Algorithm 690
Chebyshev Polynomial Software for Elliptic-Parabolic Systems of PDEs

M. BERZINS and P. M. DEW
University of Leeds

PDECHEB is a FORTRAN 77 software package that semidiscretizes a wide range of time-dependent partial differential equations in one space variable. The software implements a family of spatial discretization formulas, based on piecewise Chebyshev polynomial expansions with C^0 continuity. The package has been designed to be used in conjunction with a general integrator for initial value problems to provide a powerful software tool for the solution of parabolic-elliptic PDEs with coupled differential algebraic equations. Examples are provided to illustrate the use of the package with the DASSL d.a.e. integrator of Petzold [18].

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

General Terms: Algorithms

Additional Key Words and Phrases: Elliptic parabolic equations, method of lines, spatial discretization methods

I. INTRODUCTION

In recent years there has been considerable interest in the development of general-purpose codes for time-dependent partial differential equations (see the surveys by Machura and Sweet [15], Ortega and Voigt [24], and Hindmarsh [25]). These codes are generally based on the method of lines using the backward differentiation method for the temporal integration. The C^1 collocation code PDECOL, written by Madsen and Sincovec [16], is the first widely available general-purpose code to provide the user with the option of selecting the order of the approximation to be used in spatial discretization.

In this paper we describe the PDECHEB software package, which uses a family of spatial discretization formulas that are based upon C^0 continuous piecewise polynomials and apply to a wide range of parabolic/elliptic systems of PDEs with coupled differential-algebraic equations. The advantage of using C^0 continuity, compared with C^1 continuity used by PDECOL, is its

Authors' address: School of Computer Studies, The University, Leeds LS2 9JT, U.K.

much wider applicability (e.g., problems with material interfaces and with discontinuous initial and boundary conditions) coupled with the fact that it is possible to derive a complete class of formulas including first and second order. The formulas were derived by Berzins and Dew [3] using an improved form of the generalized Chebyshev method of Berzins and Dew [2]. A recent theoretical analysis of the method for steady state problems is due to Funaro [26].

The discretization method is similar to the C^0 collocation methods discussed by Diaz [9], by Carey et al. [8], and by Leyk [14]. In all these methods the approximate solution is continuous at a set of spatial breakpoints and satisfies the differential equation at a number of collocation points between each pair of breakpoints; Diaz uses transformed Gauss or Jacobi points while Leyk uses the zeros of a Legendre polynomial. Dunn and Wheeler [7] discuss the effect of using different choices of collocation points and show that the method converges for any distinct set of collocation points. The finite element approach of these authors is to fix the degree of the approximating polynomial and to increase the number of spatial elements to obtain the required accuracy.

In contrast the method implemented in PDECHEB, is a global element method, in the sense that the number of elements is normally fixed by physical constraints, and it is the degree of the polynomial that can be adjusted to obtain the required accuracy. PDECHEB uses the transformed Chebyshev collocation points between each pair of breakpoints so that the method exploits the well-known approximation properties of *global* Chebyshev polynomial collocation methods; see Canuto and Quateroni [6]. Thus, PDECHEB combines the advantages of global approximation methods with the flexibility of the C^0 collocation approach in handling material interfaces and boundary conditions. The software described here provides a powerful variable order (in space) discretization method that is applicable to a broad class of time-dependent PDEs in one space dimension.

The paper is organized as follows. Section 2 defines the class of coupled ODE/PDE problems that PDECHEB can be used to solve. Sections 3 and 4 provide an outline of the spatial discretization method used in the software. The PDECHEB software has been designed to spatially discretize the coupled PDE/ODE problem into a system of time-dependent ODEs; Section 5 describes how the software can be used in conjunction with the DASSL integrator for differential-algebraic initial value problems to compute the numerical solution. The precise form of the user interface to the software package is described in Section 6, with reference to a number of numerical examples, which are also used to illustrate the performance of the software.

2. DEFINITION OF THE PROBLEM CLASS

The PDECHEB package has been designed to provide software to solve a wide range of physically realistic PDE problems. The problem class of coupled systems of PDE and ODE is based on that of Schryer [20] who illustrates its usefulness [21] by showing that it encompasses multiphase

PDE problems and PDE problems with coupled moving boundaries or integro-differential equations. The main difficulty in attempting to spatially discretize a broad class of problems is that many unsuitable or even ill-posed problems are included. The PDECHEB package is not intended to be used for hyperbolic and diffusion-convection equations which require discretization methods that take specific properties of the solution into account, such as traveling waves.

The PDECHEB package can be used to spatially discretize the time-dependent system of NPDE partial differential equations

$$Q_k(x, t, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_t, \mathbf{u}_{xt}, v, \dot{v}) = x^{-m} \frac{\partial}{\partial x}\left(x^m R_k(x, t, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_t, \mathbf{u}_{xt}, v, \dot{v})\right)$$
$$(x, t) \in \Omega = [a, b] \times (0, t_e),$$
$$a < b,\ k = 1, \cdots, NPDE. \tag{2.1}$$

The vector v and its time derivative $\dot{v}$ are assumed to be the solution of a coupled ODE initial value problem, see Eq. (2.2) below. The vector $\mathbf{u}(x, t)$ is defined by

$$\mathbf{u}(x, t) = [u_1(x, t), \ldots, u_{NPDE}(x, t)]^T,$$

the vectors $\mathbf{u}_x(x, t)$, $\mathbf{u}_t(x, t)$ and $\mathbf{u}_{xt}(x, t)$ are similarly defined. The nonnegative integer m denotes the space geometry type when m is greater than zero, a must be greater than or equal to zero.

The solution of the coupled ODE system v is assumed to be defined by the system of equations

$$\mathbf{F}(v, \dot{v}, \xi, \mathbf{u}^*, \mathbf{u}_x^*, \mathbf{R}^*, \mathbf{u}_t^*, \mathbf{u}_{xt}^*) = 0 \tag{2.2}$$

where vector $\boldsymbol{F}$, the ODE variables $v(t)$ and their time derivatives $\dot{v}$ are vectors with NV components. The ODE may be coupled to the PDE at a vector of space points $\boldsymbol{\xi}$ of length n_ξ, where

$$\xi_i \epsilon [a, b],\ i = 1, \cdots, n_\xi.$$

The vector $\mathbf{u}^*$ is composed of

$$\mathbf{u}^* = [\mathbf{u}^{\mathrm{T}}(\xi_1, t), \mathbf{u}^{\mathrm{T}}(\xi_2, t), \cdots, \mathbf{u}^{\mathrm{T}}(\xi_{n\xi}, t)]^{\mathrm{T}}$$

and the other vectors $\mathbf{u}_x^*$, $\mathbf{R}^*$, $\mathbf{u}_t^*$, $\mathbf{u}_{xt}^*$ are similarly defined. This approach follows the original method of Schryer [20], rather than his later method [21] in which the PDE solution in the coupled ODE (Eq. (2.2)) is expressed not in terms of its values at the spatial coupling points, but only in terms of its B-spline coefficients. Schryer [21] points out that the two methods are equivalent.

2.1. Boundary Conditions and Initial Conditions

The function R in Eq. (2.1) may be thought of as a flux, e.g., $R = Ku_x$; it is convenient to use this flux in the definition of the boundary conditions. Thus

the boundary conditions are defined by

$$\beta_k(x,t)R(x,t,\mathbf{u},\mathbf{u}_x,\mathbf{u}_t,\mathbf{u}_{xt},v,\dot{v}) = \gamma_k(x,t,\mathbf{u},\mathbf{u}_x,\mathbf{u}_t,\mathbf{u}_{xt},v,\dot{v}) \quad \text{where} \quad k = 1,\ldots,NPDE \text{ and } x = a \text{ or } x = b \tag{2.3}$$

and the initial conditions are assumed to have the form

$$\mathbf{u}(x,0) = \mathbf{k}(x),\ x \in [a,b], \tag{2.4}$$

$$v = \mathbf{k}_v. \tag{2.5}$$

2.2. Polar Coordinates

In the case when the integer m in Eq. (2.1) is greater than zero, we have to make special provision for the polar form of the differential operator by using the technique of Berzins and Dew [2]. Equation (2.1) is rewritten as

$$\overline{Q}_k(x,t,\mathbf{u},\mathbf{u}_x,\mathbf{u}_t,\mathbf{u}_{xt},v,\dot{v}) = \frac{\partial}{\partial x}\left(R_k(x,t,\mathbf{u},\mathbf{u}_x,\mathbf{u}_t,\mathbf{u}_{xt},v,\dot{v})\right), \quad k = 1,\ldots,NPDE \tag{2.7}$$

where the function $Q_k(\cdots)$ is defined by

$$\overline{Q}_k(x,t,\cdots) = Q_k(x,t,\cdots) - \frac{m}{x}R_k(x,t,\cdots) \tag{2.8}$$

unless $a = 0$ when the limit value of (2.1) as x tends to 0 is used, i.e.,

$$\overline{Q}_k(a,t,\cdots) = \frac{1}{m+1}Q_k(a,t,\cdots). \tag{2.9}$$

The form of the PDE that we consider in describing the software is that given by Eq. (2.7).

3. OUTLINE OF CHEBYSHEV C^0 COLLOCATION METHOD

In this section we provide a description of the key features of the numerical method used by the PDECHEB software. In order to use PDECHEB, the user selects a number (at least two) of breakpoints to define the spatial mesh

$$a = X_0 < X_1 < \cdots < X_{NEL} = b, \tag{3.1}$$

where the X_j are the breakpoints. This mesh partitions the interval $[a,b]$ into NEL elements,

$$I_j = [X_{j-1}, X_j] \quad \text{of length} \quad h_j = X_j - X_{j-1},\ j = 1,2,\ldots,NEL. \tag{3.2}$$

The breakpoints are chosen by the user to suit the application. However, if the function Q in Eq. (2.1) has a discontinuity with respect to the spatial variable x at some point in the interval $[a,b]$, then this point must be one of the breakpoints.

PDECHEB approximates the kth component of the solution to Eq. (2.1), (2.2), and (2.3) by a continuous piecewise polynomial approximation $U_k(x,t)$

using a polynomial of degree N in each element. We consider the case $N \geq 2$; as this makes it possible to describe the method of Berzins and Dew [3] in a very straightforward manner.[1] The degree of N of the polynomial is chosen by the user. The approximate solution has the form

$$U_{k,j}(x,t) = \sum_{i=0}^{N} a_{j,i}(t) T_i(W_j(x)), \quad x \in I_j \quad k = 1, \ldots, NPDE. \tag{3.3}$$

where $U_{k,j}(x,t)$ is the restriction of $U_k(\cdot,t)$ to the element I_j, $T_i(\cdot)$ is the Chebyshev polynomial of the first kind of degree i and W_j is defined as the linear map of the interval $[X_{j-1}, X_j]$ onto $[-1,1]$.

