A Set of Library Routines for Solving Parabolic Equations in One Space Variable

P. M. DEW
University of Leeds
and
J. E WALSH
University of Manchester

The general design of a set of routines for parabolic equations is considered, with particular reference to the problem definition and the user interface. The algorithm is based on the method of lines and is restricted to one space dimension, with a simple finite-difference approximation in space The basic routine is supplied with fairly general facilities, and two simpler drivers are also provided. The solution of elliptic–parabolic systems is discussed, with two examples to illustrate the treatment of difficulties The routines are available in Mark 8 of the NAG library.

Key Words and Phrases: parabolic equations, method of lines
CR Categories 4 9, 5 17

1. INTRODUCTION

The purpose of this paper is to consider the design of some library routines for the numerical solution of a class of parabolic partial differential equations. The routines discussed here are available in the DO3 chapter of the Numerical Algorithms Group library (Mark 8). The equations are limited to one space dimension, and the numerical solution is based on the method of lines, using finite-difference approximations in space to reduce the partial differential equations to a system of ordinary differential equations in time.

The discretization routine is based on the code published by Sincovec and Madsen [13]. Another subroutine package incorporating the Sincovec and Madsen code has been produced by Hyman [8]; this is designed to handle very general partial differential equations in one space variable and time, and as a result it has a rather complicated interface for the user. The routines discussed in this report are deliberately restricted to equations of parabolic type, with the aim of making them easier to use and less likely to go wrong because of inconsistencies in the problem. This policy also influences the choice of integration method discussed

Authors' addresses: P. M Dew, Department of Computer Studies, University of Leeds, Leeds LS2 9JT, England; J E. Walsh, Department of Mathematics, University of Manchester, Manchester M13 9 PL, England.

below. Our design strategy is based on the assumption that library routines are likely to be used by a wide range of programmers with very different levels of mathematical background.

2. DESCRIPTION OF THE PROBLEM

A simple example of a parabolic equation in one space dimension is given by the following

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(g\frac{\partial u}{\partial x}\right) + f\left(u, \frac{\partial u}{\partial x}\right), \tag{1}$$

which we will use to illustrate the properties of the problem. Equation (1) is first order in t, which usually represents a time variable, and second order in the space variable x. Typical boundary conditions associated with (1) are of the form

$$\begin{array}{llll} u \text{ specified for } & a \le x \le b, & \text{at} & t = 0, \\ u \text{ or } \dfrac{\partial u}{\partial x} \text{ specified at} & x = a, b, & \text{for} & t > 0. \end{array} \tag{2}$$

Thus we have an initial-value problem in t, and a boundary-value problem in x.

We may interpret (1) as the equation of heat conduction in a bar, where u is the temperature and g the conductivity. In general, the coefficient g may be a function of u, x, and t, but we have suppressed the arguments for simplicity. If the material properties change at some point c in the interval $a < x < b$, the coefficient g and the source term f may be discontinuous at this point. The differential equation does not hold at $x = c$, but there is usually a condition

$$u \text{ continuous}, \qquad \left(g\frac{\partial u}{\partial x}\right) \text{ continuous} \tag{3}$$

at c, which is sufficient to define the solution across the interface. To define the boundary conditions at a and b, the general form may be taken as

$$pu + q\frac{\partial u}{\partial x} = r, \tag{4}$$

which includes the two forms in (2) as special cases. This may be used to represent any of the usual types of boundary condition for heat-conduction problems, including linear and nonlinear radiation conditions.

To ensure the stability of the numerical solution of (1), we have to make certain restrictions on the coefficients. For stability in the time direction we need $g > 0$, with integration in the direction of t increasing. If $g < 0$, or if the integration is backward in time, the effect of small perturbations is likely to grow exponentially. We also exclude the case $g = 0$, for which the second derivative in x is absent. In this case, the equation takes one of the forms

$$\begin{aligned} \frac{\partial u}{\partial t} &= f(u, x, t), \\ \frac{\partial u}{\partial t} &= \alpha\frac{\partial u}{\partial x} + \phi(u, x, t), \end{aligned} \tag{5}$$

or variants of these. The first of (5) is essentially an ordinary differential equation in the time direction, and the second is an equation of hyperbolic type. In both cases, boundary conditions of the form (4) at $x = a$ and $x = b$ are not appropriate for defining the numerical solution of the differential equation; in particular, the second equation is hyperbolic and requires only one boundary condition.

If we want to consider a similar problem in n space dimensions, we have an equation of the form

$$\frac{\partial u}{\partial t} = \sum_{i,j=1}^{n} a_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + F\left(u, \frac{\partial u}{\partial x}, \ldots, \frac{\partial u}{\partial x_n}\right), \tag{6}$$

which is of parabolic type if the matrix $[a_{ij}]$ is positive definite. The region of integration is normally closed in space and open in the forward direction in time. The problem has an elliptic character in the space variables, and the forward integration in time requires the solution of large systems of implicit equations at each step. The numerical solution of (6) for $n > 1$ presents a major computational problem, involving geometrical difficulties and considerations of efficiency that are not present in (1). It was therefore decided to limit the algorithms discussed here to one space dimension, and to leave the problem of higher dimensions for later development.

However, it is not necessary to restrict the parabolic equation to Cartesian coordinates, as given in (1). We therefore make an extension of the basic equation, and consider the following form:

$$\frac{\partial u}{\partial t} = \frac{1}{x^m} \frac{\partial}{\partial x} \left\{ x^m g(u) \frac{\partial u}{\partial x} \right\} + f\left(u, \frac{\partial u}{\partial x}\right). \tag{7}$$

The values $m = 0, 1, 2$ correspond to problems in Cartesian, polar, and spherical polar coordinates, respectively. The coefficients g and f in (7) may be functions of x and t as well as of u and $\partial u/\partial x$, but these arguments are omitted for clarity. When the point $x = 0$ lies in the interval of integration, we need the condition

$$\frac{\partial u}{\partial x} = 0 \quad \text{at} \quad x = 0, \tag{8}$$

for $m > 0$, to ensure that there is no singularity at this point.

