ELSEVIER

Applied Numerical Mathematics 16 (1994) 201-225

APPLIED NUMERICAL MATHEMATICS

Time integration of three-dimensional numerical transport models [☆]

B.P. Sommeijer [*], P.J. van der Houwen, J. Kok

CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherlands

This paper is dedicated to Professor Robert Vichnevetsky to honor him on the occasion of his 65th birthday

Abstract

We analyse three-dimensional models for computing transport of salinity, pollutants, suspended material (such a sediment or mud), etc. The main purpose of this paper is to present an overview of the various possibilities for the time discretization of the advection and diffusion terms that can take advantage of the parallelization and vectorization facilities offered by CRAY-type computers. Among the suitable time integration techniques, we have both explicit and implicit methods. In explicit methods, the parallelization is straightforward, but these methods are hampered by a severe time step restriction due to stability. This can be avoided by selecting an implicit method; however, such a choice necessitates the frequent solution of systems of equations. For the implicit methods considered in this survey, these systems essentially have a tridiagonal structure. Even for this relatively simple form, the greater part of the total solution time is spent in solving these systems. Therefore, this part of the algorithm needs special attention in order to get good performance on parallel/vector architectures. Following a suggestion of Golub and Van Loan, we have experimented with several implementations on a CRAY YMP4, which will be reported.

Keywords: Transport models; 3D advection-diffusion equations; Numerical time integration; Parallelism

1. Introduction

The mathematical model describing transport processes of salinity, pollutants, suspended material (such as sediment or mud), etc., is defined by a system of 3D advection-diffusion-reac-

[☆] The investigations reported in this paper were supported by the NCF (National Computing Facilities Foundation) who provided the authors with a grant (CRG 93.01) from Cray's University Research and Development Grant Program.

[*] Corresponding author. Fax: (+31-20)592-4199. E-mail: bsom@cwi.nl.

tion equations

$$\frac{\partial c_i}{\partial t} = -\frac{\partial}{\partial x}(uc_i) - \frac{\partial}{\partial y}(vc_i) - \frac{\partial}{\partial z}((w-w_{\mathrm{f}})c_i) + \frac{\partial}{\partial x}\left(\varepsilon_x \frac{\partial c_i}{\partial x}\right) + \frac{\partial}{\partial y}\left(\varepsilon_y \frac{\partial c_i}{\partial y}\right) + \frac{\partial}{\partial z}\left(\varepsilon_z \frac{\partial c_i}{\partial z}\right) + g_i, \tag{1.1a}$$

where

- c_i: concentrations of the contaminants,
- u, v, w: local fluid velocities in x, y and z directions,
- w_{f}: fall velocity, only nonvanishing in the case of suspended material,
- ε_x, ε_y, ε_z: diffusion coefficients in x, y and z directions,
- g_i: source and reaction terms.

The velocities u, v, w, and the diffusion coefficients ε_x, ε_y, ε_z are assumed to be known in advance. The fall velocity w_{f} is only relevant in the case of modeling transport of suspended material, and may be a nonlinear function of the concentration (cf. Toro et al. [18]). The terms g_i describe chemical reactions, emissions from sources, etc., and therefore depend on the concentrations c_i. Thus, the mutual coupling of the equations in the system (1.1a) is only due to the functions g_i. In the present paper we shall concentrate on the numerical modeling of a single transport equation. The extension to systems, including chemical terms, is the subject of current research. In the following, we omit the index i occurring in (1.1a).

The physical domain in space is bounded by the vertical, closed boundary plane $p(x, y) = 0$, the water elevation surface $z = \zeta(t, x, y)$, and the bottom profile $z = -d(x, y)$. The functions p, ζ and d are also assumed to be known in advance. The boundary conditions along these boundary planes depend on the particular application at hand. Most considerations in this paper apply to the case where they are of the general, linear form:

$$\begin{array}{lll} \text{Vertical boundaries:} & p(x,y)=0: & a_{\mathrm{v}}(t,x,y,z)c + b_{\mathrm{v}}\dfrac{\partial c}{\partial n} = c_{\mathrm{v}}(t,x,y,z), \\ \text{Water surface boundary:} & z=\zeta(t,x,y): & a_{\mathrm{s}}(t,x,y)c + b_{\mathrm{s}}\dfrac{\partial c}{\partial z} = 0, \\ \text{Bottom boundary:} & z=-d(x,y): & a_{\mathrm{d}}(t,x,y)c + b_{\mathrm{d}}\dfrac{\partial c}{\partial z} = c_{\mathrm{d}}(t,x,y). \end{array} \tag{1.1b}$$

Here $\partial/\partial n$ denotes differentiation along the inward normal, a_{v}, a_{s}, a_{d}, c_{v}, and c_{d} are given functions and b_{v}, b_{s} and b_{d} are given constants. If the constants b_{v}, b_{s} or b_{d} vanish, then the boundary condition will be called a Dirichlet-type condition.

Given the initial concentration $c(t_0, x, y, z)$, the concentration c can be computed in space and time by solving the initial boundary value problem (1.1). The numerical solution of this 3D problem requires powerful computing facilities such as offered by the CRAY supercomputers. In order to exploit these facilities, the spatial computational domain should be as simple as possible. There are two obvious approaches to simplify the spatial computational domain, viz. (i) coordinate transformations and (ii) the "dummy-point" approach. In both cases, the spatial computational domain becomes a rectangular box which can be discretized by means of a uniform rectangular grid leading to efficient implementations on vector computers.

In the case of coordinate transformations, the 3D physical domain itself is mapped on a rectangular box. This approach has the additional advantage that the physical boundaries are exactly represented. However, the price we pay is firstly, a much more complicated mathematical model and secondly, as a consequence of the transformation, the possibility of introducing coefficient functions of large magnitude. The transformation of only one space variable already leads to quite unattractive formulas. We illustrate this by working out the transformation of the z-variable, the so-called *σ-transformation*. In the σ-transformation, the variables c and z are replaced by the new variables c^* and σ which are related according to

$$c(t, x, y, z) = c^*(t, x, y, \sigma) = c^*(t, x, y, \theta(t, x, y, z)),$$
$$\sigma = \theta(t, x, y, z) := \frac{z - \zeta(t, x, y)}{h(t, x, y)}, \tag{1.2}$$

where h is the water height defined by $h := d(x, y) + \zeta(t, x, y)$. Evidently, σ takes values in the unit interval $[-1, 0]$. From (1.2) the transformed equation in terms of c^*, t, x, y, and σ is easily obtained by substitutions like

$$\frac{\partial c}{\partial t} = \frac{\partial c^*}{\partial t} + \frac{\partial c^*}{\partial \sigma}\frac{\partial \theta}{\partial t}, \quad \frac{\partial c}{\partial x} = \frac{\partial c^*}{\partial x} + \frac{\partial c^*}{\partial \sigma}\frac{\partial \theta}{\partial x}, \quad \ldots, \quad \frac{\partial c}{\partial z} = \frac{\partial c^*}{\partial \sigma}\frac{\partial \theta}{\partial z} = \frac{\partial c^*}{h\partial \sigma}.$$

Since the function $\theta(t, x, y, z)$ depends both on t and all spatial variables, the transformed equation is much more complicated, and therefore more expensive, than the original transport equation (1.1a). Furthermore, these expressions show the introduction of large coefficients $\partial\theta/\partial x$ and $\partial\theta/\partial y$ into the model in the case of a rapidly changing bottom profile $d(x, y)$. Similarly, transformation of the (x, y)-variable introduces large coefficients in the case of a rapidly changing coast geometry. As a consequence, the numerical solution process should be based on an implicit time integration in order to avoid restrictive time step conditions.

The second, "dummy-point" approach encloses the whole physical domain in a rectangular box and considers this box as the computational domain. As a consequence, there may be a lot of meaningless, dummy grid points. However, in spite of performing calculations at these dummy points, the regular grid facilitates efficient implementation which compensates the additional computational effort. Compared with the coordinate transformation approach, the advantage is the simple mathematical model, the disadvantage is a less accurate representation of the physical boundaries.

The two approaches just outlined can be combined. For example, in [1], we find such a combined approach for the shallow water equations. In the present paper, we shall follow the dummy-point approach.

2. Semidiscretization

A widely used approach for the discretization of (1.1) is to apply the method of lines (MOL). This semidiscretization process transforms the partial differential equation into a system of ordinary differential equations (ODEs) by discretizing only the spatial derivatives and leaving time continuous. Another approach is the so-called *direct* discretization, in which the derivatives in space and time are *simultaneously* replaced by discrete analogues. For a comparison of both techniques we refer to [5]. In this paper we shall follow the MOL approach.

Before applying the semidiscretization process, it is convenient to rewrite equation (1.1a) by taking into account the particular applications we have in mind: we shall only consider the *incompressible* case, that is, we assume $u_x + v_y + w_z = 0$, and furthermore, the diffusion coefficients ε_x, ε_y, ε_z will be assumed to be constant. Recalling that we concentrate on a single transport equation, (1.1a) simplifies to

$$\frac{\partial c}{\partial t} = -u\frac{\partial c}{\partial x} - v\frac{\partial c}{\partial y} - w\frac{\partial c}{\partial z} + \frac{\partial(w_{\mathrm{f}}c)}{\partial z} + \varepsilon\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2}\right) + g. \tag{2.1}$$

The first step in the semidiscretization process is to let the physical domain be enclosed by a rectangular box, containing the grid points

$$P_{j,k,m} := (x_0 + j\Delta x,\ y_0 + k\Delta y,\ z_0 + m\Delta z),$$
$$j = 0,\ldots,d_x+1, \quad k = 0,\ldots,d_y+1, \quad m = 0,\ldots,d_z+1, \tag{2.2a}$$

where (x_0, y_0, z_0) corresponds with the southwest corner point at the bottom of the rectangular box. It will be assumed that the mesh parameters Δx, Δy and Δz are such that the physical boundaries can be approximated by a subset of grid points with sufficient accuracy. This subset of boundary grid points will be denoted by $\partial\mathbb{B}$. Of course, some geometrical constraints on the location of the boundary points have to be imposed; however, such aspects are beyond the scope of this paper. The boundary grid points divide the remaining grid points in outer and inner ones lying "outside" and "inside" $\partial\mathbb{B}$. These sets of points are denoted by $\mathbb{B}_{\mathrm{out}}$ and $\mathbb{B}_{\mathrm{in}}$. Furthermore, we assume that the enclosing box is such that no grid points of $\partial\mathbb{B}$ are on the boundary planes of the box (i.e., the indices j, k and m of the boundary points satisfy $1 \leqslant j \leqslant d_x$, $1 \leqslant k \leqslant d_y$, and $1 \leqslant m \leqslant d_z$).

