A METHOD OF LINES PACKAGE, BASED ON MONOMIAL SPLINE COLLOCATION, FOR SYSTEMS OF ONE DIMENSIONAL PARABOLIC DIFFERENTIAL EQUATIONS

THEODORE B. NOKONECHNY, AND PATRICK KEAST
Department of Mathematics, Statistics and Computing Science, Dalhousie University
Halifax, Nova Scotia, B3H 4J5, Canada
E-mail: nokon@cs.dal.ca, keast@cs.dal.ca

and

PAUL H. MUIR
Department of Mathematics and Computing Science, Saint Mary's University
Halifax, Nova Scotia, Canada
E-mail: muir@cs.dal.ca

ABSTRACT

We consider systems of parabolic partial differential equations in one space variable, for which we describe a method of lines algorithm based on monomial spline collocation for the discretization of the spatial domain. While the usual application of this technique transforms the system of partial differential equations into a system of time dependent ordinary differential equations which can be integrated by standard initial value solvers, our approach leads to coupled systems of differential-algebraic equations. These equations are solved using a well known package, DASSL, which we have modified to take advantage of the special structure of the Jacobians which arise.

1. The Problem Class

In this paper we consider systems of $NPDE$ parabolic partial differential equations (PDEs). The systems considered are of the form

$$\frac{\partial u}{\partial t}(t,x) = f(t,x,u(t,x),u_x(t,x),u_{xx}(t,x)), \tag{1}$$

for $x_a < x < x_b$, $t_0 < t \le t_{out}$, with initial conditions at $t = t_0$ given by

$$u(t_0,x) = u_0(x), \quad x_a \le x \le x_b, \tag{2}$$

subject to the separated boundary conditions at $x = x_a$ and $x = x_b$ given by

$$\begin{aligned} b_{x_a}(t,u(t,x_a),u_x(t,x_a)) &= 0, \quad t_0 \le t \le t_{out}, & (3)\\ b_{x_b}(t,u(t,x_b),u_x(t,x_b)) &= 0, \quad t_0 \le t \le t_{out}, & (4)\end{aligned}$$

where

$$u(t,x) = [u_1(t,x),u_2(t,x),\ \ldots\ ,u_{NPDE}(t,x)]^T, \tag{5}$$

$$\begin{aligned}
u_x(t,x) &= \left[\frac{\partial u_1}{\partial x}(t,x), \frac{\partial u_2}{\partial x}(t,x),\ \ldots\ , \frac{\partial u_{NPDE}}{\partial x}(t,x)\right]^T , && (6)\\
u_{xx}(t,x) &= \left[\frac{\partial^2 u_1}{\partial x^2}(t,x), \frac{\partial^2 u_2}{\partial x^2}(t,x),\ \ldots\ , \frac{\partial^2 u_{NPDE}}{\partial x^2}(t,x)\right]^T , && (7)\\
f(t,x,u,u_x,u_{xx}) &= \left[f_1(t,x,u,u_x,u_{xx}),\ \ldots\ , f_{NPDE}(t,x,u,u_x,u_{xx})\right]^T , && (8)\\
u_0(x) &= \left[u_{0,1}(x), u_{0,2}(x),\ \ldots\ , u_{0,NPDE}(x)\right]^T , && (9)\\
b_{x_a}(t,u,u_x) &= \left[b_{x_a,1}(t,u,u_x),\ \ldots\ , b_{x_a,NPDE}(t,u,u_x)\right]^T , && (10)\\
b_{x_b}(t,u,u_x) &= \left[b_{x_b,1}(t,u,u_x),\ \ldots\ , b_{x_b,NPDE}(t,u,u_x)\right]^T . && (11)
\end{aligned}$$

In (8), (10), and (11), for brevity, we suppress the dependance on t, x of $u = u(t,x)$, $u_x = u_x(t,x)$, and $u_{xx}(t,x)$. The domain and range of the individual elements of (5)-(11) are

$$u_i(t,x),\ \frac{\partial u_i}{\partial x}(t,x),\ \frac{\partial^2 u_i}{\partial x^2}(t,x) : [t_0,\infty)\times[x_a,x_b]\rightarrow \mathbf{R}, \qquad (12)$$

$$f_i(t,x,u,u_x,u_{xx}) : [t_0,\infty)\times[x_a,x_b]\times\mathbf{R}^{NPDE}\times\mathbf{R}^{NPDE}\times\mathbf{R}^{NPDE}\rightarrow \mathbf{R}, \qquad (13)$$

$$u_{0,i}(x) : [x_a,x_b]\rightarrow \mathbf{R}, \qquad (14)$$

$$b_{x_a,i}(t,u,u_x),\ b_{x_b,i}(t,u,u_x) : [t_0,\infty)\times\mathbf{R}^{NPDE}\times\mathbf{R}^{NPDE}\rightarrow \mathbf{R}, \qquad (15)$$

for $i = 1,\ldots,NPDE$. Each $u_i(t,x)$ is assumed to have sufficient differentiability appropriate for (1). All components of f in (8) and of b_{x_a} and b_{x_b} in (10)-(11) are assumed to be differentiable with respect to their arguments. The problem class includes all systems of parabolic PDEs in one space variable which have (fixed) separated boundary conditions. For a treatment of PDE systems which include a moving boundary and/or coupled ODEs, the reader is referred to Berzins and Dew[5]. It is assumed that the system (1)-(4) is such that the existence of a unique solution is assured.