For the sake of clarity, we consider the case of one PDE and drop the subscript k used in Section 2 and Eq. (3.3). On substituting the approximate solution (3.3) for the exact solution in Eqs. (2.1), (2.2), and (2.3), we can use interpolation to define the piecewise polynomial approximations $\overline{Q}(x,t)$ and $R(x,t)$. These two polynomials are of degree N in each interval and are defined by

$$\begin{aligned} \overline{Q}(x_{j,i},t) &= \overline{Q}(x_{j,i},t,\mathbf{U},\mathbf{U}_x,\mathbf{U}_t,\mathbf{U}_{xt},\mathbf{V},\dot{\mathbf{V}}) \\ R(x_{j,i},t) &= R(x_{j,i},t,\mathbf{U},\mathbf{U}_x,\mathbf{U}_t,\mathbf{U}_{xt},\mathbf{V},\dot{\mathbf{V}}) \\ & j = 1,2,\ldots,NEL;\ i = 0,1,2,\ldots,N, \end{aligned} \tag{3.4}$$

where the vectors $\mathbf{U}$, $\mathbf{U}_x$, $\mathbf{U}_{xt}$, and $\mathbf{U}_t$ are of length one, as there is only one PDE, and the transformed Chebyshev points $\{x_{j,i}\}$ are defined by

$$W_j(x_{j,i}) = \cos\left(\frac{(N-i)\pi}{N}\right), \quad j = 1,2,\ldots,NEL,\ i = 0,1,\ldots,N \tag{3.5}$$

where $W_j(x)$ is the linear map defined in Eq. (3.3). The vectors $\mathbf{V}$ and $\dot{\mathbf{V}}$ are approximations to v and $\dot{v}$ produced by the numerical methods described below. Although the exact PDE flux function $R(x,t)$ is assumed to be continuous at the breakpoints, the function $\overline{Q}(x,t)$ is allowed to be discontinuous at these points. (In fact the *numerical* approximation to the PDE flux may also happen to be discontinuous at the breakpoints; see Eq. (3.13) below.)

3.1. COLLOCATION EQUATIONS

The transformed Chebyshev points (excluding the breakpoints) are the collocation points used by Berzins and Dew [3]. In other words, the computed solution and its space and time derivatives are calculated to satisfy the collocation-like equations:

$$\begin{aligned} \overline{Q}(x_{j,i},t) - \frac{\partial R}{\partial x}(x_{j,i},t) &= 0 \\ & j = 1,2,\ldots,NEL;\ i = 1,2,\ldots,N-1, \end{aligned} \tag{3.6}$$

where the points $x_{j,i}$ are defined by Eq. (3.5).

[1] In the case when $N = 1$, Berzins and Dew [3] showed that the method is a lumped Galerkin finite element method with linear basis functions or a second-order finite difference method.

In the case of just one PDE, and if

$$Q(\ldots) = \frac{\partial u}{\partial t} - f\left(x, t, u, \frac{\partial u}{\partial x}\right), \tag{3.7}$$

Eq. (3.6) defines $\partial U/\partial t$ *explicitly* at the collocation point $x_{j,\iota}$,

$$\frac{dU_{j,\iota}}{dt} = \frac{\partial R}{\partial x}(x_{j,\iota}, t) + \frac{m}{x_{j,i}} R(x_{j,\iota}, t) + f\left(x_{j,\iota}, t, U_{j,\iota}, \frac{dU_{j,\iota}}{dx}\right)$$

where $U_{j,\iota} = U(x_{j,\iota}, t)$ $j = 1, 2, \ldots, NEL$; $i = 1, 2, \ldots, N-1$.

3.2. Boundary and Breakpoint Conditions

The polynomial $U(x, t)$ is continuous at each breakpoint and is required to satisfy the finite-element type orthogonality condition:

$$\int_a^b \left(\bar{Q}(\cdots) - \frac{dR}{dx}(\cdots)\right) \bar{p}_j(x)\, dx = 0 \tag{3.8}$$

where $\bar{p}_j(x)$ is the linear basis (hat) functions defined by

$$\bar{p}_j(X_\iota) = 1, \qquad i = j,$$
$$\bar{p}_j(X_i) = 0 \qquad \textit{otherwise} \text{ and } j = 0, 1, \cdots, NEL.$$

Equation (3.8) is integrated by parts to get

$$\int_a^b \bar{Q}(\cdots)\bar{p}_j(x)\, dx + R(\cdots)\frac{d\bar{p}_j}{dx}\, dx = \left[R(\cdots)\bar{p}_j\right]_a^b \tag{3.9}$$

where the term on the right side of the equality may be seen to be zero from Eq. (3.9) unless $j = 0$ or $j = NEL$, in which case the boundary conditions (2.3) are used to substitute for the values of the flux $R(\cdots)$ at the boundaries. At the left-hand boundary,

$$\beta(a, t)\int_a^{X_1} R\frac{d\bar{p}_0}{dx} + \bar{Q}\bar{p}_0\, dx = -\gamma\left(a, t, U, \frac{\partial U}{\partial x}, \frac{\partial U}{\partial t}, \frac{\partial^2 U}{\partial x\, \partial t}, \mathbf{V}, \dot{\mathbf{V}}\right) \tag{3.10}$$

and at the right-hand boundary,

$$\beta(b, t)\int_{X_{NEL-1}}^b R\frac{d\bar{p}_{NEL}}{dx} + \bar{Q}\bar{p}_{NEL}\, dx = \gamma\left(b, t, U, \frac{\partial U}{\partial x}, \frac{\partial U}{\partial t}, \frac{\partial^2 U}{\partial x\, \partial t}, \mathbf{V}, \dot{\mathbf{V}}\right). \tag{3.11}$$

Berzins and Dew [3] showed that the integrals in Eq. (3.9) can be approximated by the $N+1$ point Clenshaw–Curtis quadrature rule with weights denoted by $\{\lambda_\iota\}_{\iota=0}^N$, to derive the equations at the boundaries. In the case when $N > 1$ these equations can be further simplified by using the

collocation equations (3.6) to get collocation-like equations at the boundaries.

$$\beta(a,t)\overline{Q}(a,t) = \beta(a,t)\frac{\partial R}{\partial x}(a,t) + \left[\beta(a,t)R(a,t) - \gamma\left(a,t,U,\frac{\partial U}{\partial x},\frac{\partial U}{\partial t},\frac{\partial^2 U}{\partial x\,\partial t},\mathbf{V},\dot{\mathbf{V}}\right)\right]\frac{2}{\lambda_N h_1}$$

$$\beta(b,t)\overline{Q}(b,t) = \beta(b,t)\frac{\partial R}{\partial x}(b,t) + \left[-\beta(b,t)R(b,t) + \gamma\left(b,t,U,\frac{\partial U}{\partial x},\frac{\partial U}{\partial t},\frac{\partial^2 U}{\partial x\,\partial t},\mathbf{V},\dot{\mathbf{V}}\right)\right]\frac{2}{\lambda_N h_{NEL}} \quad (3.12)$$

In the case when $Q(\ldots)$ has the form given by Eq. (3.7), it can be seen that Eqs. (3.12) define the time derivatives at the boundaries, providing that the functions $\beta(a,t)$ and $\beta(b,t)$ are nonzero.

In a similar way, Berzins and Dew [3] showed that quadrature rules and the collocation equations (3.6) can be used to rewrite the interior breakpoint condition (3.9) as[2]

$$h_j\overline{Q}(x_{j,N},t) + h_{j+1}\overline{Q}(x_{j+1,0},t) = h_j\frac{\partial R}{\partial x}(x_{j,N},t) + h_{j+1}\frac{\partial R}{\partial x}(x_{j+1,0},t) + \left[R(x_{j+1,0},t) - R(x_{j,N},t)\right]\frac{2}{\lambda_N}. \quad (3.13)$$

Although the transformed Chebyshev points $x_{j,N}$ and $x_{j+1,0}$ are both equal to the breakpoint X_j for $j = 1,2,\ldots,NEL-1$, we denote by $\overline{Q}(x_{j,n},t)$ the value of $\overline{Q}$ evaluated at X_j using the polynomial in I_j and by $\overline{Q}(x_{j+1,0},t)$ the value of $\overline{Q}$ evaluated at X_j using the polynomial in I_{j+1}. This takes into account possible discontinuities in the functions $Q(\cdots)$ at the breakpoints $\{X_j\}$ by using the values of $\overline{Q}(x,t)$ as x tends to the breakpoint from above and below. In practice, it is straightforward for the user of the software to specify any such discontinuities, as is shown in Section 6.3.

4. EXTENSION TO COUPLED ODE EQUATIONS

In order to compute a numerical solution to the coupled ODE system (2.2) we need to be able to approximate the vectors in this equation. Equation (3.3) defines a natural interpolant based on $U(x,t)$ for the vector $\mathbf{U}^*$. This interpolant can be differentiated to generate $\mathbf{U}^*_x$. As $\partial U/\partial t$ is also a continuous piecewise polynomial of degree N, we can define a similar interpolant to form $\mathbf{U}^*_t$ and $\mathbf{U}^*_{xt}$. In this way we can generate approximate solution and derivative values for any coupling points in the interval $[a,b]$, except the interior breakpoints at which the derivatives $\partial U/\partial x$ and $\partial^2 U/\partial x\partial t$ in

[2]Although the PDE flux is assumed to be continuous the simplification in Berzins and Dew [3] incorrectly assumed that the numerical flux $R(x,t)$ was also continuous at the breakpoints and so neglected the bracketed term $[\ldots]$ in Eq. (3.13)

general will be discontinuous. For this reason the software will warn against the use of breakpoints as coupling points and will supply the weighted average of the left and right derivative values as the derivative value at the breakpoint. For example, in the case of $\partial U/\partial x$:

$$\frac{\partial U}{\partial x}(X_{j+1}, t) = \frac{1}{h_j + h_{j+1}} \left[h_j \frac{\partial U}{\partial x}(x_{j,N}, t) + h_{j+1} \frac{\partial U}{\partial x}(x_{j+1,0}, t) \right], \quad j = 1, \ldots, j-1. \tag{4.1}$$

The approach used to define the residual of the coupled ODE equations (2.2) is thus to interpolate the approximate PDE solution $U(x, t)$ and its derivatives to obtain approximate values at the coupling points $\boldsymbol{\xi}$ and to substitute these values with the approximations to $\mathbf{V}$ and $\dot{\mathbf{V}}$ in Eq. (2.2).

$$\mathbf{F}(\mathbf{V}, \dot{\mathbf{V}}, \boldsymbol{\xi}, \mathbf{U}^*, \mathbf{U}_x^*, \mathbf{R}^*, \mathbf{U}_t^*, \mathbf{U}_{xt}^*) = 0. \tag{4.2}$$

This equation may then be viewed as an implicit ODE for $\mathbf{V}$ and $\dot{\mathbf{V}}$.