Next the spatial differential operators are replaced by finite differences, so that we can approximate (2.1) by a system of ODEs. There is a considerable amount of literature on the spatial discretization of advection-dominated partial differential equations. For a recent overview we refer to [17], in which both symmetric and unsymmetric discretizations (i.e., upwind) are discussed. An advantage of upwinding is that so-called "wiggles", which may lead to negative concentrations, can be suppressed. This positivity is crucial for models in which chemical terms are involved. This kind of semidiscretization will be discussed in a forthcoming paper where such a model will be studied in more detail. In the present paper, where the emphasis is on the vectorization and parallelization features of the various time integrators, we shall use second-order, symmetric differences (see also [16, p. 24] where the discretization of nondissipative advection is discussed).

Furthermore, we denote the numerical approximations at $P_{j,k,m}$ to c, u, ... by capitals $C_{j,k,m}$, $U_{j,k,m}, \ldots$, and we introduce the spatial shift operators $\mathscr{X}_\pm$, $\mathscr{Y}_\pm$ and $\mathscr{Z}_\pm$ defined by

$$\mathscr{X}_\pm C_{j,k,m} := C_{j\pm1,k,m}, \qquad \mathscr{Y}_\pm C_{j,k,m} := C_{j,k\pm1,m}, \qquad \mathscr{Z}_\pm C_{j,k,m} := C_{j,k,m\pm1}.$$

Then, on the *computational* set of grid points $\mathbb{S}$ defined by

$$\mathbb{S} := \{P_{j,k,m}\colon 1 \leqslant j \leqslant d_x,\ 1 \leqslant k \leqslant d_y,\ 1 \leqslant m \leqslant d_z\} \tag{2.2b}$$

the associated ODEs take the form

$$\begin{aligned}\frac{\mathrm{d}C_{j,k,m}}{\mathrm{d}t} = &-\left\{\frac{1}{2\Delta x}U_{j,k,m}[\mathscr{X}_+ - \mathscr{X}_-] + \frac{1}{2\Delta y}V_{j,k,m}[\mathscr{Y}_+ - \mathscr{Y}_-]\right.\\ &\qquad\left. + \frac{1}{2\Delta z}W_{j,k,m}[\mathscr{Z}_+ - \mathscr{Z}_-]\right\}C_{j,k,m}\\ &+\varepsilon\left\{\frac{1}{(\Delta x)^2}[\mathscr{X}_+ - 2 + \mathscr{X}_-] + \frac{1}{(\Delta y)^2}[\mathscr{Y}_+ - 2 + \mathscr{Y}_-]\right.\\ &\qquad\left. + \frac{1}{(\Delta z)^2}[\mathscr{Z}_+ - 2 + \mathscr{Z}_-]\right\}C_{j,k,m}\\ &+\frac{1}{2\Delta z}\omega(C)_{j,k,m}[\mathscr{Z}_+ - \mathscr{Z}_-]C_{j,k,m} + g_{j,k,m},\end{aligned} \tag{2.3a}$$

where $\omega(c) := w_{\mathrm{f}}(c) + c\partial w_{\mathrm{f}}(c)/\partial c$.

The next step is to take into account the boundary conditions. If $P_{j,k,m}$ is a (physical) boundary point where the boundary condition is of Dirichlet type, then $C_{j,k,m}$ is explicitly given (see (1.1b)), so that by means of (numerical) differentiation with respect to time, we obtain ODEs of the form

$$\frac{\mathrm{d}C_{j,k,m}}{\mathrm{d}t} = d_{j,k,m}(t), \tag{2.3b}$$

which should replace the ODEs occurring in (2.3a) at all Dirichlet-type boundary points (here, $d_{j,k,m}(t)$ is explicitly determined by the boundary conditions).

If $P_{j,k,m}$ is a boundary point of non-Dirichlet type, then the corresponding ODE in (2.3a) asks for the concentration at one or more outer grid points. By means of the boundary conditions, the concentrations in these auxiliary outer points can explicitly be expressed in terms of concentrations at inner (or boundary) grid points.

Finally, we assume vanishing concentrations at all points in the computational set $\mathbb{S}$ which are in $\mathbb{B}_{\text{out}}$, as well as on the boundaries of the enclosing box, i.e.,

$$C_{j,k,m} = 0, \qquad j = 0, d_x + 1, \quad k = 0, d_y + 1, \quad m = 0, d_z + 1. \tag{2.3c}$$

In conclusion, the equations (2.3) corresponding to the set of grid points $\mathbb{S}$ as defined by (2.2b) define a second-order consistent semidiscretization of the initial–boundary value problem (1.1) of dimension $d := d_x d_y d_z$. Notice that only the concentrations defined by this system of ODEs corresponding to the grid points of $\mathbb{B}_{\text{in}}$ and $\partial\mathbb{B}$ are relevant.

In the analysis of time integrators for (2.3), it is more convenient to represent the system (2.3) in the form

$$\begin{aligned}\frac{\mathrm{d}C(t)}{\mathrm{d}t} = \boldsymbol{F}(t, \boldsymbol{C}(t)) := &A_x(t)\boldsymbol{C}(t) + A_y(t)\boldsymbol{C}(t) + A_z(t)\boldsymbol{C}(t) + N_z(\boldsymbol{C}(t))\boldsymbol{C}(t)\\ &+ \boldsymbol{G}(t, \boldsymbol{C}(t)) + \boldsymbol{B}_x(t) + \boldsymbol{B}_y(t) + \boldsymbol{B}_z(t),\end{aligned} \tag{2.4}$$

tives in space and time are *simultaneously* replaced by discrete analogues. For a comparison of both techniques we refer to [5]. In this paper we shall follow the MOL approach.

Before applying the semidiscretization process, it is convenient to rewrite equation (1.1a) by taking into account the particular applications we have in mind: we shall only consider the *incompressible* case, that is, we assume $u_x + v_y + w_z = 0$, and furthermore, the diffusion coefficients ε_x, ε_y, ε_z will be assumed to be constant. Recalling that we concentrate on a single transport equation, (1.1a) simplifies to

$$\frac{\partial c}{\partial t} = -u\frac{\partial c}{\partial x} - v\frac{\partial c}{\partial y} - w\frac{\partial c}{\partial z} + \frac{\partial(w_f c)}{\partial z} + \varepsilon\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2}\right) + g. \tag{2.1}$$

The first step in the semidiscretization process is to let the physical domain be enclosed by a rectangular box, containing the grid points

$$\begin{aligned} &P_{j,k,m} := (x_0 + j\Delta x,\ y_0 + k\Delta y,\ z_0 + m\Delta z), \\ &j = 0,\dots,d_x + 1, \quad k = 0,\dots,d_y + 1, \quad m = 0,\dots,d_z + 1, \end{aligned} \tag{2.2a}$$

where (x_0, y_0, z_0) corresponds with the southwest corner point at the bottom of the rectangular box. It will be assumed that the mesh parameters Δx, Δy and Δz are such that the physical boundaries can be approximated by a subset of grid points with sufficient accuracy. This subset of boundary grid points will be denoted by $\partial\mathbb{B}$. Of course, some geometrical constraints on the location of the boundary points have to be imposed; however, such aspects are beyond the scope of this paper. The boundary grid points divide the remaining grid points in outer and inner ones lying "outside" and "inside" $\partial\mathbb{B}$. These sets of points are denoted by $\mathbb{B}_{\text{out}}$ and $\mathbb{B}_{\text{in}}$. Furthermore, we assume that the enclosing box is such that no grid points of $\partial\mathbb{B}$ are on the boundary planes of the box (i.e., the indices j, k and m of the boundary points satisfy $1 \leqslant j \leqslant d_x$, $1 \leqslant k \leqslant d_y$, and $1 \leqslant m \leqslant d_z$).

Next the spatial differential operators are replaced by finite differences, so that we can approximate (2.1) by a system of ODEs. There is a considerable amount of literature on the spatial discretization of advection-dominated partial differential equations. For a recent overview we refer to [17], in which both symmetric and unsymmetric discretizations (i.e., upwind) are discussed. An advantage of upwinding is that so-called "wiggles", which may lead to negative concentrations, can be suppressed. This positivity is crucial for models in which chemical terms are involved. This kind of semidiscretization will be discussed in a forthcoming paper where such a model will be studied in more detail. In the present paper, where the emphasis is on the vectorization and parallelization features of the various time integrators, we shall use second-order, symmetric differences (see also [16, p. 24] where the discretization of nondissipative advection is discussed).

Furthermore, we denote the numerical approximations at $P_{j,k,m}$ to c, u, ... by capitals $C_{j,k,m}$, $U_{j,k,m}$, ..., and we introduce the spatial shift operators $\mathscr{X}_\pm$, $\mathscr{Y}_\pm$ and $\mathscr{Z}_\pm$ defined by

$$\mathscr{X}_\pm C_{j,k,m} := C_{j\pm 1,k,m}, \qquad \mathscr{Y}_\pm C_{j,k,m} := C_{j,k\pm 1,m}, \qquad \mathscr{Z}_\pm C_{j,k,m} := C_{j,k,m\pm 1}.$$

Then, on the *computational* set of grid points $\mathbb{S}$ defined by

$$\mathbb{S} := \{P_{j,k,m}\colon 1 \leqslant j \leqslant d_x,\ 1 \leqslant k \leqslant d_y,\ 1 \leqslant m \leqslant d_z\} \tag{2.2b}$$

the associated ODEs take the form

$$\begin{aligned}\frac{\mathrm{d}C_{j,k,m}}{\mathrm{d}t} = &-\left\{\frac{1}{2\Delta x}U_{j,k,m}[\mathscr{X}_{+}-\mathscr{X}_{-}] + \frac{1}{2\Delta y}V_{j,k,m}[\mathscr{Y}_{+}-\mathscr{Y}_{-}]\right.\\ &\left. + \frac{1}{2\Delta z}W_{j,k,m}[\mathscr{Z}_{+}-\mathscr{Z}_{-}]\right\}C_{j,k,m}\\ &+\varepsilon\left\{\frac{1}{(\Delta x)^2}[\mathscr{X}_{+}-2+\mathscr{X}_{-}] + \frac{1}{(\Delta y)^2}[\mathscr{Y}_{+}-2+\mathscr{Y}_{-}]\right.\\ &\left. + \frac{1}{(\Delta z)^2}[\mathscr{Z}_{+}-2+\mathscr{Z}_{-}]\right\}C_{j,k,m}\\ &+\frac{1}{2\Delta z}\omega(C)_{j,k,m}[\mathscr{Z}_{+}-\mathscr{Z}_{-}]C_{j,k,m} + g_{j,k,m},\end{aligned} \tag{2.3a}$$

where $\omega(c) := w_{\mathrm{f}}(c) + c\partial w_{\mathrm{f}}(c)/\partial c$.