We now consider a suitable form for systems of parabolic equations. In practical work, such systems often include elliptic equations, which represent the steady state of parabolic equations. We therefore want to allow for the case where the time derivative is absent for certain equations in the system, and so we introduce a coefficient on the left-hand side, giving

$$c_i \frac{\partial u_i}{\partial t} = \sum_{j=1}^{n} \frac{\partial}{\partial x} \left(g_{ij} \frac{\partial u_j}{\partial x} \right) + f_i, \qquad i, = 1, 2, \ldots, n. \tag{9}$$

(This can easily be generalized to other coordinate systems as in (7).) Taking $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$, system (9) may be written in matrix notation as follows:

$$\mathrm{D} \frac{\partial \mathbf{u}}{\partial t} = \frac{\partial}{\partial x} \left(\mathrm{G} \frac{\partial \mathbf{u}}{\partial x} \right) + \mathbf{f}, \tag{10}$$

where G is a general matrix, and D is a diagonal matrix. The diagonal form of D is rather restrictive, but if we take a general D it becomes more difficult to determine whether the system is parabolic in the time direction. Assuming the form (9), suppose that certain coefficients c_i corresponding to elliptic equations are identically zero. Taking the zero c_i to be the last $n - p$ values, we can partition (10) as follows:

$$\begin{bmatrix} D_p & 0 \\ 0 & 0 \end{bmatrix} \frac{\partial}{\partial t} \begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_e \end{bmatrix} = \frac{\partial}{\partial x} \left\{ \begin{bmatrix} G_{p_1} & G_{p_2} \\ G_{e_1} & G_{e_2} \end{bmatrix} \frac{\partial}{\partial x} \begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_e \end{bmatrix} \right\} + \mathbf{f}, \tag{11}$$

where roughly speaking the first p equations are parabolic, and the last $(n - p)$ are elliptic. If we neglect the coupling terms G_{p_2}, G_{e_1}, we can give a condition for stability of the parabolic equations in the linear case, though this would not be adequate for the general case. We do not attempt a complete test for stability, but it is required by the routines that at least one of the c_i should be nonzero at each mesh point. Further conditions on the equations are discussed later.

3. THE NUMERICAL ALGORITHM

A general approach to the numerical solution of parabolic systems is given by the method of lines. For a single equation of the form of (1) or (7), we approximate the partial derivatives on the right-hand side in terms of discrete values of u at a set of mesh points in the interval $a \leq x \leq b$. By using these approximations to represent the partial differential equation at each mesh point, we reduce it to a system of ordinary differential equations in the time direction, which is then solved numerically. Clearly the same method can be applied to (9), but in this case some of the coefficients c_i may be zero, and we therefore obtain a mixed system of ordinary differential equations and algebraic equations.

The ordinary differential equations resulting from the method of lines form a stiff system when the original problem is parabolic. It is therefore appropriate to use a method of integration suitable for stiff systems. A characteristic of the most useful methods of this type is that any oscillations present in the solution are quickly damped out. This property is clearly not correct for hyperbolic equations where a different type of integration method is appropriate. Because of this, and because it is difficult to handle problems with a shock front, the restriction of the problem class to equations of parabolic type is important. For the routines discussed here, the discretization in the space direction is based on finite differences, and a version of Gear's method is used for the time integration. We now consider these two parts of the algorithm in more detail.

3.1 Finite-Difference Discretization

The finite-difference discretization is based on the algorithm given by Sincovec and Madsen [13]. The reasons for selecting finite differences to represent the space derivatives are as follows:

(i) Most users are familiar with finite differences, and they will be able to appreciate the problem of mesh selection. This is particularly important in the present routines because there is no error estimate or error control for the spatial discretization.

(ii) The method is versatile and can be applied to a large class of problems, including those with material discontinuities.

(iii) On the basis of limited numerical testing [1], the performance of the finite-difference method is comparable with that of finite elements using linear basis functions.

(iv) Higher order approximations (used, for example, in PDECOL written by Sincovec and Madsen) give more difficulties for problems that have a discontinuity between the initial and boundary conditions, and the choice of a suitable mesh is less straightforward. However, it is intended that routines based on higher order approximations will be considered for future developments of the chapter.

A useful reference on finite-difference methods is [9].

The discretization routine has two main differences from the Sincovec and Madsen code. The approximation is modified so that the function g_{ij} in (9) is evaluated at mesh points only, in order to simplify the specification of the routine that defines the parabolic equations. This does not reduce the order of accuracy for a mesh with a smoothly varying step length, and numerical testing [1] suggests that it has little effect on the results. Further, a number of restrictions are imposed on the problem definition in order to reduce the possibility of failure in the routines and make the interface more straightforward.

To illustrate the main features of the discretization, we consider first a single parabolic equation of the form (9), with coordinate system as in (7),

$$c(u, x, t)\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left[x^m g(u, x, t)\frac{\partial u}{\partial x}\right] + f\left(u, \frac{\partial u}{\partial x}, x, t\right), \qquad a < x < b, \tag{12}$$

subject to the boundary conditions

$$p(t)u + q(t)\frac{\partial u}{\partial x} = r(u, t) \qquad \text{at} \quad x = a \quad \text{and} \quad x = b. \tag{13}$$

This form is chosen so that simple Dirichlet and Neumann conditions can be represented easily. The mesh points are chosen so that

$$a = x_1 < x_2 < \cdots < x_{N_p} = b,$$

where N_p is the number of mesh points. At internal mesh points x_j the derivatives are replaced by central differences

$$\left(\frac{\partial u}{\partial x}\right)_{x_j} \approx \frac{u_{j+1} - u_{j-1}}{x_{j+1} - x_{j-1}},$$

$$\left\{x^{-m}\frac{\partial}{\partial x}\left[x^m g(u, x, t)\frac{\partial u}{\partial x}\right]\right\}_{x_j}$$

$$\approx \frac{m+1}{x_{j+(1/2)}^{m+1} - x_{j-(1/2)}^{m+1}}\left\{x_{j+(1/2)}^m\left(\frac{g_{j+1} + g_j}{2}\right)\left(\frac{u_{j+1} - u_j}{x_{j+1} - x_j}\right) - x_{j-(1/2)}^m\left(\frac{g_j + g_{j-1}}{2}\right)\left(\frac{u_j - u_{j-1}}{x_j - x_{j-1}}\right)\right\}, \tag{14}$$

where

$$u_j = u(x_j, t), \qquad g_j = g(u_j, x_j, t), \qquad x_{j+(1/2)} = \tfrac{1}{2}(x_{j+1} + x_j).$$

It may be shown by Taylor expansion that this representation has second-order accuracy when the step length varies smoothly and g is continuous. The difference replacement requires modification at the boundary, and we consider $x = a$ as an example. There are two cases arising from (13).

