Algorithm 731: A Moving-Grid Interface for Systems of One-Dimensional Time-Dependent Partial Differential Equations

J. G. BLOM
CWI
and
P. A. ZEGELING
RUU

In the last decade, several numerical techniques have been developed to solve time-dependent partial differential equations (PDEs) in one dimension having solutions with steep gradients in space and in time. One of these techniques, a moving-grid method based on a Lagrangian description of the PDE and a smoothed-equidistribution principle to define the grid positions at each time level, has been coupled with a spatial discretization method that automatically discretizes the spatial part of the user-defined PDE following the method of lines approach We supply two FORTRAN subroutines, CWRESU and CWRESX, which compute the residuals of the differential algebraic equations (DAE) system obtained from semidiscretizing, respectively, the PDE and the set of moving-grid equations. These routines are combined in an enveloping routine SKMRES, which delivers the residuals of the complete DAE system To solve this stiff, nonlinear DAE system, a robust and efficient time-integrator must be applied, for example, a BDF method such as implemented in the DAE solvers SPRINT [Berzins and Furzeland 1985; 1986; Berzins et al. 1989] and DASSL [Brenan et al. 1989; Petzold 1983]. Some numerical examples are shown to illustrate the simple and effective use of this software interface.

Categories and Subject Descriptors: G.1.0 [**Numerical Analysis**]: General; G.1.8 [**Numerical Analysis**]: Partial Differential Equations; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms Algorithms, Performance, Reliability

Additional Key Words and Phrases Lagrangian methods, mathematical software, method of lines, moving grids, partial differential equations, time-dependent problems

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 Organization for the Advancement of Research (NWO) through the Netherlands Foundation for the Technical Sciences (STW), Contract CWI 59.0922.

Authors' addresses: J. G. Blom, CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands; P. A. Zegeling, Rijksuniversiteit Utrecht, Mathematisch Instituut, Budapestlaan 6, POB 80010, 3508 TA Utrecht

1. INTRODUCTION

In this paper we describe a FORTRAN 77 software interface for the spatial discretization of systems of one-dimensional time-dependent partial differential equations (PDEs). The problem of solving PDEs using an interface on fixed grids, by which the user only needs to define the PDE and accompanying boundary conditions in two subroutines has already been treated by, for example, Sincovec and Madsen [1975], Berzins and Furzeland [1986], and Bakker [1977]. However, in many models from physics and chemistry, steep spatial and temporal gradients may occur in the solution that cause problems if a numerical solution method on a fixed grid is used. In such cases moving-grid methods are likely to be more successful. In Furzeland et al. [1990], an evaluation has been made of the behavior of three moving-grid methods with respect to efficiency and robustness on three difficult test problems having steep moving fronts. On account of the hopeful results of this investigation, we decided to develop a moving-grid interface, that is, an interface with the possibility to move the grid points continuously in time, and to couple this interface with one of the methods from Eurzeland et al. [1990], namely, the method due to Dorfi and Drury [1987]. For this purpose, we implemented a finite-element discretization for the Lagrangian forms of the PDE. This discretization method, borrowed from Skeel and Berzins [1990], is akin to a nonlinear Galerkin method and is of second-order accuracy in space, even for polar problems. The resulting interface is available in standard FORTRAN 77 both in single and double precision. It consists of two subroutines, CWRESU and CWRESX. CWRESU computes the residuals of the semidiscretized Lagrangian PDE and is a modification of the subroutine SKEEL4 of the SPRINT package [Berzins and Furzeland 1985; 1986; Berzins et al. 1989]. CWRESX computes the residuals of a differential algebraic equation (DAE) system that governs the grid motion. These subroutines are called from a subroutine SKMRES, which is the moving-grid equivalent of the SPRINT routine SKLRES. The coupled DAE system is stiff and should be integrated in time with a robust DAE solver. Originally, the PDE interface module had been adapted to the SPRINT package so that a SPRINT user can make use of the moving-grid spatial discretization routine in a completely analogous way as the fixed-grid discretization module SPDIFF of SPRINT. However, it is set up in a way that makes it possible to use the interface also with other DAE solvers. In this paper we describe its use when coupled with the public-domain code DASSL [Brenan et al. 1989; Petzold 1983]. For more information on the use of the interface in the SPRINT environment, we refer to an internal CWI report [Blom and Zegeling 1989].

Our paper is divided into six sections and an Appendix. Section 2 presents an outline of the considered moving-grid method. In Section 3 we define the class of PDE problems that is allowed in the software interface. Section 4 describes the semidiscrete approximation of the PDE system. In Section 5 we discuss the interface itself. We also show by an example the ease of use of our software interface in combination with the DAE solver DASSL. Section 6

contains two numerical examples to underline the performance of the moving-grid method.

2. THE MOVING-GRID METHOD

In this section we give a brief description of the moving-grid techique that controls the spatial grid movement in time. This technique is due to Dorfi and Drury [1987]. For the theoretical background and some analytical aspects of the method, we refer to Verwer et al. [1987; 1989].

A system of PDEs with NPDE component equations

$$u_t = f(u, x, t), \qquad x_L < x < x_R, \qquad t > t_0, \tag{2.1}$$

is transformed to its Lagrangian form, using the total differential $\dot{u} = du/dt = u_t + u_x \dot{x}$,

$$\dot{u} - u_x \dot{x} = f(u, x, t). \tag{2.2}$$

We then choose N time-dependent grid points

$$x_L = X_0 < \cdots < X_i(t) < X_{i+1}(t) < \cdots < X_{N+1} = x_R, \tag{2.3}$$

using a second-order discretization in space we obtain from (2.2),

$$\dot{U}_i - \frac{(U_{i+1} - U_{i-1})}{(X_{i+1} - X_{i-1})} \dot{X}_i = F_i, \qquad t > t_0, \qquad 1 \le i \le N. \tag{2.4}$$

Here, U_i and F_i represent the semidiscrete approximation to the exact PDE solution u, respectively, the right-hand-side function $f(u, x, t)$, at the point $(x, t) = (X_i(t), t)$. To solve the ODE system (2.4), additional equations are required for the time-dependent grid points X_i, which are as yet unknown. For this purpose the next quantities are defined:

$$\begin{aligned} &n_i := (\Delta X_i)^{-1}, \qquad \Delta X_i := X_{i+1} - X_i, \\ &\tilde{n}_i := n_i - \kappa(\kappa + 1)(n_{i+1} - 2n_i + n_{i-1}), \qquad 0 \le i \le N \\ &\qquad (n_{-1} := n_0, n_N =: n_{N+1}), \end{aligned} \tag{2.5}$$

where n_i stands for the so-called point concentration of the grid and $\kappa \ge 0$ denotes a spatial smoothing parameter. Now we define implicitly, in terms of $\tilde{n}_i$, the movement of the grid points X_i

$$\frac{\tilde{n}_{i-1} + \tau \dot{\tilde{n}}_{i-1}}{M_{i-1}} = \frac{\tilde{n}_i + \tau \dot{\tilde{n}}_i}{M_i}, \qquad 1 \le i \le N, \tag{2.6}$$

where $\tau \ge 0$ is a time-smoothing parameter and M_i is a monitor function, here chosen as follows:

$$M_i := \sqrt{\alpha + \mathrm{NPDE}^{-1} \sum_{J=1}^{\mathrm{NPDE}} (U_{i+1}^J - U_i^J)^2 / (\Delta X_i)^2}, \tag{2.7}$$

which is a semidiscrete representation of the first derivative solution functional

$$m(u) = \sqrt{\alpha + \|u_x\|^2}\,.$$

Shortly, κ determines the level of clustering of the grid points, and the arclength monitor M_t determines the shape of the X_t-distribution. The parameter τ prevents the grid movement from adjusting immediately to new values of the monitor function M_t, therefore trying to avoid temporal oscillations in the grid that may cause relatively large errors, when applied to solutions with steep gradients. Eqs. (2.4) and (2.6) are combined to yield the system of DAEs

$$\begin{aligned} \dot{U} - D\dot{X} &= F, \\ \tau B\dot{X} &= g, \end{aligned} \tag{2.8}$$

where definitions (2.5) have been used, D and B are solution-dependent matrices, and F and g are solution-dependent vectors, containing all information about the monitor function and the right-hand side of the PDE itself, respectively. System (2.8) can be rearranged in the linearly implicit form

$$A(Y)\dot{Y} = L(Y), \tag{2.9}$$

where

$$Y := (\ldots, U_t^1, \ldots, U_t^{\mathrm{NPDE}}, X_t, \ldots)^T.$$

This stiff DAE system may be solved, for example, by a BDF integrator. The user of the interface who is interested in the details of the moving-grid method is advised to read Verwer et al. 1989.

3. PDE DEFINITION

The class of allowable PDE problems is derived from the class defined in the interface of the SPRINT package [Berzins and Furzeland 1985; 1986; Berzins et al. 1989] and is equivalent to the PDE definition used in the NAG library routine D03PCF [N. A. G. 1991],

$$\sum_{k=1}^{\mathrm{NPDE}} C_{j,k}(x,t,u,u_x)\frac{\partial u^k}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m R_j(x,t,u,u_x)\right) - Q_j(x,t,u,u_x) \tag{3.1}$$

for $j = 1, \ldots, \mathrm{NPDE}$ and $x \in [x_L, x_R]$, $t > t_0$, $m \in \{0, 1, 2\}$, where NPDE denotes the number of PDEs, $u := (u^1, u^2, \ldots, u^{\mathrm{NPDE}})^T$ is the solution vector, and R_j and Q_j can be thought of, in special cases, as flux and source terms, respectively. The solution u and the functions $C_{j,k}$, R_j, and Q_j are expected to be continuous functions. For problems in Cartesian coordinates, $m = 0$, whereas $m = 1$ indicates a cylindrical polar coordinate system, and $m = 2$ a spherical polar one. In the case $m > 0$, we require that $x_L \geq 0$.

The user-specified boundary conditions belonging to system (3.1) have the master equation form

$$\beta_j(x,t)R_j(x,t,u,u_x) = \gamma_j(x,t,u,u_x)$$

$$\text{at} \quad x = x_L \quad \text{and} \quad x = x_R, \quad \text{for} \quad j = 1,\ldots,\text{NPDE}, \quad (3.2)$$

where β_j and γ_j are continuous functions of their variables. If $m > 0$ and $x_L = 0$, the boundedness of the solution near $x = 0$ must be ensured. This requires either the specification of the solution at $x = 0$ or implies that $R_j|_{x=x_L} = 0$. Simple boundary conditions, such as Dirichlet or Neumann conditions, are very easy to define, using eq. (3.2), as shown in the example in Section 5. As can be seen in Section 2, the appearance of the time derivatives in (3.1) is an essential part of the moving-grid technique, which means that the interface, in principle, is aimed at parabolic and first-order hyperbolic systems of equations (in this case a dummy boundary equation has to be supplied). But, if one or more equations in a system of PDEs are elliptic, this still fits in the class (putting $C_{j,k}$ equal to 0 for some index j; $k = 1,\ldots,\text{NPDE}$).

The initial conditions must satisfy

$$u(x,t_0) = u^0(x) \quad \text{for} \quad x \in [x_L, x_R], \quad (3.3)$$

where u^0 is a piecewise continuous function of x with NPDE components.

Note that the possibility of coupling (3.1) to a system of ordinary differential equations, as is allowed in the SPRINT interface and in the NAG routine D03PHF, is not recommended for the time being, because the effects, arising from moving the spatial grid in time with respect to the fixed coupling points, have not yet been investigated.

To summarize, the user must define the interval $[x_L, x_R]$, the number of grid points, the functions $C_{j,k}$, R_j, Q_j, β_j, and γ_j, the initial time t_0, the vector u^0, and the numbers m and NPDE. In our opinion, the class of PDE problems defined by (3.1), (3.2), and (3.3) is sufficiently general to treat a number of miscellaneous problems stemming from engineering, physics, and chemistry. Of course, these problems should have solutions that are "sufficiently" smooth in space and time; that is, no real shocks can be solved.

4. SPATIAL DISCRETIZATION

In order to reduce accuracy problems that arise for coefficients like x^{-m} in (3.1) when x is near zero and $m > 0$, a spatial discretization method is used that is second order in space. The nonlinear Galerkin-based method is extensively described in Skeel and Berzins [1990]. In the following we give a summary of this discretization method when applied to the PDE class (3.1) transformed to its Lagrangian form. We omit, however, the error analysis, which can be found in Skeel and Berzins 1990.

First, we apply the Lagrangian transformation. Let w be defined by $w := u_x \dot{x}$ and S_j by

$$S_j = S_j(x, t, u, u_x, \dot{u}, w) := \sum_{k=1}^{\mathrm{NPDE}} C_{j,k}(\dot{u}^k - w^k) + Q_j.$$

Then system (3.1) becomes

$$S_j = x^{-m}(x^m R_j)_x \quad \text{for} \quad j = 1, \ldots, \mathrm{NPDE}, \tag{4.1}$$

with $C_{j,k}$, Q_j, and R_j defined as before. On the spatial grid (2.3), we formulate the Galerkin method for (4.1). Introduce the approximation U^k of u^k:

$$U^k(x) = \sum_{i=0}^{N+1} U_i^k \phi_i^{(m)}(x).$$

Let $\psi_i^{(m)}$ denote the test functions. The trial fuctions $\phi_i^{(m)}$ and the test functions $\psi_i^{(m)}$ are given in an internal CWI report [Blom and Zegeland 1989]. Introduce the weight function x^m, and integrate (4.1) on $[x_L, x_R]$ partially, so as to obtain

$$\int_{x_L}^{x_R} x^m \psi_i^{(m)} S_j \, dx = x_R^m \psi_i^{(m)}(x_R) R_j|_{x=x_R} - x_L^m \psi_i^{(m)}(x_L) R_j|_{x=x_L} - \int_{x_L}^{x_R} x^m \frac{d\psi_i^{(m)}}{dx} R_j \, dx \tag{4.2}$$

for $j = 1, \ldots, \mathrm{NPDE}$ and $i = 0, \ldots, N+1$. Using the fact that $\psi_i^{(m)}(x) = 0$ for $x \le X_{i-1}$ and $x \ge X_{i+1}$, we get for $i = 1, \ldots, N$ and $j = 1, \ldots, \mathrm{NPDE}$

$$\int_{X_{i-1}}^{X_{i+1}} x^m \psi_i^{(m)} S_j \, dx = -\int_{X_{i-1}}^{X_{i+1}} x^m \frac{d\psi_i^{(m)}}{dx} R_j \, dx. \tag{4.3}$$