The next step is to take into account the boundary conditions. If $P_{j,k,m}$ is a (physical) boundary point where the boundary condition is of Dirichlet type, then $C_{j,k,m}$ is explicitly given (see (1.1b)), so that by means of (numerical) differentiation with respect to time, we obtain ODEs of the form

$$\frac{\mathrm{d}C_{j,k,m}}{\mathrm{d}t} = d_{j,k,m}(t), \tag{2.3b}$$

which should replace the ODEs occurring in (2.3a) at all Dirichlet-type boundary points (here, $d_{j,k,m}(t)$ is explicitly determined by the boundary conditions).

If $P_{j,k,m}$ is a boundary point of non-Dirichlet type, then the corresponding ODE in (2.3a) asks for the concentration at one or more outer grid points. By means of the boundary conditions, the concentrations in these auxiliary outer points can explicitly be expressed in terms of concentrations at inner (or boundary) grid points.

Finally, we assume vanishing concentrations at all points in the computational set $\mathbb{S}$ which are in $\mathbb{B}_{\text{out}}$, as well as on the boundaries of the enclosing box, i.e.,

$$C_{j,k,m} = 0, \qquad j = 0, d_x+1, \quad k = 0, d_y+1, \quad m = 0, d_z+1. \tag{2.3c}$$

In conclusion, the equations (2.3) corresponding to the set of grid points $\mathbb{S}$ as defined by (2.2b) define a second-order consistent semidiscretization of the initial–boundary value problem (1.1) of dimension $d := d_x d_y d_z$. Notice that only the concentrations defined by this system of ODEs corresponding to the grid points of $\mathbb{B}_{\text{in}}$ and $\partial\mathbb{B}$ are relevant.

In the analysis of time integrators for (2.3), it is more convenient to represent the system (2.3) in the form

$$\begin{aligned}\frac{\mathrm{d}C(t)}{\mathrm{d}t} = \boldsymbol{F}(t, \boldsymbol{C}(t)) := &\boldsymbol{A}_x(t)\boldsymbol{C}(t) + \boldsymbol{A}_y(t)\boldsymbol{C}(t) + \boldsymbol{A}_z(t)\boldsymbol{C}(t) + \boldsymbol{N}_z(\boldsymbol{C}(t))\boldsymbol{C}(t)\\ &+\boldsymbol{G}(t, \boldsymbol{C}(t)) + \boldsymbol{B}_x(t) + \boldsymbol{B}_y(t) + \boldsymbol{B}_z(t),\end{aligned} \tag{2.4}$$

where $C(t)$ is a vector of dimension d containing all concentrations defined at the points of $\mathbb{S}$, and $A_x(t)$, $A_y(t)$, $A_z(t)$, and $N_z(C)$ are $d \times d$ matrices with (at most) three nonzero elements in each row. Notice that the matrix N_z vanishes if the fall velocity w_{f} is absent in (1.1a). The vectors $\boldsymbol{B}_x(t)$, $\boldsymbol{B}_y(t)$ and $\boldsymbol{B}_z(t)$ represent the inhomogeneous contributions of the boundary conditions at the vertical east and west boundaries, at the vertical north and south boundaries, and at the surface and bottom boundaries, respectively.

In order to get some insight into the magnitude of the entries in these arrays, we consider the case where the fluid velocities and the diffusion coefficients can be considered as (locally) constant in space. Then, ignoring boundary conditions and omitting the spatial subscripts (j, k, m), the matrices A_x, A_y, A_z and N_z are defined by the stencils

$$\begin{aligned}
A_x &:= \frac{1}{(\Delta x)^2}\left[\varepsilon + \tfrac{1}{2}\Delta x\, U \quad -2\varepsilon \quad \varepsilon - \tfrac{1}{2}\Delta x\, U\right]_{x\text{-direction}},\\
A_y &:= \frac{1}{(\Delta y)^2}\left[\varepsilon + \tfrac{1}{2}\Delta y\, V \quad -2\varepsilon \quad \varepsilon - \tfrac{1}{2}\Delta y\, V\right]_{y\text{-direction}},\\
A_z &:= \frac{1}{(\Delta z)^2}\left[\varepsilon + \tfrac{1}{2}\Delta z\, W \quad -2\varepsilon \quad \varepsilon - \tfrac{1}{2}\Delta z\, W\right]_{z\text{-direction}},\\
N_z &:= \frac{1}{2\Delta z}\left[-\omega(C) \quad 0 \quad \omega(C)\right]_{z\text{-direction}}.
\end{aligned} \tag{2.5}$$

These approximate values will be used in our stability analysis of the various time integrators.

3. Basic time integration methods

In this section we discuss potential time integrators for integrating the space-discretized transport equation (2.4) on CRAY-type computers. We will respectively consider

- stabilized Runge methods,
- locally one-dimensional methods,
- Richardson extrapolation of locally one-dimensional methods,
- operator splitting methods,
- hopscotch methods,
- nested operator splitting methods,
- predictor–corrector methods.

These methods have been selected because of their attractive numerical features (such as stability and accuracy), and their suitability for an efficient implementation as well (aspects like vectorization, parallelization and storage requirements). With the exception of the stabilized Runge method, these methods are implicit; however, this implicitness is of a very mild nature: these methods are at most "tridiagonally" implicit. Using a special technique called "vectorization-across-the-linear-systems" (see Section 5), the solution of the tridiagonal systems can be performed with high speed on a vector processor.

3.1. Stabilized Runge methods

Suppose that we apply to (2.4) the explicit method

$$\begin{aligned}
C^{(1)} &= C_n + \alpha_1 \Delta t\, F(t_n, C_n),\\
C^{(2)} &= C_n + \alpha_2 \Delta t\, F\big(t_n, C^{(1)}\big),\\
&\vdots\\
C^{(q-2)} &= C_n + \alpha_{q-2} \Delta t\, F\big(t_n, C^{(q-3)}\big),\\
C^{(q-1)} &= C_n + \tfrac{1}{2} \Delta t\, F\big(t_n, C^{(q-2)}\big),\\
C_{n+1} &= C_n + \Delta t\, F\big(t_n + \tfrac{1}{2}\Delta t, C^{(q-1)}\big).
\end{aligned} \tag{3.1}$$

This method is a q-stage Runge–Kutta method in which the t-arguments in the first $q-1$ stages are frozen. Per step, it requires q calls of the right-hand side function $F(t, C)$ with only two different values of the t-argument. Thus, in problems where the t-update of $F(t, C)$ is costly, the method (3.1) may be attractive, even for larger values of q. In the case of (2.4), the main effort in evaluating $F(t, C)$ comes from the evaluation of the matrices $A_x(t)$, $A_y(t)$, $A_z(t)$, and $N_z(t, C(t))$, so that (3.1) is effectively a *two-stage* method. Furthermore, it is second-order accurate in time for any set of fixed, bounded coefficients $\{\alpha_1, \ldots, \alpha_{q-2}\}$, the storage requirements are modest and the structure of the system (2.4) allows an extremely efficient implementation on vector computers. Moreover, in the applications we have in mind, the vector loops are so long that the availability of several parallel vector processors can be fully exploited without sacrificing vector speed.

The only drawback is the stability condition on the size of the time step which is of the form

$$\Delta t \leqslant \frac{\beta(q)}{\rho(\partial F/\partial C)}, \tag{3.2}$$

where $\rho(\cdot)$ denotes the spectral radius function and $\beta(q)$ defines the stability boundary of the method. The coefficients $\{\alpha_1, \ldots, \alpha_{q-2}\}$ can be employed for relaxing the stability condition (3.2) (cf. [9]). The resulting method may be considered as a stabilized Runge method (if $q = 2$, then Runge's original method appears). A suitable choice of the coefficients depends on the type of the system (2.4), i.e., the nature of the spectrum of $\partial F/\partial C$. We distinguish the real, imaginary and "left halfplane" stability boundary. These boundaries are relevant in the case where the eigenvalues of $\partial F/\partial C$ are on the negative axis, the imaginary axis and in the left halfplane, respectively.

If the diffusion terms are dominating in (2.4), i.e., if

$$\Delta x \ll 4\varepsilon |U|^{-1}, \qquad \Delta y \ll 4\varepsilon |V|^{-1}, \qquad \Delta z \ll 4\varepsilon \min\{|W|^{-1}, |\omega(C)|^{-1}\},$$

then the eigenvalues of $\partial F/\partial C$ are located in a long, narrow strip along the negative real axis and the spectral radius is given by

$$\rho(\partial F/\partial C) \approx 4\varepsilon\big((\Delta x)^{-2} + (\Delta y)^{-2} + (\Delta z)^{-2}\big). \tag{3.3a}$$

In this case, the so-called Runge–Kutta–Chebyshev (RKC) method, which possesses the nearly optimal real stability boundary $\beta(q) = 2q^2/3$, should be effective [10,15]. Since the computational effort per step increases linearly with q, whereas the stability boundary is quadratic in q, the RKC method is an efficient time integrator for diffusion-dominated problems.

If advection is dominating, then, within the family of stabilized Runge methods, the four-stage method with $(q, \alpha_1, \alpha_2) = (4, \frac{1}{4}, \frac{1}{3})$ is recommended. This method has the same stability polynomial as the standard, fourth-order Runge–Kutta method, that is, it satisfies the stability condition (3.2) with imaginary stability boundary $\beta(4) = 2\sqrt{2}$. In advection-dominated problems, we have

$$\rho(\partial F/\partial C) \approx \frac{|U|}{\Delta x} + \frac{|V|}{\Delta y} + \frac{|W| + |\omega|}{\Delta z}, \tag{3.3b}$$

showing that the stability condition (3.2) requires Δt to be of $O(\min(\Delta x, \Delta y, \Delta z))$. We shall refer to this second-order, four-stage method as the RK24 method.

The stabilized Runge methods are in a sense special-purpose methods because the coefficients can be tuned to a particular problem class. This makes them suitable candidates for use in operator splitting methods to be discussed in Section 3.3.

3.2. *Locally one-dimensional methods*

We shall consider the locally one-dimensional (LOD) method of Yanenko [19] in its original form and a modification which is of interest for a parallel implementation (cf. [12]).