Our code is not intended to solve problems which have rapidly changing boundary layers, since it uses a currentlycurrently fixed spatial mesh which is not suited to such problems. However, if this method is combined with a mesh refinement scheme, then boundary layers may be dealt with. Much recent work has been done on adaptive MOL algorithms which attempt to adjust the spatial discretization. Based on some estimate of the spatial and/or temporal errors, these methods attempt to adapt to the problem by changes in one or more of the time steps, the spatial mesh, and the order of approximation. Verwer et al[20,21] discuss a process called '*static-regridding*', in which the grid is adapted at discrete times. Lawson et al[12] and Berzins et al[6] describe a combined spatial and temporal approach. In Huang and Russell[10] a moving mesh procedure is used to control spatial errors, with the *number* of mesh points being kept fixed, while their *location* is adapted to the solution behaviour. Flaherty and Moore[9,16] use a combined order and spatial adaptivity. The above references, and the references therein, provide a good background to the literature of adaptive methods for parabolic differential equations.

We discuss a Method of Lines (MOL) approach to (1), in which the spatial variable is discretized, resulting in a system of Initial Value Ordinary Differential Algebraic Equations (ODE/DAEs). The spatial discretization technique we will employ will be collocation at Gaussian points in each subinterval of the given mesh. There are several MOL codes available, some based on finite differences for the spatial discretization, and others based on C^0 collocation, or collocation using B-Splines. Commercial packages which provide MOL codes include for example, the NAG library and IMSL. Among public domain codes available from netlib are, for example, PDECOL[14] and EPDCOL[11] (both using B-Splines), PDEONE[13] (which uses finite differences), and PDECHEB[5] (C^0 collocation), all of which are available through netlib from the Transactions on Mathematical Software collection. The paper by Carroll[7] includes a comparison of several one dimensional MOL codes. In this paper our code will be compared to EPDCOL, a modified form of PDECOL.

There are three principal components in the MOL approach to the solution of a system of parabolic equations, namely:

(a) The spatial discretization.

(b) The solver used for the resulting ODE/DAE system.

(c) The techniques used to solve the linear systems which arise.

For (a) and (c) we will use techniques not used in any other MOL code which is generally available. In Section 2 we describe the discretization which we use, and show that a system of DAEs is produced. In Section 3, we discuss the DAE system, and the Jacobian which arises in the solution of the nonlinear systems occurring in the time step. In Section 4 we look at the linear algebraic problems which arise. These have a special form and our package takes full advantage of this, using the package ABDPACK[15]. For (b) we use the program DASSL[19], in a modified form to allow interaction with ABDPACK. In Section 5, the user interface for the software is explained, and we show the results of applying the new software to a problem taken from the paper in which PDECOL[14] appeared.

2. The Spatial Discretization

The spatial discretization will be carried out by collocating at the Gauss-Legendre points. This is, in itself, nothing new; Gauss-Legendre collocation is used in both PDECOL and EPDCOL. However, both of these codes use B-Splines as the basis functions, whereas we will use a monomial basis. Our choice of monomial splines rather than B-splines[8] is motivated by earlier developments in software for boundary value ordinary differential equations. The widely used code COLSYS[1], developed about 20 years ago, makes use of B-splines. However, later investigation by Ascher et al[4] indicated that monomial splines were a preferred choice, and this led to the development of a modified versoin of COLSYS called COLNEW. This code uses monomials and is reported to give improved performance both in execution time and storage requirements.

2.1. The Basis Functions

First we describe the monomial spline basis used for the spatial discretization. The notation used is essentially that used in Ascher, Mattheij and Russell[2]. The spatial mesh $X = \{x_i\}_{i=1}^{NINT+1}$ is a partition of the interval $[x_a, x_b]$, defined by

$$x_a = x_1 < x_2 < \ldots < x_{NINT} < x_{NINT+1} = x_b. \tag{16}$$

We will use $\mathcal{M}_X^{\kappa,m}$ to denote the set of $C^{m-1}[x_a, x_b]$ monomial splines of order κ (degree $(\kappa - 1)$) defined on the mesh X. Thus

$$\mathcal{M}_X^{\kappa,m} = \{v \in C^{m-1}[x_a, x_b] : v|_{[x_i,x_{i+1}]} \in \mathcal{P}_\kappa,\ i = 1, \ldots, NINT\}, \tag{17}$$

where $\mathcal{P}_\kappa$ is the set of all polynomials of order κ or less, and $\kappa > m \geq 0$. The dimension of the space $\mathcal{M}_X^{\kappa,m}$ is $\kappa \cdot NINT - m(NINT - 1) = (\kappa - m)NINT + m$. In our code we assume that $m = 2$, but for generality we keep m as a variable here.

In order to define the collocation points we require two additional pieces of notation. First, define the mesh step size sequence $H = \{h_i\}_{i=1}^{NINT}$ by

$$h_i = x_{i+1} - x_i. \tag{18}$$

Second, let $\{\rho_r\}_{r=1}^{k}$ be the k Gauss-Legendre points (the zeros of the degree k Legendre polynomial) on the interval $[0, 1]$ where

$$0 < \rho_1 < \rho_2 < \ldots < \rho_k < 1, \tag{19}$$

where $k = \kappa - m$. We now define the collocation point sequence $\Xi = \{\xi_j\}_{j=1}^{NCPTS}$ by

$$\begin{aligned} \xi_1 &= x_1, & (20)\\ \xi_{l+r} &= x_i + h_i\rho_r, \quad \text{where } l = (i-1)k + 1, & \\ & \text{for } i = 1, \ldots, NINT, \quad r = 1, \ldots, k, & (21)\\ \xi_{NCPTS} &= x_{NINT+1}. & (22) \end{aligned}$$

Note that, Ξ is an increasing partition of $[x_a, x_b]$ and the inclusion of x_1 and x_{NINT+1} with the collocation points is done for notational convenience as in Madsen et al[14]. The number of collocation points is $NCPTS = k \cdot NINT + m$, which is the dimension of the space $\mathcal{M}_X^{\kappa,m}$.