Case 1: *Dirichlet Boundary Condition.* In this case $q(t) = 0$ at $x = a$, and the value of u at x_1 is obtained directly from the boundary condition

$$u_1 = \frac{r(t)}{p(t)}.$$

We make the restriction that r must be a function of t only. To avoid complications (particularly for a system of parabolic equations), the algorithm uses a dummy equation

$$\frac{du_1}{dt} = 0$$

at the boundary point x_1.

Case 2: *General Boundary Condition.* When $q(t) \neq 0$, the first mesh point must be included in the discretization. An expression for the derivative at the boundary is given by

$$\left(\frac{\partial u}{\partial x}\right)_{x_1} = \left[\frac{r(u, t) - p(t)u}{q(t)}\right]_{x_1}, \tag{15}$$

and the spatial operator is approximated by

$$\left(x^{-m}\frac{\partial}{\partial x}\left[x^m g \frac{\partial u}{\partial x}\right]\right)_{x_1} \approx \frac{m+1}{x_{3/2}^{m+1} - x_1^{m+1}}\left\{x_{3/2}^m \frac{(g_2 + g_1)}{2}\frac{(u_2 - u_1)}{(x_2 - x_1)} - x_1^m g_1\left(\frac{\partial u}{\partial x}\right)_{x_1}\right\}. \tag{16}$$

This equation has only first-order accuracy. By using the above expressions it is easily seen that (12) and (13) can be reduced to a system of ordinary differential equations of the form

$$C_j \frac{du_j}{dt} = F_j, \qquad j = 1, 2, \ldots, N_p. \tag{17}$$

The method extends naturally to a system of parabolic equations such as (9), subject to boundary conditions of the form

$$p_i(t)u_i + q_i(t)\frac{\partial u_i}{\partial x} = r_i(\mathbf{u}, t), \qquad i = 1, 2, \ldots, n, \quad \text{at} \quad x = a \quad \text{and} \quad b.$$

At a new time step, the coefficients p_i, q_i, r_i, $i = 1, 2, \ldots, n$, are first evaluated; then if any $q_i = 0$, the corresponding u_i is calculated and a dummy equation

$(du_i/dt) = 0$ is introduced as above. The coefficients r_i are then reevaluated. (When $q_i = 0$, the corresponding r_i must be independent of $\mathbf{u}$.)

3.2 Integration in the Time Direction

The finite-difference discretization applied to a system of parabolic equations (9) reduces the problem to that of solving a system of ordinary differential equations of the form (17), which may be written

$$C_{ij}(\mathbf{u}, t)\frac{du_{ij}}{dt} = F_{ij}(\mathbf{u}, t), \qquad i = 1, 2, \ldots, n, \qquad j = 1, 2, \ldots, N_p, \tag{18}$$

where the vector $\mathbf{u}$ is now extended as follows

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{N_p} \end{bmatrix}, \quad \mathbf{u}_j = \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{n,j} \end{bmatrix}. \tag{19}$$

In the discretization routine, the coefficients C_{ij} and F_{ij} are computed from the definition of the parabolic system, the vector of mesh points, the current solution vector $\mathbf{u}$, and the time t. If c_i in (9) is zero for some i, then $C_{ij} \equiv 0$ for the corresponding i and all j, so that (18) represents a mixed system of algebraic and ordinary differential equations.

To investigate the behavior of the system (18), we consider the simple heat conduction equation (or diffusion equation)

$$c\frac{\partial u}{\partial t} = g\frac{\partial^2 u}{\partial x^2}, \qquad g/c \text{ a positive constant}, \quad 0 < x < 1,$$

subject to Dirichlet boundary conditions

$$u(0, t) = u(1, t) = 0.$$

The central-difference replacement leads to

$$\frac{d\mathbf{u}}{dt} = \frac{g}{c}A\mathbf{u}, \qquad A = \frac{1}{h^2}\begin{bmatrix} -2 & +1 & 0 & & \cdots & 0 \\ +1 & -2 & +1 & 0 & \cdots & 0 \\ 0 & +1 & -2 & +1 & \cdots & 0 \\ \vdots & & \text{-----} & \text{-----} & \text{-----} & \text{-----} \\ 0 & \cdots & & 0 & +1 & -2 \end{bmatrix}, \tag{20}$$

where the mesh points are equally spaced at interval h. The eigenvalues of A are all negative, and a measure of the stiffness of (20) is given by

$$S(A) = \frac{\max}{\min}\left|\frac{\text{eigenvalue of } A}{\text{eigenvalue of } A}\right| = O(N_p^2).$$

The system is mildly stiff; nevertheless, if we want to avoid very short time-steps it is necessary to use an implicit method to compute the solution. The simplest implicit method for parabolic equations is the trapezium rule (or Crank-Nicolson method). However, this has a low fixed order, and for a general integration routine it is more satisfactory to use one of the variable-order, variable-step methods, which aim to integrate the system as efficiently as possible over a range while meeting a specified accuracy criterion. The most widely used variable-order routines for parabolic systems use Gear's method, with backward differentiation formulas that are stiffly stable [5]. This method has the effect of damping any oscillations in the numerical solution, which makes it suitable for purely parabolic problems. If the solution is oscillatory, Gear's method will generally be less successful, and for systems that do not behave like parabolic equations some caution is necessary in estimating the accuracy of the results.

Although Gear's method is normally implemented for systems written in the standard form $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$, it can be easily extended to a general implicit system

$$\boldsymbol{\phi}(t, \mathbf{y}, \mathbf{y}') = \mathbf{0}, \tag{21}$$

as described in [5] and [2]. For simplicity, we describe the method in terms of a constant time step δt. The derivative $\mathbf{y}'(t)$ at $t = t_p$ is approximated by the backward-difference formula

$$\mathbf{y}_p' \approx \frac{1}{\delta t} \sum_{i=1}^{k} \nabla^{(i)} \mathbf{y}(t_p) = \frac{\mathbf{y}(t_p) - \sum_{i=1}^{k} \alpha_i \mathbf{y}(t_p - i\delta t)}{\beta_0 \delta t}, \tag{22}$$

where the coefficients $\{\alpha_1, \alpha_2, \ldots, \alpha_k, \beta_0\}$ are those given by [5]. On substituting (22) into (21), we see that an approximate solution $\mathbf{y}_p \approx \mathbf{y}(t_p)$ is given implicitly by

$$\boldsymbol{\phi}\left(t_p, \mathbf{y}_p, \frac{\mathbf{y}_p - \sum_{i=1}^{k} \alpha_i \mathbf{y}_{p-i}}{\beta_0 \delta t} \right) = 0. \tag{23}$$