3.2.1. *LOD method of Yanenko*

The LOD method of Yanenko is based on the idea of splitting the right-hand side according to the spatial derivatives and to perform an integration step by integrating the fractional equations sequentially by means of the highly stable backward Euler method. When applied to (2.4) we obtain

$$\begin{aligned}
&C^{(1)} - \Delta t\, A_x(t_{n+1})C^{(1)} = C_n + \Delta t\, B_x(t_{n+1}),\\
&C^{(2)} - \Delta t\, A_y(t_{n+1})C^{(2)} = C^{(1)} + \Delta t\, B_y(t_{n+1}),\\
&C^{(3)} - \Delta t\left[A_z(t_{n+1}) + N_z(C^{(3)})\right]C^{(3)} = C^{(2)} + \Delta t\, B_z(t_{n+1}),\\
&C_{n+1} - \Delta t\, G(t_{n+1}, C_{n+1}) = C^{(3)}.
\end{aligned} \tag{3.4}$$

This method is first-order accurate and will be referred to as the LOD1 method. In addition to evaluating the matrices A_x, A_y, and A_z, the costs per step consist of solving $d_y d_z$ tridiagonal, linear systems of dimension d_x, $d_x d_z$ tridiagonal, linear systems of dimension d_y, $d_x d_y$ tridiagonal, nonlinear systems of dimension d_z, and d nonlinear scalar equations. We remark that, without reducing the order of accuracy, the matrix $N_z(C^{(3)})$ in the third stage may be replaced by $N_z(C_n)$. Furthermore, if G does not depend on C, then the fourth stage can be omitted by replacing $B_z(t_{n+1})$ with $B_z(t_{n+1}) + G(t_{n+1})$ and by setting $C_{n+1} = C^{(3)}$.

The overall costs of the method (3.4) seem to be extremely high, both in computational volume per step and with respect to storage. The present (shared memory) CRAY computers offer an amount of storage which is in the order of 1–2 Gigabytes, corresponding to 128–256 Megawords (double precision). Since this LOD1 method requires 16 arrays (cf. Table 1), including the arrays for storing the velocity fields, grids of up to 8–16 million grid points can be treated.

With respect to the computational effort per step, we observe that in this particular case where so many systems of equal dimension are involved, the solution process can be implemented rather efficiently on vector computers by executing the substeps of the many tridiagonal solvers vectorwise (cf. [2, p. 156] and Section 5 of the present paper). This technique was successfully used by de Goede [1] in solving the three-dimensional shallow water equations, and will be referred to as *vectorization-across-tridiagonal-solvers*. Thus, for LOD methods, the computational effort per step can be substantially reduced on vector computers. Moreover, the favourable stability characteristics of the underlying backward Euler method enable us to take arbitrarily large stepsizes as far as stability is concerned (see e.g. [14] for a discussion on the stability of LOD methods). This property, together with the technique of vectorization-across-tridiagonal-solvers, implies that the overall costs of the LOD approach are acceptable. However, since the LOD1 method is only first-order accurate, large stepsizes may destroy the accuracy. In order to retain accuracy, we may use Richardson extrapolation, to be discussed in Section 3.2.3.

3.2.2. *Parallel LOD method*

Instead of sequential integration of the fractional equations, we may perform parallel integrations to obtain the method

$$\begin{aligned}
&C^{(1)} - \Delta t\, A_x(t_{n+1})C^{(1)} = C_n + \Delta t\, B_x(t_{n+1}),\\
&C^{(2)} - \Delta t\, A_y(t_{n+1})C^{(2)} = C_n + \Delta t\, B_y(t_{n+1}),\\
&C^{(3)} - \Delta t\big[A_z(t_{n+1}) + N_z(C^{(3)})\big]C^{(3)} = C_n + \Delta t\, B_z(t_{n+1}),\\
&C^{(4)} - \Delta t\, G\big(t_{n+1}, C^{(4)}\big) = C_n,\\
&C_{n+1} = \tfrac{1}{4}\big(C^{(1)} + C^{(2)} + C^{(3)} + C^{(4)}\big).
\end{aligned} \tag{3.5}$$

This modification of Yanenko's method is again first-order accurate and differs from (3.4) in that all component equations use the same initial value C_n, rather than using the result of the preceding equation as starting value. As a consequence, the vectors $C^{(i)}$, $i = 1,\dots,4$ can be computed in parallel. This parallel method becomes attractive on computers where not all parallel vector processors can be fully exploited for vector-loop computations.

The stability is governed by the variational equation

$$\begin{aligned}
&\Delta C_{n+1} = R(\Delta t)\Delta C_n,\\
&R(\Delta t) := \tfrac{1}{4}\Big(\big[I - \Delta t\, A_x\big]^{-1} + \big[I - \Delta t\, A_y\big]^{-1} + \big[I - \Delta t\, A_z - \Delta t\, N_z\big]^{-1}\\
&\qquad\qquad + \big[I - \Delta t\, \partial G/\partial C\big]^{-1}\Big).
\end{aligned} \tag{3.6}$$

Under mild conditions on the matrices A_x, A_y, A_z, N_z, and $\partial G/\partial C$, unconditional stability can be proved along the same lines as followed in [14].

3.2.3. Richardson extrapolation

By applying Richardson extrapolation, the accuracy behaviour of LOD methods can be improved, and, since Richardson extrapolation is highly parallel, the sequential costs are hardly increased.

Let us represent the LOD method we want to accurize in the compact form

$$C_{n+1} = L(\Delta t, C_n), \tag{3.4$'$}$$

where the operator L defines the particular method under consideration. Assuming that this method is first-order accurate, we define the two-point and three-point, *local* Richardson extrapolation method by the formulas

$$C_{n+1} = -L(\Delta t, C_n) + 2L\left(\tfrac{1}{2}\Delta t, L\left(\tfrac{1}{2}\Delta t, C_n\right)\right), \tag{3.7a}$$

$$C_{n+1} = \tfrac{1}{2}\left[L(\Delta t, C_n) - 8L\left(\tfrac{1}{2}\Delta t, L\left(\tfrac{1}{2}\Delta t, C_n\right)\right) + 9L\left(\tfrac{1}{3}\Delta t, L\left(\tfrac{1}{3}\Delta t, L\left(\tfrac{1}{3}\Delta t, C_n\right)\right)\right)\right], \tag{3.7b}$$

respectively. The *sequential* (or, *effective*) computational costs of the two-point and three-point methods are twice and three times the costs of applying the operator L, because all terms occurring in (3.7) can be computed concurrently. Notice that in the case of (3.7b) there is almost perfect load balancing on two processors, because the computational costs of computing the first two terms is roughly equal to that of computing the last term. Although the sequential (effective) costs per step for (3.7b) are a factor 3/2 higher than those for (3.7a), it is likely that, as far as accuracy is concerned, (3.7b) allows us to take stepsizes that are much more than a factor 3/2 larger. We shall refer to {(3.4), (3.7a)} and {(3.4), (3.7b)} as the LOD2 and LOD3 methods, respectively.

We computed numerically the stability region of the LOD2 method and found that the method is, like the LOD1 method (3.4), unconditionally stable (assuming that the diffusion coefficient ε is nonnegative and the diagonal matrix $\partial G/\partial C$ has nonpositive diagonal entries). The LOD3 method was verified to be "almost" unconditionally stable in the sense that amplifications by factors 1.0001 occur if the eigenvalues of A_x, A_y, A_z, N_z or $\partial G/\partial C$ approach certain parts of the imaginary axis.

3.3. Operator splitting methods

LOD methods are based on the excellent stability behaviour of the backward Euler method. However, we have seen that the RKC method and the RK24 method are potential candidates if, respectively, diffusion and advection are dominating. This suggests replacing the (dimensional) LOD splitting of the right-hand side function by diffusion–advection splitting (operator splitting), that is, we write

$$\frac{\mathrm{d}C(t)}{\mathrm{d}t} = A_1(t)C(t) + A_2(t)C(t) + N_z(C(t))C(t) + G(t, C(t)) + B_1(t) + B_2(t),$$

where the matrices A_2 and A_1 are symmetric and skew-symmetric, respectively, and where

$B_1(t)$ and $B_2(t)$ stem from the contributions of the boundary conditions. Next, we integrate the fractional equations

$$\frac{dC(t)}{dt}=A_1(t)C(t)+N_z(C(t))C(t)+B_1(t),$$
$$\frac{dC(t)}{dt}=A_2(t)C(t)+G(t, C(t))+B_2(t). \tag{3.8}$$

Again, two versions can be distinguished, viz. the Yanenko-type version and a parallel version.

3.3.1. Yanenko-type method

As in the LOD method of Yanenko, the idea is to integrate the fractional equations (3.8) sequentially in each step. This yields the first-order accurate method

$$C^{(1)}=L_1(\Delta t, C_n), \qquad C_{n+1}=L_2(\Delta t, C^{(1)}), \tag{3.9}$$

provided that L_1 and L_2 define integration methods that are at least first-order accurate (cf. [12]). An efficient method is obtained by tuning the operator L_1 to the special properties of A_1 and N_z, and the operator L_2 to the special properties of A_2 and G. Since the eigenvalues of A_1 and N_z are purely imaginary L_1 should define an integration method that is suitable for equations whose Jacobian matrices possess imaginary spectra, that is, we should choose a *hyperbolic* solver. Likewise, the Jacobian associated with the second equation has real eigenvalues, so that this equation should be integrated by a *parabolic* solver.

Suppose that the operators L_1 and L_2 are respectively defined by the RK24 method and the q-stage RKC method of Section 3.1. Then, the stability is determined by the stability conditions

$$\Delta t \leqslant \frac{2\sqrt{2}}{\rho(A_1(t_n)+N_z(C_n))}, \qquad \Delta t \leqslant \frac{2q^2}{3\rho(A_2(t_n)+\partial G/\partial C(t_n, C_n))}. \tag{3.10}$$

By means of the parameter q the step of the RKC method can be tuned to that of RK24.

3.3.2. Parallel versions

Consider the first-order method

$$C_{n+1}=\tfrac{1}{2}[L_1(\Delta t, C_n)+L_2(\Delta t, C_n)], \tag{3.11a}$$

or the second-order modification

$$C_{n+1}=\tfrac{1}{2}[L_1(\Delta t, L_2(\Delta t, C_n))+L_2(\Delta t, L_1(\Delta t, C_n))], \tag{3.11b}$$

where L_1 and L_2 are defined as before, for example, by suitable stabilized Runge methods. The method (3.11a) modifies (3.9) in the same way as (3.5) modifies (3.4). Evidently, the expressions $L_1(\Delta t, C_n)$ and $L_2(\Delta t, C_n)$ can be computed in parallel. The stability condition for (3.11) is identical with (3.10).

