1991

P.A. Zegeling, J.G. Verwer, J.C.H. van Eijkeren

Application of a moving-grid method to a class of 1D brine transport problems in porous media

Department of Numerical Mathematics Report NM-R9112 June

CWI, nationaal instituut voor onderzoek op het gebied van wiskunde en informatica

CWI is the research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a non-profit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands organization for scientific research (NWO).

Application of a Moving-Grid Method to a Class of 1D Brine Transport Problems in Porous Media

P.A. Zegeling*, J.G. Verwer

Centre for Mathematics and Computer Science (CWI)

P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

J.C.H. van Eijkeren

National Institute of Public Health and Environmental Hygiene (RIVM)

P.O. Box 1, 3720 BA Bilthoven, The Netherlands

The background of this paper is the study of transport of pollutants by groundwater flow when released from a repository in a rock salt formation. Flow in regions surrounding such formations may be strongly influenced by variations in salt concentrations, a factor requiring special attention in the development of realistic mathematical models for predicting transport of pollutants. Indispensable for this development are advanced numerical methods. The aim of this paper is to illustrate the application of such a method to a class of nonlinear, brine transport problems in one space dimension. Our method is based on the method-of-lines for solving time-dependent partial differential equations. The method is of the finite-difference type, implicit, and thus applicable to wide classes of (one-space dimensional) partial differential equation systems. The main feature of the method, however, is that it can automatically move the spatial grid for evolving time and thus is able to refine the grid in regions with large spatial transitions. The grid refinement has proven to be a very valuable facility in the numerical modeling of brine transport problems involving low and high salt concentrations. From the user's point of view, an additional advantage of the moving-grid method is that it can be implemented in advanced, user-oriented method-of-lines software packages based on implicit stiff ODE solvers. In the brine transport application discussed here we have used the package SPRINT.

1980 Mathematics subject classification : Primary: 76S05. Secondary: 65M20, 65M50.
1987 CR Categories : G.1.8.
Key words & Phrases : fluid-flow/solute-transport in porous media, groundwater flow, moving-grid finite-difference method, method of lines.
Note : This paper will be submitted for publication elsewhere.

* This work has been carried out in connection with a joint CWI/Shell project on 'Adaptive Grids'. For this project Paul Zegeling has received support from the 'Netherlands foundation for the Technical Sciences' (STW), Contract no. CWI 59.0922.

1. INTRODUCTION

The subject of this paper originates from the problem of disposal of hazardous waste, e.g., high-level radioactive waste, in salt formations. The most probable mechanism for release of these wastes to the biosphere is by transport via groundwater. Existing standard mathematical models for the study of groundwater flow and brine transport assume that the salt concentration is less than or equal to seawater concentration. This, however, is not true for flows in the vicinity of rock salt formations. In the vicinity of these formations, e.g. salt domes, the salt concentration may become very large and in fact to an extent that the groundwater flow is really influenced by the salt concentration. Recent theoretical and experimental hydrological studies indicate that for such high-concentration

situations the involved basic equations of flow and transport need to be modified [9,10]. This involves a significant effort in numerical modeling since the partial differential equations (PDEs) which show up cannot be solved by analytical means. The contents of the current paper has its origin in part of these numerical modeling studies.

We discuss the application of a numerical moving-grid method, originally developed for general time-dependent PDEs in one space dimension, to a specific class of nonlinear, brine transport problems borrowed from [7]. Our purpose is twofold. Firstly, while focussing on the application, we wish to show that this numerical method is a valuable tool for modeling nonlinear (brine) transport problems in one-space dimension, specifically so for problems having solutions with rapid transitions, such as a solute front transported in the soil or a sharp fresh-salt water interface. Secondly, while now focussing on the numerical analysis aspects, we wish to show that for the class of transport problems chosen, the grid-movement approach is successful and may provide a notable improvement compared to the more traditional approach of time-stepping on a fixed spatial grid.

The numerical method is based on the well-known numerical method-of-lines (MOL) approach for solving time-dependent PDEs (see, e.g., [5], Ch. 10 and [11]). The method is of the finite-difference type and implicit, and thus applicable to wide classes of one-space dimensional PDE systems. In addition, the main feature of the method is that for evolving time it automatically refines the spatial grid in regions with large spatial transitions. Since it is a Lagrangian type method, in many cases of practical interest the grid movement also softens the solution behaviour in time, so that larger time steps can be taken than on a fixed spatial grid. The actual moving-grid algorithm underlies the principle of spatial equidistribution and is provided with appropriate grid regularization procedures to cater for smooth grid trajectories. The principal ideas for this regularization emanate from [6] and a further comprehensive discussion of the complete moving-grid algorithm can be found in [14] (see also [8] and the references therein).

From the user's point of view, an additional advantage of the moving-grid method is that it can be implemented in most of the method-of-lines software packages based on sophisticated implicit stiff ODE/DAE solvers. We mention, for example, the BDF solvers developed by Gear, Byrne, Hindmarsh, Petzold and others (see e.g. [4]). In the brine transport problem application discussed here, we have used the FORTRAN package SPRINT [1]. SPRINT (Software for Problems in Time) is a package developed for solving general algebraic, ordinary and partial differential equations. So far SPRINT has been used mainly for one-space dimensional problems, as its core is formed by implicit stiff ODE/DAE solvers (of BDF type). In [3] SPRINT has been provided with a software interface based on the moving-grid method here considered. This MGI (moving-grid interface), being an extension of the fixed-grid interface based on [12], is a most convenient tool for researchers who wish to concentrate on modeling their physics, since it automatically carries out the spatial discretization, thus relieving them from numerical choices to be made and saving programming time. The use of MGI merely requires that the mathematical problem be formulated in terms of FORTRAN statements. Consequently, both the spatial discretization and the temporal integration can then be left to the package and the user only has to set to some numerical control parameters, like a local tolerance parameter for the numerical integration in time, the number of points for the spatial discretization, and some parameters controlling the grid movement. In the experiments reported here, we have used the MGI from [3].

The contents of the paper is as follows. Section 2 is devoted to the moving-grid method. Here an outline is given on the most important properties and principles of this numerical method, including a brief discussion of the involved numerical control parameters. Knowledge of the effect of these control parameters is required in order to be able to apply the numerical method in a sensible way. The class of 1D coupled fluid-flow/salt-transport problems we focuss on is discussed in Section 3. The physical properties involved here are advection-dispersion and in case of dominant advection solutions with rapid spatial and temporal transitions arise. In Section 4 we then present results of a number of illustrative numerical tests with members of this problem class, thereby emphasizing the occurrence of the rapid transitions and the use of the moving-grid method to handle these efficiently. The final Section 5 is devoted to concluding remarks.

2. THE MOVING-GRID ALGORITHM AND ITS IMPLEMENTATION

In this section a brief introduction is given to the moving-grid algorithm, to its implementation, and to its use. As mentioned before, the main principles behind the algorithm are due to [6]. For a comprehensive discussion on theoretical aspects of the method the interested reader is further referred to [14,8].

2.1. The moving-grid algorithm

We will present the algorithm along the lines of the numerical method-of-lines (MOL) approach for solving time-dependent PDEs. Consider an abstract Cauchy problem for a system of PDEs in one space dimension,

$$(2.1) \qquad \frac{\partial u}{\partial t} = f(u), \quad x_L < x < x_R, \quad t > 0,$$

where $u = u(x,t)$ and f represents a spatial differential operator of at most order 2. This operator may depend explicitly on the spatial variable x and the temporal variable t as well, but this dependence is suppressed in our notation. For the time being we do not discuss boundary conditions, since these are dealt with in the usual way. Throughout it is assumed that the exact solution u has (a sufficient number of) finite temporal and spatial derivatives. It is emphasized that these are allowed to be very large. We thus focuss on problems possessing solutions u with very large spatial and temporal variations, but do not consider problems with genuine discontinuous solutions.

In the MOL approach the discretization of PDEs like (2.1) is carried out in two stages. In the first stage $f(u)$ is discretized on a selected space mesh, which converts the Cauchy problem (2.1) into a Cauchy problem for a system of ODEs with time t as independent variable. The second stage of the discretization then deals with the numerical integration in time of this semi-discrete system. Let us discuss the first stage, which here takes place in a moving reference frame. First we choose N time-dependent grid points $X_i(t)$, $1 \le i \le N$, defining the space grid

$$(2.2) \qquad X: \; x_L = X_0 < \ldots < X_i(t) < X_{i+1}(t) < \ldots < X_{N+1} = x_R, \qquad t \ge 0.$$

As yet the trajectories $X_i(t)$ are unknown, but they are supposed to be (sufficiently often) differentiable. Next, along each trajectory $x(t) = X_i(t)$ we introduce the total derivative

$$(2.3) \qquad u' = x'\, u_x + u_t = X'_i\, u_x + f(u), \qquad 1 \le i \le N,$$

and spatially discretize the space operators $\partial/\partial x$ and f so as to obtain the Lagrangian semi-discrete system

$$(2.4) \qquad U'_i = X'_i\, [(U_{i+1} - U_{i-1})/(X_{i+1} - X_{i-1})] + F_i, \qquad t > 0, \quad 1 \le i \le N.$$

Here, U_i and F_i represent the semi-discrete approximation to their exact counterparts u and f at the point $(x,t) = (X_i(t),t)$. The finite-difference replacement for f is, in principle, still free to choose. We discuss this in Section 2.3. Note that we use the standard, central finite-difference approximation for u_x. Also note that the boundary values U_0 and U_{N+1} are to be defined from the semi-discretization of the physical boundary conditions.

