A vector/parallel method for a three-dimensional transport model coupled with bio-chemical terms

B.P. Sommeijer and J. Kok

Department of Numerical Mathematics

NM-R9503 1995

Report NM-R9503
ISSN 0169-0388

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

A Vector/Parallel Method for a Three-Dimensional Transport Model Coupled with Bio-Chemical Terms

B.P. Sommeijer and J. Kok
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

A so-called fractional step method is considered for the time integration of a three-dimensional transport-chemical model in shallow seas. In this method, the transport part and the chemical part are treated separately by appropriate integration techniques. This separation is motivated by the fact that the coupling in both parts is completely different: in the transport part we have coupling over the grid points (due to advection-diffusion operators), but there is no mutual coupling between the various species. In the chemical part, the species are coupled (due to reaction terms) per grid point, but here the coupling in space is absent. To solve the transport part, we use a specially constructed method of Hopscotch-type. Since the chemistry in water is usually of low activity, an explicit Runge-Kutta method can be used for integrating the chemical part. We discuss the implementational aspects of these methods for multi-vectorprocessors, and show performance results obtained on a Cray C98/4256.

AMS Subject Classification (1991): 65M12, 65M20
CR Subject Classification (1991): G.1.7, G.1.8
Keywords and Phrases: Transport models, shallow water, 3D advection-diffusion-reaction equations, numerical time integration, vectorization, parallel processing.
Note: This work was sponsored by the Stichting Nationale Computerfaciliteiten (National Computing Facilities Foundation, NCF) who provided the authors with a grant from the Cray Research University Grants Program.

1. Introduction

Greatly inspired by the tremendous increase of the available computerpower, the development of sophisticated mathematical models for the numerical simulation of transport problems is an active field of research. Nevertheless, we are still far away from the final goal, i.e., a thorough understanding, and control, of the marine ecosystem dynamics in shallow seas.

Initially, in the seventies, two-dimensional hydrodynamical models were developed (see e.g. [11, 17]) and implemented on the fastest computers of those days. Later, mainly in the eighties, when computer power increased both with respect to speed and memory capacity, these models were extended to three spatial dimensions (see e.g. [2, 3, 12]). Fully vectorized codes for such problems are nowadays used on a routine basis and enabled a reduction of the computation time by roughly one order of magnitude (see e.g. [5]).

The second step is to couple the hydrodynamical model to a transport model. By this we mean that the output of the hydrodynamical model, i.e., the velocity field, is used as input for the calculation of the transport of salinity, pollutants or suspended sediments, which are carriers of nutrients and contaminants and consequently have a large impact on the ecosystem. Numerical models for transport simulation have been developed, say, during the last 10 years. Much effort has been spent on the algorithmic side (see e.g. [15]) as well as on the implementational aspects. With respect to the implementation, we can distinguish between shared-memory multi-vectorprocessors, mainly aiming at vectorization properties of the algorithms (see e.g. [16]), and message-passing distributed-memory architectures, where (massive) parallelism is the main goal (see e.g. [13]). Probably, a combination of both types of supercomputing will be characteristic for the next generation of high-performance machines. For a survey on parallel computations in the field of Computational Fluid Dynamics we refer to [14].

The next step is to model the concentration of *various* species in combination with their mutual biological or chemical interactions. The main purpose of this paper is to develop a suitable time integration technique for this kind of simulations and to implement the model on a multi-vectorprocessor.

The mathematical modeling of such transport phenomena involves the numerical solution of 3D partial differential equations of advection-diffusion-reaction type. The *total* solution of such a transport model has many different aspects, such as: the representation of the physical boundaries, which are typically irregular in these applications, the choice of the spatial discretization technique, the choice of the corresponding data structures (which has a significant influence on the performance), the treatment of the boundary conditions at the sea bed and at the free water surface, the choice of the time integration method, etc. In this paper we will focus on this last aspect, but it is clear that the tremendous computational task imposed by this kind of problems can only be tackled if *each* of these aspects is thoroughly investigated and treated by sophisticated numerical techniques in combination with the most powerful computers.

As we shall see in Section 4, the advection-diffusion part and the reaction part in the present model will be treated separately, due to a completely different coupling of the unknown concentrations in the discrete model. With respect to the choice of the time integration method for the advection-diffusion part we refer to previous work. In [15] and [16], we have studied various time integration techniques for the efficient solution of a *single* 3D transport problem. The conclusion formulated in these papers was that the Odd-Even Line Hopscotch (OELH) method is an excellent candidate, both with respect to algorithmic properties and because of its suitability to efficiently exploit the facilities offered by Cray machines. Therefore, we adopt this method to solve the advection-diffusion part in our present model. The new aspect of this paper is to combine the OELH method with a method that efficiently integrates the reaction part in the equations. We will study several ODE techniques for the bio-chemical reaction terms, discuss the interaction of these two solvers, and show performance results obtained on the CRAY C98/4256.

The paper is organized as follows: in Section 2, we will discuss the actual transport model that has been studied. Section 3 briefly outlines the spatial (or, semi-) discretization method and the consequences for the data structures that we need. In the next section, and this is the principal part of the present paper, the time integration methods are described, including the motivations for their choice. In Section 5 we briefly discuss how to solve, on full vector speed, the linear systems occurring in the Hopscotch integrator and the ODE systems originating from the chemical terms. The performance of the total algorithm is illustrated in Section 6 by applying it to a large-scale test example. A summary is formulated in Section 7.

2. Definition of the mathematical model

The mathematical model describing transport processes in three dimensions is given by the system of advection-diffusion-reaction equations [20]

$$\begin{aligned}(2.1)\qquad \frac{\partial c_i}{\partial t} = &-\frac{\partial}{\partial x}(uc_i) - \frac{\partial}{\partial y}(vc_i) - \frac{\partial}{\partial z}\big((w - w_f)c_i\big) + \\ &\frac{\partial}{\partial x}\Big(\varepsilon_x \frac{\partial c_i}{\partial x}\Big) + \frac{\partial}{\partial y}\Big(\varepsilon_y \frac{\partial c_i}{\partial y}\Big) + \frac{\partial}{\partial z}\Big(\varepsilon_z \frac{\partial c_i}{\partial z}\Big) + s_i\,(t, x, y, z) + \\ &g_i(t, x, y, z, c_1, c_2, \dots, c_m), \qquad i = 1, \dots, m,\end{aligned}$$

where c_i are the unknown concentrations of the contaminants. The local fluid velocities u, v, w (in x, y, z directions, respectively) have to be provided by a hydrodynamical model; since this paper merely discusses the solution of (2.1), the velocity field is considered to be known in advance. The fall velocity w_f, which may be a nonlinear function of the concentration, is only relevant in the case of modeling transport of suspended material. Sinks and sources (emissions) are modeled by the terms s_i ; the terms g_i describe bio-chemical reactions and therefore depend on the concentrations c_i. Hence, the mutual coupling of the equations in the system (2.1) is only due to these functions g_i. Finally, the diffusion coefficients ε_x, ε_y, and ε_z are assumed to be given functions.

The physical domain in space is bounded by vertical, closed boundary planes, by the water elevation surface, and by the bottom profile. On these boundaries, Dirichlet, Neumann or mixed boundary conditions have to be prescribed. Supplementing this with an initial condition, the concentrations c_i can be computed in space and time.

3. Spatial discretization

We will follow the method of lines (MOL) approach, that is, we first discretize the spatial derivatives in (2.1), followed by the numerical time integration of the resulting system of ordinary differential equations (ODEs).

Before applying the semi-discretization process, it is convenient to rewrite equation (2.1) by taking into account the particular applications we have in mind. As usual, we shall only consider the *incompressible* case, that is, we assume $u_x + v_y + w_z = 0$. Although not essential, the diffusion coefficients ε_x, ε_y, ε_z are assumed to be constant and equal to ε. Furthermore, in this paper we shall not take into account the fall velocity term w_f. Then, (2.1) simplifies to

$$\frac{\partial c_i}{\partial t} = -u\frac{\partial c_i}{\partial x} - v\frac{\partial c_i}{\partial y} - w\frac{\partial c_i}{\partial z} + \varepsilon\left(\frac{\partial^2 c_i}{\partial x^2} + \frac{\partial^2 c_i}{\partial y^2} + \frac{\partial^2 c_i}{\partial z^2}\right) + s_i(t, x, y, z) + g_i(t, x, y, z, c_1, c_2, \ldots, c_m), \qquad i = 1, \ldots, m. \tag{3.1}$$

As motivated in [15] we shall follow what we called the ‘dummy-point’ approach. By this we mean that the whole physical domain is enclosed by a rectangular box, possibly introducing many artificial points. However, the regular grid structure, allowing for an efficient implementation on vectorprocessors, usually compensates for the introduced overhead.