3.4. Hopscotch methods

The disadvantage of the LOD-type and fractional stabilized Runge methods is their low order in time due to the fractioning of the differential equation. An alternative approach is again based on operator splitting, but it does not integrate fractions of equations but the full equation in the subsequent stages. Consider a, for the moment, arbitrary splitting of the right-hand side function $\boldsymbol{F}$ in (2.4):

$$F(t, C(t)) = A(t, C(t)) + B(t, C(t)). \tag{3.12}$$

Then we can define a family of second-order splitting methods by writing

$$\begin{aligned} &C_{n+1/2} - \tfrac{1}{2}\Delta t\, A(t_{n+1/2}, C_{n+1/2}) = C_n + \tfrac{1}{2}\Delta t\, B(t_n, C_n), \\ &C_{n+1} - \tfrac{1}{2}\Delta t\, B(t_{n+1}, C_{n+1}) = C_{n+1/2} + \tfrac{1}{2}\Delta t\, A(t_{n+1/2}, C_{n+1/2}). \end{aligned} \tag{3.13}$$

The stability of these methods can be studied by linearizing the operators A and B to obtain

$$\begin{aligned} &\Delta C_{n+1} = R(\Delta t)\Delta C_n, \\ &R(\Delta t) := \left[I - \tfrac{1}{2}\Delta t\, B\right]^{-1}\left[I + \tfrac{1}{2}\Delta t A\right]\left[I - \tfrac{1}{2}\Delta t A\right]^{-1}\left[I + \tfrac{1}{2}\Delta t\, B\right]. \end{aligned} \tag{3.14}$$

Stability requires the eigenvalues of $R(\Delta t)$ within the unit circle, or equivalent, the eigenvalues of

$$\begin{aligned} &R^*(\Delta t) = \left[I - \tfrac{1}{2}\Delta t\, B\right]R(\Delta t)\left[I - \tfrac{1}{2}\Delta t\, B\right]^{-1} = R_A(\Delta t)R_B(\Delta t), \\ &R_A(\Delta t) := \left[I + \tfrac{1}{2}\Delta t\, A\right]\left[I - \tfrac{1}{2}\Delta t\, A\right]^{-1}, \qquad R_B(\Delta t) := \left[I + \tfrac{1}{2}\Delta t\, B\right]\left[I - \tfrac{1}{2}\Delta t\, B\right]^{-1}. \end{aligned} \tag{3.15}$$

Thus, the stability is essentially determined by the matrices $R_A(\Delta t)$ and $R_B(\Delta t)$.

The most familiar splitting is the so-called Alternating Direction Implicit (ADI) splitting where the right-hand side function is split according to the spatial dimensions. Unfortunately, within the family (3.13), the ADI idea can only be applied to two-dimensional problems. However, possibilities for three-dimensional problems are offered by the odd–even hopscotch (OEH) splitting, the odd–even line hopscotch (OELH) splitting and nested operator splitting (for a discussion of hopscotch techniques we refer to [3]).

3.4.1. Odd–even hopscotch splitting

For any vector V, let V_o denote the vector with zero components at all grid points $P_{j,k,m}$ where $j+k+m$ assumes *even* values, and likewise, let V_x denote the vector with zero components at all grid points $P_{j,k,m}$ where $j+k+m$ assumes *odd* values. Then we may define the odd–even hopscotch (OEH) method by

$$A(t, C(t)) := F_o(t, C(t)), \qquad B(t, C(t)) := F_x(t, C(t)). \tag{3.16}$$

On substitution into (3.13), we find that {(3.13), (3.16)} can be represented in the form

$$\begin{aligned} &C_{x,n+1/2} = C_{x,n} + \tfrac{1}{2}\Delta t\, F_x(t_n, C_n) = C_{x,n} + \left(C_{x,n} - C_{x,n-1/2}\right), \\ &C_{o,n+1/2} = C_{o,n} + \tfrac{1}{2}\Delta t\, F_o(t_{n+1/2}, C_{n+1/2}), \\ &C_{o,n+1} = C_{o,n+1/2} + \tfrac{1}{2}\Delta t\, F_o(t_{n+1/2}, C_{n+1/2}) = C_{o,n+1/2} + \left(C_{o,n+1/2} - C_{o,n}\right), \\ &C_{x,n+1} = C_{x,n+1/2} + \tfrac{1}{2}\Delta t\, F_x(t_{n+1}, C_{n+1}). \end{aligned} \tag{3.17}$$

Since only scalarly implicit relations are to be solved, the OEH method is hardly more expensive than a one-stage explicit RK method. However, in this case were A and B have zero rows, we cannot use the local-mode analysis as is the case for LOD methods (see e.g. [16] for a discussion on the local-mode analysis). Hence, we cannot simply require that the eigenvalues of $R_A(\Delta t)$ and $R_B(\Delta t)$ are within the unit circle to obtain the stability condition for the OEH method. A more sophisticated analysis is necessary. For the model problem (2.5), such an analysis has been carried out in [7]. Since the costs per step of the OEH methods are roughly equivalent to only one function call, the OEH method may offer an alternative to the RK24 method.

3.4.2. *Odd–even line hopscotch splitting*

Next, let V_o and V_x denote the vectors with zero components at grid points $P_{j,k,m}$ where $j+k$ assumes even and odd values, respectively. As before, we define A and B according to (3.16). The resulting OELH method can again be represented in the form (3.17). Evidently, the implicit equations for $\boldsymbol{C}_{o,n+1/2}$ and $\boldsymbol{C}_{x,n+1}$ become sets of tridiagonally implicit relations that can be solved efficiently by using the vectorization-across-the-tridiagonal-solvers technique.

The derivation of the stability condition for the OELH method is possible along the lines of the OEH analysis given in [7].

3.5. *Nested operator splitting based on ADI*

Suppose that we define the operators A and B in (3.12) by

$$\begin{aligned} A(t, \boldsymbol{C}(t)) &:= A_x(t)\boldsymbol{C}(t) + A_y(t)\boldsymbol{C}(t) + \boldsymbol{B}_x(t) + \boldsymbol{B}_y(t),\\ B(t, \boldsymbol{C}(t)) &:= A_z(t)\boldsymbol{C}(t) + N_z(\boldsymbol{C}(t))\boldsymbol{C}(t) + \boldsymbol{G}(t, \boldsymbol{C}(t)) + \boldsymbol{B}_z(t). \end{aligned} \tag{3.18}$$

Since this splitting allows the application of the standard normal mode analysis, we conclude that we have unconditional stability if the eigenvalues of $A_x + A_y$ and $A_z + N_z + \partial\boldsymbol{G}/\partial\boldsymbol{C}$ are in the left halfplane. However, this is only true if the two stages of (3.13) are really solved for $\boldsymbol{C}_{n+1/2}$ and $\boldsymbol{C}_{n+1}$. Let us solve these two stages iteratively by operator splitting to obtain a "nested" splitting method. In particular, we may apply ADI iteration. For the first stage, this yields

$$\begin{aligned} \boldsymbol{C}^{(j+1/2)} - \tfrac{1}{2}\Delta t\, A_x \boldsymbol{C}^{(j+1/2)} &= \boldsymbol{C}_n + \tfrac{1}{2}\Delta t\big[A_y\boldsymbol{C}^{(j)} + B(t_n, \boldsymbol{C}_n) + \boldsymbol{B}_x + \boldsymbol{B}_y\big],\\ \boldsymbol{C}^{(j+1)} - \tfrac{1}{2}\Delta t\, A_y \boldsymbol{C}^{(j+1)} &= \boldsymbol{C}_n + \tfrac{1}{2}\Delta t\big[A_x\boldsymbol{C}^{(j+1/2)} + B(t_n, \boldsymbol{C}_n) + \boldsymbol{B}_x + \boldsymbol{B}_y\big], \end{aligned} \tag{3.19a}$$

where the arrays A_x, A_y, $\boldsymbol{B}_x$ and $\boldsymbol{B}_y$ are assumed to be evaluated at $t_{n+1/2}$. The iteration error of this inner iteration method satisfies the recursion

$$\begin{aligned} \boldsymbol{C}^{(j+1)} - \boldsymbol{C}_{n+1/2} &= Q(\Delta t)\big[\boldsymbol{C}^{(j)} - \boldsymbol{C}_{n+1/2}\big],\\ Q(\Delta t) &:= \tfrac{1}{4}(\Delta t)^2\big[I - \tfrac{1}{2}\Delta t A_y\big]^{-1} A_x \big[I - \tfrac{1}{2}\Delta t A_x\big]^{-1} A_y, \end{aligned} \tag{3.20}$$

so that we have a comparable situation as in the stability recursion (3.14), that is, we have convergence if the eigenvalues of the matrix

$$Q^*(\Delta t) := \tfrac{1}{4}(\Delta t)^2 A_x\big[I - \tfrac{1}{2}\Delta t A_x\big]^{-1} A_y\big[I - \tfrac{1}{2}\Delta t A_y\big]^{-1} \tag{3.21}$$

are within the unit disk. Since it is now justified to apply the normal mode analysis, we conclude that we have unconditional convergence if A_x and A_y have their normal mode eigenvalues in the left halfplane. Furthermore, the low wavenumber components are quickly removed from the iteration error for sufficiently small Δt. However, for the high wavenumber components, we have $Q^*(\Delta t) \approx 1$, so that these components are hardly damped in the iteration process. A possible remedy is to use smoothing operators for damping of the highly oscillatory components. In [13], a detailed analysis on general smoothing techniques is given, and an extremely cheap implementation on vector computers can be found in [11].

The second stage of (3.13) can be solved in a similar way by splitting B according to (3.18):

$$\begin{aligned} &C^{(j+1/2)} - \tfrac{1}{2}\Delta t[A_z + N_z]C^{(j+1/2)} \\ &\quad = C_{n+1/2} + \tfrac{1}{2}\Delta t\left[A(t_{n+1/2}, C_{n+1/2}) + G(t_{n+1}, C^{(j)}) + B_z\right], \\ &C^{(j+1)} - \tfrac{1}{2}\Delta t\, G(t_{n+1}, C^{(j+1)}) \\ &\quad = C_{n+1/2} + \tfrac{1}{2}\Delta t\left[A(t_{n+1/2}, C_{n+1/2}) + A_z C^{(j+1/2)} + N_z C^{(j+1/2)} + B_z\right], \end{aligned} \tag{3.19b}$$

where now the arrays A_z, N_z, and B_z are evaluated at t_{n+1}. We remark that the matrix N_z in the first stage of this iteration process can be replaced by $N_z(C_n)$ without reducing the order of accuracy. Again, we arrive at the convergence condition requiring that the normal mode eigenvalues of $A_z + N_z$ and $\partial G/\partial C$ should be in the left halfplane.

We remark that one iteration in each of the four stages of {(3.19a), (3.19b)} suffices to achieve second-order accuracy. The initial iterate in (3.19a) may be defined by C_n and in (3.19b) by the result of (3.19a). The resulting four-stage nested ADI method (3.19) differs from the four-stage LOD method (3.4) by the right-hand sides and by the factor $\frac{1}{2}\Delta t$ instead of Δt in the left-hand sides.

