JOURNAL OF COMPUTATIONAL PHYSICS 55, 278–301 (1984)

Implementation of a Variable Stepsize Variable Formula Method in the Time-Integration Part of a Code for Treatment of Long-Range Transport of Air Pollutants

ZAHARI ZLATEV, RUWIM BERKOWICZ, AND LARS P. PRAHM*

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

Received April 13, 1983; revised September 20, 1983

A mathematical model consisting of two partial differential equations is used to study the long-range transport of sulphur di-oxide and sulphate over Europe. The discretization of the first-order space derivatives (*the advection terms*) is carried out by a pseudospectral (Fourier) algorithm. A special technique is applied in the discretization of the second-order space derivatives (*the diffusion terms*). Two large systems of ordinary differential equations are solved at each time-step. It is shown that these systems can efficiently be treated by a variable stepsize variable formula method (VSVFM) based on the use of predictor-corrector schemes. The stepsize selection strategy and the formula selection strategy are discussed in detail. An attempt to carry out both an accuracy control and a stability control is made at each time-step. The great efficiency of the VSVFM implemented in our software as well as the reliability of the results are illustrated by numerical experiments, in which real meteorological data (for 1979) at the grid-points of a space domain covering the whole of Europe were used. The main ideas, implemented in the time-integration part, might be applied in many other situations, where the systems of ordinary differential equations arising after the space discretization are only moderately stiff (so that the stability requirements are dominant over the accuracy requirements on a large part of the time-interval but the use of implicit time-integration algorithms that require solving systems of algebraic equations at each time-step is not justified). As an illustration only it should be mentioned that such an application has been carried out in connection with models describing long-range transport of nitrogen pollutants over Europe.

1. INTRODUCTION

Systems of partial differential equations (PDEs) arise often in models simulating physical phenomena in different fields of science and engineering. Splitting up techniques are commonly used to divide the computational process into several stages. The time derivatives are discretized during one of these stages. It will be shown how the computations during the time-discretization can be fully automized when predictor-corrector schemes are in use. It will be demonstrated that the automization of the time-discretization *may* lead to a considerable increase of the

* Present address: Danish Meteorological Institute, Lyngbyvej 100, DK-2100 Copenhagen Ø, Denmark.

0021-9991/84 $3.00

efficiency. The ideas are fairly general and can be applied in connection with many different models. In order to facilitate the description, models describing long-range transport of sulphur di-oxide (SO_2) and sulphate (SO_4) will be considered. Moreover, some salient features of these models will be exploited to reduce the computational work. However, the programs are well-structured, [46], and can easily be modified for some other processes. As an illustration only, it should be mentioned that the programs have successfully been used to study long-range transport of nitrogen pollutants over Europe.

The mathematical model treated in this paper is defined as

$$\frac{\partial c}{\partial t} = -u\frac{\partial c}{\partial x} - v\frac{\partial c}{\partial y} + K_x\frac{\partial^2 c}{\partial x^2} + K_y\frac{\partial^2 c}{\partial y^2} + \frac{\partial}{\partial z}\left(K_z\frac{\partial c}{\partial z}\right) + Q, \tag{1.1}$$

$$\frac{\partial c^*}{\partial t} = -u\frac{\partial c^*}{\partial x} - v\frac{\partial c^*}{\partial y} + K_x\frac{\partial^2 c^*}{\partial x^2} + K_y\frac{\partial^2 c^*}{\partial y^2} + \frac{\partial}{\partial z}\left(K_z\frac{\partial c^*}{\partial z}\right) + Q^*, \tag{1.2}$$

$$x \in [a_1, b_1], \qquad y \in [a_2, b_2], \qquad z \in [a_3, b_3], \qquad t \in [a, b], \tag{1.3}$$

$$c(a_1, y, z, t) = c(b_1, y, z, t), \qquad c^*(a_1, y, z, t) = c^*(b_1, y, z, t), \tag{1.4}$$

$$c(x, a_2, z, t) = c(x, b_2, z, t), \qquad c^*(x, a_2, z, t) = c^*(x, b_2, z, t), \tag{1.5}$$

$$\frac{\partial c(x, y, a_3, t)}{\partial z} = \frac{\partial c(x, y, b_3, t)}{\partial z} = \frac{\partial c^*(x, y, a_3, t)}{\partial z}$$

$$= \frac{\partial c^*(x, y, b_3, t)}{\partial z} = 0, \tag{1.6}$$

$$c(x, y, z, a) = f(x, y, z), \qquad c^*(x, y, z, a) = f^*(x, y, z),$$

$$f \text{ and } f^* \text{ being given.} \tag{1.7}$$

The quantities involved in the above model are interpreted as: (i) the unknown functions $c(x, y, z, t)$ and $c^*(x, y, z, t)$ are concentrations of SO_2 and SO_4 in the atmosphere, (ii) $u(x, y, z, t)$ and $v(x, y, z, t)$ are wind velocities along the Ox and Oy axes, respectively, (iii) K_x, K_y and K_z are diffusion coefficients, (iv) $Q(x, y, z, t, c)$ and $Q^*(x, y, z, t, c, c^*)$ represent different sources and/or sinks.

The model described by (1.1)–(1.7) is obtained by the use of the following assumptions: (a) K_x and K_y are positive constants, (b) K_z is a non-negative function, which is piecewise constant in z (more precisely, for each (x^*, y^*, t^*) with $x^* \in [a_1, b_1]$, $y^* \in [a_2, b_2]$ and $t^* \in [a, b]$ there exists a non-negative number h, which is called the mixing height, such that $K_z(x^*, y^*, z, t^*) = 0$ if $h \leqslant z \leqslant b_3$ and $K_z(x^*, y^*, z, t^*)$ does not vary in z if $a_3 \leqslant z \leqslant h$) and (c) there is no vertical advection (this means that $w(x, y, z, t) \equiv 0$, where w is the vertical component of the wind velocity). Experience shows that satisfactory results can be achieved under these assumptions when long-range transport of air pollutants is simulated. This will be illustrated by Experiment 4 in Section 5.

A large system of ordinary differential equations (ODEs) is obtained after the

discretization of the space derivatives in (1.1)–(1.2). Predictor-corrector schemes can conveniently be applied in the solution of this system of ODEs at the points of some grid

$$T_k = \{t_k/t_0 = a,\ t_{k-1} < t_k\ (k = 1(1)K),\ t_K = b,\ K \in \mathbb{N}\} \tag{1.8}$$

defined on the time interval $[a, b]$. Assume that a constant time-stepsize, $\Delta t = t_k - t_{k-1} = (b-a)/K$ $(k = 1(1)K)$, is used in the time-integration algorithm selected. Denote

$$\bar{u} = (u, v), \qquad u^* = (\|\bar{u}\|)^{-1}, \qquad \|\cdot\| \text{ being some appropriate norm.} \tag{1.9}$$

If a predictor-corrector scheme is used, then it is necessary to restrict the time-stepsize Δt as follows

$$\Delta t \leqslant du^* \tag{1.10}$$

in the efforts to achieve stable computations. The constant d depends (see [6, 47, 51]): (A) *on the time-integration algorithm* and (B) *on the space discretization.* Assume that both the time-integration algorithm and the space discretization are fixed. Then one can evaluate u^* (by checking the appropriate input data) and determine a value for Δt, using (1.10), that will hopefully give stable results. However, if the time interval $[a, b]$ is long, then the value of Δt so found may lead to a very inefficient computational process. Indeed, two or three stormy days will give a very small value of Δt and the computational process will be very expensive. It must be emphasized that the system of ODEs is normally very large. The number of equations is often larger than 10^4 when the model is three-dimensional. Our experiements with a $32 \times 32 \times 9$ grid in the space domain gave rise to a system of 18,432 ODEs. This shows that the efficiency of the computational process is highly desirable and that the three-dimensional model can be used in practice only if one is able to ensure great efficiency of the calculations.

The main purpose in this paper is to demonstrate that the efficiency can be increased considerably if the old-fashioned manner of implementation of the time-discretization algorithms with a constant Δt is replaced by a modern application of a variable time-stepsize. The usefulness of this approach can be justified as follows. Consider the grid (1.8) and denote by

$$\Delta t_k = t_k - t_{k-1}, \qquad k = 1(1)K, \tag{1.11}$$

the time-stepsize which is to be applied at time-step k. Then (1.10) can be replaced by

$$\Delta t_k \leqslant du_k^*, \tag{1.12}$$

where u_k^* is inverse of the norm of the wind velocity vector $\bar{u}$ in some neighbourhood of the current integration point t_k. It is clear that in general u_k^* will be larger than u^* (and for many grid-points t_k may be considerably larger). Therefore (1.12) will normally allow us to select the time-stepsize in a more flexible and, what is more

important, in a more efficient way than (1.10). In order to increase the flexibility and the efficiency one can also change the time-integration algorithm from one step to another. The implementation of the above ideas leads to the construction of a *variable stepsize variable formula method* (VSVFM; see, for example [40–44]) for the time-discretization part of the solution process.