The function $u(t,x)$ is approximated by $U(t,x)$, which is a linear combination of functions from the space $\mathcal{M}_X^{k+m,m}$ with coefficients which are functions of t. The basis functions we use for $\mathcal{M}_X^{k+m,m}$ are monomials locally, each with support over only one subinterval. If we consider the restriction of $U(t,x)$ to the subinterval $[x_i, x_{i+1}]$, for $i = 1, \ldots, NINT$, we have $U_i(t,x) = U(t,x)|_{[x_i,x_{i+1}]}$ where

$$U_i(t,x) = \sum_{j=1}^{m} \phi_j(x - x_i) y_{i,j}(t) + h_i^m \sum_{j=1}^{k} \psi_j \left(\frac{x - x_i}{h_i} \right) z_{i,j}(t). \tag{23}$$

Further, $U_i(t,x) : [t_0, \infty) \times [x_i, x_{i+1}] \to \mathbf{R}^{NPDE}$, and $y_{i,j}(t)$, $z_{i,j}(t) : [t_0, \infty) \to \mathbf{R}^{NPDE}$, are unknown functions of time. The set of m functions,

$$\{\phi_j(x)\}_{j=1}^{m} = \left\{ \frac{x^{j-1}}{(j-1)!} \right\}_{j=1}^{m}, \tag{24}$$

gives the local representation of the first m (canonical) monomial basis functions. The k functions, $\{\psi_j(x)\}_{j=1}^{k}$ are chosen to satisfy the following conditions:

1. $\psi_j(x) : [0,1] \to \mathbf{R}$ is of order $k+m$, for $j = 1, \ldots, k$.
2. $\frac{d^{l-1}\psi_j}{dx^{l-1}}(0) = 0$ for $j = 1, \ldots, k$, and $l = 1, \ldots, m$.

These requirements do not completely define $\{\psi_j(x)\}_{j=1}^{k}$, and consequently this set is not unique. In order to specify the set uniquely, one choice is to have

$$\frac{d^{l-1}\psi_j}{dx^{l-1}}(1) = \delta_{j-k+m,l}, \quad j, l = 1, \ldots, k, \tag{25}$$

where $\delta_{i,j}$ is the Kronecker delta. This would imply that $\{\psi_j(x)\}_{j=1}^{k} = \{\frac{x^{m+j-1}}{(m+j-1)!}\}_{j=1}^{k}$, so that the basis functions ψ_j are defined in the same way as the functions ϕ_j. A second choice is given by

$$\frac{d^m \psi_j}{dx^m}(\rho_r) = \delta_{j,r}, \quad j, r = 1, \ldots, k. \tag{26}$$

The choice of (26), known as a Runge-Kutta representation[4] is used in the monomial spline package employed by our PDE code, and has been shown to have benefits over (25), see Ascher et al[2].

2.2. The Collocation Equations

We require the approximate function $U(t,x)$ to satisfy the PDE at the $NCPTS$ collocation points $\Xi = \{\xi_j\}_{j=1}^{NCPTS}$. Special treatment is required for the boundary conditions, and therefore the collocation points ξ_1 and ξ_{NCPTS} will be considered in detail later. For the collocation points on the i-th subinterval we require $U(t,x)$ to satisfy the PDE (1). For $i = 1, \ldots, NINT$, this gives

$$\frac{\partial U_i}{\partial t}(t, \xi_{l+r}) = f\left(t, U_i(t, \xi_{l+r}), \frac{\partial U_i}{\partial x}(t, \xi_{l+r}), \frac{\partial^2 U_i}{\partial x^2}(t, \xi_{l+r})\right), \tag{27}$$

where $l = (i-1)k + 1$ and $r = 1, \ldots, k$. The left hand side of (27) expands to

$$\sum_{j=1}^{m} \phi_j(\xi_{l+r} - x_i)\frac{dy_{i,j}}{dt}(t) + h_i^m \sum_{j=1}^{k} \psi_j\left(\frac{\xi_{l+r} - x_i}{h_i}\right)\frac{dz_{i,j}}{dt}(t). \tag{28}$$

Thus, the collocation equations of (27) simplify to

$$\sum_{j=1}^{m} \phi_j(h_i\rho_r)\frac{dy_{i,j}}{dt}(t) + h_i^m \sum_{j=1}^{k} \psi_j(\rho_r)\frac{dz_{i,j}}{dt}(t) = f_{i,r}(t), \tag{29}$$

for $r = 1, \ldots, k$ and $i = 1, \ldots, NINT$, where

$$f_{i,r}(t) = f\left(t, U_i(t, \xi_{l+r}), \frac{\partial U_i}{\partial x}(t, \xi_{l+r}), \frac{\partial^2 U_i}{\partial x^2}(t, \xi_{l+r})\right), \tag{30}$$

and $l = (i-1)k + 1$. These lead to the initial value ODE system

$$V_i\frac{d\vec{y}_i}{dt}(t) + W_i\frac{d\vec{z}_i}{dt}(t) = \vec{f}_i(t), \quad i = 1, \ldots, NINT, \tag{31}$$

where, using the symbol $\otimes$ for Kronecker Product,

$$V_i = [\phi_j(h_i\rho_r); j = 1, \ldots, m; r = 1, \ldots, n] \otimes I_{NPDE} \in \mathbf{R}^{k\times m} \otimes \mathbf{R}^{NPDE\times NPDE}, \tag{32}$$

$$W_i = [h_i^m\psi_j(\rho_r); j, r = 1, \ldots, k] \otimes I_{NPDE} \in \mathbf{R}^{k\times k} \otimes \mathbf{R}^{NPDE\times NPDE}, \tag{33}$$

$$\vec{y}_i(t) = \left[y_{i,1}^T(t), \ldots, y_{i,m}^T(t)\right]^T \in \mathbf{R}^m \otimes \mathbf{R}^{NPDE}, \tag{34}$$

$$\vec{z}_i(t) = \left[z_{i,1}^T(t), \ldots, z_{i,k}^T(t)\right]^T \in \mathbf{R}^k \otimes \mathbf{R}^{NPDE}, \tag{35}$$

$$\vec{f}_i(t) = \left[f_{i,1}^T(t), \ldots, f_{i,k}^T(t)\right]^T \in \mathbf{R}^k \otimes \mathbf{R}^{NPDE}, \tag{36}$$

and I_{NPDE} is the identity matrix in $\mathbf{R}^{NPDE\times NPDE}$.