3.6. Predictor–corrector methods

The methods reviewed so far are either unconditionally, strongly stable but only first-order in time, or second-order in time but at best marginally stable. In this section, we introduce predictor–corrector-type methods that are both second-order in time and unconditionally, strongly stable. For the corrector we choose the second-order backward differentiation formula (BDF):

$$C_{n+1} = L_n(C_{n+1}), \quad L_n(C) := -\tfrac{1}{3}C_{n-1} + \tfrac{4}{3}C_n + \tfrac{2}{3}\Delta t\, F(t_{n+1}, C). \tag{3.22a}$$

Let F be split according to (3.12). Then, we iterate this BDF corrector by means of the iteration scheme

$$\left.\begin{aligned} &C^{(j+1/2)} - \tfrac{2}{3}\Delta t\, A(t_{n+1}, C^{(j+1/2)}) = L_n(C^{(j)}) - \tfrac{2}{3}\Delta t\, A(t_{n+1}, C^{(j)}), \\ &C^{(j+1)} - \tfrac{2}{3}\Delta t\, B(t_{n+1}, C^{(j+1)}) = L_n(C^{(j+1/2)}) - \tfrac{2}{3}\Delta t\, B(t_{n+1}, C^{(j+1/2)}), \end{aligned}\right. \quad j = 0, 1, \ldots. \tag{3.22b}$$

Clearly, if this iteration method converges, then it converges to the second-order corrector solution, and at the same time, it generates an unconditionally L-stable solution.

In first approximation, the condition for convergence is obtained by linearization of A and B. Then, the iteration error satisfies

$$C^{(j+1)} - C_{n+1} = Q(\Delta t)\left[C^{(j)} - C_{n+1}\right],$$

$$Q(\Delta t) := \tfrac{4}{9}(\Delta t)^2\left[I - \tfrac{2}{3}\Delta t\, B\right]^{-1} A \left[I - \tfrac{2}{3}\Delta t\, A\right]^{-1} B. \tag{3.23}$$

Again assuming that the normal mode analysis can be applied, that is, A and B share the same eigensystem, we conclude that we have unconditional convergence if the normal mode eigenvalues are in the left halfplane (compare the discussion of (3.20)).

The actual solution of the stages in (3.22) again requires an (inner) iteration process. For example, if A and B are defined according to (3.18), then the inner iteration method can be defined as in the nested ADI method described in Section 3.5. We remark that the hopscotch splittings will not lead to unconditional convergence.

In actual computation, we hope to restrict the computational effort to only a few iterations. As to the order of accuracy, this is justified, because, assuming that the predictor formula providing the initial iterate $C^{(0)}$ is at least of zero order, all iterates $C^{(j)}$ with $j \geqslant 1$ are already second-order accurate. However, what about the stability when not iterating to convergence. We shall consider the stability of (3.22) after one full iteration, that is the stability of $C_{n+1} = C^{(1)}$. Assuming that the operators A and B in the splitting (3.12) are linear, we may write

$$\Delta L_n(C) := -\tfrac{1}{3}\Delta C_{n-1} + \tfrac{4}{3}\Delta C_n + \tfrac{2}{3}\Delta t[A + B]\Delta C,$$

$$\begin{aligned}\Delta C^{(1/2)} &= \left[I - \tfrac{2}{3}\Delta t\, A\right]^{-1}\left[\Delta L_n(C^{(0)}) - \tfrac{2}{3}\Delta t\, A \Delta C^{(0)}\right] \\ &= \tfrac{1}{3}\left[I - \tfrac{2}{3}\Delta t\, A\right]^{-1}\left[-\Delta C_{n-1} + 4\Delta C_n + 2\Delta t\, B\Delta C^{(0)}\right],\end{aligned}$$

$$\begin{aligned}\Delta C_{n+1} &= \left[I - \tfrac{2}{3}\Delta t\, B\right]^{-1}\left[\Delta L_n(C^{(1/2)}) - \tfrac{2}{3}\Delta t\, B \Delta C^{(1/2)}\right] \\ &= \tfrac{1}{3}\left[I - \tfrac{2}{3}\Delta t\, B\right]^{-1}\left[-\Delta C_{n-1} + 4\Delta C_n + 2\Delta t\, A\Delta C^{(1/2)}\right].\end{aligned}$$

Hence,

$$9\left[I - \tfrac{2}{3}\Delta t\, A\right]\left[I - \tfrac{2}{3}\Delta t\, B\right]\Delta C_{n+1} = 3\left[-\Delta C_{n-1} + 4\Delta C_n\right] + 4(\Delta t)^2 AB\Delta C^{(0)}. \tag{3.24}$$

Let us assume that $C^{(0)}$ is computed by a one-step formula satisfying the variational equation

$$\Delta C^{(0)} = P\Delta C_n,$$

where P is a given matrix. Let us assume that A, B and P are Toeplitz matrices and have a common eigensystem. Then, we may expand C_n in terms of $\zeta^n V$, where V is an eigenvector of A, B and P with eigenvalues $\lambda(A)$, $\lambda(B)$ and $\lambda(P)$, and ζ^n is the corresponding coefficient. On substitution into (3.24) we obtain the characteristic equation

$$[3 - 2\Delta t\, \lambda(A)][3 - 2\Delta t\, \lambda(B)]\zeta^2 - 4\left[3 + (\Delta t)^2\lambda(ABP)\right]\zeta + 3 = 0. \tag{3.25}$$

As an example, we consider the splitting (3.12) in which A corresponds to the diffusion part and B to the advection part of the equation. Hence, $\lambda(A)$ and $\lambda(B)$ are assumed to be real and purely imaginary, respectively. Defining $x := \Delta t\ \lambda(A)$, $\mathrm{i}y := \Delta t\ \lambda(B)$ and choosing the "trivial" prediction $C^{(0)} = C_n$ (i.e., $P = I$), the characteristic equation reads

$$[3-2x][3-2\mathrm{i}y]\zeta^2 - 4[3+\mathrm{i}xy]\zeta + 3 = 0. \tag{3.25$'$}$$

Applying Schur's theorem (cf., e.g., [4, p. 299]), a straightforward, but tedious, calculation leads to the conclusion that – for all nonpositive x – the roots ζ of (3.25′) satisfy $|\zeta| < 1$ (with the exception of the origin, of course, where we have one characteristic root equal to 1). Since the eigenvalues of the discretized diffusion operator are negative indeed, we conclude that this predictor–corrector method with the above splitting is unconditionally stable. Finally, it is of interest to remark that one characteristic root $\zeta \to 1$ if *both* x and y tend to infinity. However, along the axis ($x = 0$ or $y = 0$), both characteristic roots tend to zero. Hence, in the case of purely advection or purely diffusion, this method is L-stable.

3.7. Summary of characteristics of the time integrators

In Table 1, we summarize the main properties of the various integrators discussed in the preceding sections.

4. Numerical experiments

To test the most promising methods surveyed in Section 3.7, we take the following example problem

$$\frac{\partial c}{\partial t} + \frac{\partial(uc)}{\partial x} + \frac{\partial(vc)}{\partial y} + \frac{\partial(wc)}{\partial z} = \varepsilon\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)c + g(t, x, y, z), \quad 0 \leqslant t \leqslant T, \tag{4.1a}$$

with Neumann conditions of the form $\partial c/\partial n = h(t, x, y, z)$ on all boundaries of the physical domain $0 \leqslant x \leqslant L_x$, $0 \leqslant y \leqslant L_y$, $-L_z \leqslant z \leqslant 0$. In our experiments, it is convenient to have the exact solution at our disposal. Therefore, we prescribe the concentration c as

$$c(t, x, y, z) = \exp\left\{\frac{z}{L_z} - \frac{t}{T^\gamma} - \left(1 - \frac{t}{T^\gamma}\right)\left[\left(\frac{x}{L_x} - \frac{1}{2}\right)^2 + \left(\frac{y}{L_y} - \frac{1}{2}\right)^2\right]\right\}, \tag{4.1b}$$

where $L_x = L_y = 20{,}000$ [m], $L_z = 100$ [m], and γ is a (dimensionless) parameter which is set to 1.05. The concentration and the diffusion coefficient (to be specified in the tables of results) are assumed to be measured in kg/m^3 and m^2/s, respectively.

We distinguish two cases, referred to as the *small-scale problem* and the *large-scale problem.* In the small-scale problem, the spatial grid is defined by $d_x = d_y = 41$, $d_z = 11$, and the fluid velocities u, v and w are kept *constant* with values to be specified in the tables of results. This problem will serve to get insight into the stability behaviour and the effect of the diffusion coefficient on the performance of the selected methods. In the large-scale problem, the grid

Table 1
Characteristics of the time integrators

Method	RK24	OEH	OELH	LOD1
Formula	(3.1)	{(3.13), (3.16)}	{(3.13), (3.16)}	(3.4)
Δt condition	conditional	conditional	conditional	unconditional
Stability	strong	weak	weak	strong
Order of accuracy in time	2	2	2	1
Effective $F(t, C)$ calls per step	2	1	1	1
Tridiagonal systems per step	none	none	$d_x d_y$	$d_x d_y + d_x d_z + d_y d_z$
Vectorization aspects	+ +	+	+	+ +
Parallelization aspects	+ +	+	+	+ +
Ease of implementation	+ +	+	+ / −	−
Number of arrays [a]	3	4	6.5	13
Method	**LOD2 [b]**	**LOD3 [b]**	**Nested ADI**	**BDF-ADI**
Formula	{(3.4), (3.7a)}	{(3.4), (3.7b)}	(3.19)	(3.22)
Δt condition	unconditional	unconditional	unconditional	unconditional
Stability	strong	strong	weak	strong
Order of accuracy in time	2	3	2	2
Effective $F(t, C)$ calls per step	2	3	2	$\geqslant 2$
Tridiagonal systems per step	$2(d_x d_y + d_x d_z + d_y d_z)$	$3(d_x d_y + d_x d_z + d_y d_z)$	$d_x d_y + d_x d_z + d_y d_z$	$d_x d_y + d_x d_z + d_y d_z$
Vectorization aspects	+ +	+ +	+ +	+ +
Parallelization aspects	+ +	+ +	+	+
Ease of implementation	−	−	−	−
Number of arrays [a]	26	27	13	14

[a] Excluding the storage needed to store the velocity field (u, v, w).
[b] For the LOD2 and LOD3 methods it is assumed that the terms in (3.7a) and (3.7b) are computed *concurrently*. Implementing these methods *sequentially* would reduce the required number of arrays to 14 and 15, respectively. However, in that case, the number of effective F-calls and tridiagonal systems per step should be multiplied by a factor 1.5 for the LOD2 method, and by a factor 2 for the LOD3 method.

