B-spline Collocation for Two Dimensional, Time-Dependent, Parabolic PDEs

By

Zhi Li

A Thesis Submitted to
Saint Mary's University, Halifax, Nova Scotia
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Applied Science

August, 2012, Halifax, Nova Scotia

Approved:	Dr. Paul Muir Supervisor Department of Mathematics and Computing Science
Approved:	Dr. Christina C. Christara External Examiner Department of Computer Science University of Toronto
Approved:	Dr. Walt Finden Supervisory Committee Member Department of Mathematics and Computing Science
Approved:	Dr. Norma Linney Supervisory Committee Member Department of Mathematics and Computing Science
Approved:	Dr. Madine VanderPlaat Graduate Student Representative Department of Sociology and Criminology
Date:	August 24, 2012

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-89991-5

Our file Notre référence
ISBN: 978-0-494-89991-5

B-spline Collocation for Two Dimensional, Time-Dependent, Parabolic PDEs

By

Zhi Li

A Thesis Submitted to
Saint Mary's University, Halifax, Nova Scotia
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Applied Science

August, 2012, Halifax, Nova Scotia

Approved: _____
Dr. Paul Muir
Supervisor
Department of Mathematics and Computing Science

Approved: _____
Dr. Christina C. Christara
External Examiner
Department of Computer Science
University of Toronto

Approved: _____
Dr. Walt Finden
Supervisory Committee Member
Department of Mathematics and Computing Science

Approved: _____
Dr. Norma Linney
Supervisory Committee Member
Department of Mathematics and Computing Science

Approved: _____
Dr. Madine VanderPlaat
Graduate Student Representative
Department of Sociology and Criminology

Date: _____

Abstract

B-spline Collocation for Two-Dimensional, Time-Dependent, Parabolic PDEs

By Zhi Li

In this thesis, we consider B-spline collocation algorithms for solving two-dimensional in space, time-dependent parabolic partial differential equations (PDEs), defined over a rectangular region. We propose two ways to solve the problem: (i) The Method of Surfaces: Discretizing the problem in one of the spatial domains, we obtain a system of one-dimensional parabolic PDEs, which is then solved using a one-dimensional PDE system solver. (ii) Two-dimensional B-spline collocation: The numerical solution is represented as a bi-variate piecewise polynomial with unknown time-dependent coefficients. These coefficients are determined by requiring the numerical solution to satisfy the PDE at a number of points within the spatial domain, i.e., we collocate simultaneously in both spatial dimensions. This leads to an approximation of the PDE by a large system of time-dependent differential algebraic equations (DAEs), which we then solve using a high quality DAE solver.

Date: August 24, 2012

Acknowledgments

First of all, I would like to thank my supervisor, Dr. Paul Muir for his patience, suggestions and assistance through out the whole process of doing the thesis. The rigorous research attitude I learned from him will benefit my future study and work. I also would like to thank my external examiner, Dr. Christina C. Christara (University of Toronto), and my supervisory committee, Dr. Walt Finden (Saint Mary's University) and Dr. Norma Linney (Saint Mary's University), for their valuable comments and suggestions on my thesis. Thank you to my entire family for your love, support.

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we discuss the development of two numerical algorithms for solving time-dependent parabolic PDEs in two space dimensions, x and y. We assume a problem class having the form

$$u_t(x,y,t) = f(x,y,t,u(x,y,t),u_x(x,y,t),u_y(x,y,t),u_{xx}(x,y,t),u_{xy}(x,y,t),u_{yy}(x,y,t)), \tag{1.1}$$

for $(x,y,t) \in \Omega \times (t_0, t_{out}]$, where $u : R \times R \times R \rightarrow R^n$, $f : R \times R \times R \times R^n \times R^n \times R^n \times R^n \times R^n \times R^n \rightarrow R^n$, and $\Omega = \{(x,y) | a < x < b, c < y < d\}$.
The boundary conditions at $x = a$ and $x = b$ are assumed to have the form

$$g_a(y,t,u(a,y,t),u_x(a,y,t),u_y(a,y,t)) = 0, \;\; g_b(y,t,u(b,y,t),u_x(b,y,t),u_y(b,y,t)) = 0,$$

while the boundary conditions at $y = c$ and $y = d$ have the form

$$g_c(x,t,u(x,c,t),u_x(x,c,t),u_y(x,c,t)) = 0, \;\; g_d(x,t,u(x,d,t),u_x(x,d,t),u_y(x,d,t)) = 0,$$

for $t \in (t_0, t_{out}]$, where $g_a, g_b, g_c, g_d : R \times R \times R^n \times R^n \times R^n \rightarrow R^n$.
The initial conditions at $t = t_0$ are given by

$$u(x,y,t_0) = u_0(x,y), \;\; (x,y) \in \Omega \cup \partial\Omega,$$

where $u_0 : R \times R \rightarrow R^n$.

The first approach we consider is an approach we call the Method of Surfaces

(MOS). The MOS is obtained from a natural generalization of the method of lines (MOL) [Sch91] (We will briefly review the MOL in the next chapter). We obtain a system of 1D parabolic PDEs from the 2D PDE by discretizing the problem in the y domain. The solution of the 1D system approximates the solution of the 2D problem. We can then make use of high quality software for 1D parabolic PDEs in order to obtain an approximate solution to the 2D problem. An example of a software package for 1D parabolic PDEs is BACOL [WKM04a, WKM04c]. We propose two ways to implement the MOS: one based on finite differences and the other based on B-spline collocation.

The second approach we consider uses B-spline collocation in a tensor product framework to simultaneously discretize both spatial domains. The approximate solution is represented as a bi-variate piecewise polynomial implemented in terms of a B-spline basis [dB77, dB78], with unknown time-dependent coefficients. By requiring the approximate solution to satisfy the PDE and boundary conditions at certain points in the spatial domain, a system of differential algebraic equations (DAEs) is obtained. Since, in the two-dimensional case, the DAE system is usually large, we use DASPK [BHP94] to solve it. DASPK is a newer version of the well-known DASSL code [Pet83], and is able to solve large scale DAEs efficiently. Our preliminary implementation of this approach is called BACOL2D. Since BACOL2D is natural extension of BACOL, many of the algorithms employed in BACOL are applicable to BACOL2D. We also describe an algorithm that uses a fast block LU scheme with modified alternate row and column elimination with partial pivoting for the treatment of certain almost block diagonal (ABD) [ACR81, DFK83b] linear systems that arise during the computation.

In Chapter 2, we provide a review of the relevant literature. Chapter 3 discusses the MOS. Chapter 4 discusses 2D B-spline collocation and some numerical experiments. The BACOL2D code will be described in Chapter 5. We will present numerical results

in Chapter 6 using BACOL2D to demonstrate the existence of points within the spatial domain where the collocation solution is superconvergent. Such superconvergent values may be useful for error estimation. Chapter 7 provides our conclusions and suggestions for future work.

In the remainder of this chapter we describe the experimental determination of the order of convergence of a numerical solution and a matrix tensor product.

1.1 Order of Convergence of a Numerical Solution

In Chapter 4, we will numerically investigate the order of convergence of the 2D B-spline collocation solutions we obtain, and determine the order of convergence of the superconvergent points in Chapter 6. Here we explain how we experimentally determine the order of convergence of a numerical solution.

Assume we have a problem with a known solution so that the error of an approximate solution can be computed. Let E_1 and E_2 denote the errors of two solution approximations obtained from computations based on two different 2D uniform square spatial meshes having mesh sizes of h_1 and h_2 ($h_1 > h_2$). Here h_1 is the length of the side of each square of the mesh associated with E_1; h_2 is the corresponding quantity associated with E_2.

Assume that

$$E_1 = O(h_1^R); \text{ then } E_1 \approx Ch_1^R,$$

$$E_2 = O(h_2^R); \text{ then } E_2 \approx Ch_2^R,$$

where C is a constant independent of h_1 and h_2, and R is the unknown order of convergence of the numerical method that was used to obtain the approximate solutions.

Then

$$\frac{E_1}{E_2} \approx \left(\frac{h_1}{h_2}\right)^R,$$

which gives

$$\log\left(\frac{E_1}{E_2}\right) \approx R \cdot \log\left(\frac{h_1}{h_2}\right).$$

Then the order of convergence of the numerical method is given (approximately) by

$$R \approx \frac{\log\left(\frac{E_1}{E_2}\right)}{\log\left(\frac{h_1}{h_2}\right)}.$$

1.2 Matrix Tensor Product

At several places in the thesis, we will consider matrix tensor products. Let V be an $m \times n$ matrix and M be a $p \times q$ matrix.Then the tensor product (or Kronecker product) $V \otimes M$ is the $mp \times nq$ block matrix given by

$$V \otimes M = \begin{bmatrix} v_{11}M & v_{12}M & \cdots & v_{1n}M \\ v_{21}M & v_{22}M & \cdots & v_{2n}M \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1}M & v_{m2}M & & v_{mn}M \end{bmatrix}.$$

A property of the tensor product that will be used in Chapter 4 is

$$(A \otimes B)(C \otimes D) = AC \otimes BD,$$

where A, B, C and D are matrices of appropriate sizes.

Chapter 2

Literature Review

In the chapter, we will briefly review some of the literature associated with the numerical solution of 1D and 2D PDEs. We will also review software packages for these problem classes. In the last section we will review, in more detail, the package BACOL–a high order B-spline adaptive collocation solver for 1D PDEs.

2.1 Numerical solution of 1D PDEs

The problem class

We will consider 1D PDEs of the form

$$u_t(x,t) = f(x,t,u(x,t),u_x(x,t),u_{xx}(x,t)),$$

for $x \in (a,b)$, $t \in (t_0,+\infty)$, with $u : R \times R \to R^n$, $f : R \times R \times R^n \times R^n \times R^n \to R^n$. The initial conditions at $t = t_0$ are given by

$$u(x,t_0) = u_0(x), x \in [a,b],$$

and the separated boundary conditions are given by

$$b_L(t,u(a,t),u_x(a,t)) = 0, \quad b_R(t,u(b,t),u_x(b,t)) = 0,$$

where $b_L, b_R : R \times R^n \times R^n \to R^n$, and $t \in (t_0, +\infty)$.

