New NAG Library Software for First-Order Partial Differential Equations

S. V. PENNINGTON and M. BERZINS
The University of Leeds

New NAG Fortran Library routines are described for the solution of systems of nonlinear, first-order, time-dependent partial differential equations in one space dimension, with scope for coupled ordinary differential or algebraic equations. The method-of-lines is used with spatial discretization by either the central-difference Keller box scheme or an upwind scheme for hyperbolic systems of conservation laws. The new routines have the same structure as existing library routines for the solution of second-order partial differential equations, and much of the existing library software is reused. Results are presented for several computational examples to show that the software provides physically realistic numerical solutions to a challenging class of problems.

Categories and Subject Descriptors: G.1.8 [**Numerical Analysis**]: Partial Differential Equations —*difference methods, hyperbolic equations, method of lines;* G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: NAG Fortran Library

1. INTRODUCTION

In recent years there have been a number of general-purpose software packages developed for the solution of systems of time-dependent partial differential equations (PDEs). Machura and Sweet [1980] gave a survey of such software, and since then there have been many additions. A common feature of a number of these software packages is that they use a method-of-lines approach in which sophisticated ordinary differential equation (ODE) initial-value software is used for the time integration of the semi-discretized PDEs. One example of a method-of-lines package is SPRINT, developed by Berzins et al. [1989], which offers a set of routines aimed at solving a wide range of one-dimensional, time-dependent PDEs. The modular structure of

Authors' address: School of Computer Studies, The University of Leeds, Leeds LS2 9JT, United Kingdom.
Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1994 ACM 0098-3500/94/0300-0063 $03.50

this package has enabled the inclusion of new methods in spatial discretization, time integration, and matrix algebra, and as such the package has evolved over time. SPRINT is used within both industrial and academic research. Versions of the software have been included in the NAG Fortran Library in the form of individual routines for the solution of second-order parabolic PDEs, with scope for coupled ordinary differential equations. Spatial discretization is by either a finite difference or a C0 collocation scheme, and there are a number of options regarding time integration and matrix algebra (see Berzins [1990]). Adaptive spatial remeshing is available for the finite-difference scheme in the Mark 16 (1993) release.

Although the aim of many method-of-lines packages has been to solve as wide a class of problems as possible (see, for example, Schryer [1990]), much of the available software in this area is most suited to the solution of second-order parabolic PDEs. However, many common physical problems are described by first-order PDEs, such as conservation laws, or by first- and second-order systems, such as the inviscid boundary layer equations. Systems of mixed order can be reduced to first order by the introduction of one or more new variables and are often more amenable in this form. Some first-order PDEs can be solved using spatial discretization schemes and software intended for higher-order problems, although care must be taken in the imposition of boundary conditions.

However, many first-order problems are hyperbolic in nature, and space-time characteristic behavior is important. Unless artificial dissipation is introduced via second-order terms, some type of one-sided or upwind differencing is essential for hyperbolic problems with steep gradients or shocks. The principal difficulty with upwind schemes is in determining the direction of information flow which may change in space or time, and hence it is generally inappropriate to specify or determine a fixed direction of upwinding before integration commences. Also, the imposition of boundary conditions for first-order hyperbolic PDEs must take into account the nature of the characteristics. As far as the authors are aware, there are many experimental codes but no widely available software packages which offer upwind differencing for a wide range of hyperbolic problems.

This paper describes two spatial discretization schemes for the solution of first-order PDEs to be included in the D03P subchapter of the NAG Fortran Library: (i) the central-difference Keller box scheme for the solution of general first-order problems and (ii) an upwind scheme for the solution of hyperbolic problems in conservation law form, based on the solution of a Riemann problem at each mesh point combined with a flux-limiter method. Routines for scheme (i) are included in the Mark 16 (1993) release of the NAG Library, and those for scheme (ii) are intended for the Mark 17 release. The new routines have the same structure and flexibility as the existing routines for the solution of second-order PDEs, reusing much of the existing SPRINT-based software. This paper will demonstrate that it is possible to provide software for a wide range of first-order systems within a method-of-lines framework.

2. PROBLEM CLASSES

The master equation form of the PDEs and coupled ODEs for the existing finite difference routines for second-order problems is

$$\sum_{j=1}^{\mathrm{NPDE}} P_{i,j} \frac{\partial u_j}{\partial t} + Q_i = x^{-m} \frac{\partial}{\partial x}(x^m R_i),$$

$$i = 1,2,\ldots,\mathrm{NPDE},\ a \le x \le b,\ t \ge t_0, \quad (1)$$

$$F_i\left(t, \underline{v}, \dot{\underline{v}}, \underline{\xi}, \underline{u}^*, \underline{u}_x^*, \underline{R}^*, \underline{u}_t^*, \underline{u}_{xt}^*\right) = 0, \qquad i = 1,2,\ldots,\mathrm{NCODE}, \quad (2)$$

where $P_{i,j}$ and R_i depend on x, t, $\underline{u}$, $\underline{u}_x$, and $\underline{v}$, and Q_i depends on $x, t, \underline{u}, \underline{u}_x, \underline{v}$ and linearly on $\dot{\underline{v}}$. The parameter m is an integer whose value depends on the coordinate system in use. The vector $\underline{u}(x,t)$ is the set of PDE solution values

$$\underline{u}(x,t) = [u_1(x,t),\ldots,u_{\mathrm{NPDE}}(x,t)]^T, \quad (3)$$

and $\underline{u}_x$ is its partial derivative with respect to x. The vector $\underline{v}(t)$ is the set of coupled ODE solution values

$$\underline{v}(t) = [v_1(t),\ldots,v_{\mathrm{NCODE}}(t)]^T, \quad (4)$$

and $\dot{\underline{v}}$ denotes its derivative with respect to time.

In the ODE part, given by (2), $\underline{\xi}$ represents a vector of coupling points at which the ODEs are coupled to the PDEs, and $\underline{u}^*$, $\underline{u}_x^*$, $\underline{R}^*$, $\underline{u}_t^*$, and $\underline{u}_{xt}^*$ are the functions $\underline{u}$, $\underline{u}_x$, $\underline{R}$, $\underline{u}_t$, and $\underline{u}_{xt}$ evaluated at these points. Each F_i may depend only linearly on time derivatives. Note that Eq. (2) may be an algebraic equation for some or all i, meaning that there are no time derivatives present. Unless otherwise stated, any references to ODEs should be taken to include such cases.

The boundary conditions have the form

$$\beta_i(x,t)R_i(x,t,\underline{u},\underline{u}_x,\underline{v}) = \gamma_i(x,t,\underline{u},\underline{u}_x,\underline{v},\dot{\underline{v}}), \qquad i = 1,2,\ldots,\mathrm{NPDE}, \quad (5)$$

where $x = a$ or b. The function γ_i may depend only linearly on $\dot{\underline{v}}$.

For the new Keller box scheme routines the PDE master equation includes the general problem class of the finite-difference scheme (Eq. (1)) minus the second-order term on the righthand side, to leave the general first-order system

$$\sum_{j=1}^{\mathrm{NPDE}} P_{i,j} \frac{\partial u_j}{\partial t} + Q_i = 0, \qquad i = 1,2,\ldots,\mathrm{NPDE},\ a \le x \le b,\ t \ge t_0, \quad (6)$$

where $P_{i,j}$ depends on x, t, $\underline{u}$, $\underline{u}_x$, and $\underline{v}$, and Q_i depends on $x, t, \underline{u}, \underline{u}_x, \underline{v}$ and linearly on $\dot{\underline{v}}$.

The ODE master equation is similar to that for the finite-difference scheme (Eq. (2)), but does not involve the function R which is not defined for the Keller box scheme, or the second-order term $\underline{u}_{xt}$, and so is given by

$$F_i\left(t, \underline{v}, \dot{\underline{v}}, \underline{\xi}, \underline{u}^*, \underline{u}_x^*, \underline{u}_t^*\right) = 0, \qquad i = 1,2,\ldots,\mathrm{NCODE}. \quad (7)$$

Since the box scheme leads naturally to an implicit ODE system (see Section 3.1.2), and there is no flux-type function R in the Keller box scheme master equation, the boundary conditions have a more general form than in the finite-difference scheme, given by

$$G_i^L(x,t,\underline{u},\underline{u}_t,\underline{v},\dot{\underline{v}}) = 0 \quad \text{at} \quad x = a, i = 1,2,\dots,\text{NLEFT}, \tag{8}$$

at the lefthand boundary, and

$$G_i^R(x,t,\underline{u},\underline{u}_t,\underline{v},\dot{\underline{v}}) = 0 \quad \text{at} \quad x = b, i = 1,2,\dots,\text{NRIGHT}, \tag{9}$$

at the righthand boundary, where NLEFT + NRIGHT = NPDE.

The functions G_i^L and G_i^R may depend only linearly on time derivatives. Also, G_i^L and G_i^R must not depend on $\underline{u}_x$, since boundary spatial derivatives are not determined explicitly in the Keller box scheme. If the problem involves derivative (Neumann) boundary conditions then it is often possible to restate such boundary conditions in terms of permissible variables. Alternatively, an ODE variable equal to the required derivative at the boundary can be introduced, and the boundary condition can be stated in terms of this variable.

For the new upwind-scheme routines the master equation for the PDEs again has the generality and flexibility of the finite-difference scheme problem class, but is modified slightly to be in conservation law form with a convective flux F_i and a source term S_i, that is

$$\sum_{j=1}^{\text{NPDE}} P_{i,j} \frac{\partial u_j}{\partial t} + \frac{\partial F_i}{\partial x} = S_i, \qquad i = 1,2,\dots,\text{NPDE}, \tag{10}$$

where $P_{i,j}$ and F_i depend on x, t, $\underline{u}$, and $\underline{v}$, and S_i depends on $x, t, \underline{u}, \underline{v}$ and linearly on $\dot{\underline{v}}$. Note that F_i does not depend on $\underline{u}_x$ as the equations would then be of second order and thus in the problem class of the existing finite-difference routines.

The ODE part is exactly as in the Keller box scheme, given by Eq. (7). The boundary conditions have a similar general form to those in the Keller box scheme given by Eqs. (8) and (9), but will be described in more detail in Sections 3.2 and 5.

3. ALGORITHM DESCRIPTION

The PDEs are solved using the method-of-lines (see Berzins et al. [1989]), in which the PDEs are discretized in space on a mesh x_i, $i = 1,2,\dots,\text{NPTS}$, using an appropriate spatial discretization method, resulting in a system of NPTS nonlinear coupled ODEs for each PDE. A number of these ODEs may be without a time derivative, in which case the system is of differential algebraic form. Algebraic equations often arise at the boundaries of the spatial domain where the PDEs are replaced by boundary conditions not involving time derivatives.

The system is of the form

$$\underline{A}(\underline{y},t)\dot{\underline{y}} = \underline{g}(\underline{y},t), \tag{11}$$

where the square matrix $\underline{A}$ may be singular, indicating a differential algebraic system. The solution vector $\underline{y}(t)$ is defined by

$$y_k(t) = u_j(x_i, t), \quad \text{where} \quad k = \text{NPDE} \times (i-1) + j, \tag{12}$$

for $i = 1, \ldots, \text{NPTS}$, $j = 1, \ldots, \text{NPDE}$, and

$$y_l(t) = v_m(t), \quad \text{where} \quad l = \text{NPDE} \times \text{NPTS} + m, \tag{13}$$

for $m = 1, \ldots, \text{NCODE}$.

The solution vector $\underline{y}(t)$ is thus made up of all the PDE variables at mesh point x_1, all the PDE variables at mesh point x_2 etc., followed by any coupled ODE variables, making a total of NPDE × NPTS + NCODE solution components.

The ODE (or differential algebraic) system given by Eq. (11), along with an initial condition $\underline{y}(0) = \underline{s}$ say, forms an initial-value problem which can then be integrated in time by standard methods (see Section 3.3).

3.1 Spatial Discretization

3.1.1 *Finite-Difference Scheme.* The spatial discretization methods for the new routines have some similarities to the finite-difference scheme of Skeel and Berzins [1990] used in the existing routines, and hence a brief summary is given here for future reference.

Consider for notational convenience the single general PDE

$$p\frac{\partial u}{\partial t} + q = \frac{\partial r}{\partial x}, \tag{14}$$

where p, q, and r are functions of x, t, u, and u_x.

The spatially discretized form of the above equation is

$$\frac{h_i p_{i-1/2} + h_{i+1} p_{i+1/2}}{h_i + h_{i+1}} \dot{u}_i + \frac{h_i q_{i-1/2} + h_{i+1} q_{i+1/2}}{h_i + h_{i+1}} = \frac{r_{i+1/2} - r_{i-1/2}}{1/2(h_i + h_{i+1})}, \tag{15}$$

where $\dot{u}_i$ denotes the time derivative of u_i, $h_i = x_i - x_{i-1}$, and the subscript $i + 1/2$ on p, q, and r denotes evaluation at $x = (x_i + x_{i+1})/2$, $u = (u_i + u_{i+1})/2$, and $u_x = (u_{i+1} - u_i)/h_{i+1}$ (and similarly for the subscript $i - 1/2$).