In the following sections we shall show how a VSVFM is built-in in our package ADM (the constant stepsize constant formula part of ADM is fully documented in [46], the different algorithms applied in the subroutines of ADM are discussed in [45–51]). We shall also show that the implementation of a VSVFM leads to a full automization of the time-discretization part of the solution process. All problems connected with the choice of an optimal time-stepsize, which is often a very difficult task, are completely left to the code. As an additional benefit an error estimation of the result obtained by the time-discretization algorithm is easily computable. Thus the code will attempt to keep the errors in the time-integration algorithm smaller than an error tolerance TOL prescribed by the user. From the above analysis it becomes apparent that by varying both the time-stepsize and the formula (the time-discretization algorithm) the code not only attempts to preserve the stability during the computational process, but also attempts to keep the errors in the time-integration part of this process under a prescribed (in advance) level. In this way both the robustness of the code and the reliability of the results are considerably increased. This means that runs over long intervals $[a, b]$ can successfully be carried out (and, in fact, have been carried out by us) in an efficient, reliable and robust way when a flexible choice of the time-stepsize and the formula is allowed at each time-step.

2. Computations at an Arbitrary Time-Step

Consider the grids

$$X_M = \{x_m / x_0 = a_1,\ x_m = x_{m-1} + \Delta x,\ \Delta x = (b_1 - a_1)/2M,\ x_{2M} = b_1,\ M \in \mathbb{N}\}, \tag{2.1}$$

$$Y_N = \{y_n / y_0 = a_2,\ y_n = y_{n-1} + \Delta y,\ \Delta y = (b_2 - a_2)/2N,\ y_{2N} = b_2,\ N \in \mathbb{N}\}, \tag{2.2}$$

$$Z_p = \{z_p / z_0 = a_3,\ z_p = z_{p-1} + \Delta z,\ \Delta z = (b_3 - a_3)/P,\ z_p = b_3,\ P \in \mathbb{N}\}, \tag{2.3}$$

$$\mathcal{G} = X_M \times Y_N \times Z_p. \tag{2.4}$$

Grid $\mathcal{G}$ contains $L = (2M + 1)(2N + 1)(P + 1)$ grid-points. In the numerical treatment of the model (1.1)–(1.7) the continuous functions and the space derivatives of the unknown functions are replaced (at the beginning of each time-step) by grid functions each of which contains L components. Each component of a grid function is a value of the corresponding continuous function on some point of grid $\mathcal{G}$.

The first-order space derivatives are discretized by using a pseudospectral (Fourier) algorithm; [6, 19, 26]. This means that:(i) a trigonometric interpolation polynomial, which is a truncated Fourier series of c (or c^*), is built by the use of the values of c (or c^*) at the points of grid $\mathcal{G}$ (or part of them), (ii) the derivative of the

trigonometric polynomial with regard to x (or y) is calculated, (iii) the values of the derivatives of the trigonometric polynomials at the grid-points under consideration are accepted as values of the first order derivatives at the same grid-points. The subroutines of the FFT package developed by Swarztrauber [36] are attached to ADM as a *black box* in order to speed up the computations in the above algorithm. Our implementation of the pseudospectral (Fourier) algorithm is discussed in [45–51].

A special procedure is used in the treatment of the terms containing second-order derivatives (this procedure is justified in [48, 50], but it has been applied in [1]). By this procedure it is possible: (i) to calculate the corrections which are to be performed in the solution vector and which are due to the diffusion terms, (ii) to remove the stability restrictions on the time-stepsize due to the diffusion terms ([50]). The use of the special procedure is based on the basic assumptions made (see Section 1 and note the assumption that K_x and K_y are positive constants could be replaced by an assumption similar to the assumption made for K_z).

The time-discretization will be discussed in the next sections.

Let $t \in [a, b]$ be an arbitrary but fixed value of the time-variable. Assume that the values of $c(x, y, z, t)$ for $\forall (x, y, z) \in \mathcal{G}$ are given. The algorithm used in ADM to calculate the values of $c(x, y, z, t + \Delta t)$ for $\forall (x, y, z) \in \mathcal{G}$ can be described as follows.

ALGORITHM 2.1. *Computing approximations of* $c(x, y, z, t + \Delta t)$ *for* $\forall (x, y, z) \in \mathcal{G}$.

Step A. *For each parallel to Ox grid-line of* $\mathcal{G}$ *perform the following operations*: (A1) *carry out a forward FFT using the values of* $c(x, y, z)$ *on the grid-line under consideration*, (A2) *calculate approximations to the Fourier coefficients of* $\partial c(x, y, z, t)/\partial x$ *on the grid-line under consideration*, (A3) *multiply the Fourier coefficients of both* $c(x, y, z, t)$ *and* $\partial c(x, y, z, t)/\partial x$ *(on the grid-line under consideration) with appropriate exponentials of type* $\exp(-\mu^2 K_x d_x \Delta t)$ *(d_x being a constant;* $\mu = 0, \pm 1, \pm 2, \ldots, \pm M$), (A4) *perform a backward FFT for both* $c(x, y, z, t)$ *and* $\partial c(x, y, z, t)/\partial x$ *on the grid-line under consideration.*

Step B. *For each parallel to Oy grid-line of* $\mathcal{G}$ *perform the following operations*: (B1) *carry out a forward FFT using the values of* $\tilde{c}(x, y, z, t)$ *($\tilde{c}(x, y, z, t)$ being the transformed after Step* A *solution* $c(x, y, z, t)$*) on the grid-line under consideration*, (B2) *calculate approximations to the Fourier coefficients of* $\partial \tilde{c}(x, y, z, t)/\partial y$ *on the grid-line under consideration*, (B3) *multiply the Fourier coefficients of both* $\tilde{c}(x, y, z, t)$ *and* $\partial \tilde{c}(x, y, z, t)/\partial y$ *(on the grid-line under consideration) with appropriate exponentials of type* $\exp(-\nu^2 K_y d_y \Delta t)$ *(d_y being a constant;* $\nu = 0, \pm 1, \pm 2, \ldots, \pm N$), (B4) *perform a backward FFT for both* $\tilde{c}(x, y, z, t)$ *and* $\partial \tilde{c}(x, y, z, t)/\partial y$ *on the grid-line under consideration.*

Step C. *Use some time-integration algorithm to calculate approximations (at the points of* $\mathcal{G}$*) to* $\bar{c}(x, y, z, t)$*, where* $\bar{c}(x, y, z, t)$ *is the transformed after Step* B *solution* $\tilde{c}(x, y, z, t)$.

Step D. *For each parallel to Oz grid-line* (*and only for the points at which* $K_z > 0$) *perform the following operations*: (D1) *carry out a forward* even *FFT using the values of* $\bar{c}(x, y, z, t + \Delta t)$ *on the grid-line under consideration*, (D2) *multiply the Fourier coefficients of* $\bar{c}(x, y, z, z + \Delta t)$ (*on the grid-line under consideration*) *by appropriate exponentials of type* $\exp(-\rho^2 K_z d_z \Delta t)$ (d_z *being a constant*; $\rho = 0, 1, 2, \ldots, P$), (D3) *perform a backward* even *FFT using the modified* (*after Step* D2) *Fourier coefficients of* $\bar{c}(x, y, z, t + \Delta t)$ *and accept the results as values of the desired solution* $c(x, y, z, t + \Delta t)$ *at the points of the grid-line under consideration.*

Remark 2.1. Algorithm 2.1 can also be used to calculate approximations to the values of $c^*(x, y, z, t + \Delta t)$ for $\forall (x, y, z) \in \mathscr{G}$. In this way it is possible to handle (1.1) and (1.2) separately at each time-step. This is very convenient (let us mention only the fact that if the results obtained in the consideration of (1.1) are not acceptable, then the stepsize is reduced and the calculations are repeated without attempt to carry out any calculations with (1.2); see Section 4). However, this procedure can be applied only when Q does not depend on c^*. If Q depends on c^*, then Algorithm 2.1 should be applied to vector $(c, c^*)^T$ (instead of to vector c) in other words (1.1) and (1.2) should be considered simultaneously at each time-step.

Denote by $U(t)$, $V(t)$, $\bar{Q}(t)$, $\bar{Q}^*(t)$, $g(t)$ and $g^*(t)$ the grid-functions (on $\mathscr{G}$) corresponding to the continuous functions u, v, Q, Q^*, c and c^* respectively. The performance of the first two steps in Algorithm 2.1 leads to replacing (1.1) and (1.2) by two systems of ODEs (*the independent variable t is omitted*):

$$\frac{dg}{dt} = (US + V\bar{P}\bar{S}\bar{P})\, g + \bar{Q}, \qquad \frac{dg^*}{dt} = (US + V\bar{P}\bar{S}\bar{P})\, g^* + \bar{Q}^*, \tag{2.5}$$

where S and $\bar{S}$ are $L \times L$ quasi-diagonal matrices induced by the pseudospectral operators used in the discretization of the first-order space derivatives. S contains $(2N + 1)(P + 1)$ diagonal blocks each of which is a $(2M + 1) \times (2M + 1)$ skew-symmetric matrix. $\bar{S}$ contains $(2M + 1)(P + 1)$ diagonal blocks each of which is a $(2N + 1) \times (2N + 1)$ skew-symmetric matrix. $\bar{P}$ is a permutation matrix. S, $\bar{S}$ and $\bar{P}$ are never used explicitly in the computational process (this would require very much storage). The vectors

$$f = (US + V\bar{P}\bar{S}\bar{P})\, g + \bar{Q}, \qquad f^* = (US + V\bar{P}\bar{S}\bar{P})\, g^* + \bar{Q}^* \tag{2.6}$$

are calculated and the systems

$$\frac{dg}{dt} = f, \qquad \frac{dg^*}{dt} = f^* \tag{2.7}$$

are solved. However, the matrices S and $\bar{S}$ are needed in the stability analysis.