4.1. Null Boundary Conditions

The broad problem class defined in Section 2 includes systems of PDE that may contain equations with only first-order derivatives (such as a continuity equation), and hence have only one boundary condition. PDECHEB can be applied to such problems by using the other boundary position as an extra collocation point and by making use of the coupled ODE/PDE problem interface. Consider the case of one PDE and suppose that there is no boundary condition at $x = b$. An extra ODE variable V is then defined at the spatial coupling point $\xi = b$ by the algebraic equation

$$V = R(b, t). \tag{4.3}$$

Consequently, V will be set equal to the numerically computed flux at the boundary during the integration. A *pseudo* boundary condition at $x = b$ is then defined by

$$\beta(b, t) = 1, \quad \gamma(b, \cdots) = V. \tag{4.4}$$

On substituting these values into the second of the pair of Eqs. (3.12) and using the definition of V provided by (4.3), we see that this equation then reduces to the collocation equation at the point $x = b$. The idea can be extended to systems of first order equations and also used for periodic in space boundary conditions.

5. INTEGRATION USING THE METHOD OF LINES

The success of the method of lines in solving coupled systems of ordinary and partial differential equations lies in combining efficient and general spatial discretization methods with sophisticated ODE initial value problem integrators being used to perform the time integration. The essence of the method is

to spatially discretize a system of *NPDE* time-dependent partial differential equations with a spatial mesh of *NPTS* points and with *NV* coupled ordinary differential equations into a system of *NPTS***NPDE* + *NV* coupled ordinary differential equations of the form:

$$\mathbf{F}(\mathbf{U}, \dot{\mathbf{U}}, T) = 0, \mathbf{U}(0) = \mathbf{k}. \tag{5.1}$$

Each solution component of these equations defines either one component of the PDE solution at a single mesh point or one of the coupled ODE components.

5.1. Ordering of the ODE Solution Vector

The following convention is used by the PDECHEB software in ordering the ODE solution vector $\mathbf{U}(t)$ of Eq. (5.1). We assume that the system of *NPDE* PDE is discretized using *IBK* spatial breakpoints, a polynomial degree of *NPOLY* and that there are *NV* coupled ODEs. (In the description that follows, *NPOLY* rather than *N* will be used to denote the polynomial degree, as this has a less ambiguous meaning.) The PDE solution components are stored in the first *NPDE* × *NPTS* components of the vector $\mathbf{U}(t)$, and *NPTS* will be defined below. The ODE components are stored in the last *NV* components of $\mathbf{U}(t)$, i.e.,

$$\mathbf{U}_l(t) = V_m \quad \text{where} \quad l = NPDE \times NPTS + m,\ m = 1, \ldots, NV \tag{5.2}$$

and $\mathbf{V}$ is the solution of the coupled ODE system of dimension *NV*. In our case the value of *NPTS* is (*IBK* − 1)*(*NPOLY*) + 1 where *NPOLY* is the degree of the approximating polynomial used between each pair of spatial mesh points and where *IBK* is the number of breakpoints.

Using the above ordering, the system of ordinary differential equations in time defined by the Chebyshev C^0 collocation method (e.g., Eqs. (3.6), (3.12), and (3.13)) can equivalently be written as Eqs. (5.1) where the vector $\mathbf{U}(t)$ is defined by

$$\mathbf{U}(t) = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_{NEL} \\ \mathbf{V} \end{bmatrix}, \mathbf{U}_j = \begin{bmatrix} \mathbf{U}_{j,0} \\ \mathbf{U}_{j,1} \\ \vdots \\ \mathbf{U}_{j,N-1} \end{bmatrix}, j = 1, 2, \ldots, NEL - 1,$$

$$\mathbf{U}_{NEL} = \begin{bmatrix} U_{NEL,0} \\ \mathbf{U}_{NEL,1} \\ \vdots \\ \mathbf{U}_{NEL,N} \end{bmatrix}, \mathbf{V} = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{NV} \end{bmatrix} \tag{5.3}$$

$N = NPOLY$, and $\mathbf{U}_{j,i} = \mathbf{U}(x_{j,i}, t)$ where

$$x_{j,i} = \tfrac{1}{2}\left(X_{j+1} + X_j + (X_{j+1} + X_j)\cos\left(\frac{N-i}{N}\pi\right)\right) \tag{5.4}$$

and the vector $\mathbf{U}(x, t)$ is defined as in Eq. (2.1). The total number of ordinary differential/algebraic equations for a system of NPDE PDEs, NV coupled ODEs, IBK breakpoints and using a piecewise polynomial of degree NPOLY is given by *NEQ* where

$$NEQ = NPDE^*(\text{IBK} - 1)^*(\text{NPOLY}) + NPDE + NV. \tag{5.5}$$

5.2. Initial Conditions

The initial condition for $\mathbf{U}(t)$ is found by evaluating the function $k(x)$ at the transformed Chebyshev points in each element and by using the initial condition for $\mathbf{V}$, see Eqs. (2.5) and (2.6). The value of $\mathbf{U}(0)$ is thus defined by substituting

$$U_{j,i} = k(x_{j,i}), \quad j = 1, \cdots, NEL, \qquad i = 0, \ldots, N \quad \text{and}$$
$$[\mathbf{V}(0)]_l = [\mathbf{k}_v]_l, \qquad l = 1, \cdots, NV. \tag{5.6}$$

into Eqs. (5.3).

5.3. Error Control in the Method of Lines

The advantage of using high-order spectral spatial discretization methods is that high accuracy can be achieved using a small number of spatial mesh points, see Berzins and Dew [3, 4] and the recent monograph by Boyd [6]. Once the spatial discretization method has been chosen, it is desirable to integrate the ODE system in time with just sufficient accuracy so that the temporal error does not significantly corrupt the spatial accuracy. However, in most existing software based on the method of lines, the standard procedure is to control the local time error per step with respect to a supplied accuracy tolerance which is, in general, independent of the spatial discretization error and also of the global time error in the computed solution.

One solution to this problem is adopted in the software of Schonauer [22] and an alternative, but still experimental, approach is the time error control of Berzins et al.; see Berzins [27] and the references therein. Both methods have been used to control the temporal error so that it is dominated by the spatial error. This approach is all the more important in the case of high-order spectral methods, as the high spatial accuracy achieved makes it difficult to select beforehand a local error time integration tolerance that is fine enough to allow the accuracy of the spatial discretization to be observed yet coarse enough to allow the time integration to be efficient.

In most of the codes available for solving time dependent ODEs, including those used here, the routines attempt to control the local time integration error in the computed solution with regards to an accuracy tolerance supplied by the user, TOL. The i.v.p. to be solved is given by Eq. (5.1) with the true solution $\{\mathbf{U}(t_n)\}_{n=0}^{p}$ approximated by $\{\mathbf{V}(t_n)_{n=0}^{p}$ at a set of discrete times $0\text{-}t_0 < t_1 < \cdots < t_p = t_e$ by a time integration method with requested absolute local error accuracy, TOL. The local solution on $[t_n, t_{n+1}]$, $\mathbf{y}_{n+1}(t, \text{TOL})$, is the solution of the i.v.p.

$$\mathbf{F}(\mathbf{y}_{n+1}(t, \text{TOL}), \mathbf{y}_{n+1}'(t, \text{TOL}), t), \mathbf{y}_{n+1}(t_n, \text{TOL}) = \mathbf{V}(t_n). \tag{5.7}$$

The local error per step at t_{n+1} is given by

$$\mathbf{le}_{n+1}(\mathrm{TOL}) = \mathbf{V}(t_{n+1}) - \mathbf{y}_{n+1}(t_{n+1}, \mathrm{TOL}). \tag{5.8}$$

In general, the time global error is not even proportional to the local error tolerance, TOL, [27]. This makes it difficult to select a local error per step tolerance that will ensure that the spatial error dominates without some experimentation.

6. OUTLINE OF THE PDECHEB SOFTWARE

This section describes the main components of the PDECHEB software and how they may be combined with a suitable integrator for initial value differential algebraic equations. The three main components of the PDECHEB software are summarized here; the complete interfaces are listed in the accompanying algorithm of Berzins and Dew [4].

6.1. INICHB

This is the initialization routine that the user must call before starting the time integration. This routine checks the user's choice of NPOLY (the degree of the approximating polynomial) and XBK(IBK) (the breakpoint array), computes certain vectors and matrices needed by the discretization method, and stores all the information that has to be passed to the discretization routine, PDECHB, in the work space array WKRES(NWKRES). The INICHB routine also generates the initial solution vector for the ODE integrator Y(NEQ) and an array of spatial mesh points X(NPTS) at which the PDE solution values are computed.

6.2. PDECHB

In order to use an integrator for the solution of d.a.e. problems such as (5.1), the user is generally required to write a simple calling program for the integrator. The user must also specify initial values for the ODE solution vector $\mathbf{U}$ and must specify a FORTRAN subroutine that defines the residual of Eq. (5.1) when called with approximate values of $\mathbf{U}$ and $\dot{\mathbf{U}}$ that are estimated by the integrator. This routine is repeatedly called by the ODE integrator which supplies an approximate solution vector U(NEQ) and its approximate time derivative vector UDOT(NEQ). The PDECHB subroutine computes the residual vector RES(NEQ) which is obtained by substituting the vectors U and UDOT into the d.a.e. system being solved, i.e., for Eq. (5.1) that results when the collocation software is used to discretize a mixed PDE/ODE problem. This residual is defined by

$$\mathbf{RES} = \mathbf{F}(\mathbf{U}, \dot{\mathbf{U}}, T). \tag{6.1}$$

PDECHB computes this residual by using the workspace information generated by INICHB and a few simple subroutines that the user supplies to define the PDE/ODE system being solved.