The Skeel and Berzins scheme is in fact a modified form of the Keller box scheme which discretizes second-order PDEs in such a way as to obtain an explicit ODE system, that is, each ODE involves just one time derivative (cf. Eq. (17)). This is achieved by the introduction of a second-order error. Details can be found in Skeel and Berzins [1990].

3.1.2 *Keller Box Scheme.* Keller [1970] introduced this finite-difference method for the spatial and temporal discretization of PDEs of second order (or higher) reduced to first order by the introduction of new variables.

Within the method-of-lines framework the box scheme is used to discretize the PDEs in space only. Spatial derivatives are replaced by central differences using mesh points x_i and x_{i+1}, for each i, and all variables and their time derivatives are averaged over these two mesh points.

For the single general PDE

$$p\frac{\partial u}{\partial t} + q = 0, \tag{16}$$

where p and q are functions of x, t, u, and u_x, the spatially discretized form is

$$p_{i+1/2}\frac{(\dot{u}_i + \dot{u}_{i+1})}{2} + q_{i+1/2} = 0, \tag{17}$$

where, as before, $\dot{u}_i$ denotes the time derivative of u_i, and the subscript $i + 1/2$ on p and q denotes evaluation at $x = (x_i + x_{i+1})/2$, $u = (u_i + u_{i+1})/2$, and $u_x = (u_{i+1} - u_i)/h_{i+1}$, where $h_i = x_i - x_{i-1}$.

Note that since Eq. (17) involves two time derivatives, the ODE system for this scheme is implicit (cf. the explicit form of Eq. (15)).

As an example consider the simple first-order system

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial v}{\partial x}, \tag{18}$$

$$v = \frac{\partial u}{\partial x}, \tag{19}$$

resulting from the reduction of the heat equation $\partial u/\partial t = \alpha\partial^2 u/\partial x^2$ to first order. The semidiscrete form of the system (18) and (19) is

$$\frac{\dot{u}_i + \dot{u}_{i+1}}{2} = \alpha\frac{v_{i+1} - v_i}{h_{i+1}}, \tag{20}$$

$$\frac{v_i + v_{i+1}}{2} = \frac{u_{i+1} - u_i}{h_{i+1}}. \tag{21}$$

The Keller box scheme routines described in this paper are not primarily intended for the solution of second-order problems reduced to first order. The existing NAG routines for second-order problems are generally more efficient as they solve second-order equations directly, thus involving fewer variables and function evaluations. Exceptions include higher or mixed-order problems (see problem 1.1 for example) for which the existing routines are unsuitable.

3.1.3 *Upwind Scheme.* The discretization scheme will be described for the single general PDE

$$p\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = s, \tag{22}$$

where p, f, and s are functions of x, t, and u.

The convective flux term $\partial f/\partial x$ is discretized using an upwind scheme based on Godunov's [1959] method involving the solution of a Riemann problem at each midpoint of the mesh, combined with a flux-limiter method (see LeVeque [1990], Chapter 17, and Osher [1985]). The two-dimensional form of this discretization method on quadrilateral meshes is analyzed in depth by Spekreijse [1987] and has been used on a variety of problems by Koren [1989] and others.

The source term s is treated as in the finite-difference method of Skeel and Berzins [1990] (see Section 3.1.1).

The spatially discretized form of Eq. (22) is very similar to that for the finite-difference scheme (Eq. (15)), that is,

$$\frac{h_i p_{i-1/2} + h_{i+1} p_{i+1/2}}{h_i + h_{i+1}} \dot{u}_i + \frac{f_{i+\frac{1}{2}} - f_{i-1/2}}{\frac{1}{2}(h_i + h_{i+1})} = \frac{h_i s_{i-1/2} + h_{i+1} s_{i+1/2}}{h_i + h_{i+1}}, \quad (23)$$

where $h_i = x_i - x_{i-1}$ as before, and $s_{i+1/2}$ is defined by

$$s_{i+1/2} = s\left(x_{i+1/2}, t, \tfrac{1}{2}(u_i + u_{i+1})\right), \quad (24)$$

where $x_{i+1/2} = (1/2)(x_i + x_{i+1})$.

Note that if u_i is viewed as an approximation to the average over the cell $[x_{i-1/2}, x_{i+1/2}]$ then the discretization is conservative, meaning that the time variation of u over the whole domain depends only on the flux at the boundaries—flux contributions at internal cell interfaces cancel (see LeVeque [1990], Chapter 12).

The upwinding occurs in the calculation of the flux terms in Eq. (23), given by

$$f_{i+1/2} = \hat{f}\left(x_{i+1/2}, t, u_L(x_{i+1/2}, t), u_R(x_{i+1/2}, t)\right), \quad (25)$$

where $\hat{f}(x, t, u_L(x,t), u_R(x,t))$ is the *numerical flux* function (see later) expressed in terms of the upwinded *Left* and *Right* values $u_L(x,t)$ and $u_R(x,t)$ at (x,t). These left and right values are calculated using standard upwinding techniques combined with a flux limiter (see for example LeVeque [1990]) suitably modified for a nonuniform mesh, so that

$$u_L(x_{i+1/2}, t) = u_i + \frac{h_{i+1}}{2} \frac{(u_i - u_{i-1})}{h_i} B(r_i), \quad (26)$$

$$u_R(x_{i+1/2}, t) = u_{i+1} - \frac{h_{i+1}}{2} \frac{(u_{i+2} - u_{i+1})}{h_{i+2}} B\left(\frac{1}{r_{i+1}}\right), \quad (27)$$

where the function B is some limiter function, which for this particular scheme is that proposed by Van Leer [1974] given by

$$B(r_i) = \frac{r_i + |r_i|}{1 + |r_i|}, \quad (28)$$

and

$$r_i = \frac{u^C_{i+1/2} - u_i}{u^U_{i+1/2} - u_i}, \quad (29)$$

where the superscripts C and U denote linearly interpolated centered and (left) upwind values respectively, given by

$$u^C_{i+1/2} = \tfrac{1}{2}(u_i + u_{i+1}), \tag{30}$$

$$u^U_{i+1/2} = u_i + \frac{h_{i+1}}{2}\frac{(u_i - u_{i-1})}{h_i}. \tag{31}$$

Alternatively r_i can be written in the form

$$r_i = \frac{(u_{i+1} - u_i)/h_{i+1}}{(u_i - u_{i-1})/h_i}, \tag{32}$$

which is the ratio of the derivatives of the centered and (left) upwind interpolants.

In the case of a system of equations the individual solution components are "limited" using Eqs. (26) and (27) to give a vector of left and right solution values at each midpoint, as in Spekreijse [1987] and Koren [1989].

Having obtained the left and right values of u at each midpoint $x_{i+1/2}$, the numerical flux $\hat{f}$ (a vector in the case of systems) is then to be calculated by some method. A simple way would be to average the left and right values of u and calculate the corresponding flux, that is,

$$\hat{f} = f(x, t, \tfrac{1}{2}(u_L + u_R)), \tag{33}$$

or to average the fluxes corresponding to the left and right values, that is,

$$\hat{f} = \tfrac{1}{2}(f(x, t, u_L) + f(x, t, u_R)). \tag{34}$$

Note that these two averages are generally not equal.

However, such averaging is equivalent to central differencing and hence does not take into account the nature of the space-time characteristics. The physically correct value for $\hat{f}$ is obtained from the solution of the Riemann problem given by

$$\partial u/\partial t + \partial f/\partial x = 0, \tag{35}$$

(where x is the distance to the midpoint $x_{i+1/2}$) with discontinuous initial values $u = u_L$ for $x < 0$ and $u = u_R$ for $x > 0$.

The numerical flux is required at every midpoint of the mesh at every time step. In theory these Riemann problems can be solved exactly (as in Godunov's [1959] original method), but to do so in practice would be computationally expensive, particularly for nonlinear equations which may require an iterative solution. Since u_L and u_R are approximate, and much of the information from the Riemann solution is subsequently lost by averaging, a much less expensive approximate solution is considered adequate (see LeVeque [1990] for example). Indeed Roe [1989] gives examples where an approximate Riemann solver gives *better* results than the exact solution.

Approximate Riemann Solvers. There are a number of methods for obtaining an approximate solution to the Riemann problem, due to Harten and Lax, Steger and Warming, Van Leer, Osher, and Roe among others. Van Leer et al. [1987] give references and brief descriptions of these methods and compare them for the Euler and Navier-Stokes equations, recommending Osher's [Engquist and Osher 1981; Osher and Solomon 1982] and Roe's [1981] schemes for their ability to accurately represent flow phenomena such as shocks, boundary layers, and contact discontinuities.

For Osher's approximate Riemann solver [Engquist and Osher 1981; Osher and Solomon [1982] the numerical flux is (omitting the dependence on x and t for notational convenience)

$$\hat{f}(u_L, u_R) = \tfrac{1}{2}(f(u_L) + f(u_R)) - \tfrac{1}{2}\int_{u_L}^{u_R} |A(u)|\, du, \tag{36}$$

where $A(u) = \partial f/\partial u$ is the Jacobian matrix, and $|A|$ is a matrix defined by

$$|A| = A^{+} - A^{-}, \tag{37}$$

where

$$A^{+} = P\Lambda^{+}P^{-1}, \qquad A^{-} = P\Lambda^{-}P^{-1} \quad \text{and} \quad A = P\Lambda P^{-1}, \tag{38}$$

where Λ is the diagonal matrix of eigenvalues of A, and Λ^{+} and Λ^{-} are diagonal matrices such that Λ^{+} has only the positive elements of Λ, and Λ^{-} only the negative, so that $\Lambda = \Lambda^{+} + \Lambda^{-}$. The columns of the matrix P are the right eigenvectors of A, and the rows of P^{-1} are the left eigenvectors of A.

The integration in Eq. (36) is carried out along a path piecewise parallel to the eigenvectors of $A(u)$, that is, along the wave paths in the phase space of u (see Engquist and Osher [1981] and Osher and Solomon [1982] for details). An example of the application of Osher's approximate Riemann solver is given in the Appendix.

For Roe's [1981] approximate Riemann solver the numerical flux is

$$\hat{f}(u_L, u_R) = \tfrac{1}{2}(f(u_L) + f(u_R)) - \tfrac{1}{2}\left|\bar{A}(u_L, u_R)\right|(u_R - u_L), \tag{39}$$

where $\bar{A}(u_L, u_R)$ is a linearized form (generally nonunique) of the Jacobian A satisfying the following:

(i) $\bar{A}(u_L, u_R)(u_R - u_L) = f(u_R) - f(u_L)$;
(ii) $\bar{A}(u, u) = A(u)$;
(iii) $\bar{A}$ has real eigenvalues with linearly independent eigenvectors.

$|\bar{A}|$ is defined as in Eqs. (37) and (38). Equation (39) can also be written as

$$\hat{f}(u_L, u_R) = \tfrac{1}{2}(f(u_L) + f(u_R)) - \tfrac{1}{2}\sum_{j=1}^{\mathrm{NPDE}} \alpha_j |\lambda_j| e_j, \tag{40}$$

where λ_j and e_j are the eigenvalues and (right) eigenvectors respectively of $\bar{A}$, and the α_j are obtained from

$$u_R - u_L = \sum_{j=1}^{\mathrm{NPDE}} \alpha_j e_j. \tag{41}$$

As a simple example of the numerical flux obtained using these schemes, consider Burgers' equation in the inviscid limit:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0. \tag{42}$$

Osher's scheme gives

$$\hat{f}(u_L, u_R) = \begin{cases} \frac{1}{2}u_L^2, & u_L, u_R > 0, \\ \frac{1}{2}u_R^2, & u_L, u_R < 0, \\ 0, & u_L \le 0 \le u_R, \\ \frac{1}{2}(u_L^2 + u_R^2), & u_L \ge 0 \ge u_R, \end{cases} \tag{43}$$

and Roe's scheme gives

$$\hat{f}(u_L, u_R) = \begin{cases} \frac{1}{2}u_L^2, & u_s > 0, \\ \frac{1}{2}u_R^2, & u_s \le 0, \end{cases} \tag{44}$$

where $u_s = (1/2)(u_L + u_R)$, the propagation speed of the discontinuity (u_L, u_R).

The calculation of the numerical flux is highly problem dependent and hence cannot be automatically performed in a general-purpose code such as this. The user is required to choose a Riemann solver and provide the code to calculate the numerical flux given any u_L and u_R. In many cases this is a simple task, with the calculation of the numerical flux being straightforward and involving only a few lines of code. More practical details are given in Section 5.

Source Terms. It should be noted that some problems with source terms are to be treated with caution. Recently, LeVeque and Yee [1990] studied a model linear advection equation with a parameter-dependent source term and found that for large source terms, stable and reasonable-looking solutions can be obtained which are in fact incorrect, with the discontinuity in the wrong location. The same problem has been solved here (see problem 2.2), with similar results. LeVeque and Yee showed that such incorrect speeds of propagation of discontinuities—typically one spatial mesh point per time step—are due to lack of spatial resolution in evaluating the source terms. A subsequent report by Griffiths et al. [1991] formalizes this work and proposes a scheme for which convergence of the numerical propagation speed to the true propagation speed is proven, but which can exhibit oscillations and divergence if certain monotonicity conditions on the source term are violated. However, their conclusions are limited in applicability, and the research is continuing.

