COMPUTING BRINE TRANSPORT IN POROUS MEDIA WITH AN ADAPTIVE-GRID METHOD*

R. A. TROMPERT, J. G. VERWER AND J. G. BLOM

Centre for Mathematics and Computer Science (CWI), P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

SUMMARY

An adaptive-grid finite-difference method is applied to a model for non-isothermal, coupled flow and transport of brine in porous media. In the vicinity of rock salt formations the salt concentration in the fluid becomes large, giving rise to disparate scales in the salt concentration profiles. A typical situation one encounters is that of a sharp freshwater–saltwater interface that moves in time. In such situations adaptive-grid methods are more effective than standard fixed-grid methods, since they refine the space grid locally and, hence, provide for substantial reduction in the number of grid points, memory use and CPU time. The adaptive-grid method of this paper is a static, local uniform grid refinement method. Its main feature is that it integrates on nested sequences of locally uniformly refined Cartesian space grids, which are automatically adjusted in time to follow rapid spatial transitions. Variable time steps are used to cope with rapid temporal transitions, including a fast march to possible steady-state solutions. For time stepping, the implicit, second-order BDF scheme is used. Two specific example problems are numerically illustrated. The main physical properties involved here are advection and dispersion and in case of dominant advection sharp freshwater–saltwater interfaces arise.

KEY WORDS Finite differences Adaptive-grid methods Local uniform grid refinement Method of lines Fluid flow/solute transport in porous media Brine transport

1. INTRODUCTION

We discuss the application of an adaptive-grid finite-difference method to a non-isothermal model for coupled flow and transport of brine in porous media. This model originates from a safety assessment study on disposal of radioactive wastes in rock salt formations, like salt domes. A characteristic of groundwater flow near salt domes is a large variation in the salt concentration in the flowing fluid. Recent theoretical and experimental hydrological studies indicate that in these situations the involved basic equations of flow and transport need to be re-examined.[4,5] These basic equations have been used traditionally in models where the salt concentration is low and more or less constant, e.g. that of seawater. However, a typical modelling scenario for flows near salt domes involves disparate scales in the salt concentration profile in the flow domain, as occurring, for example, with a moving freshwater–saltwater interface. The study of such flows requires a significant effort in numerical modelling and the work reported in this paper has its origin in such numerical modelling studies. The physical

* This work was carried out as part of contract research by order of the Laboratory for Soil and Groundwater Research of RIVM — the Dutch National Institute of Public Health and Environmental Protection – in connection with Project 'Locatiespecifieke Modelvalidatie' 725205. Financial support for this project was provided by the Dutch Ministry of Economic Affairs.

0271–2091/93/010043–21$15.50

Received March 1992
Revised July 1992

model we use is similar to that in Reference 4, except that we employ the classical form of Darcy's law for the momentum balance equation for the fluid (brine) and Fick's law for the dispersion of the salt. On the other hand, while Reference 4 focuses on the idealized case of one spatial dimension (see also Reference 15), here we consider the two-dimensional case and also take temperature effects into account.

For the computation of steep moving salt fronts, standard numerical methods may not be feasible because they necessitate a very fine grid covering the entire spatial domain for all values of time. In such cases, adaptive-grid methods provide a remedy. An adaptive-grid method refines the space grid only locally, hence striving for a substantial reduction in the number of grid points, memory use and CPU time. The adaptive-grid method applied in this paper is the local uniform grid refinement method discussed in our earlier papers.[11–14] This method has been developed for the numerical solution of a wide class of time-dependent partial differential equations (PDEs) having solutions with rapid transitions like steep moving fronts, emerging layers, etc. This class includes the current brine transport model and variants thereof. Our computer implementation has been written for two spatial dimensions. This, however, is no essential restriction. The adaptive-grid technique used can be extended to three spatial dimensions.

In Section 2 we describe the brine transport model we have used and give the complete data set for two specific example problems. These two examples are connected with Intraval test case 13.[7] The main physical properties involved here are advection and dispersion and, in the case of dominant advection, sharp freshwater–saltwater interfaces arise. The section is self-contained so that interested readers can also use these two examples as test problems. In Section 3 we outline the adaptive-grid method for a general class of time-dependent PDEs. The main feature of the method is that it integrates on nested sequences of locally uniformly refined Cartesian space grids, which are automatically adjusted in time to follow rapid spatial transitions. For time stepping, the implicit second-order backward differentiation formula (BDF) is used. Variable time steps are used to cope with rapid temporal transitions, including a fast march to possible steady-state solutions. Our use of the implicit BDF method means that we treat the brine transport problem fully coupled. The arising coupled systems of non-linear algebraic equations are solved with the modified Newton method in combination with the iterative, preconditioned conjugate gradient squared method (CGS)[10] for the arising linear systems problems. The actual numerical illustrations are discussed in Section 4. We finish the paper in Section 5 with some final remarks.

2. THE (2D) FLUID FLOW/SALT TRANSPORT PROBLEM

2.1. The model

In the modelling 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.[4] The set for the fluid constitutes the fundamental balance of mass of the fluid supplemented with Darcy's law. Similarly, for each solute the associated set constitutes the balance of mass supplemented with conservation of momentum through a Fickian law. In our case, the fluid is water impregnated with salt (brine) and there is only one solute, the salt. 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. We will take into account the temperature, but deformation effects and porosity changes are omitted. 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. Sources and sinks are also omitted here but can easily be added.

We, thus, consider a particular model for non-isothermal, single-phase, two-component saturated flow, which is constituted by a system of three PDEs basic to groundwater flow: a continuity equation (balance equation for the brine mass $n\rho$), a transport equation (balance equation for the salt mass concentration $n\rho\omega$), and a temperature equation (balance equation for energy $c^{\mathrm{m}}\rho^{\mathrm{m}}T$). This model has been borrowed from Reference 4 and from the private note[6] and is briefly described below. Since our purpose is to apply a numerical solution method, we have tried to omit, as much as possible, technical details which are of no direct interest for our purpose. Readers interested in such details and other background material are referred to Reference 1 (see, e.g. Chapter 7). This textbook discussed at length the derivation of closely related models for groundwater flow.

The continuity equation supplemented with the Darcy law reads

$$\frac{\partial}{\partial t}(n\rho)+\operatorname{div}(\rho\mathbf{q})=0, \quad \mathbf{q}=-\frac{\mathbf{K}}{\mu}(\operatorname{grad} p-\rho\mathbf{g}), \tag{1}$$

where n is the porosity parameter of the porous medium, ρ the mass density of the fluid, $\mathbf{q}$ the Darcy velocity of the fluid, $\mathbf{K}$ the permeability tensor of the porous medium, μ the dynamic viscosity of the fluid, p the hydrodynamic pressure and $\mathbf{g}$ the acceleration of gravity vector. Noteworthy is the fact that in low salt concentration cases one often works with the so-called Boussinesq approximation, which assumes that variations in the liquid's density are negligible in the balance equation (1). This equation is then replaced by the standard continuity equation

$$\operatorname{div}\mathbf{q}=0. \tag{2}$$

Hence, the flow is then supposed to be divergence-free and in numerical methods (2) is commonly replaced by a standard second-order elliptic equation (the density variation remains in the gravity term in Darcy's law). Throughout, we use (1) as our main interest is in the high salt concentration case, where the Boussinesq approximation should not be used.

The transport equation supplemented with the Fickian law reads

$$\frac{\partial}{\partial t}(n\rho\omega)+\operatorname{div}(\omega\rho\mathbf{q}+\rho\mathbf{J}^{\omega})=0, \quad \mathbf{J}^{\omega}=-n\mathbf{D}\operatorname{grad}\omega, \tag{3}$$

where ω is the salt mass fraction (mass concentration of the salt in the brine/mass density ρ of the brine), $\mathbf{J}^{\omega}$ is the salt dispersion flux vector and $\mathbf{D}$ is the dispersion tensor of the solute salt defined as

$$n\mathbf{D}=(nd_{\mathrm{m}}+\alpha_{\mathrm{T}}|\mathbf{q}|)I+\frac{\alpha_{\mathrm{L}}-\alpha_{\mathrm{T}}}{|\mathbf{q}|}\mathbf{q}\mathbf{q}^{\mathrm{T}}, \quad |\mathbf{q}|=(\mathbf{q}^{\mathrm{T}}\mathbf{q})^{1/2}. \tag{4}$$

The coefficients d_{m}, α_{T}, α_{L} stem from molecular diffusion and transversal and longitudinal dispersion, respectively. Hence, $\omega\rho\mathbf{q}$ is the salt mass flux due to advection and $\rho\mathbf{J}^{\omega}$ the salt mass flux due to molecular diffusion and mechanical dispersion (dispersion caused by flow). In real flow situations the molecular diffusion is usually significantly smaller than the mechanical dispersion.