There are many methods for solving 1D PDEs - see, e.g., [HV03, MRB05, Goc02, LeV07, VWSS01], but in this thesis, we will consider the MOL. The MOL is a general technique for solving time-dependent PDEs. The basic idea is that since there exist several good software packages for solving ordinary time-dependent differential equations (ODEs) or DAEs (i.e. a coupled system of ODEs and algebraic equations), we might approximate only the spatial derivatives, e.g., u_x, u_{xx}, instead of both the time derivative and spatial derivatives appearing in the PDE. This spatial discretization process leads to a system of ODEs or DAEs, (once the boundary conditions are included) and we can solve this system using a high quality ODE or DAE solver.

We next discuss the general MOL approach in more detail. First, the MOL algorithm discretizes the spatial derivatives using, for example, finite difference methods, finite element methods, or collocation methods. This leads to a system of initial value ODEs that approximate the original PDE. There are two ways to deal with the boundary conditions. In the earliest MOL codes, the numerical software available for the time integration could only handle ODE systems; therefore the boundary conditions had to be differentiated. That is the MOL software required the user to differentiate the boundary conditions with respect to time. These ODEs were then combined with the ODE system arising from the discretization of the PDEs, and this combined ODE system was solved with an ODE solver. However, the boundary conditions were solved only approximately by the ODE solver.

About thirty years ago, software packages which could handle DAEs began to appear. This meant that the boundary conditions could now be treated directly instead of requiring them to be differentiated. The boundary conditions are treated as algebraic equations, and are coupled with the ODEs which approximate the original PDEs, leading to a DAE system. There are many good quality DAE solvers, such as DASSL and RADAU5 [HW93].

The standard MOL approach fixes the mesh points that partition the spatial domain. The advantage is that the resulting system can be solved in a straight forward manner by an ODEs solvers with adaptive temporal error control. This means that the solver can adapt time steps and control an estimate of the temporal error.

In the standard MOL, adaptive error control is limited to the time dimension, and it thus is difficult to resolve sharp spatial dimension features of a PDE solution, such as a traveling wave front. In such cases the spatial error may dominate the temporal error, and the MOL algorithm would need more mesh points in the spatial layer regions (where the solution has rapid variations) to achieve high accuracy approximate solutions. Without adaptive spatial error control, it is impossible to determine the appropriate number and locations of the spatial mesh points.

Considering this limitation of the standard MOL, we observe that it is important to be able to move the mesh points to let them concentrate in the layer regions. The adaptive MOL (AMOL) [VWSS01] emerged to adapt the mesh in the space dimensions. The AMOL can be classified according to one or more of the following strategies:

h–refinement: refining or coarsening of the spatial mesh based on, e.g., a posteriori error estimates [Moo01, AO00],

p–refinement: varying the order of the numerical method used to approximate the solution in each subinterval of the spatial mesh,

r-refinement: also called the Moving Mesh Method (MMM) [HR11]; this involves relocating a fixed number of mesh points to layer regions of the physical domain.

There are several AMOL software packages, that can adjust the locations of the mesh points based on a monitor function, putting more of the available points in layer regions where the solution changes dramatically. When the mesh adaption algorithm allows for changing the location of the spatial mesh points *and the number of mesh points*, the method can provide tolerance control of the spatial error.

2.2 Software for 1D PDE

In this section we review several MOL software packages for the numerical solution of 1D PDEs.

M3RK

M3RK [Ver80b, Ver80a] is a software package based on an explicit three-step Runge-Kutta method for the time integration of discretized parabolic equations. The spatial dimension of the PDE is discretized using a Galerkin method [HV03] based on piecewise quadratic polynomials [Bak77]. The author later extended the software to solve 2D and 3D problems, including adaptive mesh refinement.

PEDCOL

PEDCOL [MS79] employs a fixed spatial mesh and uses a B-spline collocation method to discretize the PDEs. It requires the user to differentiate the boundary conditions to get ODEs. Since it uses a fixed spatial mesh, it has no control over the spatial errors. PDECOL calls the time integrator STIFIB, which is a slightly modified version of GEARIB [Gea71, Hin76]. GEARIB has two types of time integration formulas; the first is a family of Adams' methods [Goc02]; the second is a family of backward differentiation formulas (BDFs) [Gea71]. The authors suggest using the second class of methods because of the better stability properties. The linear systems that arise during the computation are assumed to have a band matrix structure and are solved by a band solver.

EPDCOL

EPDCOL [KM91] is an improvement of PDECOL. Based on the observation that

the Newton iteration matrix has an ABD structure, the use of the band linear system solver is replaced by COLROW [DFK83a, DFK83b], an ABD linear system solver. The authors show that this modification saves over 50 percent in total execution time.

MOVCOL

MOVCOL [HR96] uses a MMM based on cubic Hermite collocation to discretize the PDEs. The boundary conditions are treated as algebraic equations. DASSL is used to deal with the resulting DAE system.

HPNEW

HPNEW [Moo01], is an evolution of HPDASSL [Moo95], with spatial error control based on interpolation error estimates. The interpolation error estimates are based on a generalization of the error formula for the Lagrange interpolating polynomial. A finite-element Galerkin method is employed to implement the spatial discretization. The boundary conditions are treated as algebraic equations, and the resulting DAE system is solved by DASSL. HPNEW employs an adaptive hp-refinement strategy in space.

BACOL

BACOL uses B-spline collocation to discretize the PDEs, leading to a system of ODEs. The boundary conditions are treated directly as algebraic equations, and the resulting DAE system is solved using DASSL. Because DASSL uses BDFs, i.e., multistep methods, after a spatial remeshing, past solution values must be interpolated to get the required new values at previous time steps. DASSL controls the temporal error and at the end of each time step, BACOL computes a high-order estimate of

the spatial error and requires this estimate to satisfy the user tolerance. Numerical experiments [WKM04b] show that BACOL is more efficient than similar available codes such as PDECOL/EPDCOL, MOVCOL, and HPNEW, especially for solutions that have narrow spikes or boundary layers. We will describe BACOL in much more detail later in this chapter.

BACOLR

BACOLR [WKM08] is a new version of BACOL; the two codes share many of the same algorithms, such as the spatial discretization. The most significant difference is that BACOLR uses a modified version of RADAU5 which is based on a fifth-order implicit Runge-Kutta method of Radau type. The main modification of RADAU5 is to allow it to use the COLROW routines CRDCMP and CRSLVE (the ABD system solvers). Because RADAU5 is a one-step DAE solver, the solution from previous time steps no longer needs to be saved, and the interpolation of past solution values is not required. However for each time step, there is more work to do, since an implicit Runge-Kutta method is used. For problems for which the resultant DAE system has a Jacobian with eigenvalues near the imaginary axis, BACOLR is superior to BACOL. This happens because BACOL uses DASSL which is based on a BDFs which have stability issues for such problems [WKM08].

2.3 Numerical solution of 2D PDEs

We are going to implement 2D B-spline collocation for 2D time-dependent parabolic PDE, and solve the resulting DAEs by DASPK. To our knowledge, nobody has done this before. Previous experience with 1D parabolic PDE solvers has shown this to be

a promising approach.

We now briefly identify related work. R.D. Russell and W. Sun [RS97] used Hermite cubic spline collocation combined with a tensor product spatial basis to solve elliptic PDEs. They also suggested a fast algorithm based upon a matrix block eigenvalue decomposition for the numerical solution of the linear systems that arise. Later W. Sun [Sun01] suggested tensor product B-spline collocation for elasticity problems (non-time-dependent). Y. Wang [Wan95] used a parallel B-spline collocation method for 2D linear parabolic, separable PDEs. He separated the boundary conditions from the system of collocation conditions, solved the boundary conditions first, and then substituted the solution values on the boundary back into the system of collocation conditions. S. Wendel [WMKL93] used a 2D B-spline finite element method for the numerical solution of 2D PDEs.

There are many collocation methods that have been used to solve 2D problems. These include methods such as optimal Quadratic Spline Collocation (QSC) and Cubic Spline Collocation (CSC) that were extended to 2D elliptic BVPs for rectangular domains [HVR88, Chr94]. The advantage of these two methods is that since they use only one collocation point per subinterval, the linear systems that arise are the smallest among all types of piecewise polynomial collocation methods for this problem class.

For a 2D PDE whose solution has rapid variation, one useful approach is to use a MMM. A MMM typically controls the mesh movement using a moving mesh PDE (MMPDE) [HR11]. K. S. Ng [Ng05] used CSC and algorithms from [HR98b, HR98a] that employ MMM. He used CSC to treat the MMPDE to take advantage of the high order convergence of the CSC algorithm and its lower computational costs. For more details about the MMPDE approach, please refer to the book by W. Huang and R. D. Russell [HR11].

E.N. Houstis, W. Mitchell and J.R. Rice consider special collocation algorithms

(GENCOL [HMR85a], HERMCOL, INTCOL [HMR85b, HMR85c]) for elliptic problems on rectangular domains. (GENCOL is applicable to quite general domains and is thus more general than the other collocation packages mentioned above.) These packages are included in the ELLPACK [DR87, BL90, DMPP86] library. W. Huang and R. D. Russell [HS94, Hua01] introduced the MMPDE approach, and with W. Cao [CHR99, HR98b, HR98a, HR11], they studied the MMM for 2D problems. S. Adjerid et al. [AAF01, AFMW92, AM02], have constructed a posteriori estimates for the spatial errors of finite element MOL solutions of 2D parabolic PDEs.

2.4 Software for 2D PDEs

In this section we review selected software packages for 2D PDEs.

ELLPACK

ELLPACK [DR87, BL90, DMPP86] is an extension of a general framework for solving various elliptic PDEs in two-dimensions on general domains or in three-dimensions on rectangular domains. The ELLPACK system includes finite element methods (FEM), parallel execution, and a graphical user interface for problem specification and solution.