The solution of systems (2.7) is the main topic of the discussion in the next sections. More details about Algorithm 2.1 and/or the different parts of this

algorithm can be found in [45–51]. In these references many topics, which are not discussed in this paper (as, for example, the periodicity on the boundaries), are also treated.

3. Automatic Integration of the Systems of ODEs Obtained After the Space Discretization

Consider grid T_k from (1.8), assume that a constant time-stepsize Δt is used and denote $g_k = g(t_k)$. Special PECE (predictor-evaluation-corrector-evaluation) schemes of the type

$$\tilde{g}_k = \tilde{\alpha} g_{k-1} + (1-\tilde{\alpha}) g_{k-2} + \Delta t \sum_{i=1}^{s} \tilde{\beta}_i f_{k-i} \qquad \text{(prediction)}, \tag{3.1}$$

$$\tilde{f}_k = (US + V\bar{P}\bar{S}\bar{P})\tilde{g}_k + \bar{Q} \qquad \text{(evaluation)}, \tag{3.2}$$

$$g_k = \alpha g_{k-1} + (1-\alpha) g_{k-2} + \Delta t \beta_0 \tilde{f}_k + \Delta t \sum_{i=1}^{s} \beta_i f_{k-i} \qquad \text{(corrector)}, \tag{3.3}$$

$$f_k = (US + V\bar{P}\bar{S}\bar{P}) g_k + \bar{Q} \qquad \text{(evaluation)} \tag{3.4}$$

are used in the solution of the first system (2.7). The same schemes are also applied in the solution of the second system (2.7). It is assumed that: (i) $\tilde{\alpha} \in \mathbb{R}$, $\alpha \in [0, 2)$ (this pair will be chosen so that the stability properties of the PECE scheme close to the imaginary axis are as good as possible, [51], (ii) the set $\{\tilde{\beta}_i\}_{i=0}^{s}$ is determined so that the predictor (3.1) is a linear multistep formula of order s, (iii) the set $\{\beta_i\}_{i=0}^{s}$ is determined so that the corrector (3.3) is a linear multistep formula of order $s+1$. The PECE scheme (3.1)–(3.4) with (i)–(iii) satisfied is called an $(\tilde{\alpha}, \alpha) - P_s EC_{s+1} E$ scheme. Such schemes have been studied in [37, 40, 41, 43, 44, 51–54].

In order to increase the efficiency of the computational process the use of both different stepsizes and different $(\tilde{\alpha}, \alpha) - P_s EC_{s+1} E$ schemes at different parts of interval $[a, b]$ should be allowed. Assume that grid T_k is not equidistant and let Δt_k be the stepsize which has to be used at step k. Consider a set $\mathcal{F}$ of J $(\tilde{\alpha}, \alpha) - P_s EC_{s+1} E$ schemes. Denote the elements of $\mathcal{F}$ by $F_1, F_2, \ldots, F_J$. If varying both the stepsize and the element of set $\mathcal{F}$ from one step to another is allowed and if the element $F_j \in \mathcal{F}$ has to be specified at step k, then the formulae corresponding to (3.1) and (3.3) must be rewritten as

$$\tilde{g}_k = \tilde{\alpha}_j g_{k-1} + (1-\tilde{\alpha}_j) g_{k-2} + \Delta t_k \sum_{i=1}^{s_j} \tilde{\beta}_{ji}(\bar{h}_{kj}) f_{k-i}, \tag{3.5}$$

$$g_k = \alpha_j g_{k-1} + (1-\alpha_j) g_{k-2} + \Delta t_k \beta_{j0}(\bar{h}_{kj}) \tilde{f}_k + \Delta t_k \sum_{i=1}^{s_j} \beta_{ji}(\bar{h}_{kj}) f_{k\ i}, \tag{3.6}$$

where index j shows that (3.5) and (3.6) are corresponding to element $F_j \in \mathcal{F}$, while by the notation $\tilde{\beta}_{ji}(\bar{h}_{kj})$ and $\beta_{ji}(\bar{h}_{kj})$ the fact that these coefficients are dependent on vector $\bar{h}_{kj} = (\Delta t_k, \Delta t_{k-1}, ..., \Delta t_{k-s_j+1})$ (i.e., on the last s_j stepsize used in the time-integration) is expressed. The methods based on (3.5) and (3.6) are called *predictor-corrector variable stepsize variable formula methods* (PC VSVFMs); in [40, 41, 43, 44] it is shown that the fundamental properties (consistency, zero-stability and convergence) of the constant stepsize constant formula methods (3.1)–(3.4) are preserved in the transition to PC VSVFMs based on (3.5)–(3.6). Therefore we may concentrate our attention on the problems connected with the accuracy control and stability control when PC VSVFMs based on (3.5)–(3.6) are in use.

Consider again (3.1)–(3.4). The local truncation errors made in the computations at step k by this scheme are (see [23, 24, 34]):

$$\tilde{T}_k = \tilde{C}_{s+1}(\Delta t)^s g^{(s+1)}(t_k) + O((\Delta t)^{s+1}), \qquad T_k = O((\Delta t)^{s+1}) \tag{3.7}$$

for the predictor and corrector, respectively ($\tilde{C}_{s+1}$ being a constant dependent on the particular choice of the predictor only). The expressions for $\tilde{T}_k$ and T_k are derived under the assumptions (see, for example, [23, pp. 27–30]): (a) the previous values of the unknown function are calculated exactly, (b) the stepsize is sufficiently small. Nevertheless, the real-life computations indicate that these relationships can successfully be used to build-in both an accuracy control and a stepsize selection strategy in a code for solving ODEs. Indeed, observe that under assumptions (a) and (b) we have

$$\tilde{T}_k \approx g(t_k) - \tilde{g}_k, \qquad T_k \approx g(t_k) - g_k \qquad (g(t_k) \text{ is the exact solution at } t_k). \tag{3.8}$$

Therefore it is clear that the non-negative real numbers

$$\text{ERROR} = \| g_k - \tilde{g}_k \| \quad (\text{ERROR}^* = \| g_k^* - \tilde{g}_k^* \| \text{ for the second system (2.7)}) \tag{3.9}$$

give us an estimation of the principal part of the local truncation error made in the calculations with the predictor formula (3.1). It is also clear that this estimation is rather conservative (because the corrected values, g_k and g_k^*, are calculated with a formula whose order is higher than the order of the predictor). This means that we can expect the actual error to be smaller when the calculations are stable. It is apparent that the error estimations (3.9) can also be used in connection with (3.5)–(3.6) (at least when some restrictions on the stepsize selection strategy are imposed, [40, 41, 43, 44]). Assume now that some error tolerance TOL is prescribed. Then g_k is considered as acceptable if

$$\text{ERROR} \leqslant \text{TOL} \quad (\text{the test for } g_k^* \text{ is similar; TOL}^* \neq \text{TOL can be used}). \tag{3.10}$$

If the acceptability criterion (3.10) is not satisfied, then the calculations have to be repeated by specifying a smaller Δt_k. If the acceptability criterion (3.10) is satisfied, then the code proceeds with Part D of Algorithm 2.1 (or with the next step, when the model is two-dimensional, which is formally expressed by setting $K_z \equiv 0$). *If* (3.10) *is*

satisfied, then it is very important to determine an optimal value for the next stepsize Δt_{k+1} *by the use of which we can expect to calculate acceptable approximations* g_{k+1} *and* g^*_{k+1} *(i.e., we can expect* (3.10) *to be satisfied and, thus, it will not be necessary to repeat the calculations at step* $k+1$ *with a smaller stepsize*). This can be achieved by the use of the following device. Assume that a formula corresponding to $F_j \in \mathcal{F}$ is used at time-step k. Then (neglecting the terms containing $(\Delta t_k)^{s_j+1}$) we have

$$\text{ERROR} \approx |\tilde{C}_{s_j+1}(\bar{h}_{kj})|\,(\Delta t_k)^{s_j}\,\|g^{(s_j+1)}(t_k)\|, \tag{3.11}$$

and a similar relation holds for ERROR*. The next stepsize Δt_{k+1} should be determined so that

$$|\tilde{C}_{s_j+1}(\bar{h}_{k+1,j})|\,(t_{k+1})^{s_j}\,\|g^{(s_j+1)}(t_k)\| \approx \text{TOL} \tag{3.12}$$

(g^* and TOL* should be related in a similar way). Assume that

$$\tilde{C}_{s_j+1}(\bar{h}_{k+1,j}) \approx \tilde{C}_{s_j+1}(\bar{h}_{kj}), \qquad g^{(s_j+1)}(t_{k+1}) \approx g^{(s_j+1)}(t_k) \tag{3.13}$$

(and that a similar relation holds for g^*). An optimal stepsize, by which acceptable approximations g_{k+1} and g^*_{k+1} should be expected at the end of step $k+1$, can be calculated by

$$\Delta t_{k+1} = \delta \min\left[\left(\frac{\text{TOL}}{\text{ERROR}}\right)^{1/s_j}, \left(\frac{\text{TOL}^*}{\text{ERROR}^*}\right)^{1/s_j}\right]\Delta t_k, \tag{3.14}$$

where $0 < \delta < 1$ is a constant by which an attempt to compensate for the use of (3.13) with an equality sign is made.