The initial value ODEs which result from the collocation process can then be summarized by the system:

$$A_c\frac{dY}{dt}(t) = F_c(t, Y(t)), \tag{37}$$

where , with $N = \kappa \cdot NINT + m = (k+m)NINT + m$,

$$Y(t) = \left[\vec{y_1}^T, \vec{z_1}^T, \vec{y_2}^T, \vec{z_2}^T, \ldots, \vec{y}_{NINT}^T, \vec{z}_{NINT}^T, \vec{y}_{NINT+1}^T\right]^T \in \mathbf{R}^N \otimes \mathbf{R}^{NPDE}, \qquad (38)$$

$$F_c(t, Y(t)) = \left[\vec{0}_1^T, \vec{f_1}^T, \vec{0_2}^T, \vec{f_2}^T, \vec{0_2}^T, \ldots, \vec{f}_{NINT}^T, \vec{0_2}^T, \vec{0_1}^T\right]^T \in \mathbf{R}^N \otimes \mathbf{R}^{NPDE}. \qquad (39)$$

Further, $\vec{0_1} \in \mathbf{R}^{NPDE}, \vec{0_2} \in R^{NPDE \cdot m}$, are vectors of zeros, and $A_c \in \mathbf{R}^{N \times N} \otimes \mathbf{R}^{NPDE \times NPDE}$. In the components of the two vectors $Y(t)$ and $F_c(t, Y(t))$ we have suppressed the dependence on t. The blocks of zeros in the vector F_c are matched by sets of rows of zeros in A_c. The first and last blocks of zeros correspond to the (as yet not imposed) boundary conditions; the other blocks of zeros correspond to the continuity conditions as described in the next section. Consequently, the structure of A_c can be inferred from (31) and is described later, in conjunction with the Jacobian matrix defined in Sections 3 and 4.

2.3. The Continuity Equations

The continuity conditions satisfied by $U(t, x)$ must be explicitly imposed, since they are not built into the basis functions, as is the case with the B-splines. In our case, ($m = 2$), this means applying continuity and first derivative conditions at each of the mesh points x_i, $i = 2, \ldots, NINT + 1$, but we will, as before, express these in terms of a variable m. These conditions result in algebraic constraints on the initial value ODE system (37). One way to deal with these constraints is to differentiate them with respect to time and combine them with the initial value ODEs arising from the collocation equations (37) as is done with the boundary equations in PDECOL. However, we have chosen to leave the continuity conditions as algebraic constraints resulting in a DAE system.

The m conditions to be satisfied by $U(t, x)$ at each internal mesh point and at the right hand end point are:

$$\begin{aligned} U_i(t, x_{i+1}) &= U_{i+1}(t, x_{i+1}), & (40)\\ \frac{\partial U_i}{\partial x}(t, x_{i+1}) &= \frac{\partial U_{i+1}}{\partial x}(t, x_{i+1}), & (41)\\ &\vdots & \\ \frac{\partial^{m-1} U_i}{\partial x^{m-1}}(t, x_{i+1}) &= \frac{\partial^{m-1} U_{i+1}}{\partial x^{m-1}}(t, x_{i+1}), & (42) \end{aligned}$$

for $i = 1, \ldots, NINT$. Recalling the definition of $U_i(t, x)$ in (23) and the properties of the local monomial spline basis elements, we can write this system as:

$$-C_i \vec{y_i}(t) - D_i \vec{z_i}(t) + I \vec{y}_{i+1} = \vec{0}_{NPDE}, \quad i = 1, \ldots, NINT, \qquad (43)$$

where

$$\begin{aligned}
C_i &= \left[\frac{d^{r-1}\phi_j}{dx^{r-1}}(h_i)\right] \otimes I_{NPDE} \in \mathbf{R}^{m\times m} \otimes \mathbf{R}^{NPDE\times NPDE}, && (44)\\
D_i &= \left[h_i^{m+1-r}\frac{d^{r-1}\psi_j}{dx^{r-1}}(1)\right] \otimes I_{NPDE} \in \mathbf{R}^{m\times k} \otimes \mathbf{R}^{NPDE\times NPDE}, && (45)\\
I &= I_m \otimes I_{NPDE} \in \mathbf{R}^{m\times m} \otimes \mathbf{R}^{NPDE\times NPDE}. && (46)
\end{aligned}$$

The algebraic constraints resulting from the continuity equations can be summarized by the system

$$B_c Y(t) = \vec{0}_N \otimes \vec{0}_{NPDE}, \tag{47}$$

where $\vec{0}_N$ is the zero vector of $\mathbf{R}^N$, and $B_c \in \mathbf{R}^{N\times N} \otimes \mathbf{R}^{NPDE\times NPDE}$. Once again, the structure of B_c can be inferred from (43) and is described later in Sections 3 and 4.

2.4. The Boundary Conditions

For the collocation method, the final requirement is that $U(t,x)$ satisfies the boundary conditions. The boundary conditions give:

$$\begin{aligned}
b_{x_a}(t, U(t,\xi_1), U_x(t,\xi_1)) &= \vec{0}_{NPDE}, && (48)\\
b_{x_b}(t, U(t,\xi_{NCPTS}), U_x(t,\xi_{NCPTS})) &= \vec{0}_{NPDE}. && (49)
\end{aligned}$$