The integration over an interval $[X_{i-1}, X_i]$ is performed by numerical quadrature using one quadrature point $\xi_{i-1/2}$. After applying the numerical integration on $[X_{i-1}, X_i]$ and $[X_i, X_{i+1}]$ and lumping (i.e., evaluation of $\dot{u}^k$ takes place in X_i rather than in ξ), (4.3) yields

$$\begin{aligned} S_j^{i-1/2} \int_{X_{i-1}}^{X_i} x^m \psi_i^{(m)} \, dx + S_j^{i+1/2} \int_{X_i}^{X_{i+1}} x^m \psi_i^{(m)} \, dx \\ = -\xi_{i-1/2}^{\mu} R_j^{i-1/2} \int_{X_{i-1}}^{X_i} x^{m-\mu} \frac{d\psi_i^{(m)}}{dx} \, dx \\ - \xi_{i+1/2}^{\mu} R_j^{i+1/2} \int_{X_i}^{X_{i+1}} x^{m-\mu} \frac{d\psi_i^{(m)}}{dx} \, dx + E, \end{aligned} \tag{4.4}$$

where

$$S_j^{i-1/2} = S_j\Big(\xi_{i-1/2}, t, U(\xi_{i-1/2}), U_x(\xi_{i-1/2}), \dot{U}_i, W_i\Big),$$

$$S_j^{i+1/2} = S_j\Big(\xi_{i+1/2}, t, U(\xi_{i+1/2}), U_x(\xi_{i+1/2}), \dot{U}_i, W_i\Big), \quad \text{with}$$

$$W_i = \frac{U_{i+1} - U_{i-1}}{X_{i+1} - X_{i-1}} \dot{X}_i,$$

$$R_j^{i+1/2} = R_j\Big(\xi_{i+1/2}, t, U(\xi_{i+1/2}), U_x(\xi_{i+1/2})\Big),$$

and E stands for the total error due to interpolation, quadrature, and lumping. For the definition of μ, we make a distinction between two special cases:

(1) the regular case ($m = 0$ or $x_L > 0$): $\mu = m$, and
(2) the singular case ($m > 0$ and $x_L = 0$): $\mu = -1$.

The choice of ξ depends on μ. On an interval $[x_i, X_{i+1}]$, we choose $\xi_{i+1/2} = \gamma_{-\mu, i+1/2}$, where $\gamma_{p, i+1/2}$ denotes the Gauß-point for the weight function x^p; that is,

$$\int_{X_i}^{X_{i+1}} (x - \gamma_{p, i+1/2}) x^p \, dx = 0. \tag{4.5}$$

The numerical integration in (4.4) is then second-order accurate. If we neglect the error E in (4.4), we arrive at a semidiscrete approximation of (4.1), for $i = 1, \ldots, N$,

$$S_j^{i-1/2} \frac{X_i^{m+1} - \zeta_{i-1/2}^{m+1}}{m+1} + S_j^{i+1/2} \frac{\zeta_{i+1/2}^{m+1} - X_i^{m+1}}{m+1} = \zeta_{i+1/2}^{m-\mu} \xi_{i+1/2}^{\mu} R_j^{i+1/2} - \zeta_{i-1/2}^{m-\mu} \xi_{i-1/2}^{\mu} R_j^{i-1/2}, \tag{4.6}$$

with

$$\zeta_{i+1/2}^{m+1} = -\int_{X_i}^{X_{i+1}} x^{m+1} \frac{d\psi_i^{(m)}}{dx} \, dx, \quad \text{and} \quad \zeta_{i+1/2}^{0} = 1.$$

For a list of the test functions $\psi_i^{(m)}$, the trial functions $\phi_i^{(m)}$, the quadrature points $\xi_{i+1/2}$, and the integrals $\zeta_{i+1/2}^{m+1}$; see Blom and Zegeling [1989]. In Skeel and Berzins [1990], a justification is given for all choices of the parameters and functions. There it is shown that the spatial discretization method is second-order accurate, both in the regular and the singular case.

The right boundary equation in (3.2), $\beta_j(x_R, t) R_j|_{x=x_R} = \gamma_j|_{x=x_R}$, is combined with the semidiscrete approximation of (4.2) with $i = N + 1$,

$$S_j^{(N+1)-1/2} \frac{x_R^{m+1} - \zeta_{N+1/2}^{m+1}}{m+1} + \zeta_{N+1/2}^{m-\mu} \xi_{N+1/2}^{\mu} R_j^{N+1/2} = x_R^m R_j|_{x=x_R}, \tag{4.7}$$

to eliminate $R_j|_{x=x_R}$.

In the regular case, the same procedure is applied to the left boundary equation in (3.2), $\beta_j(x_L, t)R_j|_{x=x_L} = \gamma_j|_{x=x_L}$ and

$$S_j^{0+1/2}\frac{\zeta_{1/2}^{m+1} - x_L^{m+1}}{m+1} - \xi_{1/2}^m R_j^{1/2} = -x_L^m R_j|_{x=x_L}, \tag{4.8}$$

from which we can eliminate $R_j|_{x=x_L}$. If x_L and m are both zero, we take $x_L^m = 1$. In the singular case, however, we distinguish two possibilities. If a Dirichlet condition is specified at $x = 0$ ($\beta = 0$), the left boundary equation $\gamma_j|_{x=x_L} = 0$ is imposed. If $\beta \neq 0$, we use just the semidiscrete approximation of (4.2) with $i = 0$, which gives

$$\frac{S_j^{0+1/2}}{m+1} - \xi_{1/2}^{-1} R_j^{1/2} = 0. \tag{4.9}$$

This means that for $X_1 \to x_L$ the boundary equation tends to the zero flux conditon $R_j|_{x=x_L} = 0$, which is a natural constraint for polar problems.

It is easy to derive that, for $m = 0$, the semidiscrete approximation of (4.1) as given by (4.6) is analogous to a semidiscretization by central differences.

Similar to Section 2, eqs. (4.6) and (2.6) are combined to obtain the stiff system of DAEs:

$$A(Y)\dot{Y} = L(Y). \tag{4.10}$$

In the next section, we discuss the interface that implements this spatial discretization.

5. THE INTERFACE

5.1 Choice of the Moving-Grid Parameters

The moving-grid method as described in Section 2 still has three parameters that should be specified by the user: the time-smoothing parameter τ, the spatial smoothing parameter κ, and the monitor regularizing parameter α. The choice of the parameters κ and α is not very critical; experiments have shown that a variation of these parameters does not result in a drastic change of performance. A wrong choice of τ can result in starting problems or grid lagging behind. A problem-dependent guidance of how to choose τ will be given below. It is felt, however, that the choice of the monitor function (e.g., dependent on solution curvature instead of solution variation) is probably more important than the value of τ (see also Verwer et al. [1989]). The monitor function (2.7), which is based on the first derivative of the solution, has appeared to be the most robust in a method of lines approach (cf., Blom and Verwer [1989]).

We now give an indication as how the parameters should be chosen.