The temperature equation is given by

$$\frac{\partial(c^{\mathrm{m}}\rho^{\mathrm{m}}T)}{\partial t}+\operatorname{div}(cT\rho\mathbf{q}+\mathbf{J}^{\mathrm{T}})=0, \quad \mathbf{J}^{\mathrm{T}}=-\mathbf{H}\operatorname{grad}T, \tag{5}$$

where T is temperature, c specific heat capacity of the porous medium, $\mathbf{J}^{\mathrm{T}}$ the heat flux vector and

$\mathbf{H}$ the heat conductivity tensor of the porous medium defined as

$$\mathbf{H}=(\kappa+\lambda_{\mathrm{T}}|\mathbf{q}|)I+\frac{\lambda_{\mathrm{L}}-\lambda_{\mathrm{T}}}{|\mathbf{q}|}\mathbf{q}\,\mathbf{q}^{\mathrm{T}}. \tag{6}$$

The first term in (5) represents the variation of heat mass or energy $c^{\mathrm{m}}\rho^{\mathrm{m}}T$ in the medium, the second term the advective flux of heat mass by the fluid, and the third the conductive flux of heat by both the porous medium and the fluid. The density expression $c^{\mathrm{m}}\rho^{\mathrm{m}}$ in the first term of (5) satisfies the relation

$$c^{\mathrm{m}}\rho^{\mathrm{m}}=nc\rho+(1-n)\rho^{\mathrm{s}}c^{\mathrm{s}}, \tag{7}$$

where the second term refers to the solid phase. Material properties allow this term to be taken constant. In (6) the parameter κ is the coefficient of heat conductivity and λ_{T} and λ_{L} are, respectively, the transversal and longitudinal heat conductivity coefficient. In most applications $\kappa \gg \lambda_{\mathrm{T}}$, λ_{L} such that the diagonal matrix κI dominates the tensor $\mathbf{H}$.

The density ρ and viscosity μ are supposed to obey the state equations

$$\rho=\rho_0 \exp[\alpha(T-T_0)+\beta(p-p_0)+\gamma\omega], \tag{8}$$

$$\mu=\mu_0(T)\,m(\omega), \quad m(\omega)=1+1{\cdot}85\omega-4{\cdot}0\omega^2, \tag{9}$$

where ρ_0 is the reference density of fresh water, p_0 a reference pressure, T_0 a reference temperature, α a temperature coefficient, β a compressibility coefficient, γ a salt coefficient and $\mu_0(T)$ a possibly temperature-dependent reference viscosity. In our examples the porosity n is taken constant and the permeability tensor $\mathbf{K}$ is chosen to be of the diagonal form $\mathbf{K}=\operatorname{diag} K$, where in the first example K is a constant too and in the second one K has a jump.

The system of three balance equations (1), (3) and (5) can be rewritten as a system having pressure p, salt mass fraction ω and temperature T as dependent variables, in the numerical experiments, we have worked with this particular system. An elementary calculation yields

$$n\rho\left(\beta\frac{\partial p}{\partial t}+\gamma\frac{\partial \omega}{\partial t}+\alpha\frac{\partial T}{\partial t}\right)+\operatorname{div}(\rho\mathbf{q})=0, \tag{10}$$

$$n\rho\frac{\partial \omega}{\partial t}+\rho\mathbf{q}\cdot\operatorname{grad}\omega+\operatorname{div}(\rho\mathbf{J}^{\omega})=0, \tag{11}$$

$$c^{\mathrm{m}}\rho^{\mathrm{m}}\frac{\partial T}{\partial t}+\rho c\mathbf{q}\cdot\operatorname{grad}T+\operatorname{div}\mathbf{J}^{\mathrm{T}}=0, \tag{12}$$

where (10) is the brine flow equation, (11) the salt transport equation and (12) the temperature equation. Note that in the derivation of (12) the assumption that the expression $(1-n)\rho^{\mathrm{s}}c^{\mathrm{s}}$ contained in (7) is constant is used. Equations (10)–(12) form a system of coupled non-linear parabolic PDEs. Equation (10) generalizes the standard continuity equation (2) used in the Boussinesq approximation. A special feature of the model is that the compressibility coefficient β is very small or even zero. If $\beta=0$, then the 3×3 matrix multiplying the temporal derivative vector (p_t, ω_t, T_t) is singular and (10) is effectively replaced by an equation without temporal derivatives, like in the Boussinesq approximation. We stipulate that for the implicit BDF integration method there is no need to distinguish between $\beta=0$ and $\beta\neq0$.

Equation (11) is of the advection–dispersion type and, as usual, numerically difficult to solve if it is advection-dominated. This depends on the relative size of the velocity vector $\mathbf{q}$ compared to that of the symmetric matrix $n\mathbf{D}$. Let L_x and L_y be proper length scales in the x- and y-direction, respectively. Then, if we put $\alpha_T = \alpha_L$, we recover the one-dimensional, scaled Peclet numbers

$$Pe_x = \frac{L_x q_1}{n d_m + \alpha_T |\mathbf{q}|}, \qquad Pe_y = \frac{L_y q_2}{n d_m + \alpha_T |\mathbf{q}|}. \tag{13}$$

If $\alpha_T \neq \alpha_L$, then the derivation becomes more intricate as the cross derivatives as well as the size of the two velocity components play a role. However, as a rule of thumb, one may still use the Peclet numbers (13) as a first measure of the dominance of advection. In practice, one encounters values for Pe_x and Pe_y in the range 10^2–10^4, say.

For the temperature equation (12), which is similar to (11) and of the advection–heat-conduction type, the situation is somewhat different, In realistic brine transport applications $\mathbf{H}$ is largely determined by the diagonal contribution κI, because normally $\kappa \gg \lambda_T, \lambda_L$. Hence, here the ratio between advection and heat conduction is largely determined by the quotient of κ and the components of $\rho c \mathbf{q}$. Specifically, denoting $\mathbf{q} = [q_1, q_2]^T$, the one-dimensional scaled Peclet numbers are now $Pe_x = L_x \rho c q_1 / \kappa$ and $Pe_y = L_x \rho c q_2 / \kappa$. These numbers measure the dominance of advection, provided κI dominates $\mathbf{H}$. As a rule, the dominance of advection in (12) will be less than in (11), and (12) will more or less behave like a standard parabolic equation with a modest advection term.

2.2. *Data for example problem I*

Our examples are connected with Intraval test case 13 described in Reference 7. This laboratory experiment deals with the displacement of fresh water by brine in a thin vertical column filled with a porous medium. Salt water of a high concentration is injected through gates at the bottom of the column giving rise to a freshwater–saltwater front moving in all directions into the column. In Reference 7 the experiment was carried out under isothermal conditions. We assume non-isothermal conditions and suppose that warm brine is injected. Because the column is very thin, the flow can be considered as being two-dimensional in space. In our numerical experiment (discussed in Section 4) it is supposed that two gates are used for saltwater injection so that initially two disjunct saltwater–freshwater fronts exists which later interact and merge into one front. Due to dispersion, the fronts smooth out with time in all directions. For t sufficiently large the fronts disappear completely which means that the whole medium is filled with the high salt concentration fluid.

In Example I the model is considered on the time–space domain $[0 < t \leqslant t_{\text{end}}] \times \Omega$, where the flow domain Ω representing the vertical column is the unit square $\Omega = \{(x, y): 0 < x, y < 1\}$. Below we will write $p(x, y, t)$, etc., and since Ω represents a vertical cross-section, the independent variable y stands for a vertical variable with unit vector pointing upward. Hence, the acceleration of gravity vector takes the form $\mathbf{g} = (0, -g)$, where g is the acceleration due to gravity. The initial values at $t = 0$ at Ω are taken as

$$p(x, y, 0) = p_0 + (1 - y)\rho_0 g, \quad \omega(x, y, 0) = 0, \quad T(x, y, 0) = T_0, \tag{14}$$

which correspond, respectively, to hydrostatic pressure, fresh water and a non-heated medium. For $0 < t \leqslant t_{\text{end}}$ the following boundary conditions are imposed:

$x = 0, 1$ and $0 < y < 1$: $\qquad q_1 = 0, \ \dfrac{\partial \omega}{\partial x} = 0, \ \dfrac{\partial T}{\partial x} = 0,$

$y = 1$ and $0 < x < 1$: $\qquad p = p_0, \ \dfrac{\partial \omega}{\partial y} = 0, \ \dfrac{\partial T}{\partial y} = 0,$

$$y=0 \quad \text{and} \quad \tfrac{1}{11}\leqslant x\leqslant \tfrac{2}{11},\ \tfrac{9}{11}\leqslant x\leqslant \tfrac{10}{11}: \qquad (\omega\rho\mathbf{q}+\rho\mathbf{J}^{\omega})_2=\omega_0\rho(\omega_0, T_c, p)q_c,\ q_2=q_c,$$

$$(c\rho T\mathbf{q}+\mathbf{J}^{T})_2=c\,\rho(\omega_0, T_c, p)\ T_c q_c, \tag{15}$$

$$y=0 \quad \text{and} \quad 0<x<\tfrac{1}{11},\ \tfrac{2}{11}<x<\tfrac{9}{11},\ \tfrac{10}{11}<x<1: \quad \frac{\partial\omega}{\partial y}=0, \quad q_2=0, \quad \frac{\partial T}{\partial y}=0.$$

The third line is connected with the two gates where the warm brine is injected with a prescribed flux and velocity. The other conditions are self-evident. All remaining problem data are contained in Table I.