VLUGR2

VLUGR2 [BTV96] is an adaptive-grid software package for 2D systems of PDEs. It uses a finite difference method to implement the spatial discretization. Since large scale linear systems arise, this software package employs a Krylov subspace method to solve the linear systems. The linear systems can either be solved by BiCGStab [vdV92] with ILU preconditioning [Saa03] or GCRO [dSF93] with a simple (block) diagonal scaling. There is also an option to use a matrix-free implementation. While

VLUGR2 can deal with domains which can be described by right-angled polygons, the author has also extended the software to VLUGR3 [BV96] that can deal with an arbitrary "brick-structured" domains.

CLAWPACK

CLAWPACK [FL03, LeV96], can be used to solve 1-D and 2-D hyperbolic problems. CLAWPACK computes numerical solutions using a wave propagation approach that includes adaptive mesh refinement.

KASKADE

KASKADE 3.0 [BER95], adopts an object-oriented (OO) approach, and was developed for the solution of PDEs in one, two, or three space dimensions. Adaptive FEM techniques are employed to compute numerical solutions. The software employs a posteriori error estimation, local mesh refinement and multilevel preconditioning.

DIFFPACK

DIFFPACK [BL97, Lan03] is a comprehensive set of tools for solving PDEs based on OO programming. It supports FEM; the elements include the multilinear and multiquadratic type elements, 3D box elements, triangles and tetrahedrons.

SPRINT2D

M. Berzins et al. [BFP+98, BPPW97], developed the SPRINT2D software which solves time-dependent PDEs in two-space variables. The class of problems solved includes systems of parabolic, elliptic, and hyperbolic equations. It uses a finite volume method (FVM) to discretize the space domain. The software uses unstructured tri-

angular meshes and adaptive local error control in both space and time; h-refinement is used to perform any spatial adaptivity.

P2MESH

P2MESH [BM02], developed by Enrico Bertolazzi et al., is a generic object-oriented interface between 2-D unstructured meshes and FEM/FVM-based PDE solvers. Thus it can perform (i) mesh generation, (ii) formulation of the discrete algebraic equations (i.e., discretization of the PDE) (iii) solution of the discrete algebraic equations, and (iv) visualization of the numerical solution. One of the most important features of P2MESH is that it employs OO programming techniques, thus isolating the data structure design from the implementation.

deal.II

deal.II [BHK07] is an open source finite element library. It uses OO concepts to break the implementation into smaller blocks such as defining meshes, linear algebra, input/output capabilities. It has support for the solution of PDEs in one, two, and three space dimensions, and h, p, and hp refinement is fully supported. A variety of FEMs are available. deal.II also has features such as: a complete linear algebra library including methods for sparse matrices and Krylov subspace solvers.

Hermes

Hermes [VvZ07] is a C++ library for adaptive hp-FEM solvers. It can solve ODEs, linear PDEs and time-dependent nonlinear PDEs. The central part of the solver is the FEM/hp-FEM module which contains algorithms for processing and adaptation of finite element meshes, numerical quadrature, solution of systems of linear and

nonlinear algebraic equations, a posteriori error estimation, etc.

2.5 Overview of BACOL

The BACOL package, mentioned earlier, is the basis for the two approaches we consider in this thesis. We therefore provide a detailed review of this package in this section.

Spatial discretization

The spatial discretization is the same as the one described earlier for the one-dimensional time-dependent parabolic PDEs solvers PDECOL and EPDCOL. The spatial discretization is based on B-spline collocation at Gauss points. We assume a spatial mesh that partitions $[a, b]$

$$a = x_0 < x_1 < \cdots < x_N = b.$$

We associate with this mesh, piecewise polynomials of degree p, i.e., we have a polynomial of degree p for each subinterval, $[x_{i-1}, x_i], i = 1, \cdots, N$, and continuity $\nu - 1$. The approximation subspace is

$$S_p^\nu := \{V(x) | V \in C^{\nu-1}, x \in [a, b]; V \in P_p, x \in [x_{i-1}, x_i], i = 1, \cdots, N\},$$

where P_p is the space of polynomials of degree p, and $p > \nu > 0$. The dimension of S_p^ν is $NC = (p - \nu + 1)N + \nu$. In BACOL, C^1-continuity at the internal mesh points is imposed. Consequently, $\nu = 2$, and the dimension of this piecewise polynomial subspace is $NC = N(p-1) + 2$.

To represent the piecewise polynomials, BACOL employs a B-spline basis. Let $\{B_j(x)\}_{j=1}^{NC}$ be the B-splines basis associated with the above mesh with C^1-continuity at the internal mesh points. The degree of the B-splines basis polynomials, p, $3 \leq p \leq$

11, is specified by the user. The approximate vector solution, $U(x,t)$, is expressed in the form

$$U(x,t) = \sum_{j=1}^{NC} w_j(t) B_j(x),$$

where $w_j(t)$ is the vector $(w_j(t) : R \rightarrow R^n)$ of time-dependent B-spline coefficients multiplying the jth (scalar) B-spline basis function.

An important property of the B-spline basis is that on any subinterval $[x_{i-1}, x_i)$, at most $p+1$ basis functions are non-zero, namely, $\{B_m\}_{m=(i-1)(p-1)+1}^{i(p-1)+2}$. Note that we then have

$$U(x,t) = \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t) B_m(x),$$

$$U_x(x,t) = \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t) B_m'(x), \tag{2.1}$$

$$U_{xx}(x,t) = \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t) B_m''(x).$$

The B-splines are implemented through a collection of Fortran subroutines built around an algorithm for the stable evaluation of B-splines of arbitrary degree and arbitrary continuity (the B-spline package [dB77, dB78]).

Let the mesh subinterval size sequences be $\{h_i\}_{i=1}^N$, where $h_i = x_i - x_{i-1}$, and let $\{\rho_i\}_{i=1}^{p-1}$ be the Gaussian points [BS73] on $[0,1]$ such that

$$0 < \rho_0 < \rho_1 < \cdots < \rho_{p-1} < 1.$$

The collocation points in the x domain are then defined by

$$\xi_1 = a, \quad \xi_l = x_{i-1} + h_i \rho_j, \quad \xi_{NC} = b,$$

where $l = 1 + (i-1) \cdot (p-1) + j$, for $i = 1, \cdots, N$, $j = 1, \cdots, p-1$.

The collocation method requires the approximate solution to satisfy the PDEs at the collocation point sequence, $\{\xi_i\}_{i=2}^{NC-1}$. Substituting the approximate solution and its derivatives into the 1D PDE and evaluating the resulting expression at ξ_l gives the collocation condition

$$U_t(\xi_l, t) = f(\xi_l, t, U(\xi_l, t), U_x(\xi_l, t), U_{xx}(\xi_l, t)). \tag{2.2}$$

Substituting for U, U_x and U_{xx} from (2.1) in (2.2) then gives

$$\sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m'(t) B_m(\xi_l) =$$

$$f\left(\xi_l, t, \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t) B_m(\xi_l), \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t) B_m'(\xi_l), \right. \tag{2.3}$$

$$\left. \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t) B_m''(\xi_l)\right),$$

where $\xi_l \in [x_{i-1}, x_i)$, $i = 1, \cdots N$, $j = 1, \cdots, p-1$, $l = 1 + (i-1) \cdot (p-1) + j$.

An important aspect of the BACOL code is that the boundary conditions are treated in their original form. We substitute the approximate solution and its first derivative into each of the boundary conditions; this gives

$$b_L(t, U(a,t), U_x(a,t)) = 0, \quad b_R(t, U(b,t), U_x(b,t)) = 0.$$

Rewriting the above using (2.1) gives

$$0 = b_L\left(t, \sum_{m=1}^{p+1} w_m(t)B_m(a), \sum_{m=1}^{p+1} w_m(t)B_m^{'}(a)\right), \tag{2.4}$$

$$0 = b_R\left(t, \sum_{m=(N-1)(p-1)+1}^{N(p-1)+2} w_m(t)B_m(b), \sum_{m=(N-1)(p-1)+1}^{N(p-1)+2} w_m(t)B_m^{'}(b)\right). \tag{2.5}$$

Considering the boundary conditions and collocation conditions together ((2.4), (2.3) and (2.5)), one obtains a DAE system ($N(p-1)n$ ODEs coupled with $2n$ algebraic equations) of the form

$$0 = b_L\left(t, \sum_{m=1}^{p+1} w_m(t)B_m(a), \sum_{m=1}^{p+1} w_m(t)B_m^{'}(a)\right),$$

$$\sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m^{'}(t)B_m(\xi_l) =$$

$$f\left(\xi_l, t, \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t)B_m(\xi_l), \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t)B_m^{'}(\xi_l),\right.$$

$$\left.\sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t)B_m^{''}(\xi_l)\right),$$

$$\xi_l \in [x_{i-1}, x_i), \quad l = 1 + (i-1)\cdot(p-1) + j, \quad i = 1, \cdots, N, \quad j = 1, \cdots, p-1,$$

$$0 = b_R\left(t, \sum_{m=(N-1)(p-1)+1}^{N(p-1)+2} w_m(t)B_m(b), \sum_{m=(N-1)(p-1)+1}^{N(p-1)+2} w_m(t)B_m'(b)\right).$$

We can write the above DAE system in a matrix system form:

$$A_x \underline{W}_t(t) = \underline{F}(t, \underline{W}(t)). \tag{2.6}$$

The top and bottom blocks of rows of A_x corresponding to the boundary conditions are zero. The internal block rows of A_x correspond to the collocation conditions. Figure 2.1 shows the ABD structure of A_x.