Time-Smoothing Parameter τ, $\tau \geq 0$. It is obvious that the grid will not be adapted if τ tends to infinity. In fact, choosing τ very large allows the user to run the problem on a fixed grid. On the other hand, if $\tau = 0$ the equidistribution relation

$$\tilde{n}_i(t) = c(t)M_i(t) \tag{5.1}$$

will be solved. So, if the initial grid satisfies (5.1) at t_0, as is the case, for example, for a uniform grid and a constant or linear initial solution, τ can be chosen equal to zero (cf. the first two numerical examples in Verwer et al. [1989] and the examples in Section 6). If at t_0 (5.1) does not hold and if the initial grid has to be adapted, or if time smoothing is desired, a typical value for τ is 10^{-3}. Note, however, that τ should be related to the critical time scale of the problem.

Spatial Smoothing Parameter κ, $\kappa \geq 0$. An adjacent grid ratio of 1.5 (i.e., $\kappa = 2$) was found to be satisfying for all problems tested. It is a bit conservative and therefore demands an unnecessarily large number of grid points if the solution possesses extremely sharp spatial gradients. For less spatial smoothing, $\kappa = 1$ will suffice. The choice $\kappa = 0$, implying that there will be *no* spatial smoothing, is, in general, not recommended, since it can be shown, in the case $m = 0$, that the spatial discretization is equivalent to a central-difference approximation and that the reliability of such a discretization is dependent on a smooth grid distribution.

Monitor Regularizing Parameter α, $\alpha > 0$. This parameter is added to the monitor function to ensure that M_i is strictly positive. The choice $\alpha = 1$ results in the arclength monitor. Experiments reveal, however, that if the solution is very flat, a uniform grid will be used, because α dominates the arclength monitor. This can result in a slightly worse performance than on a nonuniform grid, especially in the case of evolving solutions. The choice $\alpha = 0.01$ results in a more sensitive adaptation of the grid and is still large enough to be used in case of solutions with steep gradients. Note, however, that it is assumed that both the variation in the solution and the interval $[x_L, x_R]$ are reasonably scaled; otherwise, the choice of α should be rescaled; for example, for scalar equations,

$$\alpha = 0.01\left(\frac{u_{\max} - u_{\min}}{x_R - x_L}\right)^2$$

to ensure its regularizing role.

5.2 How to Use the Interface

First, we give a description of the use of the interface with an arbitrary DAE solver. Then an elaborated example follows, showing how to use the interface in combination with the public-domain code DASSL [Brenan et al. 1989; Petzold 1983].

5.2.1 *General Description.* Figure 1 shows part of the documentation of the module that describes the use of the moving-grid discretization routine. For a description of the call of the initialization routine SETSKM, see Figure 2, which shows the header and the documentation of this routine. The header documentation of the residual routine SKMRES is shown in Figure 3.

5.2.2 *Example of Use with DASSL.* In this subsection we describe how to use the moving-grid interface with DASSL [Brenan et al. 1989; Petzold 1983],

```
C-----------------------------------------------------------------------
C
C How to use this module
C ----------------------
C 1.  Set NPDE = # PDEs to be solved.
C     Set NPTS = # mesh points to be used.
C                (NC=NPTS-2 is # internal points)
C     Set M for space coordinate type
C         = 0 for Cartesian,  = 1 for cylindrical, = 2 for spherical.
C     Specify a workspace of size at least  (NPDE+1)*NPTS+(6+NPDE)*NPDE
C     for use by the routine SKMRES which defines the DAE system being
C     solved by the integrator.
C
C     Call the initialization routine SETSKM, see the documentation at
C     the head of this routine for the precise details of the call.
C
C     Set TS and TOUT for start and end integration times.
C     Initialize data as required for time integration,
C                - see documentation of DAE solver.
C     Call the DAE solver with as residual routine SKMRES or an
C     enveloping routine to satisfy the header requirements.
C
C 2.  Provide a set of routines which describe the precise form of the
C     PDEs to be solved. Three routines must be provided and the names
C     of these routines are fixed. These routines are:
C        SPDEF    forms the functions C, Q and R of the PDE in a
C                 given x-point.
C        BNDR     forms the functions BETA and GAMMA associated with the
C                 boundary conditions for the PDE.
C        UINIT    supplies the initial values of the PDE part.
C                 An initial uniform grid is generated by SETSKM and
C                 provided in Y(NPDE+1,I), I=1,NPTS. If required, a user
C                 can redefine the mesh in a nonuniform way.
C     The headers of these routines are:
C
C      SUBROUTINE SPDEF (T, X, NPDE, U, DUDX, C, Q, R, IRES)
C      INTEGER NPDE, IRES
C      REAL T, X
C      REAL U(NPDE), DUDX(NPDE), C(NPDE,NPDE), Q(NPDE), R(NPDE)
C
C      SUBROUTINE BNDR (T, BETA, GAMMA, U, DUDX, UDOT, NPDE,
C     +                 LEFT, IRES)
C      INTEGER NPDE, IRES
C      LOGICAL LEFT
C      REAL T
C      REAL BETA(NPDE), GAMMA(NPDE),
C     +     U(NPDE), DUDX(NPDE), UDOT(NPDE)
C
C      SUBROUTINE UINIT (NPDE, NPTS, Y)
C      INTEGER NPDE, NPTS
C      REAL Y(NPDE+1,NPTS)
C
C-----------------------------------------------------------------------
```

Fig. 1. Documentation of the module describing the use of the moving-grid discretization routine.

the DAE solver from Petzold. DASSL and all other necessary source, for example, LINPACK [Dongarra et al. 1979] routines and the routine MACHAR [Cody 1988], were obtained from Netlib [Dongarra and Grosse 1987] (netlib@ornl.gov).

As mentioned before, the user should specify the problem description routines for the initial solution vector u^0 and the functions $C_{j,k}$, R_j, Q_j, β_j,

```
C-----------------------------------------------------------------------
C
      SUBROUTINE SETSKM (NEQN, NPDE, NPTS, XL, XR, TAU, KAPPA, ALPHA,
     +                   Y, RWK, NRWK, M, TS, IBAND, IRES)
C
C-----------------------------------------------------------------------
C Purpose:
C -------
C Initializing routine for moving-grid spatial discretization.
C
C Parameters:
C ----------
      INTEGER NEQN, NPDE, NPTS, NRWK, M, IRES
      REAL XL, XR, TAU, KAPPA, ALPHA, TS
      REAL Y(*), RWK(NRWK)
C
C NEQN  Exit: the size of the DAE system generated when the PDE +
C       the grid equations are discretized. This value is (NPDE+1)*NPTS.
C NPDE  Entry: the number of PDEs.
C NPTS  Entry: the number of spatial mesh points, including the
C       boundary points.
C XL    Entry: left boundary point.
C XR    Entry: right boundary point.
C TAU   Entry: time-smoothing parameter.
C       If the initial grid satisfies the grid equation with TAU=0 at
C       TS=0, TAU can be chosen equal to zero. If this is not the case
C       and if the initial grid has to be adapted, or if time-smoothing
C       is desired a typical value of TAU = 1E-3, but TAU should be
C       related to the time scale of the problem.
C KAPPA Entry: spatial smoothing parameter (REAL).
C       KAPPA = 2.0 was found to be satisfying for all problems tested.
C       For less spatial smoothing KAPPA = 1.0 will suffice.
C ALPHA Entry: monitor regularizing parameter.
C       ALPHA = 0.01 is recommended (for a well-scaled system of PDEs).
C Y     Exit: array of length >= (NPDE+1)*NPTS that contains the initial
C       (uniformly spaced) grid and the corresponding initial PDE
C       solution values. This array must be passed across as a one-
C       dimensional array of length NEQN to the DAE solver. This
C       array is ordered as
C          PDE comp.  : Y((NPDE+1)*L + J) L=0,...,NPTS-1,
C                                         J=1,...,NPDE
C          grid points: Y((NPDE+1)*(L+1)) L=0,...,NPTS-1.
C RWK   workspace of length NRWK for the residual routine SKMRES which
C       actually performs the semi-discretization of the PDEs and
C       defines the grid equations.
C NRWK  Entry: dimension of workspace RWK.
C       NRWK must be >= (NPDE+1)*NPTS + (6+NPDE)*NPDE.
C M     Entry: integer >= 0 which determines the coordinate system used.
C          0: Cartesian coordinates,
C          1: cylindrical polar coordinates,
C          2: spherical polar coordinates.
C TS    Entry: the time at which the integration starts.
C IBAND Exit: an upper bound on the half bandwidth of the Jacobian
C       matrix when this module is used. (If the DAE solver is called
C       with banded matrix routines this parameter should be
C       supplied to MATSET (SPRINT) or to DASSL (IWORK(1) and IWORK(2)).
C IRES  Exit: this parameter is set to -1 if an error is found by
C       this routine.
C
C-----------------------------------------------------------------------
```