The value for Δt_{k+1} given by (3.14) is commonly used in the ODE solvers ([7–16, 20–22, 27, 29]). However, it must be emphasized that this use is based on the expectation that the computations at step $k+1$ are stable. Normally, no stability control is made in the general-purpose software (see the above references again). It is expected that the code will reject the step and repeat the calculations with a smaller stepsize when Δt_{k+1} is too large and the computations are not stable; this auticorrelation mechanism is discussed in [28] (see also [30, 53]). However, the autocorrelation is carried out by some extra computational effort, which is very considerable when the system of ODEs is large (and this is the case when long-range transport phenomena are treated numerically). *Therefore one should attempt to ensure stability* in order to prevent rejections of steps and extra computations. For the special class of problems under consideration this can be done in a very cheap and efficient way. By the use of the results from [47, 51] it is easily seen that one should expect the computational process to be stable if

$$\Delta t_k \leqslant \frac{h_{\text{imag}}}{\pi}\left[\frac{\bar{u}_k(M-1)}{M\,\Delta x} + \frac{\bar{v}_k(N-1)}{N\,\Delta y}\right]^{-1}, \tag{3.15}$$

where $\bar{u}_k = \max(|u(x, y, z, t)|)$, $\bar{v}_k = \max(|v(x, y, z)|)$ (the maxima being taken for $x \in [a_1, b_1]$, $y \in [a_2, b_2]$, $z \in [a_3, b_3]$ and $t \in T_k^*$; T_k^* being a neighbourhood of the current integration point t_k), and h_{imag} being the length of the absolute stability interval on the positive part of the imaginary axis containing the origin ([51]). Strictly speaking, (3.15) will ensure stability only in the case: $v \equiv 0$, $Q \equiv 0$, $u \equiv u^*$ (u^* being a constant). In general, one can expect the results to be stable also in the case where the above conditions are not satisfied when there is a sufficiently large domain to the left of the imaginary axis that belongs to the absolute stability region of the time-integration scheme under consideration. This requirement may be crucial in some situations. We shall illustrate this by the following example. Let ($\lambda > 0$): $u \equiv 1$, $v \equiv 0$, $Q = -\lambda c$. Then the first system (2.5) can be rewritten as (I being the identity matrix):

$$\frac{dg}{dt} = Sg - \lambda g = (S - I)\, g. \tag{3.16}$$

The eigenvalues of S are purely imaginary ([19, 51]). Therefore only the length of the absolute stability interval on the imaginary axis is of interest when $\lambda = 0$. Denote the eigenvalues of S by ν_i ($i = 0(1)\, 2M$) and the eigenvalues of $T = S - \lambda I$ by μ_i ($i = 0(1)\, 2M$). Since $Tz = \mu z$ (z being a vector with $2M + 1$ components) implies $Sz = (\mu + \lambda) z$, the eigenvalues of T and S are related by $\mu_i = \nu_i - \lambda$ ($i = 0(1)\, 2M$). Thus the eigenvalues of T are in the negative half-part of the complex plane. In the long-range transport pollution models different kinds of deposition are often introduced by terms $-\lambda c$, but the value of λ is typically small compared with the modulus of the largest eigenvalue of matrix S.

The following conclusions can be drawn from the above analysis as well as from many experiments carried out with long-range transport models:

(i) The use of a stability check in an attempt to ensure stable computations is very efficient for this class of problems. We are not interested in carrying out computations with systems containing, say, 18432 ODEs which may be rejected (and if so should be repeated) only because the stepsize does not satisfy the stability requirements. We are interested in preventing rejections and this can efficiently be done if check (3.15) is carried out (see the experiments in Section 5; note also that Experiment 4 indicates that check (3.15) together with the other checks in ADM not only ensure stable computations but also provide reliable results).

(ii) Formula (3.15) shows clearly that we are interested in time-integration algorithms for which h_{imag} is large. Algorithms with increased stability regions have been developed in [18, 20, 33, 37, 52]. The ideas used in these references have been applied to develop special algorithms with large h_{imag} (see also [51]).

(iii) The algorithms with large h_{imag} are stable in a narrow strip along the imaginary axis (see Fig. 4.1). The last part of the analysis given in this section shows that this may be insufficient sometimes. Therefore a time-integration algorithm with a large stability region *must* be included in the code.

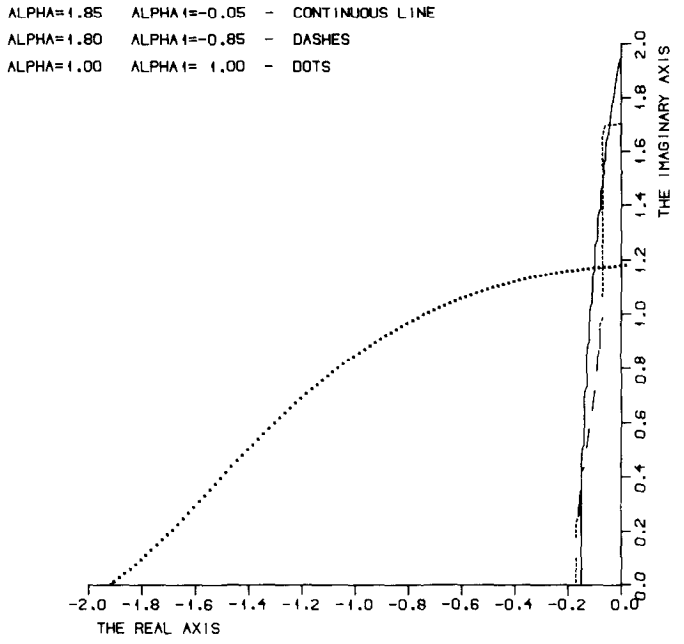

FIG. 4.1. Stability properties of the basic predictor-corrector schemes selected for package ADM. The parameters ALPHA and ALPHA1 correspond to α and $\tilde{\alpha}$, respectively. The third scheme (which is an Adams predictor-corrector scheme) is normally selected by the code only when the deposition rate is very high.

In the next section the time-integration schemes actually used in ADM will be presented and the rules for changing the stepsize and/or the algorithm will be discussed.

4. IMPLEMENTATION OF AN AUTOMATIC INTEGRATION OPTION IN PACKAGE ADM

The number of elements in the particular set $\mathcal{F}$ of basic $(\tilde{\alpha}, \alpha) - P_s EC_{s+1}E$ schemes used in ADM is $J = 3$. The values of parameters $\tilde{\alpha}$ and α as well as the values of s_j $(j = 1, 2, 3)$ for the predictor-corrector schemes chosen are given in Table 4.1. The values of the parameters h_{imag} (for the predictor-corrector schemes in set $\mathcal{F}$) are also given in Table 4.1. It is necessary to emphasize that: (i) the values of h_{imag} for the algorithms chosen by us are larger than the values of h_{imag} for the corresponding Adams schemes which are commonly used in the general-purpose solvers for non-stiff ODEs (see, e.g. [10, 11, 20–22, 27, 29]), (ii) the schemes chosen are absolutely stable in sufficiently large domains to the left of the imaginary axis (see Fig. 4.1). Moreover, an Adams P_3EC_4E scheme, which is very useful in some

TABLE 4.1

Some Important Characteristics of the Basic $(\tilde{\alpha}, \alpha) - P_s EC_{s+1} E$ Schemes Applied in the Particular Set $\mathcal{F}$ Selected for Package ADM[a]

F_j	$\tilde{\alpha}_j$	α_j	s_j	h_{imag}
F_1	−0.05	1.85	2	1.95
F_2	−0.85	1.80	3	1.70
F_3	1.00	1.00	3	1.17

[a] Notice that the last scheme is a well-known Adams predictor-corrector method; its parameter h_{imag} is considerably smaller than the values of this parameter for the other two schemes, but its stability region is large and, therefore, it is useful in the cases where the deposition rate is high.

extreme situations (that occur very seldom for our class of problems) is included in set $\mathcal{F}$.