In Figure 2.1, the ith subblock, S_i, $i = 1, \cdots, N$, is a $n(p-1) \times n(p+1)$ matrix of the form,

$$\begin{bmatrix} B_l(\xi_{l+1})I_n & B_{l+1}(\xi_{l+1})I_n & \cdots & B_{l+p}(\xi_{l+1})I_n \\ B_l(\xi_{l+2})I_n & B_{l+1}(\xi_{l+2})I_n & \cdots & B_{l+p}(\xi_{l+2})I_n \\ \vdots & \vdots & \ddots & \vdots \\ B_l(\xi_{l+p-1})I_n & B_{l+1}(\xi_{l+p-1})I_n & \cdots & B_{l+p}(\xi_{l+p-1})I_n \end{bmatrix}, \tag{2.7}$$

where $l = 1 + (i-1)(p-1)$, and I_n is the $n \times n$ identity matrix.

The software COLROW (CRDCMP and CRSLVE) is employed in BACOL to implement an LU decomposition with modified alternate row and column elimination with partial pivoting to solve the ABD systems that arise.

The vector of unknown coefficients, $\underline{W}(t)$, has the form

$$\underline{W}(t) = \begin{bmatrix} w_1(t) \\ w_2(t) \\ \vdots \\ w_{NC}(t) \end{bmatrix}.$$

The right hand side vector of (2.6) is

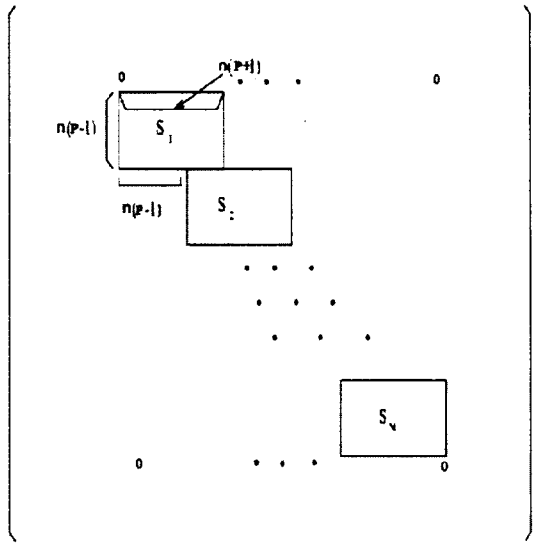

Figure 2.1: The ABD structure of A_x appearing in (2.6). The top and bottom are n by n blocks of zeros. Each block, S_i, is an $n(p-1)$ by $n(p+1)$ matrix. The overlap between the S_i blocks is $2n$, p is the degree of the piecewise polynomials, N is the number of subintervals, n is the number of PDEs.

$$\underline{F}(t,\underline{W}(t)) = \begin{bmatrix} b_L\left(t, \sum_{m=1}^{p+1} w_m(t)B_m(a), \sum_{m=1}^{p+1} w_m(t)B'_m(a)\right) \\ \vdots \\ f\left(\xi_l, t, \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t)B_m(\xi_l), \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t)B'_m(\xi_l), \right. \\ \left. \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} w_m(t)B''_m(\xi_l)\right), \\ \vdots \\ b_R\left(t, \sum_{m=(N-1)(p-1)+1}^{N(p-1)+2} w_m(t)B_m(b), \sum_{m=(N-1)(p-1)+1}^{N(p-1)+2} w_m(t)B'_m(b)\right) \end{bmatrix}.$$

1D B-spline Projection matrix

At the collocation points in the x domain,

$$U(\xi_l,t) = \sum_{i=1}^{NC} w_i(t)B_i(\xi_l), \quad l = 1,\cdots,NC. \tag{2.8}$$

We can rewrite (2.8) in matrix form as

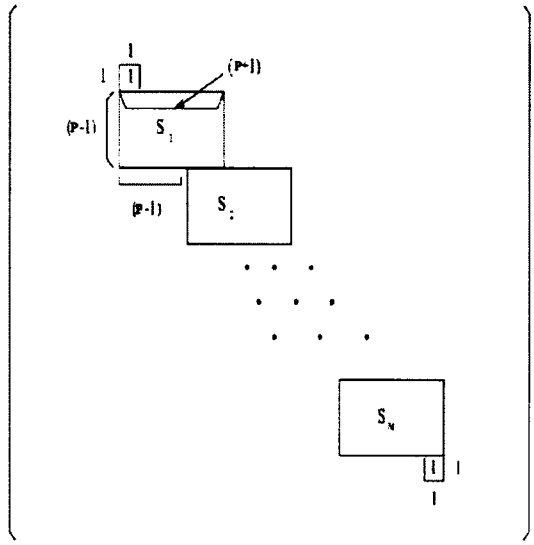

Figure 2.2: The ABD structure of the matrix, M_x, appearing in (2.9). Each block S_i is a $(p-1)$ by $(p+1)$ matrix. The overlap between the S_i blocks is 2, p is the degree of the piecewise polynomials, N is the number of subintervals.

$$\underline{U}(\underline{\xi}, t) = M_x \underline{W}(t), \tag{2.9}$$

where $\underline{\xi} = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_{NC} \end{bmatrix}$, and $\underline{U}(\underline{\xi}, t) = \begin{bmatrix} U(\xi_1, t) \\ U(\xi_2, t) \\ \vdots \\ U(\xi_{NC}, t) \end{bmatrix}$.

Figure 2.2 shows the structure of the matrix, M_x, appearing in (2.9)

In M_x, the ith subblock, S_i, $i = 1, \cdots, N$, still has the form (2.7), but I_n is replaced by 1.

Spatial Error Estimation

As we know, since error in the approximation of the PDE system is unavoidable, it is essential to assess the accuracy of the approximate solution. It is important to make a distinction between error control in time and error control in space. In BACOL, DASSL controls an estimate of the local temporal error at each time step.

BACOL computes a high order estimate of the spatial error, after each successful time step, and adjusts the spatial mesh so that the spatial error estimate satisfies the user tolerance for each successful time step.

In addition to the solution approximation, $U(x,t)$, BACOL computes a second global collocation solution, $\overline{U}(x,t)$, using degree $p+1$ piecewise polynomials on the same spatial mesh. BACOL obtains a posteriori spatial error estimates by comparing $U(x,t)$ and $\overline{U}(x,t)$ (see below). If the estimated error does not satisfy the user provided tolerance, then BACOL will generate a new mesh by approximately equidistributing the estimated error over each subinterval of a new mesh. The total number of mesh points may also be changed.

BACOL generally employs a warm restart after a remeshing, as suggested in [AFMW92]. This means that BACOL interpolates the B-spline coefficients related to degree $p+1$ solution, $\overline{U}(x,t)$, associated with the old mesh, to the new mesh at the current time step. Since the BDF methods employed in DASSL have up to a maximum order of 5, BACOL employs interpolation at the current step and, at most, the last 5 time steps to obtain the updated coefficient values. After several warm starts, if the spatial error estimate still does not satisfy the tolerance, BACOL will perform a cold start. This means that BACOL will continue using the same spatial mesh but will restart DASSL at order 1 using a very small time step.

BACOL computes two normalized spatial error estimates. The normalized spatial error estimate, $E_s(t)$, for the sth PDE component over the whole interval $[a,b]$, is

$$E_s(t) = \sqrt{\int_b^a \left(\frac{U_s(x,t) - \overline{U}_s(x,t)}{ATOL_s + RTOL_s |U_s(x,t)|} \right)^2 dx}, \quad s = 1, \cdots, n,$$

while the normalized error estimate for the ith subinterval over all s components is

$$\hat{E}_i(t) = \sqrt{\sum_{s=1}^{n} \int_{x_{i-1}}^{x_i} \left(\frac{U_s(x,t) - \overline{U}_s(x,t)}{ATOL_s + RTOL_s |U_s(x,t)|} \right)^2 dx}, \quad i = 1, \cdots, N.$$

where t is the time at the end of the current time step, $ATOL_s$, $RTOL_s$ are the absolute and relative tolerances for the sth PDE component, $U_s(x,t)$ is the sth component of the collocation solution of degree p, and $\overline{U}_s(x,t)$ is the sth component of the second collocation solution of degree $p+1$.

In order to assess the distribution of the error over the spatial subintervals, BACOL calculates three parameters related to the $\hat{E}_i(t)$ and $E_s(t)$ values,

$$r_1 = \max_{1\le i\le N}(\hat{E}_i(t))^{1/(p+1)}, \quad r_2 = \frac{\sum_{i=1}^{N}(\hat{E}_i(t))^{1/(p+1)}}{N},$$

$$\text{and } E(t) = \max_{1\le s\le NPDE} E_s(t),$$

where r_1 is related to the maximum error estimate over all subintervals and r_2 is related to the average error estimate over all the subintervals. Then the ratio r_1/r_2 gives an indication of whether the error is distributed approximately equally over the subintervals. The spatial error satisfies the tolerance when $E(t) \le 1$.

Time integration

As mentioned earlier, BACOL employs a modification of DASSL, which uses the BDF–linear multistep methods for the time integration of the DAEs arising from the discretized PDEs and the boundary conditions. DASSL employs a Newton iteration for the solution of the nonlinear systems that arise. The primary modifications to DASSL are as follows:

- Because the Newton matrices have an ABD structure, a new option for a linear solver was added to DASSL: COLROW (CRDCMP and CRSLVE) which implements an LU decomposition with modified alternate row and column elimination with partial pivoting to solve the ABD systems is now included as an option with DASSL.

- In DDASLV (one of the DASSL subroutines), the rows of the ABD system corresponding to the algebraic equations are scaled by $1/\Delta t$. This improves the conditioning of the Newton matrices. (Δt is the current time step).

Chapter 3

The Method of Surfaces (MOS)

The MOS is, to our knowledge, a new approach for solving 2D problems; it is an extension of the well known MOL. In the MOL, as explained in Chapter 2, we discretize the spatial domain, transforming a 1D PDE into a system of time-dependent DAEs and then use software for numerical time integration to integrate the DAE solution forward in time t. In a simple MOL implementation, we use a mesh of points to partition the x domain and at each mesh point, x_i, we approximate the solution, $u(x_i, t)$. The function $u(x_i, t)$ is a curve perpendicular to the x axis running along the surface $u(x, t)$.