Fig. 2. Documentation of the initialization routine SETSKM.

```
C-----------------------------------------------------------------------
C
      SUBROUTINE SKMRES (NEQN, T, Y, YDOT, RES, IRES, RWK, NRWK)
C
C-----------------------------------------------------------------------
C Purpose:
C -------
C Enveloping routine to compute the residual of the PDE and of the grid
C equations. SKMRES checks on node-crossing, partitions the workspace
C and calls CWRESU for the spatial discretization and the computation
C of the residual of the PDE in Lagrangian formulation and CWRESX for
C the spatial discretization and the residual computation of the grid
C equations.
C
C Parameters:
C ----------
      INTEGER NEQN, IRES, NRWK
      REAL T
      REAL Y(NEQN), YDOT(NEQN), RES(NEQN), RWK(NRWK)
C
C NEQN  Entry: the size of the DAE system generated when the PDE +
C       the grid equations are discretized.
C T     Entry: evaluation time.
C Y     Entry: array of length NEQN containing the DAE vector consisting
C       of the spatial mesh and the corresponding initial PDE solution
C       values at time T. This array is ordered as
C          PDE comp.  : Y((NPDE+1)*L + J) L=0,...,NPTS-1,
C                                         J=1,...,NPDE
C          grid points: Y((NPDE+1)*(L+1)) L=0,...,NPTS-1.
C RES   Exit: residual vector.
C       If IRES = -1 RES should contain only the part of the residual
C       dependent on the time-derivative, if IRES /= -1 RES should
C       contain the full residual  A*ydot - g.
C IRES  Entry: see above.
C       Exit: 2, if setup routine SETSKM has not been called.
C             3, if one of the DAE solutions in the vector Y is not
C                acceptable.
C RWK   working storage of length NRWK.
C NRWK  Entry: dimension of RWK. Should be >= NEQN + (6+NPDE)*NPDE.
C
C-----------------------------------------------------------------------
```

Fig. 3. Documentation of the residual routine SKMRES.

and γ_j, and, through the initializing routine SETSKM, the interval $[x_L, x_R]$, the number of grid points, the initial time t_0, the number m, the number of PDEs, and the method parameters τ, κ, and α. Next, the DAE solver should be called according to its specifications with (an enveloping routine of) SKMRES for the residual evaluation.

The easiest way to describe how the problem description routines should be written is by a simple example. Consider the following problem from electrodynamics, which can be found, for example, in Bakker [1977]:

$$u_t = \varepsilon p u_{xx} - g(u - v)$$
$$v_t = p v_{xx} + g(u - v)$$

$$\text{so} \quad m = 0, \quad \text{NPDE} = 2, \quad \text{and} \quad \begin{matrix} R_1 = \varepsilon p u_x, & Q_1 = g(u - v), \\ R_2 = p v_x, & Q_2 = -g(u - v), \end{matrix} \tag{5.2}$$

with

$$g(z) = e^{\eta z/3} - e^{-2\eta z/3},$$

$0 \le x \le 1$ and $0 \le t \le 4$; $\varepsilon = 0.143$, $p = 0.1743$, and $\eta = 17.19$.

The left boundary condition (LEFT = .TRUE.) is given by

$$u_x = 0 \quad \text{and} \quad v = 0 \quad \text{at} \quad x = 0 \quad (\beta_1 = 1, \gamma_1 = 0; \beta_2 = 0\ \gamma_2 = v),$$

the right boundary condition (LEFT = .FALSE.) is

$$u = 1 \quad \text{and} \quad v_x = 0 \quad \text{at} \quad x = 1 \quad (\beta_1 = 0, \gamma_1 = u - 1; \beta_2 = 1, \gamma_2 = 0),$$

and the initial conditions are

$$u = 1 \quad \text{and} \quad v = 0 \quad \text{at} \quad t = 0.$$

Note that Neumann boundary conditions can be put in the master equation (3.2) in two different ways. The first is to put $\beta = 0$ and $\gamma = u_x$, which results in equal solution values for the boundary point and its neighbor. The second, and in our opinion more natural, is setting $\beta = 1$ and $\gamma = 0$, as is done above. Now, the flux term R will be used, and the solution values will only be the same in the limit case.

The routines UINIT, SPDEF, and BNDR are listed in Figure 4. The component u of the PDE at the ith grid point is held as Y(1, i) in the package, and the component v as Y(2, i); the i-th grid point itself is stored in Y(3, i). The main program, which calls the initializing routine SKMRES and the DAE solver, is shown in Figure 5.