The stepsize selection strategy is very simple. Let $\text{TOL} = \text{TOL}^*$ be the error tolerance prescribed by the user. Introduce $\text{TOL2} = 4\,\text{TOL}$ and $\text{TOL1} = \text{TOL}/4$. The approximations g_k and g_k^* are considered as acceptable when

$$\| g_k - \tilde{g}_k \| \leqslant \text{TOL2} \| g_k \|, \qquad \| g_k^* - \tilde{g}_k^* \| \leqslant \text{TOL2} \| g_k^* \|. \tag{4.1}$$

If the first relation in (4.1) is not satisfied, then the step is rejected (without any attempt to calculate an approximation to g_k^*) and the calculations are repeated with a stepsize $\Delta t_k/2$. If the second relation in (4.1) is not satisfied, then again the step is rejected and the calculations are repeated with a stepsize $\Delta t/2$. If any of the relations in (4.1) indicates failure 3 times successivaly, then a special restarting procedure with a one-step method is activated (a similar device is used in [10, 11, 29, 54]; the necessity of such a procedure is justified in [29]). If *any* of the following relations is satisfied,

$$\begin{aligned} \text{TOL} \| g_k \| &\leqslant \| g_k - \tilde{g}_k \| \leqslant \text{TOL2} \| g_k \|, \\ \text{TOL} \| g_k^* \| &\leqslant \| g_k^* - \tilde{g}_k^* \| \leqslant \text{TOL2} \| g_k^* \|, \end{aligned} \tag{4.2}$$

then the step is considered as successful, but the stepsize (for the next step) is reduced by setting: $\Delta t_{k+1} = 0.75 \Delta t_k$. If any of the following relations is satisfied,

$$\begin{aligned} \text{TOL1} \| g_k \| &\leqslant \| g_k - \tilde{g}_k \| \leqslant \text{TOL} \| g_k \|, \\ \text{TOL1} \| g_k^* \| &\leqslant \| g_k^* - \tilde{g}_k^* \| \leqslant \| g_k^* \|, \end{aligned} \tag{4.3}$$

[redacted]

$$\| g_k - \tilde{g}_k \| \leqslant \text{TOL1} \| g_k \| \wedge \| g_k^* - g_k^* \| \leqslant \text{TOL1} \| g_k^* \|, \tag{4.4}$$

then the stepsize is increased (using (3.14) with $\delta = 0.9$).

Several additional rules are used in the stepsize selection strategy. The most important of these additional rules are: (A) the stepsize is not increased if the change is not significant (*if* $\Delta t_{k+1}/\Delta t_k < 1.05$, *then* $\Delta t_{k+1} = \Delta t_k$ *is used at step* $k+1$), (B) the stepsize is not increased with a very large amount (*if* $\Delta t_{k+1}/\Delta t_k > 2$, *then* $\Delta t_{k+1} = 2\Delta t_k$ *is used at step* $k+1$).

The main rules in the formula selection strategy can be outlined as follows. Consider the beginning of the computations at step k (or the beginning of the computations at the *rejected* step k, when some of acceptability criteria have failed). The right-hand side of (3.15) is calculated for the 3 schemes in set $\mathcal{F}$ (see Table 4.1). Let STAB1, STAB2 and STAB3 be the calculated values of the right-hand side of (3.15) by the use of F_1, F_2 and F_3, respectively. If $\Delta t_k \leqslant$ STAB3, then F_3 is used at step k (without changing Δt_k). If $\Delta t_k \leqslant$ STAB2 and $\Delta t_k >$ STAB3, then F_2 is used at step k (without changing Δt_k). If $\Delta t_k \leqslant$ STAB1 and $\Delta t_k >$ STAB2, then F_1 will be used at step k (without changing Δt_k). If $\Delta t_k >$ STAB1, then the stepsize is reduced (to satisfy the stability requirements) by setting: $\Delta t_k =$ STAB1 and F_1 is used at step k.

Finally, it should be mentioned that a special starting procedure has to be used in the calculations of the first two approximations (g_k and g_k^* for $k = 1, 2$). A starting procedure whose predictor is the first-order explicit Euler formula and the corrector is the second order trapezoidal rule is attached to package ADM. The same procedure is used as a restarting procedure, when the code rejects the step 3 times successively (see the beginning of this section).

5. Efficiency of the VSVFM Option of Package ADM

From Section 4 it becomes apparent that some extra computational work per step is needed when a VSVFM is implemented in the time-integration part of Algorithm 2.1 (Step C). This extra work is needed to carry out: (a) accuracy control, (b) stability control, (c) changes of the stepsize and (d) changes of the formula. Therefore the crucial question is: *can we obtain a considerable compensation for the extra computational work per step by reducing the number of steps*? An answer to this question is given in this section. Numerical results obtained in runs with real meteorological data are used as illustrations of our statements. These data cover the period from 25.12.1979 to 10.01.1980 and have been collected within project EMEP (European Monitoring and Evaluation Program) in which practically all European countries participate.

Many runs have been carried out with the use of the CSCFM (constant stepsize constant formula method) option and with the use of the VSVFM option of ADM. We could present many tables with numerical results which illustrate clearly the efficiency of the latter option. However, we decided to follow another way by which not only is the efficiency shown but also it is explained *why* the VSVFM option is so efficient for our class of problems. Two-dimensional models are in use in the first 4

TABLE 5.1

The Parts of the Total Computing Time (in Percent) Used by the Subroutines of Package ADM When the Two Options in This Package Are Applied[a]

Subroutines	CSCFM option	VSVFM option
Part A + Part B	77%	72%
Part C	3%	10%
Overhead	20%	18%

[a] The EMEP data have been used in this run for the period from December 25, 1978, to January 31, 1979 (38 days). The calculations have been carried out on CRAY1 at Reading, England.

experiments (these are formally obtained by $K_z \equiv 0$), while some results obtained in runs with three-dimensional models are given in the last two experiments.

EXPERIMENT 1. We have run (1.1)–(1.7) with meteorological data for January 1979 on CRAY1 with option $ON = F$. The CFT compiler with $ON = F$ produces a chart giving the computing times taken by the different subroutines during the run. The results are given in Table 5.1. The computing time taken by the subroutines calculating u, v, Q and Q^* is given as "Overhead" in Table 5.1. The computing time taken for input-output operations is also included in the Overhead.

Table 5.1 shows clearly that the computational work in Step C (of Algorithm 2.1) is increased about 3 times when the VSVFM option is used (compared with the computational work in Step C for the CSCFM option). Nevertheless, the computational work in Step C for the VSVFM option is still a very small part of the total computational work for the run with the VSVFM option (only 10%). Thus, *this experiment indicates that the total computing time will be reduced (and, moreover, the reduction will be considerable) if the number of time-steps is reduced when the VSVFM is applied.* This statement is based on the fact that the two options differ only in Step C of Algorithm 2.1. The subroutines, which carry out the operations in the other parts of Algorithm 2.1, are identical for the two options. The same is true for the subroutines, which perform the operations included in the Overhead (Table 5.1). This means that the implementation of the time-integration is the only difference between the two options and, moreover, the computing time for the time-integration part is a very small part of the total computing time also when the VSVFM option is in use. By our next experiment we shall show that a very considerable reduction of steps should be expected when the VSVFM option is applied and we shall explain why this should be so.

EXPERIMENT 2. Since $M = N = 16$ and $\Delta x = \Delta y = 150000m$ in our space discretization and since the maximal length of the wind velocity vector for January 1979 was 43 m/sec (3.15) shows that the CSCFM option based on scheme F_2 can be used with a stepsize $\Delta t = 1800$ sec (a little larger stepsize can be used with F_1 but this

TABLE 5.2

Numerical Results Obtained with the EMEP Data from December 25, 1978, to January 31, 1979 (38 days) on an IBM 3033 Computer under the FORTH Compiler[a]

Option	Number of time-steps	Number of rejected steps	Average length of stepsize (in min)	CPU-time
CSCFM	1824	0	30.0	548.29
VSVFM	931	0	58.78	296.75

[a] The CPU-time is measured in seconds.

causes some problems with the reading of the meteorological data; data fields must be read at the end of every sixth hour). Numerical results obtained with the VSVFM and CSCFM (based on F_2) options are given in Table 5.2. It is seen that the VSVFM option performs in a nearly optimal way. The average stepsize for this option is nearly twice as large as that for the CSCFM option. Since the time-integration part is only a small part of the total work, the reduction of steps (achieved by the flexible VSVFM option) leads to a very considerable reduction of the computing time. A careful examination of the variation of the stepsize shows that the VSVFM option uses small stepsizes around the points of the time-interval where the length of the wind velocity vector is large (see also Experiment 6). However, there is no need to use such small step-sizes when the length of the wind velocity vector is not large and

TABLE 5.3

Numerical Results Obtained during a Long Run with the VSVFM Option on CRAY1 in Reading, England[a]

Interval of integration	Length of the interval (in min)	Number of time-steps	Number of rejected steps	Average stepsize (in min)
25.12.1978–31.01.1979	54720	931	0	58.78
01.02.1979–28.02.1979	40320	706	3	57.11
01.03.1979–31.03.1979	44640	710	1	62.87
01.04.1979–30.04.1979	43200	592	0	72.97
01.05.1979–31.05.1979	44640	613	3	72.82
01.06.1979–30.06.1979	43200	533	1	81.05
01.07.1979–31.07.1979	43200	505	1	88.40
01.08.1979–31.08.1979	44640	551	0	81.02
01.09.1979–30.09.1979	43200	594	0	72.72
01.10.1979–31.10.1979	44640	644	1	69.32
01.11.1979–30.11.1979	43200	712	2	60.67
01.12.1979–31.12.1979	44640	876	1	50.96
25.12.1978–31.12.1979	535680	7967	13	67.23

[a] The EMEP data have been used in this run.

the code is *able* to increase quickly the stepsize. This explains why the implementation of a VSVFM option is very useful for problems of the class which is of interest for us.