2.3. Data for example problem II

The numerical method can handle jumps in the permeability by imposing continuity of fluxes over the permeability interfaces into the finite-difference spatial discretization scheme. Inside the regions, where the permeability is continuous, the PDEs are then treated in the standard way. This second example, which is a modification of Example I, serves to illustrate this. However, we here consider the extreme case of total impermeability for part of the flow domain. The impermeable region is $\{(x, y);\ 0\leqslant x\leqslant 0{\cdot}5,\ 0{\cdot}4\leqslant y\leqslant 0{\cdot}6\}$. We also consider the flow now to be isothermal and incompressible ($\beta=0$).

Because inside the region of impermeability both (10) and (11) reduce to $\partial\omega/\partial t=0$, we replace them by

$$p(x, y, t)=p_0+(1-y)\rho_0 g, \qquad \omega(x, y, t)=0, \tag{16}$$

and solve outside the region the original continuity and transport equation, using all further problem data of Example I. The first equation of (16) means hydrostatic pressure. This is

Table I. Data for example Problem I

Porosity	n	0·4
Permeability	K (m^2)	10^{-10}
Acceleration due to gravity	g (m s^{-2})	9·81
Molecular diffusion	d_m (m^2 s^{-1})	0
Transversal dispersivity	α_T (m)	0·002
Longitudinal dispersivity	α_L (m)	0·01
Heat capacity (c_f)	c (J kg^{-1} K^{-1})	4182
Heat conductivity	κ (J s^{-1} m^{-1} K^{-1})	4·0
Transversal heat conductivity coefficient	λ_T (J m^{-2} K^{-1})	10^{-3}
Longitudinal heat conductivity coefficient	λ_L (J m^{-2} K^{-1})	10^{-2}
Solid heat capacity	c^s (J kg^{-1} K^{-1})	840
Solid density	ρ^s (kg m^{-3})	2500
Freshwater density	ρ_0 (kg m^{-3})	1000
Reference temperature	T_0 (K)	290
Reference pressure	p_0 (kg m^{-1} s^{-2})	10^5
Temperature coefficient	α (K^{-1})	$-3{\cdot}0\times10^{-4}$
Compressibility coefficient	β (m s^2 kg^{-1})	$4{\cdot}45\times10^{-10}$
Salt coefficient	γ	ln 1·2
Reference viscosity	μ_0 (kg m^{-1} s^{-1})	10^{-3}
Reference salt mass fraction	ω_0	0·25
Vertical inlet velocity	q_c (m s^{-1})	10^{-4}
Inlet temperature	T_c(K)	292
End value of time	t_{end} (s)	10^6

a natural condition in view of the fact that we impose total impermeability. The second equation is the now-trivial transport equation. This equation is also natural because inside the region of impermeability the salt concentration is to remain zero. The requirement of continuity of fluxes thus leads to the additional conditions

$$\begin{aligned} &0 \leqslant x \leqslant 0{\cdot}5 \quad \text{and} \quad y = 0{\cdot}4,\ 0{\cdot}6: && \frac{\partial p}{\partial y} = -\rho_0 g, \quad \frac{\partial \omega}{\partial y} = 0, \\ &x = 0{\cdot}5 \quad \text{and} \quad 0{\cdot}4 < y < 0{\cdot}6: && \frac{\partial p}{\partial x} = 0, \quad \frac{\partial \omega}{\partial x} = 0. \end{aligned} \tag{17}$$

Of course, these flux conditions can also be interpreted as new boundary conditions because the modification of Example I amounts to a modification of the flow domain.

3. THE ADAPTIVE-GRID METHOD

3.1. Static regridding and local uniform grid refinement

For time-dependent PDEs two sorts of adaptive-grid methods are distinguished, dynamic and static methods. While dynamic (in time) methods adapt the grid in a continuous-time manner, like classical Lagrangian methods, static (in time) methods adapt the grid only at discrete times. Our method is of the static type and based on the technique called Local Uniform Grid Refinement (LUGR). Here we present an outline of this technique.

The LUGR idea is to cover $\Omega \cup \partial\Omega$ with a sequence of locally nested, uniformly refined subgrids which are adapted at discrete times to follow rapid spatial transitions. We adapt at each time level and each (full) time step then consists of repeated integrations on the nested finer-and-finer local subgrids. Within each full time step the integration starts at the coarsest base grid and each of the single integrations spans the same time step. Loosely speaking, on each local subgrid we repeatedly solve a new initial boundary value problem. Required initial values are defined by interpolation from the next coarser subgrid or taken from a possibly existing subgrid from the previous time step interval. Boundary values required at internal boundaries are also interpolated from the next coarser subgrid. The generation of the nested subgrids is continued up to a level considered fine enough for resolving the fine-scale structure at hand. Having completed the integration on the finest level, the process is repeated for the next time step. We then use the most accurate solution available and grid points already existing at a certain level of refinement are used for step continuation. In conclusion, assuming a one-stage implicit integration scheme like the backward differentiation method (BDF),[8] each full time step consists of the following operations:

1. Integrate on the coarse base grid.
2. If the desired accuracy in space or the maximum number of levels is reached, go to 7.
3. Determine new finer uniform subgrid at forward time.
4. Interpolate internal boundary values at forward time.
5. Provide required solution values at backward time levels.
6. Integrate on gridlevel using the same steplength and go to 2.
7. Inject fine grid values in coinciding coarse grid points.

An important point to notice is that we repeatedly use all subgrids, in the order from coarse to fine. This way we generate the required boundary conditions at the internal boundaries and keep the local subgrids uniform. This approach necessarily leads to some overhead costs. On the other hand, the work load on the coarser grids will normally be small and we consider the use of uniform grids attractive. Uniform grids allow an efficient use of vector-based algorithms and finite-difference approximations on uniform grids are more accurate and faster to compute than on non-uniform grids. In this respect, the current approach is to be contrasted with pointwise refinement leading to truly non-uniform grids. Finally, the actual refinement is cellular and carried out by bisection of the sides of grid cells. We refer to the figures presented later in the paper for an illustration of the LUGR grid structure.

3.2. The mathematical formulation

We will now give a mathematical formulation of the LUGR method when using the implicit BDF method for time stepping. Following the method of lines approach, the LUGR method can then be formulated in a mathematically comprehensive way for a wide class of PDEs. To this end, we consider the abstract Cauchy problem for the following general system of equations

$$\begin{aligned} G(x, y, t, u, u_t, u_x, u_y, u_{xx}, u_{xy}, u_{yy}) &= 0 \quad (x, y) \in \Omega, \qquad t > t_0, \\ H(x, y, t, u, u_t, u_x, u_y) &= 0, \quad (x, y) \in \partial\Omega, \qquad t > t_0, \\ u(x, y, t_0) &= u^0(x, y), \quad (x, y) \in \Omega \cup \partial\Omega, \ t = t_0, \end{aligned} \tag{18}$$

while the two-dimensional space domain Ω is here supposed to be of rectangular form. Needless to say that the brine transport problems of the previous section fit into the general format (18). We suppose the PDE problem to be of second-order in space and, for at least one component, first-order in time. Trivially, we also assume that the specific problem at hand that fits into format (18) is well-posed and possesses a unique solution $u(x, y, t)$. This vector-valued solution $u(x, y, t)$ is also supposed to be as often differentiable on $(\Omega \cup \partial\Omega) \times \{t > t_0\}$ as the numerical method requires. Recall, however, that we aim at non-smooth solutions, like steep travelling fronts. Thus, here, non-smoothness means the occurrence of rapid transitions in the space–time domain, with a sufficient degree of differentiability.

LUGR methods use local subgrids of varying size in time and, thus, generate approximation vectors of a varying dimension. This varying dimension complicates the formulation. We circumvent this problem by expanding the fine local subgrids over the entire domain $\Omega \cup \partial\Omega$, so that integration takes place over the fine local subgrids and interpolation over their entire complement in $\Omega \cup \partial\Omega$. However, interpolation over the entire complement is redundant in the actual application and, thus, takes place only in the formulation. We return back to this point later.

Let Ω_k, $1 \leqslant k \leqslant r$, be uniform space grids with the integer r denoting the number of refinement levels. For simplicity, we suppose here that r is fixed. Each grid covers the whole of $\Omega \cup \partial\Omega$ and Ω_{k+1} is obtained from Ω_k by cellular refinement. With (18) we now associate the differential-algebraic equation (DAE) system

$$F_k\left(t, U_k(t), \frac{\mathrm{d}}{\mathrm{d}t} U_k(t)\right) = 0, \qquad t > t_0, \quad U_k(t_0) = U_k^0, \tag{19}$$

obtained by spatial discretization on $\Omega_k \cup \partial\Omega_k$. We employ second-order finite differences, which is easy due to the uniformity of the grids. At interior points, the standard central scheme is used and on boundaries a combination of this central scheme with a one-sided scheme. Hence, U_k and F_k are grid functions on Ω_k and it is assumed that solution components existing at $\partial\Omega$ are contained in U_k. This means that the semi-discrete boundary conditions are treated as algebraic equations. Consequently, in virtually any application equation, (19) is a DAE system, even if $\partial u/\partial t$ in (18) can be explicitly specified.