6.3. INTERC

One of the tasks that commonly needs to be performed after the ODE integrator has returned the solution at the required time to postprocess the results. This is necessary because only the solution values at the spatial

mesh points are returned by the integrator. For this reason, a routine is provided to allow the user to generate the value of the solution between these mesh point values by interpolation. In common with PDECHB, information about the discretization is passed to this routine by means of the workspace initialized by INICHB.

We now describe how these components may be linked with an integrator for the solution of ODE of the form of Eq. (5.1).

6.4. Choice of Time Integrator

The very general class of PDE problems discretized by the PDECHEB software and the general systems of ODEs that must be integrated makes it necessary to use the most general-purpose integrator that is currently available. The approach that we have adopted is therefore to describe how the software introduced above can be used with the DASSL integrator of Petzold [18]. There are three reasons for using DASSL.

(1) DASSL is one of the most general widely available codes for solving ODEs of the form of Eq. (5.1). The code is well tested and is based on the well-known backward differentiation formulas of Gear [11].

(2) A number of other codes exist or are being developed to solve the same problem class, see Brenan et al. [18] Ch. 18. Examples of such codes are the DAEINT code of Morrison [17] and the code outlined by Rheinboldt [19]. The description provided here of how to use the discretization software with DASSL should make it relatively simple to use the software with other integrators.

(3) The interface to DASSL is particularly convenient for use with the method of lines as the routine—generic name RESID—to define the residual of Eq. (5.3) and has the simple form:

```
      SUBROUTINE RESID ( T, Y, YDOT, RES, IRES, WKRES, IWKRES)
      INTEGER NEQ, IWKRES (1), IRES
      DOUBLE PRECISION T, Y(1), YDOT(1), RES(1), WKRES(1)
        DO 10 I = 1, NEQ
            set-RES(I) to the Ith component of the residual RES,
            as in Eq. (5.3)
10      CONTINUE
      RETURN
      END
```

The workspaces WKRES and IWKRES are not used by the DASSL integrator, but are present solely for the use of the RESID routine. In our case the work space WKRES contains the discretization information provided by the initialization routine INICHB. The workspace IWKRES is not used.

6.5. An Overview of Using DASSL and PDECHEB

There are three main steps in writing a program to use PDECHEB with the DASSL integrator. These steps are sketched out in Figure 1.

The first step consists of defining the workspaces and parameters needed by DASSL and the collocation routines and in calling the initialization routine INICHB. The user must define the number (IBK) and positioning of the breakpoints (the array XBK(IBK)) and the degree of approximating

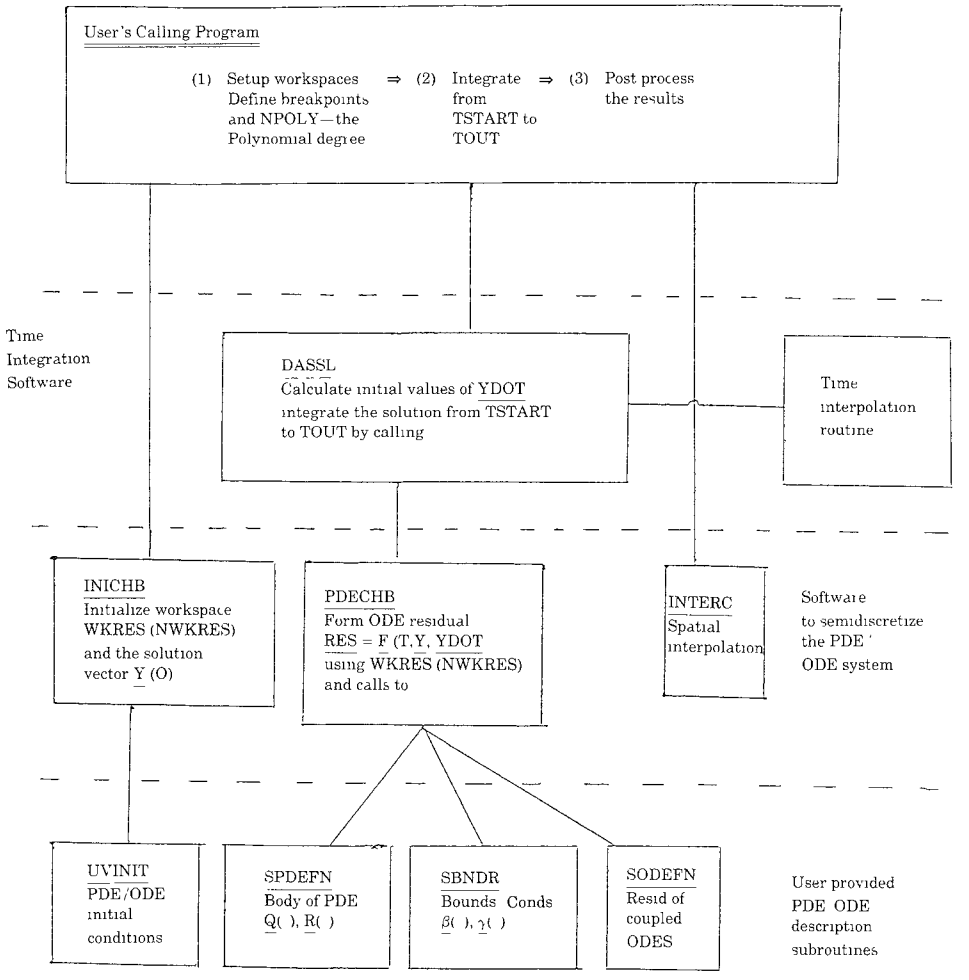

Fig 1. An overview of using DASSL and C^0-Collocation.

polynomial (NPOLY). Given the user-defined number of PDEs (NPDE) and the number of coupled ODEs (NV), the number of ODEs integrated by DASSL (NEQ) is given by Eq. (5.6). The size of the collocation workspace (WKRES(NWKRES)) can also be calculated. The DASSL workspaces Y, YDOT, RWORK, and IWORK can also be declared. The call to the routine INICHB can then be made. Precise details of the call to INICHB are provided by Berzins and Dew [4].

The second step is the call to DASSL to integrate the PDE/ODE system from time T to TOUT. The form of the ODE system solved by DASSL is provided by the discretization routine PDECHB, which in turn uses the workspace provided by the INICHB routine (WKRES(NWKRES)) and the user supplied subroutines that define the form of the coupled ODE/PDE system.

The final step consists of postprocessing the solution. The user may first call the time interpolation routine of DASSL to recover the solution at the

required time level and then call the space interpolation routine INTERC to obtain any solution values required at points that are not at the spatial mesh points, as supplied in the array X(NPTS).

The following skeleton program illustrates the steps outlined above for the case of two PDEs with a coupled ODE and spatial coupling point. Eleven equally spaced breakpoints are used with a cubic polynomial being used (NPOLY = 3) to approximate the PDE solution between each pair of breakpoints.

```
C  C0  COLLOCATION PARAMETERS.
      PARAMETER (IBK = 11, NEL = IBK - 1, NPDE = 2,      NV = 1,
     1           NPOLY  = 3, NPTS = NEL*NPOLY + 1,  NXI = 1,
     2           NEQ    = NPTS*NPDE + NV,
     3           NWKRES= 2*(NPOLY + 1)*(NPOLY + NEL + 2)
                 +2 + NV +
     4                   NPDE*(7*(NPOLY + 1 + NXI) + 8),
C  DASSL TIME INTEGRATION PARAMETERS.
     5           MAXORD = 5, LRW = 40*(MAXORD + 4)*NEQ +
                 NEQ**2,
     6           LIW = 20 + NEQ)
      INTEGER IWORK(LIW), INFO(15), IBAND, M, ITIME, I, IDID,
     1        IRESWK, IDEV
      DOUBLE PRECISION XBK(IBK), X(NPTS), X(NPTS), Y(NEQ), ATOL,
     1        WKRES(NWKRES), RWORK(LRW), XI(NXI), T, TOUT,
             RTOL
      EXTERNAL PDECHB, DGEJAC
      T = 0.0D0
      M = 0
      IDEV = 6
      DO 10 I = 1, IBK
 10     XBK(IBK) = 0.1D0*(I - 1)
C        INITIALIZE THE PDE WORKSPACE.
      CALL INICHB (NEQ, NPDE, NPTS, X, Y, WKRES, NWKRES, M, T,
     1     IBAND, ITIME, XBK, IBK, NEL, NPOLY, NV, NXI, XI,
          IDEV)
C        SET UP DASSL PARAMETERS
      DO 20 I = 1, 11
 20     INFO(I) = 0
C  REMOVE COMMENT C FROM NEXT LINE FOR BANDED MATRIX
     OPTION
C     INFO(6) = 1
      ATOL = 1.0D - 4
      RTOL = 1.0D - 3
C       BANDED MATRIX OPTION WHEN INFO(6) = 1
      IF(  INFO(6) .EQ. 1)THEN
           IWORK(1) = IBAND
           IWORK(2) = IBAND
      END  IF
      TOUT = 0.1DO
      CALL DASSL(PDECHB, NEQ, T, Y, YDOT, TOUT, INFO, RTOL, ATOL,
     1           IDID, RWORK, LRW, IWORK, LIW, WKRES, IRESWK,
                 DGEJAC)
C
C       INSERT POSTPROCESSING HERE E.G. SPACE INTERPOLATION.
C
      STOP
      END
```

6.6. Choice of Linear Algebra Routines When Solving PDE Problems

One of the limitations of using DASSL is that in the most widely available original version of DASSL only full matrix or banded matrix routines are available. There is, however, work in progress to develop versions of DASSL that use sparse matrix techniques and the matrix-free GMRES iterative solver, see Brenan et al. [18, p. 137]. In the case of a problem consisting only of PDEs, the Jacobian matrix of the d.a.e. system (5.1) generated by PDECHEB is block diagonal. The form of the block diagonal system and its solution are both discussed by Keast et al. [12]. Although it would be possible to amend DASSL to include the block-diagonal matrix routines described by Keast et al. [12], and this would probably result in similar speed-ups to those reported by Keast and Muir [13] with PDECOL, we have chosen not to modify the DASSL code by including these routines. The main reason for this is that the coupled ODE/PDE problem class that we are trying to solve may result in a bordered block-diagonal Jacobian matrix, which may have its rightmost NV columns full and its bottom NV rows full, where NV is the number of coupled ODE variables. The most appropriate linear algebra routines for the LU decomposition of such matrices are probably sparse matrix routines. The sparsity pattern of a typical matrix is illustrated in Figure 2, which shows the sparsity pattern of a matrix for the case when there are two PDEs, a polynomial of degree 3 is used in each of the three elements and there are two coupled d.a.e.s.