In this paper, we shall use (standard) *symmetric* differences to approximate the spatial differential operators $\partial/\partial x$, $\partial/\partial y$ and $\partial/\partial z$. For a discussion on this choice, we again refer to [15] and to the recent overview [23], in which a considerable amount of literature on the spatial discretization of advection dominated partial differential equations is surveyed. In a forthcoming paper we plan to study *unsymmetric* (i.e., upwind) discretizations and the consequences for the choice of the time integration techniques.

Along the lines described in [15] and [16], equation (2.1), together with the initial condition and the boundary conditions is converted into the semidiscrete initial value problem

$$\frac{d\boldsymbol{C}_i(t)}{dt} = A_i(t)\boldsymbol{C}_i(t) + \boldsymbol{S}_i(t) + \boldsymbol{G}_i(t, \boldsymbol{C}_1(t), \boldsymbol{C}_2(t), \ldots, \boldsymbol{C}_m(t)), \qquad \boldsymbol{C}_i(0) = \boldsymbol{C}_{0,i}, \qquad i = 1, \ldots, m, \tag{3.2}$$

where each $\boldsymbol{C}_i(t)$ is a vector of dimension $N := N_x \cdot N_y \cdot N_z$ (N_x, N_y, N_z being the number of grid points in the various spatial dimensions, respectively), containing the concentrations of c_i, defined at all grid points of the computational domain. $A_i(t)$ are N-by-N matrices representing the spatial discretizations of the advection-diffusion operator in (3.1); the boundary conditions for the various concentrations c_i are supposed to be incorporated into the matrices $A_i(t)$. Furthermore, the N-dimensional vectors $\boldsymbol{S}_i(t)$ and $\boldsymbol{G}_i(t, \boldsymbol{C}_1(t), \boldsymbol{C}_2(t), \ldots, \boldsymbol{C}_m(t))$ are the discrete analogues of the source-sink functions s_i and the reaction terms g_i, respectively.

In the description of time integrators for (3.2), it is more convenient to represent this system in the form

$$\frac{d\boldsymbol{C}_i(t)}{dt} = \boldsymbol{F}_i(t, \boldsymbol{C}_i(t)) + \boldsymbol{G}_i(t, \boldsymbol{C}_1(t), \boldsymbol{C}_2(t), \ldots, \boldsymbol{C}_m(t)), \qquad \boldsymbol{C}_i(0) = \boldsymbol{C}_{0,i}, \qquad i = 1, \ldots, m. \tag{3.3}$$

Owing to the above ‘dummy-point’ approach, the data structures in the code are extremely simple: the arrays containing the concentrations $\boldsymbol{C}_i$ all have the same size and the subscripts are running over all indices, without making exceptions, since the dummy points are included. This approach very well fits the concepts of a vector machine. For a more extensive discussion of this topic we refer to [15] and [16].

4. Time integration methods

Before we specify the actual time integration methods that we have used, we will first discuss the nature of the (right-hand side function of the) ODE system (3.3). In particular, it is important to take into account the *coupling* of the unknowns in the functions $\boldsymbol{F}_i$ and $\boldsymbol{G}_i$, since this coupling has a large influence on the selection of an optimal algorithm.

Clearly, each $\boldsymbol{F}_i$ only depends on $\boldsymbol{C}_i$ and thus all $\boldsymbol{F}_i$ are uncoupled; the coupling of the various species enters the problem by means of the functions $\boldsymbol{G}_i$. This observation suggests to split the right-hand side of (3.3) and to integrate the resulting fractional systems

$$\text{(4.1a)} \qquad \frac{\mathrm{d}\boldsymbol{C}_i(t)}{\mathrm{d}t} = \boldsymbol{F}_i(t, \boldsymbol{C}_i(t)) \,, \qquad i = 1, \dots, m,$$

$$\text{(4.1b)} \qquad \frac{\mathrm{d}\boldsymbol{C}_i(t)}{\mathrm{d}t} = \boldsymbol{G}_i(t, \boldsymbol{C}_1(t), \boldsymbol{C}_2(t), \dots, \boldsymbol{C}_m(t)), \qquad i = 1, \dots, m,$$

separately, by different methods. For obvious reasons, (4.1a) and (4.1b) will respectively be referred to as the ‘transport part’ and the ‘chemical part’ of the original problem.

Let us first consider the transport part. Since the m ODE-systems in (4.1a) are completely independent, the various concentrations can be calculated concurrently. Notice that each of these systems is of dimension N, the total number of grid points, and that the coupling in each component of $\boldsymbol{F}_i$ corresponds to the 7-point stencil that we used in the spatial discretization. Hence, these systems are characterized by their large dimension, and the banded structure in the Jacobian matrices $\partial\boldsymbol{F}_i/\partial\boldsymbol{C}_i$.

The situation is completely different in the chemical part: here we have a large number (i.e., N) of uncoupled ODE-systems of relatively small dimension m. This is a consequence of the fact that the chemical reactions between the various species are defined in each grid point. To be more precise, if the component of $\boldsymbol{C}_i$ at grid point (j,k,l) is denoted by $C_{i,(j,k,l)}$, then the corresponding component of $\boldsymbol{G}_i$ depends on t, and on $C_{1,(j,k,l)}, C_{2,(j,k,l)}, \dots, C_{m,(j,k,l)}$ only. Hence, the chemical part is characterized by a massive number of relatively small ODE-systems, possessing Jacobian matrices which do not have a special structure (in principle, they can be full). Moreover, in contrast to the $\boldsymbol{F}_i$, the functions $\boldsymbol{G}_i$ are usually nonlinear.

In view of the above observations, we are led to the following algorithm:

```
to advance the solution over one step from $t_n$ to $t_{n+1} = t_n + \Delta t$ :
(i)     start with $\boldsymbol{C}_i^n \approx \boldsymbol{C}_i(t_n)$,  $i = 1, \dots, m$;
(ii)    integrate $\mathrm{d}\boldsymbol{C}_i(t)/\mathrm{d}t = \boldsymbol{F}_i(t, \boldsymbol{C}_i(t))$ ,  $i = 1, \dots, m$, using a suitable transport-solver, to
        obtain an approximation to the solution of (4.1a) at $t_{n+1}$;
(iii)   let the results obtained in (ii) be the initial values to integrate, for each grid point $(j,k,l)$,
        $\mathrm{d}C_{1,(j,k,l)}(t)/\mathrm{d}t = G_{1,(j,k,l)}(t, C_{1,(j,k,l)}(t), C_{2,(j,k,l)}(t), \dots, C_{m,(j,k,l)}(t))$,
              ⋮              ⋮              ⋮              ⋮              ⋮
        $\mathrm{d}C_{m,(j,k,l)}(t)/\mathrm{d}t = G_{m,(j,k,l)}(t, C_{1,(j,k,l)}(t), C_{2,(j,k,l)}(t), \dots, C_{m,(j,k,l)}(t))$,
        by means of an appropriate chemical-ODE solver;
(iv)    the results of (iii) are collected into $\boldsymbol{C}_i^{n+1}$, which approximate $\boldsymbol{C}_i(t_{n+1})$, the solution
        of the ‘full’ ODE system (3.3)
end of the step
```

Given this algorithm, we now have to make a choice for the transport-solver and for the chemical-ODE-solver. In [15] and [16] we compared various time integration techniques on the basis of their suitability to solve a *single* transport equation without a reaction term, i.e., problem (4.1a) with $m = 1$. It turned out that the Odd-Even Line Hopscotch (OELH) method is an outstanding technique for integrating such an equation on multi-vectorcomputers. Therefore, in part (ii) of the above algorithm we adopt the OELH method as our transport-solver. The choice for the chemical-ODE solver is less evident. Observing that chemical reactions in *water* are usually quite slow (or, in ODE terms, they yield *nonstiff* ODEs), and that modest accuracy in the present application is sufficient, we decided to consider the class of second-order, explicit Runge-Kutta (RK) methods. (In passing, we remark that the chemistry in *air* pollution models usually changes from stiff (starting at sunrise) to nonstiff (starting at sunset); see [22]). Along the lines described in [9], it can be proved that the concatenation of several consistent methods in the way described in the aforementioned algorithm will result in a so-called fractional step method which is, in general, only first-order accurate in time. In this connection we remark that the parts (ii) and (iii) in the above algorithm can be reversed, i.e., we first treat the chemical part, followed by the transport part. Of particular interest is to interchange these parts in alternating time steps to obtain the sequence {OELH, RK, RK, OELH, OELH, RK, ...}. This 'symmetrical' splitting has been suggested by Strang [18] and usually reduces the splitting error. In a forthcoming paper we will investigate this effect.