At this stage of development the internal grid points X_i are still free to choose. The purpose is to let them move in an automatic way such that the grid X becomes fine in regions of high spatial activity and coarse in regions where the spatial variation is low. One way to accomplish this is to apply the *equidistribution* idea. For this purpose we introduce the point-concentration values [6]

$$(2.5) \qquad n_i = (\Delta X_i)^{-1}, \quad \Delta X_i = X_{i+1} - X_i, \qquad 0 \le i \le N,$$

and the equidistribution equation

$$(2.6) \qquad n_{i-1} / M_{i-1} = n_i / M_i, \qquad 1 \le i \le N,$$

where $M_i \ge \sqrt{\alpha} > 0$ represents a so-called monitor value that reflects the variation in space. The parameter $\alpha > 0$ serves to ensure that M_i remains strictly positive. Trivially, by (2.6), n_i is proportional with M_i. Hence the larger M_i, the smaller the grid interval $[X_i,X_{i+1}]$. Thus the equidistribution idea assumes that if some measure of the spatial error is available, here taken to be represented by M_i, then a good choice for the grid X would be one for which the error is equidistributed over X.

In applications the monitor M_i is usually taken to be a semi-discrete replacement of a solution functional m(u) containing one or more spatial derivatives (note that the variables U_i and X_i are still time continuous). Lest we miss the obvious, the choice of monitor is important because it plays a decisive role in the actual local grid refinement. Following [2,8,14], in the present implicit MOL approach we advocate the first derivative monitor

$$(2.7) \qquad M_i = \Big(\alpha + \frac{1}{NPDE} \sum_{j=1}^{NPDE} \beta_j\, (\Delta U_i^j)^2\, (\Delta X_i^j)^{-2}\Big)^{1/2}, \qquad \Delta U_i^j = U_{i+1}^j - U_i^j \ .$$

Here NPDE denotes the number of partial differential equations of the system (2.1) and U_i^j is the j-th component of the vector variable U_i. Note that, at a given point of time, (2.7) is a semi-discrete replacement of $m(u) = (\alpha + \|u_x\|^2)^{1/2}$, where $\|\ .\ \|$ denotes the involved weighted Euclidean norm. With $\alpha = 1$ we have the well-known arc-length monitor which places grid points along uniform arc-length intervals. We use α as a free parameter which can eventually be used for tuning purposes. In fact, the main purpose of this tuning parameter is to keep the monitor values positive, saying that a small value of α suffices. Clearly, α should not be taken too 'large' compared to the maximum of $\|u_x\|^2$, since this would result in a uniform grid, approximately. The weighting parameters β_i in (2.7) serve to make it possible to let certain components dominate the equidistribution. This may be desirable in case of a badly scaled problem, for example. The actual choice of the monitor parameters $\alpha, \beta_1, ..., \beta_{NPDE}$ will influence the outcome of a numerical simulation and, therefore, their optimal choice is problem dependent. On the other hand, our experience is that with the monitor (2.7) the method is quite robust and that a bad parameter choice merely effects the resulting accuracy. This means that given a well described problem class, like the brine transport problems, a close-to-optimal choice is normally easy to determine. We will discuss the monitor-parameter selection further with the actual numerical examples.

Finally, a slight drawback of arclength-type monitors like (2.7) is that these emphasize steep solution parts but tend to neglect, to some extent, the important transition regions. For example, when the solution has the form of a steep travelling front, such as a fresh/salt water front, usually many points are placed in its steepest part where the first derivative is maximal, but less at the top and foot of the front. One then often observes that part of the points in the steepest part are redundant. Curvature monitors, as based for example on $m(u) = (\alpha + \|u_{xx}\|^2)^{1/4}$, form a nice remedy here since a curvature monitor will emphasize the foot and top of the front and not spoil points just within the steepest part where they are not needed. Unfortunately, experiments reported in [2] have revealed that in combination with the implicit MOL approach followed here, curvature monitors readily render the grid-equation system (2.6) too nonlinear for an efficient and robust time-stepping process. As a result, it is yet more attractive to work with (2.7). Furthermore, the abovementioned drawback is partly remedied by a positive side-effect of the spatial-grid smoothing facility which we discuss next.

2.2. Grid smoothing

The Cauchy ODE problem resulting from the first MOL stage thus reads

$$(2.8a) \quad U'_i = X'_i \, [(U_{i+1} - U_{i-1})/(X_{i+1} - X_{i-1})] + F_i, \quad t > 0, \quad 1 \le i \le N,$$

$$(2.8b) \quad n_{i-1} / M_{i-1} = n_i / M_i, \quad t > 0, \quad 1 \le i \le N.$$

After prescribing the initial data for both U_i and X_i, $1 \le i \le N$, and the boundary values U_0 and U_{N+1} from a semi-discretization of the physical boundary conditions, system (2.8) can be numerically integrated in time so as to obtain the final fully discrete solution on the moving grid X. However, since (2.8b) prescribes X in an implicit way in

terms of the unknowns U_i, there is little control over the grid movement. For example, it may happen that the grid distance ΔX_i varies extremely rapidly over X or that for evolving time the trajectories $X_i(t)$ tend to oscillate. A too large variation in ΔX_i may be detrimental to spatial accuracy and it is clear that temporal grid oscillations do hinder the numerical time-stepping since the grid trajectories are computed automatically by numerical integration. Following [6,8,14], we therefore employ two so-called grid-smoothing procedures, one for generating a spatially smooth grid and the other for avoiding temporal grid oscillations. This involves a modification of the grid-equation system (2.8b).

The modified grid-equation system is given by

$$(2.9) \qquad (\mathbf{n}_{i-1} + \tau\frac{d}{dt}\mathbf{n}_{i-1}) / M_{i-1} = (\mathbf{n}_i + \tau\frac{d}{dt}\mathbf{n}_i) / M_i, \qquad t > 0, \quad 1 \le i \le N,$$

where $\mathbf{n}_i = n_i - \kappa(\kappa+1)\,(n_{i+1} - 2n_i + n_{i-1})$ with $n_{-1} = n_0$, $n_{N+1} = n_N$. We note in passing that in the actual implementation n_i is replaced by $(\Delta X_i)^{-1}$ and n'_i by $-\Delta X'_i/(\Delta X_i)^2$. The modification thus results in a 5-point coupled, time-dependent grid-equation system. A consequence of the grid-smoothing is that, in addition to the monitor parameters $\alpha, \beta_1, ..., \beta_{NPDE}$, two new grid parameters have been introduced, namely κ and τ.

The parameter $\kappa \ge 0$ is connected with the spatial grid-smoothing. One can prove that any grid X solving (2.9) satisfies

$$(2.10) \qquad \frac{\kappa}{\kappa+1} \le \frac{n_{i-1}}{n_i} \le \frac{\kappa+1}{\kappa},$$

showing that we have control over the variation in ΔX_i. Through κ we can now easily control grid clustering and grid expansion. Loosely speaking, the monitor function still determines the shape of X and the new parameter κ the level of clustering. Note that the extreme value $\kappa = \infty$ yields a uniform grid.

Of importance is to emphasize that for a given number of points N, and any given distribution of monitor function values M_i, κ determines the minimal and maximal interval lengths. In actual application the minimum should of course be related to the expected small scale features in the sought solution. Let us therefore, by way of illustration, suppose that in a transition from small to large gradients and back, a solution requires a local refinement with a factor of 10^m. Let N_{loc} be the number of points in this transition region and suppose that precisely half this number is used in the transition from small to large gradients, and the remaining half backwards. Further suppose that in the transition the bounds of (2.10) apply. Then, to determine the required minimum for N_{loc}, we may put $(1+1/\kappa)^{0.5N_{loc}} = 10^m$, so that N_{loc} should at least be equal to

$$(2.11) \qquad N_{loc} = 2m\,\ln(10) / \ln(1+1/\kappa) \approx 4.6m / \ln(1+1/\kappa).$$

For example, for $m = 3$ and $\kappa = 1$ and 2, we have, respectively, $N_{loc} \approx 20$ and 34. Using the 'rule of thumb' (2.11), one can now make a quick estimate of the number of points needed for a particular problem by summing the

minimum number required to solve each small scale feature. Needless to say that this estimate is somewhat crude. Finally, while referring to the discussion on the arclength-type monitor at the end of Section 2.1, we see from (2.10) that the larger κ will be taken, the more points will be placed at the top and foot of a steep front. Of course, we now tacitly assume that in the steepest part of the front there are already enough points to let this happen. In our application we usually choose $\kappa = 2$. To our experience, with this value of κ we not only obtain a rather modestly graded space grid, but also keep a sufficient number of points within the actual transitions of $\partial u/\partial x$.

The parameter $\tau \geq 0$ is connected with the temporal grid-smoothing and serves to act as a delay factor for the grid movement. More precisely, the introduction of the temporal derivative of the grid X forces the grid to adjust over a time interval of length τ from old to new monitor values, which provides a tool for suppressing grid oscillations and hence to obtain a smoother progression of X(t). However, it is emphasized that choosing τ too large will result in a grid X that lags too far behind any moving steep spatial transition. In fact, it can be shown that for $\tau \rightarrow \infty$ a nonmoving grid results. In situations where temporal grid-smoothing is really advisable, one should therefore choose τ not too large. For practical purposes a good choice is one which is close to the minimal temporal stepsize taken in the numerical integration, so that the influence of past monitor values is felt only over one or a few time steps.