Another issue concerning problems with source terms is whether it is generally appropriate to disregard the source term in the calculation of the numerical flux using an approximate Riemann solver. There have been a number of proposed methods for treating conservation laws with source terms (termed *inhomogeneous* conservation laws); Van Leer [1984] suggests that the associated Riemann problems at each midpoint of the mesh should have piecewise stationary initial distributions, rather than piecewise uniform as for the Riemann problems considered here. However, the method of solution is very complicated, and it is presented as a research tool from which more practical schemes may be derived. The methods proposed by Roe [1986] and Glaister [1988] apply only to the Roe [1981] approximate Riemann solver and involve additional terms in the numerical flux function. Sweby [1989] applies TVD (Total Variation Diminishing) schemes to inhomogeneous conservation laws utilizing a change in dependent variable to obtain a homogeneous PDE (no source term), but concludes that there is a large question remaining about the optimal treatment of the source term.

In conclusion, the above source term issues are still unresolved and are the subject of ongoing research. With the present algorithm, care should be taken when interpreting results for problems with source terms. In particular it is essential to employ a fine mesh for problems with propagating discontinuities and to check for the occurrence of nonphysical propagation speeds by comparing results for different mesh sizes. When possible, it may be advisable to avoid source terms by a change of variable (as in Sweby [1989]).

3.2 Boundary Conditions

First-order problems generally require just one boundary condition for each PDE variable, compared with second-order parabolic problems which require a boundary condition at *both* boundaries. The position of the boundary conditions should be chosen with care in order to be consistent with the flow of information in and out of the solution domain, represented by the direction of the space-time characteristics at the boundaries.

Consider the simple system

$$\frac{\partial \underline{u}}{\partial t} + \underline{A}\frac{\partial \underline{u}}{\partial x} = 0, \tag{45}$$

where

$$\underline{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \qquad \underline{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad 0 \le x \le 1. \tag{46}$$

The characteristic variables are u_1 and u_2 associated with the characteristics defined by $dx/dt = 1$ and $dx/dt = -1$ respectively. At the boundary $x = 0$, the characteristic for u_1 points into the domain, and hence information concerning u_1 is carried into the solution domain at the boundary. A boundary condition is therefore required for u_1 at $x = 0$. The characteristic for u_2 points out of the domain at $x = 0$, and so information about u_2 at this boundary is transported from inside the domain. Hence no condition for u_2 alone should be imposed at this boundary. Note however that any combination of u_1 and u_2 can be specified at the boundary provided that the coefficient of the u_1 term is nonzero. The same applies at $x = 1$, with u_1 and u_2 interchanged.

In the Keller box scheme the discrete form of the PDEs is evaluated at each midpoint of the mesh. The user specifies the boundary conditions in the form of Eqs. (8) and (9) at the appropriate boundary points. The total number of boundary conditions must be equal to the number of PDEs in order to give the required number of equations for the solution values at all mesh points. Note that the general form of the boundary conditions allows the inclusion of conditions which are not just simple specifications of boundary values for each variable. For example, the heat equation problem given by Eqs. (18) and (19) will generally require a boundary condition for u at both ends of the domain (and none for v), which can be easily handled using the interface adopted here.

In the upwind scheme the discrete equations are evaluated at each interior mesh point. The boundary conditions are supplied by the user at the appropriate boundary points as in the Keller box scheme above. However, an equation is required by the scheme for each PDE at *both* boundary points. The discrete equations cannot be evaluated at boundary points since information is required from outside the domain. This situation arises in many discretization schemes (see Yee et al. [1982] for example), and the extra information required by the scheme is supplied in the form of *numerical* boundary conditions (as opposed to the physical boundary conditions required by the continuous problem).

In order to be consistent with the characteristic directions, the numerical boundary conditions must be derived from information already present. A common method is to extrapolate the variables from the interior of the domain. For example, when solving the problem given by Eqs. (45) and (46) using the upwind scheme, a numerical boundary condition will be required for u_2 at $x = 0$ (and for u_1 at $x = 1$). Linear extrapolation gives (for a uniform mesh of size h)

$$u_2(0,t) = 2u_2(h,t) - u_2(2h,t). \tag{47}$$

In general, however, the characteristic variables—usually linear combinations of the original variables—should be extrapolated to obtain the numerical boundary conditions (see Gottlieb et al. [1982] for example). In the simple problem above the characteristic variables are the same as the original variables, whereas for the same problem with

$$\underline{A} = \begin{bmatrix} -\frac{1}{2} & -1 \\ -1 & -\frac{1}{2} \end{bmatrix} \tag{48}$$

the characteristic variables are $u_1 - u_2$ and $u_1 + u_2$, with $u_1 + u_2$ being associated with the outgoing characteristic at the lefthand boundary defined by $dx/dt = -3/2$. The numerical boundary conditions should be given as

$$u_1(x_1,t) + u_2(x_1,t) = (u_1 + u_2)^E = u_1^E + u_2^E, \tag{49}$$

where the superscript E denotes the linearly extrapolated value as in Eq. (47). Since $u_1(x_1, t)$ is a known physical boundary condition, Eq. (49) reduces to a numerical boundary condition for u_2. The numerical boundary condition at the righthand boundary is derived similarly from the characteristic variable $u_1 - u_2$.

In summary, for the Keller box scheme exactly one (physical) boundary condition is required for each PDE variable, chosen in accordance with the space-time characteristics at the boundaries; and for the upwind scheme two boundary conditions (generally one physical and one numerical) must be provided for each variable. The suggested method of obtaining numerical boundary conditions is to extrapolate the characteristic variables as described above. The flexibility of the upwind-scheme routines allows the use of other methods such as the solution of the characteristic equations associated with outgoing characteristic variables (see problem 2.4 for example). This flexibility also makes it possible to solve nonstandard problems for which there are either none or two physical boundary conditions for each variable (for example, the equation $\partial u/\partial t \pm x\partial u/\partial x = 0$ on the domain $-1 \leq x \leq 1$).

3.3 Time Integration

Full details of the time integration algorithms used in the D03P subchapter of the NAG library can be found in Berzins [1986], Berzins and Furzeland [1992], and Berzins et al. [1987; 1989]. In summary, there are two methods available: the Theta method (see Berzins and Furzeland [1992]) and the Gear backward differentiation (b.d.f.) formula up to order 5 (see Berzins [1986]). Both are implicit variable step-size methods in which the time step is chosen automatically throughout the integration to satisfy a user-specified local error tolerance. The b.d.f. algorithm provides a high-accuracy method in which the order of the formula is automatically varied up to a maximum of 5. The Theta method is only first-order accurate but may be as, or more, efficient for large systems of ODEs (resulting from many PDEs and/or very fine meshes) and coarse tolerances in method-of-lines applications (see Berzins and Furzeland [1992]).

Both the Theta and b.d.f. methods use either (modified) Newton iteration or functional iteration to solve the nonlinear equations at each time step. The type of iteration can be specified beforehand and fixed, or the algorithm can be allowed to switch between the two methods for efficiency. When using functional iteration, care must be taken to ensure that the ODEs are in normal form, meaning that the matrix $\underline{A}$ in Eq. (11) should be the identity matrix.

Although the Theta method is implicit, the use of functional iteration means that only the evaluation of the ODE residuals is required—no large systems of equations need to be solved. The computational cost is thus comparable with that of an explicit method. Hence in many cases it is the preferred solution method.

One of the salient features of hyperbolic PDEs is that the time step of explicit time integration methods is limited by the well-known Courant CFL condition in order to maintain stability, that is,

$$\frac{\Delta t \lambda}{\Delta x} < Cn, \tag{50}$$

where λ is the modulus of the maximum eigenvalue of the Jacobian matrix, and Cn is the Courant number, equal to 1 for the explicit forward Euler method with a three-point spatial discretization scheme (see LeVeque [1990] for example).

Yee [1987] proves that a similar stability condition holds for the Theta method combined with discretization methods such as those described here. This condition is

$$\frac{\Delta t \lambda}{\Delta x} < \frac{2}{3(1 - \theta)}, \tag{51}$$

where θ is the parameter used in the Theta method.

The functional iteration convergence requirement restricts the allowed time step in much the same way as a finite-stability region does for an explicit method (see Berzins and Furzeland [1992]). Berzins [1991] shows for a simple advection equation that the functional iteration convergence condition (with convergence rate equal to 0.5) is a more stringent requirement on the time step than Eq. (51). Thus if the functional iteration converges then the time step will automatically satisfy Eq. (51), and so the method will be stable.

The trade-off between accuracy and stability is a research issue of current interest. The approach employed here is to control the local error in time in a standard way but to allow the user to impose a CFL condition (if necessary) by including an option to limit the time step employed. In the testing of the software it was found that the limit was not often reached since controlling the local error so that the spatial error dominates generally resulted in CFL numbers less than 1. A future approach may be to control automatically the global time error so that the spatial error dominates (see Berzins [1991]).

3.4 Adaptive Spatial Remeshing

In order to obtain an accurate solution using as few mesh points as possible, it is often desirable to automatically adapt the mesh to follow the time-dependent nature of the solution. This is particularly useful for the solution of problems such as the variable-width boundary layer equations, where ideally the mesh should be very fine in the boundary layer region where the solution gradient is highly nonuniform, whereas a coarser mesh is adequate in the remainder of the domain.

The NAG library contains optional routines from the SPRINT package [Berzins et al. 1989] which employ a discrete time step method developed by Furzeland [1985]. These routines automatically create a new mesh based on the current solution profile at certain time steps; and the solution is then interpolated onto the new mesh, and the integration continues. A new mesh is calculated in order to equally distribute the integral of some function over the domain, so that the mesh spacing is inversely proportional to the function value. A typical choice for this so-called "monitor function" is the second space derivative of the solution value at each point, resulting in refinement in regions where the solution gradient changes most rapidly. This monitor function is used in the boundary layer equations example in Section 6. There are various other remeshing criteria such as adjacent mesh ratios and the difference between consecutive meshes.

The remeshing process is quite computationally expensive, and for some problems the saving made on the reduction of mesh points may be outweighed by the additional computations required for remeshing. The efficiency of the remeshing process depends on the choice of the monitor function and other remeshing criteria and on the frequency of the updates. The latter should be chosen carefully so that the mesh keeps up with the evolving solution while avoiding unnecessary updates.

As discussed by Furzeland [1985], discrete-time remeshing may not be particularly suitable for hyperbolic problems with shocks, since the optimal mesh should follow the space-time characteristic behavior. Ideally a quasi-Lagrangian method with a continuously moving mesh is required. Furzeland et al. [1990] discuss a number of such approaches in a method-of-lines framework and find that these are not without their own difficulties, particularly with regard to providing robust software for a broad class of problems. The fixed-mesh approximate Riemann solver approach adopted here appears to offer a viable alternative, though at the expense of requiring the user to provide a solution to the Riemann problem.

An additional problem mentioned by Furzeland [1985], and discussed in more detail by Swartz [1987], concerns the CFL stability condition for hyperbolic problems described in the last section. By design, adaptive remeshing will result in very small mesh spacing in part(s) of the solution domain, and hence very small time steps may be required to avoid instabilities. It may be possible to avoid this by the use of implicit time integration methods or localized time stepping, but it is not clear that this approach would be beneficial. In testing the present software, it has proved difficult to obtain better results for hyperbolic problems in terms of accuracy per unit of computational time by using the adaptive-remeshing option. Nevertheless, for problems in which accurate representation of shocks is essential, the remeshing option can be useful provided care is taken in the choice of monitor function and remeshing parameters. Spatial derivatives tend to infinity around a shock, and so they are generally unsuitable for the monitor function specification, leading to very narrow regions of extreme refinement.

An alternative monitor function based on a cosine function has proved successful for problems with a single shock. Each time remeshing is to occur, the shock is located by finding the maximum spatial derivative of the solution. Having previously decided on the desired width of the region of refinement, the monitor function is assigned using a cosine function centered on the shock so that it has a maximum value at the shock and then decreases smoothly down to zero at either side. Thus the subsequent refinement is limited in degree, and large enough in extent to ensure that the shock does

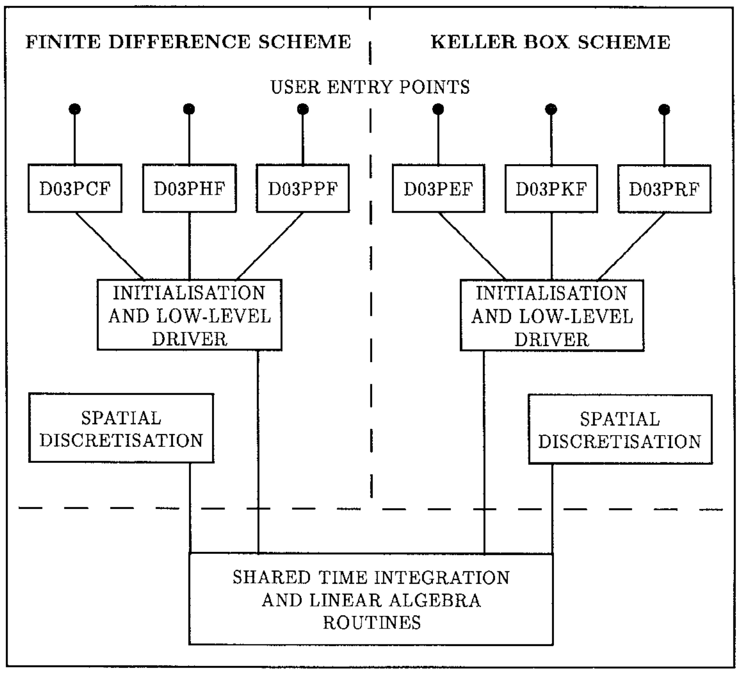

Fig. 1. Structure of existing finite-difference scheme and new Keller box scheme routines

not move out of the refined region between remeshing. Details and results can be found in the NAG Library Manual (Mark 17) (see Numerical Algorithms Group [1993]).