EXPERIMENT 3. The previous experiment shows that the performance of the VSVFM option is very efficient when the time-interval is relatively short. The next question is: *What will happen if the time-integration interval is long*? The EMEP data have also been used in a run with a time-interval from 24.12.1978 to 31.12.1979 (i.e., the length of the time-integration interval is 372 days; the corresponding length in the previous experiments was 38 days). The run, which is described below, has been carried out on CRAY1 at ECMWF (European Centre for Medium Range Weather Forecasts, Reading, England). The results are given in Table 5.3. It is seen that the *average* length of the stepsize is larger than 1 hour. The total number of steps is 7967. The computing time for the whole run was about 9 min. The number of rejected steps is an important factor when the efficiency of a VSVFM option is considered. This number is very small in this run (13 rejected steps or 0.16% of the

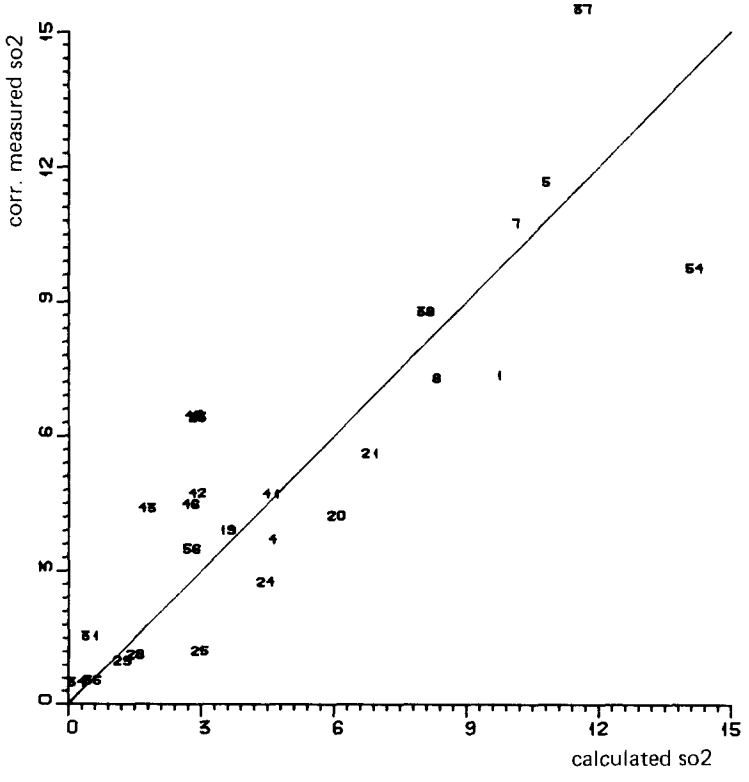

FIG. 5.1. Comparison of the mean values (for 1979) of sulphur di-oxide calculated by the VSVFM option of package ADM with the mean values of sulphur di-oxide observed at different EMEP stations.

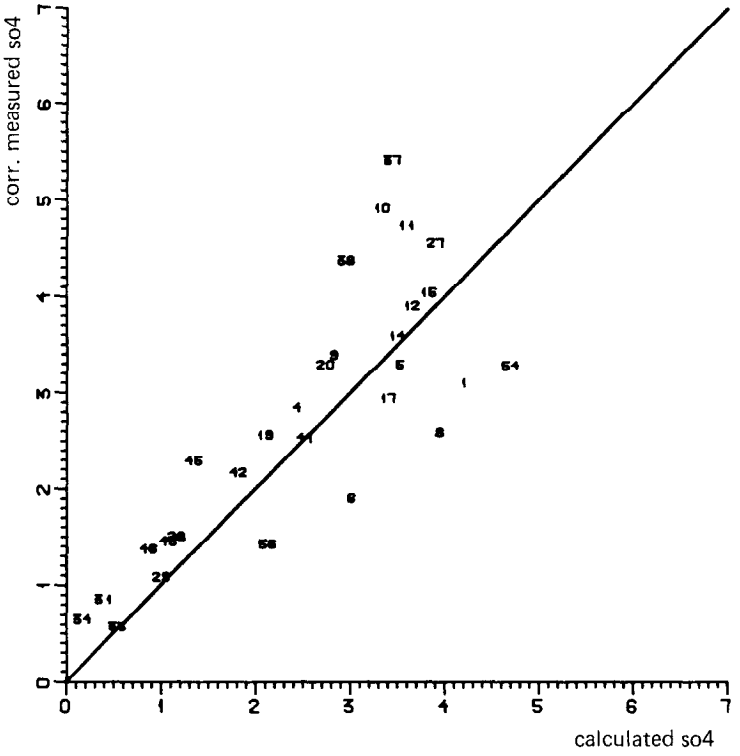

FIG. 5.2. Comparison of the mean values (for 1979) of sulphate calculated by the VSVFM option of package ADM with the mean values (for 1979) of sulphate observed at different EMEP stations.

EXPERIMENT 4. The following question is very important: *Are the results obtained by the VSVFM option reliable*? Many different experiments have been carried out in order to answer this question. A very straightforward test is the comparison of the calculated results with measured results at different stations located in the countries participating in the EMEP project. Many such tests have been carried out (a detailed report about the results being in preparation). Only the results of the comparisons of SO_2 and SO_4 concentrations for 1979 are given here (Figs. 5.1 and 5.2, respectively). Each EMEP station is denoted (on Figs. 5.1 and 5.2) by an integer. Two coordinates are attached to each station; the abscissa is the calculated result, while the observed result is the ordinate. It is seen from the figures that the coincidence between the calculated results and the observed results is quite satisfactory. Thus, the comparison indicates that the results obtained by the use of the VSVFM option are reliable.

EXPERIMENT 5. Three dimensional runs have also been carried out. It must be emphasized that the use of PECE schemes in this situation is possible only because the diffusion terms are treated in a special way in package ADM, by which the stability restrictions caused by these terms are removed ([48]). If the special procedure for treatment of the diffusion terms is not applied, then the main difficulty will be caused by the fact that Δz is very small ($\Delta z = 300m$; compare this with $\Delta x = \Delta y = 150000m$).

A run with the EMEP data from 25.12.1978 to 31.01.1979 (on an IBM 3033) takes approximately 81 min CPU time when the CSCFM option is in use. The CPU time for the same run with the VSVFM option is nearly halved.

EXPERIMENT 6. We have already mentioned that in the general-purpose codes for solving systems of ODEs the stepsize is varied only by the use of the accuracy checks (some of them are similar to (3.9)–(3.10); others are more complicated, [13, 35]). Exploiting the specific features of the systems of ODEs arising after the space

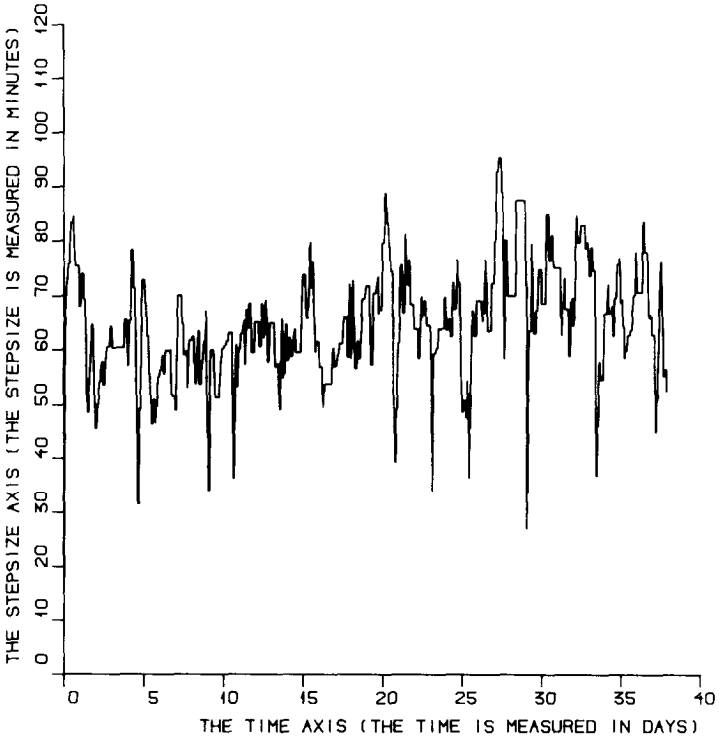

FIG. 5.3. The variation of the time-stepsize (measured in minutes) during a run with the EMEP data for the period from December 25, 1978 to January 31, 1979 (38 days). The curve given in this figure should be compared with the curve in Fig. 5.4. It is seen that the stepsize is normally large when the norm of the wind velocity vector is small and vice versa.