2.3. Integration in time

Conform to the MOL approach we now have semi-discretized the PDE system (2.1) on an approximately equidistributing, moving grid. The semi-discrete formulation consists of the combined equations (2.8a) - (2.9), where the relations $n_i = (\Delta X_i)^{-1}$ and $n'_i = -\Delta X'_i/(\Delta X_i)^2$ are used to convert the dependence on the point concentration values into a 'natural' dependence on the grid points X_i. We recall that the boundary values U_0 and U_{N+1} are to be defined from the spatial discretization of physical boundary conditions, just as when using only equation (2.8a) on a nonmoving grid. The combined equations can be rearranged in the standard, linearly implicit ODE system form

$$A(Y)Y' = L(t,Y), \quad t > 0, \quad Y(0) \text{ given}, \tag{2.12}$$

where the vector variable Y assumes the natural ordering of unknowns U_i^j, X_i. Hence,

$$Y = (\ldots , U_i^1, \ldots , U_i^{NPDE}, X_i, \ldots). \tag{2.13}$$

Form (2.12) is a standard format for various well-known stiff ODE/DAE solvers. Note that without temporal grid-smoothing ($\tau = 0$) equation (2.9) is of purely algebraic form, so that (2.12) then becomes a differential-algebraic equation (DAE) system. As already mentioned in the introduction, the numerical results in this paper have been obtained with the LSODI-based BDF solver of the SPRINT package. A similar solver is DASSL [4] which we have also applied successfully elsewhere [14]. From the user's point of view it is of interest to note that in our moving-

grid application these solvers are employed in essentially the same way as in the conventional nonmoving MOL approach. In Section 2.5 we will summarize what is to be done for calling them.

2.4. A moving-grid interface

As the integration in time is done automatically by the stiff integrator, it makes sense to also automize the spatial discretization of the PDE operator with its boundary conditions. This is particularly attractive for researchers who wish to concentrate on modeling their physics, since it saves programming time and relieves them from numerical choices to be made. Such a FORTRAN interface for use with the moving-grid method has been developed in [3]. We have also used this interface, called MGI, in the tests reported in Section 4.

MGI is an extension of the fixed-grid interface from [12] which is available in the SPRINT package. The discretization is based on a central 2-nd order finite-volume scheme and covers the following PDE system:

$$\text{(2.14a)} \quad \sum_{k=1}^{NPDE} C_{jk}(x,t,u,u_x) \frac{\partial u^k}{\partial t} = x^{-m} \frac{\partial}{\partial x} (x^m R_j(x,t,u,u_x)) - Q_j(x,t,u,u_x), \quad x_L < x < x_R, \quad t > 0.$$

Index j runs from 1 to NPDE, u^k is the k-th component of the vector-valued function u, and R_j, Q_j can be thought of as flux and source or sink terms, respectively. The parameter m serves to cover polar co-ordinates (m = 1 or 2). In our present application we work in Cartesian co-ordinates and thus m = 0. The coefficient functions C_{jk}, R_j and Q_j are supposed to be at least continuous.

The boundary conditions should fit in the MGI master form

$$\text{(2.14b)} \quad \chi_j(x,t))R_j(x,t,u,u_x) = \gamma_j(x,t,u,u_t,u_x), \quad x = x_L, x_R,$$

and at the initial time the standard initial condition $u(x,0) = u_0$ is supposed. For a detailed description of MGI with its underlying spatial discretization we refer to [3]. The fluid-flow/salt-transport problem discussed in the Section 3 fits in the master form (2.14).

2.5. Initialization and choice of parameters

The application of MGI and the stiff ODE solver requires three main tasks from the user: (i) Formulation of the PDE problem with the boundary conditions, (ii) Initialization of solution and grid variables, and (iii) Choice of control parameters. These are, for the spatial discretization, the monitor parameters α, β_1, ... , β_{NPDE} and the grid-smoothing parameters κ, τ, and for the temporal integration the choice of norm for error measurement and local tolerance parameters *atol* (absolute error) and *rtol* (relative error) that govern the automatic selection of stepsize and order in the BDF code implemented in SPRINT. Task (i) involves writing two FORTRAN subroutines in a prescribed format. Since this is merely technical, and outside the scope of the present paper, we refer for this to [3] where some illustrative examples are discussed. In the remainder of this section we pay attention to (ii) and (iii).

Trivially, the initial values $U_i(0)$ are taken from the physical initial function u(x,0) and thus equal $u(X_i(0),0)$. The choice for X(0) is less trivial, however, and needs some further comments. Suppose that u(x,0) is already locally very steep. Then it seems desirable to adjust X(0) to this initial solution profile. A natural choice for X(0) then would be the grid that solves, at the initial time t = 0, the spatially smoothed grid-equation system

$$(2.15) \quad \mathbf{n}_{i-1} / M_{i-1} = \mathbf{n}_i / M_i, \quad 1 \le i \le N.$$

Because $U_i(0) = u(X_i(0),0)$, (2.15) constitutes a system of N nonlinear algebraic equations in the N unknowns $X_i(0)$. Hence this would require a separate nonlinear equation procedure for solving X(0), prior to the numerical integration process. Fortunately, an interesting side-effect of the temporal grid-smoothing defined by equation (2.9), is that such a separate procedure is redundant. We will explain this.

For τ small, (2.9) can be interpreted as a singularly perturbed form of (2.15) and, consequently, for t close to 0 the grid X(t) very rapidly adjusts itself to a grid which is close to the grid that would be obtained when using system (2.15) for $t \ge 0$. Because we use an implicit method, in fact the implicit Euler method is used at the start, this rapid grid adjustment is accurately followed by the integration method and normally after one or a few integration steps the grid is already in good position. Hereby it is assumed that the parameter τ is, approximately, of the same size as the initial stepsize. Inaccuracies due to a crude initial guess for X(0) are not felt then, because due to the implicitness the PDE is not discretized at the initial time t = 0. Of course, if a good guess for X(0) is available, then the user is advised to use it.

To our experience, the actual choice made for the grid parameters $\alpha, \beta_1, \ldots, \beta_{NPDE}$ and κ, τ will influence the performance of the moving-grid method mainly in connection with the spatial accuracy, and much less so in connection with robustness. This makes it fairly easy to quickly find a near-optimal choice for the class of problems at hand. Because this is most clearly demonstrated through actual numerical examples, we will specify the parameter set with the numerical examples in Section 4.

3. THE 1D FLUID-FLOW / SALT-TRANSPORT PROBLEM

Disposal of radioactive wastes in rock salt formations, like salt domes, is being considered as a serious possibility by a number of countries. An integral part of the safety assessment of nuclear waste disposal is the study of mathematical models for nuclide transport to the geosphere via groundwater flow. Existing standard models for groundwater flow and salt transport assume that the salt concentration is less than or equal to seawater concentration. In such low-concentration situations the models in use have been sufficiently validated and in many cases of interest the fluid flow and the salt concentration equation can be treated uncoupled. However, for flows in the vicinity of rock salt formations the salt concentration may become high and influence the fluid density to an extent that it effects the fluid flow. On the other hand, salt is transported by the fluid and thus fluid flow and salt transport are mutually coupled. The existing standard models and their uncoupled treatment are then no longer adequate for safety assessment which makes it interesting to study this intricate situation. Recent theoretical and experimental hydrological studies

[9,10] indicate that for such high-concentration situations the involved basic equations of flow and transport need to be modified, which requires a significant effort in numerical modeling. At this stage the moving-grid method enters the scene, because in the high-concentration situations studied also large concentration gradients prevail, making the use of fixed-grid methods inefficient.

In the modeling of transport of M solutes by groundwater flow generally M + 1 sets of equations appear, viz. one set for each solute and a set for the flowing fluid [9]. The set for the fluid, which is in our case brine, constitutes the fundamental balance of mass property of the fluid supplemented with a Darcy-law expressing conservation of momentum. Similarly, for each solute the associated set constitutes the balance of mass property supplemented with conservation of momentum through a Fickian-type law. If temperature changes are important, an energy equation should be added. Also, if deformation effects of the porous medium and porosity changes are important, then an additional set of equations for the solid phase of the porous medium has to be provided. In the present study we do not consider temperature or deformation effects and assume only one solute, the salt. We thus consider an isothermal, single-phase, single-component saturated flow model in the idealized case of one space dimension. It is further assumed that no external body forces except gravity exist and that the two brine components, water and salt, do not react or adsorb. This specific model, which we have borrowed from the RIVM report [7], has been selected for demonstration purposes. We emphasize that the combined use of the moving-grid interface and the stiff ODE solver readily admits the numerical treatment of more complicated models.

The model comprises the following set of equations. For the fluid we have

$$\frac{\partial}{\partial t}(n\rho) + \frac{\partial}{\partial x}(\rho q) = 0, \tag{3.1a}$$

$$q = -\frac{k}{\mu}\left(\frac{\partial p}{\partial x} + \rho g\right), \tag{3.1b}$$

and for the salt

$$\frac{\partial}{\partial t}(n\rho\omega) + \frac{\partial}{\partial x}(\rho\omega q + \rho J) = 0, \tag{3.2a}$$

$$J = -\lambda\,|q|\,\frac{\partial \omega}{\partial x}. \tag{3.2b}$$

The fluid density ρ is supposed to obey the equation of state

$$\rho = \rho_0 \exp\left(\beta(p - p_0) + \gamma\omega\right), \tag{3.3}$$

with constant reference density ρ_0, constant reference pressure p_0, constant compressibility coefficient β, and constant salt coefficient γ. Other constants are porosity n, permeability k, viscosity μ, gravity g and dispersion length λ. The various variables are the (Darcy) velocity of the fluid q, the hydrodynamic pressure p and the salt-mass

fraction ω. We thus consider the medium to be homogeneous with respect to porosity, permeability and viscosity. However, inhomogeneities, and also sources and sinks, can easily be taken into account in the numerical solution process.