4. SOFTWARE STRUCTURE

The new routines have a direct correlation with the existing NAG Library routines for the solution of second-order PDEs described in Berzins [1990], and much of the existing software is reused. In Figure 1 the general structure is shown for the existing finite-difference scheme routines along with those for the new Keller box scheme. (The structure and names of the routines differ slightly from those in Berzins [1990], which appeared before the software was finalized.) For clarity the existing C0 collocation scheme routines and the new upwind-scheme routines are not included, but they have exactly the same form.

There are three new routines for the Keller box scheme:

(1) D03PEF—an easy-access routine for the solution of systems of first-order PDEs on a fixed mesh, with b.d.f. time integration and banded matrix algebra only (equivalent to the existing finite-difference routine D03PCF);

(2) D03PKF—a flexible routine for the solution of systems of first-order PDEs (with optional coupled ODEs) on a fixed mesh, with options regard-

ing time integration methods and parameters, and full, banded, or sparse matrix algebra (equivalent to the existing finite-difference routine D03PHF);

(3) D03PRF—the same as D03PKF but with the added option of adaptive spatial remeshing (equivalent to the finite-difference routine D03PPF included at Mark 16).

At the time of publication, the upwind-scheme routines are provisionally named D03PFF, D03PLF, and D03PSF, with the same functionality as the Keller box scheme routines above.

5. USER INTERFACE

The user interfaces for the new routines are very similar to those for the existing NAG routines for second-order PDEs (full details can be found in the NAG Library Manual [Numerical Algorithms Group 1993]). The user writes a Fortran program to set the required parameters and options and to call the appropriate NAG routine. This calling program must contain several simple subroutines which specify the problem; the names of these subroutines are passed as arguments to the NAG routine. For the new and existing routines, subroutines are required to describe the PDE(s), the boundary conditions, the ODE(s) (if applicable), and in the case of the adaptive remeshing routines, the initial conditions and the user-specified monitor function. The only significant change from the earlier routines is that the upwind scheme requires an additional subroutine to evaluate the solution of the Riemann problem.

The master equations for the PDEs and coupled ODEs are given for the existing and new routines in Section 2. In the existing finite-difference routines the user is required to evaluate specific functions in the PDE master equation, that is, the functions $P_{i,j}$, Q_i, and R_i in Eq. (1); whereas in the Keller box scheme routines the user evaluates the lefthand side of Eq. (6), given values of $\underline{u}$, $\underline{u}_x$, $\underline{u}_t$, $\underline{v}$, and $\dot{\underline{v}}$ at the midpoints of the mesh, that is, the function G_i given by

$$G_i = \sum_{j=1}^{\text{NPDE}} P_{i,j} \frac{\partial u_j}{\partial t} + Q_i, \qquad i = 1, 2, \ldots, \text{NPDE}. \tag{52}$$

Similarly, for the boundary conditions the user is required to evaluate the functions G_i^L and G_i^R in Eqs. (8) and (9).

However, the precise specification of the PDEs and the boundary conditions for the Keller box scheme depends on the value of one of the arguments on entry to the user-supplied subroutines. The value of this integer argument IRES determines the terms to be included in the evaluation of G_i. If IRES $= -1$ on entry then only those terms which depend explicitly on time derivatives should be included, whereas if IRES $= 1$ then **all** terms should be included.

As an example, consider the system (18) and (19) in Section 3.1.2. The PDE description subroutine required for the solution of this problem using the routine D03PEF is

```
      SUBROUTINE PDEDEF(NPDE, T, X, U, UDOT, UX, RES, IRES)
C  ..Scalar Arguments..
      DOUBLE PRECISION T, X
      INTEGER          IRES, NPDE
C  . Array Arguments..
      DOUBLE PRECISION RES(NPDE), U(NPDE), UDOT(NPDE), UX(NPDE)
C  ..Scalars in Common..
      DOUBLE PRECISION ALPHA
C  ..Common Blocks..
      COMMON           /AL/ALPHA
C  ..Executable Statements..
      IF (IRES.EQ.-1) THEN
         RES(1) = UDOT(1)
         RES(2) = 0.0D0
      ELSE
         RES(1) = UDOT(1) - ALPHA*UX(2)
         RES(2) = U(2) - UX(1)
      END IF
      RETURN
      END
```

For the upwind scheme, the specification of the PDEs is similar to that for the finite-difference scheme, in that the user supplies the functions $P_{i,j}$ and S_i, evaluated at midpoints of the mesh. However, the flux function F_i is **not** evaluated in the PDE subroutine and indeed is not required explicitly by the routines. Only the numerical flux at the midpoints of the mesh is needed (see Section 3.1.3) and is evaluated separately in the user-supplied approximate Riemann solver subroutine, given left and right solution values. The form of this subroutine is given below for the Burgers equation example described in Section 2.1.2 for the case $\hat{f} = (1/2)u_L^2$.

```
      SUBROUTINE RMFLUX(T, X, NPDE, ULEFT, URIGHT, RFLUX)
C  ..Scalar Arguments..
      DOUBLE PRECISION T, X
      INTEGER          NPDE
C  ..Array Arguments..
      DOUBLE PRECISION RFLUX(NPDE), ULEFT(NPDE), URIGHT(NPDE)
C  ..Executable Statements..
      RFLUX(1) = 0.5D0*ULEFT(1)**2
      RETURN
      END
```

A more complicated example (for the Euler equations) is given in the Appendix.

The boundary conditions for the upwind scheme are given in the form

$$G_i^L(x, t, \underline{u}, \underline{v}, \underline{\dot{v}}) = 0 \quad \text{at} \quad x = a,\ i = 1, 2, \ldots, \text{NPDE}, \tag{53}$$

at the lefthand boundary, and

$$G_i^R(x,t,\underline{u},\underline{v},\underline{\dot{v}}) = 0 \quad \text{at} \quad x = b, i = 1,2,\ldots,\text{NPDE}, \tag{54}$$

at the righthand boundary.

It should be noted that spatial derivatives at the boundary are not included in the arguments of the boundary condition subroutine, since they are not calculated explicitly in this scheme. The user may include derivatives in the boundary conditions by calculating and expressing them in terms of the variables at and adjacent to the boundary (see problem 2.4 for example). However, this may lead to instabilities if the one-sided differencing opposes the characteristic direction at the boundary. Alternative methods of handling derivative boundary conditions include replacing the derivative terms with permissible variables (if possible) or introducing an ODE variable equal to the required derivative at the boundary and expressing the boundary condition in terms of this new variable.

As an example of the user-supplied boundary condition subroutine, consider the second problem in Section 3.2 given by Eqs. (45) and (48), with a physical boundary condition for u_1 at each boundary and a numerical boundary condition at each boundary derived from the linearly extrapolated characteristic variables. The subroutine is

```
      SUBROUTINE BNDARY(NPDE, NPTS, T, X, U, IBND, RES, IRES)
C     ..Scalar Arguments..
      DOUBLE PRECISION T
      INTEGER          IBND, IRES, NPDE, NPTS
C     ..Array Arguments..
      DOUBLE PRECISION RES(NPDE), U(NPDE, NPTS), X(NPTS)
C     ..Local Scalars..
      DOUBLE PRECISION UBC1, UBC2, UEX1, UEX2
C     ..Executable Statements..
      IF (IBND.EQ.0) THEN
C     ..Lefthand boundary..
C     ..Extrapolate variables to boundary..
        UEX1 = 2.0D0*U(1,2) - U(1,3)
        UEX2 = 2.0D0*U(2,2) - U(2,3)
        UBC1 = .......(Left-hand physical b.c. for U(1))
      RES(1) = U(1,1) - UBC1
      RES(2) = U(2,1) + UBC1 - UEX1 - UEX2
      ELSE
C     ..Righthand boundary..
C     ..Extrapolate variables to boundary..
        UEX1 = 2.0D0*U(1,NPTS-1) - U(1,NPTS-2)
        UEX2 = 2.0D0*U(2,NPTS-1) - U(2,NPTS-2)
        UBC2 = .......(Right-hand physical b.c. for U(1))
        RES(1) = U(1,NPTS) - UBC2
        RES(2) = U(2,NPTS) - UBC2 + UEX1 - UEX2
      END IF
      RETURN
      END
```

6. EXAMPLES

The examples described in this section demonstrate that the software is capable of solving a broad class of first-order PDE systems, including both standard and nonstandard problems. Problems with moving boundaries or nonlocal boundary conditions are handled quite simply by the introduction of coupled ODEs. The results illustrate how the upwind scheme provides accurate and oscillation-free solutions to problems with shocks and discontinuities.

Unless stated otherwise, the given results required between 10 and 30 seconds of CPU time on an SGI 4D240 (single-processor) workstation.

6.1 Keller Box Scheme

Problem 1.1. The boundary layer equations describing inviscid flow over a semi-infinite plate, in first-order form:

$$\frac{\partial u}{\partial t} = -\frac{\partial v}{\partial x}, \tag{55}$$

$$u\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + \frac{\partial w}{\partial x}, \tag{56}$$

$$w = \frac{\partial u}{\partial x}, \tag{57}$$

for $0 \le x \le \infty$ and $t \ge 0$. The boundary conditions are $u(0,t) = v(0,t) = 0$ and $u(\infty, t) = 1$ for $t \ge 0$, and the initial conditions are $u(x,0) = 1$, $v(x,0) = 0$, and $w(x,0) = 0$ for $x \ge 0$. Note that the independent variable t in this problem is not time but a second space variable; at any value of t, the spatial domain $0 \le x \le \infty$ is the line perpendicular to the plate at a distance t along the plate, with u and v being the velocities parallel and perpendicular to the plate respectively. The solution becomes constant at around $t = 5$, meaning that downstream of this point on the plate there is no further change in the solution. Note that the semi-infinite domain must be truncated at some finite value of $x = xmax$ say, and with a simple coordinate transformation the domain can be taken to be $0 \le x \le 1$.

An adaptive mesh is appropriate for this problem as the solution gradient changes rapidly at the edge of the boundary layer, and the thickness of the boundary layer changes with time (i.e., distance downstream). Also, since there are three PDEs, a reduction in the number of mesh points should have a significant effect on the CPU time required. The second spatial derivative of u is used as the monitor function. Note that remeshing is performed only after the initial discontinuity in u has been smoothed somewhat (at about $t = 0.2$); until then the initial mesh is chosen to be finer towards the plate to provide adequate resolution at the start.

Figures 2 and 3 show the approximate solutions for u and v (the extra variable w is not shown) at $t = 5.0$, that is, at a distance 5.0 along the plate, with $xmax = 100$. An adaptive mesh of 61 points is used, with the b.d.f. method of time integration, Newton iteration, and banded matrix algebra.

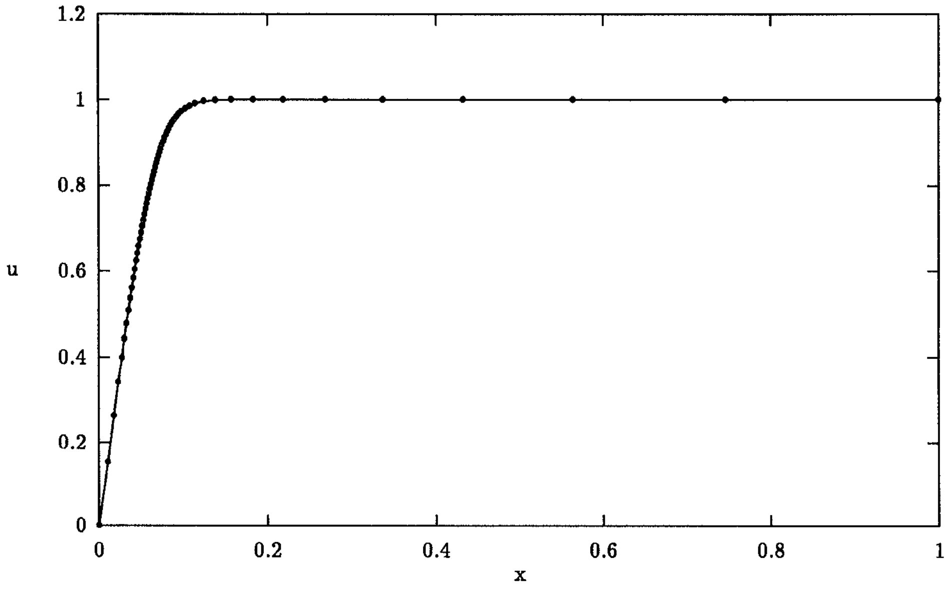

Fig. 2. The approximate solution for u in Problem 1.1 at $t = 5.0$ using an adaptive mesh of 61 points.