These equations represent a set of non-linear algebraic constraints on the vector $Y(t)$ which we denote as

$$F_B(t, Y(t)) = \vec{0}_N \otimes \vec{0}_{NPDE}. \tag{50}$$

Once again, there is the option of differentiating the boundary conditions, (48)-(49), with respect to t, or of applying them directly. In our code we choose to differentiate them, and this gives:

$$\begin{aligned}
\frac{\partial b_{x_a}}{\partial U}(t,U,U_x)\frac{\partial U}{\partial t} + \frac{\partial b_{x_a}}{\partial U_x}(t,U,U_x)\frac{\partial U_x}{\partial t} + \frac{\partial b_{x_a}}{\partial t}(t,U,U_x) &= \vec{0}_{NPDE}, && (51)\\
\frac{\partial b_{x_b}}{\partial U}(t,U,U_x)\frac{\partial U}{\partial t} + \frac{\partial b_{x_b}}{\partial U_x}(t,U,U_x)\frac{\partial U_x}{\partial t} + \frac{\partial b_{x_b}}{\partial t}(t,U,U_x) &= \vec{0}_{NPDE}. && (52)
\end{aligned}$$

In (51) U and U_x are evaluated at (t,ξ_1), and in (52) they are evaluated at (t,ξ_{NCPTS}). Simplification of (51)-(52), using the definition of $U_i(t,x)$ in (23), yields the following linear initial value ODEs

$$\left[\begin{array}{cc} \frac{\partial b_{x_a}}{\partial U}(t,U,U_x) & \frac{\partial b_{x_a}}{\partial U_x}(t,U,U_x) \end{array}\right] \frac{d\vec{y}_1}{dt}(t) = -\frac{\partial b_{x_a}}{\partial t}(t,U,U_x) \tag{53}$$

$$\left[\begin{array}{cc} \frac{\partial b_{x_b}}{\partial U}(t,U,U_x) & \frac{\partial b_{x_b}}{\partial U_x}(t,U,U_x) \end{array}\right] \frac{d\vec{y}_{NINT+1}}{dt}(t) = -\frac{\partial b_{x_b}}{\partial t}(t,U,U_x) \tag{54}$$

where $\vec{y}_1(t)$ and $\vec{y}_{NINT+1}(t)$ are defined by (34) with $m = 2$. For future convenience, we define

$$TOP = \left[\begin{array}{cc} \frac{\partial b_{x_a}}{\partial U}(t, U, U_x) & \frac{\partial b_{x_a}}{\partial U_x}(t, U, U_x) \end{array} \right], \tag{55}$$

$$BOT = \left[\begin{array}{cc} \frac{\partial b_{x_b}}{\partial U}(t, U, U_x) & \frac{\partial b_{x_b}}{\partial U_x}(t, U, U_x) \end{array} \right]. \tag{56}$$

Recalling the definition of $Y(t)$ in (38), we can define a matrix $A_{dBdt} \in \mathbf{R}^{N \times N} \otimes \mathbf{R}^{NPDE \times NPDE}$ so that (53) and (54) are described by the system

$$A_{dBdt} \frac{dY}{dt}(t) = F_{dBdt}(t, Y(t)), \tag{57}$$

where

$$F_{dBdt}(t, Y(t)) = \left[-\frac{\partial b^T_{x_a}}{\partial t}(t, U, U_x), 0, \ldots, 0, -\frac{\partial b^T_{x_b}}{\partial t}(t, U, U_x) \right]^T \in \mathbf{R}^N \otimes \mathbf{R}^{NPDE}. \tag{58}$$

The matrix A_{dBdt} will be very sparse with only the top and bottom blocks having non-zero entries as shown later in the following sections.

3. The DAE System

There are four options for constructing the initial value ODE or DAE system depending on how we treat the boundary and continuity conditions. We may decide to include the two sets of conditions as algebraic conditions, or we may choose to differentiate with respect to time one or the other, or both, of the sets, and form an ODE system. It seems more natural to include the continuity conditions as algebraic constraints, and we have done so for $t > t_0$. Similarly it seems reasonable to impose the boundary conditions as algebraic constraints, thus producing a system of DAE initial value problems. The code DASSL[19] is designed to handle such systems. But a problem arises when the code starts at t_0.

Consider the generic initial value DAE system

$$AY'(t) + BY(t) - F(t, Y(t)) = \vec{0}_N \otimes \vec{0}_{NPDE}, \tag{59}$$

where $Y(t_0)$ and $Y'(t_0)$ are known, and $Y(t_{out})$ is desired. In order to start DASSL it is necessary to give accurate values for $Y(t_0)$ and $Y'(t_0)$. The values of $Y(t_0)$ can be obtained by interpolating $u_0(x)$ on the set Ξ. In the case of an explicit ODE, $Y'(t_0)$ can be obtained directly from the ODE by solving a linear system (implicitly inverting A). But if the boundary and continuity conditions are imposed as algebraic constraints then $A = A_c$ and $B = B_c$, so that A is singular, having large blocks of zeros corresponding to non-zeros in B_c, and having TOP and BOT all zeros. If, however, we differentiate with respect to t both the continuity and boundary conditions and apply (59) at $t = t_0$, then $A = A_c + A_{dBdt} + B_c$ is non-singular, and $Y'(t_0)$ can be obtained.

For $t > t_0$ we keep the continuity conditions as algebraic constraints, and differentiate the boundary conditions, adding them to the system of ODEs in (59), therefore, giving

$$A = A_c + A_{dBdt}, \quad B = B_c, \quad F(t, Y(t)) = F_c(t, Y(t)) + F_{dBdt}(t, Y(t)). \tag{60}$$

DASSL also requires the Jacobian or iteration matrix $\mathcal{J}$ given by

$$\mathcal{J} = \alpha \frac{\partial G}{\partial Y'} + \frac{\partial G}{\partial Y} \in \mathbf{R}^{N \times N} \otimes \mathbf{R}^{NPDE \times NPDE}, \tag{61}$$

where α is chosen by DASSL to accelerate convergence, and where G is the residual given by

$$G(t, Y(t), Y'(t)) \equiv AY'(t) + BY(t) - F(t, Y(t)) = \vec{0}_N \otimes \vec{0}_{NPDE}, \tag{62}$$