The solution $\mathbf{y}_p$ is computed iteratively using Newton's method

$$J_m\{\mathbf{y}_p^{(m+1)} - \mathbf{y}_p^{(m)}\} = -\beta_0 \delta t\, \boldsymbol{\phi}\left(t_p, \mathbf{y}_p^{(m)}, \frac{\mathbf{y}_p^{(m)} - \sum_{i=1}^{k} \alpha_i \mathbf{y}_{p-i}}{\beta_0 \delta t} \right), \qquad m = 0, 1, \ldots,$$

where J_m is the Jacobian matrix defined by

$$J_m = \left[\beta_0 \delta t \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{y}} + \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{y}'} \right]_{(t_p, \mathbf{y}_p^{(m)})}.$$

In practice the Jacobian matrix is recalculated only when the rate of convergence of the iteration is slow, and normally the same Jacobian matrix is used over a number of steps. It can be shown that (23) is a kth-order method.

In Gear's implementation of the backward differentiation formula, the previous solution values $y_{p-1}, \ldots, y_{p-k}$ are not stored directly; instead, scaled estimates of the first k derivatives at t_p are retained. This simplifies the expressions for the error that are used in controlling the order and step size. The construction of a routine for Gear's method (for systems of standard form) is fully described in [4].

In the parabolic routines we want to solve ordinary differential equations of the form (18). The function ϕ of (21) is given by

$$\phi_{ij} = C_{ij}(\mathbf{u}, t)\frac{du_{ij}}{dt} - F_{ij}(\mathbf{u}, t) \tag{24}$$

and hence the Jacobian matrix is

$$J = \left[\mathrm{D} - \delta t \beta_0 \frac{\partial \phi}{\partial \mathbf{u}}\right],$$

where ϕ has the same ordering as $\mathbf{u}$ in (19), and D is a diagonal matrix with C_{ij} $(\mathbf{u}, t)$ on the diagonal. Because $F_{ij}(\mathbf{u}, t)$ involves only $\mathbf{u}_{j-1}$, $\mathbf{u}_j$, and $\mathbf{u}_{j+1}$, the Jacobian matrix is block tridiagonal with each block of order n. This property is used to reduce the storage and computation time. If we make additional restrictions on the problem specification as follows,

(i) the coefficient g_{ij} in (9) satisfies

$$g_{ij} = 0, \qquad \text{if} \quad i \neq j,$$
$$g_{ii} = g_{ii}(u_i, t);$$

(ii) the functions f_i and c_i have limited dependence on the derivatives of $\mathbf{u}$,

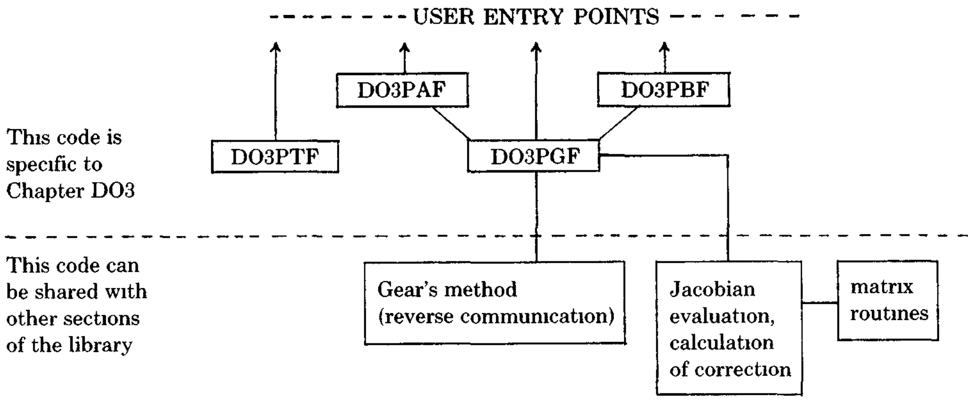
$$f_i = f_i\left(\mathbf{u}, \frac{\partial u_i}{\partial x}, t\right), \qquad c_i = c_i\left(\mathbf{u}, \frac{\partial u_i}{\partial x}, t\right);$$

then the bandwidth of the Jacobian matrix can be reduced further to $2n + 1$.

3.3 Elliptic-Parabolic Solver

The routines are designed to handle systems that may include both elliptic and parabolic equations. (As mentioned earlier, a check is included to ensure that at least one equation is parabolic.) It has been found by experiment that Gear's method is less robust when solving the differential equations arising from an elliptic-parabolic system. In certain cases a solution may not exist at all, if the system is inconsistent. If there is a solution, the time integration may be difficult because the Jacobian matrix is nearly singular (since D is singular), and its condition cannot be improved by reducing the time step, which is the usual expedient when the iteration does not converge satisfactorily. Numerical experiments have indicated that more Jacobian matrix evaluations are generally required when solving a mixed system of algebraic and ordinary differential equations. In an attempt to make the code more robust, the elliptic part of the system is isolated at the initial stage and solved directly to obtain initial conditions for the time integration that are consistent with the discretization. The conditions specified by the user are taken as a first estimate, which is then improved by using Newton iteration. If the Jacobian matrix of the elliptic part is singular, or if the iteration fails to converge, the routine is terminated with a suitable error indicator. This has the advantage that only an approximate solution to the elliptic

Table I General Layout of the Parabolic Routines


USER ENTRY POINTS
DO3PAF
DO3PBF
This code is specific to Chapter DO3
DO3PTF
DO3PGF
This code can be shared with other sections of the library
Gear's method (reverse communication)
Jacobian evaluation, calculation of correction
matrix routines

part has to be provided initially, and any inconsistency in the specification of the problem is likely to be revealed by lack of convergence. To start Gear's method, an estimate of the derivatives with respect to t is required, and for the elliptic part of the system this is set to zero.

4. STRUCTURE OF THE ROUTINES

In the design of a library routine it is essential that the specification should be convenient and comprehensible to the user. The design of the interface is particularly difficult for partial differential equations because the definition of the problem requires a large amount of information, which has to be passed through the parameter list. In the NAG routines we have tried to balance the need for generality in the problem definition and the facilities provided, against the length of the parameter list and reasonable simplicity in use.