In the following subsections, the OELH and RK methods will be discussed in more detail.

4.1. The Odd-Even Line Hopscotch method

Hopscotch methods [7] belong to the class of so-called operator splitting methods. The specific Odd-Even Line Hopscotch (OELH) method that we have in mind, is discussed in detail in [15, 16 and 21]. Here, we briefly summarize this method to make this paper self-supporting. In the description of the OELH method for integrating (4.1a), we shall omit the index *i*, since all contaminants are treated in an analogous way.

Let the right-hand side function $\boldsymbol{F}$ be split into

$$\text{(4.2)} \qquad \boldsymbol{F}(t,\boldsymbol{C}(t)) = \boldsymbol{F}_o(t,\boldsymbol{C}(t)) + \boldsymbol{F}_*(t,\boldsymbol{C}(t)),$$

where $\boldsymbol{F}_o$ is the vector that is obtained from $\boldsymbol{F}$ by replacing all components corresponding to a grid point $P_{j,k,l}$ where $j+k$ assumes an *even* value by zero. Similarly, $\boldsymbol{F}_*$ is obtained if all elements in $\boldsymbol{F}$ corresponding to a grid point for which $j+k$ is *odd*, are replaced by zero. Notice that the third index, l, is not involved in this definition, which means that we apply the same splitting on each horizontal plane of the grid, or equivalently, *all* grid points lying on the *same* vertical grid line are either in $\boldsymbol{F}_o$ or in $\boldsymbol{F}_*$. Now we define the two-stage, second-order splitting method

$$\text{(4.3)} \qquad \begin{aligned} \boldsymbol{C}^{n+1/2} &= \boldsymbol{C}^n + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_o(t_n+\Delta t/2, \boldsymbol{C}^{n+1/2}) + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_*(t_n,\boldsymbol{C}^n), \\ \boldsymbol{C}^{n+1} &= \boldsymbol{C}^{n+1/2} + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_o(t_n+\Delta t/2, \boldsymbol{C}^{n+1/2}) + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_*(t_{n+1},\boldsymbol{C}^{n+1}), \end{aligned}$$

where $\boldsymbol{C}^n$ denotes the numerical solution at $t = t_n$, and Δt is the stepsize; $\boldsymbol{C}^{n+1/2}$ can be considered as an intermediate approximation. Using the definition of the splitting functions $\boldsymbol{F}_*$ and $\boldsymbol{F}_o$, (4.3) can be rewritten in terms of the '$*$-components' and the 'o-components' as

$$\text{(4.4)} \qquad \begin{aligned} \boldsymbol{C}_*^{n+1/2} &= \boldsymbol{C}_*^n + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_*(t_n,\boldsymbol{C}^n), \\ \boldsymbol{C}_o^{n+1/2} &= \boldsymbol{C}_o^n + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_o(t_n+\Delta t/2,\boldsymbol{C}^{n+1/2}), \\[1ex] \boldsymbol{C}_o^{n+1} &= \boldsymbol{C}_o^{n+1/2} + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_o(t_n+\Delta t/2,\boldsymbol{C}^{n+1/2}), \\ \boldsymbol{C}_*^{n+1} &= \boldsymbol{C}_*^{n+1/2} + \tfrac{1}{2}\Delta t\, \boldsymbol{F}_*(t_{n+1},\boldsymbol{C}^{n+1}). \end{aligned}$$

The *explicit* (Forward Euler) steps over $\Delta t/2$ (i.e., the first and third line in (4.4)) are of course straightforward. The *implicit* second and fourth line in (4.4) (here we recognize Backward Euler) are more laborious. However, since the $\boldsymbol{F}$ function originates from a *three*-point coupling in the horizontal directions, the neighbouring points that we need in approximating $\partial/\partial x$, $\partial/\partial y$, $\partial^2/\partial x^2$ and $\partial^2/\partial y^2$ have just been updated to the new time level in the preceding explicit part of the step. Consequently, we have implicitness in the vertical direction only. Because we also used a three-point coupling to discretize $\partial/\partial z$ and $\partial^2/\partial z^2$, this implicit part of the algorithm results in the solution of a tridiagonal system along each vertical grid line.

Observe that the second and third line in (4.4) have identical $\boldsymbol{F}_0$-evaluations. This means that we can save the second one of these evaluations and replace the third line in (4.4) by $\boldsymbol{C}_0{}^{n+1} = \boldsymbol{C}_0{}^{n+1/2} + \left(\boldsymbol{C}_0{}^{n+1/2} - \boldsymbol{C}_0{}^{n}\right)$. We remark that a similar substitution is not possible in the first line in (4.4) (using the last $\boldsymbol{F}_*$-evaluation from the previous time step), because the $\boldsymbol{C}$-argument in the $\boldsymbol{F}_*$-function has been changed due to the treatment of the chemistry. Notice that the Strang-splitting, as mentioned in the previous section, possesses two successive applications of the OELH method. Therefore, in the second one of these applications, the so-called 'fast-form' can be used in *both* explicit stages of the OELH method.

Summarizing, the total amount of work (for each contaminant) consists in solving $N_x{\cdot}N_y$ uncoupled, tridiagonal systems, each of dimension N_z. We recall that the fall velocity w_f is not taken into account, implying that the transport part of the problem is linear. If w_f plays a role in the model, then a (modified) Newton process can be applied. Apart from this linear algebra work, the scheme (4.4) requires "1.5" $\boldsymbol{F}$-evaluation over the whole field.

The main reason to construct this particular variant within the hopscotch family is that usually the most severe restriction on the timestep originates from the discretization in the *vertical* direction; that is, the terms $|w|/\Delta z$ and $\varepsilon_z/(\Delta z)^2$ are usually larger than the similar expressions corresponding to the horizontal direction. Hence, in explicit methods, the timestep will be dictated by stability conditions induced by the vertical discretization. Since the OELH method is explicit in the horizontal, but implicit in the vertical, it usually allows for timesteps which are in accordance with the size that we would require for accuracy reasons (see [21] for a detailed stability analysis of the OELH method).

4.2. Explicit Runge-Kutta methods

As said before, the chemical reactions that we will consider are quite slow and give rise to nonstiff ODEs. Therefore, explicit Runge-Kutta (RK) methods will be used to integrate in each spatial grid point the m-dimensional system of ODEs (cf. (4.1b))

$$\frac{\mathrm{d}\boldsymbol{C}(t)}{\mathrm{d}t} = \boldsymbol{G}(t, \boldsymbol{C}), \tag{4.5}$$

where $\boldsymbol{C}(t)$ contains the m components of the vectors $\boldsymbol{C}_i(t)$ that correspond to the same spatial grid point. Hence, we have N independent systems of the form (4.5).

Since the accuracy demands in the present application are quite modest, we restrict our considerations to RK methods of second order. A well known method of this type reads (see e.g. [8])

$$\begin{aligned} \boldsymbol{C}^{(1)} &= \boldsymbol{C}^n + \tfrac{1}{2}\Delta t\, \boldsymbol{G}(t_n, \boldsymbol{C}^n), \\ \boldsymbol{C}^{n+1} &= \boldsymbol{C}^n + \Delta t\, \boldsymbol{G}(t_n{+}\Delta t/2, \boldsymbol{C}^{(1)}). \end{aligned} \tag{4.6}$$

Here, Δt again denotes the timestep, the quantity $\boldsymbol{C}^{(1)}$ denotes an intermediate result, and $\boldsymbol{C}^{n+1}$ is an approximation to the concentrations of the contaminants at the point $t = (n{+}1)\Delta t$. Notice that, in this particular context, the initial value for this ODE-step, denoted by $\boldsymbol{C}^n$, is the result of the step taken by the transport solver (cf. the algorithm in Section 4). The stability region of the method (4.6) does not include points on the imaginary axis (apart from the origin); in the case that the Jacobian matrix $\partial\boldsymbol{G}/\partial\boldsymbol{C}$ has eigenvalues on the imaginary axis, (4.6) will behave unstable and an alternative is offered by [8]

$$\begin{aligned} \boldsymbol{C}^{(1)} &= \boldsymbol{C}^n + \tfrac{1}{2}\Delta t\, \boldsymbol{G}(t_n, \boldsymbol{C}^n), \\ \boldsymbol{C}^{(2)} &= \boldsymbol{C}^n + \tfrac{1}{2}\Delta t\, \boldsymbol{G}(t_n{+}\Delta t/2, \boldsymbol{C}^{(1)}), \\ \boldsymbol{C}^{n+1} &= \boldsymbol{C}^n + \Delta t\, \boldsymbol{G}(t_{n+1}, \boldsymbol{C}^{(2)}). \end{aligned} \tag{4.7}$$

This scheme is also second-order accurate, and it includes the interval $[-2i, +2i]$ in its stability region, whereas the intersection of the stability region with the negative real axis is the same as for (4.6), viz. -2. The price to pay for this enhanced stability is one extra evaluation of the function $\boldsymbol{G}$. In the numerical experiments, both RK methods will be tested.