It is clear from the definition of G in (62) that

$$\mathcal{J} = \alpha A + B - \frac{\partial F}{\partial Y}. \tag{63}$$

The structure of $\mathcal{J}$ is determined by the support of the basis functions. To aid in describing the iteration matrix $\mathcal{J}$ we will make the following definitions:

$$\begin{aligned}
\widehat{TOP} &= \alpha\, TOP - \left[\begin{array}{cc} \frac{\partial F_1}{\partial Y_1} & \frac{\partial F_1}{\partial Y_2} \end{array} \right], && (64)\\
\widehat{BOT} &= \alpha\, BOT - \left[\begin{array}{cc} \frac{\partial F_N}{\partial Y_{N-1}} & \frac{\partial F_N}{\partial Y_N} \end{array} \right], && (65)\\
\hat{V}_i &= \alpha V_i - \left[\frac{\partial F_{I+r}}{\partial Y_{J+j}} \right], && (66)\\
& \quad \text{for } r = 1, \ldots, k,\ j = 1, \ldots, m, && (67)\\
\hat{W}_i &= \alpha W_i - \left[\frac{\partial F_{I+r}}{\partial Y_{J+j}} \right], && (68)\\
& \quad \text{for } r = 1, \ldots, k,\ j = m, \ldots, k+m, && (69)
\end{aligned}$$

where $I = (i-1)(k+m) + 1$ and $J = (i-1)(k+m)$.

A special note regarding $\widehat{TOP}$ and $\widehat{BOT}$ is appropriate. Since it is unrealistic to expect the user to provide $\frac{\partial \frac{\partial b_{x_a}}{\partial t}}{\partial U}$, $\frac{\partial \frac{\partial b_{x_a}}{\partial t}}{\partial U_x}$, $\frac{\partial \frac{\partial b_{x_b}}{\partial t}}{\partial U}$, and $\frac{\partial \frac{\partial b_{x_b}}{\partial t}}{\partial U_x}$, we assume that these functions are identically zero. This results in an iteration matrix $\mathcal{J}$ which is not exact if this assumption is false. However, the dependence of $\mathcal{J}$ on the boundary information is assumed to be weak, so that the only affect of our assumption is possibly to slow convergence of the Newton iterations. This assumption has also been used in PDECOL [14] and EPDCOL[11].

4. The Linear Algebra component

The iteration matrix $\mathcal{J}$ can be shown to have the structure, (called almost block diagonal), illustrated in Figure 1, for the case when $NINT = 4$, $m = 2$ and $k = 3$, i.e. using splines of degree 4, or order 5. Generally, each element in this matrix is a full matrix in $\mathbf{R}^{NPDE \times NPDE}$. A detailed description of this is given in Nokonechny[18]. The package ABDPACK[15] is designed to perform a decomposition of this type, without additional storage being required. The code DASSL, has two options for the linear algebra component, namely full matrices and band matrices. If the band option is used on the matrix $\mathcal{J}$ considerable fill-in occurs (see Majaess et al[15]), as shown in Figure 1. It is therefore important to treat such systems using linear algebra software adapted to the particular structure shown in the figure. We have modified DASSL to include a third linear algebra option allowing us to use ABDPACK. It should be noted that the changes required to DASSL are greatly simplified as a result of careful structuring of the original code.

Figure 1: The iteration matrix $\mathcal{J}$.

5. The Software

In this section we describe a package called MSCPDE (**M**onomial **S**pline **C**ollocation for **P**arabolic Partial **D**ifferential **E**quations) implementing the algorithm described earlier.

5.1. The Components of Package

There are four principal components of the package:

(i) The control program, MSCPDE. This subroutine provides the interface between the user's calling program and the underlying packages required to solve the PDE. Additionally, the subroutine MSCPDE manages the allocation of storage to the various arrays required, from a block of workspace provided by the user.

(ii) The monomial spline package, which implements the low level details of the basis functions (see Muir et al[17]).

(iii) DASSL, the DAE solver.

(iv) ABDPACK, the linear algebra package.

MSCPDE requires the user to provide information about the system of PDEs (1), F and DERIVF, the initial conditions (2), UINIT, and the boundary conditions (3)-(4), DIFBXA and DIFBXB, in differentiated form, as indicated in (51)-(52). We have retained the same user interface as PDECOL[14] and EPDCOL[11], with the exception of the routine for the boundary conditions. In PDECOL and EPDCOL both differentiated boundary conditions are defined in the one routine BNDRY. We have replaced BNDRY by two routines DIFBXA and DIFBXB so that the left and right differentiated boundary conditions are specified by two separate routines DIFBXA and DIFBXB for the left and right hand boundaries, respectively. The detailed descriptions of these components are omitted. For these we refer to Nokonechny[18]. The code itself, with sample driver programs, is available through anonymous ftp at ftp.cs.dal.ca.

As an example we consider the system of partial differential equations given by:

$$\frac{\partial u}{\partial t} = v^2\frac{\partial^2 u}{\partial x^2} + 2u\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} - uv - u^2 + 10, \tag{70}$$

$$\frac{\partial v}{\partial t} = u^2\frac{\partial^2 v}{\partial x^2} + 2v\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + uv - v^2, \tag{71}$$

with initial conditions given by

$$\begin{aligned} u(0,x) &= 0.5(x+1), \qquad (72)\\ v(0,x) &= \pi, \qquad (73) \end{aligned}$$

and subject to the boundary conditions

$$\begin{aligned} u(t,0) &= 0.5 \qquad (74)\\ v(t,0) &= \pi \qquad (75)\\ \frac{\partial u}{\partial x}(t,1) + sin(u(t,1)v(t,1)) &= 0.5 \qquad (76)\\ \frac{\partial v}{\partial x}(t,1) - cos(u(t,1)v(t,1)) &= 1 \qquad (77) \end{aligned}$$