The policy adopted is similar to that of the ordinary differential equations and optimization sections of the library, described by Gladwell [7] and Gill et al. [6]. There is a rather general routine (DO3PGF) at the lowest level, which is intended for use only by sophisticated programmers, and we supply simpler "drivers" that call this routine. One of the drivers provided (DO3PAF) is designed for a single parabolic equation, and the other (DO3PBF) for a simple system of parabolic equations. The detailed specification of each routine is given in the NAG library manual [10]. The general layout of the routines is shown in Table I, and we describe the main features below.

4.1 DO3PAF Routine

This is a simple routine for the user who wishes to solve a single parabolic equation, or to gain some experience of the routines in a simple case before trying to solve a more complicated system. DO3PAF is designed for a single equation of the form

$$\frac{\partial u}{\partial t} = x^{-m} \frac{\partial}{\partial x}\left(x^m g(u, x, t) \frac{\partial u}{\partial x}\right) + f\left(u, \frac{\partial u}{\partial x}, x, t\right), \qquad a \le x \le b, \quad t > 0, \tag{25}$$

subject to an initial condition and the boundary conditions (13). The mesh points are taken to be equally spaced. The user need only specify the ends of the range, the number of mesh points, the initial values of u at the mesh points, and two subroutines PDEF (to evaluate the functions g and f) and BNDY (to evaluate the coefficients in the boundary conditions). The simple structure of (25) makes it possible to include a number of checks to ensure that the problem is correctly posed for the routine. The checks are as follows:

(i) At *each* mesh point (including $x = a$ and $x = b$) the function g must be positive, and the integration must be in the forward direction in t. These checks should detect an unstable problem (or more commonly a problem that has been incorrectly specified).

(ii) The boundary conditions must have a reasonable form. Specifically, the coefficients p and q must not be zero simultaneously, and if q is zero, then r must be a function of t only. If $m > 0$, then $x = 0$ must not lie within the range; if it is an endpoint, the corresponding boundary condition must be either $(\partial u/\partial x) = 0$, or $u = r(t)$.

These checks are made on every call of the discretization routine. The parabolic equation is solved over a range; the user merely specifies the initial and final values of t. The error conditions arising from the time integration are discussed below.

Because DO3PAF is designed for a single equation, the parameter lists for the subroutines PDEF and BNDY are shorter than the corresponding lists required in DO3PBF and DO3PGF. This creates a problem when the simple driver calls the low-level routine. One solution is to use two sets of dummy subroutine names in the main driver. All the routines DO3PAF, DO3PBF, and DO3PGF then call the main driver, which has an indicator to distinguish the cases. An alternative solution is to ask the user to specify *fixed*-name subroutines for defining the equation and boundary conditions in the simple driver DO3PAF. This makes it possible to use DO3PGF as the main driver. We preferred the second solution because it simplifies the specification while not unduly restricting the user.

4.2 DO3PBF Routine

This routine is designed to solve quite general parabolic (or parabolic-elliptic) systems, but the number of facilities provided has been kept to a minimum. This ensures that the parameter list is as short as possible. It is expected that DO3PBF will be the most widely used routine. The actual system solved is

$$c_i\left(\mathbf{u}, \frac{\partial u_i}{\partial x}, x, t\right) \frac{\partial u_i}{\partial t} = x^{-m}\left\{\frac{\partial}{\partial x}\left[x^m g_i(u_i, x, t)\frac{\partial u_i}{\partial x}\right]\right\} + f_i\left(\mathbf{u}, \frac{\partial u_i}{\partial x}, x, t\right), \qquad i = 1, 2, \ldots, n, \qquad a \le x \le b, \tag{26}$$

where the vector $\mathbf{u}$ is defined as in eq. (10). The restriction imposed on the arguments of the functions g_i, f_i, and c_i are to ensure that the bandwidth of the

Jacobian matrix is limited to $2n + 1$. This reduces both storage and computation time, which can be significant in a large problem. The form (26) does not seriously restrict the practical problems that can be solved, and we feel that the bandwidth limitation is worthwhile. (The alternative would be to provide a parameter that allowed the user to specify the bandwidth, but this would make an added difficulty for inexperienced programmers.)

The user can specify certain standard mesh spacings (equally spaced points, or points projected from a circle at equal angular distances), or he can specify the individual mesh points. As in DO3PAF, the routine integrates the problem over a specified range, with the user providing the initial and final values of t.

To ensure that (26) is of parabolic type, we make the following checks at each mesh point

(i) there is at least one nonzero value of c_i;
(ii) if $c_i \neq 0$, then $g_i/c_i > 0$.

The first check is a simple test to ensure that the system includes at least one parabolic equation. It is not applied at the first and last mesh points $x = a$ and $x = b$. The second check is a test for stability; if it is not satisfied, the integration in the forward t direction is likely to be unstable.

The boundary conditions are of the form

$$p_i(t)u_i + q_i(t)\frac{\partial u_i}{\partial x} = r(u_i, t), \qquad i = 1, 2, \ldots, n, \tag{27}$$

and the restrictions given in (ii) of Section 4.1 apply for each equation in (27).

4.3 DO3PGF Routine

As mentioned earlier, this routine is intended for the sophisticated user and is designed to solve a general parabolic system (9), with coordinate options as in (7). The only restriction on the functions c_i, g_{ij}, and f_i is that c_i must be nonzero for at least one i at each mesh point. The integration in the t variable may be either forward or backward, and the user must ensure that the problem is correctly posed for the routine. Although (9) is not as general as the specification in [13], it should be sufficiently general to cover a large number of parabolic systems arising in practice.

The boundary conditions are of the form given by (27), with one minor extension: the coefficients p_i and q_i may both be zero at either $x = a$ or $x = b$. (It is then assumed in the routine that r_i is also zero.) This case corresponds to a null boundary condition, and for the problem to be correctly posed we require the corresponding g_{ij} to be zero for all j. However, there is no guarantee that the integration will be successful in this case, and the option should be used with caution. The position and nature of the boundary conditions are particularly critical in determining the stability of such problems. A numerical example illustrating these points is given in Section 5.