In order to use banded matrix routines with DASSL, it is necessary to define the upper (MU) and lower (ML) half bandwidths of the ODE system being integrated and to supply them as optional inputs to DASSL. The size of these bandwidths when the C^0 collocation discretization is used is given by the parameter IBAND, whose value is defined by the setup routine INICHB. The values of this parameter is NPDE*(NPOLY + 1) − 1 when there are no coupled ordinary differential equations present. The convention used by the discretization routines in ordering the ODE solution vector (described in Section 5.1) means that banded matrix routines cannot be used with coupled ODE/PDE problems.

6.7. Analytic Jacobian Matrices and DASSL

The DASSL code has an option by which the user can supply analytic Jacobian matrices. In the case of the system of equations (5.1), the Jacobian matrix that must be supplied to DASSL has the form

$$\frac{\partial \mathbf{F}}{\partial \mathbf{U}} + cj\frac{\partial \mathbf{F}}{\partial \dot{\mathbf{U}}}. \tag{6.2}$$

In the case of PDECHEB the partial derivatives in Eq. (6.2) depend on the partial derivatives of the vector functions $\mathbf{Q}()$, $\mathbf{R}()$, and $\mathbf{F}()$ defined in Eqs. (2.1) and (2.2) and also on the form of the spatial discretization method. In order to supply the analytic Jacobian to DASSL it is necessary for the user to specify explicitly the dependence of all these functions on all their arguments except x and t. In order to do this, the user will have to supply the eight

Consider the case of two p.d.e.s (NPDE = 2, variables $U_1(x,t)$ and $U_2(x,t)$) with three elements, two coupled d.a.e.s (variables V_1 , V_2) and a cubic polynomial (NPOLY = 3) in each element. It is assumed that there are two coupling points ξ_1 and ξ_2 one in the first and one in the third spatial elements.

```
boundary      a b a b a b a b                                   e f
conditions    c d c d c d c d                                   g h
              a b a b a b a b                                   e f
              c d c d c d c d                                   g h   pde eqns
              a b a b a b a b                                   e f   element 1
              c d c d c d c d                                   g h
breakpoint    a b a b a b a b a b a b a b                       e f
conditions    c d c d c d c d c d c d c d                       g h
                        a b a b a b a b                         e f
                        c d c d c d c d                         g h   pde eqns
                        a b a b a b a b                         e f   element 2
                        c d c d c d c d                         g h
breakpoint              a b a b a b a b a b a b a b a b e f
conditions              c d c d c d c d c d c d c d g h
                                  a b a b a b a b e f
                                  c d c d c d c d g h
                                  a b a b a b a b e f   pde eqns
                                  c d c d c d c d g h   element 3
boundary                          a b a b a b a b e f
conditions                        c d c d c d c d g h
              i j i j i j i j     m n m n m n m n p q    coupled
              r s r s r s r s     m n v w v w v w x y    dae eqns

                pde variables                           dae variables
```

The letters in the diagram are the possible non-zero entries in the Jacobian matrix which may be characterised in the following way

```
a , b  represent the dependence of pde 1 on U1(x,t)  and U2(x,t) respectively
c , d   ..  ..   ..  ..   ..  .. pde 2 .. U1(x,t) and U2(x,t) respectively .
e , f   ..  ..   ..  ..   ..  .. pde 1 .. V1  and V2  respectively .
g , h   ..  ..   ..  ..   ..  .. pde 2 .. V1 and V2  respectively .
i , j   ..  ..   ..  ..   ..  .. dae 1 .. U1(x,t) and U2(x,t)  defined at ξ1 .
r , s   ..  ..   ..  ..   ..  .. dae 2 .. U1(x,t) and U2(x,t)  defined at ξ1 .
m , n   ..  ..   ..  ..   ..  .. dae 1 .. U1(x,t) and U2(x,t) defined at ξ2
v , w   ..  ..   ..  ..   ..  .. dae 2 .. U1(x,t) and U2(x,t) defined at ξ2
p , q   ..  ..   ..  ..   ..  .. dae 1 .. V1 and V2 respectively.
x , y   ..  ..   ..  ..   ..  .  dae 2 .. V1 and V2 respectively.
```

It should be noted that a third coupling point in the second element would have meant that the bottom two rows of the sparsity pattern were (potentially) full.

Fig. 2. Example sparsity pattern for PDECHEB Jacobian.

NPDE by NPDE matrices:

$$\frac{\partial \mathbf{Q}}{\partial \mathbf{u}}, \frac{\partial \mathbf{Q}}{\partial \mathbf{u}_x}, \frac{\partial \mathbf{Q}}{\partial \mathbf{u}_t}, \frac{\partial \mathbf{Q}}{\partial \mathbf{u}_{xt}},$$

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}}, \frac{\partial \mathbf{R}}{\partial \mathbf{u}_x}, \frac{\partial \mathbf{R}}{\partial \mathbf{u}_t}, \frac{\partial \mathbf{R}}{\partial \mathbf{u}_{xt}}, \tag{6.3}$$

the four NPDE by NV matrices:

$$\frac{\partial \mathbf{Q}}{\partial v}, \frac{\partial \mathbf{Q}}{\partial \dot{v}}, \frac{\partial \mathbf{R}}{\partial v}, \frac{\partial \mathbf{R}}{\partial \dot{v}},$$

the five NV by NXI matrices:

$$\frac{\partial \mathbf{F}}{\partial \mathbf{u}^*}, \frac{\partial \mathbf{F}}{\partial \mathbf{u}_x^*}, \frac{\partial \mathbf{F}}{\partial \mathbf{R}^*}, \frac{\partial \mathbf{F}}{\partial \mathbf{u}_t^*}, \frac{\partial \mathbf{F}}{\partial \mathbf{u}_{xt}^*},$$

and the two NV by NV matrices:

$$\frac{\partial \mathbf{F}}{\partial v}, \frac{\partial \mathbf{F}}{\partial \dot{v}}.$$

For the user to supply *nineteen* such matrices is a complex and error-prone task. For this reason we have not supplied the means by which an analytic Jacobian matrix is made available to DASSL. Instead we have provided a dummy routine, DGEJAC, that is supplied in the call to DASSL in place of the analytic Jacobian routine.

6.8. Intermediate Output with DASSL

The normal mode of operation of DASSL is that it integrates from T to TOUT without providing any intermediate output. In many situations it is useful to have information at the end of every timestep. DASSL has an option to operate in this mode. This option is activated by specifying INFO(3) = 1 prior to the call to DASSL. After each successful step in this mode DASSL returns with IDID = 1. The intermediate output can then be generated and the integrator recalled. The situation is best summarized by the following fragment of code taken from the example of Section 6.5.

```
      TOUT = 0.1DO
C
C       SET INTERMEDIATE OUTPUT MDE
C
   30 INFO(3) = 1
      CALL DASSL(PDECHB, NEQ, T, Y, YDOT, TOUT, INFO, RTOL, ATOL,
     1           IDID, RWORK, LRW, IWORK, LIW, WKRES, IRESWK,
                 DGEJAC)
C
C       INSERT INTERMEDIATE MODE ( END OF EACH TIMESTEP )
C       POSTPROCESSING HERE
C
      IF ( IDID .EQ . 1)GOTO 30
C
C       INSERT END OF INTERVAL ( TOUT REACHED ) POSTPROCESS-
        ING HERE
C       E.G. SPACE INTERPOLATION.
C
```

It should also be noted that most good d.a.e. integrators have this mode of

operation available, so that in general there is no need to modify the integrator to provide end-of-timestep output facilities.

6.9. Error Estimation and Calculation

One of the main advantages of the spatial discretization method in this paper is that there is a wide range of methods available. This also makes it difficult to choose the optimal degree of polynomial and the optimum set of breakpoints. The general approach advocated here is that the smoother the problem solution, the higher the degree of polynomial that should be used. At the same time it is difficult for the user to determine how accurate the numerical method is when there is no analytic solution available. One solution to this problem is to have some means of estimating the global error as in the methods of Berzins [27] and Schonauer [22]. Unfortunately, in the case of spectral method of lines, the Chebyshev polynomial estimate of Berzins [27] is not yet proven for the very broad class of problems defined by Eqs. (2.1) to (2.5). Although in the future an estimate of this type will be used to control the global error, this is beyond the scope of this paper.

A widely applicable, but not particularly robust, technique for estimating the error is Chebyshev polynomial-based methods is suggested by Boyd [28]. The simple truncation error estimate suggested by Boyd [28] is as follows. Consider the Chebyshev polynomial coefficients of the solution as defined by Eq. (3.3). In the case when the coefficients $a_{j,i}(t)$ behave like i^{-k}, an estimate of the truncation error caused by using only the first N + 1 polynomial coefficients is some multiple of $a_{j,N}(t)$. Boyd also suggests that the rate of decrease of the coefficient size is a reasonable measure of the degree of polynomial to use. In other words, the coefficients of the higher degree polynomials should be small.

It is for these reasons that we have provided a means of computing the Chebyshev polynomial coefficients of the solution.

The calculation of the polynomial coefficients of the solution is performed by the matrix-vector multiplication described by Berzins and Dew [2] using the matrix Ω (see Eq. A2 in their paper). This matrix is stored in the first NPTL × NPTL elements of the array WKRES. Suppose that a system of NPDE PDEs is being solved with NEL spatial elements and a polynomial of degree NPOLY in each element. In this case for each PDE there are NPTL = NPOLY + 1 polynomial coefficients per element and suppose that these coefficients are to be stored in the array COEFF(NPDE, NEL, NPTL) with the coefficient of $T_l(x)$ for the Kth PDE and the Jth element stored in. COEFF(K, J, I). The code to compute these coefficients is the following.

```
C
      NPTL = NPOLY + 1
      DO 100 I = 1, NEL
C       ITH ELEMENT
C       IU IS THE COMPONENT OF SOLUTION VECTOR AT LHS
C       OF ELEMENT I
        IU = (I - 1)*(NPOLY)*NPDE + 1
        DO 80 IS = 1, NPTL
```

```
              DO 80 JK = 1, NPDE
                COEFF(JK, IS, I) = 0.0D0
   80         CONTINUE
              DO 90 IS = 1, NPTL
                DO 90 JS = 1, NPTL
                  DO 90 JK = 2, NPDE
                  COEFF(JK, IS, I) = COEFF(JK, IS, I) +
     1                               WKRES(IS + (JS - 1)*NPTL)*Y
                                     (IU + JS + JK - 2)
   90         CONTINUE
  100       CONTINUE
C
```

The coefficient $a_{J,N}(t)$ for the K-th PDE is then returned in the array element COEFF(K, J, N).