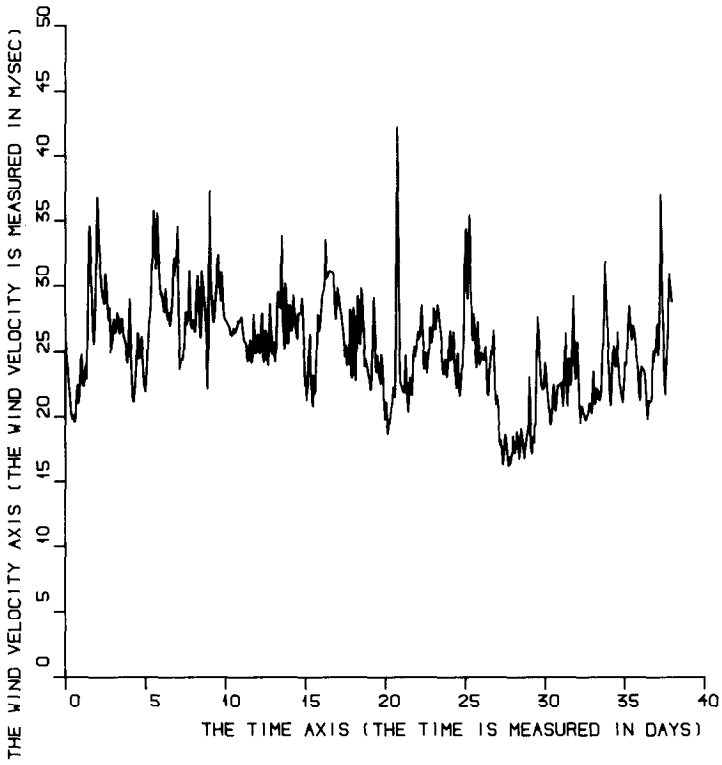

FIG. 5.4. The variation of the norm of the wind velocity vector on the space domain under consideration (measured in m/sec) found by the use of the EMEP data for the period from December 25, 1978 to January 31, 1979 (38 days). The curve given in this figure should be compared with the curve in Fig. 5.3. It is seen that if the norm of the wind velocity vector is large, then the stepsize selected by the code is normally small and vice versa.

discretization of PDEs describing long-range transport of air pollution, we have introduced a second check, (3.15), by which an attempt to carry out a stability control is made. The implementation of this check increases the computational work per step. Therefore it is necessary to justify the introduction of this check. An experiment with EMEP data from 25.12.1978 to 31.01.1979 has been performed in order to show that, since the basic PECE schemes are very accurate (the orders of the correctors are up to 4), the stepsize is often restricted by check (3.15). This means that: *if check* (3.15) *is not introduced, then the code will take into account the stability restrictions by rejecting many steps*, [28, 53], *which is very inefficient and should be avoided if possible.* It is obvious that if the stepsize is mainly restricted by (3.15), then the stepsize will normally be large when the norm of the wind velocity vector is small and vice versa. It is also obvious that this will only be a tendency because many other factors have influence on the stepsize selection strategy too (e.g.,

the accuracy, the formulae used, the deposition rate, among others). Nevertheless, this tendency is very clearly observed when Figs. 5.3 and 5.4 are compared. Similar results were obtained in many other runs. The efficiency of the stability check in the efforts to keep the number of rejected steps small is also clearly seen in the long run shown in Table 5.3 (look at the numbers of rejected steps for the different months).

6. Some Concluding Remarks

Several different questions may be asked in connection with our choice of algorithms and with the implementation of the selected algorithms. By the remarks given below we shall try to answer the questions, which have in our opinion to be answered.

Remark 6.1. *On the use of implicit formulae.* The PECE schemes as defined by (3.1)–(3.4) are *explicit* methods (there are no unknown quantities on the right-hand side of the formulae used in the computational process). We shall assume here that an implicit method is applied together with some Newton-like iterative process which require solving systems of algebraic equations. Under this assumption the use of implicit formulae in the time-integration part of Algorithm 2.1 seems to be prohibitive for long-range transport models of our type. Indeed, the meteorological data are often given in intervals of 6 hours. This imposes an upper bound on the stepsize for implicit methods (in order to avoid a loss of information). On the other hand, the PECE schemes can be used (also if the interval of integration is very long) with an average stepsize of about 1 hour. This indicates that the total number of steps can not be reduced by a factor larger than 6 when accurate implicit methods are used instead of the PECE schemes chosen by us. Our calculations show that such a reduction is far from sufficient to compensate for the increase of the computational work per step with the implicit methods due to the fact that large systems of algebraic equations must be solved at each time-step.

Remark 6.2. *On the use of general-purpose software for* ODEs. There exist many excellent general-purpose codes for solving ODEs ([2–5, 8, 17, 20–22, 27–32, 38]). However, the direct implementation of such a code in the case where long-range transport of air pollution is studied is not very attractive. The codes using implicit methods (with a Newton-like iteration) can not be applied in our case (see the previous remark). In the codes using explicit methods (including PECE schemes) as a rule formulae of Runge–Kutta type or Adams predictor-corrector schemes are adopted ([8–17, 20–22, 27, 29, 35, 54]). The former methods are not suitable because they require many evaluations of the right-hand side. In our case an evaluation of the right-hand side is equivalent to the space discretization + a considerable part of the overhead (see Table 5.1). This means that the evaluation of the right-hand side is a very expensive process. The use of formulae of Adams type only is not suitable because these have poor stability properties close to the imaginary axis (this is especially true for formulae of high order).

Another reason for development of a special software for our class of problems is the fact that it is possible to incorporate a very cheap and reliable stability check for the system of ODEs arising after the space discretization. In the general purpose software for solving ODEs the stepsize is normally selected by using accuracy checking criteria only. A good code *will* produce stable results also if only the accuracy is taken into account in the stepsize selection ([28]). However, the time-integration will be carried out with many rejections of time-steps (see Sections 3 and 5). Our numerical results indicate that the efficiency is increased considerably because an attempt to combine the accuracy criteria with some stability criteria is carried out in the time-integration part of package ADM.

Remark 6.3. *On the use of some other time-integration schemes.* Since the evaluation of the right-hand side of the system of ODEs is very expensive in our situation (see the previous remark), it seems to be advantageous to use a single formula (where 1 evaluation per step only is needed) instead of the PECE schemes (where 2 evaluations per step are carried out). This means that the computational work per step for the PECE schemes is, roughly speaking, twice as large as that for the single formulae. However, it turns out that the number of steps can also be reduced (roughly speaking by a factor of 2) when PECE schemes are used. This is so because $0 \leqslant h_{\text{imag}} \leqslant 1$ for a single formula of type (3.1), while $0 \leqslant h_{\text{imag}} \leqslant 2$ for the PECE schemes ([18, 47, 51]) and, moreover, it is possible to construct schemes with $h_{\text{imag}} \approx 2$. The use of PECE schemes has at least two extra advantages: (i) the known functions (u, v, as well as the terms in Q and Q^* that depend on the time only) are calculated only once per step and (ii) a cheap and reliable error estimation is obtainable. The question whether it is profitable to apply more complicated schemes ($P(EC)^m$, $PE(CE)^m$) or not is open. The answer is positive if one could construct a particular algorithm of this type with $h_{\text{imag}} \approx m$ for the $P(EC)^m$ schemes or $h_{\text{imag}} \approx m + 1$ for the $PE(CE)^m$ schemes. If such schemes with $m > 2$ (or $m + 1 > 2$ in the latter case) can be constructed, then they will be even more efficient than our schemes (the number of steps will be reduced and this will lead to a reduction of the number of evaluations of the known functions u, v, Q and Q^*). We plan to include such schemes in package ADM.

Remark 6.4. *On the error tolerance.* The use of the error tolerance TOL (Section 4) is based on the assumption that the errors made in the space

have no reliable checking criteria for the errors made during the space discretization. This implies that one should not specify a stringent error tolerance TOL. In all runs described above TOL = 0.1 was used. It should also be mentioned that our accuracy check is rather crude. When only a system of ODEs is solved, it is possible to attempt to estimate the global discretization error ([13, 35]). In our opinion such an attempt is justified only in conjunction with an attempt to estimate the global discretization error made during the space discretization process.

Remark 6.5. *On the use of other space discretization algorithms.* The VSVFMs

can be applied in the time-integration part also when other space discretization algorithms are used in Step A, Step B and Step D of Algorithm 2.1 (instead of the pseudospectral Fourier discretization). One could use, for example, finite differences or finite elements. The use of a space discretization with any of these classes of methods may be combined with application of some error estimation concerning the space discretization part of the solution process (and this is an advantage of these methods). However, it may be necessary to use other time-integration schemes. It may even be necessary to use different time-integration schemes; some in connection with the *advection terms*, others in connection with the *diffusion terms*. This means that it may be necessary to divide a global time-step into several fractional steps ([25, 39]). It should be mentioned that the use of fractional steps may sometimes be necessary even when the pseudospectral Fourier discretization is in use (if the special procedure from [48] can not be applied).