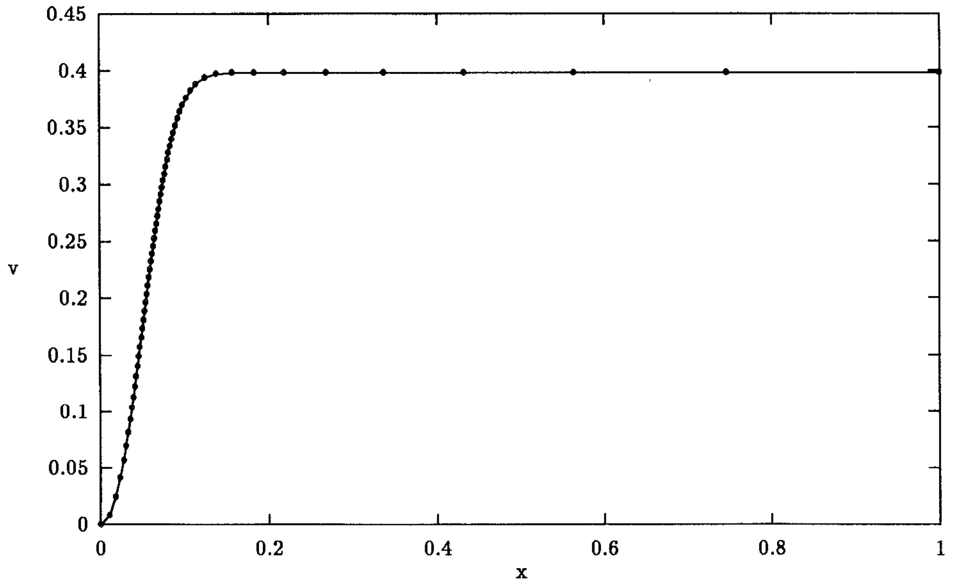

Fig. 3. The approximate solution for v in Problem 1.1 at $t = 5.0$ using an adaptive mesh of 61 points.

Problem 1.2. A population dynamics problem governed by an integro-differential equation with a nonlocal boundary condition (from Fairweather and López-Marcos [1991]):

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = -I(t)u, \tag{58}$$

for $0 \le x \le a$ and $t \ge 0$, where

$$I(t) = \int_0^a u(x,t)\,dx. \tag{59}$$

The initial condition is

$$u(x,0) = \frac{\exp(-x)}{2 - \exp(-a)}, \tag{60}$$

and the nonlocal boundary condition at $x = 0$ is

$$u(0,t) = g\left(\int_0^a b(x, I(t))u(x,t)\,dx, t\right), \tag{61}$$

where the functions $b(x, y)$ and $g(z, t)$ are given by

$$b(x,y) = \frac{xy\exp(-x)}{(y+1)^2}, \tag{62}$$

and,

$$g(z,t) = \frac{4z(2 - 2\exp(-a) + \exp(-t))^2}{(1 - \exp(-a))(1 - (1 + 2a)\exp(-2a))(1 - \exp(-a) + \exp(-t))}. \tag{63}$$

The exact solution to this problem is

$$u(x,t) = \frac{\exp(-x)}{1 - \exp(-a) + \exp(-t)}. \tag{64}$$

The difficulty with this problem is that the PDE and the boundary condition involve solution values across the whole domain, whereas the PDE description subroutine is called separately for each midpoint of the mesh. This is overcome by introducing two coupled algebraic equations

$$v_1(t) = \int_0^a u(x,t)\,dx, \tag{65}$$

$$v_2(t) = \int_0^a x\exp(-x)u(x,t)\,dx, \tag{66}$$

so that Eq. (58) becomes

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = -v_1 u, \tag{67}$$

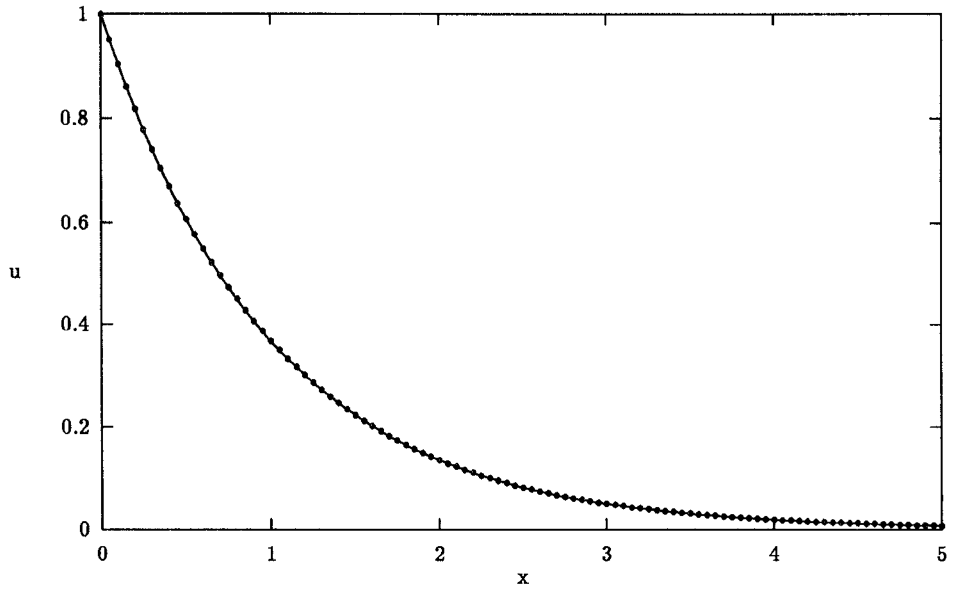

Fig. 4. The solution of Problem 1.2 at $t = 5.0$ using a fixed uniform mesh of 101 points. The dots represent the approximate solution, and the line shows the exact solution.

and the boundary condition is

$$u(0,t) = \frac{g(1,t)v_1v_2}{(v_1+1)^2}. \tag{68}$$

The integrals in Eqs. (65) and (66) are calculated numerically using the trapezoidal rule, using values of the integrand at a set of points across the domain—the coupling points of the PDE/ODE system. For a fixed mesh it is desirable that the coupling points coincide with the mesh points, thus avoiding the need to interpolate the solution at the coupling points. Interpolation is unavoidable for a nonfixed mesh, but adaptive remeshing would not be very useful for this problem anyway, since the gradients are moderate and slowly varying with time. Hence a fixed uniform mesh is used, with the same mesh of coupling points.

Figure 4 shows the approximate and exact solutions at $t = 5.0$ for $0 \le x \le 5.0$ using a fixed uniform mesh of 101 points, with the b.d.f. method of time integration, Newton iteration, and full matrix algebra.

6.2 Upwind Scheme

Problem 2.1. Burgers' equation in the inviscid limit with two initial square waves propagating in opposite directions, taken from a recent paper by Yang and Przekwas [1992] on the comparative performances of advanced shock-capturing schemes. The problem involves the propagation and collision of shocks and the expansion of discontinuities.

The governing equation is

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0, \tag{69}$$

and the initial profile is given by

$$u(x,0) = \begin{cases} 1.0, & 0.2 \le x \le 2.0, \\ -0.5, & 2.0 < x \le 3.0, \\ -1.0, & 3.0 < x \le 4.8, \\ 0.0, & \text{otherwise.} \end{cases} \tag{70}$$

The problem includes two shocks initially at $x = 2$ and $x = 3$, moving to the right and the left respectively. At time $t = 1$ the two shocks collide and form a single shock moving to the left. Also, there are two expansion discontinuities at $x = 0.2$ and $x = 4.8$, expanding to the right and left respectively. The exact solution at later times is easily found from the initial profile, using the known speeds of propagation or expansion of the discontinuities.

The solution domain is $0 \le x \le 5$, with a physical boundary condition $u(5,t) = 0$, and a numerical boundary condition in terms of extrapolated variables at $x = 0$ (see Section 3.2). In this case the numerical boundary condition at $x = 0$ will be equivalent to $u(0,t) = 0$ since the solution is uniform at $x = 0$ during the time of integration, chosen to be $0 \le t \le 2$. The numerical flux is given by Osher's scheme (see section on approximate Riemann solvers).

Figures 5 and 6 show the approximate and exact solutions at $t = 0.75$ before the shocks collide and at $t = 2.0$ when the shocks have combined into a single one. A uniform mesh of 161 points is used, with the Theta method of time integration and functional iteration. The results compare very favorably with all of those presented by Yang and Przekwas [1992].

Problem 2.2. The linear advection equation with a nonlinear source term (see LeVeque and Yee [1990]) and discontinuous initial profile:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = pu(u-1)\left(u - \frac{1}{2}\right), \tag{71}$$

for $0 \le x \le 1$ and $t \ge 0$. The discontinuity is modeled by a ramp function of width 0.01 and gradient 100, so that the exact solution at any time $t \ge 0$ is

$$u(x,t) = 1.0 + \max(\min(\delta, 0), -1), \tag{72}$$

where $\delta = 100(0.1 - x + t)$.

The initial profile is given by Eq. (72). A physical boundary condition $u(0,t) = 1.0$ is imposed at $x = 0$, with a numerical boundary condition at $x = 1$ which will be equivalent to $u(1,t) = 0$ provided that the discontinuity does not reach $x = 1$ during the time of integration (as in Problem 2.1). The numerical flux is simply $\hat{f} = u_L$ at all times.

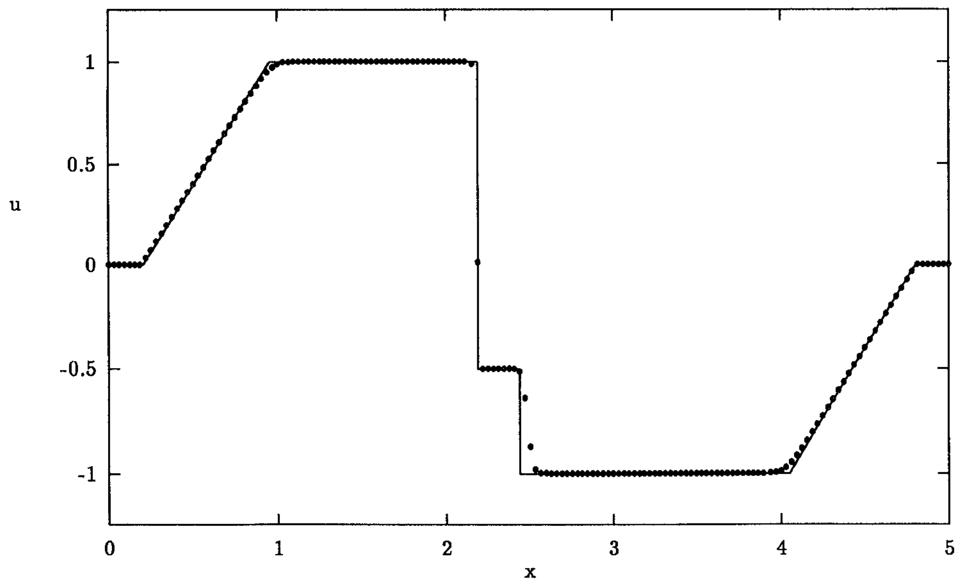

Fig. 5. The solution of Problem 2.1 at $t = 0.75$ using a uniform mesh of 161 points. The dots represent the approximate solution, and the line shows the exact solution.

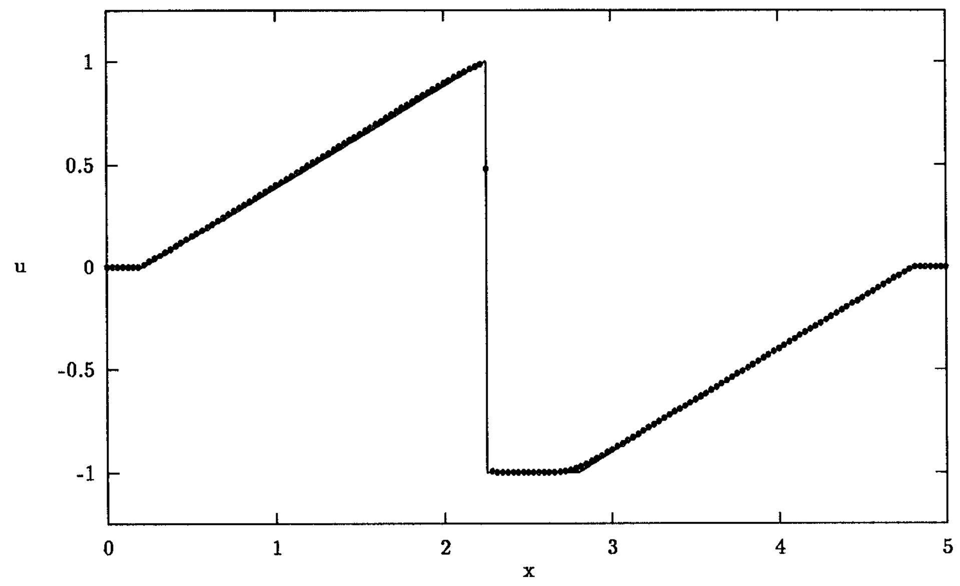

Fig. 6. The solution of Problem 2.1 at $t = 2.0$ using a uniform mesh of 161 points. The dots represent the approximate solution, and the line shows the exact solution.