We next introduce some more notation. Let d_k be the length of U_k and let S_k with $\dim S_k = d_k$ be the vector space of grid functions on Ω_k. The adaptive-grid method is formulated as an approximation method in the spaces $\{S_k\}$ where $U_k^n \in S_k$ denotes the approximation to the natural restriction of $u(x, y, t)$ on $\Omega \cup \partial\Omega$ to Ω_k at time $t = t_n$. We will employ the following four matrix operators:

1. the unit matrix $I_k : S_k \to S_k$,
2. a diagonal matrix $D_k^n : S_k \to S_k$ with entries $(D_k^n)_{ii}$ either unity or zero; $D_1^n = I_1$,
3. the natural restriction operator $R_{rk} : S_r \to S_k$ from Ω_r to Ω_k,
4. an interpolation operator $P_{k-1k} : S_{k-1} \to S_k$ from Ω_{k-1} to Ω_k.

For DAE systems like (19) we denote the well-known implicit s-step BDF integration method by

$$F\left(t_n, U^n, \frac{U^n - V^{n-1}}{\theta_s \Delta t}\right) = 0, \quad V^{n-1} = a_1 U^{n-1} + \cdots + a_s U^{n-s}, \tag{20}$$

where $\Delta t = t_n - t_{n-1}$ and V^{n-1} is the history vector çollecting values computed at backward time points. In this formulation the step size Δt may be considered variable. If $\Delta t = \Delta t_n$ is variable, then the integration coefficients a_j are dependent on the previous step sizes $\Delta t_{n-1}, \ldots, \Delta t_{n-s+1}$.[8]

The full LUGR time step from t_{n-1} to t_n, which consists of r consecutive integration/interpolation steps on the grids $\Omega_1, \Omega_2, \ldots, \Omega_r$, is now shortly formulated as

$$(I_k - D_k^n) U_k^n = (I_k - D_k^n) P_{k-1k} U_{k-1}^n, \tag{21a}$$

$$D_k^n F_k\left(t_n, U_k^n, (U_k^n - V_k^{n-1}) \frac{1}{\theta_s \Delta t}\right) = 0, \qquad k = 1, \ldots, r, \tag{21b}$$

where

$$V_k^{n-1} = R_{rk}[a_1 U_r^{n-1} + \ldots + a_s U_r^{n-s}], \tag{21c}$$

and the matrix D_k^n is supposed to be known prior to the computation at grid Ω_k. These diagonal matrices define the local subgrids upon which the actual numerical integration step is carried out. Specifically, if at a certain node integration is to take place, then, by definition, $(D_k^n)_{ii} = 1$, while at nodes where interpolation is carried out, $(D_k^n)_{ii} = 0$. The actual definition of D_k^n, and, hence, the actual selection of integration and interpolation nodes, is made by the refinement strategy, which is discussed later. The nesting property of the integration domains is also induced by this strategy and cannot be recovered from the above formulation, as it is hidden in the actual definition of D_k^n.

Formula (21a) represents the interpolation step. Note that for $k = 1$ formula (21a) is auxiliary since $D_1^n = I_1$ (also U_{k-1}^n does not exist for $k = 1$). The choice of interpolant is, in principle, still free, but we mostly work with the fourth-order Lagrangian interpolant. Formula (21b) represents the implicit BDF integration step over the local subgrid in use, which is carried out after the interpolation step (21a). The history vector formula (21c) contains the restriction operator R_{rk}, showing that at each grid level the finest grid solutions from past time levels are used for step continuation. Observe that (21b) is coupled to (21a), since the evaluation in (21b) calls for solution components of U_k^n living at grid interfaces (internal boundaries) through the coupling in the finite-difference grid. These grid interface components are defined by the interpolation step (21a).

Obviously, (21) contains redundant operation because it describes a computation in the (grid expanded) spaces S_k, In paractice we of course only execute (21b) at nodes for which the associated entry of D_k^n equals one. Furthermore, we apply restricted interpolation, which means interpolation only at nodes where needed, rather than over the whole of the grid Ω_{k-1}, as suggested by (21a). We stress that this restricted interpolation does not interfere with the formulation, due to the implicit assumption that the local subgrids are nested and, hence, that interpolation nodes are always located in the previous subgrid. Consequently, (21) provides an exact description of the computed approximations in reality, but it also defines redundant values as a result of formulating in the (grid expanded) spaces S_k. Trivially, in the actual application we omit the redundant operations to reduce the overhead costs.

3.3. The refinement strategy

A strategy should fulfil two basic accuracy requirements. It should induce a sufficient local refinement in regions where the spatial errors are larger than elsewhere, and it should involve automatic control of the inevitable interpolation errors. This second requirement is often neglected, but may be of significant importance. The reason is that if we regrid at each time step, we interpolate at each time step. Interpolation errors then can accumulate linearly with the time steps, so that reducing Δt may eventually result in error growth, rather than in error decay. Although less, this threat remains if we would not regrid at every time step, but per certain number >1 of such steps.

In Reference 12, where implicit Euler is considered ($s=1$ in (20)) for the explicit ODE case $\mathrm{d}U_t/\mathrm{d}t=F(t, U)$, we have developed a strategy which meets both requirements. There we demand that the refinement is such that the spatial accuracy on the composite final grid is comparable to the spatial accuracy on Ω_r if integration would take place on the whole of Ω_r without any adaptation at all. As far as accuracy is concerned, this is the maximum we can ask for. In addition, this way we force the interpolation error to remain negligible when compared with the common spatial error on Ω_r. This means that the accumulation of the interpolation errors is automatically controlled. The refinement analysis of Reference 12 can be extended straightforwardly to the s-step BDF method for the explicit ODE case. However, for genuine DAE problems like (19), this extension is not straightforward (the spatialy discretized brine transport problem with zero compressibility coefficient β is a genuine DAE problem). We will, therefore, report the refinement analysis for this more difficult case in a sequel to this paper and use instead here the more simple strategy developed in Reference 11. This refinement strategy is heuristic in the sense that the two accuracy requirements above are not really guaranteed. However, according to our experience, it will work very satisfactorily in most practical cases.

The strategy from Reference 11 decides which grid cells from the current integration domain will be refined, as well as the number of levels. Hence, the number of grid levels r may now vary from time step to time step. The strategy is governed by a user-specified tolerance number TOLS and by the curvature expression

$$\mathrm{ESTS}=(\Delta x)^2\,|u_{xx}|+(\Delta y)^2\,|u_{yy}|\,, \tag{22}$$

which is computed using the five-point finite-difference scheme. Note that ESTS acts as a local spatial error indicator. The ESTS values are componentwise scaled with scale($i$pde), which is a user-specified scaling value for the estimated size of the component $i$pde in the PDE system. Hence, we use a relative-error indicator.

Suppose we have just completed the level-k integration of the time step $t_{n-1}\to t_n$. We then compute the maximum $\mathrm{EST}_{\max}$ of the scaled ESTS values over all grid points of the level-k integration domain. If $\mathrm{EST}_{\max}\leqslant \mathrm{TOLS}$, then the local refinement for the current time step is done and the full time step is finished. If $\mathrm{EST}_{\max}>\mathrm{TOLS}$, then it is decided to create a new grid level $(k+1)$ within the time step $t_{n-1}\to t_n$ and a newly refined local subgrid is determined. To determine the new level $(k+1)$ subgrid, all scaled ESTS values are subdued to a second accuracy test which reads

$$\mathrm{ESTS}>2^{-2(r-k+1)}\,\mathrm{EST}_{\max}\,, \tag{23}$$

where r is the anticipated number of refinement levels, which is estimated as

$$r = \text{entier}\ [\log(\text{EST}_{\max}/\text{TOLS})/2\log 2] + k + 1. \tag{24}$$

In addition, we impose $r \leqslant \text{MAXLEV}$, the user-specified maximum number of levels allowed. Note that (23) is more stringent than the first test, $\text{EST}_{\max} > \text{TOLS}$. If (23) holds at a grid point, the four cells surrounding this point will be placed in the newly refined subgrid and once all tests (23) have been made, the refined grid cells are clustered together to form the newly refined subgrid upon which the integration step $t_{n-1} \to t_n$ is to be redone. This subgrid may be disjunct. We refer to Reference 11 for a justification of (23) and for other details on the implemented refinement strategy. This paper also gives a more or less full account of the data structure, memory use, interpolation aspects, etc.

3.4. *Time integration aspects*

We have implemented the two-step BDF method (20) of order two which we apply in the variable step size mode. The integration coefficients then are

$$a_1 = (c+1)^2/(c^2+2c), \quad a_2 = -1/(c^2+2c), \quad \theta_2 = (c+1)/(c+2), \tag{25}$$

with $c = \Delta t_{n-1}/\Delta t_n$ the bounded step size ratio. We recall that variable time stepping is a prerequisite for our example problems as they show a highly distinct behaviour in time. As starting formula, we employ the one-step BDF method (20) of order one (backward Euler).