6. NUMERICAL EXAMPLES

The moving-grid interface has been applied to a wide class of problems. We used the moving-grid interface to solve the three numerical examples from Verwer et al. [1989], namely, a flame propagation model, a "hot spot" problem from combustion theory, and a semilinear hyperbolic system with, as solution, two waves traveling in opposite directions. The results were comparable with those in that paper, as could be expected since the spatial discretization method used in the interface results for problems in a Cartesian coordinate system ($m = 0$) in the central-difference method used in Verwer et al. [1989]. In Zegeling and Blom [1992], which is devoted to an evaluation of another moving-grid method, the gradient-weighted moving-finite-element method, we compared the interface with GWMFE among others, on Burgers' equation (convection-diffusion) with a smooth initial solution and a linear heat conduction problem with a shifting and oscillating pulse as solution. The application of the moving-grid interface to a class of 1-D brine transport problems is reported in van Eijkeren et al. [1991] and Zegeling et al. [1992]. In all of these problems, the method implemented in the moving-grid interface appeared to be robust and efficient. However, as shown also in Zegeling and Blom [1992], it has problems with solutions having discontinuous derivatives (resulting in smearing and/or oscillations), largely different monitor values in different parts of the domain (oscillations), or near shocks (small time steps caused by (temporary) node crossing). The addition of more grid points improves the

```
      SUBROUTINE UINIT (NPDE, NPTS, Y)
C
C Routine for PDE initial values.
C Entry:
C   Y(NPDE+1,i) = X_i; uniform mesh, generated by package
C Exit:
C   Y(NPDE+1,i) = X_i; mesh, optionally changed by user
C   Y(     k,i) = u_k(X_i,t0); initial value of k-th component
C                                    i = 1,.., NPTS
C
      INTEGER NPDE, NPTS
      REAL Y(NPDE+1,NPTS)
C
      INTEGER I

      DO 10 I = 1, NPTS
         Y(1,I) = 1.0
         Y(2,I) = 0.0
   10 CONTINUE

      RETURN
      END



      SUBROUTINE SPDEF (T, X, NPDE, U, DUDX, C, Q, R, IRES)
C
C Routine to describe the body of the PDE system.
C The PDE is written as
C   NPDE                 k                      -m   m
C   sum C  (x,t,u,u ) u  + Q (x,t,u,u )  = x   (x  R (x,t,u,u ) ) .
C   k=1  jk        x   t    j        x             j        x   x
C The functions C, Q and R must be defined in this routine.
C
      INTEGER NPDE, IRES
      REAL T, X
      REAL U(NPDE), DUDX(NPDE), C(NPDE,NPDE), Q(NPDE), R(NPDE)
C
      INTEGER J, K
      REAL EPS, ETA, GZ, P, Z
      DATA EPS /0.143/, ETA /17.19/, P /0.1743/
      DO 10 K = 1, NPDE
         DO 20 J = 1, NPDE
            C(J,K) = 0.0
   20    CONTINUE
         C(K,K) = 1.0
   10 CONTINUE

      Z  = U(1) - U(2)
      GZ = EXP(ETA*Z/3) - EXP(-2*ETA*Z/3)
      Q(1) =  GZ
      Q(2) = -GZ

      R(1) = EPS*P * DUDX(1)
      R(2) =     P * DUDX(2)

      RETURN
      END



      SUBROUTINE BNDR (T, BETA, GAMMA, U, DUDX, UDOT, NPDE,
     +                 LEFT, IRES)
```

Fig. 4. The routines UINIT, SPDEF, and BNDR for the example problem.

```
C
C Boundary conditions routine
C The boundary conditions are written as
C   BETA (x,t) R (x,t,u,u ) = GAMMA (x,t,u,u )
C       j       j        x        j        x
C The functions BETA and GAMMA should be defined in this
C routine.
C
      INTEGER NPDE, IRES
      LOGICAL LEFT
      REAL T
      REAL BETA(NPDE), GAMMA(NPDE),
     +     U(NPDE), DUDX(NPDE), UDOT(NPDE)
C
      IF (LEFT) THEN
         BETA (1) = 1.0
         GAMMA(1) = 0.0
         BETA (2) = 0.0
         GAMMA(2) = U(2) - 0.0
      ELSE
         BETA (1) = 0.0
         GAMMA(1) = U(1) - 1.0
         BETA (2) = 1.0
         GAMMA(2) = 0.0
      ENDIF

      RETURN
      END
```

Fig 4—*(Continued).*

performance (including the time-stepping behavior), but makes the method less efficient, of course.

Here we discuss the results obtained for two other problems, namely, the problem stemming from electrodynamics, which was also used as an example in the description of the usage of the interface, and a reaction-diffusion equation in cylindrical polar coordinates. We present our numerical results in plots, wherein an accurate reference solution is denoted by a solid line, while the marks show the grid distribution and the PDE approximation. The integrtion history is given by

—STEPS: total number of successful time steps,

—JACS: total number of Jacobian evaluations, and

—BS: total number of backsolves.

6.1 Problem I: A Problem from Electrodynamics

This problem is described in Section 5 (formula (5.2)). The steady-state problem has been discussed in Bus 1976, pp. 113–116. It is a singular perturbation problem, the solution of which first develops steep boundary layers very rapidly, whereas for later times a smooth stationary solution results (at approximately $t = 3.0$).

A reference solution has been computed using a fixed grid with 500 equidistant points and a time-tolerance value for the ODE solver of 10^{-7}. Bakker [1977] solved this problem on a fixed nonuniform grid using a grid spacig of 0.01 near the boundaries. We have solved it starting with a uniform

```
C -----------------------------------------------------------------------
C
      INTEGER MXNPDE, MXNPTS, MXNEQ, MXLIW, MXLRW, MXNRWK
      PARAMETER (MXNPDE = 2, MXNPTS = 101)
      PARAMETER (MXNEQ  = (MXNPDE+1) * MXNPTS)
      PARAMETER (MXLIW = 20+MXNEQ, MXLRW = (6*MXNPDE+20)*MXNEQ)
      PARAMETER (MXNRWK = MXNEQ+(6+MXNPDE)*MXNPDE)
      INTEGER INFO(15), IWORK(MXLIW), IPAR(1)
      REAL Y(MXNEQ), YPRIME(MXNEQ), RTOL(1), ATOL(1), RWORK(MXLRW),
     +     RWK(MXNRWK)
C
C   Y     : Grid and solution values
C   YPRIME: Derivative of Y
C   INFO  : Task_communication with DASSL
C   RTOL  : Relative tolerance for DASSL
C   ATOL  : Absolute tolerance for DASSL
C   RWORK : (Optional) REAL input values for DASSL
C   IWORK : (Optional) INTEGER input values for DASSL
C   RWK   : Workspace SKMRES
C
      INTEGER NPTS, NPDE
      COMMON /MOLIF/ NPTS, NPDE
C
C   NPTS  : # grid points                        -I (needed in
C   NPDE  : # partial differential equations     -I  residual routine)
C
      REAL XL, XR, T0, TE
C
C XL     Left boundary
C XR     Right boundary
C T0     Starting time
C TE     Final time
C
      EXTERNAL RESID, SETSKM, SDASSL
C
C -----------------------------------------------------------------------
C
      INTEGER I, IBAND, IDID, IRES, M, NEQ
      REAL ALPHA, KAPPA, TAU
C
C ccc Initialize problem parameters
      NPDE = 2
      XL   = 0.0
      XR   = 1.0
      T0   = 0.0
      TE   = 4.0
C
C ccc Initialize method parameters, grid, solution and derivative at T0
C     DASSL input
C
C   Method parameters
      NPTS  = 21
      M     = 0
      TAU   = 0.0
      KAPPA = 2.0
      ALPHA = 0.01
C
C   Call initialization routine SETSKM; determine initial grid;
C     store initial values of U in Y
      CALL SETSKM (NEQ, NPDE, NPTS, XL,XR, TAU, KAPPA, ALPHA,
     +             Y, RWK, MXNRWK, M, T0, IBAND, IRES)
      IF (IRES .EQ. -1) THEN
         STOP 'Error in SETSKM'
      ENDIF
```