parameters are given by $d_x = d_y = 101$ and $d_z = 11$, amounting to 10^5 grid points. The velocity fields are prescribed by the analytical expressions

$$\begin{aligned} u(t, x, y, z) &= C_1 \sin\left(\frac{x}{L_x} + \frac{y}{L_y}\right) \sin\left(\beta \frac{z}{L_z}\right) f(t), \\ v(t, x, y, z) &= C_2 \cos\left(\frac{x}{L_x} + \frac{y}{L_y}\right) \sin\left(\beta \frac{z}{L_z}\right) f(t), \\ w(t, x, y, z) &= \left[-\frac{C_1}{L_x} \cos\left(\frac{x}{L_x} + \frac{y}{L_y}\right) + \frac{C_2}{L_y} \sin\left(\frac{x}{L_x} + \frac{y}{L_y}\right)\right]\left[-\frac{L_z}{\beta} \cos\left(\beta \frac{z}{L_z}\right)\right] f(t), \end{aligned} \tag{4.2}$$

with $f(t) = \cos(t/T)$ and $C_1 = 3$, $C_2 = 4$, $\beta = 0.05$. These fluid velocities satisfy the relation for local mass balance, i.e.,

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0.$$

Table 2
Global errors for the small-scale problem with $\varepsilon = 0$

Case	$u = v$	w	T	Δt	$\Delta t\, u/\Delta x$	$\Delta t\, w/\Delta z$	OELH	LOD3	LOD2	LOD1	RK24
I	1	1	1000	100	0.2	10	0.011	0.011	0.084	4.1	*
				50	0.1	5	0.0085	0.0085	0.022	2.1	*
				25	0.04	2	0.0077	0.0079	0.0062	0.84	0.0071
II	10	1	2000	50	1.0	5	0.021	0.022	0.024	5.7	*
	11	1	2000	50	1.1	5	*	0.020	0.023	5.6	*
III	1	5	2000	50	0.1	25	0.078	0.096	0.43	*	*
			4000	50	0.1	25	0.16	0.17	0.52	*	*
	1	10	4000	50	0.1	50	0.31	0.32	2.3	*	*
	1	20	4000	50	0.1	100	0.62	0.62	9.7	*	*
IV	1	0.5	2000	50	0.1	2.5	0.0080				0.0076
			4000	50	0.1	2.5	0.016				0.016
			8000	50	0.1	2.5	0.033				0.033
	3	0.5	2000	50	0.3	2.5	0.0083				0.0078
			4000	50	0.3	2.5	0.019				1.54
			8000	50	0.3	2.5	0.036				*

The computational complexity of this test problem is substantial and resembles that of the large-scale problems occurring in realistic situations.

In both test problems, the source function g in (4.1a) and the function h defining the boundary conditions follow from the exact solution (4.1b).

4.1. Stability tests in the pure advection case

Since a vanishing diffusion coefficient yields the most stringent case as far as stability is concerned and the most difficult case with respect to accuracy, we will first give results for $\varepsilon = 0$. Table 2 lists the global errors (with respect to the PDE solution (4.1b)) for various methods when applied to the small-scale problem for different values of the fluid velocities.

From this table we see that all LOD methods behave stably, independent of the values of the fluid velocities and the size of the time step. It is clear, however, that the first-order LOD method is not sufficiently accurate.

From the results given in the cases II and III the stability constraints for the OELH method are also clear: since this method is implicit in the vertical, large values for $\Delta t\,|w|/\Delta z$ do not cause any stability problems. We might say that the OELH method is "unconditionally stable in the vertical" (cf. case III). This is in sharp contrast with its behaviour in the horizontal; here we meet the constraint $\Delta t \max(|u|/\Delta x, |v|/\Delta y) \leqslant 1$ (see case II; an $*$ denotes an unacceptable performance). However, in many practical situations this condition on the time step is not a very severe restriction. Moreover, if the OELH method behaves stably, then it is very accurate; the errors are similar to those of LOD3, but OELH is much cheaper per step.

The stability behaviour of RK24 nicely obeys the theoretical considerations discussed in Section 3.1. The experiments given in case IV show that RK24 behaves unstably as soon as $\Delta t(|u|/\Delta x + |v|/\Delta y + |w|/\Delta z)$ gets larger than $2\sqrt{2}$. Note that, with respect to the horizontal spatial discretization this condition is less restrictive than that of OELH; however, since the

Table 3
Global errors for the small scale problem with $\varepsilon = 0.5$

Case	$u=v$	w	T	Δt	$\Delta t u/\Delta x$	$\Delta t w/\Delta z$	OELH	LOD3	LOD2	LOD2	RK24
I	1	1	1000	100	0.2	10	0.00086	0.0048	0.037	0.37	*
				50	0.1	5	0.00072	0.0025	0.017	0.18	*
				25	0.04	2	0.00067	0.00074	0.0042	0.073	0.0071
II	10	1	2000	50	1.0	5	0.00067	0.0026	0.018	0.18	*
	11	1	2000	50	1.1	5	*	0.0026	0.018	0.18	*
III	1	5	2000	50	0.1	25	0.0017	0.032	0.20	2.5	*
			4000	50	0.1	25	0.0016	0.032	0.20	2.5	*
	1	10	4000	50	0.1	50	0.0029	0.091	0.75	9.2	*
	1	20	4000	50	0.1	100	0.0055	0.32	3.00	35.1	*
IV	1	0.5	8000	50	0.1	2.5	0.00067				0.00067
	3	0.5	8000	50	0.3	2.5	0.00067				0.00067
	5	0.5	8000	50	0.5	2.5	0.00070				*

most stringent stability condition is usually imposed by the vertical spatial discretization, OELH is to be preferred.

4.2. *The effect of diffusion on stability and accuracy*

Next we illustrate the effect of adding diffusion in the model problem. Its influence on the stability is quite small (with the current values of Δx, Δy and Δz), but the effect on the accuracy is significant (see Table 3).

Again, the accuracy of the LOD1 method is not satisfactory, for the LOD2 method it is acceptable, and the extension to the third-order variant is worth the extra effort. The OELH — whenever stable — yields a much better accuracy. The same is true for the RK24 method, but this method is subject to a very stringent stepsize restriction. We remark that, due to stability conditions, the errors obtained by OELH and RK24 are dominated by the *spatial* discretization error, whereas the accuracies produced by the LOD methods still contain a substantial contribution from the time integration. Furthermore, for RK24 we observe the small stabilizing effect of a nonvanishing diffusion, which is in correspondence with the well known shape of its stability region.

4.3. *The large-scale problem*

In Table 4 we give the results of a selected number of time integration techniques when applied to the large-scale problem {(4.1), (4.2)} with $\varepsilon = 0.5$ and $T = 10{,}000$.

For this problem the spatial discretization error is given by 0.00064. Table 4 shows that the errors of all methods converge to this value for $\Delta t \to 0$. However it is clear that the OELH is the most accurate time integration process, since already for Δt as large as 500, the time integration error is almost negligible compared to the spatial error. Again we observe that the first-order LOD1 method (and to some extent the second-order variant as well) needs very small time steps to let the temporal error be of the same order as the spatial error. If the RK24

Table 4
Global errors for the large-scale problem {(4.1), (4.2)} with $\varepsilon = 0.5$ and $T = 10{,}000$

No. steps	Δt	OELH	LOD3	LOD2	LOD1	RK24
10	1000	0.00089	0.13	0.50	1.81	*
20	500	0.00069	0.030	0.15	0.65	*
40	250	0.00065	0.0062	0.044	0.27	*
80	125	0.00064	0.0013	0.012	0.12	*
160	62.5		0.00063	0.0033	0.056	0.00064
320	31.2		0.00064	0.00093	0.027	0.00064
640	15.6			0.00062	0.013	
1280	7.8				0.0068	

method is stable, then its accuracy is sufficient, however a stable behaviour requires a (too) small time step. Again, the use of smoothing techniques may be helpful in such situations (see [11,13,18]).

In conclusion, for this problem, the OELH method is by far superior, especially if we take into account the required computational effort per step.

5. Implementational aspects

In Section 4 we have discussed several experiments in which we focused on the numerical properties of the various methods (viz., stability and accuracy). The major aim of these tests was to select an appropriate time integration technique, of course, also taking into account the parallelization/vectorization properties that the particular methods possess (cf. Table 1). The actual implementation of the integrator on a CRAY-type machine is described in [6].

From the results presented in Section 4, we draw the conclusion that an *implicit* method is most suitable to perform the time integration. Within this class of methods, the OELH and LOD3 method seem to be very promising candidates. A common feature of both methods is that they spend most of their time in solving the tridiagonal systems. We recall that the LOD method has to solve such systems in all three spatial directions, whereas the OELH method is only implicit in the vertical direction, requiring the solution of $d_x d_y$ systems of dimension d_z. Therefore, an efficient implementation of this part of the solution process is of crucial importance and will be discussed in the next subsections.

5.1. Vectorization-across-tridiagonal-systems

Here we will give a brief outline of the vectorization-across-tridiagonal-systems approach; details can be found in [6].

Both the LOD and OELH integrator give rise to the frequent solution of tridiagonal systems of the form

$$Tx = b, \tag{5.1a}$$

where T is a block-tridiagonal matrix of the special form

$$T = I - \Delta t\, J, \tag{5.1b}$$

I being the identity matrix and J denoting the Jacobian matrix resulting from solving the implicit relation using Newton's method.

Due to the special properties of both time integration techniques, (5.1) represents a large number of *uncoupled* tridiagonal systems (cf. Table 1 for details). A straightforward approach to solve N uncoupled systems, each of dimension d, would be

ALG1.

```
for $i = 1$ until $N$
    factorize each of the $d \times d$ blocks into $LU$
    and solve the corresponding $LUx = b$.
```

Although this i-loop is perfectly parallelizable, the *recursive* nature of the body of this loop prevents vectorization and hence ALG1 will result in a bad performance on (shared memory) vector machines.

A great improvement can be obtained by interchanging the loops [2, p. 156]:

ALG2.

```
for $j = 1$ until $d$
    perform for each of the $N$ systems the required
    calculations in the factorization and
    forward substitution process.

for $j = d$ until 1 with step $-1$
    perform for each of the $N$ systems the
    backward substitution process.
```

Now, the bodies of these loops, which involve vectors of length N, are very well suited for vectorization. Since usually $N \gg d$, we may expect a good performance. Algorithm ALG2 has been implemented on the CRAY YMP4 and the results (both for scalar mode and vector mode) are given in Table 5. In this table we list the CPU time and the associated Mflop rate for a tridiagonal system corresponding to a physical domain with $d_x = d_y = 101$ and $d_z = 11$ (cf. the large-scale problem discussed in Section 4).