The set of equations can be formulated as a system of two PDEs with pressure p and salt concentration ω as independent variables. To this end we compute, from (3.3),

$$\frac{\partial \rho}{\partial t} = \rho\beta \frac{\partial p}{\partial t} + \rho\gamma \frac{\partial \omega}{\partial t} \tag{3.4}$$

and subsitute into (3.1a) to obtain the fluid-mass balance equation

$$n\rho\beta \frac{\partial p}{\partial t} + n\rho\gamma \frac{\partial \omega}{\partial t} = - \frac{\partial}{\partial x}(\rho q). \tag{3.5}$$

Further, substituting (3.1a) into (3.2a) yields the salt transport equation

$$n\rho \frac{\partial \omega}{\partial t} = - \rho q \frac{\partial \omega}{\partial x} - \frac{\partial}{\partial x}(\rho J). \tag{3.6}$$

A trivial inspection shows that system (3.5) - (3.6) fits into the MGI format (2.14a).

We have used this form as input for the numerical solution method. A few comments are in order. First, substitution of the expression for J into (3.6) yields the advection-dispersion equation

$$n\rho \frac{\partial \omega}{\partial t} = - \rho q \frac{\partial \omega}{\partial x} + \frac{\partial}{\partial x}(\rho\lambda|q| \frac{\partial \omega}{\partial x}), \tag{3.7}$$

showing that in the present model the physical salt-transport phenomena are advection and dispersion. Molecular diffusion is absent here. It is easily built in, however, since this merely amounts to adding a small constant to $\lambda |q|$. Assuming 'frozen' coefficients, we see that the Peclet number is

$$Pe = | \frac{L\rho q}{\rho\lambda|q|} | = \frac{L}{\lambda} , \tag{3.8}$$

where L denotes the physical length of the medium. Hence, for $\lambda << L$ advection dominates and this is just the physical situation that gives rise to steep concentration gradients. Another point worth to mention is that the compressibility coefficient β is very small compared to the salt coefficient γ. In fact, it is often zero, in which case the balance equation (3.5) reduces to

$$n\rho\gamma \frac{\partial \omega}{\partial t} = - \frac{\partial}{\partial x}(\rho q) \tag{3.9}$$

and $\partial p/\partial t$ is absent. We then have two equations for $\partial\omega/\partial t$ of which (3.5) can be rewritten to a PDE containing only spatial derivatives. Hence, this rules out the possibility of explicit time-stepping. Note that if we would also put $\gamma = 0$, that then the density ρ is constant and the mass balance equation reduces to the simple pressure equation

$$(3.10) \qquad \frac{\partial^2 p}{\partial x^2} = 0.$$

Of course, a zero salt coefficient γ is not realistic in our application. However, we emphasize that the numerical method can handle these different situations by just integrating the system (3.5) - (3.6), illustrating the versatility of the current numerical MOL approach.

To complete the model description, we must give the initial and boundary conditions. Defining the space-time domain as [0,L] x [0,T], the initial and boundary conditions we have imposed for $\omega(x,t)$ and $p(x,t)$ are, respectively,

$$(3.11a) \qquad \omega(x,0) = 0, \qquad 0 \le x \le L,$$

$$(3.11b) \qquad \omega(0,t) = \omega_0 > 0, \quad \frac{\partial \omega}{\partial x}(1,t) = 0, \qquad 0 < t \le T,$$

and

$$(3.12a) \qquad p(x,0) = p_0\,[(1-x/L)p_{left} + (x/L)p_{right}], \qquad 0 \le x \le L,$$

$$(3.12b) \qquad p(0,t) = p_0\,p_{left}, \quad p(L,t) = p_0\,p_{right}, \qquad 0 < t \le T,$$

where ω_0 is the left-end salt concentration and p_{left} and p_{right} are pressure coefficients. Note that (3.11), (3.12) easily fit into the MGI master format (2.14b).

We have selected these conditions with the aim of generating a travelling salt front. Note that at the initial time there is no salt in the medium and that the inflow value $\omega_0 > 0$. Hence, assuming appropriate model data, this should give rise to a travelling front. The steepness and speed of the front will of course be determined by the complete set of physical data. A characteristic set is given in Table 1. This table comprises all data needed to run the problem, except for the end time T, the dispersion length λ, and the pressure coefficients p_{left} and p_{right}. Finally, for numerical work, a.o. to quickly see how the various numerical control parameters are to be chosen, but also for interpreting and comparison purposes, it is more convenient to solve the problem in scaled, dimensionless form, which is what we have done. We refer to Table 1 for the scaling relations with the dimensionless values of all quantities involved. From these relations one can check that all equations are left invariant (note that this also holds for (3.3) due to the fact that after scaling $p_0 = 1$). Hence, in the remainder we have worked with the same set of equations as discussed above. The pressure coefficients p_{left} and p_{right} are left unchanged and will be specified with the numerical examples.

Table 1. Model data. Bold face notation is used for the non-scaled quantities.

	Non-scaled	Scaled
Time	$\mathbf{0 < t < T}$ [sec]	$t = \mathbf{t}/\mathbf{t_0}$, $\mathbf{t_0} = \frac{\mu L^2}{\mathbf{k_0 p_0}} = 10^4$
Space	$\mathbf{0 < x < L}$ [m]	$x = \mathbf{x}/\mathbf{L}$
End time	$\mathbf{T}$ [sec]	$T = \mathbf{T}/\mathbf{t_0}$
Domain length	$\mathbf{L} = 1$ m,	$L = 1$
Pressure	$\mathbf{p}$ [kg/m/sec^2]	$p = \mathbf{p}/\mathbf{p_0}$
Salt concentration	ω	$\omega = \omega/\omega_0$
Density	ρ [kg/m^3]	$\rho = \rho/\rho_0$
Permeability	$\mathbf{k} = \mathbf{k_0} = 10^{-12}$ m^2,	$k = \mathbf{k}/\mathbf{k_0} = 1$
Viscosity	$\mu = \mu_0 = 10^{-3}$ kg/m/sec,	$\mu = \mu/\mu_0 = 1$
Reference pressure	$\mathbf{p_0} = 10^5$ kg/m/sec^2,	$p_0 = 1$
Salt inflow concentration	$\omega_0 = 0.26$	$\omega_0 = 1$
Reference density	$\rho_0 = 10^3$ kg/m^3,	$\rho_0 = 1$
Gravity force	$\mathbf{g} = 9.81$ m/sec^2,	$g = (\rho_0 \mathbf{L} g)/\mathbf{p_0} = 0.0981$
Porosity	$\mathbf{n} = 0.2$,	$n = 0.2$
Salt coefficient	$\gamma = 0.69$,	$\gamma = \gamma\omega_0 = 0.1794$
Dispersion length	λ [m]	$\lambda = \lambda/L$
Compressibility coefficient	$\beta = 10^{-10}$ msec^2/kg,	$\beta = \beta \mathbf{p_0} = 10^{-5}$

4. NUMERICAL EXAMPLES

We will present results of three numerical examples. To simplify the demonstration, these results have been obtained with a fixed set of numerical control parameters:

(4.1a) $atol = rtol = TOL = 10^{-5}$ (temporal integration)

(4.1b) $\kappa = 2, \tau = 10^{-3}, \alpha = 10^{-2}, \beta_1 = 0, \beta_2 = 1$ (grid movement)

(4.1c) $X_i(0) = \frac{i}{N+1}$, $0 \le i \le N+1$ (uniform initial grid)

SPRINT was called in standard mode, thus providing automatic initial-stepsize selection and numerical Jacobian evaluation. The Euclidean norm was used for local error control while (4.1a) was imposed for all components of the dependent-variable vector Y (see (2.13): NPDE = 2, $u_1 = p$, $u_2 = \omega$). Note that TOL = 10^{-5} is quite small. However, to accurately simulate the rapid birth of the salt front, which arises from the inconsistency between the initial and

left-end salt concentration, a small tolerance value is natural. In this connection we also emphasize that we always started on a uniform grid, just for convenience of use. This means that immediately after start the method should rapidly cluster most of the grid points near the left boundary.

Also the grid parameters take on more or less standard values, except for β_1. The choice $\beta_1 = 0$ means that the pressure gradient $\partial p/\partial x$ is not taken into account in the monitor (2.7). We decided to omit $\partial p/\partial x$ in the monitor since in our examples $\partial p/\partial x$ varies very slowly and thus acts more or less in the same manner as the constant regularization parameter α. In such cases a too large value for the near constant pressure gradient yields a unnecessarily large regularization effect. This, in turn, would imply that variations in the concentration gradient $\partial\omega/\partial x$ become of lesser importance in the spatial equidistribution than desired.

4.1. Example I

The first example is defined by the data of Table 1, together with $\lambda = 0.001$, $T = 5$ and $p_{left} = 1.7$, $p_{right} = 1.0$. With this choice of pressure initial function the arising salt front travels to the right boundary and finally renders a steady state for p and ω with p equal to the linear initial pressure and $\omega = \omega_0 = 1$. The steady state starts to settle at about $t = 2$, far before the end time $T = 5$ has been reached. Consequently, due to the uniform salt concentration, at about $t = 2$ the grid should again become uniform. Hence this example provides an interesting test for the moving-grid method. The pressure p undergoes only a marginal change for $t > 0$ and below we will therefore only plot ω.

Figure 1 depicts the grid and salt concentrations at some values of t for $N_2 = N + 2 = 25$ and 50. We see that the grid accurately reflects the anticipated solution behaviour. At very early times the grid points rapidly cluster near $x = 0$, then the cluster travels with the front and when the steady state is reached, a uniform grid appears. While $N_2 = 25$ results in a little overshoot at the top and in a little smearing at the foot, $N_2 = 50$ gives already very accurate salt concentration profiles. The profiles for $N = 100$ (not shown here) do equal those for $N = 50$ up to plotting accuracy.