The MOS is obtained from a natural generation of the MOL. In the 2D case, the domain of the solution $u(x, y, t)$ is a 3D volume (Figure 3.1). By discretizing, say, the y domain, we then consider a collection of surfaces $u(x, y_i, t)$, where y_i is a mesh point in the y domain. Then we approximate each of the surfaces and together these approximations give an approximation to $u(x, y, t)$. When we discretize only the y domain, we get a system of 1D PDEs, and then we can use a 1D PDE solver, e.g., BACOL, to solve this system of 1D PDEs.

Any standard method can be used for the discretization of y domain, e.g., finite difference methods, finite element methods, or collocation methods. We will next

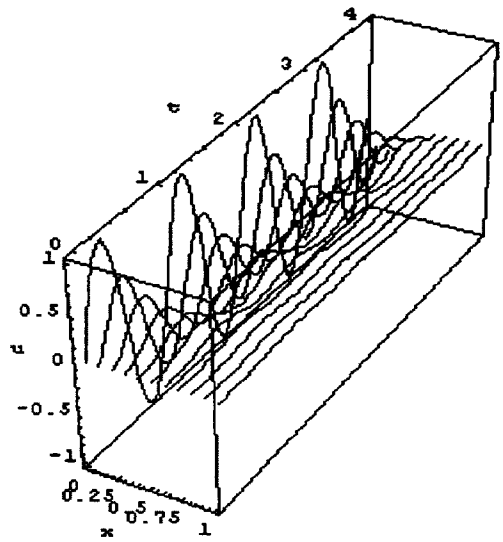

Figure 3.1: The Method of Lines: solution approximations for fixed x_i values are obtained for $t = 0 \cdots 4$. From [mol]

discuss further details for a finite difference type of MOS algorithm.

3.1 A Finite Difference based MOS Scheme

3.1.1 Discretization of the y domain using Finite Differences

Assume the general form for a 2D PDE given by (1.1). We consider the case of a uniform mesh in the y domain, but with a slight generalization of the approach a non-uniform mesh can be treated.

In order to apply finite differences in the y domain, we divide the y domain into M sections of equal length, $\triangle y = \frac{d-c}{M}$, and then $y_i = c + i\triangle y, i = 0, \ldots, M$. Let $\widehat{u}_i(x,t)$ be the numerical approximation to $u(x, y_i, t)$, $i = 0, \ldots, M$. Then $\widehat{u}_i(x,t)$ is the ith surface (i.e., "slice") of $u(x,y,t)$.

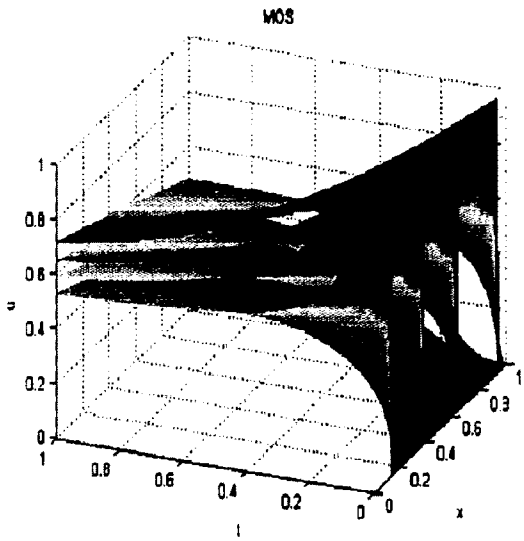

Figure 3.2: The Method of Surfaces: solution approximations for fixed y_i values are obtained for $t = 0 \cdots 1$

The 2D time-dependent parabolic PDE (1.1) can be discretized using central difference schemes for the first and second derivatives of y; for the ith surface, $\widehat{u}_i(x,t)$, we have

$$(\widehat{u}_i)_t(x,t) = f\left(x, y_i, t, \widehat{u}_i(x,t), (\widehat{u}_i)_x(x,t), \frac{\widehat{u}_{i+1}(x,t) - \widehat{u}_{i-1}(x,t)}{2\triangle y}, (\widehat{u}_i)_{xx}(x,t), \right.$$
$$\left. \frac{(\widehat{u}_{i+1})_x(x,t) - (\widehat{u}_{i-1})_x(x,t)}{2\triangle y}, \frac{\widehat{u}_{i+1}(x,t) - 2\widehat{u}_i(x,t) + \widehat{u}_{i-1}(x,t)}{\triangle y^2}\right), \; i = 1, \cdots, M-1, \tag{3.1}$$

for $(x,t) \in (a,b) \times (t, t_{out}]$. The initial conditions at $t = t_0$ associated with (3.1) are given by

$$\widehat{u}_i(x,t) = u_0(x, y_i), \; x \in [a,b], \; i = 1, \cdots, M-1. \tag{3.2}$$

The boundary conditions associated with (3.1) are

$$g_a\left(y_i, t, \widehat{u}_i(a,t), (\widehat{u}_i)_x(a,t), \frac{\widehat{u}_{i+1}(a,t) - \widehat{u}_{i-1}(a,t)}{2\triangle y}\right) = 0, \tag{3.3}$$

$$g_b\left(y_i, t, \widehat{u}_i(b,t), (\widehat{u}_i)_x(b,t), \frac{\widehat{u}_{i+1}(b,t) - \widehat{u}_{i-1}(b,t)}{2\triangle y}\right) = 0, \tag{3.4}$$

where $t \in (t, t_{out}]$, $i = 1, \ldots, M-1$. Thus, the solution approximations $\widehat{u}_i(x,t)$, $i = 1, \ldots, M-1$, are defined by the above system of 1D PDEs with associated boundary and initial conditions.

The remaining boundary conditions (not yet considered) are

$$g_c(x, t, u(x,c,t), u_x(x,c,t), u_y(x,c,t)) = 0,$$

$$g_d(x, t, u(x,d,t), u_x(x,d,t), u_y(x,d,t)) = 0,$$

for $t \in (t_0, t_{out}]$. Substituting $\hat{u}_0(x,t) \approx u(x, y_0, t) = u(x,c,t)$ and $\hat{u}_M(x,t) \approx u(x, y_M, t) = u(x,d,t)$ into these boundary conditions and employing one-sided finite differences, we get

$$g_c\left(x, t, \hat{u}_0(x,t), \frac{d}{dx}\hat{u}_0(x,t), \frac{\widehat{u}_2(x,t) - \widehat{u}_0(x,t}{2\triangle y}\right) = 0, \tag{3.5}$$

$$g_d\left(x, t, \hat{u}_M(x,t), \frac{d}{dx}\hat{u}_M(x,t), \frac{\widehat{u}_M(x,t) - \widehat{u}_{M-2}(x,t}{2\triangle y}\right) = 0. \tag{3.6}$$

We will assume that we can explicitly solve the above equations, (3.5), (3.6), to obtain $\widehat{u}_0(x,t)$ *and* $\widehat{u}_M(x,t)$*, and then these functions can be substituted in the PDEs (3.1) and boundary conditions (3.3), (3.4). This assumption requires that the boundary conditions have the simpler form*

$$g_c\left(x, t, \hat{u}_0(x,t)\right) = 0, \; g_d\left(x, t, \hat{u}_M(x,t)\right) = 0.$$

(Alternatively, we could consider (3.5), (3.6), to be algebraic equations and combine them with the PDEs (3.1). However, this algebraic/1D PDE system does not have a form that can be solved by BACOL.)

In summary, we have a coupled system of $(M-1)$ 1D time-dependent PDEs (3.1) to be solved by BACOL, with initial conditions (3.2) and boundary conditions (3.3), (3.4) (where we have explicitly obtained $\widehat{u}_0(x,t)$ and $\widehat{u}_M(x,t)$ from (3.3), (3.4).)

3.1.2 Finite Difference Based MOS Example

In this section we consider an example, using BACOL to solve the system of 1D time-dependent parabolic PDEs arising from the application of the finite difference based MOS.

Consider the 2D time-dependent parabolic PDE (the 2D Burgers' equation), (where ξ is a constant),

$$\frac{\partial u}{\partial t} = \xi\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - u\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right).$$

The problem domain is $(x,y) \in (0,1) \times (0,1)$, $t > 0$; the boundary and initial conditions chosen so that the true solution is

$$u(x,y,t) = \frac{1}{1+e^{\frac{x+y-t}{2\xi}}}.$$

The initial condition is

$$u(x,0) = \frac{1}{1+e^{\frac{x+y}{2\xi}}}, (x,y) \in [0,1] \times [0,1],$$

and the boundary conditions are

$$u(0,y,t) - \frac{1}{1+e^{\frac{y-t}{2\xi}}} = 0, \quad t \in (0,1), y \in (0,1), \quad (g_a = 0)$$

$$u(1,y,t) - \frac{1}{1+e^{\frac{1+y-t}{2\xi}}} = 0, \quad t \in (0,1), y \in (0,1), \quad (g_b = 0)$$

$$u(x,0,t) - \frac{1}{1+e^{\frac{x-t}{2\xi}}} = 0, \quad t \in (0,1), x \in (0,1), \quad (g_c = 0)$$

$$u(x,1,t) - \frac{1}{1+e^{\frac{1+x-t}{2\xi}}} = 0, \quad t \in (0,1), x \in (0,1). \quad (g_d = 0)$$

Assume we divide the y domain into M subintervals; let $\triangle y = \frac{1}{M}$. Then for the internal points, y_i, the discretization of the y domain using the finite difference method gives the following system of 1D parabolic time-dependent PDEs, involving the unknown solution components, $\widehat{u}_i(x,t)$, $i = 1, \ldots, M-1$,

$$(\hat{u}_i)_t = \xi(\hat{u}_i)_{xx} - \hat{u}_i(\hat{u}_i)_x + \xi\left(\frac{\hat{u}_{i+1} - 2\hat{u}_i + \hat{u}_{i-1}}{\triangle y^2}\right) + \hat{u}_i\left(\frac{\hat{u}_{i+1} - \hat{u}_{i-1}}{2\triangle y}\right), i = 1, \ldots, M-1. \tag{3.7}$$

The corresponding initial conditions for $\widehat{u}_i(x,t)$ are

$$\hat{u}_i(x,0) = \frac{1}{\left(1+e^{\frac{x+y_i}{2\xi}}\right)}, \quad i = 1, \cdots, M-1. \tag{3.8}$$

The boundary conditions for $\widehat{u}_i(x,t)$ are

$$\hat{u}_i(0,t) - \frac{1}{\left(1+e^{\frac{y_i-t}{2\xi}}\right)} = 0, \quad i = 1, \cdots, M-1, \tag{3.9}$$

$$\hat{u}_i(1,t) - \frac{1}{\left(1+e^{\frac{1+y_i-t}{2\xi}}\right)} = 0, \quad i = 1, \cdots, M-1. \tag{3.10}$$

To obtain $\hat{u}_0(x,t)$ and $\hat{u}_M(x,t)$, we apply the remaining boundary conditions ($g_c = 0$, $g_d = 0$); we get,

$$\hat{u}_0(x,t) = \frac{1}{1+e^{\frac{x-t}{2\xi}}}, \quad \hat{u}_M(x,t) = \frac{1}{1+e^{\frac{1+x-t}{2\xi}}}. \tag{3.11}$$

These can be substituted into (3.7).