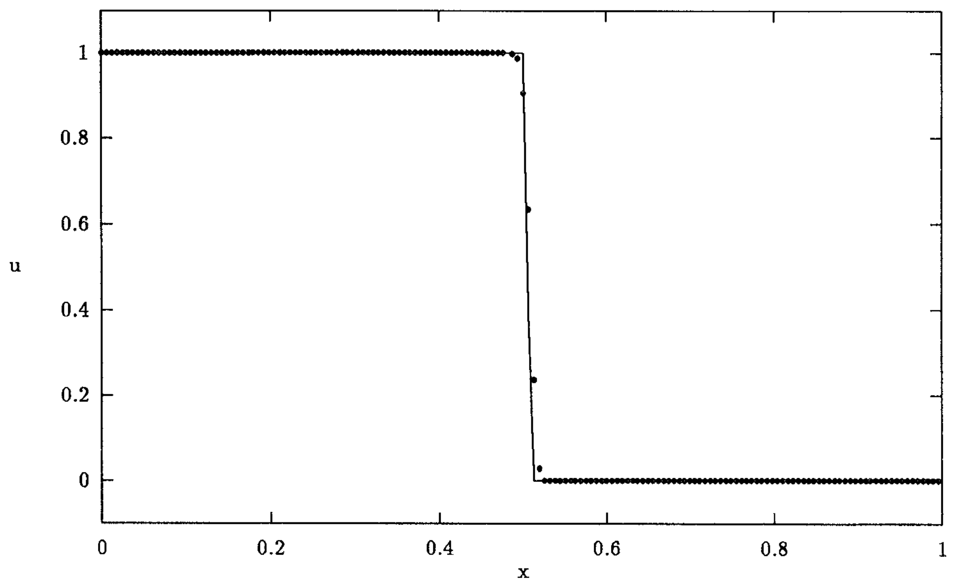

Fig. 7. The solution of Problem 2.2 at $t = 0.4$ for $p = 100$ using a uniform mesh of 161 points. The dots represent the approximate solution, and the line shows the exact solution.

Figure 7 shows the approximate and exact solutions at $t = 0.4$ for $p = 100$ using a uniform mesh of 161 points, with the Theta method of time integration and functional iteration.

It can be seen that for this value of p there is good agreement between the numerical and exact solutions. However, for much larger p (of the order 1000) incorrect propagation speeds are obtained with this mesh size, as described in the Section on Source Terms.

Problem 2.3. The standard shock-tube test problem of Sod [1978] (also called the *Riemann problem*) which involves the Euler equations of gas dynamics: The problem models the flow of air in a long tube following the sudden breakdown of a diaphragm separating two initial gas states at different pressures and densities. There is an exact solution to this problem (see Sod [1978]) containing simultaneously a shock wave, a contact discontinuity, and an expansion fan, and hence it is a particularly useful test problem for hyperbolic PDE algorithms.

The governing equations are

$$\frac{\partial \rho}{\partial t} + \frac{\partial m}{\partial x} = 0, \tag{73}$$

$$\frac{\partial m}{\partial t} + \frac{\partial}{\partial x}\left(\frac{m^2}{\rho} + (\gamma - 1)\left(e - \frac{m^2}{2\rho}\right)\right) = 0, \tag{74}$$

$$\frac{\partial e}{\partial t} + \frac{\partial}{\partial x}\left(\frac{me}{\rho} + \frac{m}{\rho}(\gamma - 1)\left(e - \frac{m^2}{2\rho}\right)\right) = 0, \tag{75}$$

where ρ is the density; m is the momentum, such that $m = \rho u$, where u is the velocity; e is the specific energy; and γ is the (constant) ratio of specific heats. The pressure p is given by

$$p = (\gamma - 1)\left(e - \frac{\rho u^2}{2}\right). \tag{76}$$

The solution domain is $0 \le x \le 1$ for $0 < t \le 0.2$, with the initial discontinuity at $x = 0.5$, so that the initial conditions are

$$\begin{aligned} &\rho(x,0) = 1, && m(x,0) = 0, && e(x,0) = 2.5, && \text{for } x < 0.5, \\ &\rho(x,0) = 0.125, && m(x,0) = 0, && e(x,0) = 0.25, && \text{for } x > 0.5. \end{aligned} \tag{77}$$

The solution is uniform and constant at both boundaries for the spatial domain and time of integration stated; and hence the physical and numerical boundary conditions are indistinguishable, and both are given by the initial conditions above.

The Osher approximate Riemann solver is used to obtain the numerical fluxes (see Hemker and Spekreijse [1986] for example). The required subroutine for this algorithm is to be made available in the NAG library (see the Appendix).

Figures 8, 9, and 10 show the approximate and exact solutions in terms of the density ρ, the velocity u, and the pressure p, at $t = 0.2$, for $\gamma = 1.4$, using a uniform mesh of 141 points, with the Theta method of time integration and functional iteration.

Problem 2.4. The shallow-water equations for waves on a sloping beach (see Watson and Peregrine [1992]): Sinusoidal waves of increasing amplitude are incident upon a sloping beach, pushing the waterline up the beach and forming highly asymmetric waves with steep fronts. This moving-boundary problem is solved by using a coordinate transformation to obtain a fixed domain, thereby introducing additional source terms. The position of the shoreline is determined by the introduction of a coupled ODE.

Following Watson and Peregrine [1992], the physical seaward-boundary condition is specified in terms of the incoming characteristic variables, and the numerical seaward-boundary condition is supplied in the form of the characteristic equation associated with the outgoing characteristic variable. This numerical boundary condition method is an alternative to the simple extrapolation of the outgoing characteristic variable described in Section 3.2. The results are very similar for the two methods in this particular example.

The governing equations are

$$\frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} = 0, \tag{78}$$

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2 + h\right) = \frac{db}{dx}, \tag{79}$$

where $h(x,t)$ is the water depth; $u(x,t)$ is the water velocity (height averaged); and $b(x)$ describes the shape of the beach in terms of its distance

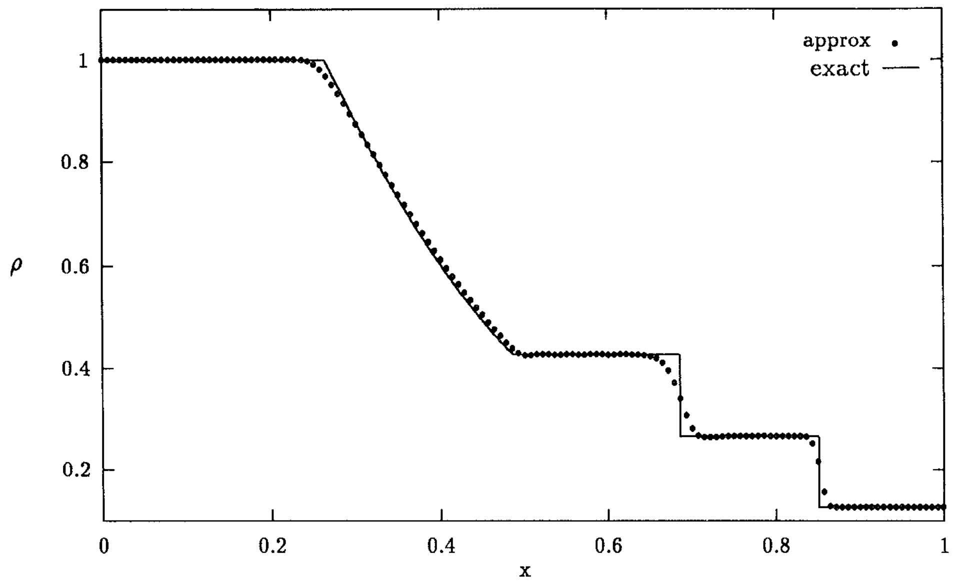

Fig. 8. The approximate and exact solutions for the density ρ in Problem 2.3 at $t = 0.2$ using a uniform mesh of 141 points

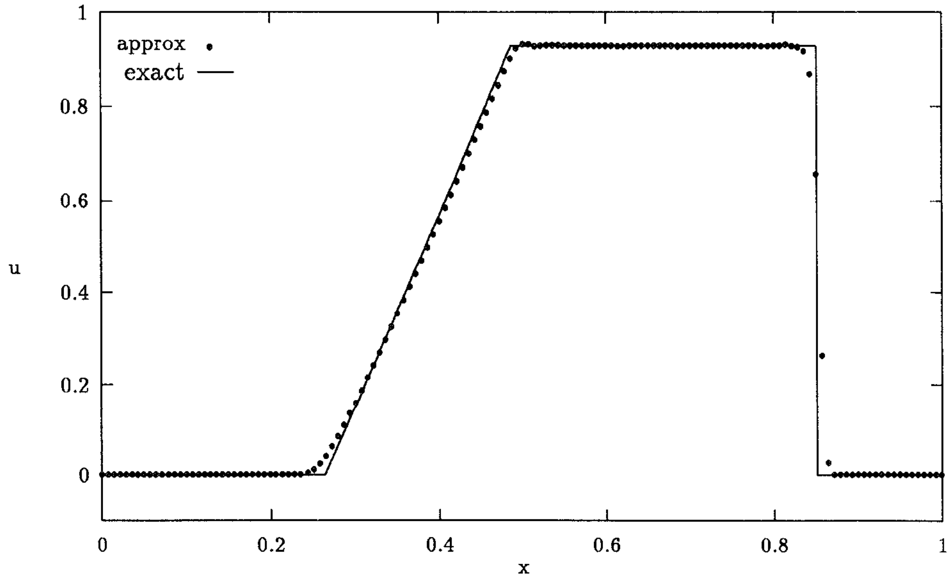

Fig. 9. The approximate and exact solutions for the velocity u in Problem 2.3 at $t = 0.2$ using a uniform mesh of 141 points.

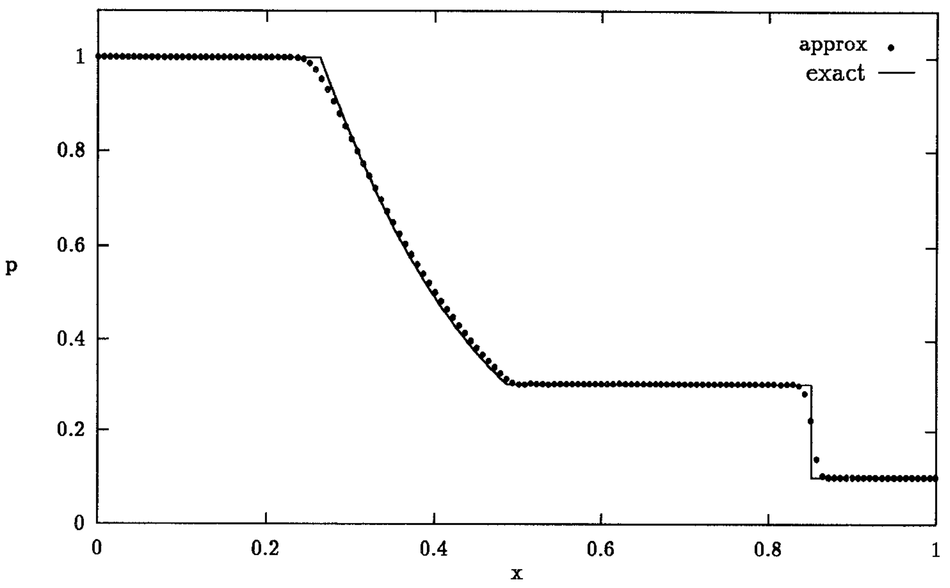

Fig. 10. The approximate and exact solutions for the pressure p in Problem 2.3 at $t = 0.2$ using a uniform mesh of 141 points.

below the initial constant water level. In this example the beach is taken to be of constant slope, and $b(x)$ is normalized so that $b(x) = -x$.

The solution domain is $-1 \le x \le x_s(t)$, where $x_s(t)$ is the position of the moving shoreline, and the boundary conditions are

$$h(x_s, t) = 0, \tag{80}$$

$$u(x_s, t) = \frac{dx_s}{dt}, \tag{81}$$

$$u(-1, t) + 2\sqrt{h(-1, t)} = f(t) + 2\sqrt{b(-1)}, \tag{82}$$

where $f(t) = 0.6[\sin(2\pi t/0.475) - \sin(2\pi t/0.525)]$ for this example. Eqs. (80) and (81) simply specify that the water depth is zero at the shoreline and that the water velocity at the shoreline is equal to the rate of change of the shoreline position x_s (this condition will be used to define the shoreline position). The seaward-boundary condition given by Eq. (82) specifies the incoming characteristic variable $u + 2\sqrt{h}$. Note that this boundary condition does not assume any knowledge of $u(-1, t)$ or $h(-1, t)$ individually; it has simply been chosen (using prior experience) to produce an appropriate solution at later times. The initial conditions are $u(x, 0) = 0$, $h(x, 0) = b(x) = -x$ for all x, and $x_s(0) = 0$.