Table 2 shows integration history for $N_2 = 25$, 50 and 100 and serves to provide insight in the costs of the implicit numerical integration method. The given data have the following meaning: STEPS = number of integration steps; JACS = number of Jacobian updates; RESIDS = total number of evaluations of the ODE system, including those needed for the Jacobian updates; NITER = total number of Newton iterations; CPU = central processing time on an ALLIANT/FX4 computer, using one processor. Note that our decision to start on a uniform initial grid has its price. For example, for $N_2 = 100$ more than half the number of steps is used to reach $t = 0.1$. In fact, at $t = 10^{-4}$, 10^{-3}, 10^{-2}, we have, respectively, STEPS = 39, 152, 271. A great deal of these steps is needed simply to adjust the initial grid to the very steep concentration profile at the very early times (see the right upper plot in Figure 1). Therefore, somehow adjusting the initial grid to the expected solution profile at the first forward time level will reduce STEPS significantly. We also wish to remark that the method efficiently detects the steady state, since for $t > 2$ the temporal stepsizes are rapidly increased and very few steps are required to complete the integration. Finally, we have also tabulated $\omega_{max} - \omega_0$, which is the maximal overshoot at the given points of time. We see that already for $N_2 = 25$ the overshoot is very little.

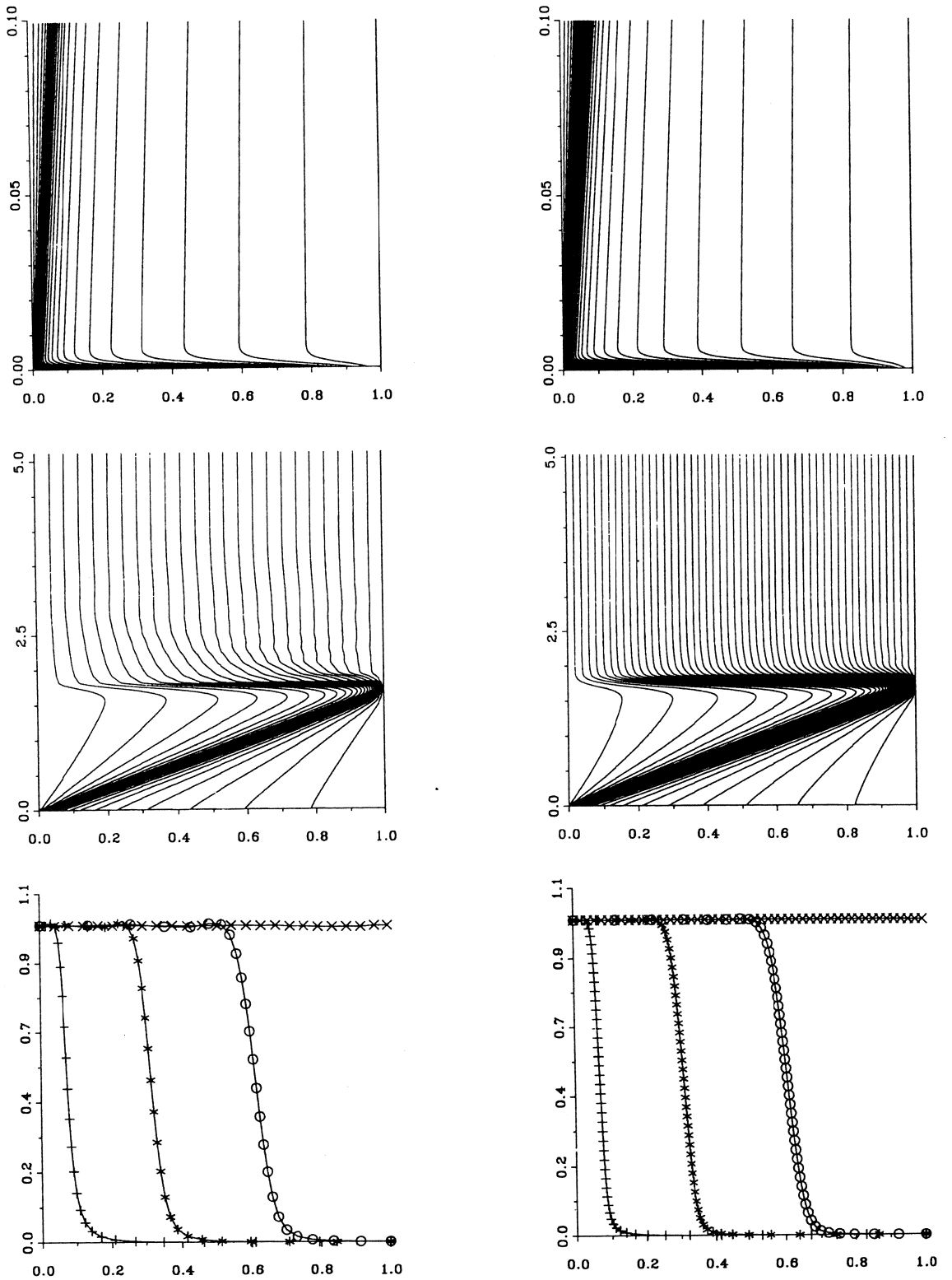

Figure 1. Example I, first-derivative monitor: Gridlines and salt concentration profiles at t = 0.1, 0.5, 1.0, 5.0. The left part of the figure corresponds with $N_2 = 25$ and the right part with $N_2 = 50$. Note the difference in scaling in each of the two gridline plots.

Table 2. Example I, first-derivative monitor: Integration histories.

		STEPS	JACS	RESIDS	NITER	ω_{max} - 1.0	CPU time (sec.)
$N_2 = 25$	t = 0.1	149	39	931	407	7.0_{10}-3	--
	t = 0.5	198	47	1175	545	7.0_{10}-3	--
	t = 1.0	220	52	1302	607	6.0_{10}-3	--
	t = 5.0	565	144	3532	1610	--	140
$N_2 = 50$	t = 0.1	202	53	1282	574	8.0_{10}-4	--
	t = 0.5	225	58	1413	638	1.0_{10}-3	--
	t = 1.0	234	61	1481	667	1.0_{10}-3	--
	t = 5.0	450	118	2904	1335	--	236
$N_2 = 100$	t = 0.1	301	83	1988	882	3.0_{10}-4	--
	t = 0.5	317	87	2085	927	3.0_{10}-4	--
	t = 1.0	326	89	2140	956	6.0_{10}-4	--
	t = 5.0	529	142	3533	1648	--	566

Table 3. Example I, second-derivative monitor: Integration histories.

		STEPS	JACS	RESIDS	NITER	ω_{max} - 1.0	CPU time (sec.)
$N_2 = 25$	t = 0.1	324	112	2627	914	3.0_{10}-3	--
	t = 0.5	549	191	4459	1543	3.0_{10}-3	--
	t = 1.0	653	236	5490	1889	3.0_{10}-3	--
	t = 5.0	1623	519	12642	4734	--	518
$N_2 = 50$	t = 0.1	419	146	3453	1222	5.0_{10}-5	--
	t = 0.5	509	182	4289	1510	8.0_{10}-5	--
	t = 1.0	542	191	4512	1596	1.0_{10}-4	--
	t = 5.0	1292	494	11470	3943	--	941

A further inspection of the salt concentration plots shows that, as expected, the first derivative monitor (2.7) places quite a number of points just within the front where $\partial\omega/\partial x$ is largest. Fortunately, the spatial grid-smoothing, resulting in relation (2.10), has the nice side-effect of keeping a substantial number of points at the foot and top of the front, where $\partial\omega/\partial x$ becomes smaller and finally zero. This only works, of course, if κ is taken not too small. Note that there should be enough points at the foot and top so as to avoid wiggles, since the spatial

discretization is based on a common central finite-volume scheme. As already mentioned in Section 2.2, in this connection a better monitor will be $m(u) = (\alpha + \|u_{xx}\|^2)^{1/4}$. To illustrate this we have again solved Example I, but now using this second-derivative instead of the first-derivative monitor. The results depicted in Figure 2, obtained for $N_2 = 25$ and 50, indeed confirm our expectation. The grid positioning near and within the front is now excellent. There are less points within the steep front and more at the foot and top of the wave, resulting in more accuracy. In fact, the result for $N_2 = 25$ is of almost the same accuracy as the previous result with 50 points. On the other hand, the integration history shown in Table 3 reveals that the time-stepping has become more costly now which, in terms of CPU time, completely annihilates the advantage of the better grid positioning near the front. Also note that the grid trajectories start to oscillate when the grid is supposed to remain fixed and uniform. This convincingly shows that the grid-equation system is now much harder to solve numerically, which is in line with our experiences reported in [2]. In fact, the main conclusion of [2] is that use of the first-derivative monitor generally leads to a more robust algorithm. We therefore prefer to use this monitor in the present implicit MOL algorithm. However, other types of algorithms, like the one in [13], may well benefit from the curvature monitor.

Finally, to present a comparison with the standard fixed-grid approach, we have also solved Example I on a fixed uniform grid. For convenience of testing, this fixed grid was enforced using the MGI with an available option that replaces the grid-equation system (2.9) by the trivial set $\dot{X}_i = 0$, $t > 0$, $1 \le i \le N$. This means that we truly simulate the standard fixed-grid approach, except that the dimension of the semi-discrete system is larger (roughly 3N instead of 2N), due to the fact that the values X_i are still treated as unknowns. Table 4 presents integration histories and maximal overshoot-values for $N_2 = 25$, 50 and 100. Of course, as exemplified by these values, these numbers of points are far from enough to give the same accuracy as on the moving grids. However, a comparison like this not only serves to give insight in accuracy performances, but is also of clear interest in connection with the use of implicit integration methods. The reason is that due to the grid-movement we have not only N additional unknowns to integrate, but the grid-equation system (2.9) defining these unknowns is also nonlinear. As a rule, more nonlinearity hinders implicit time-stepping and manifests itself in more Newton iterations, more Jacobian updates, and also more time steps. Needless to say that the additional effort should not annihilate the anticipated gain in using less grid points. Table 4 is self-evident and confirms our expectation. It indeed shows that on a fixed grid the time-stepping is easier. The required number of steps, the average number of Newton iterations per step, and the number of Jacobian updates all are smaller, resulting in significantly less CPU time for the same number of grid points (recall that the listed CPU times for the fixed-grid experiments are somewhat too large due to the fact that in reality the dimension of the semi-discrete system is about a factor 2/3 smaller). On the other hand, when we also consider accuracy, then the moving-grid approach is yet to be preferred (compare the maximal overshoot). We may conclude from this comparison that the moving-grid approach is of clear advantage when compared to the fixed-grid approach. However, the solutions sought should be sufficiently steep in order to justify the higher costs for the time stepping. This is no doubt the case with $\lambda = 10^{-3}$ and smaller values, but not if λ would be substantially larger.