Then the 1D PDE system (3.7) (with the substitutions (3.11)) with initial conditions (3.8) and boundary conditions (3.9), (3.10) can be provided to BACOL.

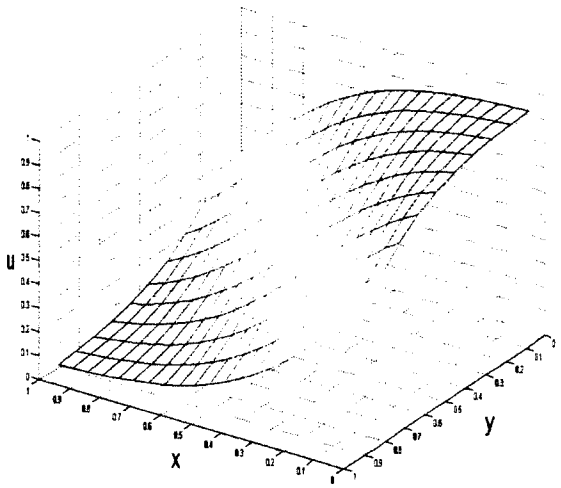

Figure 3.3: Approximate solution of the 2D Burgers' equation, as computed by the finite difference based MOS, for $\xi = 0.1$ at $t = 1$.

BACOL can then be used to solve the 1D system from $t = 0$ to $t = 1$. This gives the solution approximations $\hat{u}_i(x,t)$, $i = 1, \cdots, 11$.

We set $\xi = 0.1$, and $M = 11$. The solution components $\hat{u}_i(x,t) \approx u(x, y_i, t)$, for $i = 1, \cdots, 11$, at $t = 1$ are plotted in Figure 3.3. The source code for solving this problem with BACOL is given in Appendix A.1.

3.2 A B-spline Gaussian Collocation MOS Scheme

Here we discuss details for a B-spline Gaussian Collocation MOS Scheme.

3.2.1 Discretization of the y domain using B-spline Gaussian Collocation

Let $\{y_i\}_{i=0}^{M}$ be a spatial mesh in $[c,d]$, with $c = y_0 < y_1 < \cdots < y_M = d$. We define a piecewise polynomial of degree q on each subinterval $[y_{i-1}, y_i]$, $\quad i = 1, \cdots, M$, and require the piecewise polynomial to be C^1–continuous at the internal mesh points. We will implement the piecewise polynomials using a B-spline basis. Consequently

the dimension of this piecewise polynomial subspace is $MC = M(q-1) + 2$.

Let $\{\eta_i\}_{i=1}^{q-1}$ be the canonical Gaussian points on $[0,1]$; $0 < \eta_1 < \cdots < \eta_{q-1} < 1$. The images of these points within each subinterval will be the collocation points in the y domain. The collocation method requires the approximate solution to satisfy the PDE at the internal collocation points and the boundary conditions at $y = c$ and $y = d$.

The approximate solution, $U(x, y, t)$, is represented in the form:

$$U(x, y, t) = \sum_{j=1}^{MC} \widehat{c}_j(x, t) B_j(y), \tag{3.12}$$

where $B_j(y)$ is the jth B-spline basis function and $\widehat{c}_j(x, t)$ is the corresponding (unknown) x and t dependent B-spline basis coefficient. The B-spline basis has the property that for any y such that $y_{i-1} < y < y_i, 1 \leqslant i \leqslant M$, at most $q + 1$ B-spline basis functions, namely, $B_m(y)_{m=(i-1)(q-1)+1}^{i(q-1)+2}$, have nonzero values.

We define the mesh subinterval size sequences $\{k_i\}_{i=1}^{M}$ by $k_i = y_i - y_{i-1}$. The collocation points in the y domain are defined to be

$\gamma_1 = c$,

$\gamma_l = y_{i-1} + k_i \eta_j$, where $l = 1 + (i-1)(q-1) + j$, $i = 1, \cdots, M$, $j = 1, \cdots, q-1$.

$\gamma_{MC} = d$.

The collocation conditions associated with the original 2D PDE yield the following 1D PDEs:

$$U_t(x, \gamma_l, t) = f\left(x, \gamma_l, t, U(x, \gamma_l, t), U_x(x, \gamma_l, t), U_y(x, \gamma_l, t),\right.$$

$$\left. U_{xx}(x, \gamma_l, t), U_{xy}(x, \gamma_l, t), U_{yy}(x, \gamma_l, t)\right), \tag{3.13}$$

where $l = 1 + (i-1)(q-1) + j$, $i = 1, \cdots, M$, $j = 1, \cdots, q-1$, $(l = 2, \cdots, MC - 1)$.

Substitution of (3.12) into (3.13) gives (taking into account the special property of the B-spline basis mentioned above)

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_t B_m(\gamma_l) =$$

$$f\left(x, \gamma_l, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m(\gamma_l) , \right.$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m'(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_{xx} B_m(\gamma_l),$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m'(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m''(\gamma_l) \right) , \tag{3.14}$$

$\gamma_l \in [y_{l-1}, y_l), \quad l = 1+(i-1)(q-1)+j, \ i = 1, \cdots, M, \ j = 1, \cdots, q-1, \ (l = 2, \cdots, MC-1),$

for $(x,t) \in (a,b) \times (t, t_{out}]$. The collocated initial conditions (obtained by requiring the approximate solution to satisfy the initial condition at the collocation points in the y domain) at $t = t_0$ are given by

$$U(x, \gamma_l, t_0) = U_0(x, \gamma_l), \tag{3.15}$$

$$l = 1 + (i-1)(q-1) + j, \quad i = 1, \cdots, M,$$

$$j = 1, \cdots, q-1, \quad (l = 2, \cdots, MC-1), \quad x \in [a,b].$$

Substitution of (3.12) into (3.15) gives:

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(\gamma_l) = U_0(x, \gamma_l), \tag{3.16}$$

$\gamma_l \in [y_{l-1}, y_l), \quad l = 1+(i-1)(q-1)+j, \ i = 1, \cdots, M, \ j = 1, \cdots, q-1, \ (l = 2, \cdots, MC-1).$

The collocated boundary conditions are:

$$g_a(\gamma_l, t, U(a, \gamma_l, t), U_x(a, \gamma_l, t), U_y(a, \gamma_l, t)) = 0, \tag{3.17}$$

$$g_b(\gamma_l, t, U(b, \gamma_l, t), U_x(b, \gamma_l, t), U_y(b, \gamma_l, t)) = 0, \tag{3.18}$$

$$l = 1 + (i-1)(q-1) + j,\ i = 1, \cdots, M,\ j = 1, \cdots, q-1,\ (l = 2, \cdots, MC-1).$$

Substitution of (3.12) into (3.17), (3.18) gives:

$$g_a\left(\gamma_l, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(a,t))_x B_m(\gamma_l), \right.$$
$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m'(\gamma_l)\right) = 0, \tag{3.19}$$

$$g_b\left(\gamma_l, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(b,t))_x B_m(\gamma_l), \right.$$
$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m'(\gamma_l)\right) = 0, \tag{3.20}$$

$$l = 1 + (i-1)(q-1) + j,\ i = 1, \cdots, M,\ j = 1, \cdots, q-1,\ (l = 2, \cdots, MC-1).$$

Thus (3.14) represents a system of $(M-2)$ 1D PDEs together with $(M-2)$ initial conditions (3.16), and $2(M-2)$ boundary conditions (3.19), (3.20).