The following simple step size strategy has been implemented. For the backward Euler step we use the time error monitor $(\Delta t)\partial u/\partial t$ and for all following steps $1/2(\Delta t)^2\partial^2 u/\partial t^2$. Hence, the temporal local error control is based on the last Taylor term taken into account by the integration formula. The derivatives are estimated by using the available approximate u-values at the current time level t_n and past levels t_{n-1}, t_{n-2} (simple differencing). After each level integration step, we invoke the test

$$\text{ESTT} \leqslant \text{TOLT}, \tag{26}$$

where TOLT is a user-specified time error tolerance and ESTT is the maximum of the monitor values computed over the integration domain in use. Also here scaling of ESTT is carried out. If (26) is false, then the full time step is rejected. Otherwise, the level integration step is accepted. For each such step a new Δt is computed such that the predicted value of the monitor is equal to TOLT/2. The minimum of these new Δt values is then taken as the step size Δt_{new} to be attempted in the new full step. However, in case of a step rejection, $\Delta t_{\text{new}} = 0{\cdot}8\Delta t_{\text{new}}$ and in all cases we require that $\Delta t_{\text{old}}/3 \leqslant \Delta t_{\text{new}} \leqslant 2\Delta t_{\text{old}}$ to avoid too large jumps in the step size selection. Finally, Δt_{new} is corrected with a small number to assume that the next output point always coincides with a time level t_n, assuming that till the next output point Δt does not change.

3.5. *The solution of the non-linear and linear systems*

Because we use an implicit integration method and treat PDE problems like (10)–(12) fully coupled, we are facing the task of solving large coupled systems of non-linear algebraic equations. Note that this is required for each grid level within any full time step and that the dimension of the systems varies per full time step and per level. Needless to say that the implicit equation solution is highly important for efficiency and robustness. In our research code we use the modified Newton method (Jacobian computation only at the start of the iteration) in combination with the iterative, preconditioned conjugate gradient-squared method (CGS)[10] for solving the arising coupled systems of linear algebraic equations. In the remainder of this section we will briefly outline how we use the iterative modified Newton and CGS methods.

The Newton implementation ideas have been borrowed from Reference 2 and are to a large extent determined by the PDE system format (18). For any PDE problem fitting in this format, the required Jacobian matrix for the Newton process is computed in a completely automatic manner. To illustrate this, consider the 1D form

$$G(u_t, u_x, u_{xx})=0, \tag{27}$$

for which the Jacobian matrix is tridiagonal. Recall that we use central three-point finite differencing on a uniform space grid with grid distance Δx. The diagonal entries, corresponding with internal grid points, are then easily seen to be defined by the functional

$$\frac{\partial G}{\partial u_t}\frac{1}{\theta_s \Delta t}-\frac{\partial G}{\partial u_{xx}}\frac{2}{(\Delta x)^2}. \tag{28}$$

Similar expressions are easily found for the non-diagonal entries and for grid points whose finite-difference expression is dependent on the boundary condition. In our research code the partial derivatives are estimated by a simple first-order difference formula, so that the user does not need to specify these. For the general format (18) the Jacobian computation goes completely identical. See Reference 9 for related work.

The Jacobian is computed at the beginning of each integration step. A difficulty at the initial time step is that, in general, no expression for $\partial u/\partial t$ is available in explicit form. At the initial time step we, therefore, put the temporal derivative to zero and to compensate for this rough guess we use the damped modified Newton iteration with a damping factor equal to 0·75 and allow a maximum of 20 iterations to get the integration started. This difficulty exists only for the initial coarse grid step, because in all other cases solution approximations are available to generate approximations for $\partial u/\partial t$. In these cases we use the modified Newton iteration without damping and allow a maximum of 10 iterations.

The Newton stopping criterion is based on the relative change in successive solution values. It reads

$$\max_{i\text{pde}} \max_i \frac{|u^{(k)}_{i\text{pde};i}-u^{(k-1)}_{i\text{pde};i}|}{w_{i\text{pde};i}}<1, \tag{29}$$

where

$$w_{i\text{pde};i}=10^{-2}\min(\text{TOLS},\ \text{TOLT})\max\left[|u^{(k)}_{i\text{pde};i}|,\ 10^{-2}\text{scale}(i\text{pde})\right]. \tag{30}$$

The upper index k refers to the kth Newton iterate, the lower index $i$pde to the ith PDE component, and the lower index i to the grid points in the two-dimensional integration domain. Note that the stopping criterion is a relative-error test. We decide that the Newton iteration terminates unsuccessfully when either the stopping criterion is not satisfied after the allowed maximum number of iterations, or when the relative change in successive solution values is increasing and the stopping criterion is not satisfied. If this is the case, then the current time step is rejected and we repeat this full time step with $\Delta t_{\text{new}}=\Delta t_{\text{old}}/4$.

The linear systems arising in the iterative Newton process are iteratively solved using the CGS method with ILU preconditioning.[10] For this purpose we incorporated the public domain code from the SLAP library written by Anne Greenbaum and Mark K. Seager, which is available from Netlib.[3] This code performs quite successfully for our brine transport applications, and also in other tests, but needed some adaptations because its breakdown and stopping criteria are obviously not intended for solving subsequent iterations in a Newton process. For example, the stop criterion in the Netlib code is relative to the right-hand-side vector which vanishes in the Newton iteration. Here we experienced some difficulties. If we denote the non-linear algebraic equation system by $R(Y)=0$ and the kth linear system to be solved by

$$\frac{\partial R}{\partial Y}(Y^0)(Y^{(k)}-Y^{(k-1)})=-R(Y^{(k-1)}), \tag{31}$$

our (still heuristic) stopping criterion reads

$$\max_{i\text{pde}} \max_i \frac{|(K^{-1}r^{(j)})_{i\text{pde};i}|}{w_{i\text{pde};i}} < \frac{1}{2\times\text{maximal no. of Newton iterations}} \tag{32}$$

where K is the ILU decomposition of the Jacobian matrix $\partial R/\partial Y$ and $r^{(j)}$ is the residual of the linear system after the jth CGS iteration.

4. NUMERICAL ILLUSTRATIONS

4.1. Example problem I

Because the initial salt mass fraction is zero, a steep salt front will emerge near the two gates immediately when they are opened. This means that at the start of the integration process a fine space mesh is required in the neighbourhood of the gates, together with small temporal integration steps, so as to stimulate accurately the rapid onset of the fronts. The fine grid is realized by grid refinement near the gates and the small temporal step size simply by starting with a small step size value. For early times in the integration the two salt fronts are steep ($\alpha_T=0{\cdot}002$, $\alpha_L=0{\cdot}01$) and for later times they smooth out due to the physical dispersion. In fact, for $t\to\infty$ the system runs into steady state with uniform salt mass fraction $\omega=\omega_0$. This means that for later times the space grid should be coarsened again and, most importantly, larger and larger step sizes in time should be taken to realize an efficient march to steady state. These comments also apply to the temperature distribution. Immediately after the gates are opened, a steep temperature front emerges. However, this front travels slower than the salt front and smooths out more rapidly since heat is absorbed by the porous medium.

Using fourth-order Lagrangian interpolation (the operator P^k_{k-1} of Section 3), we prescribed the following set of numerical parameters:

$$\text{TOLT}=0{\cdot}01,\quad \Delta t_0=0{\cdot}001,\quad \text{TOLS}=0{\cdot}01,\quad \Delta x=\Delta y=1/20,\quad \text{MAXLEV}=3,$$
$$\text{scale }(1)=10^5,\quad \text{scale }(2)=0.25,\quad \text{scale }(3)=290, \tag{33}$$

where Δx, Δy are the grid sizes of the coarsest grid. Since MAXLEV$=3$, the finest grid size allowed is 1/80. For the chosen pair of dispersivity coefficients α_T and α_L, this is sufficiently small to avoid wiggles (the spatial discretization in the LUGR code is based on central differences). Note that the length of the integration interval is 10^6 (steady state), which illustrates that a large variation in step size Δt will occur.

Tables II and III provide an insight into the performance and efficiency of the integration process. Apparently, both the variable step size and the modified Newton strategies work successfully. Only one time-step rejection was counted and no Newton failure has occurred. The average number of modified Newton iterations per step per level is slightly above 3, which seems reasonable in view of the non-linearities in the brine transport model. Note that we prefer to choose the maximal allowed number of modified Newton iterations relatively large to avoid, as much as possible, Newton failures, since such a failure involves a full time-step rejection. The

Table II. Example I. Intergration history information for the numerical parameter set (33)

No. of accepted time steps = 158	No. of rejected time steps = 1	No. of Newton failures = 0
No. of Newton iterations level 1 = 554	No. of Newton iterations level 2 = 492	No. of Newton iterations level 3 = 442

Table III. Example I. No. of CGS iterations for the numerical parameter set (33) counted per modified Newton iteration and levelwise accumulated over all time steps

	Level 1	Level 2	Level 3
1st Newton iteration	1931	1940	1980
2nd Newton iteration	1387	1034	931
3rd Newton iteration	577	269	170
4th Newton iteration	140	35	21
5th Newton iteration	41	2	2
6th Newton iteration	2	1	1
7th Newton iteration	5	0	0
8th Newton iteration	2	0	0
	——+	——+	——+
	4085	3281	3105

average number of CGS iterations per modified Newton iteration is about 7, which is very effective. Note that these numbers refer to the complete integration process over the entire time interval $[0, 10^6]$, using a total of 158 full time steps with a heavily varying step size Δt. Also note that at level 2 and 3 the work load differs per time step, since at these levels the integration domains and, hence, the size of the matrix problems vary in time.