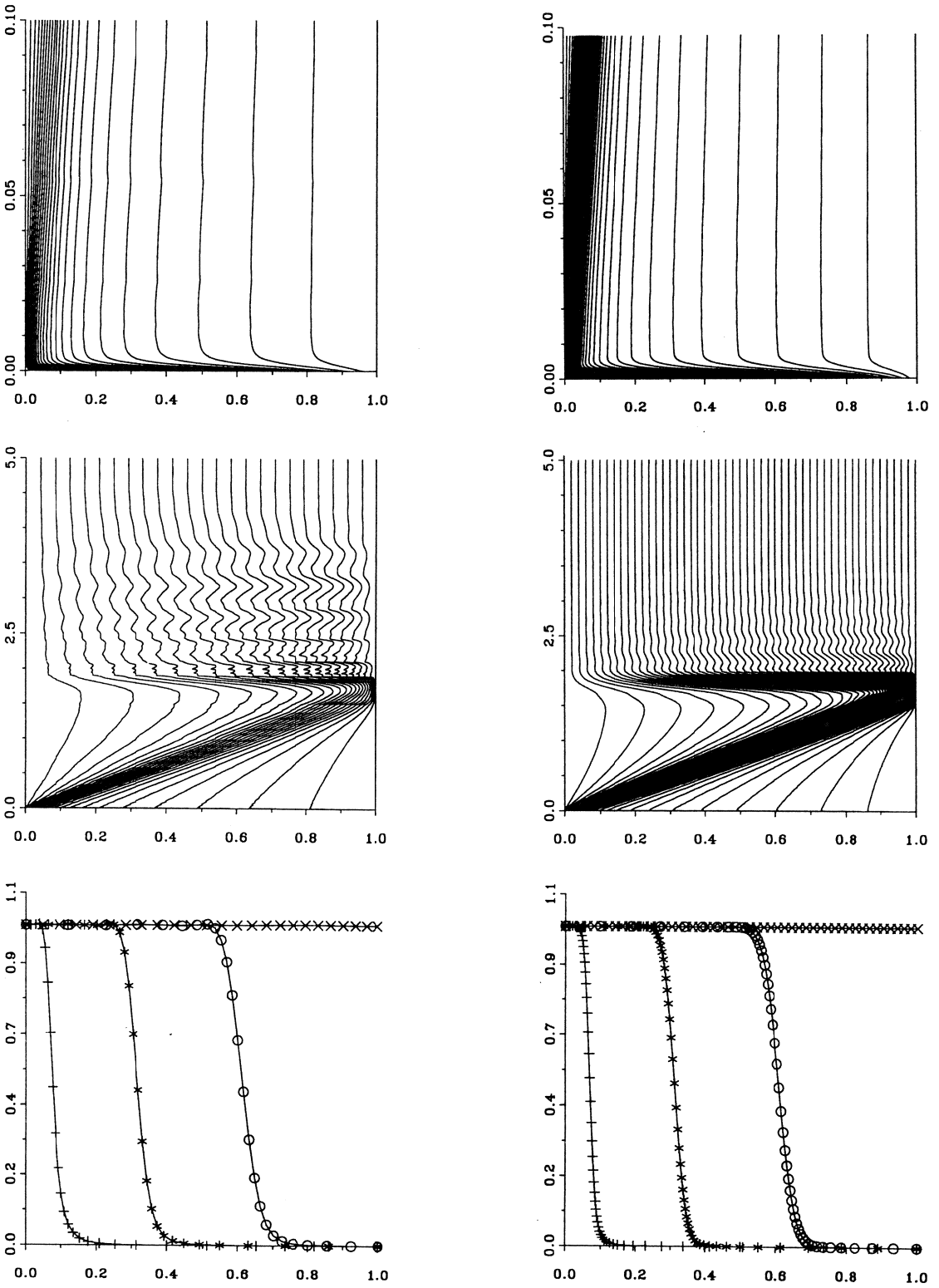

Figure 2. Example I, second-derivative monitor: Gridlines and salt concentration profiles at t = 0.1, 0.5, 1.0, 5.0. The left part of the figure corresponds with N_2 = 25 and the right part with N_2 = 50. Note the difference in scaling in each of the two gridline plots.

Table 4. Example I, nonmoving grid: Integration histories.

		STEPS	JACS	RESIDS	NITER	ω_{max} - 1.0	CPU time (sec.)
$N_2 = 25$	t = 0.1	33	14	280	87	meaningless	--
	t = 0.5	52	15	339	133	1.2_{10}-1	--
	t = 1.0	70	16	397	178	1.6_{10}-1	--
	t = 5.0	207	24	798	473	--	34
$N_2 = 50$	t = 0.1	35	14	284	91	3.0_{10}-2	--
	t = 0.5	73	16	398	179	1.1_{10}-1	--
	t = 1.0	119	19	545	285	1.2_{10}-1	--
	t = 5.0	278	32	1065	634	--	92
$N_2 = 100$	t = 0.1	42	13	287	107	6.0_{10}-2	--
	t = 0.5	100	18	490	243	6.0_{10}-2	--
	t = 1.0	158	22	673	374	4.0_{10}-2	--
	t = 5.0	256	32	1022	591	--	171

4.2. Example II

The second example is also defined by the data of Table 1, but now $p_{left} = 1.11$, $p_{right} = 1.0$, T = 500 and λ = 0.0001. The smaller pressure gradient in the initial function has two effects. First, it yields a smaller fluid velocity resulting in a larger time scale, which explains the larger value for T. The second and more interesting effect is that the travelling salt front now comes to a stand still before it has reached the right boundary. This happens at about t = 150, at which point of time the front lies near x = .6. The reason is that the fluid velocity q tends to zero, uniformly in x, which settles the system into a steady state and this takes place long before the salt front has reached the right boundary. We note that this phenomenon is rather special in the sense that it heavily depends on the initial pressure gradient. The stand still of the salt front is lost with a relatively slight change in this gradient. Obviously, the simulation of this rather subtle situation requires an accurate balancing of gravity force ρg and pressure gradient force $\partial p/\partial x$ in the Darcy velocity expression (3.1b). Finally, we have made the dispersion length ten times smaller than in the previous example, giving a Peclet number of 10.000 and a much steeper front. Recall that the spatial discretization of the MGI is based on a common central finite-volume scheme, which means that the present problem with $\lambda = 10^{-4}$ provides a rather difficult test for the moving-grid method.

Figure 3 shows the computed grid and salt concentration profiles at some values of time for $N_2 = N + 2 = 25$ and 50. Like in the previous example, we see that the grid movement accurately reflects the anticipated solution behaviour. For early times it is completely similar, while for later times the cluster around the steep salt front remains in position. We also see that $N_2 = 25$ now results in more overshoot, due to the fact that the dispersion

length is ten times smaller than in the previous example. However, $N_2 = 50$ again results in a very accurate solution and the profiles for $N_2 = 100$ (not shown here) do equal those for $N_2 = 50$ up to plotting accuracy.

Table 5 contains part of the integration history for $N_2 = 25$, 50 and 100, providing the same information as in the previous tables. With this table we wish to call attention for an inherent model difficulty stemming from the absolute value function in the dispersion-flux expression $\rho J = - \rho\lambda\, |q|\, \omega_x$. In the table this difficulty manifests itself in the large number of time steps and Jacobian updates used over the 'near steady-state interval' [200,500] for $N_2 = 100$ (recall that the steady state starts to settle at about t = 150). While the code easily detects the numerical steady state solution with 25 and 50 points, which can be concluded from the few number of steps needed to integrate from t = 200 to t = 500, this is clearly not the case with 100 points. In fact, with 100 points this 'near steady-state part' of the integration interval requires 1038 - 423 = 615 integration steps and 707 - 105 = 602 Jacobian updates, which is rather extreme. What has happened here is that the iterative Newton algorithm repeatedly has failed to converge, so that the strategy of the code keeps the temporal stepsize down and keeps asking for new Jacobians.

The cause for the Newton convergence failure lies with $|q|$ if $q \approx 0$. The following observations explain this. Due to the absolute value function, entries of the Jacobian matrix contain sign(q). Consequently, if $q \approx 0$, then during the Newton iteration approximate values for q readily change sign. Since the size of entries is large, as they contain terms $(\Delta x_i)^{-2}$ and Δx_i can be very small, it happens that during the iteration process entries frequently change their value from large positive to large negative, or vice versa. No doubt this severely hinders the convergence of the iterative Newton process and, as we have observed, often will lead to convergence failures and requests for a Jacobian update. This explains why the march to steady state in the case of 100 points is so troublesome. However, we stipulate that also with 25 and 50 points the march to steady state eventually becomes troublesome. It all depends on the size of the computed velocities q and is a matter of accuracy. With lesser points the computed velocities arrive in the troublesome regime for larger values of time when the system has become sufficiently stationary or, in other words, when the numerical velocities have become sufficiently small. Ironically, with 100 points the accuracy is sufficiently good to have the troublesome Newton convergence behaviour already for $200 < t < 500$.