Fig. 5. Main program, which calls the initializing routine SKMRES and the DAE solver.

```
C
C   Initial Yprime = 0
      DO 1 I = 1, NEQ
         YPRIME(I) = 0.0
    1 CONTINUE
C
C   Initialize DASSL input
      DO 5 I = 1, 15
         INFO(I) = 0
    5 CONTINUE
C    Both tolerances are scalars (default)
      ATOL(1) = 1E-3
      RTOL(1) = 1E-3
C    Banded Jacobian
      INFO( 6) = 1
      IWORK(1) = IBAND
      IWORK(2) = IBAND
C    Y, YPRIME probably inconsistent at T0
      INFO(11) = 1
C
C
C
C ccc DASSL loop
C     Call DASSL with as residual routine RESID, the enveloping routine
C     of SKMRES
      CALL SDASSL (RESID, NEQ, T0, Y, YPRIME, TE, INFO, RTOL, ATOL,
     +             IDID, RWORK, MXLRW, IWORK, MXLIW, RWK, IPAR, JAC)
C
           - - - - - - -
C
      SUBROUTINE RESID (T, Y, YPRIME, DELTA, IRES, RWK, IPAR)
      INTEGER IRES
      INTEGER IPAR(*)
      REAL T
      REAL Y(*), YPRIME(*), DELTA(*), RWK(*)
C
C Determine DAE system for DASSL
C   residual DELTA = A*YPRIME - G
C
C Entry:
C   T     : Current time
C   Y     : Current grid + solution
C   YPRIME: Time derivative of Y
C Exit:
C   DELTA : A*YPRIME - G
C   IRES  : -1, if user thinks solution is illegal or ico node crossing
C
C -----------------------------------------------------------------------
C
      INTEGER NPTS, NPDE
      COMMON /MOLIF/ NPTS, NPDE
      SAVE /MOLIF/
C
C
      EXTERNAL SKMRES
C
C -----------------------------------------------------------------------
C
      INTEGER NEQ, NRWK
C
      NEQ  = NPTS*(NPDE+1)
      NRWK = NEQ + (6+NPDE)*NPDE
C
```

Fig. 5—*(Continued).*

```
C ccc Call SKMRES with IRES=0 to compute total residual
      CALL SKMRES (NEQ, T, Y, YPRIME, DELTA, IRES, RWK, NRWK)
      IF (IRES .EQ. 2) THEN
         IRES = -2
         RETURN
      ELSE IF (IRES .EQ. 3) THEN
         IRES = -1
         RETURN
      ENDIF
C
      RETURN
      END
```

Fig. 5—*(Continued).*

grid with 21 points and using a time-tolerance value of 10^{-3} for DASSL. The moving-grid parameters were $\tau = 0$, $\kappa = 2$, and $\alpha = 0.01$.

Figure 6 shows a plot of the grid movement over the total time interval with a close-up at the start of the time integration, and two plots of the PDE solution components at times $t = 0.001$, 0.01, 0.1, and 4.0. At $t = 4.0$, the steady-state solution has been reached. The integration costs were STEPS = 87, JACS = 23, and BS = 185. For the integration from $t = 2.0$ to $t = 4.0$, only four steps were needed, which shows that the steady-state performance of the method is quite good.

6.2 Problem II: A Reaction-Diffusion Problem in Cylindrical Coordinates

Our second problem is the scalar reaction-diffusion equation in cylindrical coordinates ($m = 1$), which also served as an example in Part 2 of the SPRINT User's Manual [Berzins and Furzeland 1986],

$$T_z = \frac{1}{r}\frac{\partial}{\partial r}(\beta r T_r) + \gamma \exp\left(\frac{T}{1 + \varepsilon T}\right), \qquad 0 < r < 1 \qquad 0 \le z \le 1,$$

where $T(r, z)$ is the temperature in the cylinder and β, γ, and ε are given thermal properties. The axial direction, z, is treated as a timelike coordinate. The boundary conditions for the PDE are

$$T_r(0, z) = 0, \qquad T(1, z) = 0, \qquad z > 0,$$

and the initial solution is

$$T(r, 0) = 0, \qquad 0 \le r \le 1.$$

With the SPDIFF discretization interface from SPRINT, the problem has been solved using a uniform grid of 41 grid points, for $\beta = 0.1$, $\gamma = 1.0$, and $\varepsilon = 0.1$. We used the same problem parameters except for the diffusion parameter, which we took much smaller, $\beta = 10^{-4}$. The method parameters were the same as for the previous problem, that is, 21 grid points, a uniform initial mesh, a time tolerance of 10^{-3}, and the moving-grid parameters $\tau = 0$, $\kappa = 2$, and $\alpha = 0.01$. To compute the reference solution, we used 101 grid points and a time tolerance of 10^{-7}.

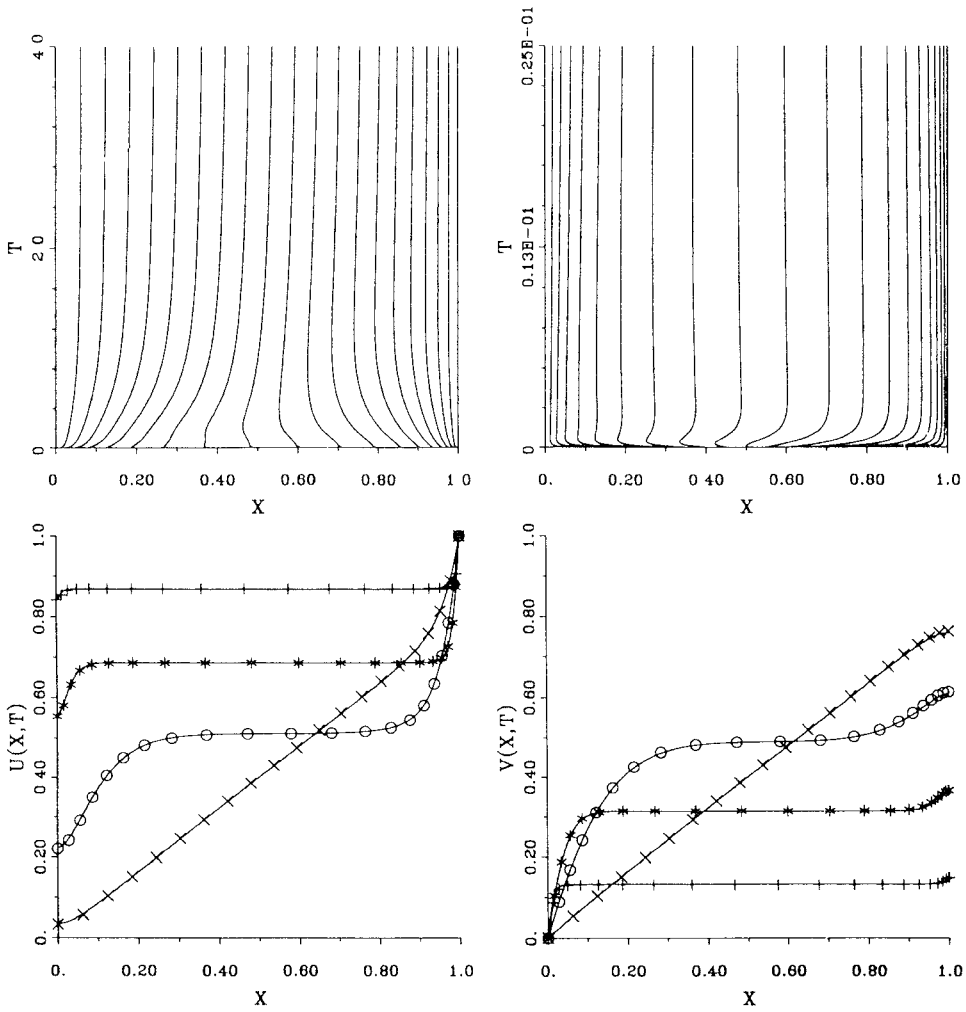

Fig. 6. Problem I (NPTS = 21). Grid with close-up near initial time and solution components at times $t = 0.001$, 0.01, 0 1, and 4 0 (+, ∗, ○, and ×, resp.).