Although DO3PGF is written to integrate over a specified range, extra facilities

Table II Structure of the Algorithm for DO3PGF

```
BEGIN
      SET up the machine dependent constants (e g , relative precision),
      CHECK the initial parameters for consistency (e g., that mesh points are correctly ordered,
      $x_i > x_{i-1}$),
      SET up the parameters that partition the workspace,
      IF initial call of the routine (or a restart of the integration)
      THEN
            CALL the discretization routine DO3PTF to compute the coefficients $C_{ij}$, $F_{ij}$,
            IF there are any elliptic equations THEN
            BEGIN
                  Isolate the elliptic part of the system and refine the initial solution vector so
                  that it satisfies the discrete equations arising from the elliptic part to an
                  accuracy of half the tolerance specified If convergence is not obtained then
                  return to the user with the appropriate error flag set,
                  RECOMPUTE the coefficients $C_{ij}$, $F_{ij}$ by recalling DO3PTF
            END;
            ESTIMATE the initial step size and the first derivative $du/dt$ at the initial value of $t$,
            for Gear's method
      ELSE fetch the integration parameters from the workspace,
      IF required call the monitor output subroutine,
      START of main integration loop,
      STEP  If sufficiently close to TOUT and not initial call THEN goto TE,
            CALL Gear's method to compute the solution at the next time step (see Table III),
            IF error flag set in Gear's method THEN goto ERROR;
            IF required call the monitor output subroutine,
            go to STEP,
TE    INTERPOLATE the solution vector to give solution at TOUT,
      SAVE the integration parameters in the workspace,
      IF required call the monitor output subroutine,
      RETURN to the calling program,
ERROR Set the error flag and return to the calling program
END
```

are provided so that the solution vector can be monitored after each successful step in the t direction. The integration can also be terminated after any step. There is flexibility in the choice of the error test and also in specifying the bandwidth of the Jacobian matrix.

The general structure of the DO3PGF routine is given in Table II. The integration is terminated when the variable t has passed the specified endpoint TOUT. (An option is provided to stop the integration when t is exactly TOUT, to allow for the case when the solution cannot proceed beyond this point. This option is not shown in Table II to avoid giving too much detail.)

4.4 DO3PTF Routine

This is the finite-difference discretization routine that computes the coefficients C_{ij}, F_{ij} in (24), using the mesh points, the current solution vector, and the subroutines defining the parabolic system. The routine is designed for the problem specifications given in DO3PAF, DO3PBF, and DO3PGF, and all the checks are performed within this routine. The structure of the routine is similar to that of the code in [13], except that the modification to evaluate the function g at mesh points only requires some reorganization of the calculation. This change was made so that all the coefficients in the parabolic equation would be evaluated at the same points and the user could define them all in a single subroutine PDEF.

4.5 The Gear Routine

The Gear routine is written in "reverse communication" form; currently it is based on the code given in [4]. The use of "reverse communication" means that the evaluation of the Jacobian matrix and vector ϕ of (24) are performed in the calling program as shown in Table III. This form is convenient for a general library, because the basic Gear routine can then be used in different contexts, for ordinary differential equations as well as for parabolic equations.

Since the finite-difference discretization is of relatively low order, only modest accuracy is likely to be required in the t integration. Therefore we do not use the higher order backward difference formulas, and the maximum order in the Gear routine is restricted to four. This reduces the size of the workspace and improves the stability of the method.

To make the integration robust, we include three standard error traps:

(1) We have incorporated the limiting precision test discussed in [11] for the Adams method. Suppose the error test used in Gear's method is of the form

$$\|\mathbf{e}\|_w \leq \text{TOL}, \tag{28}$$

where $\|\cdot\|_w$ denotes the weighted error norm and TOL is the tolerance. Then

(i) the tolerance must satisfy

$$\text{TOL} \geq 20U\|\mathbf{u}\|_w,$$

where U is the relative precision of the machine (the smallest number such that $1 + U \neq 1$) and $\mathbf{u}$ is the solution vector,

(ii) the step size must satisfy

$$|\delta t| \geq 4U|t| + \sigma,$$

where t is the current value of the independent variable and σ is the smallest positive number that can be stored in the machine.

The integration is terminated with an error indication when either (i) or (ii) is not satisfied. This error often occurs when the solution is unstable. If the integration is to be continued, it is necessary to relax the accuracy requirement.

(2) In Section 3.2 it was stated that the Jacobian matrix was reevaluated when the rate of convergence of the iteration was slow. If at any step a second or third Jacobian evaluation is required, the step size is halved for each evaluation. If the iteration is unsuccessful after three such evaluations, the integration is terminated. Similarly, if the estimate of the local error fails the error test three times in one step, the integration is terminated. In both cases the routine can be recalled

Table III Method of using GEAR routine in DO3PGF

```
          calling program
                .
                .
                .
ENTRY CALL GEAR (    , T, DT, BETA, Y, DY, COR, IND,    )
          IF IND = 0 THEN integration has finished, check error flag
          ELSE
            BEGIN
              IF IND = 2 THEN
                BEGIN
                  FORM the Jacobian matrix J = $\left\{\frac{\partial\phi}{\partial y} - \mathrm{DT\,BETA}\,\frac{\partial\phi}{\partial y'}\right\}$
                  COMPUTE the LU factors of J
                END,
              FORM the correction COR, using the LU factors of J, from J.c =
              −DT BETA $\phi(T, y, y')$,
              goto ENTRY (to recall the GEAR code)
            END
                .
                .
                .
```

Note that Y, DY, CORR are arrays containing the values of the vectors $\mathbf{y}$, $\mathbf{y}'$ and $\mathbf{c}$, respectively The correction $\mathbf{c}$ is used to obtain the new estimate of the solution vector and its derivative; that is,

$$\mathbf{y}_{\text{new}} = \mathbf{y} - \mathrm{DT\ BETA}.\mathbf{c}$$

and

$$\mathbf{y}'_{\text{new}} = \mathbf{y}' - \mathbf{c}$$

to restart the integration with order 1. This error should not arise very often; when it does it probably means that the problem has been incorrectly specified or that the solution is discontinuous for some value of t.

(3) The integration is terminated if the Jacobian matrix is almost singular. The step size is halved and the user can recall the routine to continue the integration. This error may occur when solving an elliptic-parabolic system, in which case reducing the step size may not be successful. In such a case it is advisable to check the problem specification.