Finally, we wish to stress once more that the troublesome march to steady state originates from the Jacobian matrix needed in the iterative solution process and not from the integration formula itself. In fact, we have also run the problem with $|q|$ replaced by $\sqrt{q^2 + \varepsilon}$ with $\varepsilon = 10^{-5}$, which completely remedies the situation and a normal march to steady state is observed with very large stepsizes towards the end value T, even up to $T = 10^{12}$. When the modeling does not allow this slight modification in the dispersion-flux expression, an alternative remedy is to change the expression for $|q|$ only in the entries of the Jacobian matrix, so as to avoid the sign changes. This involves only a little change of the Jacobian matrix and thus should not interfere significantly with the convergence behaviour of the iterative Newton method.

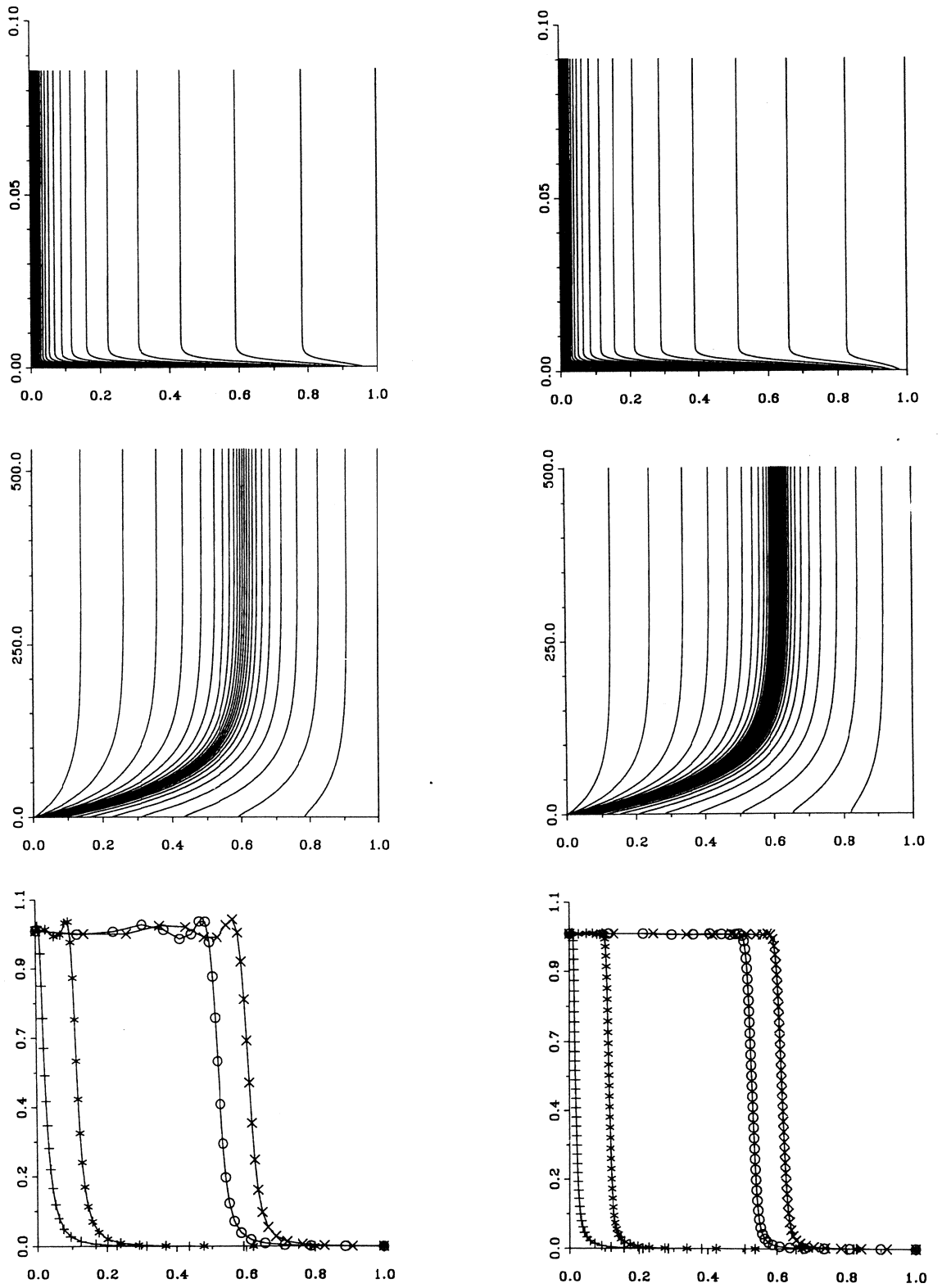

Figure 3. Example II, first-derivative monitor: Gridlines and salt concentration profiles at t = 1, 10, 200, 500. The left part of the figure corresponds with $N_2 = 25$ and the right part with $N_2 = 50$. Note the difference in scaling in each of the two gridline plots.

Table 5. Example II, first-derivative monitor: Integration histories.

		STEPS	JACS	RESIDS	NITER	ω_{max} - 1.0	CPU time (sec.)
$N_2 = 25$	t = 1	160	40	958	421	2.0_{10}-2	--
	t = 10	239	64	1509	654	3.0_{10}-2	--
	t = 100	335	92	2155	930	3.0_{10}-2	--
	t = 200	354	95	2244	980	4.0_{10}-2	--
	t = 500	373	101	2371	1072	4.0_{10}-2	95
$N_2 = 50$	t = 1	249	59	1473	685	4.0_{10}-3	--
	t = 10	300	72	1787	826	2.0_{10}-3	--
	t = 100	338	79	1983	931	2.0_{10}-3	--
	t = 200	355	81	2048	968	2.0_{10}-3	--
	t = 500	379	88	2204	1033	2.0_{10}-3	185
$N_2 = 100$	t = 1	312	85	2058	926	4.0_{10}-4	--
	t = 10	346	93	2253	1015	4.0_{10}-4	--
	t = 100	410	103	2538	1168	7.0_{10}-4	--
	t = 200	423	105	2592	1196	8.0_{10}-4	--
	t = 500	*1038*	*707*	*12700*	*3344*	9.0_{10}-4	*1753*

4.3. Example III

This example is derived from Example I by changing the salt concentration value $\omega(0,t) = 1$ to the step function

$$\omega(0,t) = \begin{cases} 1, & 0 < t \le 0.75, \\ 0, & 0.75 < t \le 5.0. \end{cases} \tag{4.2}$$

Thus for $0 < t \le 0.75$ the two solutions are equal and at $t = 0.75$ the step function generates a second front at $x = 0$ resulting in a block-form concentration profile. The block then travels to the right boundary and eventually the system runs into steady state with uniform zero salt concentration. For the moving-grid method this solution is more difficult to compute, since now two travelling fronts are present which appear and disappear at different values of t. Hence, instead of two times, four times the solution shape is drastically changed and the automatic grid movement and stepsize control should be able to cope with these drastic changes. For example, without neglecting the already existing first front, at $t = 0.75$ the method must rapidly cluster grid points at the left boundary and decrease the time step to timely see the onset of the second front. Therefore, for the same accuracy, roughly twice the number of grid points and time-stepping effort will be needed as for Example I.

We have solved the problem using N_2 = 25, 50, 100. As shown in Figure 4, 25 points is not enough, but with 50 points the solution is already fairly accurate. Comparison of the plots for 50 and 100 points reveals only very small differences at the top of the computed salt block profile and we may conclude that the results are very satisfactory. The gridline plots nicely reveal the onset of the second front where very small time steps have been taken, similar as at t = 0 (compare with Figure 1). Further, the arrival of the two fronts at the right boundary can also be clearly recovered from the plots, together with the change to the uniform steady state grid. Note that also here small time steps are needed to accurately simulate the rapid solution change.

The integration costs tabulated in Table 6 indeed show that the time-stepping effort is about twice as large as for Example I. As anticipated, comparison of Tables 2 and 6 reveals that it are mainly the drastic changes in the solution shape that determine the costs. Once the front is in existence, the time stepping is done very efficiently, as can be deduced from the number of Jacobian updates listed in Table 2 at t = 1.0 and 0.1. To gain further insight in the integration costs, we have also run Example III on a fixed grid with 100 points, similar as Example I. At t = 5, we counted STEPS = 380, JACS = 49, RESIDS = 1546, NITER = 1546 and CPU = 258. As to be expected, we see that also on a fixed grid the costs are larger (compare with last row of Table 4). However, less than twice as large, which we owe to the fact that on a fixed grid we only have to cope with the drastic changes in solution shape since the grid movement is absent here. While for 100 fixed points the accuracy is far from sufficient (see Figure 5), we estimate that with 400 to 500 fixed points the results will be close to that of 100 moving points, both in connection with accuracy and CPU time. Obviously, for a smaller value of the dispersion length λ such a comparison will readily favour the moving-grid approach.