The integration costs amounts to STEPS = 47 (25 to reach $z = 0.1$), JACS = 23, and BS = 130. In Figure 7 plots of the grid movement and of the PDE solution at $z = 0.4$, 0.6, 0.8, and 1.0 are shown, from which it is clear that, again, the grid is adapted rapidly enough to prevent inaccuracies in the approximation of spatial derivatives caused by a too course grid.

ACKNOWLEDGMENTS

We are grateful to Martin Berzins for the helpful discussion during his visit to CWI and for sending us the latest version of the code implementing Skeel

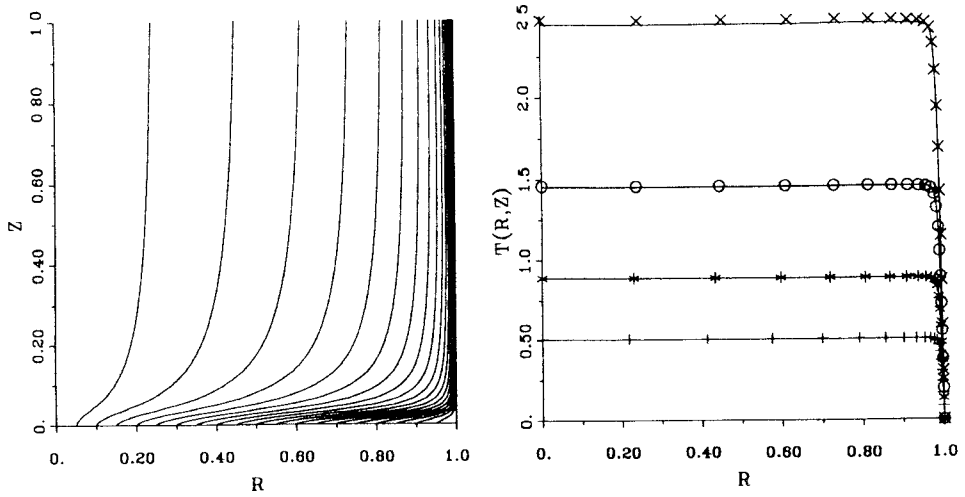

Fig. 7. Problem II (NPTS = 21). Grid and solution at z = 0.4, 0.6, 0.8, and 1.0 (+, *, O, and ×, resp.).

and Berzins's discretization on a fixed grid. We want to thank Jan Verwer for his interest and his guidance during this project.

REFERENCES

N. A. G. 1991. *NAG FORTRAN Library Manual, Mark 15*. N.A.G. Ltd, Oxford, U.K.

BAKKER, M. 1977. Software for semi-discretization of time-dependent partial differential equations in one space variable. Rep. NW 52/77, Mathematisch Centrum, Amsterdam, The Netherlands.

BERZINS, M., AND FURZELAND, R. M. 1985. A user's manual for SPRINT—A versatile software package for solving systems of algebraic, ordinary and partial differential equations: Part 1—Algebraic and ordinary differential equations. Rep. TNER.85.058, Thornton Research Centre, Shell Research Ltd., U.K.

BERZINS, M., AND FURZELAND, R. M. 1986. A user's manual for SPRINT—A versatile software package for solving systems of algebraic, ordinary and partial differential equations: Part 2—Solving partial differential equations. Rep. 202, Dept. of Computer Studies, Univ. of Leeds, U.K.

BERZINS, M., DEW, P. M., AND FURZELAND, R. M. 1989. Developing software for time-dependent problems using the method of lines and differential-algebraic integrators. *Appl. Numer. Math. 5*, 5, 375–397.

BLOM, J. G., AND VERWER, J. G. 1989. On the use of the arclength and curvature monitor in a moving-grid method which is based on the method of lines. Rep. NM-N8902, Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands.

BLOM, J. G. AND ZEGELING, P. A. 1989. A moving-grid interface for systems of one-dimensional time-dependent partial differential equations. Rep. NM-R8904, Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands.

BRENAN, K. E., CAMPBELL, S. L., AND PETZOLD, L. R. 1989. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. North-Holland, Amsterdam, The Netherlands.

BUS, J. C. P., ED. 1976. *Colloquium Numerieke Programmatuur, MC Syllabus 29.1a*. Mathematisch Centrum, Amsterdam, The Netherlands.

CODY, W. J. 1988. Algorithm 665: MACHAR: A subroutine to dynamically determine machine parameters. *ACM Trans. Math. Softw. 14*, 4 (Dec.), 303–311.

DONGARRA, J. J., AND GROSSE, E. 1987. Distribution of mathematical software via electronic mail. *Commun ACM 30*, 5 (May), 403–407.

DONGARRA, J. J., BUNCH, J. R., MOLER, C B., AND STEWART, G. W. 1979 *LINPACK User's Guide*. SIAM, Philadelphia, Pa.

DORFI, E. A., AND DRURY, L. O'C. 1987. Simple adaptive grids for 1-D initial value problems. *J. Comput. Phys. 69*, 1 (March), 175–195.

VAN EIJKEREN, J. C H., ZEGELING, P. A., AND HASSANIZADEH, S. M. 1991. Practical use of SPRINT and a moving-grid interface for a class of 1D non-linear transport problems. Rep. 959101001, National Institute of Public Health and Environmental Protection, Bilthoven, The Netherlands.

FURZELAND, R. M., VERWER, J. G., AND ZEGELING, P. A. 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*, 2 (Aug.), 349–388.

PETZOLD, L. R. 1983. A description of DASSL: A differential/algebraic system solver In *IMACS Transactions on Scientific Computation*, Stepleman, R. S., Ed North-Holland, Amsterdam.

SINCOVEC, R. F., AND MADSEN, N. K. 1975. Software for nonlinear partial differential equations *ACM Trans. Math. Softw. 1*, 3 (Sept.), 232–260.

SKEEL, R. D., AND BERZINS, M. 1990. A method for the spatial discretization of parabolic equations in one space variable. *SIAM J. Sci Stat. Comput 11*, 1 (Jan.), 1–32.

VERWER, J. G., BLOM, J. G , FURZELAND, R. M , AND ZEGELING, P. A. 1989. A moving-grid method for one-dimensional PDEs based on the method of lines. In *Adaptive Methods for Partial Differential Equations*, J. E. Flaherty, P. J. Paslow, M. S. Shephard, and J. D Vasilakis, Eds , SIAM, Philadelphia, Pa., 160–175.

ZEGELING, P. A., AND BLOM, J. G. 1992. An evaluation of the gradient-weighted moving-finite-element method in one space dimension. *J. Comput. Phys. 103*, 2 (Dec.), 422–441

ZEGELING, P. A., VERWER, J. G., AND VAN EIJKEREN, J. C. H. 1992. Application of a moving-grid method to a class of 1D brine transport problems in porous media. *Int. J. Numer. Methods Fluids 15*, 2 (July), 175–191.

Received April 1991; revised August 1992; accepted May 1993