6.9.1 *Optional Subroutine Error*. In the case when the PDE being solved has an analytic solution it is useful to be able to calculate the maximum error at the mesh points and also the Chebyshev error norm. The optional routine (ERROR) to compute the error requires that the user can write a fixed name routine

```
      SUBROUTINE EXACT ( T, NPDE, NP, XP, US)
      INTEGER NPDE, NP
      DOUBLE PRECISION T, X(NP), US(NPDE, NP)
C     ON EXIT FROM THIS ROUTINE THE ARRAY US(NPDE, NP)
C     SHOULD CONTAIN THE SOLUTION FOR THE NPDE PDES
C     AT TIME T FOR THE SPATIAL MESH POINTS IN THE
C     ARRAY X(NP)
      RETURN
      END
```

The routine ERROR returns an estimate of the Chebyshev norm of the error as well as the maximum error at the grid points. The precise form of the interface to the subroutine ERROR is given by Berzins and Dew [4].

7. EXAMPLES OF THE USE OF PDECHEB

This section illustrates some of the different types of PDEs that can be integrated using PDECHEB and DASSL. This is achieved by describing the problem description routines for a coupled PDE/ODE problem and a PDE problem with a material interface and coupled ODEs. In the first case, the analytic solution is available to allow us to demonstrate the high accuracy, rapid convergence, and efficient means of solution that can be achieved by the Chebyshev polynomial discretization method employed by PDECHEB. Other example problems are provided with the code of Berzins and Dew [4].

The user specifies the coupled PDE/ODE system so that it can be spatially discretized by writing four short FORTRAN subroutines which must have *fixed* names. One of the routines UVINIT defines the initial conditions, two more SPDEFN and SBNDR define the form of the PDE and its boundary conditions. The final routine SODEFN defines the form of the coupled ODE system. The restriction on the names of the PDE description routines is forced by the use of DASSL which does not have a mechanism to pass the names of subroutines into the RESID routine.

It should be noted that one of the parameters to each of the routines SPDEFN, SODEFN and SBNDR is the integer IRES. This control parameter can be reset to force the DASSL integrator to take the following action.

IRES	Action Taken by Integrator	Usage
− 2	The integration is stopped.	A last resort to stop integration.
− 1	Illegal solution value—the integration step is retried.	Can be used to stop the integrator generating unphysical values.

An illustration of the use of IRES for a PDE problem is provided below, in conjunction with the moving boundary problem, to ensure that the moving boundary position is always nonnegative. DASSL then tries to reduce the step-size in order to avoid this condition.

7.1. Numerical Example One — A Coupled PDE / ODE System

This problem is a one-phase Stefan problem, see Furzeland [10]. It provides a simple example of how a coupled ODE/PDE system is specified. The PDE is defined by the Equations:

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial y^2}, 0 < y < V(t), \epsilon < t < 1.0$$

$$\frac{\partial U}{\partial y}(0, t) = -\exp(-t)$$

$$U(V(t), t) = 0 \quad \text{and} \quad \frac{\partial U}{\partial y}(V(t), t) = -\dot{V}(t)$$

on the moving boundary $V(t)$. The numerical solution is started at $t = \epsilon$ with the analytic solution

$$U(y, t) = \exp(t - y) - 1, V(t) = t.$$

The parameter ϵ is a small positive quantity, which is set to the accuracy requested in the time integration. The problem is rewritten by using the coordinate transformation, see Furzeland [10],

$$x(t) = y / V(t)$$

to fix the moving boundary at $x = 1$ for all t. The equations become

$$V^2 \frac{\partial U}{\partial t} - V\dot{V}x\frac{\partial U}{\partial x} = \frac{\partial^2 U}{\partial x^2}, x\epsilon(0, 1),$$

with boundary conditions

$$\frac{\partial U}{\partial x} = -V(t)\exp(t) \quad \text{at } x = 0 \quad \text{and} \quad U(1, t) = 0.$$

In addition, the function $V(t)$ is defined implicitly by the ODE

$$\frac{\partial U}{\partial x} = -\dot{V}(t)V(t)$$

at the point $x = 1$, which is the point at which the ODE is coupled to the PDE.

We now describe the forms of the subroutines the user must provide with reference to the above example.

SBNDR. This routine defines the boundary conditions. The vectors

$$\mathbf{U}, \mathbf{U}_x, \mathbf{U}_t, \mathbf{U}_{xt}, \text{ and } \mathbf{V}$$

in Eq. (2.3) are passed across to the SBNDR routine as the FORTRAN arrays

U(NPDE), UX(NPDE), UT(NPDE), UXT(NPDE), and V(NV)

The logical variable LEFT specifies whether or not the left boundary conditions should be supplied (LEFT = .TRUE.) or whether the right boundary conditions should be supplied (LEFT = .FALSE.). The purpose of this routine is to place the values of the functions $\gamma(\cdots)$ and $\beta(\cdots)$ in Eq. (2.3) in the FORTRAN arrays GAMMA(NPDE) and BETA(NPDE).

It should be noted there are three boundary conditions for this problem. Since we have two conditions involving PDE fluxes, we have chosen to specify these as the boundary conditions for the PDE part. The reason for this is that the fluxes will be evaluated using the weak form of the PDE, as shown in Eq. (3.12). The third equation $U(1, t) = 0$ is then specified as the coupled ODE equation even though it does not involve the ODE variable $V(t)$ (which is present in the PDE and in the boundary conditions). This approach is one of the many useful practical devices described by Schryer [21] in connection with coupled PDE/ODE problems.

```
C
      SUBROUTINE SBNDR(T, BETA, GAMMA, U, UX, UT, UTX, NPDE,
      LEFT, NV,
     1                          V, VDOT, IRES)
C     Specifies boundary conditions for Jth PDE in master Eq. (5.2) form
C
C       BETA(J, T)*R(X, T, U, U_X, U_T, U_XT, V, V̇) =
      GAMMA(J, T, U, U_X, U_T, U_XT, V, V̇)
C     At either X = A or X = B.
C     The form of the flux function R( ) is specified in SPDEFN.
C     For the moving boundary example:
C       U = - V(T)*EXP(T) at X = 0 and U =  - V(T)*V(T) at X = 1
C       X                                X
        INTEGER NPDE, NV
        DOUBLE PRECISION BETA(NPDE), GAMMA(NPDE), U(NPDE),
        UX(NPDE)
                        , UT(NPDE), UTX(NPDE), T, V(1), VDOT(1)
        LOGICAL LEFT
        BETA(1) = 1.0D0
        IF(LEFT)THEN
C     known flux b.c. at X = 0
          GAMMA(1) = -V(1)*DEXP(T)
        ELSE
C     known flux b.c. at X = 1, the moving boundary condition
          GAMMA(1) = -V(1)*VDOT(1)
        END IF
        RETURN
        END
```

SPDEFN. This routine supplies the values of the functions $Q_k(\cdots)$ and $R_k(\cdots)$ in the PDE definition of Eq. (2.1). The routine SPDEFN is called once for all mesh points between a pair of consecutive breakpoints. This means that the user must define the functions Q and R as in Eq. (2.1) for a set of spatial mesh points contained in the FORTRAN array X(NPTL) where

XBK(I) = X(1) < X(2) < $\cdots$ < X(NPTL) = XBK(I + 1)

where XBK(I) is the Ith breakpoint and I = $0,\cdots,$NEL $-$ 1. The following code is the required routine for the moving boundary problem.

```
C
      SUBROUTINE SPDEFN(T, X, NPTL, NPDE, U, UX, UT, UTX, Q, R, NV,
     1                  V, VDOT, IRES)
C  Form of the PDE for moving boundary problem as in Eq. (2.1).
C  The user supplies R and Q at the array of meshpoints X(NPTL).
C
C     V*V*U - V*V*X*U = U, X IN (0, 1).
C           T          X      XX
C     where V is the ODE variable.
        INTEGER NPDE, NPTL, NV, IRES
        DOUBLE PRECISION T, X, U(NPDE, NPTL), UX(NPDE, NPTL),
        UT(NPDE, NPTL)
                  , UTX(NPDE), Q(NPDE, NPTL), R(NPDE, NPTL), V(1),
                  VDOT(1)
        DO 10 J = 1, NPTL
          R(1, J) = UX(1, J)
   10     Q(1, J) = V(1)**2*UT(1, J) - X(J)*VDOT(1)*UX(1, J)*V(1)
        RETURN
        END
```

SODEFN. The user must supply a routine, named SODEFN, that evaluates the function $\mathbf{F}(\cdots)$ in Eq. (2.2). The arrays

$$\mathbf{U}^*, \mathbf{U}_x^*, \mathbf{R}^*, \mathbf{U}_t^*, \mathbf{U}_{xt}^*, \mathbf{V}, \text{ and } \dot{\mathbf{V}}$$

are passed across into the routine SODEFN as the FORTRAN arrays

UI, UXI, RI, UTI, UTXI, V, and VDOT

respectively. These arrays are all of dimension (NPDE, NXI) except for V and VDOT which are of dimension NV and hold the solution, flux, and derivative values at the coupling point vector ξ. The coupling points are held in the FORTRAN array XI(NXI). The user must write the subroutine so that it supplies the residual of the coupled ODE system in the array VRES(NV).

```
      SUBROUTINE SODEFN(T, NV, V, VDOT, NPDE, NXI, XI, UI, UXI, RI,
     1                  UTI, UTXI, VRES, IRES)
C  Definition of coupled ODE residuals in master Eq. (2.2) form,
      INTEGER NPDE, NXI, NV, IRES
      DOUBLE PRECISION T, XI(NXI), UI(NPDE, NXI), UXI(NPDE, NXI),
     1    RI(NPDE, NXI), UTI(NPDE, NXI), UTXI(NPDE, NXI), VRES(NV),
     2    V(NV), VDOT(NV)
C  the residual VRES(1) from moving boundary condition U = 0
C  this and the fixed b.c. at X = 1 may be interchanged
      VRES(1) = UI(1, 1)
C  ires can be reset to cope with illegal values of m.b. position V(1).
      IF(V(1) .LT. 0.0D0)IRES = -1
      RETURN
      END
```