Table 6. Example III, first-derivative monitor: Integration costs at t = 1.0, 2.0, 5.0.
(see Table 2 for corresponding data at t = 0.1 and 0.5)

		STEPS	JACS	RESIDS	NITER	CPU time (sec.)
N_2 = 25	t = 1.0	427	117	2747	1191	--
	t = 2.0	658	191	4456	1914	--
	t = 5.0	1035	288	6881	3056	273
N_2 = 50	t = 1.0	429	131	2978	1238	--
	t = 2.0	623	187	4357	1877	--
	t = 5.0	920	249	6035	2735	490
N_2 = 100	t = 1.0	554	159	3734	1624	--
	t = 2.0	819	262	6008	2511	--
	t = 5.0	1009	308	7259	3154	1146

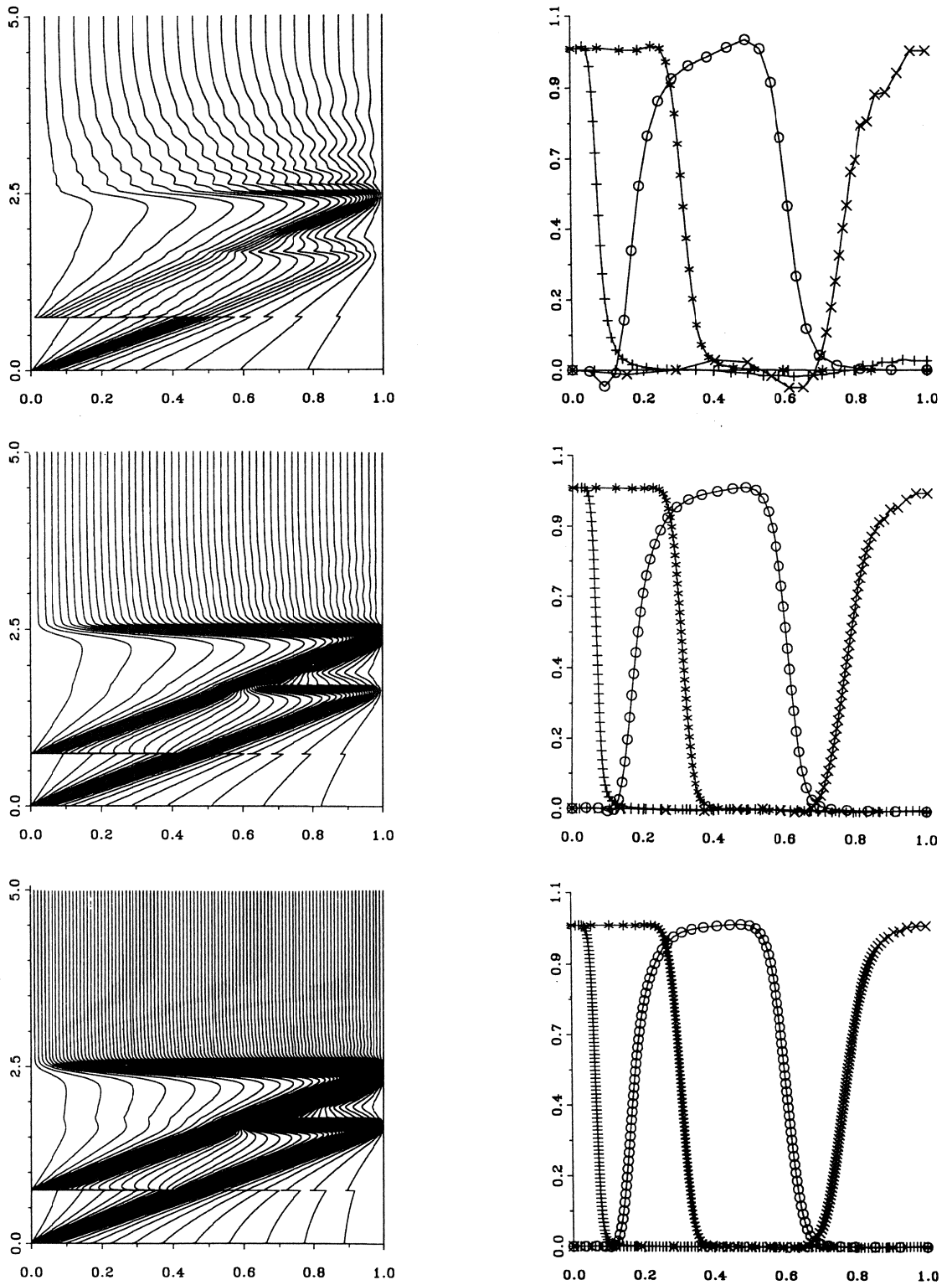

Figure 4. Example III, first-derivative monitor: Gridlines and salt concentration profiles at t = 0.1, 0.5, 1.0, 2.0, 5.0 for N_2 = 25, 50, 100.

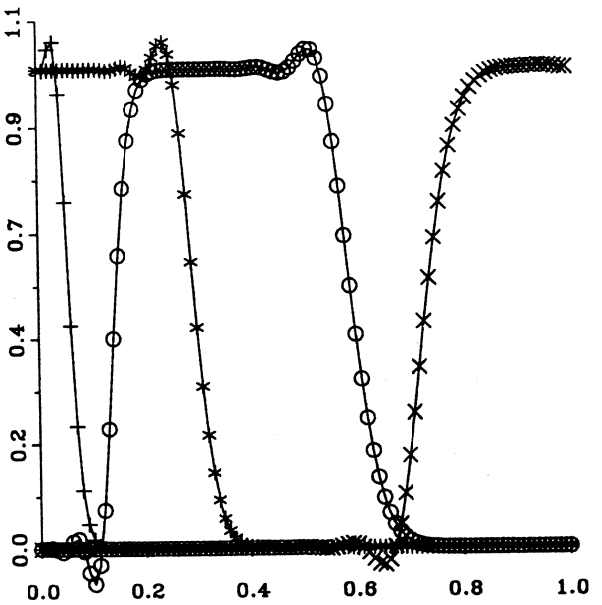

Figure 5. Example III, nonmoving grid: Salt concentration profiles at $t = 0.1, 0.5, 1.0, 2.0, 5.0$ for $N_2 = 100$.

5. CONCLUDING REMARKS

We have applied a moving-grid finite-difference method to a particular class of one-space dimensional fluid-flow/salt-transport problems with rapid spatial and temporal transitions in the salt concentration. The success of this method rests on two sorts of automatic grid-adaptation. The first adaptation is connected with the space grid and serves to cope with the rapid spatial transitions. These are dealt with by integrating on grids that spatially equidistribute a relevant measure of the error. The equidistribution is realized in a dynamic Lagrangian approach where the grid is adapted continuously in time. This feature is important since it makes it possible to accurately and efficiently follow steep travelling fronts. The second adaptation serves to cope with rapid temporal transitions and is just the use of variable stepsizes in the numerical integration. Variable stepsizes are a prerequisite when drastic solution changes have to be dealt with, like the onset of a steep front. The numerical integration has been performed with the LSODI based stiff ODE solver of the SPRINT package [1].

Our findings reported in Section 4 have convincingly shown that the method is very well suited to solve 1D brine transport models involving high concentration gradients. Because we have worked with an a priori chosen set of numerical control parameters, it is most likely that tuning of these parameters will further enhance the efficiency and accuracy for the specific model at hand. Since the method has been originally developed for general, one-space dimensional PDE systems [6,14], the method is also an excellent candidate for solving fluid-flow/solute-transport problems from other fields of application. In this connection it is worth to emphasize the user-friendly computational environment of the SPRINT package and the moving-grid interface MGI [3], which together provide a numerical software tool that requires a minimum of programming effort.

REFERENCES

[1] M. Berzins, P.M. Dew & R.M. Furzeland (1989): *Developing software for time-dependent problems using the method of lines.* Appl. Numer. Math. 5, 375 - 398.

[2] J.G. Blom & J.G. Verwer (1989): *On the use of the arclength and curvature monitor in a moving-grid method which is based on the method of lines.* Report NM-N8902, CWI, Amsterdam.

[3] J.G. Blom & P.A. Zegeling (1989): *A moving-grid interface for systems of one-dimensional time-dependent partial differential equations.* Report NM-R8904, CWI, Amsterdam.

[4] K.E. Brenan, S.L. Campbell & L.R. Petzold (1989): *Numerical solution of initial-value problems in differential algebraic equations.* North-Holland.

[5] K. Dekker & J.G. Verwer (1984): *Stability of Runge-Kutta methods for stiff nonlinear differential equations.* North-Holland.

[6] E.A. Dorfi & L. O'C. Drury (1987): *Simple adaptive grids for 1D initial-value problems.* J. Comput. Phys. 69, 175 - 195.

[7] J.C.H. van Eijkeren, P.A. Zegeling & S.M. Hassanizadeh (1991): *Practical use of SPRINT and a moving-grid interface for a class of 1D nonlinear transport problems.* Report nr. 959101001, RIVM, Bilthoven, The Netherlands.

[8] R.M. Furzeland, J.G. Verwer & P.A. Zegeling (1990): *A numerical study of three moving-grid methods for one-dimensional partial differential equations which are based on the method of lines.* J. Comput. Phys. 89, 349 - 388.

[9] S.M. Hassanizadeh & T. Leijnse (1988): *On the modeling of brine transport in porous media.* Water Resources Research 24, 321 - 330.

[10] S.M. Hassanizadeh (1990): *Experimental study of coupled flow and mass transport: a model validation exercise.* In: *Calibration and reliability in groundwater modeling,* ed. K. Kovar, IAHS Publication No. 195, Wallingford, Oxfordshire, U.K.

[11] J.M. Sanz-Serna & J.G. Verwer (1989): *Stability and convergence at the PDE / stiff ODE interface.* Appl. Numer. Math. 5, 117 - 132.

[12] R.D. Skeel & M. Berzins (1990): *A method for the spatial discretization of parabolic equations in one space variable.* SIAM J. Sci. Stat. Comput. 11, 1 - 32.

[13] J.G. Verwer, J.G. Blom & J.M. Sanz-Serna (1989): *An adaptive moving-grid method for one-dimensional systems of partial differential equations.* J. Comput. Phys. 82, 454 - 486.

[14] J.G. Verwer, J.G. Blom, R.M. Furzeland & P.A. Zegeling (1989): *A moving-grid method for one-dimensional PDEs based on the method of lines.* In: *Adaptive methods for partial differential equations,* eds. J.E. Flaherty, P.J. Paslow, M.S. Shephard & J.D. Vasilakis, SIAM publications, Philadelphia, pp. 160 - 175.