In general the user cannot be expected to provide an initial estimate of the step size. Suppose we wish to solve the linear differential system

$$\mathbf{u}'(t) = A\mathbf{u}(t),$$

using the first-order backward difference formula $(I - \delta t A)\mathbf{u}_{n+1} = \mathbf{u}_n$. It can be shown that the principal term in the local truncation error is

$$\mathbf{T} = \frac{(\delta t)^2 A^2}{2}\mathbf{u}(t) = \frac{(\delta t)^2 A}{2}\mathbf{u}'(t).$$

The error test (28) is satisfied if we use $\mathbf{T}$ as the error estimate, and if

$$\| \mathbf{T} \|_w = \tfrac{1}{2}(\delta t)^2 \| A\mathbf{u}'(t) \|_w \le \tfrac{1}{2}(\delta t)^2 \| A \|_w \cdot \| \mathbf{u}'(t) \|_w \le \mathrm{TOL},$$

so that

$$|\delta t| \le \sqrt{\frac{2\,\mathrm{TOL}}{\| A \|_w \| \mathbf{u}'(t) \|_w}}. \tag{29}$$

The algorithm used to estimate the initial time step (or after a restart) for the system (18) is

```
DT := range of integration in time;
DUNORM := $\|\mathbf{F}\|_w / \max |C_{ij}|$;
ABSDT := ABS(DT);
IF (DUNORM $\neq$ 0) THEN
    ABSDT := $\min\left\{\text{ABSDT}, \sqrt{\dfrac{2K.\text{TOL}}{\text{DUNORM}}}\right\}$;
DT = sign(DT) * max {ABSDT, 4 $U|t| + \sigma$};
```

The value of K used in the NAG code is 0.01. This should be related to $1/\|A\|_w$, and since $\|A\|_w$ depends on the stiffness and the scaling of the time variable it is possible that the initial time step may be too large or too small. In the initial step, we therefore reduce the step size by a factor of 10 (rather than 2) if it is too large; if it is too small, it will be increased by the method. The algorithm above is similar to one used by Shampine in the Adams code (cf. [12]).

4.6 Matrix Routines

The Jacobian matrix, which is required for the solution of the implicit equations in Gear's method, is found by using forward differences to approximate the derivatives. Numerical differentiation is used here because it is often rather complicated to construct the Jacobian analytically and experience with other NAG routines suggests that errors are frequently made at this point. However, a facility may be introduced in later developments for using the exact Jacobian, if it is available.

The matrix is of band form, as stated above, and in general it is block-triple-diagonal, with blocks of order n. The form is as follows:

$$J = \begin{bmatrix} B_1 & C_1 & & \\ A_2 & B_2 & C_2 & \\ & A_3 & B_3 & C_3 \\ & & \multicolumn{2}{c}{\cdots\cdots} \end{bmatrix}.$$

The triple-diagonal structure arises because the space derivatives are approximated on three neighboring points only. The detailed form of the submatrices A_j, B_j, C_j depends on the coupling between the equations of the system. In the restricted case that is solved in D03PBF, it is assumed that the ith equation is of the form shown in (26), where the dependence of f_i, g_i, c_i on the components of the solution is restricted. It can be seen that this reduces the matrices A_j, C_j to diagonal form, so that the bandwidth is $2n + 1$. If the dependence is unrestricted, A_j, C_j have to be treated as full matrices.

In solving the equations for the Newton correction, it cannot be assumed that J is diagonally dominant, because there may be elliptic equations in the system or large coefficients in A_j, C_j arising from f_i. Interchanges are therefore needed in factorizing the matrix J. Table IV gives a summary of the storage and operations required for the principal cases. We consider n equations, N_p mesh points, and give only the leading term of each expression.

Table IV. Solution of Corrector Equations, Matrix J

Case	Jacobian	Bandwidth	Storage	Multiplications
Single equation	Tridiagonal	3	$4 \cdot N_p$	$5 \cdot N_p$
Equation (26)	Band	$2n + 1$	$3N_p \cdot n^2$	$2N_p \cdot n^3$
General equation	Stepped band	$3n$	$4N_p \cdot n^2$	$(23/6)N_p \cdot n^3$
General equation	Treated as band	$4n - 1$	$6N_p \cdot n^2$	$8N_p \cdot n^3$

5. NUMERICAL TESTS

Simple examples of the use of the routines are given in the NAG manual [10]. The testing that was carried out in developing the routines had three objectives, to check the correctness of the coding, to test the effectiveness of the approximations used, and to try out rather difficult cases which are not straightforward parabolic systems. Detailed results of tests on the mesh size and error tolerance are given in [1]. It should be noted that the routines carry out error checks on the integration in the time direction, but the space approximation is not checked, and the user must devise his own procedure for validating the choice of mesh. Problems in which the distribution of mesh points is very critical are those with marked boundary layers, and those with the effect of a steep wavefront moving across the region (e.g., Burgers' equation). In the time direction, particular difficulties may arise with problems that have oscillatory solutions (for which Gear's method is not very effective) and with strongly nonlinear problems, such as those involving combustion or explosion terms.

We now consider two test problems to illustrate some aspects of the routines.

Problem 1. This is an artificial problem, designed to test the code for handling elliptic equations. The system is as follows:

$$o = \frac{\partial^2 u_1}{\partial x^2} - e^{-2u_2},$$

$$\frac{\partial u_2}{\partial t} = \frac{\partial^2 u_2}{\partial x^2} + e^{-u_1} + e^{-2u_2},$$

for $0 \le x \le 1$, $t > 0$, with boundary conditions

$$\begin{aligned}
u_2 &= \log(x + p) && \text{at} \quad t = 0,\\
\frac{\partial u_1}{\partial x} &= e^{-u_1} && \text{at} \quad x = 0,\\
u_1 &= \log(1 + p + t) && \text{at} \quad x = 1,\\
u_2 &= \log(p + t) && \text{at} \quad x = 0,\\
\frac{\partial u_2}{\partial x} &= e^{-u_1} && \text{at} \quad x = 1.
\end{aligned}$$

The analytical solution is

$$u_1 = u_2 = \log(x + p + t).$$

The parameter p can be used to vary the difficulty of the problem. For $p = 1$, the Jacobian matrix is singular at $t = 0$, and the integration is terminated automatically. For $p > 1$ the integration is successful.

Problem 2. The following system arises in the study of the deposition of charged particles in tubes [3]:

$$\begin{aligned} &\frac{1}{r}\frac{\partial}{\partial r}(ru_1) = 4\alpha u_2, \\ &(1 - r^2)\frac{\partial u_2}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u_2}{\partial r} - ru_1u_2\right), \end{aligned} \tag{30}$$

for $0 < r < 1$, $t > 0$, with initial condition

$$u_2 = 1, \qquad \text{for} \quad 0 \le r \le 1, \quad t = 0,$$

and boundary conditions

$$u_1 = \frac{\partial u_2}{\partial r} = 0 \qquad \text{at} \quad r = 0; \qquad u_2 = 0 \qquad \text{at} \quad r = 1, \quad t \ge 0.$$

If DO3PGF is used with the system in the form (30), the integration is unstable. We differentiate the first equation to make it second order and deduce another boundary condition.