Table I. Results for Model Stefan Problem

Accuracy	NEQ	NPOLY	NEL	ERROR	CPU	TOL
	10	2	4	3.29d-3	0.369	2.0d-6
Low	8	3	2	2.39d-3	0.385	2 0d-6
	6	4	1	1.15d-3	0 317	2.0d-6
	42	2	20	1.90d-6	5.85	2.0d-8
Medium	22	4	5	1.00d-6	2.52	2.0d-8
	8	6	1	5.00d-6	0.82	2.0d-8
	66	4	16	1.50d-8	18 4	1.0d-9
High	32	6	5	2.60d-9	6.16	1.0d-9
	11	9	1	1.50d-9	1.88	1.0d-9

UVINIT. The initial conditions for $\mathbf{U}(x, 0)$ and $\mathbf{V}(0)$ are supplied by the user in subroutine UVINIT.

```
      SUBROUTINE UVINIT( NPDE, NPTS, X, U, NV, V)
C     Routine for PDE initial values (start time is 0.1) at meshpoints X(NPTS)
C     and routine for initial values of auxiliary ODEs (if any)
      INTEGER NPDE, NPTS, NV
      DOUBLE PRECISION X(NPTS), U(NPDE, NPTS), TIME, V(NV)
      TIME = 0.1D0
      DO 10 I = 1, NPTS
10      U(1, I) = DEXP( TIME*(1.0D0 - X(I))) - 1.0D0
      V(1) = TIME
      RETURN
      END.
```

Note that in the case when there are no ODEs coupled to the PDEs, V and VDOT will be dummy vectors of length one. Similarly, if NXI is zero; that is, there are no coupling points between the ODE and the PDE, then all the arrays of length NXI in the call to SODEFN will be dummy arrays of length one.

The analytic solution to this problem can be used to illustrate the convergence properties of the higher order polynomial methods in PDECHEB. To assess the performance of the method, we have compared the accuracy achieved by using polynomials of different degree with the computer time taken. A summary of the experiments is given in Table I below where we have adjusted the degree of the polynomial and the number of equally spaced breakpoints so that the different methods achieve approximately the same maximum grid error (ERROR) at the end of the integration (where the error is largest). The parameter TOL is the relative and absolute error tolerance in the call to DASSL, NEQ is the number of ordinary differential equations integrated in time, NPOLY is the degree of the polynomial used in the space approximation, NEL is the number of spatial elements and CPU is the computer time in seconds taken on the Amdahl 5850 at Leeds University.

The numerical results show clearly that the accuracy is best achieved by using a single element and increasing the degree of the polynomial. Several points are worth noting. The first is that the coupled ODE/PDE form of the

above problem means that banded matrix routines cannot be used with DASSL. As the lower polynomial degree methods need to use more spatial elements than the high order methods to produce a solution to a given accuracy (and hence a larger ODE system, of size NEQ equations, must be integrated in time) the cost of forming and decomposing a Jacobian matrix which is proportional to O(NEQ^3) is much higher than for the high order methods that use a smaller value of *NEQ*. This penalizes the timing results in favor of the case when only one element is used. The second point is that we have obtained similar results on a wide variety of test problems with smooth solutions, e.g., see Berzins and Dew [2]. Our experience suggests that the remarks made by Babuska et al. [1] concerning the efficiency and accuracy of high order polynomial methods for steady-state problems also apply to time-dependent problems.

One difficulty peculiar to solving time-dependent problems with higher order polynomial methods is that as the degree of the polynomial is raised it becomes increasingly difficult to get enough accuracy from the ODE integrator so that the space error dominates. For this reason we feel that it is particularly important to have an integration algorithm that balances the space and time errors, e.g., Schonauer et al. [22], and provides an estimate of the overall error such as that used by Berzins and Dew [3] and Schonauer et al. [22, 23].

7.2. Numerical Example 2 — Pool Evaporation Problem

This section provides an example of a nonstandard problem that can be solved very effectively using the PDECHEB discretization and the DASSL integrator. The problem concerns the rate of evaporation of vapor from a pool of liquid of length one meter. Above the pool a constant (i.e., nontime varying) wind blows. There is a viscous sublayer above the pool of height $\bar{x}$ and above that is a "windy" region in which the concentration of vapor diminishes until it is negligible at a height of about $10^3\ \bar{x}$. In order to apply the method of lines to this problem, we take the spatial variable as being the height above the pool and integrate across the length of the pool.

The governing PDE for the vapor concentration $U(x, t)$ in the viscous sublayer is

$$6.81 \times 10^3 x \frac{\partial U}{\partial t} = \frac{\partial}{\partial x}\left(8.65 \times 10^{-6} \frac{\partial U}{\partial x}\right),\ x \in [0, \bar{x}) \tag{7.1}$$

and in the turbulent region above the viscous sublayer,

$$(0.7717 \log(x) + 9.313)\frac{\partial U}{\partial t} = \frac{\partial}{\partial x}\left(0.1297 x \frac{\partial U}{\partial x}\right),\ x \in (\bar{x}, 1.0] \tag{7.2}$$

and $\bar{x}$ is a fixed internal boundary defined by $\bar{x} = 5.08 \times 10^{-4}$.

The initial condition is

$$U(x, 0) = 0,\ x \in [0, 1] \tag{7.3}$$

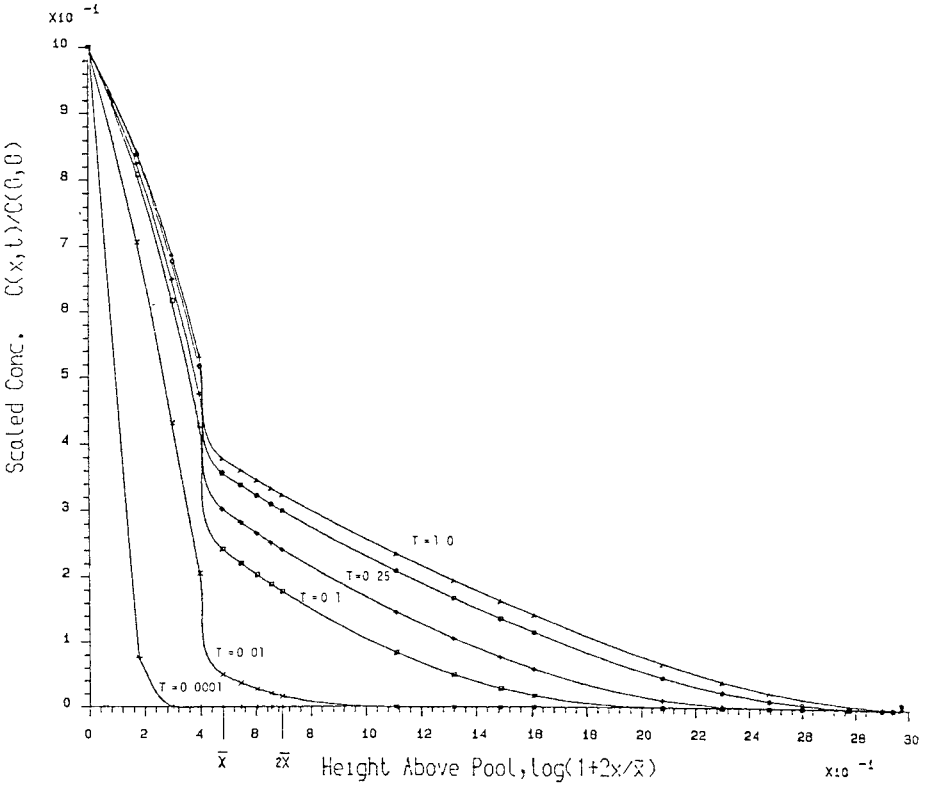

Fig. 3. Vapor concentrations for the pool evaporation problem.

and the boundary conditions are

$$U(0,t) = 0.038475 \text{ and } \frac{\partial U}{\partial x}(1,t) = 0. \tag{7.4}$$

The interface between the viscous sublayer and the turbulent region forces one of the meshpoints to be placed at $\bar{x}$. As most of the spatial variation occurs close to this point, further breakpoints were placed at $0.5\bar{x}$, $1.5\bar{x}$, $2\bar{x}$, $11\bar{x}$, and $121\bar{x}$, thus giving seven breakpoints in all. Figure 3 shows how the vapor concentration varies at a number of discrete times.

The problem has the following nonstandard feature. As a check on the accuracy of the numerical solution, it is important to compare the rate of evaporation $Q_1(t)$ at the surface of the pool with the quantity of vapor which passes above a given point in the pool $Q_2(t)$, where $Q_1(t)$ and $Q_2(t)$ are defined by

$$Q_1(t) = -7.934 \times 10^{-7} \int_0^t \frac{\partial U}{\partial x}(0,t)\, dt \tag{7.5}$$

and

$$Q_2(t) = 9.4175 \times 10^{-2} \int_0^1 p(x)U(x,t)\,dx \tag{7.6}$$

where the function $p(x)$ is defined by

$$p(x) = 6.81 \times 10^3 x,\ x \in [0, \bar{x}) \tag{7.7}$$

$$= (0.7717 \log(x) + 9.313),\ x \in (\bar{x}, 1.0]. \tag{7.8}$$

This comparison is most easily achieved by defining an extra coupled ODE for the rate of evaporation

$$\frac{dQ_1}{dt} = -7.934 \times 10^{-7} \frac{\partial U}{\partial x}(0,t). \tag{7.9}$$

(An alternative approach would be to use DASSL in one-step mode and to extract the values of $\partial U/\partial x$ at the end of each time step and to use quadrature in time to evaluate $Q_1(t)$.) Equation (7.6) is approximated by using Clenshaw–Curtis quadrature rule in space, to get

$$Q_2(t) = 9.4175 \times 10^{-2} \sum_{j=1}^{4} \frac{2}{X_{j+1} - X_j} \left(\sum_{i=1}^{NPOLY+1} p(X_{j,i})U(x_{j,i}, t)\lambda_i \right) \tag{7.10}$$

where $NPOLY$ is the degree of the polynomial used in each interval, X_j, $j = 1, \ldots, 5$ are the breakpoints, and $x_{j,i}$ are the points defined by Eq. (5.4). The coefficients λ_i are the Clenshaw–Curtis quadrature weights for the interval $[-1, 1]$ which may be accessed by the user from the COMMON block:

COMMON / SCHSZ6 / CCRULE (50)

where CCRULE is a DOUBLE PRECISION array whose Ith component contains λ_I for $I = 1, \text{NPOLY} + 1$. The accuracy of the numerical solution can then be monitored continuously by defining a new variable $Q_3(t)$ (which corresponds to the vapor discrepancy) by

$$Q_3(t) = Q_2(t) - Q_1(t). \tag{7.11}$$

Equations (7.9), (7.10), and (7.11) are then integrated with the PDE as a mixed PDE/ODE system. The initial conditions for the new variables are $Q_i(0) = 0$ for $i = 1, 2, 3$. The values of the discrepancy $Q_3(t)$ were tabulated at the same time levels as those used by Berzins et al. [5] with the SPRINT finite difference code SPDIFF and the results compared in Table II. Also shown in Table II are the results obtained by changing the degree of the polynomial to 12 to provide a high accuracy solution for comparison. In the table below CPU is the CPU time on the Amdahl 5850 and TOL is the local error tolerance (both relative and absolute).