5. Exploiting the vectorization capabilities

In Section 4.1, we have seen that the OELH method requires the solution of $N_x \cdot N_y$ tridiagonal systems, each of dimension N_z. These systems can, of course, be solved using standard software, but then the special properties of these systems are not taken into account. For example, a standard LU-factorization of each of the coefficient matrices results in a highly parallel treatment, however the vector performance will then be modest since the LU-decomposition algorithm itself is recursive and hence non-vectorizable. To exploit the vectorization capabilities of the CRAY, we have used a tailormade approach, in which each step in the decomposition algorithm is performed for all systems simultaneously. Since all systems are *uncoupled* (owing to the five-point coupling in the horizontal) and have *equal dimension* (owing to the 'dummy-point' approach), each of the decomposition steps can be performed on full vector speed. This technique, originally proposed by Golub and Van Loan [6], is called *vectorization-across-the-tridiagonal systems*. For a detailed discussion on the actual implementation we refer to [10], where speed-up factors of approximately 20 are reported. Notice that the vector length in this approach equals $N_x \cdot N_y$, which is usually sufficiently large to leave room for exploiting parallelism, simply by cutting the long loops into smaller parts, each of which will still be of substantial size. In this way an attractive combination of vectorization and parallelization is obtained, which efficiently fits the concept of a multi-vectorprocessor.

A similar reasoning can be followed to integrate the ODE systems (4.5) originating from the chemistry. Applying th same ODE method for all these uncoupled systems, the various parts in such a method can be performed in vector mode. Since here the vector length is even as large as $N_x \cdot N_y \cdot N_z$, an excellent vector performance is to be expected. This is confirmed in the numerical experiments (cf. Section 6.2.1). The above approach will be termed *vectorization-across-the-ODE systems*.

Finally, we remark that the uncoupled tridiagonal systems, as well as the uncoupled ODE systems also could be solved in a *massively* parallel way. However, since we concentrate in this paper on the implementation on multi-*vector*processors, we have exploited the vectorization properties of the algorithm.

6. Numerical experiments

The algorithms described in Section 4 will be applied to a test problem consisting of five species. This model is of the form (3.1) with $m = 5$; the source/sink terms s_i are omitted, and the chemical terms g_i are defined by

$$\begin{aligned}
& g_1 = -k_1 c_1 c_5, \\
& g_2 = -k_2 c_2, \\
(6.1\text{a}) \qquad & g_3 = +k_2 c_2 - k_3 c_3 c_5^2, \\
& g_4 = +k_3 c_3 c_5^2, \\
& g_5 = -\frac{k_1 c_1 c_5}{1 - \exp(-5 k_6)} - k_3 c_3 c_5^2,
\end{aligned}$$

where $c_1,\ldots,c_5$ denote the concentrations of BOD_5, NBOD, $(NH_4)^+$, $(NO_3)^-$, and O_2, respectively, and $k_1,\ldots,k_6$ are given reaction constants.

Equation (6.1a) models the chemical reaction of a mixture of organic components of natural sewage in surface water that is quite common in urban sewer outlets. The class of mixtures is indicated by names containing the abbreviation

BOD for Biochemical Oxygen Demand. BOD_5 is a typical mixture in which the carbonaceous part (CBOD) consumes a certain amount of oxygen in five days and NBOD stands for nitrogenous BOD. Full details on this model can be found in [19]. Equation (6.1a) is based on the following chemical reactions

$$\text{(6.1b)} \qquad \begin{array}{lcl} BOD_5 + O_2 & \xrightarrow{k_1} & W1 \\ NBOD & \xrightarrow{k_2} & NH_4^+ + W2 \\ NH_4^+ + 2O_2 & \xrightarrow{k_3} & 2H^+ + H_2O + NO_3^- \end{array}$$

where W1 and W2 are by-products which are of no interest in this study.

The physical domain is defined by the rectangular box $0 \le x \le L_x$, $0 \le y \le L_y$, $-L_z \le z \le 0$. For $c_1,\ldots,c_4$ we impose homogeneous Neumann conditions ($\partial c/\partial n = 0$) on all boundaries. For c_5 we have the same boundary conditions, except at the water surface where we prescribe a Dirichlet condition. In this way an oxygen supply is modeled to avoid that the concentration of oxygen would vanish for $t \to \infty$ (cf. the minus signs in right-hand side of the last equation in (6.1a)).

The *divergence free* velocity field (u, v, w) is prescribed by an analytical expression [4]. In terms of the scaled coordinates $\tilde{x} := x/L_x$, $\tilde{y} := y/L_y$, $\tilde{z} := z/L_z$, it reads

$$\text{(6.2)} \qquad \begin{aligned} u(t,x,y,z) &= \left\{ \tilde{y} + 3(\tilde{z}+1/2)\left[(\tilde{x}-1/2)^2 + (\tilde{y}-1/2)^2 - r^2\right]\right\} d(t), \\ v(t,x,y,z) &= \left\{ -\tilde{x} + 3(\tilde{z}+1/2)\left[(\tilde{x}-1/2)^2 + (\tilde{y}-1/2)^2 - r^2\right]\right\} d(t), \\ w(t,x,y,z) &= -3L_z\,\tilde{z}\,(\tilde{z}+1)\left\{(\tilde{x}-1/2)/L_x + (\tilde{y}-1/2)/L_y\right\} d(t), \end{aligned}$$

with $r = 1/3$ and $d(t) = \cos(2\pi t/T_p)$.

The simulation is started at $t = 0$. For the contaminants c_1 ([BOD_5]) and c_2 ([NBOD]) we prescribe an initial concentration at the center of the domain, which is fastly decaying towards the boundaries, i.e., the initial condition is of the form

$$\text{(6.3)} \qquad c_i(0,x,y,z) = \gamma_i \exp\left\{-20\left[(\tilde{x}-1/2)^2 + (\tilde{y}-1/2)^2 + (\tilde{z}+1/2)^2\right]\right\},\ i = 1, 2,$$

with $\gamma_1 = 50$ and $\gamma_2 = 25$. Since c_3 ([NH_4^+]) and c_4 ([NO_3^-]) are formed by the chemical reactions (cf. (6.1b)), these concentrations are set to zero at $t = 0$, at all grid points. The oxygen concentration c_5 is initially given the value 0.01, over the whole computational domain; this value is also used in the (time independent) Dirichlet condition for c_5 at the water surface.

In our experiments, we take the following values for the parameters: $L_x = L_y = 20\,000$ [m], $L_z = 100$ [m], $\varepsilon = 0.5$ [m^2/s], and $T_p = 43200$ [s] (12 hours). The length of the integration interval $T = 86\,400$ [s] (which equals 1 day). Realistic values for the reaction rate constants are: $k_1 = 0.1$, $k_2 = 0.035$, $k_3 = 0.075$, $k_4 = 0.15$, $k_5 = 1.0$; since these values have the dimension [day^{-1}], and we are working in the (m, kg, s)-system, these numbers should be devided by 86400 (i.e., the number of seconds in 1 day); k_6 has the value 0.1, and does not need scaling.

To obtain insight to what extent the length of the vectors will influence the performance of the algorithms, we have done experiments on several spatial grids:

$\text{Grid}_{\text{fine}}$ is defined by: $N_x = N_y = 101$, $N_z = 21$, amounting to $\approx 2.1\ 10^5$ grid points;
$\text{Grid}_{\text{middle}}$ is defined by: $N_x = N_y = 51$, $N_z = 11$, amounting to $\approx 2.9\ 10^4$ grid points;
$\text{Grid}_{\text{coarse}}$ is defined by: $N_x = N_y = 26$, $N_z = 6$, amounting to $\approx 4.1\ 10^3$ grid points.