We multiply the first equation of (30) by r^2 and differentiate with respect to r. This gives (after simplifying)

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r^2\frac{\partial u_1}{\partial r}\right) = 4\alpha\left(u_2 + r\frac{\partial u_2}{\partial r}\right). \tag{31}$$

From the original equation we obtain the second boundary condition

$$\frac{\partial}{\partial r}(ru_1) = 0 \qquad \text{at} \quad r = 1, \quad \text{for all } t.$$

To obtain the initial values of u_1, we solve eq. (31) at $t = 0$, where

$$\frac{\partial}{\partial r}(ru_1) = 4\alpha r, \qquad u_1 = 0 \quad \text{at} \quad r = 0,$$

giving

$$u_1 = 2\alpha r, \qquad \text{at} \quad t = 0.$$

It is not essential to have the analytical solution at $t = 0$, but it is a convenient way of getting the required starting approximation. Writing the second equation of (30) in the form required for DO3PGF, we obtain

$$(1 - r^2)\frac{\partial u_2}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u_2}{\partial r}\right) - \left(u_1\frac{\partial u_2}{\partial r} + u_2\frac{\partial u_1}{\partial r} + \frac{u_1u_2}{r}\right). \tag{32}$$

At the point $r = 0$, the term $(u_1u_2)/r$ has to be evaluated from the limiting form. We have, as $r \to 0$,

$$\frac{u_1u_2}{r} \to u_1\frac{\partial u_2}{\partial r} + u_2\frac{\partial u_1}{\partial r} = 2\alpha u_2^2.$$

Table V

Mesh		N_p			
		10	20	40	80
Results for Problem 2, $r = 0.95$, $t = 0.0001$					
Uniform	u_1	1.7204	1.8417	1.8650	1.8706
	u_2	0.4332	0.7108	0.6904	0.6854
Circular projection	u_1	1.8272	1.8614	1.8699	1.8719
	u_2	0.6855	0.6836	0.6840	0.6843
Elliptic, $e = 0.9$	u_1	1.8478	1.8672	1.8713	1.8721
	u_2	0.6837	0.6841	0.6845	0.6839
Results for Problem 2, $r = 0.9$, $t = 0.0001$					
Uniform	u_1	1.7123	1.7726	1.7916	1.7962
	u_2	0.8664	0.9535	0.9719	0.9763
Circular projection	u_1	1.7590	1.7889	1.7956	1.7971
	u_2	0.9156	0.9625	0.9752	0.9766
Elliptic, $e = 0.9$	u_1	1.7774	1.7934	1.7966	1.7974
	u_2	0.9468	0.9747	0.9761	0.9773
Results for Problem 2, $r = 0.95$, $t = 0.001$					
Uniform	u_1	1.6431	1.7138	1.7238	1.7261
	u_2	0.3273	0.3399	0.3385	0.3382
Circular projection	u_1	1.6888	1.7180	1.7248	1.7264
	u_2	0.3315	0.3371	0.3379	0.3381
Elliptic, $e = 0.9$	u_1	1.6925	1.7192	1.7251	1.7264
	u_2	0.3391	0.3381	0.3382	0.3381

This example illustrates the use of analysis in setting up the equations for numerical solution. There is a boundary layer near $r = 1$, and so we expect that a variable mesh will be needed, with points clustered in this region.

In Table V we give results for three different choices of mesh,

(i) equally spaced points;

(ii) points defined by projection from a circle,

$$r_j = \sin\theta_j, \qquad \theta_j = \tfrac{1}{2}\pi\frac{j-1}{N_p-1}, \qquad j = 1, 2, \ldots, N_p;$$

(iii) points defined by projection from an ellipse,

$$r_j = \frac{\sin\theta_j}{\sqrt{(1-e^2\cos^2\theta_j)}}, \qquad \theta_j \text{ as in (ii).}$$

(The meshes (i) and (ii) correspond to options provided in DO3PBF.) The tolerance for the time integration is $\frac{1}{2}\times 10^{-5}$, and the solution values in the table are obtained by interpolation on the mesh. We see that the accuracy for a fixed number of points generally increases as the points are clustered more closely near $r = 1$.

REFERENCES

1. BERZINS, M., AND DEW, P M. A note on the testing of semi-discrete methods for the solution of second order nonlinear parabolic partial differential equations in one space variable. Rep. 128, Dep. Computer Studies, Univ. of Leeds, U.K , 1979.
2 BRAYTON, R.K., GUSTAVSON, F.G., AND HACHTEL, G D. A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulae *Proc IEEE 60* (1972), 98-108.
3 CHEN, R Y. Deposition of charged particles in tubes *J. Aerosol Sci 9* (1978), 449-453
4. DEW, P M., AND WEST, M.W A package for integrating stiff systems of ordinary differential equations based on Gear's method. Rep 111, Dep. Computer Studies, Univ of Leeds, U.K., 1978
5. GEAR, C.W *Numerical Initial-Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, N.J., 1971.
6. GILL, P E., MURRAY, W., PICKEN, S.M., AND WRIGHT, M.H. The design and structure of a Fortran program library for optimization Stanford Tech. Rep. SOL77-7, Stanford Univ , Stanford, Calif., 1977.
7. GLADWELL, I. Initial value routines in the NAG library. *ACM Trans. Math. Softw 5*, 4 (Dec. 1979), 386-400.
8. HYMAN, J M. MOLID: A general purpose subroutine package for the numerical solution of partial differential equations Los Alamos Rep., Univ of California, 1979
9. MITCHELL, A R., AND GRIFFITHS, D.F *The Finite Difference Method in Partial Differential Equations* Wiley, Chichester, U.K., 1980.
10. NAG LIBRARY MANUAL (MARK 8). Numerical Algorithms Group, 7 Banbury Road, Oxford, U.K., 1980
11. SHAMPINE, L.F. Limiting precision in differential equation solvers. *Math Comput. 28* (1974), 141-144.
12. SHAMPINE, L.F , AND GORDON, M.K *Computer Solution of Ordinary Differential Equations* Freeman, San Francisco, 1975.
13 SINCOVEC, R.F , AND MADSEN, N.K Software for nonlinear partial differential equations. *ACM Trans Math. Softw 1*, 3 (Sept. 1975), 232-260.

Received August 1980, revised January 1981, accepted March 1981.