To obtain a fixed solution domain the coordinate transformation

$$y = \frac{x + 1}{x_s + 1} \tag{83}$$

is made, and Eqs. (78) and (79) thus become (substituting $b(x) = -x$)

$$(1 + x_s)\frac{\partial h}{\partial t} - y\frac{dx_s}{dt}\frac{\partial h}{\partial y} + \frac{\partial(uh)}{\partial y} = 0, \tag{84}$$

$$(1 + x_s)\frac{\partial u}{\partial t} - y\frac{dx_s}{dt}\frac{\partial u}{\partial y} + \frac{\partial}{\partial y}\left(\frac{1}{2}u^2 + h\right) = -(1 + x_s), \tag{85}$$

with solution domain $0 \leq y \leq 1$, and boundary conditions

$$h(1, t) = 0, \tag{86}$$

$$u(1, t) = \frac{dx_s}{dt}, \tag{87}$$

$$u(0, t) + 2\sqrt{h(0, t)} = f(t) + 2. \tag{88}$$

On rearranging the spatial derivative terms in Eqs. (84) and (85) the following two PDEs are obtained in a suitable form for solution:

$$(1 + v_1)\frac{\partial h}{\partial t} + \frac{\partial}{\partial y}(uh - v_2 yh) = -v_2 h, \tag{89}$$

$$(1 + x_s)\frac{\partial u}{\partial t} + \frac{\partial}{\partial y}\left(\frac{1}{2}u^2 + h - v_2 yu\right) = -(v_1 + 1) - v_2 u, \tag{90}$$

where $v_1(t) \equiv x_s(t)$ and $v_2(t) = dv_1/dt$ (the latter having been introduced in order to avoid explicit time derivatives in the flux terms).

Using the "extra" boundary condition (87), the time-dependent variables v_1 and v_2 are determined from the following two coupled ODEs:

$$\frac{dv_1}{dt} = v_2, \qquad v_2 = u(1, t). \tag{91}$$

The numerical boundary conditions are supplied in terms of extrapolated u at the shoreline boundary $y = 1$ and the characteristic equation associated with the outgoing characteristic variable $u - 2\sqrt{h}$ at the seaward boundary, that is,

$$(1 + v_1)\frac{dv_3}{dt} + (u - \sqrt{h})\frac{\partial}{\partial y}(u - 2\sqrt{h}) = v_2(\sqrt{h} - u) - (1 + v_1), \tag{92}$$

where

$$v_3(t) = u(0, t) - 2\sqrt{h(0, t)}, \tag{93}$$

is the outgoing characteristic variable at the seaward boundary. The spatial derivative term in Eq. (92) must be calculated in the user-supplied boundary condition subroutine by one-sided differences (*into* the domain, and therefore consistent with the flow of information for this characteristic variable).

In summary, there are two PDE variables and three new time-dependent variables, described completely by the PDEs (89) and (90) and the ODEs (91) and (93). The boundary conditions are given by Eqs. (86) and (88), along with numerical boundary conditions as described above. The numerical flux is calculated using Roe's scheme (see section on approximate Riemann solvers).

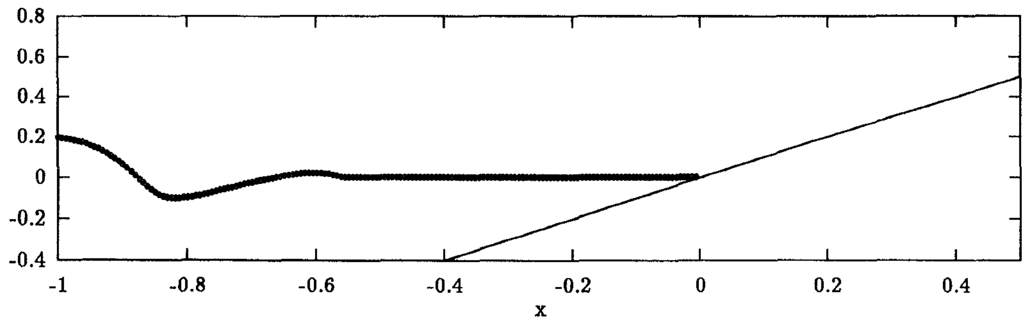

Fig. 11. The approximate solution in Problem 2.4 at $t = 0.5$ using a uniform mesh of 161 points. The beach is shown as a straight line.

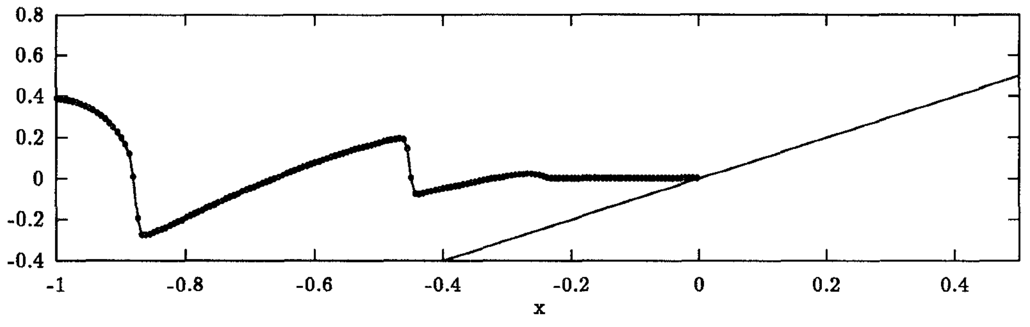

Fig. 12. As in Fig. 11 at $t = 1.0$.

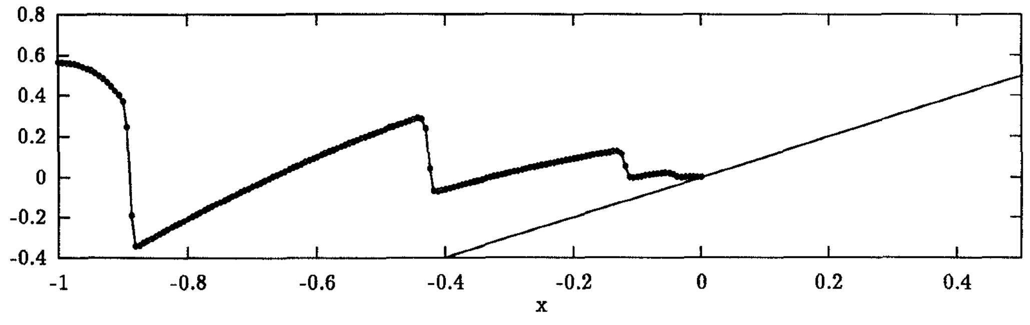

Fig. 13. As in Fig. 11 at $t = 1.5$.

Figures 11 to 15 show the approximate solutions in terms of the wave profiles at $t = 0.5$, 1.0, 1.5, 2.0, and 2.5, using a uniform mesh of 161 points, with the b.d.f. method of time integration, Newton iteration, and sparse matrix algebra. No exact solution is available for this problem, but runs with successively refined meshes showed good convergence of the solution.

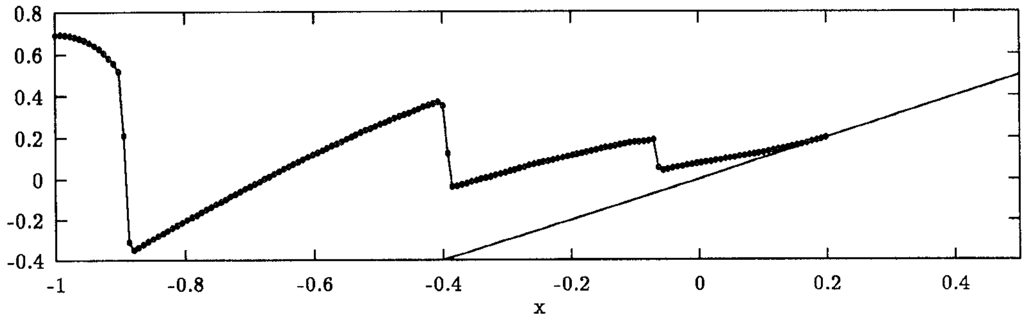

Fig. 14. As in Fig. 11 at $t = 2.0$.

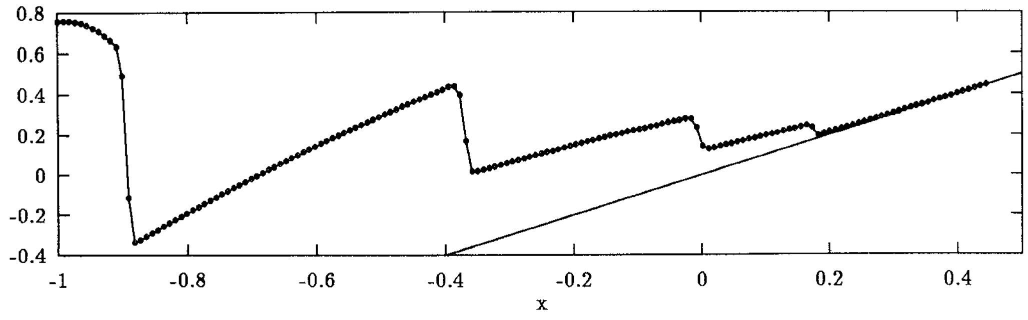

Fig. 15 As in Fig 11 at $t = 2.5$.

The solution to $t = 2.5$ took 380 seconds of CPU time on an SGI 4D240 (single-processor) workstation—the relatively high computational cost being due to the greater value of t, compared with earlier problems, and the large number of equations in the resulting ODE system.

7. SUMMARY

A set of new NAG Fortran Library routines has been described for the solution of systems of nonlinear, first-order, time-dependent partial differential equations in one space dimension, with scope for coupled ordinary differential equations. The software has a common method-of-lines framework, and much of the existing NAG Library PDE software is reused. The Keller box scheme is available for those problems for which it is suitable, along with upwind differencing for hyperbolic problems with shocks and discontinuities. A number of standard and nonstandard computational examples have been presented, illustrating the flexibility of the routines and the accurate and oscillation-free results obtainable. The disadvantage with the upwind scheme is that the user is required to provide an approximate solution to the associated Riemann problem.

The D03P subchapter of the NAG Fortran Library now forms a comprehensive set of routines for the solution of PDEs in one space dimension, with several options regarding spatial discretization, time integration, linear algebra, error control, and adaptive spatial remeshing. It is hoped that there will be future developments concerning space-time error control and adaptive remeshing and the treatment of source terms for hyperbolic problems.

APPENDIX

An Approximate Riemann Solver for the Euler Equations

A subroutine is to be included in the NAG library to calculate the Osher numerical flux for the Euler equations (as in problem 2.3). The algorithm is taken from the paper by Hemker and Spekreijse [1986] for the Euler equations in two space dimensions. A description of the algorithm is given here, followed by an example of the user-supplied subroutine required to use the NAG routine.

The Euler equations (in one dimension) are

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) = 0, \tag{94}$$

$$\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(\rho u^2 + p) = 0, \tag{95}$$

$$\frac{\partial e}{\partial t} + \frac{\partial}{\partial x}(u(e + p)) = 0, \tag{96}$$

where ρ is the density; u is the velocity; e is the specific energy; and p is the pressure given by

$$p = (\gamma - 1)\left(e - \frac{\rho u^2}{2}\right), \tag{97}$$

where γ is the (constant) ratio of specific heats.

Equations (94) to (96) can be written in the form

$$\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = 0, \tag{98}$$

where $q = (\rho, \rho u, e)^T$ and $f(q) = (\rho u, \rho u^2 + p, u(e + p))^T$.

The eigenvalues of the Jacobian $A(q) = \partial f/\partial q$ are $\lambda_1 = u - c$, $\lambda_2 = u$, and $\lambda_3 = u + c$, where $c = \sqrt{\gamma p/\rho}$ is the speed of sound.

Given left and right states q_0 and q_1, the numerical flux is as stated in the Section on Approximate Riemann Solvers, that is

$$\hat{f}(q_0, q_1) = \tfrac{1}{2}(f(q_0) + f(q_1)) - \tfrac{1}{2}\int_{q_0}^{q_1} |A(q)|\, dq, \tag{99}$$

where $|A|$ is a matrix defined by

$$|A| = A^+ - A^-, \tag{100}$$

where

$$A^+ = P\Lambda^+P^{-1}, \qquad A^- = P\Lambda^-P^{-1}, \quad \text{and} \quad A = P\Lambda P^{-1}, \tag{101}$$

where Λ is the diagonal matrix of eigenvalues of A; and Λ^+ and Λ^- are diagonal matrices such that Λ^+ has only the positive elements of Λ, and Λ^-

only the negative, so that $\Lambda = \Lambda^+ + \Lambda^-$. The columns of the matrix P are the right eigenvectors of A, and the rows of P^{-1} are the left eigenvectors of A.

The integration path in Eq. (99) is described by $q = q(s)$, $0 \le s \le 1$, $q(0) = q_0$, $q(1) = q_1$. This path is divided into three subpaths Γ_k, $k = 1,2,3$, connecting the states $q_{(k-1)/3}$ and $q_{k/3}$. On each subpath Γ_k the direction of the path $\partial q(s)/\partial s$ is tangential to the eigenvector R_m, $m = m(k)$. There are two feasible choices for $m(k)$: the so-called P-variant in which $m(1) = 1$, $m(2) = 2$, and $m(3) = 3$, so that Γ_1 corresponds to the eigenvalue λ_1, Γ_2 to λ_2, and Γ_3 to λ_3; and the O-variant in which the subpaths are taken in reverse order, that is, Γ_1 corresponds to λ_3, Γ_2 to λ_2, and Γ_3 to λ_1, so that $m(1) = 3$, $m(2) = 2$, and $m(3) = 1$.