Since we are also interested in the accuracy behaviour of the time integration methods, we have calculated a reference solution on

Grid$_{\text{reference}}$, which is defined by: $N_x = N_y = 201$, $N_z = 41$, amounting to $\approx 1.7\,10^6$ grid points,

using a very small timestep. The solution obtained on this grid will be used to measure the (global) errors.

6.1. Algorithmic tests

First, we will perform some experiments to test the *numerical* properties of the methods, like accuracy and stability of the time integration methods and the influence of the spatial grids on the accuracy.

6.1.1. Order behaviour. As specified in the previous section, the simulation is started by imposing the 'Gaussian-shaped' initial pollution (6.3) for c_1 and c_2 at the centre of the domain. The local concentration of these contaminants is then transported through the domain and reduced in magnitude by diffusion and chemical reactions. In Figure 6.1 we have plotted the reference solution for c_1 corresponding to a horizontal plane halfway down ($z = -50\,m$); to get an impression of the behaviour in time, this plot is given for 10 equally spaced points in time. Figure 6.2 shows the same information for the oxygen concentration c_5 .

Since we are interested whether the numerical methods can properly approximate these local concentrations, we focus on the neighbourhood of the maximum values. To be more precise, in measuring the errors, all grid points are considered where the numerical value of the various concentrations are at least half the maximum value of the corresponding reference solution. Let the set of grid points satisfying this condition be denoted by E_i. Notice that these sets may depend on the particular concentration $\boldsymbol{C}_i$; it turned out that $E_1,\ldots,E_4$, cover 10% - 20% of all grid points and that in E_5 all grid points are involved.

Using this norm, we measured the global errors in the end point $T = 86400$, both in absolute and relative sense. The actual values given in the tables of results are respectively defined by

$$(6.4) \qquad cd_i := \max_{E_i} \left(-{}^{10}\log|\text{absolute errors}| \right), \qquad sd_i := \max_{E_i} \left(-{}^{10}\log|\text{relative errors}| \right), \qquad i = 1, \ldots, 5.$$

Hence, cd_i can be considered as the number of correct digits for concentration $\boldsymbol{C}_i$ (restricted to the set E_i), and similarly, sd_i denotes the number of significant digits.

In Table 6.1 we show the (numerical) behaviour of the OELH method (4.4) combined with the stabilized RK method (4.7).

Table 6.1. cd_i / sd_i - values for {(4.4), (4.7)} on different spatial grids and for various timesteps

spatial grid	# steps	Δt	cd_1/sd_1	cd_2/sd_2	cd_3/sd_3	cd_4/sd_4	cd_5/sd_5
Grid$_{\text{fine}}$	216	400			unstable behaviour		
Grid$_{\text{fine}}$	432	200	1.9/2.9	2.2/2.9	3.7/2.9	8.9/2.8	3.6/1.5
Grid$_{\text{fine}}$	864	100	2.2/3.2	2.5/3.2	4.0/3.2	9.2/3.0	3.6/1.5
Grid$_{\text{middle}}$	108	800			unstable behaviour		
Grid$_{\text{middle}}$	216	400	1.4/2.5	1.8/2.5	3.2/2.5	8.6/2.4	3.2/1.2
Grid$_{\text{middle}}$	432	200	1.5/2.6	1.8/2.6	3.2/2.6	8.8/2.4	3.2/1.2
Grid$_{\text{middle}}$	864	100	1.5/2.7	1.8/2.7	3.2/2.7	8.8/2.4	3.2/1.2
Grid$_{\text{coarse}}$	54	1600			unstable behaviour		
Grid$_{\text{coarse}}$	108	800	0.8/2.1	1.1/2.1	2.6/2.1	8.3/1.9	3.2/1.1
Grid$_{\text{coarse}}$	216	400	0.8/2.1	1.2/2.1	2.6/2.1	8.3/1.9	3.2/1.1
Grid$_{\text{coarse}}$	432	200	0.9/2.1	1.2/2.1	2.6/2.1	8.3/1.9	3.2/1.1
Grid$_{\text{coarse}}$	864	100	0.9/2.1	1.2/2.1	2.6/2.1	8.3/1.9	3.2/1.1

This table gives rise to the following remarks and conclusions:
- the instabilities are caused by the OELH method and not by the chemical-ODE solver, which is clear from the fact that coarser grids allow for larger timesteps. The size of the maximal stable timestep is in accordance with the results derived in [21] for a model equation.
- on the fine grid, the temporal errors dominate; as already remarked in Section 4, the results show a first-order behaviour in time for the combined OELH-RK method.
- comparing the smallest-timestep-results on the various grids, we observe, for most concentrations, a second-order behaviour, which is in agreement with the symmetric, three-point discretization that we have used.

6.1.2. Influence of the chemical ODE-solver. Next, we test the influence of the various RK methods on the total algorithm. Since we focus on the time integration aspects, we give only results obtained on the fine grid.

In general, combining two different methods to integrate the ‘fractional’ systems (4.1a) and (4.1b) does not necessarily require the use of equal stepsizes. In particular, since the chemistry in our test model is rather slow, it seems reasonable to use a timestep for (4.1b) that is a multiple of the timestep used to integrate (4.1a). Let us call this multiple μ. The algorithm described in Section 4 can now be modified into: take μ steps of size Δt with the OELH method for the transport part, followed by one large step of size $\mu\Delta t$ with an RK method for the chemistry part.

In Table 6.2 we present results for $\mu = 1, 4, 12,$ and 32, both for the stabilized RK method (4.7) and for the traditional RK method (4.6). As mentioned before, the overall order of the {OELH, RK}-combination equals one, irrespective of the order of the particular RK method, and irrespective of the value of μ. Therefore, it is also of interest to see what the effect on the accuracy will be if we leave the class of second-order RK methods and add results for the first-order Euler method. Furthermore, we include in Table 6.2 the required CPU timings to see the gain in computer time by selecting a μ-value larger than 1.

Table 6.2. cd_i / sd_i - values for various combinations of time integration methods, obtained on Grid$_{\text{fine}}$, using Δt = 100

combination of methods	μ	cd_1/sd_1	cd_2/sd_2	cd_3/sd_3	cd_4/sd_4	cd_5/sd_5	CPU
{OELH, stabilized RK}	1	2.2/3.2	2.5/3.2	4.0/3.2	9.2/3.0	3.6/1.6	195.0
{OELH, stabilized RK}	4	2.2/3.2	2.5/3.2	4.0/3.2	8.7/2.5	3.6/1.5	177.3
{OELH, stabilized RK}	12	2.2/3.2	2.5/3.2	4.0/3.2	8.0/1.9	3.2/1.2	171.1
{OELH, stabilized RK}	32	2.2/3.2	2.5/3.2	4.0/3.2	7.5/1.3	2.8/0.8	170.5
{OELH, classical RK}	1	2.2/3.2	2.5/3.2	4.0/3.2	9.2/3.1	3.6/1.6	189.1
{OELH, classical RK}	4	2.2/3.2	2.5/3.2	4.0/3.2	8.7/2.5	3.6/1.5	173.9
{OELH, classical RK}	12	2.2/3.2	2.5/3.2	4.0/3.2	8.0/1.8	3.2/1.2	171.0
{OELH, classical RK}	32	2.2/3.2	2.5/3.2	4.0/3.2	7.4/1.2	2.8/0.8	169.7
{OELH, Forward Euler}	1	2.2/3.2	2.5/3.2	4.0/3.2	8.6/2.4	3.6/1.5	178.7
{OELH, Forward Euler}	4	2.2/3.2	2.5/3.2	3.9/3.2	8.0/1.9	3.6/1.5	172.5
{OELH, Forward Euler}	12	2.2/3.2	2.5/3.2	3.9/3.1	7.6/1.4	3.2/1.2	169.3
{OELH, Forward Euler}	32	2.2/3.2	2.5/3.2	3.7/2.9	7.2/1.1	2.8/0.8	169.0

From this table we see that the value of μ has hardly influence on the accuracy of the first three concentrations; since the reduction of the accuracies of $\boldsymbol{C}_4$ and $\boldsymbol{C}_5$ is significant for large μ, and the corresponding decrease in CPU time is only marginal, we conclude that the chemistry in this particular model is not slow enough to justify large μ-values. Confining our consideration to μ = 1 and μ = 4, a comparison of the various chemistry solvers reveals that both RK methods yield the same results, whereas Euler's method loses some accuracy for $\boldsymbol{C}_4$, without giving sufficient compensation in terms of reduced CPU time. Apparently, the second-order RK methods have a favourable influence on the error constants of the overall algorithm. In conclusion, we might say that the combination {OELH, classical RK method (4.6), $\mu = 4$} seems to be optimal for this particular test example, although changing to the more robust stabilized RK method (4.7) hardly increases the CPU time.