Figures 1 and 2 show the salt mass fraction and the temperature distribution at various time points together with the grid configuration. The figures nicely show the grid evolution in time. Note that for early times the fine grid covers only a small portion of the space domain. Upon reaching the steady state, where the distributions become constant, the number of grid levels is automatically reduced to one (not shown in the figures). Hence, the method then integrates only on the coarse 20×20 grid. A comparison of the two figures shows that the temperature front is slower than the salt front and also less steep, which is in accordance with the physics. We also see time-dependent refinement near the vertical and upper boundary of the (vertical) space domain. This particular refinement is due to the Neumann conditions (see first and second line of (15)). When the salt front hits these boundaries, the Neumann condition becomes difficult to solve on a coarse grid as this condition gives rise to a kink in the solution. Hence, a local refinement is natural. For later values of time the kink disappears and the refinement is no longer needed.

4.2. *Example problem II*

In the numerical experiment with this example we have closed the right gate so that salt water is injected only through the left one. Hence, here we deal with a single saltwater–freshwater front which first collides with the region of impermeability and then must flow around it. This gives rise to a more difficult flow pattern than in Example I and, hence, also to a more costly computation.

For ease of application of our (present) research code we used simple second-order linear interpolation. The more accurate fourth-order Lagrangian interpolant can be used too, but

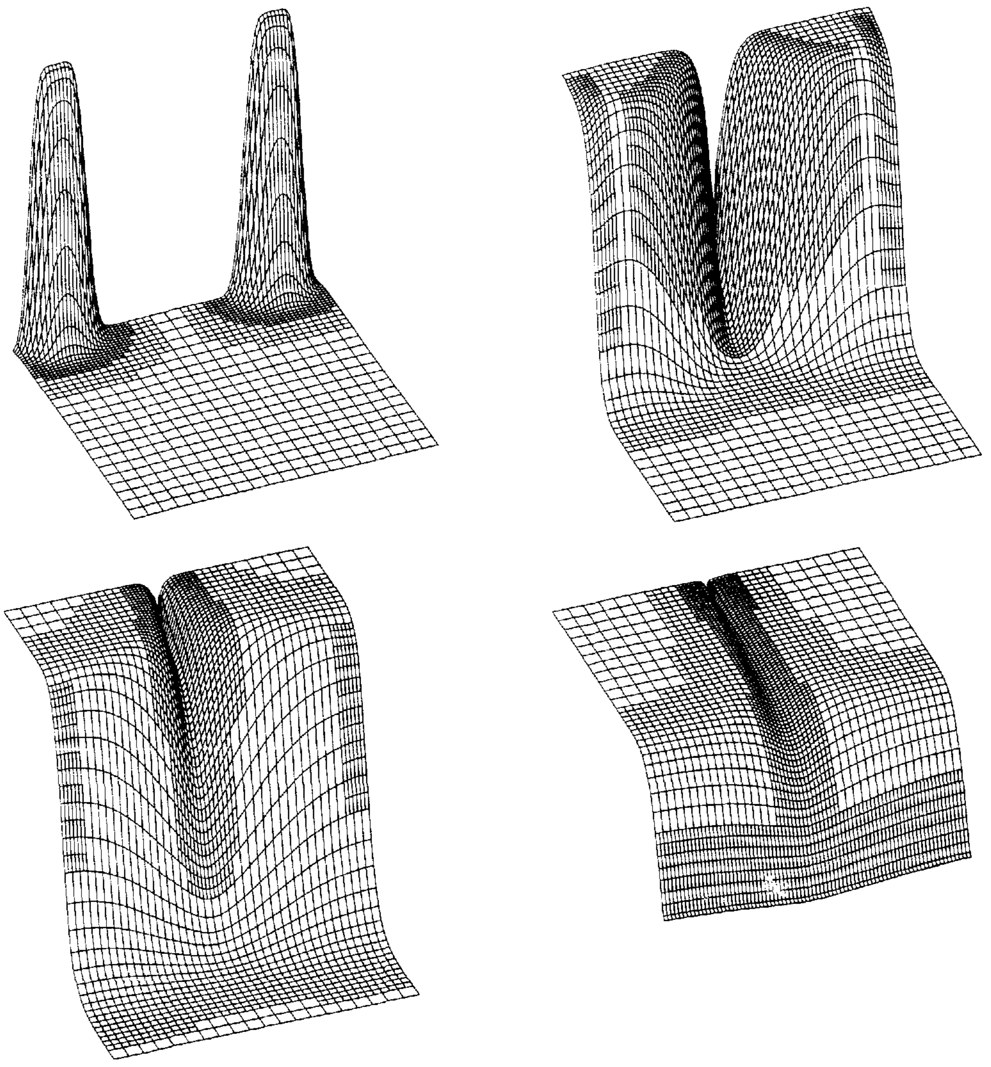

Figure 1. Example I. ω-distribution at $t=5\times10^2$, 5×10^3, 10^4, 2×10^4 with the corresponding grid configuration

requires some additional programming effort. Tables IV and V give the same information as Tables II and III, again for the numerical parameter set (33). The average number of modified Newton iterations per step per level is about 3·5 and the average number of CGS iterations per modified Newton iteration is about 8 for level 1 and 6 for levels 2 and 3. We conclude that also in this experiment the BDF scheme with the implemented solvers performs well.

Figure 3 shows the salt mass fraction at various time points, together with the grid configuration. These time points are taken larger than in Example I since the salt intrusion around the region of impermeability needs more time. Of course, for early times, i.e. before the salt front collides with this region, the salt mass concentration and the grid configuration are similar as in Figure 1, except that around the closed right gate no front exists, thus, in this neighbourhood the coarse 20×20 grid is used. A careful inspection of the solution plot for $t=30000$ reveals some

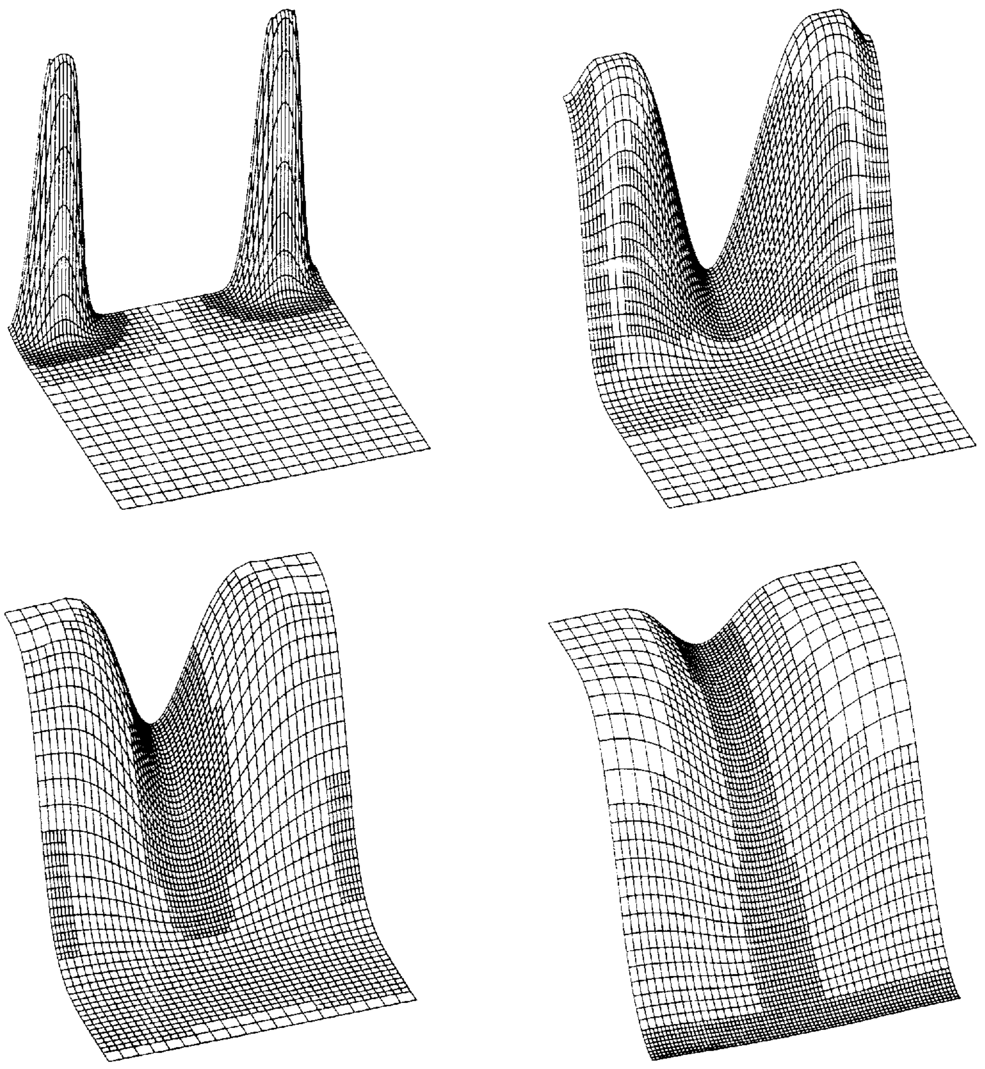

Figure 2. Example I. T-distribution at $t=5\times10^2$, 5×10^3, 10^4, 2×10^4 with the corresponding grid configuration

Table IV. Example II. Intergration history information for the numerical parameter set (33)

No. of accepted time steps = 262	No. of rejected time steps = 4	No. of Newton failures = 0
No. of Newton iterations level 1 = 992	No. of Newton iterations level 2 = 895	No. of Newton iterations level 3 = 835

small inaccuracies near the local refinements along the right vertical and upper boundary. As in Example I, the refinements are necessary to resolve accurately the Neumann boundary condition. Apparently, at this point of time, a larger region of refinement seems desirable here.