Since the dimension of the piecewise polynomial space is $MC = M(q-1)+2$, we still need two more PDEs with corresponding initial and boundary conditions. And we still have to consider the following two boundary conditions associated with $y = c$, $y = d$:

$$g_c(x, t, u(x, c, t), u_x(x, c, t), u_y(x, c, t)) = 0, \tag{3.21}$$

$$g_d(x, t, u(x, d, t), u_x(x, d, t), u_y(x, d, t)) = 0. \tag{3.22}$$

We can differentiate these two boundary conditions with respect to t to get two more PDEs:

$$g_c(x,t,U(x,c,t),U_x(x,c,t),U_y(x,c,t))_t = 0, \tag{3.23}$$

$$g_d(x,t,U(x,d,t),U_x(x,d,t),U_y(x,d,t))_t = 0. \tag{3.24}$$

We assume that (3.23) and (3.24) can be rewritten in the form:

$$U_t(x,c,t) = f_c(x,t,U(x,c,t),U_x(x,c,t),U_y(x,c,t)), \tag{3.25}$$

$$U_t(x,d,t) = f_d(x,t,U(x,d,t),U_x(x,d,t),U_y(x,d,t). \tag{3.26}$$

Substitution of (3.12) into (3.25), (3.26) gives:

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_t B_m(c) = f_c\left(x,t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t)B_m(c), \right. \tag{3.27}$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m(c), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t)B_m^{'}(c)\right),$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_t B_m(d) = f_d\left(x,t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t)B_m(d), \right. \tag{3.28}$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m(d), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t)B_m^{'}(d)\right).$$

The initial conditions corresponding to (3.27), (3.28) at $t = t_0$ are given by

$$U(x,c,t_0) = U_0(x,c), \tag{3.29}$$

$$U(x,d,t_0) = U_0(x,d). \tag{3.30}$$

Substitution of (3.12) into (3.29), (3.30) gives:

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t)B_m(c) = U_0(x,c), \tag{3.31}$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t)B_m(d) = U_0(x,d). \tag{3.32}$$

The boundary conditions corresponding to (3.27), (3.28) are (respectively)

$$g_a(c,t,U(a,c,t),U_x(a,c,t),U_y(a,c,t)) = 0, \tag{3.33}$$

$$g_b(c,t,U(b,c,t),U_x(b,c,t),U_y(b,c,t)) = 0, \tag{3.34}$$

$$g_a(d,t,U(a,d,t),U_x(a,d,t),U_y(a,d,t)) = 0, \tag{3.35}$$

$$g_b(d,t,U(b,d,t),U_x(b,d,t),U_y(b,d,t)) = 0. \tag{3.36}$$

Substitution of (3.12) into (3.33), (3.35), (3.34) and (3.36) gives:

$$g_a\left(c,t,\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m(c),\right.$$

$$\left.\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(a,t))_x B_m(c), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m'(c)\right) = 0, \tag{3.37}$$

$$g_b\left(c,t,\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m(c),\right.$$

$$\left.\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(b,t))_x B_m(c), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m'(c)\right) = 0; \tag{3.38}$$

$$g_a\left(d,t,\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m(d),\right.$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(a,t))_x B_m(d), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t) B_m'(d) \right) = 0, \tag{3.39}$$

$$g_b \left(d, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t) B_m(d), \right.$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(b,t))_x B_m(d), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t) B_m'(d) \right) = 0. \tag{3.40}$$

Thus the system of PDEs (3.14) together with the PDEs (3.27) and (3.28) give a 1D system of MC PDEs with corresponding initial and boundary conditions.

We arrange this system of 1D PDEs in the following order ((3.27), (3.14), (3.28)):

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_t B_m(c) = f_c \left(x, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(c), \right.$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m(c), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m'(c) \right) \tag{3.41}$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_t B_m(\gamma_l) =$$

$$f \left(x, \gamma_l, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m(\gamma_l) , \right.$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m'(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_{xx} B_m(\gamma_l), \tag{3.42}$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m'(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m''(\gamma_l) \right) ,$$

$$l = 1 + (i-1)(q-1) + j,\ i = 1, \cdots, M,\ j = 1, \cdots, q-1,\ (l = 2, \cdots, MC-1).$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_t B_m(d) = f_d \left(x, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(d), \right.$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(x,t))_x B_m(d), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m'(d) \right). \qquad (3.43)$$

The initial conditions arranged in a consistent ordering ((3.31), (3.16), (3.32)) are:

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(\gamma_1) = U_0(x, \gamma_1), \quad (\gamma_1 = c)$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(\gamma_l) = U_0(x, \gamma_l),$$

$$l = 1 + (i-1)(q-1) + j,\ i = 1, \cdots, M,\ j = 1, \cdots, q-1,\ (l = 2, \cdots, MC-1),$$

$$\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(x,t) B_m(\gamma_{MC}) = U_0(x, \gamma_{MC}). \quad (\gamma_{MC} = d)$$

The boundary conditions arranged according to the same ordering are ((3.37), (3.19), (3.39), (3.38), (3.20), (3.40)):

$$g_a \left(\gamma_1, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t) B_m(\gamma_1), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(a,t))_x B_m(\gamma_1), \right.$$

$$\left. \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t) B_m'(\gamma_1) \right) = 0, \quad (\gamma_1 = c)$$

$$g_a\left(\gamma_l, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(a,t))_x B_m(\gamma_l),\right.$$

$$\left.\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m'(\gamma_l)\right) = 0,$$

$$l = 1 + (i-1)(q-1) + j,\ i = 1, \cdots, M,\ j = 1, \cdots, q-1,\ (l = 2, \cdots, MC-1),$$

$$g_a\left(\gamma_{MC}, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m(\gamma_{MC}), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(a,t))_x B_m(\gamma_{MC}),\right.$$

$$\left.\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(a,t)B_m'(\gamma_{MC})\right) = 0, \quad (\gamma_{MC} = d)$$

$$g_b\left(\gamma_1, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m(\gamma_1), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(b,t))_x B_m(\gamma_1),\right.$$

$$\left.\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m'(\gamma_1)\right) = 0, \quad (\gamma_1 = c)$$

$$g_b\left(\gamma_l, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m(\gamma_l), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(b,t))_x B_m(\gamma_l),\right.$$

$$\left.\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m'(\gamma_l)\right) = 0,$$

$$l = 1 + (i-1)(q-1) + j,\ i = 1, \cdots, M,\ j = 1, \cdots, q-1,\ (l = 2, \cdots, MC-1),$$

$$g_b\left(\gamma_{MC}, t, \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B_m(\gamma_{MC}), \sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} (\widehat{c}_m(b,t))_x B_m(\gamma_{MC}),\right.$$

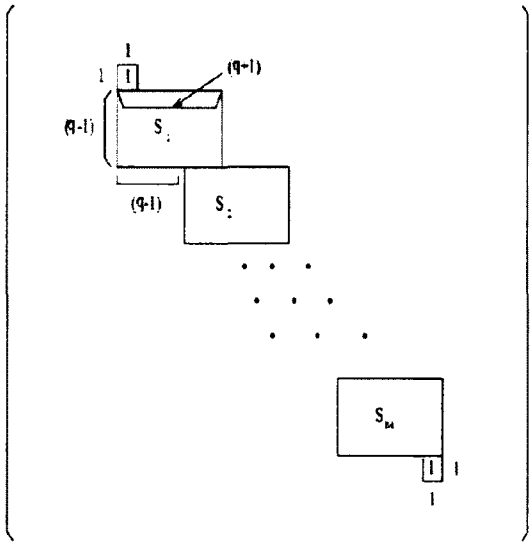

Figure 3.4: The ABD structure of G_y. Each block S_i is a $(q-1)$ by $(q+1)$ matrix. The overlap between the blocks, S_i, is 2, q is the degree of the B-spline basis polynomials, M is the number of subintervals. Comparing with (3.27), all B-spline basis functions are zero except $B_1(c) = 1$. Similarly, comparing with (3.28), all B-spline basis functions are zero except $B_{MC}(d) = 1$. These observations explain the form of the first and last rows of G_y.

$$\left.\sum_{m=(i-1)(q-1)+1}^{i(q-1)+2} \widehat{c}_m(b,t)B'_m(\gamma_{MC})\right) = 0. \quad (\gamma_{MC} = d)$$

The system of MC PDEs given above ((3.41), (3.42), (3.43)) in matrix form is

$$G_y(\underline{\hat{c}}(x,t))_t = \underline{F}(t, \underline{\hat{c}}(x,t)), \tag{3.44}$$

where G_y is a square ABD matrix of the form given in Figure 3.4 and $\underline{\hat{c}}(x,t)$ and $\underline{F}(t, \underline{\hat{c}}(x,t))$ are defined below. In Figure 3.4, each S_i, $i = 1, \ldots, M$, is a $q-1$ by $q+1$

matrix of the form

$$\begin{bmatrix} B_l(\gamma_{l+1}) & B_{l+1}(\gamma_{l+1}) & \cdots & B_{l+q}(\gamma_{l+1}) \\ B_l(\gamma_{l+2}) & B_{l+1}(\gamma_{l+2}) & \cdots & B_{l+q}(\gamma_{l+2}) \\ \vdots & \vdots & \ddots & \vdots \\ B_l(\gamma_{l+q-1}) & B_{l+1}(\gamma_{l+q-1}) & \cdots & B_{l+q}(\gamma_{l+q-1}) \end{bmatrix},$$

where $l = 1 + (i-1)(q-1)$, $i = 1, \cdots, M$. The vectors $\hat{\underline{c}}(x,t)$ and $\underline{F}(t, \hat{\underline{c}}(x,t))$ are

$$\hat{\underline{c}}(x,t) = \begin{bmatrix} \widehat{c}_1(x,t) \\ \widehat{c}_2(x,t) \\ \vdots \\ \widehat{c}_{MC}(x,t) \end{bmatrix}, \quad \underline{F}(t, \hat{\underline{c}}(x,t)) = \begin{bmatrix} F_1(t, \hat{\underline{c}}(x,t)) \\ F_2(t, \hat{\underline{c}}(x,t)) \\ \vdots \\ F_{MC}(t, \hat{\underline{c}}(x,t)) \end{bmatrix},$$

where $F_1(t, \hat{\underline{c}}(x,t))$ is the right hand side of (3.27), $F_l(t, \hat{\underline{c}}(x,t))$, $l = 2, \cdots, MC-1$, is the right hand side of (3.14), and $F_{MC}(t, \hat{\underline{c}}(x,t))$ is the right hand side of (3.28)

Then we give the 1D time-dependent PDE system (3.44) to BACOL along with the corresponding initial and boundary conditions indicated above.