6.2. Performance results

In this section we describe the performance of the algorithm when implemented on the four-processor CRAY C98/4256. In all experiments we used the CF77 compiling system. Performance results in scalar and vector mode are respectively obtained using **cf77 -Wf" -o novector"** and **cf77 -Zv**. Moreover, we frequently used the package **perfview** [1] (using the flags **-F** and **-lperf**) to obtain the Megaflop rates (i.e., 10^6 floating point operations per second) and CPU times of the various routines. All experiments have been performed on one processor. Since the clock cycle time of the CRAY C98/4256 equals 4.2 nanoseconds, and each processor is equipped with two vector pipes, the optimal vector speed equals 476 Mflops; in the exceptional case that a multiply and an addition can always be chained, this theoretical peak performance can be multiplied by a factor two.

6.2.1. Vectorization aspects. We start with a survey of the vector performance of the main routines in the code; these are: the subroutine F (for computing the derivatives due to the transport part), the subroutine TRI3P3 (for solving the tridiagonal systems), the subroutine IMPL (for solving the implicit relations in the Hopscotch method), the subroutine G (for calculating the derivatives due to the chemistry part), the subroutine RUNGEST (the stabilized Runge-Kutta method for integrating the chemistry part), and the subroutine OELH (which is the driver routine for the Hopscotch method). In Table 6.3, we list the performance results obtained in calculating the reference solution (i.e., using $\text{Grid}_{\text{reference}}$). From now on, the parameter μ, discussed in Subsection 6.1.2, is fixed to 1.

Table 6.3. Vector performance of the main routines

routine	number of calls per time step	average time (in seconds)	accumulated percentage of CPU-time spent	Mflop rate
F	15	$4.6\ 10^{-2}$	49.8	454
TRI3P3	10	$1.7\ 10^{-2}$	61.7	464
IMPL	10	$1.6\ 10^{-2}$	73.0	264
G	3	$3.6\ 10^{-2}$	80.7	593
RUNGEST	1	$1.1\ 10^{-1}$	88.3	503
OELH	5	$1.8\ 10^{-2}$	94.6	505

From this table we conclude that:

- The high Megaflop rates obtained in most routines clearly indicate that the code vectorizes pretty well. Since the underlying algorithm has attractive *numerical* properties as well, an efficient solution process for bio-chemical-transport problems is obtained.
- The code spends approximately half of its time in the subroutine F, which evaluates the discretization of the spatial differential operators. Recall that the nature of the Hopscotch method requires, per call of F, these evaluations in alternating points, yielding a stride of 2. Nevertheless, the high Megaflop rate of the routine F indicates that the Cray is capable to efficiently cope with this structure.
- This stride effect is more pronounced in the routine IMPL, since there the number of arithmetic operations is relatively low.
- The time spent by TRI3P3, to solve the tridiagonal systems, is only 11.9 % of the total time. For a code based on an *implicit* method, this percentage is pretty low, showing that the linear algebra part is efficiently treated by the vectorization-across-the-tridiagonal-systems approach.

It should be remarked that the above nice results are obtained on the very fine reference grid. An interesting aspect is the influence of the spatial resolution (i.e., the vector length) on the vector performance. In Table 6.4, we compare the results obtained on the various grids.

Table 6.4. Vector performance for different grids

	Grid$_{\text{reference}}$		Grid$_{\text{fine}}$		Grid$_{\text{middle}}$		Grid$_{\text{coarse}}$	
routine	aver. time (in sec.)	Mflops	aver. time (in sec.)	Mflops	aver. time (in sec.)	Mflops	aver. time (in sec.)	Mflops
F	$4.6\ 10^{-2}$	454	$8.1\ 10^{-3}$	339	$1.6\ 10^{-3}$	229	$4.0\ 10^{-4}$	138
TRI3P3	$1.7\ 10^{-2}$	464	$2.2\ 10^{-3}$	457	$2.8\ 10^{-4}$	460	$4.0\ 10^{-5}$	425
IMPL	$1.6\ 10^{-2}$	264	$2.7\ 10^{-3}$	196	$5.9\ 10^{-4}$	122	$1.7\ 10^{-4}$	60
G	$3.6\ 10^{-2}$	593	$5.1\ 10^{-3}$	580	$7.4\ 10^{-4}$	591	$1.3\ 10^{-4}$	579
RUNGEST	$1.1\ 10^{-1}$	503	$1.5\ 10^{-2}$	494	$2.2\ 10^{-3}$	505	$3.9\ 10^{-4}$	486
OELH	$1.8\ 10^{-2}$	505	$2.5\ 10^{-3}$	499	$3.6\ 10^{-4}$	506	$6.8\ 10^{-5}$	465
overall performance		436		353		260		164
percentage of total CPU time spent in solving the chemistry part		15.3 %		13.2 %		10.3 %		7.5 %

From this table we observe that the Megaflop rates of the routines TRI3P3, G, RUNGEST, and OELH hardly decrease when the grids become coarser. The reason for this behaviour is that in these routines the loops corresponding to the three spatial dimensions can be collapsed, resulting in one long loop (except for TRI3P3, where only the two horizontal directions can be collapsed; however, since $N_x{\cdot}N_y$ is still quite large, even on the coarsest grid, the effect is not too dramatic).

The situation is different for the routines F and IMPL. Apart from the stride effect, the vector performance is seriously reduced on coarser grids since here *only the innermost loop* (of length N_x) can be vectorized, i.e., vectorization in one spatial dimension. Hence, both Grid$_{\text{middle}}$ and Grid$_{\text{coarse}}$ are not fine enough to efficiently exploit the capabilities of the C98/4256. However, in most real-life applications, the number of grid points will be at least as large as the number used in the two fine grids.

At the end of Table 6.4 we have listed the percentage of the total CPU time that is consumed by the chemistry solver. As a consequence of the aforementioned performance reduction of F and IMPL, this percentage decreases when the grids become coarser. Hence, on a grid of realistic resolution this percentage is about 15 %.

A next question is, of course, the speed-up factor that we have obtained owing to vectorization. Therefore, we list in Table 6.5 the global performance of the code on the realistic Grid$_{\text{fine}}$, both in scalar mode and vector mode. We measure a speed-up with a factor 11.5. Obviously, this number is favourably influenced on still finer grids (for example, using Grid$_{\text{reference}}$ we found a speed-up factor of 13.9), and will decrease on coarser grids (the speed-up factors obtained on Grid$_{\text{middle}}$ and Grid$_{\text{coarse}}$ are 8.9 and 6.5, respectively).

Table 6.5. Global performance and speed-up (on Grid$_{\text{fine}}$ and Δt = 100)

	CPU (sec.)	Mflop rate
scalar mode	1133	31
vector mode	98	353
speed-up factor	11.5	

6.2.2. Parallelization aspects. Finally, we consider the prospects that the methods offer with respect to using more than one vector processor. Instead of running the code on a dedicated system, we used the autotasking facility of the CRAY (specifying **cf77 -Zp** activates the Fortran-preprocessor FPP to analyse the code) to get an indication of the speed-up that can be obtained when various vectorprocessors are used. To that end we employed the so-called **atexpert** utility [1], which is meant to produce predictions of the performance on a dedicated system. The results obtained by this utility, when running the code on Grid$_{\text{fine}}$, are given in Table 6.6. These figures show an almost linear speed-up on a multi-vectorprocessor. A typical treatment of the autotasking facility of the CRAY is to collapse as many innerloops as possible for vectorization purposes and to use the next outerloop for parallelization. As a consequence, for some routines the vector speed (on *each* vector processor) is slightly decreased, due to a shorter vector length (we observed about 10% reduction of the Mflop rates). However, the speed-up factors due to multi-processing largely compensate for this effect.