The results with the SPDIFF code were obtained by using a different integrator based on the backward differentiation formula and by using sparse

Table II. Vapor Discrepancy for Pool Problem

Code	NPOLY	Mesh points	Max $\lvert Q_3 \rvert$	$Q_1(1)$	CPU	TOL
SPDIFF	01	81	2.615d-7	4.26d-4	28.86	1.0d-5
PDECHEB	03	22	1.704d-7	4.58d-4	2.59	1.0d-5
SPDIFF	01	201	6.255d-8	4.26d-4	259.0	1.0d-9
PDECHEB	12	85	1.297d-11	4.60d-4	100.0	1 0d-9

matrix software to form and decompose the Jacobian matrix, rather than the full matrix techniques employed with DASSL. The results in Table II illustrate the power and computational efficiency of the high order polynomial formulas in PDECHEB when solving problems with smooth solutions. Although the computational cost of using PDECHEB is potentially high because full matrix routines must be used, compared to the reduced cost of using sparse matrix routines when SPDIFF is used, the accuracy of the discretization methods in PDECHEB compensates for this by allowing a small number of equations to be integrated in time. The discrepancies between the values of $Q_1(1)$ may be attributed to the different quadrature rules used to approximate (7.6) and the different space derivative approximations in (7.9).

The convention used by the discretization routines in ordering the ODE solution vector is that the coupled ODE components are stored after the PDE components. Consequently, the penultimate ODE in time, Eq. (7.9), depends on the first NPOLY + 1 ODE variables (the concentrations in the first interval which are used to form $\partial U/\partial x(0, t)$). In the case of Eq. (7.10), the variable $Q_2(t)$ depends on all the PDE solution values at the mesh points. The result of this coupling is to destroy the banded structure of the ODEs associated with the original PDE problem.

This example problem also illustrates how easily discontinuities in the function $Q(\cdots)$ at the breakpoints are dealt with by testing X(1) and X(NPTL) to determine in which element the functions $Q(\cdots)$ and $R(\cdots)$ must be evaluated. The following pseudocode shows how the SPDEFN routine for the above problem may be written. As the user-supplied routine SPDEFN is called element by element, it follows that

(1) if X(1) $< \bar{x}$ and X(NPTL) $\leq \bar{x}$, then Eq. (7.1) should be used in the above problem definition as both the left and right edges of the element lie to the left of $\bar{x}$.

(2) If, however, X(1) $\geq \bar{x}$ and X(NPTL) $> \bar{x}$, then Eq. (7.2) used in the above problem definition as both the left and right edges of the element lie to the right of $\bar{x}$.

```
      SUBROUTINE SPDEFN(T, X, NPTL, NPDE, U, UX, UT, UTX, Q, R, NV,
     1                  V, VDOT, IRES)
C  Form of the PDE for material interface problem as in Eq. (2.1).
C  The user supplies R and Q at the array of meshpoints X(NPTL).
      INTEGER NPDE, NPTL, NV, IRES
      DOUBLE PRECISION T, X, U(NPDE, NPTL), UX(NPDE, NPTL),
      UT(NPDE, NPTL)
```

```
     1    XBAR, UTX(NPDE, NPTL), Q(NPDE, NPTL), R(NPDE, NPTL), V(1),
          VDOT(1), C
          XBAR = 5.08D - 3
          IF(X(1) .LT. XBAR .AND. X(NPTL) .LE. XBAR) THEN
C           element to the left of the interface use Eq. 7.1
C           to define the functions Q and R.
          ELSE
C           element to the right of the interface use Eq. 7.2
C           to define the functions Q and R.
          ENDIF
        RETURN
        END.
```

The same approach can also be used with multiple material interfaces.

8. SUMMARY

The PDECHEB software based on the Chebyshev C^0 Collocation Method of Berzins and Dew [3] allows a wide range of Chebyshev polynomial approximations to be applied to many PDE problems in one space dimension. We have shown how the software can be used with an ODE integrator DASSL to solve different types of PDE problems. The numerical results have shown the highly accurate and efficient solutions that can be obtained by using this discretization method and the method of lines to solve PDE problems with smooth solutions. The PDECHEB code can be used to discretize PDEs for which the C^1 continuity of PDECOL is unsuitable.

ACKNOWLEDGMENTS

L. Graney of B. P. Research is to be thanked for providing the pool evaporation problem.

REFERENCES

1. Babuska, I., Zinkiewicz, O. C., Gago, J. and Oliveira, E. R. de A. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*. Wiley, New York, 1986.
2. Berzins, M., and Dew, P. M. A generalized Chebyshev method for parabolic p.d.e.s. in one space variable. *IMA J. Numer. Anal. 1*, (1981), 469–487.
3. Berzins, M., and Dew, P. M. A note on C^0 Chebyshev methods for parabolic p.d.e.s. *IMA J. Numer. Anal. 7* (1987), 15–37.
4. Berzins, M., and Dew, P. M. C^0 Chebyshev software for parabolic p.d.e.s. Algorithm submitted to *ACM Trans. Math. Softw.*, 1988.
5. Berzins, M., Dew, P. M., and Furzeland, R. M. Software tools for time-dependent equations. In *Simulation and Optimisation of Large Systems*, A. J. Osiadacz, Ed., Clarendon Press, Oxford, 1988, 35–50. (Proceedings of IMA Conference on Simulation and Optimisation of Large Systems, Sept. 1986. Reading.)
6. Canuto, C., and Quateroni, A. Variational methods in the theoretical analysis of spectral approximations. In *Spectral Methods for Partial Differential Equations*, R. G. Voigt, D. Gottlieb, and M. Y. Hussaini, Eds., SIAM, Philadelphia, Pa., 1984, 54–78.
7. Dunn Jr., R. J., and Wheeler, M. F. Some collocation Galerkin methods for two-point boundary value problems. *SIAM J. Numer. Anal. 13*, 5 (Oct. 1976), 720–733.
8. Carey, G. F., Humphrey, D., and Wheeler, M. F. Galerkin and collocation Galerkin methods with superconvergence and optimal fluxes. *Int. J. Numer. Meths. Eng. 17* (1981), 939–950.

9. Diaz, J. C. A collocation-Galerkin method for the two point boundary value problem using continuous piecewise polynomial spaces. *SIAM J. Numer. Anal. 14* (1977), 844-858
10. Furzeland, R. M. A comparative study of numerical methods for moving boundary problems. *J.I.M.A. 26* (1977), 411-429.
11. Gear, C. W. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall, Englewood Cliffs, N.J., 1971.
12. Keast, P , Fairweather, G., and Diaz, J C. A comparative study of finite element methods for the solution of second order linear two-point boundary value problems. Techn. Rep. 150/81, Dept. of Computer Science, Univ. of Toronto, Toronto, Ont , March 1981.
13. Keast, P., and Muir, P. EPDCOL: A more efficient PDECOL code. Tech. Rep. Dept of Mathematics Computing and Statistics, Univ. of Dalhousie, Halifax, N.S., 1987.
14. Leyk, Z. A C^0 collocation-like method for two point boundary value problems *Numer. Math. 49* (1986), 39-53.
15. Machura, M., and Sweet, R. A. A survey of software for partial differential equations. *ACM Trans. Math. Softw. 6*, 4 (Dec. 1980), 461-488.
16. Madsen, N., and Sincovec, R. F PDECOL: General collocation software for partial differential equations. *ACM Trans. Math. Softw. 5* (1978), 326-351.
17. Morrison, K. Optimal control of processes described by systems of algebraic-differential equations. Ph.D. thesis, Univ of London, 1984.
18. Brenan, K. E., Campbell, S. L., and Petzold, L. R. *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. North Holland, Amsterdam, 1989.
19. Rheinboldt, W. C Differential algebraic equations as differential equations on manifolds *Math. Comput., 43*, 168 (Oct. 1984), 473-482.
20 Schryer, N. Numerical solution of time-varying partial differential equations in one space variable. Computing Science Tech. Rep. 53, AT and T Bell Labs, Murray Hill, N.J., 1977.
21. Schryer, N. Partial differential equations in one space variable. Computing Science Tech. Rep 115, AT and T Bell Lab., Murray Hill, N.J., 1984.
22. Schonauer, W., Raith, K., and Glotz, G. The principle of the difference of difference quotients as a key to the self-adaptive solution of nonlinear partial differential equations. *Comput. Meth. Appl. Mech. Eng. 28* (1981), 327-359.
23. Schonauer, W., Raith, K., and Glotz, G. The SLDGL program package for the self-adaptive solution of nonlinear systems of elliptic and parabolic p.d.e.s. In *Advances in Computer Methods for P.D.E.s-IV*, R Vichnevetsky and R S. Stepleman, Eds IMACS, 1981, 117-125.
24. Ortega, J. M., and Voigt, R. G. Solution of P.D.E.s on vector and parallel computers. *SIAM Rev. 27* (1985), 149-240.
25. Hindmarsh, A. C. Current methods for large stiff o.d.e. systems. In *Numerical Mathematics and Applications*, R Vichnevetsky, and J. Vignes, Eds. North Holland, Amsterdam, (1986), 135-144.
26 Funaro, D. Domain decomposition methods for pseudo spectral approximations, Part 1 Second order equations in one dimension. *Numer. Math. 52* (1988), 329-344.
27. Berzins, M. Balancing space and time errors for spectral methods used with the method of lines for parabolic equations. Report 91.14, School of Computer Studies, The University of Leeds LS2 9JT, United Kingdom.
28. Boyd, J. P. *Chebyshev and Fourier Spectral Methods. Lecture Notes in Engineering, 49*, Springer-Verlag, 1989.

Received September 1987; revised March 1990; accepted April 1990