To provide an accuracy measure, we have included Figure 4. This figure shows the so-called breakthrough curves for the salt mass fraction ω obtained in two experiments. The first one is the

Table V. Example II. No. of CGS iterations for the numerical parameter set (33) counted per modified Newton iteration and levelwise accumulated over all time steps

	Level 1	Level 2	Level 3
1st Newton iteration	3178	2839	2893
2nd Newton iteration	2497	1487	1387
3rd Newton iteration	1286	669	360
4th Newton iteration	577	201	87
5th Newton iteration	163	1	24
6th Newton iteration	5	0	6
7th Newton iteration	5	0	0
	——+	——+	——+
	7711	5197	4757

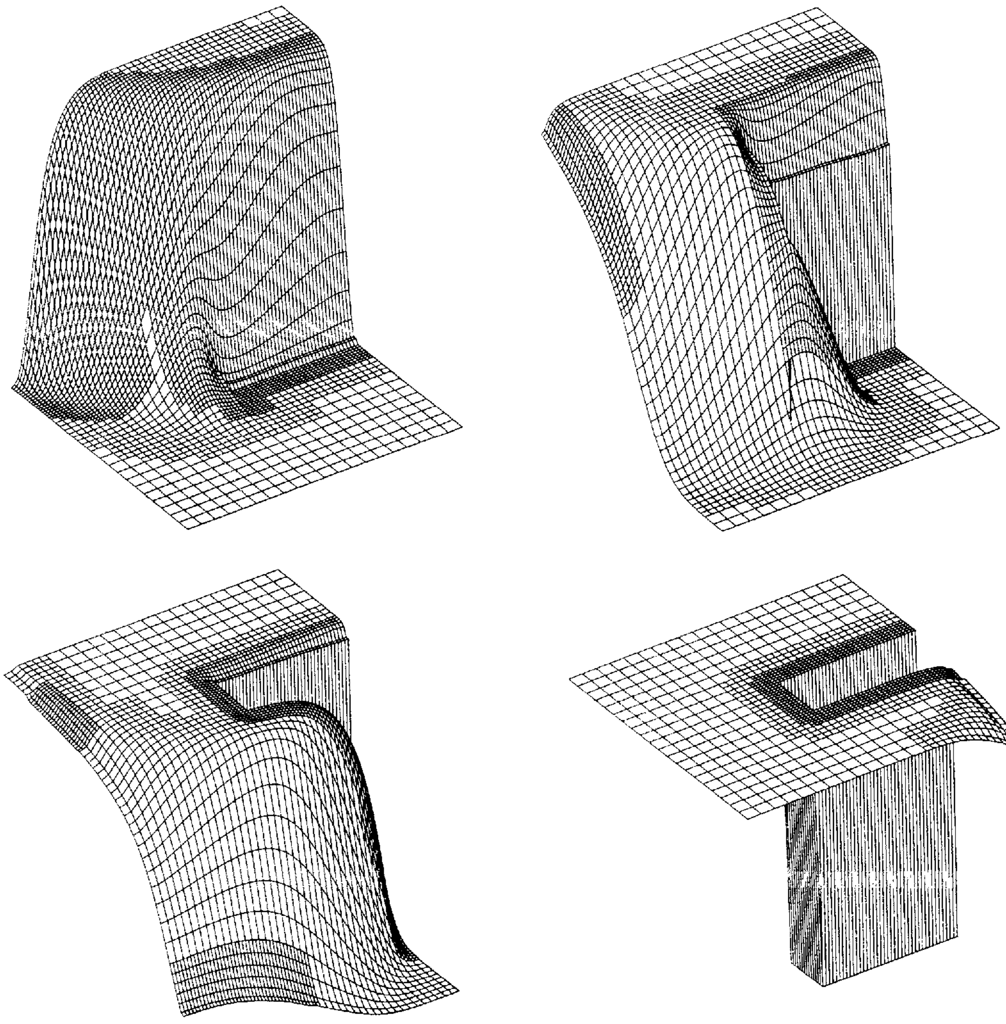
Figure 3. Example II. ω-distribution at $t = 10^4$, 2×10^4, 3×10^4, 6×10^4 with the corresponding grid configuration

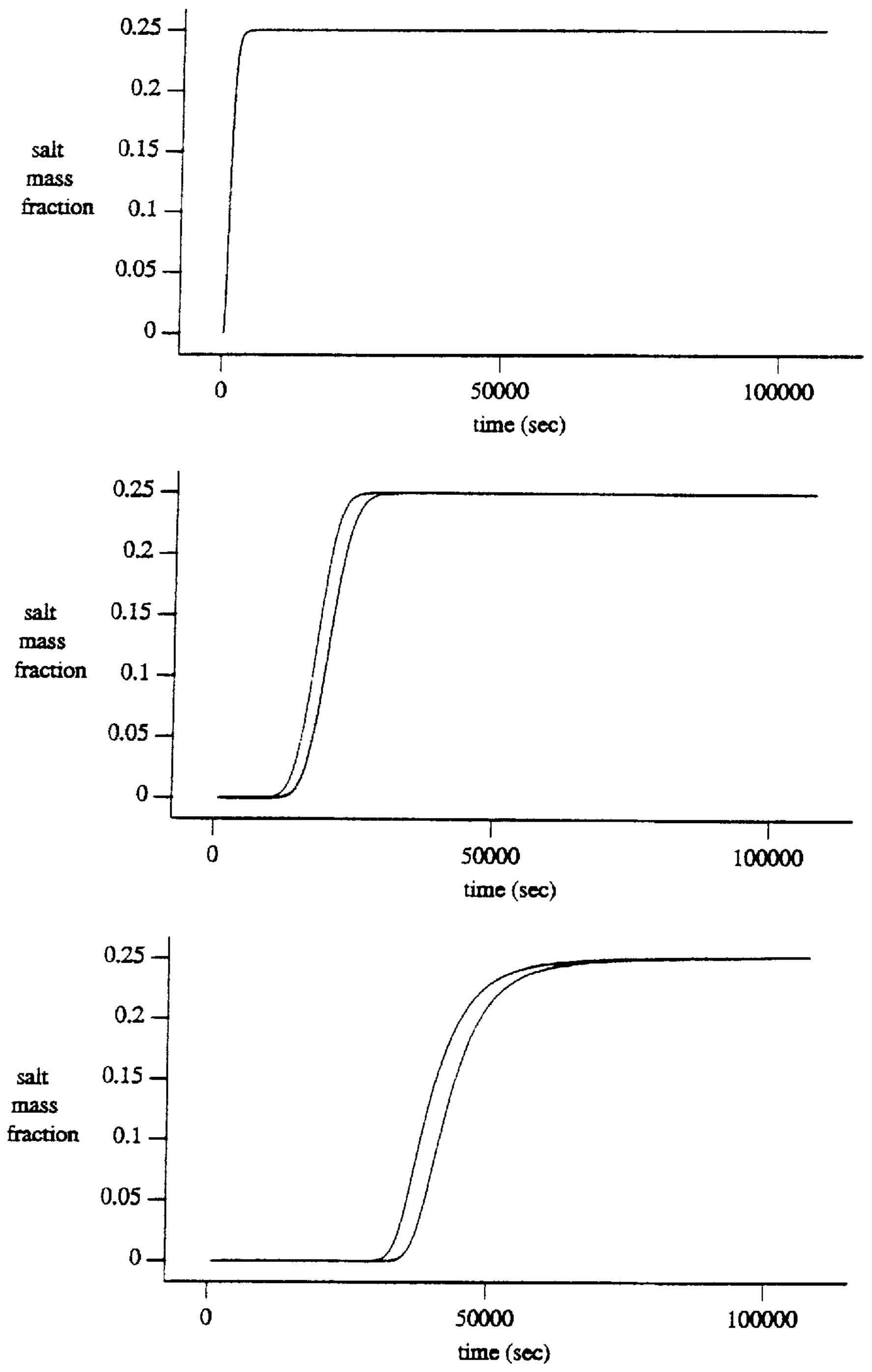

Figure 4. Example II. Breakthrough curves for ω for $0 \leqslant t \leqslant 10^6$ and $(x, y) = (0{\cdot}1125, 0{\cdot}1)$ (upper plot), $(x, y) = (0{\cdot}7625, 0{\cdot}5)$ (middle plot), $(0{\cdot}1125, 0{\cdot}8)$ (lower plot)