Table 6.6. Parallel performance of the code, estimated by atexpert

# processors:	2	4	6	8	10	12	14	16
speed-up:	1.98	3.80	5.66	7.17	9.01	9.95	11.89	13.33

7. Summary

In this paper we have discussed a so-called fractional step method for the time integration of a three-dimensional transport-chemical model in shallow water. This fractional step method consists of two separate integration techniques: one for the transport part and one for the chemistry part. The separate treatment is motivated by the fact that the coupling in both parts is completely different: in the transport part we have coupling over the grid points (due to advection-diffusion operators), but there is no mutual coupling between the various species. In the chemical part, the species are coupled (due to reaction terms) per grid point, but here the coupling in space is absent. To solve the transport part, we use a Hopscotch method, which was specially designed for this kind of applications [15] and termed Odd-Even Line Hopscotch (OELH) method. Since the chemistry in water is usually of low activity, an explicit Runge-Kutta (RK) method can be used for integrating the resulting nonstiff ODEs. We have tested several second-order RK methods, mainly differing with respect to their stability characteristics. It turned out that, for the test example that we have used, the choice of a particular RK method is not very critical. Although the overall fractional step method is only first-order accurate in time, independent of the order of the RK method, a reduction in accuracy is observed when we select Euler's method (which is of order one) for the chemistry part.

The above methods have been selected mainly on the basis of their adequate numerical properties for this kind of problems, but an additional advantage is that their implementation on a multi-vectorprocessor results in an extremely high performance. To obtain a good vector performance we have applied a ‘vectorization-across-space-approach’ to treat the linear algebra part in the OELH method on full vector speed (introduced by Golub and Van Loan [6]), and similarly, to integrate the uncoupled ODE systems originating from the chemistry.

On the basis of a large-scale test problem (involving 5 chemical species and up to $2{\cdot}10^6$ grid points, resulting in 10^7 unknowns), we measured an overall vector performance of 436 Mflops on a 1-processor CRAY C98/4256 (having a clock cycle of 4.2 nanoseconds). The routines for solving the tridiagonal systems and those for integrating the chemical ODEs maintain an excellent vector speed of approximately half a Gigaflop, even if the grid is reduced to a coarseness level with insufficient resolution to capture the physical phenomena. The reason is that in these routines the loops can be collapsed, at least over two spatial dimensions. The situation is less favourable for the routine calculating the derivatives in the OELH method, because here vectorization is restricted to one spatial dimension. However, in many practical applications, the required number of grid points in one spatial direction is usually sufficiently large to obtain Megaflop rates of say 300.

Since many of the (collapsed) loops involve an extremely large number of points, it is to be expected that a distribution of these loops over various vectorprocessors will result in an almost optimal use of a multi-vectorprocessor. This is confirmed by experiments using the Cray-utility *atexpert*; it predicts an almost linear speed-up.

As explained in Section 3, we have chosen for the so-called ‘dummy-point’ approach, which means that the physical domain is enclosed by a rectangular box. In the test problem discussed in Section 6, this approach did not lead to dummy points, so that all floating point operations were really useful operations. In practical situations however, where we have capricious geometries (e.g., in estuaries, coasts and bottom profiles), the situation is less ideal and the introduction of dummy points is unavoidable. In such cases, we should consider an *effective* Mflop rate, which obviously has to be related to the number of *useful* (i.e., effective) floating point operations. However, this observation applies to any integration method since this ‘dummy-point’ approach is inherent to the spatial discretization and the choice of the data structures. Furthermore, in many real-life applications, the whole physical domain is divided into a number of subdomains. An obvious advantage of this so-called ‘domain decomposition’ approach is that on each subdomain a spatial discretization of appropriate resolution can be used; furthermore, solving the problems on each subdomain can be done in parallel. Hence, the solution technique described in this paper can be used as a prototype solver for each of these subdomains. Since the horizontal coupling in the OELH method is rather weak, the ‘matching’ of the various subdomain-solutions is expected to result in an efficient method to obtain the solution on the whole physical domain. These aspects will be discussed in a forthcoming paper.

Hence, we conclude that the combination of the OELH method for the transport part and an explicit RK method for the chemical part results in an efficient method to solve a three-dimensional transport-chemical model in shallow water, because the resulting fractional step method possesses adequate numerical properties like accuracy and stability, and moreover, the implementation allows for an almost optimal use of the vectorization capabilities of multi-vector processorcomputers.

Acknowledgement. We are grateful to Dr. R. Vos from Delft Hydraulics for providing us with information about real-life bio-chemical reactions of which he also clearly explained the environmental relevance. We also want to acknowledge Prof. dr. P.J. van der Houwen for his many valuable comments during the preparation of this paper.

References

[1] Cray Research, Inc. *UNICOS Performance Utilities Reference Manual, SR-2040, edition 7.0.*

[2] A.M. Davies, *Formulation of a three-dimensional hydrodynamic sea model using Galerkin-eigenfunction method*, Int. J. Numer. Meth. Fluids, 3, 1983, 33-60.

[3] A.M. Davies, R.B. Grzonka and M. O'Neill, *Development and application of 3-D shallow water models on the Cray computer*, in: E.J. Pitcher (editor), *Science and engineering on CRAY supercomputers*, Springer Verlag, 1990, 323-334.

[4] D. Dunsbergen, *Particle models for transport in three-dimensional shallow water flow*, Ph. D. Thesis, Delft Technical University, 1994.

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

[6] G.H. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins Press, Baltimore, Maryland, 2nd edition, 1989.

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

[8] P.J. van der Houwen, *Construction of Integration Formulas for Initial Value Problems*, North-Holland series in Applied Mathematics and Mechanics, Vol. 19, North-Holland, 1977.

[9] P.J. van der Houwen and B.P. Sommeijer, *Fractional Runge-Kutta methods with application to convection-diffusion equations*, Impact Comput. Sc. Engng. 4, 1992, 195-216.

[10] J. Kok, *Efficient solution of linear systems for the time integration of three-dimensional transport models*, in: L. Dekker, W. Smit and J.C. Zuidervaart (editors), *Proceedings of the EUROSIM International Conference on Massively Parallel Processing; Applications and Development*, June 21-23, 1994, 79-86, Delft, The Netherlands.

[11] J.J. Leendertse, *Aspects of a computational model for long-period water-wave propagation*, Memorandum RM-5294-PR, Rand Corporation, Santa Monica, Cal., 1989.

[12] J.J. Leendertse, *A new approach to three-dimensional free surface flow modelling*, Report R-3712-Neth/RC, Rand Corporation, Santa Monica, Cal., 1989.

[13] H.X. Lin, H.H. ten Cate, L. Dekker, M. Roest, E. Vollebregt, Th. L. van Stijn and J.B. Berlamont, *Parallel simulation of 3-D flow and transport models within the NOWESP project*, in: L. Dekker, W. Smit and J.C. Zuidervaart (editors), *Proceedings of the EUROSIM International Conference on Massively Parallel Processing; Applications and Development*, June 21-23, 1994, 53-62, Delft, The Netherlands.

[14] R. Schreiber and H.D. Simon, *Towards the Teraflops Capability for CFD*, in: H.D. Simon (editor), *Parallel Computational Fluid Dynamics; Implementations and Applications*, MIT Press, 1992.

[15] B.P. Sommeijer, P.J. van der Houwen and J. Kok, *Time integration of three-dimensional numerical transport models*, Appl. Numer. Math. 16, 1994, 201-225.

[16] B.P. Sommeijer and J. Kok, *Implementation and performance of a three-dimensional numerical transport model*, to appear in Int. J. Numer. Meth. Fluids, 1995.

[17] G.S. Stelling, *On the construction of computational methods for shallow water flow problems*, Ph.D. Thesis, Delft Technical University, 1983.

[18] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal. 5,1968, 506-517.

[19] R.V. Thomann and J.A. Mueller, *Principles of surface water quality modeling and control*, Harper and Row, New York, 1987.

[20] 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, The Netherlands, 1989.

[21] 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.

[22] J.G. Verwer, J.G. Blom and W.H. Hundsdorfer, *An implicit-explicit approach for atmospheric transport-chemistry problems*, Report NM-R9501, CWI, Amsterdam, 1995.

[23] C.B. Vreugdenhil and B. Koren (eds.), *Numerical Methods for Advection-Diffusion Problems*, Notes on Numerical Fluid Mechanics, Vol. 45, Vieweg, Braunschweig, 1993.

Figure Legends

Figure 6.1 The computed concentration $\boldsymbol{C}_1$ corresponding to a horizontal plane, halfway down ($z = -50\,m$).
Plots are given for $t = k \cdot 9600$ sec., $k = 0, \ldots , 4$ (left column), and $k = 5, \ldots , 9$ (right column).
The numbers along the horizontal axes denote kilometers; the concentration is measured in kg/m^3 (vertical axis).