The states $q_{1/3}$ and $q_{2/3}$ are determined from the Riemann invariants $\psi_l^{m(k)}(q(s))$, $l \ne m$, $l = 1,2,3$, which remain constant along each subpath Γ_k. The $\psi_l^m(q)$, $m = 1,2,3$, are

$$\begin{aligned} &\psi_1^2 = u, \qquad \psi_2^1 = u + 2c/(\gamma - 1), \\ &\psi_1^3 = \psi_3^1 = \ln(p\rho^{-\gamma}) = z \quad \text{say}, \\ &\psi_3^2 = p, \qquad \psi_2^3 = u - 2c/(\gamma - 1). \end{aligned} \tag{102}$$

So $q_{1/3}$ and $q_{2/3}$ are determined from q_0 and q_1 using the expression

$$\psi_l^{m(k)}(q_{(k-1)/3}) = \psi_l^{m(k)}(q_{k/3}), \qquad k = 1,2,3, \quad l \ne m(k). \tag{103}$$

The state q is expressed in terms of the variables u, c, and z, and Eq. (103) above gives $z_{1/3} = z_0$ and $z_{2/3} = z_1$. Putting $\alpha = \exp((z_1 - z_0)/(2\gamma))$ and using $p_{1/3} = p_{2/3}$ give the following linear system

$$\begin{aligned} u_{1/3} \pm 2c_{1/3}/(\gamma - 1) &= u_0 \pm 2c_0/(\gamma - 1) = \Psi_0 \quad \text{say}, \\ u_{2/3} \mp 2c_{2/3}/(\gamma - 1) &= u_1 \mp 2c_1/(\gamma - 1) = \Psi_1 \quad \text{say}, \\ c_{2/3} = \alpha c_{1/3}&, \quad u_{2/3} = u_{1/3}, \end{aligned} \tag{104}$$

where the upper sign in $\pm$ or $\mp$ denotes the P-variant, and the lower sign denotes the O-variant. This convention is used in the rest of the Appendix.

The above system is easily solved to give

$$\begin{aligned} u_{1/3} &= u_{2/3} = (\Psi_1 + \alpha\Psi_0)/(1 + \alpha), \\ c_{1/3} &= \pm\tfrac{1}{2}(\gamma - 1)(\Psi_0 - \Psi_1)/(1 + \alpha), \\ c_{2/3} &= \alpha c_{1/3}. \end{aligned} \tag{105}$$

The eigenvalues at the points $q_{k/3}$, $k = 1,2,3$, are defined by

$$\begin{aligned} \bar\lambda_0 &= \lambda_{m(1)}(q_0) = u_0 \mp c_0, \\ \bar\lambda_{1/3} &= \lambda_{m(1)}(q_{1/3}) = u_{1/3} \mp c_{1/3}, \\ \bar\lambda_{1/2} &= \lambda_{m(2)}(q_{1/3}) = \lambda_{m(2)}(q_{2/3}) = u_{1/3} = u_{2/3}, \\ \bar\lambda_{2/3} &= \lambda_{m(3)}(q_{2/3}) = u_{2/3} \pm c_{2/3}, \\ \bar\lambda_1 &= \lambda_{m(3)}(q_1) = u_1 \pm c_1. \end{aligned} \tag{106}$$

A so-called sonic point q_{s1} exists on Γ_1 if $\bar{\lambda}_0\bar{\lambda}_{1/3} \le 0$, indicating a change in sign of $\lambda_{m(k)}(q(s))$. Similarly, a sonic point q_{s2} exists on Γ_3 if $\bar{\lambda}_{2/3}\bar{\lambda}_1 \le 0$. Sonic points are easily computed from $\lambda_{m(k)}(q(s_k)) = 0$, $k = 1, 2$.

Having defined the subpaths and calculated the intermediate states and relevant eigenvectors, we give the expression for the numerical fluxes (Eq. (99)):

$$\begin{aligned}\hat{f}(q_0, q_1) = \tfrac{1}{2}\Big[& f(q_0)\big(\operatorname{sign}(\bar{\lambda}_0) + 1\big) + f(q_{s1})\big(\operatorname{sign}(\bar{\lambda}_{1/3}) - \operatorname{sign}(\bar{\lambda}_0)\big) \\ &+ f(q_{1/3})\big(\operatorname{sign}(\bar{\lambda}_{1/2}) - \operatorname{sign}(\bar{\lambda}_{1/3})\big) \\ &+ f(q_{2/3})\big(\operatorname{sign}(\bar{\lambda}_{2/3}) - \operatorname{sign}(\bar{\lambda}_{1/2})\big) \\ &+ f(q_{s2})\big(\operatorname{sign}(\bar{\lambda}_1) - \operatorname{sign}(\bar{\lambda}_{2/3})\big) + f(q_1)\big(1 - \operatorname{sign}(\bar{\lambda}_1)\big)\Big].\end{aligned} \tag{107}$$

The user calls the NAG routine which implements the above algorithm from the RM FLUX subroutine, supplying the value of the parameter γ and indicating the required variant of the scheme via the parameter VAR (the default value being "P" indicating the P-variant). Note that the left and right solution values held in the arrays ULEFT(NPDE) and URIGHT(NPDE) correspond to the dependent variables used in problem 2.3, that is, ρ, $m = \rho u$ and e (in that order). The user must ensure that the correct solution values are passed to the NAG routine. The following code provides an example of the user-supplied subroutine.

```
      SUBROUTINE RMFLUX(T, X, NPDE, ULEFT, URIGHT, RFLUX)
C     ..Scalar Arguments..
      DOUBLE PRECISION  T, X
      INTEGER           NPDE
C     ..Array Arguments..
      DOUBLE PRECISION  RFLUX(NPDE), ULEFT(NPDE), URIGHT(NPDE)
C     ..Local Scalars..
      DOUBLE PRECISION  G
      CHARACTER*1       VAR
C     ..External Subroutines..
      EXTERNAL          D03P**
C     ..Executable Statements..
      G = 1.4D0
      VAR = 'P'
      CALL D03P**(ULEFT, URIGHT, G, VAR, RFLUX)
      RETURN
      END.
```

NOTE ADDED AT PROOF STAGE

Since the paper was accepted there has been a change to the problem class in the upwind-scheme routines intended for release at Mark 17 of the NAG Fortran Library: optional second-order terms are to be included so that the routines can be used to solve a broader class of convection-diffusion problems.

The second-order terms will be discretized separately by standard central differences.

ACKNOWLEDGMENTS

The authors wish to thank NAG Ltd. for their financial support, P. H. Gaskell for providing the code for the exact solution to Problem 2.3, G. Watson for providing information on Problem 2.4, and the referees for their useful comments.

REFERENCES

BERZINS, M. 1991. Balancing space and time errors in the method-of-lines for hyperbolic equations. In the *Athens Interdisciplinary Olympia: 1st International Symposium on Methods of Lines* (Athens, Nov.).

BERZINS, M. 1990. Developments in the NAG library software for parabolic equations. In *Scientific Software Systems*, J. C. Mason and M G. Cox, Eds. Chapman and Hall, 59–72.

BERZINS, M. 1986. A C^1 interpolant for codes based upon backward differentiation formulae. *Appl. Numer. Anal. 2*, 109–118.

BERZINS, M., AND FURZELAND, R. M. 1992. An adaptive theta method for the solution of stiff and nonstiff differential equations. *Appl. Numer. Math. 9*, 1–19.

BERZINS, M., BRANKIN, R., AND GLADWELL, I. 1987. The design of stiff integrators in the NAG library Dept. of Mathematics Rep., Univ of Manchester, 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. Num. Math. 5*, 375–397.

ENGQUIST, B., AND OSHER, S. 1981 One-sided difference approximations for nonlinear conservation laws. *Math. Comput. 36*, 154, 321–352.

FAIRWEATHER, G., AND LÓPEZ-MARCOS, J. C. 1991. A box method for a nonlinear equation of population dynamics *IMA J Num Anal. 11*, 525–538.

FURZELAND, R. M. 1985. The construction of adaptive space meshes. TNER.85.022, Shell Research Ltd., Thornton Research Centre, Chester, U.K.

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, 349–388.

GLAISTER, P. 1988. Flux difference splitting for the Euler equations in one spatial co-ordinate with area variation. *Int. J. Num. Meth. Fluids 8*, 97–119.

GRIFFITHS, D. F., STUART, A. M., AND YEE, H. C. 1991. Numerical wave propagation in an advection equation with a nonlinear source term School of Mathematical Sciences Tech. Rep., Univ. of Bath, U.K.

GODUNOV, S. K. 1959. Finite-difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sbornik 47*, 271–306. In Russian.

GOTTLIEB, D., GUNZBURGER, M., AND TURKEL, E. 1982. On numerical boundary treatment of hyperbolic systems for finite difference and finite element methods. *SIAM J. Numer. Anal. 19*, 4, 671–682.

HEMKER, P. W., AND SPEKREIJSE, S. P. 1986. Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations. *Appl. Num. Math. 2*, 475–493.

KELLER, H. B. 1970. A new difference scheme for parabolic problems. In *Numerical Solutions of Partial Differential Equations*, vol. 2, J. Bramble, Ed. Academic Press, New York, 327–350.

KOREN, B. 1989. Multigrid and defect correction for the steady Navier-Stokes equations, Ph.D. thesis, Centrum voor Wiskunde en Informatica, Amsterdam.

LEVEQUE, R J. 1990. *Numerical Methods for Conservation Laws*. Springer-Verlag, New York.

LEVEQUE, R. J., AND YEE, H. C. 1990. A study of numerical methods for hyperbolic conservation laws with stiff source terms. *J. Comput. Phys. 86*, 1, 187–210.

MACHURA, M., AND SWEET, R. A. 1980. A survey of software for partial differential equations. *ACM Trans. Math. Softw. 6*, 4, 461–488.

NUMERICAL ALGORITHMS GROUP LTD. 1993. *NAG Library Manual.* Wilkinson House, Jordan Hill Road, Oxford, U.K.

OSHER, S. 1985. Convergence of generalized MUSCL schemes. *SIAM J. Numer. Anal. 22*, 5, 947–961.

OSHER, S., AND SOLOMON, F. 1982. Upwind difference schemes for hyperbolic systems of conservation laws. *Math. Comput. 38*, 158, 339–374.

ROE, P. L. 1989. A survey of upwind differencing techniques. In Lecture Notes in Physics, vol. 323. *11th International Conference on Numerical Methods in Fluid Dynamics*, D. L. Dwoyer, M. Y. Hussaini, R. G. Voigt, Eds. Springer-Verlag, New York, 69–78.

ROE, P. L. 1986. Upwind differencing schemes for hyperbolic conservation laws with source terms. In Lecture Notes in Mathematics, vol. 1270. *Nonlinear Hyperbolic Problems*, C. Carasso, P. A. Raviart, D. Serre, Eds. Springer-Verlag, 41–51.

ROE, P. L. 1981. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys. 43*, 2, 357–372.

SCHRYER, N. L. 1990. Designing software for one-dimensional partial differential equations. *ACM Trans. Math. Softw. 16*, 1, 72–85.

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, 1–32.

SOD, G. A. 1978. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys. 27*, 1, 1–31.

SPEKREIJSE, S. 1987. Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws. *Math. Comput. 49*, 179, 135–155.

SWARTZ, B. 1987. Courant-like conditions limit reasonable mesh refinement to order h^2. *SIAM J. Sci. Stat. Comput. 8*, 924–933.

SWEBY, P. K. 1989. "TVD" schemes for inhomogeneous conservation laws. In *Notes on Numerical Fluid Mechanics*, vol. 24. *Non-linear Hyperbolic Equations—Theory, Computation Methods, and Applications*, J. Ballmann, R. Jeltsch, Eds. Springer-Verlag, New York, 599–607.

VAN LEER, B. 1984. On the relation between the upwind-differencing schemes of Godunov, Engquist-Osher and Roe. *SIAM J. Sci. Stat. Comput. 5*, 1, 1–20.

VAN LEER, B. 1974. Towards the ultimate conservative difference scheme II, Monotonicity and conservation combined in a second order scheme. *J. Comput. Phys. 14*, 4, 361–370.

VAN LEER, B., THOMAS, J. L. ROE, P. L., AND NEWSOME, R. W. 1987. A comparison of numerical flux formulas for the Euler and Navier-Stokes equations. In *Proceedings of the 8th AIAA Computational Fluid Dynamics Conference*. AIAA, 36–41.

WATSON, G., AND PEREGRINE, D. H. 1992. Low frequency waves in the surf zone. In *Proceedings of the 23rd International Conference on Coastal Engineering* (Venice, Italy). To be published. Also, Rep. AM-92-11, School of Mathematics, Univ. of Bristol, U.K.

YANG, H. Q., AND PRZEKWAS, A. J. 1992. A comparative study of advanced shock-capturing schemes applied to Burgers' equation. *J. Comput. Phys. 102*, 1, 139–159.

YEE, H. C. 1987. Construction of implicit and explicit symmetric TVD schemes and their applications. *J. Comput. Phys. 68*, 1, 151–179.

YEE, H. C., BEAM, R. M., AND WARMING, R. F. 1982. Boundary approximations for implicit schemes for one-dimensional inviscid equations for gasdynamics. *AIAA J. 20*, 9, 1203–1211.

Received December 1992; revised March 1993; accepted March 1993