Table VI. Example II. No. of CGS iterations on the uniform 80×80 grid for the numerical parameter set (33), counted per modified Newton iteration and accumulated over all 236 time steps. The average no. of CGS iterations per Newton iteration is 25·0. Per time step the average amounts to 80·2

Newton iteration no.	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	Total
	11 663	5505	1318	364	5	27	1	41	2	18 923

three-level experiment. In the second experiment we have rerun the problem, using again the parameter set (33), but now on a single 80×80 uniform grid. A third experiment on the 80×80 uniform grid with TOLT ten times smaller has been carried out to assure that the temporal error is not visible in the uniform-grid curves. This indeed turned out to be true so that the breakthrough curves obtained in the second experiment can be used to check whether the local refinement alone introduces a visible difference. Recall that the finest grid of the three-level experiment has also 1/80 as mesh width. Figure 4 shows that in the three-level experiment the salt front travels slightly faster (upper line) through the point $(x, y) = (0{\cdot}7625, 0{\cdot}5)$ at the right of the region of impermeability and $(x, y) = (0{\cdot}1125, 0{\cdot}8)$ above it, while at the point $(x, y) = (0{\cdot}1125, 0{\cdot}1)$ beneath it the curves are equal up to plotting accuracy. We blame the heuristic refinement strategy for the observed phase errors. The heuristics simply lies in the fact that the phase error can be removed by forcing more local refinement. The plots in Figure 3 indeed reveal that when the front travels through $(x, y) = (0{\cdot}7625, 0{\cdot}5)$, a grid size of 1/40 is used, whereas at the first point the finest grid size is 1/80 upon front passing. On the other hand, the mathematically rigorous local refinement theory proposed in Reference 12 assures that the local refinement does not decrease the spatial accuracy, while it also attempts to avoid unnecessiraly large local refinement. This theory also controls accumulation of the inevitable (in this case linear) interpolation errors. Apparently, the comparison supports the use of more rigorous strategies like that proposed in Reference 12.

We conclude this section with some integration history of the uniform 80×80 computation, so as to enable an efficiency comparison with the data of Tables IV and V. We counted 236 accepted time steps, 1 rejected time step, 758 modified Newton iterations and 0 Newton failures. Hence, the uniform grid integration requires somewhat less steps and also less Newton iterations than the three-level integration. However, due to the smaller matrix dimensions, in the three-level computation the required total number of CGS iterations on grid level 3 is considerably less than the total number required in the uniform grid case (4757 against 18 923; see Table VI). In the three-level computation we also have to take into account the work load for level 2 and level 1 and other overhead, but, as a rule, the finest grid computation is the most expensive one.

Finally, we give some CPU times. On the CRAY Y-MP4/464 the code needed 3969 s for the 80×80 uniform grid computation, versus 914 s. for the three-level one (using one CPU). On the INDIGO SGI workstation the CPU times are, respectively, 24 886 s and 4116 s. We emphasize that, except for the Netlib CGS routine, the code has not yet been optimized towards the CRAY architecture.

5. FINAL REMARKS

Adaptive-grid methods are meant for problems possessing rapid local transitions in their solution. The brine transport problem of Section 2 possesses such rapid transitions, both with respect to the spatial variables and temporal variable. Hence, this particular flow problem is an excellent candidate to be solved on time-dependent adaptive grids, while using, in addition, variable step sizes in time.

Our method of choice for time-dependent adaptive grids is based on the local uniform grid refinement approach. The LUGR approach of integrating on finer and finer nested subgrids, overlaying one another, provides much flexibility in the selection of the discretization schemes. An adaptive-grid LUGR method can be combined in many different ways and, if desired, highly tailored to the problem class at hand in connection with the choice of spatial discretization and time-stepping scheme.[11 – 14] Note that our Cartesian grid structure, which for certain applications can be too restrictive for geometrical reasons, is not necessary and finite-element or finite-volume schemes on structured triangular grids can be developed as well.

As mentioned in Section 3.3, in the near future we will first extend the refinement analysis of Reference 12 to the genuine DAE case. Although the present strategy from Reference 11 works alright, its heuristics leave the user with the task of choosing a reasonable value for the parameter TOLS. The comparison carried out for Example II has nicely illustrated this. The refinement analysis from Reference 12 underlies the assumption that the spatial error of the multilevel scheme should be equal to the spatial error of the finest gridlevel used without any adaptation. This, of course, is an optimal situation for LUGR methods.

We emphasize that the fully implicit LUGR method of Section 3 has not been tailored to the current brine transport problem. On the contrary, it can be used on a much wider class of PDEs of first-order in time and second-order in space. This method has in fact been set up in accordance with the method of lines approach: the spatially discretized problem, here obtained with second-order central differencing, is numerically treated by the second-order BDF method as if it is a common stiff ODE/DAE system. In the special case of the brine transport problem, this may not necessarily be the fastest way of time stepping. On the other hand, this approach is most reliable and very robust. For example, it readily admits the inclusion of more terms in the PDE problem, like sources and sinks, chemistry, etc. Also note that the brine transport problem requires an implicit method (for the mass balance equation), because the mass balance equation becomes infinitely stiff for $\beta = 0$, using the method-of-lines terminology. Obviously because the LUGR method works fully implicit and the coupled brine transport problem gives rise to large sets of non-linear algebraic equations, even on the local subgrids, an efficient solution procedure for these sets is a prerequisite. To our experience, the combination of the modified Newton method and the linear solver CGS performs very satisfactorily on the brine transport problem (see Section 4). However, we believe that there is still room for improvement since the development of fast iterative non-symmetric-matrix solvers is ongoing. Needless to say that in the setting of our Cartesian grid structure, the multigrid technique is also an excellent candidate to be tried out.

ACKNOWLEDGEMENT

We gratefully acknowledge our colleagues J. C. H. van Eijkeren, S. M. Hassanizadeh and A. Leijnse from the RIVM for the pleasant co-operation and for their advice in the selection of the example problems.

REFERENCES

1. J. Bear and A. Verruyt, *Modeling Groundwater Flow and Pollution*, Reidel, Dordrecht, 1987.
2. J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
3. J. J. Dongarra and E. Grosse, *Distribution of mathematical software via electronic mail.* Communications of the ACM 30, pp. 403–407 (netlib@research.att.com) 1987.

4. S. M. Hassanizadeh and T. Leijnse, 'On the modeling of brine transport in porous media', *Water Resources Research*, **24,** 321–330 (1988).
5. S. M. Hassanizadeh, 'Experimental study of coupled flow and mass transport: a model validation exercise', in K. Kovar (ed.), *Calibration and Reliability in Groundwater Modeling*, IAHS Publication No. 195, Wallingford, Oxfordshire, U.K, 1990.
6. S. M. Hassanizadeh and J. C. H. van Eijkeren, Private communication, 1991.
7. S. M. Hassanizadeh, A. Leijnse, W. J. de Vries and R. A. M. Stapper, 'Experimental study of brine transport in porous media, Intraval test case 13', *Report no. 725206003*, National Institute of Public Health and Environmental Protection, Bilthoven, The Netherlands, 1990.
8. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Equations*. Springer Series in Computational Mathematics 14, Springer, Berlin, 1991.
9. W. Schönauer and N. E. Schnepf, *Software considerations for the black-box solver FIDISOL for partial differential equations*, *ACM TOMS* **13,** 333–349 (1987).
10. P. Sonneveld, 'CGS, a fast Lanczos-type solver for nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **10,** 36–52 (1989).
11. R. A. Trompert and J. G. Verwer, 'A static-regridding method for two-dimensional parabolic partial differential equations', *Appl. Numer. Math.*, **8,** 65–90 (1991).
12. R. A. Trompert and J. G. Verwer, 'Analysis of the implicit Euler local uniform grid refinement method', *CWI Report NM-R9011*, 1990, to appear in *SIAM J. Sci. Stat. Comput.*, **14**(2), (1993)
13. R. A. Trompert and J. G. Verwer, 'Runge–Kutta methods and local uniform grid refinement', *CWI Report NM-R9022* (1990), to appear in *Math. Comp.*
14. J. G. Verwer and R. A. Trompert, 'An adaptive-grid finite-difference method for time-dependent partial differential equations', in D. F. Griffiths and G. A. Watson (eds.), *Proc. 14th Biennial Conference on Numerical Analysis*, Dundee, Scotland, 1991, Pitman Research Notes in Mathematics Series **260,** 267–284 (1992).
15. P. A. Zegeling, J. G. Verwer and J. C. H. van Eijkeren, 'Application of a moving-grid method to a class of 1D brine transport problems in porous media', *Int. j. numer. methods fluids*, **15,** 175–191 (1992).