Figure 6.2 The computed concentration $\boldsymbol{C}_5$ corresponding to a horizontal plane, halfway down ($z = -50\,m$).
Plots are given for $t = k \cdot 9600$ sec., $k = 0, \ldots , 4$ (left column), and $k = 5, \ldots , 9$ (right column).
The numbers along the horizontal axes denote kilometers; the concentration is measured in kg/m^3 (vertical axis).

A vector/parallel method for a three-dimensional transport model coupled with bio-chemical terms

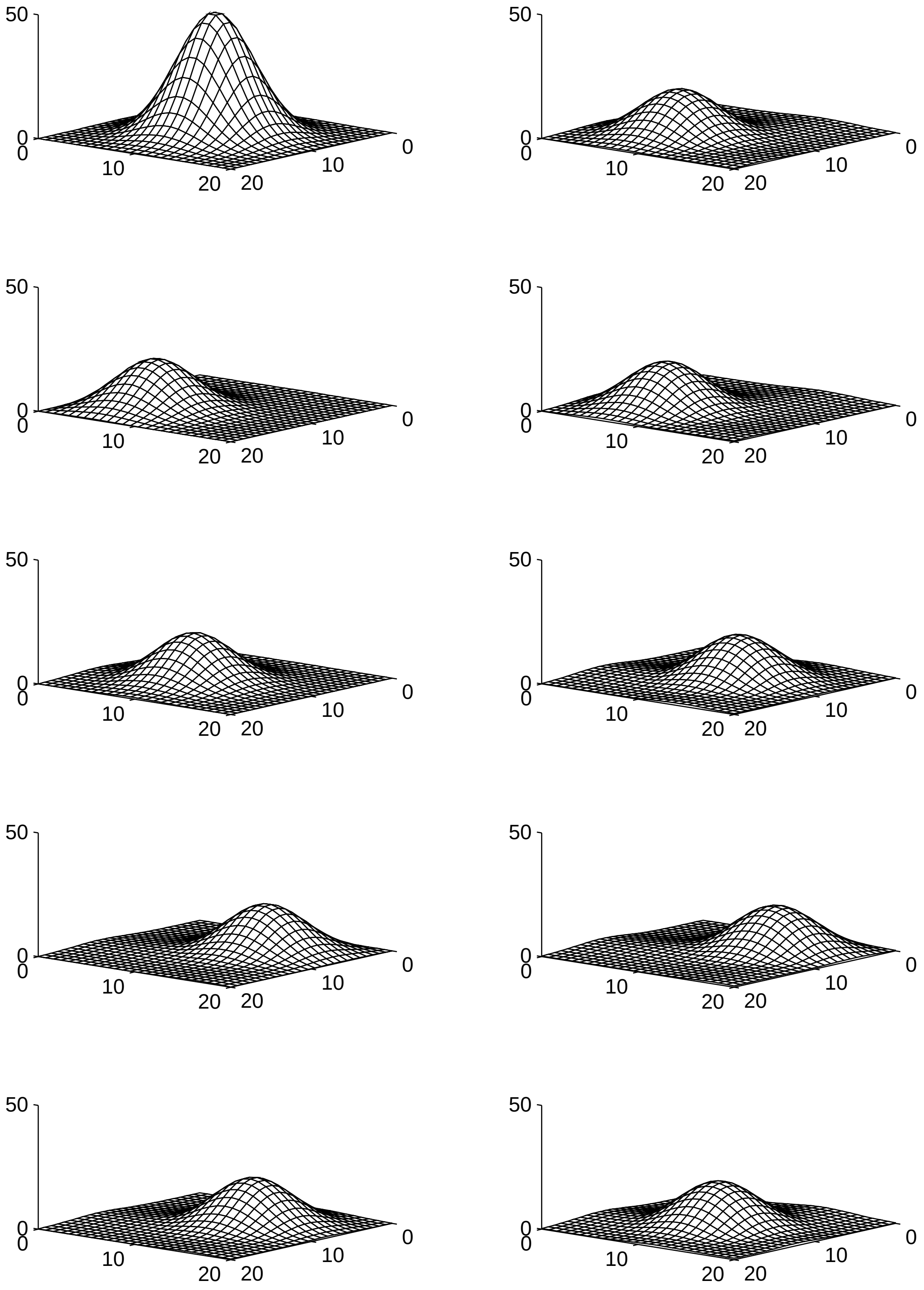
B.P. Sommeijer and J. Kok

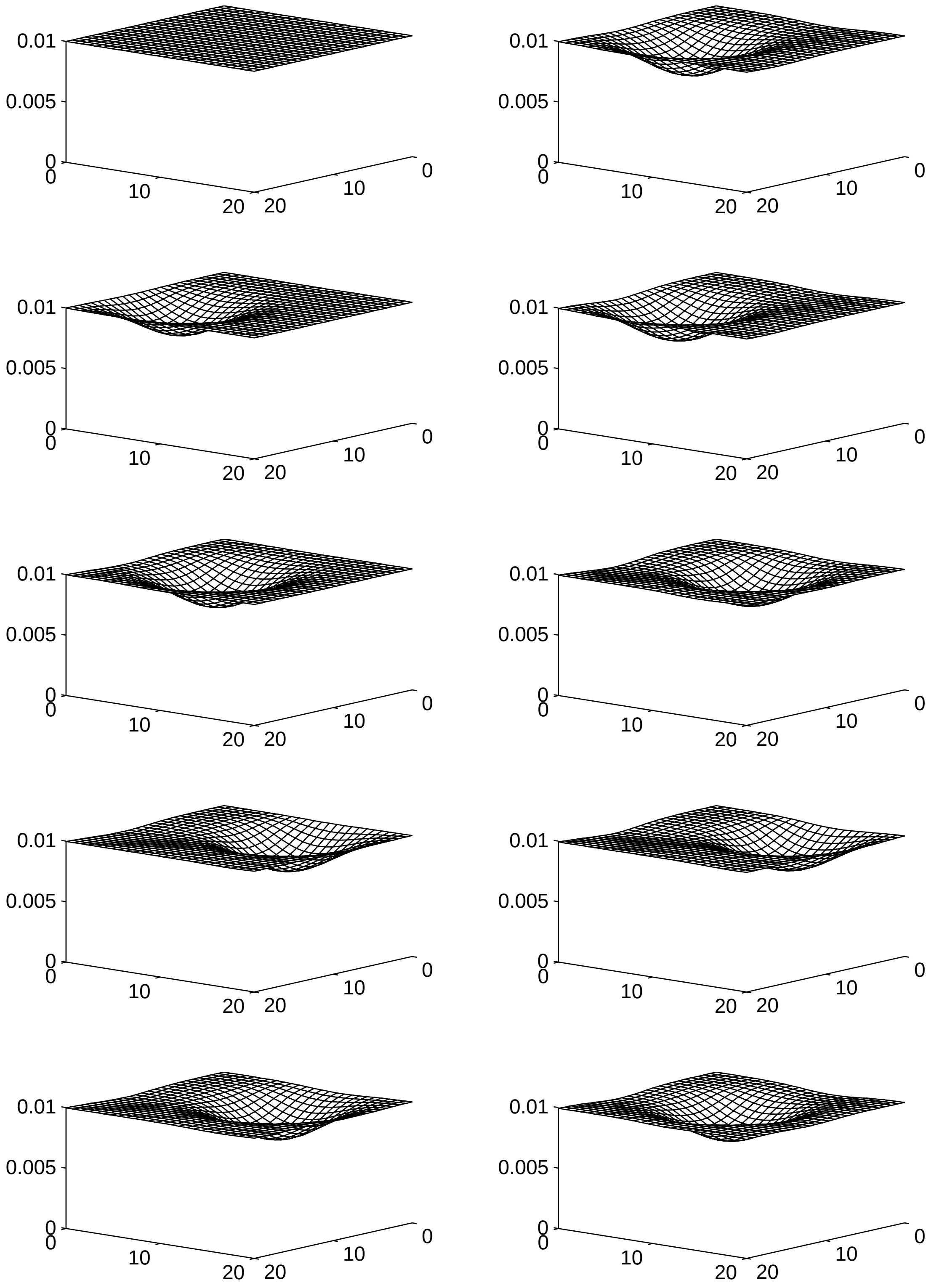
Department of Numerical Mathematics

NM-R9503 1995

Report NM-R9503
ISSN 0169-0388

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


50
0
0
10
20
20
10
0


0.01
0.005
0
0
10
20
20
10
0