This problem is example 1 in Madsen and Sincovec[14]. The results in Table 1, below, essentially reproduce the results of that paper. The number of collocation points per subinterval, K, is 2, and the number of subintervals used, $NINT$, is 20. The requested abolute and relative tolerances, $ATOL$ and $RTOL$ are 0.01 and 0.0001 respectively.

```
        T = 0.001    Steps = 26
  PDE Component 1
      0.50000E+00  0.51924E+00  0.53776E+00  0.55569E+00  0.57318E+00
      0.59035E+00  0.60729E+00  0.62409E+00  0.64080E+00  0.65746E+00
      0.67408E+00  0.69069E+00  0.70729E+00  0.72389E+00  0.74048E+00
      0.75707E+00  0.77366E+00  0.79025E+00  0.80683E+00  0.82342E+00
      0.84001E+00  0.85660E+00  0.87319E+00  0.88979E+00  0.90640E+00
      0.92302E+00  0.93968E+00  0.95639E+00  0.97318E+00  0.99009E+00
      0.10072E+01
  PDE Component 2
      0.31416E+01  0.31333E+01  0.31331E+01  0.31332E+01  0.31334E+01
      0.31335E+01  0.31336E+01  0.31337E+01  0.31337E+01  0.31338E+01
      0.31338E+01  0.31339E+01  0.31340E+01  0.31340E+01  0.31341E+01
      0.31341E+01  0.31342E+01  0.31342E+01  0.31343E+01  0.31343E+01
      0.31344E+01  0.31344E+01  0.31345E+01  0.31345E+01  0.31346E+01
      0.31346E+01  0.31347E+01  0.31348E+01  0.31348E+01  0.31349E+01
      0.31350E+01
```

Table 1: K = 2; NINT = 30; ATOL = 0.01; RTOL = 0.0001

```
        T = 0.01     Steps = 44
PDE Component 1
     0.50000E+00  0.52477E+00  0.54917E+00  0.57295E+00  0.59603E+00
     0.61840E+00  0.64011E+00  0.66122E+00  0.68178E+00  0.70185E+00
     0.72152E+00  0.74082E+00  0.75983E+00  0.77860E+00  0.79720E+00
     0.81569E+00  0.83413E+00  0.85258E+00  0.87112E+00  0.88982E+00
     0.90876E+00  0.92801E+00  0.94767E+00  0.96784E+00  0.98863E+00
     0.10102E+01  0.10325E+01  0.10559E+01  0.10805E+01  0.11065E+01
     0.11340E+01
PDE Component 2
     0.31416E+01  0.30943E+01  0.30722E+01  0.30633E+01  0.30606E+01
     0.30603E+01  0.30610E+01  0.30620E+01  0.30630E+01  0.30640E+01
     0.30649E+01  0.30658E+01  0.30667E+01  0.30675E+01  0.30683E+01
     0.30691E+01  0.30699E+01  0.30707E+01  0.30716E+01  0.30724E+01
     0.30733E+01  0.30742E+01  0.30752E+01  0.30762E+01  0.30773E+01
     0.30784E+01  0.30796E+01  0.30810E+01  0.30824E+01  0.30839E+01
     0.30858E+01

         T = 0.1     Steps = 77
PDE Component 1
     0.50000E+00  0.54716E+00  0.59704E+00  0.64874E+00  0.70148E+00
     0.75459E+00  0.80753E+00  0.85987E+00  0.91129E+00  0.96155E+00
     0.10105E+01  0.10579E+01  0.11038E+01  0.11481E+01  0.11908E+01
     0.12319E+01  0.12714E+01  0.13093E+01  0.13457E+01  0.13805E+01
     0.14140E+01  0.14460E+01  0.14766E+01  0.15060E+01  0.15341E+01
     0.15610E+01  0.15868E+01  0.16114E+01  0.16351E+01  0.16577E+01
     0.16794E+01
PDE Component 2
     0.31416E+01  0.30064E+01  0.29093E+01  0.28408E+01  0.27940E+01
     0.27641E+01  0.27471E+01  0.27406E+01  0.27424E+01  0.27510E+01
     0.27653E+01  0.27844E+01  0.28076E+01  0.28344E+01  0.28645E+01
     0.28975E+01  0.29332E+01  0.29713E+01  0.30118E+01  0.30546E+01
     0.30994E+01  0.31464E+01  0.31954E+01  0.32464E+01  0.32995E+01
     0.33545E+01  0.34115E+01  0.34705E+01  0.35315E+01  0.35947E+01
     0.36598E+01
```

Table 1(continued): K = 2; NINT = 30; ATOL = 0.01; RTOL = 0.0001

```
        T = 1.0      Steps = 95
PDE Component 1
     0.50000E+00  0.54703E+00  0.59686E+00  0.64856E+00  0.70130E+00
     0.75436E+00  0.80713E+00  0.85915E+00  0.91007E+00  0.95963E+00
     0.10077E+01  0.10540E+01  0.10987E+01  0.11417E+01  0.11829E+01
     0.12224E+01  0.12602E+01  0.12964E+01  0.13311E+01  0.13643E+01
     0.13960E+01  0.14264E+01  0.14554E+01  0.14833E+01  0.15099E+01
     0.15354E+01  0.15599E+01  0.15834E+01  0.16059E+01  0.16275E+01
     0.16482E+01
PDE Component 2
     0.31416E+01  0.30031E+01  0.29038E+01  0.28345E+01  0.27884E+01
     0.27604E+01  0.27467E+01  0.27443E+01  0.27509E+01  0.27649E+01
     0.27848E+01  0.28096E+01  0.28387E+01  0.28712E+01  0.29068E+01
     0.29451E+01  0.29858E+01  0.30286E+01  0.30735E+01  0.31202E+01
     0.31687E+01  0.32189E+01  0.32708E+01  0.33243E+01  0.33795E+01
     0.34363E+01  0.34947E+01  0.35548E+01  0.36167E+01  0.36803E+01
     0.37457E+01

       T = 10.0      Steps = 100
PDE Component 1
     0.50000E+00  0.54703E+00  0.59686E+00  0.64856E+00  0.70130E+00
     0.75435E+00  0.80712E+00  0.85914E+00  0.91006E+00  0.95962E+00
     0.10076E+01  0.10540E+01  0.10987E+01  0.11416E+01  0.11828E+01
     0.12224E+01  0.12602E+01  0.12964E+01  0.13311E+01  0.13643E+01
     0.13960E+01  0.14264E+01  0.14554E+01  0.14832E+01  0.15099E+01
     0.15354E+01  0.15599E+01  0.15833E+01  0.16058E+01  0.16274E+01
     0.16482E+01
PDE Component 2
     0.31416E+01  0.30032E+01  0.29038E+01  0.28345E+01  0.27884E+01
     0.27605E+01  0.27468E+01  0.27443E+01  0.27510E+01  0.27649E+01
     0.27848E+01  0.28097E+01  0.28387E+01  0.28713E+01  0.29069E+01
     0.29452E+01  0.29858E+01  0.30287E+01  0.30735E+01  0.31203E+01
     0.31688E+01  0.32190E+01  0.32709E+01  0.33244E+01  0.33796E+01
     0.34363E+01  0.34948E+01  0.35549E+01  0.36168E+01  0.36804E+01
     0.37458E+01
```

