. Comput.* **19** (1965), 297.
8. R. COURANT, K. O. FRIEDRICHS, AND H. LEWY, *Math. Ann.* **100** (1928), 32.
9. W. P. CROWLEY, *Mon. Weather Rev.* **96** (1968), 1.
10. G. DAHLQUIST, *BIT* **3** (1963), 27.
11. G. DAHLQUIST AND Å. BJÖRCK, "Numerical Methods," Prentice–Hall, Englewood Cliffs, N. J., 1974.
12. K. DEKKER, *BIT* **21** (1981), 66.
13. B. FORNBERG, *SIAM J. Numer. Anal.* **12** (1975), 509.
14. C. W. GEAR, "Numerical Initial Value Problems in Ordinary Differential Equations," Prentice–Hall, Englewood Cliffs, N. J., 1971.
15. C. W. GEAR, *Commun. ACM* **14** (1971), 176.
16. C. W. GEAR, *Commun. ACM* **14** (1971), 185.
17. C. W. GEAR, *SIAM Rev.* **23** (1981), 10.
18. D. GOTLIEB AND S. A. ORSZAG, "Numerical Analysis of Spectral Methods: Theory and Applications," SIAM, Philadelphia, 1977.
19. P. HENRICI, "Discrete Variable Methods in Ordinary Differential Equations," Wiley, New York/London/Sydney, 1962.
20. A. C. HINDMARSH, "GEAR: Ordinary Differential Equation Solver," Report UCRL-51186, Rev. 3, Lawrence Livermore Laboratory, Livermore, Calif., 1974.
21. P. J. VAN DER HOUWEN, "Construction of Integration Formulas for Initial Value Problems," North-Holland, Amsterdam, 1977.
22. T. E. HULL, W. H. ENRIGHT, B. M. FELLEN, AND A. E. SEDGWICK, *SIAM J. Numer. Anal.* **9** (1972), 603.
23. R. JELTSCH AND O. NEVANLINNA, "Lower Bounds for the Accuracy of Linear Multistep Methods," Bericht No. 6, Institut für Geometrie und Praktische Mathematik, Rheinisch-Westfälische Technische Hoehschule, Aachen, 1980.
24. R. JELTSCH AND O. NEVANLINNA, *Numer. Math.* **37** (1981), 61.

25. H. O. KREISS AND J. OLIGER, *Tellus* **XXIV** (1972), 199.
26. F. T. KROGH, "VODQ/SVDQ/DVDQ—Variable Order Integrators for Numerical Solution of Ordinary Differential Equations," Report NPO-11643, Jet Propulsion Laboratory, Pasadena, Calif., 1969.
27. J. D. LAMBERT, "Computational Methods in Ordinary Differential Equations," Wiley, London/New York/Sydney, 1973.
28. P. LANCASTER, "Theory of Matrices," Academic Press, New York/London, 1969.
29. P. E. LONG, JR. AND D. W. PEPPER, *J. Appl. Meteorol.* **20** (1981), 146.
30. R. MANNSHARDT, *in* "Numerical Treatment of Differential Equations" (R. Bulirsch, R. D. Grigorieff, and J. Schröder, Eds.), Lecture Notes in Mathematics, Vol. 631, p. 81, Springer-Verlag, Berlin/Heidelberg/New York, 1978.
31. F. MESINGER AND A. ARAKAWA, "Numerical Methods Used in Atmospheric Models," Garp Publications Series No. 17, World Meteorological Organization, International Council of Scientific Unions, 1976.
32. C. R. MOLENKAMP, *J. Appl. Meteorol.* **7** (1968), 160.
33. S. A. ORSZAG, *J. Fluid Mech.* **49** (1971), 75.
34. R. A. PEARSON, *in* "Numerical Simulation in Fluid Motion," (J. Noye, Ed.), p. 261, North-Holland, Amsterdam, 1978.
35. D. W. PEPPER, C. D. KERN, AND P. E. LONG, JR., *Atmos. Environ.* **13** (1979), 223.
36. L. P. PRAHM AND O. CHRISTENSEN, *J. Appl. Meteorol.* **16** (1977), 896.
37. L. F. SHAMPINE AND M. K. GORDON, "Computer Solution of Ordinary Differential Equations: The Initial Value Problem," Freeman, San Francisco, 1975.
38. L. F. SHAMPINE, H. A. WATTS, AND S. M. DAVENPORT, *SIAM Rev.* **18** (1976), 376.
39. G. D. SMITH, "Numerical Solution of Partial Differential Equations," Oxford Univ. Press, London, 1969.
40. H. J. STETTER, "Analysis of Discretization Methods of Ordinary Differential Equations," Springer-Verlag, Berlin/Heidelberg/New York, 1973.
41. P. G. THOMSEN AND Z. ZLATEV, *BIT* **19** (1979), 503.
42. Z. ZLATEV, *Numer. Math.* **31** (1978), 175.
43. Z. ZLATEV, *Numer. Math.* **37** (1981), 157.
44. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, "Numerical Treatment of the Advection–Diffusion Equation: Part I. Space Discretization," Report MST LUFT-A47, Air Pollution Laboratory, National Agency of Environmental Protection, Risø National Laboratory, Roskilde, 1981.
45. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, "Numerical Treatment of the Advection–Diffusion Equation: Part II. Test Problems," Report MST LUFT-A49, Air Pollution Laboratory, National Agency of Environmental Protection, Risø National Laboratory, Roskilde, 1981; published in *Comput. & Fluids* **11** (1983), 13.
46. Z. ZLATEV AND P. G. THOMSEN, *Int. J. Numer. Methods Eng.* **14** (1979), 1051.
47. Z. ZLATEV AND P. G. THOMSEN, *ACM Trans. Math. Software* **5** (1979), 401.
48. Z. ZLATEV AND P. G. THOMSEN, *in* "Méthods Numerérique dans les Sciences de l'Ingenieur-G.A.M.N.I.2" (E. Absi, R. Glowinski, P. Lascoux, and H. Veysseyre, Eds.), p. 221, Dunod, Paris, 1980.
49. Z. ZLATEV, J. WASNIEWSKI, AND K. SCHAUMBURG, *Comput. Chem.* **4** (1980), 13.
50. Z. ZLATEV AND O. ØSTERBY, "Absolute Stability Properties of the Explicit Linear 3-Step Formula," Report PB-124, Computer Science Department, Aarhus University, Ny Munkegade, Aarhus C, Denmark, 1980.