Remark 6.6. *Main conclusion.* The numerical results indicate that the implementation of a VSVFM option in a software for simulating long-range transport of air pollution in the atmosphere leads to a considerable increase of efficiency. Therefore such an option is useful when long runs with real meteorological data are carried out. On the other hand, a CSCFM option may be more efficient when the time-integration interval is short and/or when it is easy to determine an *optimal* value for the constant stepsize Δt by which a stable computational process should be expected. This means that both options should be kept in an attempt to allow more flexibility and to permit achieving optimal results in a wide variety of situations that may arise in practice.

REFERENCES

1. R. BERKOWICZ AND L. P. PRAHM, *Appl. Math. Modelling* **2** (1978), 205.
2. K. BURRAGE, J. C. BUTCHER, AND F. H. CHIPMAN, *BIT* **20** (1980), 185.
3. J. C. BUTCHER, K. BURRAGE, AND F. H. CHIPMAN, "STRIDE-Stable Runge–Kutta Integrator for Differential Equations," Computational Mathematics Report, March 1979, Department of Mathematics, University of Auckland, Auckland, New Zealand.
4. F. H. CHIPMAN, "Some Experiments with STRIDE," Working Papers for the 1979 SIGNUM Meeting on Numerical Ordinary Differential Equations, (R. D. Skeel, Ed.), Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana, Ill., 1979.
5. A. R. CURTIS, "The FACSIMILE Numerical Integrator for Stiff Initial Value Problems," Report AERZ-R9352, Computer Science and Systems Division, AERE Harwell, Osfordshire, England, 1979.
6. B. FORNBERG, *SIAM J. Numer. Anal.* **12** (1975), 509.
7. W. H. ENRIGHT, R. BEDET, I. FARKAS, AND T. E. HULL, "Test Results on Initial Value Methods for Non-stiff Ordinary Differential Equations," Report No. 68, Department of Computer Science, University of Toronto, Toronto, Canada, 1974.
8. W. H. ENRIGHT, T. E. HULL, AND B. LINDBERG, *BIT* **15** (1975), 10.
9. P. W. GAFFNEY, "Information and Advice on Numerical Software," Report ORNL/CSD/TM-147, Computer Sciences Division, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1981.
10. C. W. GEAR, *Comm. ACM* **14** (1971), 176.
11. C. W. GEAR, *Comm. ACM* **14** (1971), 185.
12. I. GLADWELL, *ACM Trans. Math. Software* **5** (1979), 386.

13. P. M. HANSON AND W. H. ENRIGHT, *ACM Trans. Math. Software* **9** (1983), 71.
14. A. C. HINDMARSH, "GEAR: Ordinary Differential Equation Solver," Report UCLR-51186 (rev. 3), Lawrence Livermore Laboratory, Lvermore, Calif., 1974.
15. A. C. HINDMARSH, *ACM SIGNUM Newsl.* **15** (1980), 10.
16. A. C. HINDMARSH AND G. D. BYRNE, "EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations," Report UCID-30112, Laxrence Livermore Laboratory, Livermore, Calif., 1977.
17. T. E. HULL, W. H. ENRIGHT, B. M. FELLEN, AND A. E. SEDGWICK, *SIAM J. Numer. Anal.* **9** (1982), 803.
18. R. JELTSCH AND O. NEVANLINNA *Numer. Math.* **37** (1981), 61.
19. H. O. KREISS AND J. OLIGER, *Tellus* **XXIV** (1972), 199.
20. F. T. KROGH, *J. Assoc. Comput. Mach.* **13** (1966), 374.
21. F. T. KROGH, "VODQ/SVDQ/DVDQ-Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations," Report NPO-11643, Jet Propulsion Laboratory, Pasadena, Calif., 1969.
22. F. T. KROGH, *J. Assoc. Comput. Mach.* **20** (1973), 545.
23. J. D. LAMBERT, "Computational Methods in Ordinary Differential Equations," Wiley, London/New York/Sydney, 1973.
24. L. LAPIDUS AND J. H. SEINFELD, "Numerical Solution of Ordinary Differential Equations," Academic Press, New York, 1971.
25. R. RICHTMAYER AND K. MORTON, "Difference Methods for Initial Value Problems," Interscience, New York, 1967.
26. S. A. ORSZAG, *J. Fluid Mech.* **49** (1971), 75.
27. A. E. SEDGWICK, "An Effective Variable Order Variable Step Adams Method," Report No. 53, Department of Computer Science, University of Toronto, Toronto, Canada, 1973.
28. L. F. SHAMPINE, Stiffness and non-stiff differential equations solvers, *in* "Numerische Behandlung von Differentialgleichungen" (L. Collatz, Ed.), Int. Ser. Numer. Math. Vol. 27, Birkhäuser, Basel, Switzerland, 1975.
29. L. F. SHAMPINE AND M. K. GORDON, "Computer Solution of Ordinary Differential Equations: The Initial Value Problem," Freeman, San Francisco, Calif., 1975.
30. L. F. SHAMPINE AND K. L. HIEBERT, *Comput. Math. Appl.* **3** (1977), 41.
31. L. F. SHAMPINE AND H. A. WATTS, "DEPACK-Design of a User Oriented Package of ODE Solvers," Report SAND79-2374, Sandia National Laboratories, Albuquerque, New Mexico, 1980.
32. L. F. SHAMPINE, H. A. WATTS, AND S. M. DAVENPORT, *SIAM Rev.* **18** (1976), 376.
33. H. J. STETTER, *Computing* **3** (1968), 286.
34. H. J. STETTER, "Analysis of Discretization Methods in Ordinary Differential Equations," Springer-Verlag, Berlin, 1973.
35. H. J. STETTER, *ACM Trans. Math. Software* **5** (1979), 415.
36. P. N. SWARZTRAUBER, Vectorizing the FFT's *in* "Parallel Computations" (G. Rodrigue, Ed.), Academic Press, New York/London, 1982.
37. P. G. THOMSON AND Z. ZLATEV, *BIT* **19** (1979), 503.
38. J. G. VERWER, *ACM Trans. Math. Software* **6** (1980), 236.
39. N. N. YANENKO, "The Method of Fractional Steps," Springer-Verlag, Berlin, 1971.
40. Z. ZLATEV, *Numer. Math.* **31** (1978), 175.
41. Z. ZLATEV, *Numer. Math.* **37** (1981), 157.
42. Z. ZLATEV, *SIAM J. Sci. Statist. Comput.* **2** (1981), 321.
43. Z. ZLATEV, *Computing* **31** (1983), 47.
44. Z. ZLATEV, "Variable Stepsize Variable Formula Methods Basec on Predictor-Corrector Schemes," Report No. 261, The Danish Center for Applied Mathematics and Mechanics, The Technical University of Denmark, Lyngby, Denmark, 1983.
45. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, "Numerical treatment of the Advection–Diffusion Equation: Part I—Space Discretization," Report No. MST LUFT-A47, Air Pollution Laboratory,

National Agency of Environmental Protection, Risø National Laboratory, DK-4000 Roskilde, Denmark, 1981.

46. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, "Numerical Treatment of the Advection-Diffusion Equation: Part V—Documentation of ADM01," Report No. MST LUFT-A58, Air Pollution Laboratory, National Agency of Environmental Protection, Risø National Laboratory, DK-4000 Roskilde, Denmark, 1982.
47. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, Choice of time-integration scheme in pseudospectral algorithm for advection equations, *in* "Computational Methods for Fluid Dynamics" (K. W. Morton and M. J. Baines, Eds.), pp. 303–321, Academic Press, New York/London, 1982.
48. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, "Numerical Treatment of the Advection-Diffusion Equation: Part VI—Special Treatment of the Diffusion Terms," Report No. MST LUFT-A72, Air Pollution Laboratory, National Agency of Environmental Protection, Risø National Laboratory, DK-4000 Roskilde, Denmark, 1983.
49. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, *Comput & Fluids* **11** (1983), 13.
50. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, *Atmos. Environ.* **17** (1983), 491.
51. Z. ZLATEV, R. BERKOWICZ, AND L. P. PRAHM, *J. Comput. Phys.* **51** (1983), 1.
52. Z. ZLATEV AND P. G. THOMSEN, *Internat. J. Numer. Meth. Engng.* **14** (1979), 1051.
53. Z. ZLATEV AND P. G. THOMSEN, *ACM Trans. Math. Software* **5** (1979), 401.
54. Z. ZLATEV AND P. G. THOMSEN, Differential integrators based on linear multistep methods, *in* "Méthode numérique dans les sciences de l'ingenieur-G.A.M.N.I. 2" (E. Absi, R. Glowinski, P. Lascaux and H. Veysseyre, Eds.), pp. 221–231. Dunod, Paris, 1980.