Table 5
CPU times (in milli-seconds) and Mflop rates on a CRAY YMP4, using 1 processor

Scalar mode		Vector mode		Speedup
CPU	Mflops	CPU	Mflops	
102.0	17.1	8.9	197.5	11.5

In our implementation the tridiagonal matrix J is provided by means of three arrays, housing the lower-, main- and upper diagonal elements, respectively. Each of these arrays has three subscripts, two of these correspond to a particular point in the plane and the third one runs along the elements of the particular system. As a consequence, the *ordering* of these subscripts has a serious impact on the performance (recall that LOD-type methods have to solve systems in all three spatial directions). Therefore, special provisions have been made to cope with this difficulty, resulting in an almost constant performance, independent of the particular ordering; full details of these provisions are discussed in [6], where the performance of the *total* solution process on a CRAY machine is described.

When comparing the results for scalar and vector mode, we observe a speedup of 11.5, which is in good agreement with what is expected for the CRAY YMP4 (cf. CF77 Compiling System: Fortran Reference Manual). To appreciate the obtained Mflop rates, we remark that the peak performance for this configuration (with one processor) is about 330 Mflops. Note however, that this number is based on the ideal situation: infinite loops in which the arithmetic operations can be combined to so-called "linked triads" (giving a speedup with a factor 2). However, in the current algorithm this facility can only partly be utilized. Taking this into account, the given Mflop rates indicate that the vectorization-across-tridiagonal-systems approach performs quite efficiently.

5.2. Further refinements

To reduce storage requirements, our tridiagonal solver TRIDIA overwrites the matrix J with the decomposition of $I-\Delta t\, J$. However, in a realistic implementation of a time integration process the time step will change, due to error control. For such situations TRIDIA has an option to reconstruct the matrix J from the stored decomposition of $I-\Delta t_{\text{old}}\, J$, decompose $I-\Delta t_{\text{new}}\, J$, and solve the system. Apparently, this call of TRIDIA (which will be referred to as "$\text{call}_{\text{newstep}}$") is more expensive than the standard call ("$\text{call}_{\text{standard}}$") with only decomposition and solution. Therefore, the user should only resort to this possibility when it would be more expensive (or not at all possible) to keep copies of the original matrix J. Note that if J has changed, the only feasible way to call TRIDIA is using the "standard" option. Finally, we provided TRIDIA with a third option, which can be used when both Δt and J remain unchanged. In this case the LU-decomposition part of the algorithm can be skipped and only the forward/backward substitutions are needed to find the solution. This option is denoted by "$\text{call}_{\text{save}}$". In Table 6, we present results of the tests on the CRAY YMP4. This table shows that

Table 6
CPU times (in milli-seconds) and Mflop rates on a CRAY YMP4 (using 1 processor) for various options

Option	Scalar mode		Vector mode		Speedup
	CPU	Mflops	CPU	Mflops	
$\text{call}_{\text{standard}}$	102.0	17.1	8.9	197.5	11.5
$\text{call}_{\text{save}}$	58.7	9.3	3.7	147.9	15.9
$\text{call}_{\text{newstep}}$	184.0	16.2	16.5	180.0	11.2

we obtain a satisfactory speedup, also for the nonstandard options (notice that the results of Table 5 are reproduced with the label $\text{call}_{\text{standard}}$).

6. Summary and future research

In this paper we have discussed several possible time integration techniques for the efficient solution of a transport model in three spatial dimensions. The first class of methods considered are the stabilized Runge–Kutta (RK) methods. Since these integration schemes are explicit, they allow for a straightforward and efficient implementation on vector machines. In spite of the fact that these RK methods are especially tuned to the transport problem, in the sense that they have been given optimal stability properties, they have to obey a stepsize condition which is still rather restrictive. For that reason, we have also considered implicit methods; although the stability behaviour of such methods is usually appropriate for the present application, the resulting systems of equations that have to be solved in each time step give rise to a serious obstacle. To reduce the amount of linear algebra involved, we restricted our consideration to splitting methods (both operator splitting and dimension splitting). A common feature of these methods is that implicit relations have to be solved with at most a three-point coupling. The standard approach to solve "tridiagonal" systems is recursive in nature, and therefore inappropriate for vector machines. Following a technique, initially proposed by Golub and Van Loan [2], it is possible to enhance the performance of this part of the algorithm by approximately a factor 12. Owing to this technique, the amount of linear algebra is reduced to such an extent that the implicitness in these "tridiagonally" implicit methods is feasible.

We started our survey with the locally one-dimensional (LOD) methods. In spite of their excellent stability properties, the accuracy of these methods is unacceptably low (since they are only first-order accurate in time). Therefore, we considered the remedy of Richardson extrapolation, resulting in second- and third-order methods. These methods possess (almost) the same good stability properties as the first-order LOD method, show an increased accuracy indeed, but are very expensive per step and moreover, they require an enormous amount of storage.

Next, we considered methods based on operator splitting, which means that the terms in the right-hand-side function originating from the advection and from the diffusion are separated. Choosing suitable integrators for each of these "fractional equations" may lead to efficient solvers.

The following class of methods considered are the so-called hopscotch methods. In particular, the odd–even line hopscotch (OELH) variant seems to be very attractive, since this method is only implicit in the vertical direction. This property is sufficient to maintain overall stability, since, in the present application, the most restrictive condition on the time step is imposed by the discretization in the vertical direction.

Subsequently, nested operator splitting methods based on ADI are discussed, and finally, we analyse a predictor-corrector approach based on a second-order backward differentiation formula (BDF) as corrector (to achieve the required accuracy and stability). This BDF is solved by an iteration process in which we employ (nested) operator splitting to reduce the implicitness.

All the aforementioned methods were evaluated with respect to their numerical properties and their suitability for efficient implementation on vector/parallel machines. From this evaluation we concluded that the stabilized RK methods, the LOD-type methods and the OELH scheme are the most promising. These schemes were applied to a large-scale test example and their (numerical) behaviour was examined. From this comparison we concluded that the OELH method seems to be the most efficient technique for the present application since it possesses sufficient accuracy and stability, the storage requirements are quite low, and the vectorization and parallelization capabilities are almost optimal for a CRAY-type machine. Here we remark that an extensive performance evaluation of the RK methods and the OELH method can be found in [6]. The conclusion in that paper is that both types of methods vectorize extremely well, but that the OELH method is to be preferred because of its better numerical (viz. stability) properties.

Finally, we remark that the OELH method described in the present paper is only feasible in the case that a three-point stencil is used for the discretization of the spatial differential operators (e.g., symmetric, second-order differences). It is well known, however, that symmetric differences are all right for smooth solutions, but usually lead to so-called "wiggles" whenever the solution has large spatial gradients. These wiggles may easily lead to negative concentrations, which is disastrous when we add chemical terms to the model. In such situations, upwind discretizations are much more appropriate. However, third-order upwinding will lead, in general, to a five-point coupling in each spatial direction. This has two consequences: (i) to be able to exploit the underlying idea of the hopscotch splitting (i.e., uncoupling in the horizontal), the grid points in each horizontal plane have to be divided into three subsets; a next step is the construction of a splitting method (based on a splitting of the right-hand side function into three corresponding functions), which possesses the proper numerical characteristics; and (ii) the technique of "vectorization-across-the-tridiagonal-systems" should be extended to "vectorization-across-the-fivediagonal-systems". The last aspect is relatively easy; the construction of a new splitting method is certainly not trivial and is subject of present research.

Acknowledgement

The authors are grateful to the referee for carefully reading the manuscript and for the many valuable suggestions for improvement.

References

[1] E.D. de Goede, Numerical methods for the three-dimensional shallow water equations on supercomputers, Thesis, University of Amsterdam (1992).

[2] G.H. Golub and C.F. Van Loan, *Matrix Computations* (The Johns Hopkins Press, Baltimore, MD, 2nd ed., 1989).

[3] A.R. Gourlay, Hopscotch: a fast second order partial differential equation solver, *J. Inst. Math. Appl.* 6 (1970) 375–390.

[4] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics 14 (Springer, Berlin, 1989).

[5] W.H. Hundsdorfer and R.A. Trompert, Method of lines and direct discretization — a comparison for linear advection, Report NM-R9314, CWI, Amsterdam (1993).
[6] B.P. Sommeijer and J. Kok, Implementation and performance of the time integration of a 3D numerical transport model, Report NM-R9402, CWI, Amsterdam (1994); also: *Internat. J. Numer. Meth. Fluids* (to appear).
[7] J.H.M. ten Thije Boonkkamp and J.G. Verwer, On the odd–even hopscotch scheme for the numerical integration of time-dependent partial differential equations, *Appl. Numer. Math.* 3 (1988) 183–193.
[8] M. Toro, L.C. van Rijn and K. Meijer, Three-dimensional modelling of sand and mud transport in currents and waves, Technical Report No. H461, Delft Hydraulics, Delft, Netherlands (1989).
[9] P.J. van der Houwen, *Construction of Integration Formulas for Initial Value Problems,* North-Holland Series in Applied Mathematics and Mechanics 19 (North-Holland, Amsterdam, 1977).
[10] P.J. van der Houwen and B.P. Sommeijer, On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values, *Z. Angew. Math. Mech.* 60 (1980) 479–485.
[11] P.J. van der Houwen and B.P. Sommeijer, Improving the stability of predictor-corrector methods by residue smoothing, *IMA J. Numer. Anal.* 9 (1990) 371–378.
[12] P.J. van der Houwen and B.P. Sommeijer, Fractional Runge-Kutta methods with application to convection-diffusion equations, *Impact Comput. Sci. Engrg.* 4 (1992) 195–216.
[13] P.J. van der Houwen, B.P. Sommeijer and F.W. Wubs, Analysis of smoothing operators in the solution of partial differential equations by explicit difference schemes, *Appl. Numer. Math.* 6 (1989) 501–521.
[14] J.G. Verwer, Contractivity of locally one-dimensional splitting methods, *Numer. Math.* 44 (1984) 247–259.
[15] J.G. Verwer, W.H. Hundsdorfer and B.P. Sommeijer, Convergence properties of the Runge-Kutta-Chebyshev method, *Numer. Math.* 57 (1990) 157–178.
[16] R. Vichnevetsky and J.B. Bowles, *Fourier Analysis of Numerical Approximations of Hyperbolic Equations* (SIAM, Philadelphia, PA, 1982).
[17] C.B. Vreugdenhil and B. Koren, eds., *Numerical Methods for Advection-Diffusion Problems,* Notes on Numerical Fluid Mechanics 45 (Vieweg, Braunschweig, 1993).
[18] F.W. Wubs, Numerical solution of the shallow-water equations, Thesis, University of Amsterdam (1987).
[19] N.N. Yanenko, *The Method of Fractional Steps* (Springer-Verlag, Berlin, 1971).

Note added in proof

A thorough stability analysis of the OELH method can be found in: J.G. Verwer and B.P. Sommeijer, Stability analysis of an odd-even-line hopscotch method for three-dimensional advection–diffusion problems, Report NM-R9422, CWI, Amsterdam (1994).