Table 1(continued): K = 2; NINT = 30; ATOL = 0.01; RTOL = 0.0001

It may be observed that the results in Table 1 are similar to those in Sinvocec and Madsen[14] with the number of steps taken by MSCPDE being comparable to or fewer than the number taken by PDECOL. However, the choice of tolerances is somewhat different since PDECOL only requires the user to specify a relative tolerance, whereas DASSL (and hence MSCPDE) also allows an absolute tolerance to be set. Some care must be taken in the choice of RTOL for MSCPDE, since too small a value results in a larger number of time steps being taken, without any appreciable difference in accuracy. Since the two codes are being asked to satisfy different requests, it is up to the user of MSCPDE to select ATOL and RTOL wisely.

References

1. U. M. Ascher, J. Christiansen and R. D. Russell *Algorithm 569: COLSYS: Collocation Software for boundary value ODEs*, ACM TOMS (1981) p. 223.
2. U. M. Ascher, R.M.M. Mattheij and R. D. Russell in *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, SIAM Classics in Applied Mathematics, Philadelphia, 1995.
3. G. Bader and U. Ascher, *A New Basis Implementation for a Mixed Order Boundary Value ODE Solver*, SIAM J. Sci. Stat. Anal. (1987) p. 483.
4. U. Ascher, S. Pruess and R. D. Russell, *On Spline Basis Selection for Solving Differential Equations*, SIAM J. Numer. Anal. (1983) p. 121.
5. M. Berzins and P. M. Dew, *Algorithm 690: Chebyshev Polynomial Software for Elliptic-Parabolic Systems of PDEs*, ACM TOMS (1991) p. 178.
6. M. Berzins, J. Lawson and J. Ware, *Spatial and Temporal Error Control in the Adaptive Solution of Systems of Conservation Laws*, in Advances in Computer Methods for Partial Differential Equations, VII, edited by R. Vichnevetskym D. Knight and G. Richter, IMACS (1992) p. 60.
7. J. Carroll, *A composite integration scheme for the numerical solution of systems of parabolic PDEs in one space dimension*, J. Comp. and Appl. Math. (1993) p. 327.
8. C. de Boor, *Package for Calculating with B-Splines*, SIAM J. Numer. Anal. (1977) p. 441.
9. J. E. Flaherty and P. K. Moore, *Integrated Space-time Adaptive hp-Refinement Methods for Parabolic Systems*, App. Num. Math. (1995) p. 317.
10. W. Huang, Y. Ren and R. D. Russell, *Moving Mesh Methods Based on Moving Mesh Partial Differential Equations*, J. Comp. Phys. (1994) p. 279.
11. P. Keast and P. H. Muir, *Algorithm 688: EPDCOL: A More Efficient PDECOL Code*, ACM TOMS (1991) p. 153.
12. J. Lawson, M. Berzins and P.M. Dew *Balancing Space and Time Errors in the Method of Lines*, SIAM J. Sci. Statist. Comput. (1991) p. 573.
13. N. K. Madsen and R. F. Sincovec, *Algorithm 494: PDEONE, Solutions of Systems of Partial Differential Equations*, ACM TOMS (1975) p. 261.

14. N. K. Madsen and R. F. Sincovec, *Algorithm 540 PDECOL, General Collocation Software for Partial Differential Equations*, ACM TOMS (1979) p. 326.
15. F. Majaess, P. Keast, Graeme Fairweather, and Karin R. Bennett, *Algorithm 704: ABDPACK and ABBPACK-FORTRAN Programs for the Solution of Almost Block Diagonal Linear Systems Arising in Spline Collocation at Gaussian Points with Monomial Basis Functions*, ACM TOMS (1992) p. 205.
16. P. K. Moore and J. E. Flaherty, *Adaptive Local Overlapping Grid Methods for Parabolic Systems in Two Space Dimension*, J. Comp. Phys. (1992) p. 54.
17. P.H. Muir and K. Remington, *Software for Manipulating Monomial Spline Bases*, unpublished software.
18. T. B. Nokonechny, *The Method of Lines Using Monomial Spline Collocation for Parabolic Partial Differential Equations*, MSC thesis, Dalhousie University, Halifax, Canada, 1995.
19. L. Petzold, DASSL, electronically available through netlib, at netlib.att.com.
20. R. A. Trompert and J. G. Verwer, *Analysis of the Implicit Euler Local Uniform Refinement*, SIAM J. Sci. Comput. (1993) p. 259.
21. J. G. Werver and R. A. Trompert, *Methods of Lines and Local Uniform Grid Refinement*, in Advances in Computer Methods for Partial Differential Equations, VII, edited by R. Vichnevetskym D. Knight and G. Richter, IMACS (1992) p. 60.