3.2.2 Numerical Results for the B-spline Gaussian Collocation MOS Scheme

As explained earlier, since BACOL can currently only handle a 1D PDE system, we have to differentiate the boundary conditions, (3.21), (3.22), and assume the resulting equations have the form (3.27), (3.28). As we mentioned in Chapter 2, before the DAE solvers emerged, MOL software required the user to differentiate the boundary conditions to get systems of ODEs. The use of the differentiated boundary conditions introduces error, even in 1D case, where the boundary conditions correspond to only two points. In the 2D case, the boundary conditions we have to differentiate corre-

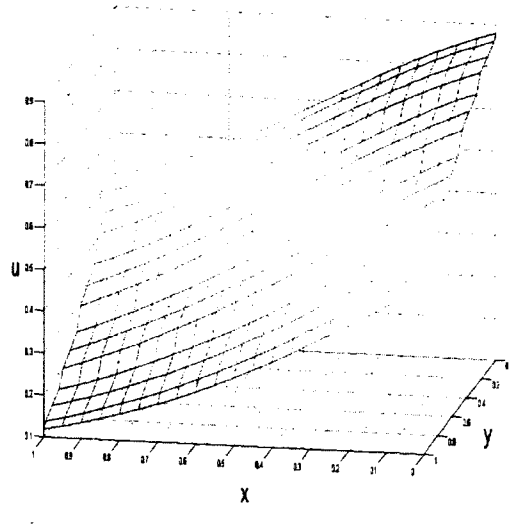

Figure 3.5: Approximate solution of 2D Burgers' equation, as computed by the B-spline based MOS, $\xi = 0.25$, $t = 1$.

spond to two lines and are collocated. The differentiated boundary conditions are solved together with the 1D PDEs obtained from collocation of the 2D PDEs. This introduces more error than in the 1D case.

For the numerical results presented here, we simplify the computation by collocating the PDE rather than the differentiated boundary conditions in order to obtain the two extra PDEs that are required.

We consider the same problem as in Section 3.1.2. We set $\xi = 0.25$, $q = 4$, $M = 4$. The numerical solution is plotted in Figure 3.5, for $\xi = 0.25$ and $t = 1$. The source code for solving this problem with BACOL is given in Appendix A.2.

Chapter 4

B-spline Gaussian Collocation for 2D Time-Dependent Parabolic PDEs

4.1 Introduction

In this chapter, we consider a B-spline Gaussian collocation algorithm in which we simultaneously discretize both spatial domains, x and y, transforming a 2D time-dependent PDE (1.1) into a system of time-dependent DAEs. We then use a numerical DAE integrator to integrate forward in time. (Thus this algorithm can be viewed as the classic MOL algorithm applied to a 2D PDE.) The approximate solution is expressed in terms of a tensor product B-spline basis in x and y with unknown time-dependent coefficients for the linear combination of the spatial basis elements. To perform the spatial discretization, we use a mesh of points to partition the x and y domains into spatial elements, and at selected points inside each rectangular element, we require the approximate solution to satisfy the PDE. We also require the approximate solution to satisfy the boundary conditions at selected points along the boundary.

In the remainder of this chapter, we will describe the 2D B-spline collocation

algorithm and its implementation in a preliminary software package which we call BACOL2D. Since, in the two-dimensional case, the DAE system resulting from the spatial discretization is usually large, we use DASPK to solve it. DASPK is based on BDFs and is a newer version of the well-known DASSL code, used in BACOL. Since BACOL2D is a natural extension of BACOL, many of the algorithms employed in BACOL are relevant to BACOL2D. We also describe a fast block LU algorithm with modified alternate row and column elimination with partial pivoting for the treatment of the ABD linear systems arising during the numerical solution of the DAEs. We will present numerical results in this chapter to demonstrate convergence rates for the collocation solution.

4.2 Spatial Discretization

We will assume $n = 1$ to simplify the discussion, but the algorithm can be used for arbitrary n.

4.2.1 B-spline basis

We consider a 2D rectangular grid based on a mesh of $N+1$ points ($N > 1$) in $[a, b]$ and a mesh of $M+1$ points ($M > 1$) in $[c, d]$ such that

$$a = x_0 < x_1 < \cdots < x_N = b, \quad c = y_0 < y_1 < \cdots < y_M = d.$$

We associate with the mesh on the x domain, C^1-continuous piecewise polynomials of degree p, i.e., we have a polynomial of degree p for the ith subinterval, $[x_{i-1}, x_i], i = 1, \cdots, N$, with C^1-continuity imposed at the internal mesh points. Consequently the dimension of this piecewise polynomial subspace is $NC = N(p-1) + 2$. Similarly, in the y domain, we have a polynomial of degree q for the ith subinterval, $[y_{i-1}, y_i], i = 1, \cdots, M$, with C^1-continuity imposed at the internal mesh points. Consequently the dimension of this piecewise polynomial subspace is $MC = M(q-1) + 2$.

To represent the piecewise polynomials, we employ B-spline bases. Let $\{B_i(x)\}_{i=1}^{NC}$, $\{D_i(y)\}_{i=1}^{MC}$, be the B-splines bases associated with the above meshes and continuity requirements. We then write the numerical solution, $U(x,y,t)$, as a linear combination of the tensor product of the B-spline bases functions in x and y with time-dependent coefficients, $w_{ij}(t)$, as follows:

$$U(x,y,t) = \sum_{i=1}^{NC}\sum_{j=1}^{MC} w_{ij}(t)B_i(x)D_j(y). \tag{4.1}$$

4.2.2 Collocation at Gaussian Points

We define the mesh subinterval size sequences $\{h_i\}_{i=1}^{N}$ by $h_i = x_i - x_{i-1}$, and $\{k_i\}_{i=1}^{M}$ by $k_i = y_i - y_{i-1}$. We define $\{\rho_i\}_{i=1}^{p-1}$ and $\{\eta_i\}_{i=1}^{q-1}$ to be the canonical Gaussian points on $[0,1]$ with $0 < \rho_1 < \cdots < \rho_{p-1} < 1$, and $0 < \eta_1 < \cdots < \eta_{q-1} < 1$. The collocation points in the x domain are then defined by

$\xi_1 = a,$

$\xi_l = x_{i-1} + h_i\rho_j,\ l = 1 + (i-1)\cdot(p-1) + j,\ i = 1,\cdots,N,\ j = 1,\cdots,p-1,$

$\xi_{NC} = b.$

The collocation points in the y domain are defined to be

$\gamma_1 = c,$

$\gamma_l = y_{i-1} + k_i\eta_j,\ l = 1 + (i-1)\cdot(q-1) + j,\ i = 1,\cdots,M,\ j = 1,\cdots,q-1,$

$\gamma_{MC} = d.$

In Figure 4.1, we show the collocation points that are inside each rectangle.

The PDE is discretized over the spatial domain by simultaneously collocating at the points $\{\xi_i\}_{i=2}^{NC-1}$ in x and the points $\{\gamma_j\}_{j=2}^{MC-1}$ in y. The collocation conditions yield the following ODEs in time:

$$U_t(\xi_i,\gamma_j,t) = f(\xi_i,\gamma_j,t,U(\xi_i,\gamma_j,t),U_x(\xi_i,\gamma_j,t),U_y(\xi_i,\gamma_j,t),$$

$$U_{xx}(\xi_i,\gamma_j,t),U_{xy}(\xi_i,\gamma_j,t),U_{yy}(\xi_i,\gamma_j,t)), \tag{4.2}$$

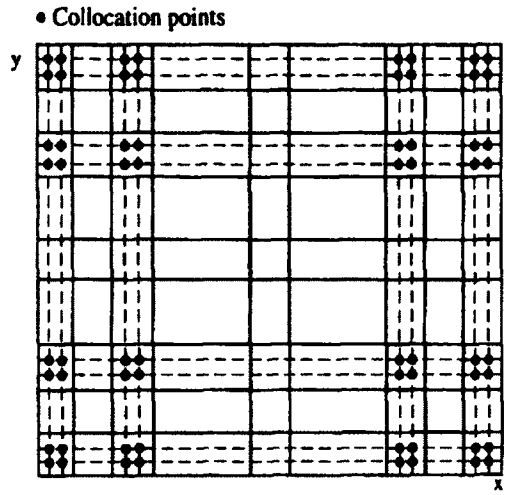

Figure 4.1: For the case $p = 3$, $q = 3$, the points inside each rectangle are the collocation points. Note that the collocation points on the boundaries are not shown in this figure. From [PW95].

where $i = 2, \cdots, NC - 1$, and $j = 2, \cdots, MC - 1$. Note that these collocation conditions do not involve $\xi_1, \xi_{NC}, \gamma_1$, or γ_{MC}. These latter points are associated with applying collocation conditions along the boundaries. These conditions are

$$0 = g_a(\gamma_j, t, U(a, \gamma_j, t), U_x(a, \gamma_j, t), U_y(a, \gamma_j, t)), \quad (\xi_1 = a), \tag{4.3}$$

$$0 = g_b(\gamma_j, t, U(b, \gamma_j, t), U_x(b, \gamma_j, t), U_y(b, \gamma_j, t)), \quad (\xi_{NC} = b), \tag{4.4}$$

$$0 = g_c(\xi_i, t, U(\xi_i, c, t), U_x(\xi_i, c, t), U_y(\xi_i, c, t)), \quad (\gamma_1 = c), \tag{4.5}$$

$$0 = g_d(\xi_i, t, U(\xi_i, d, t), U_x(\xi_i, d, t), U_y(\xi_i, d, t)), \quad (\gamma_{MC} = d), \tag{4.6}$$

where $i = 1, \cdots, NC$, $j = 1, \cdots, MC$.

We next describe the order in which the collocation conditions appear in the DAE system provided to DASPK. The algebraic equations (4.3), (4.4) are the first and last sets of equations of the DAE system. Note we separate the other two sets of boundary conditions (4.5), (4.6), and combine them with the ODEs (4.2). Then the resulting

DAE system is (4.8).

Next by substituting (4.1) into (4.2), (4.3), (4.4), (4.5) and (4.6), we get equations in the terms of the unknowns $w_{ij}(t)$. We can then rewrite these equations in matrix form:

$$A(\underline{W}(t))_t = \underline{F}(t, \underline{W}(t)). \tag{4.7}$$

In (4.7), $\underline{W}(t)$ is the B-spline coefficient vector of size $(NC \cdot MC)$; it has the form,

$$\underline{W}(t) = \begin{bmatrix} \underline{w}_1(t) \\ \underline{w}_2(t) \\ \vdots \\ \underline{w}_{NC}(t) \end{bmatrix}, \text{ where } \underline{w}_i(t) = \begin{bmatrix} w_{i1}(t) \\ w_{i2}(t) \\ \vdots \\ w_{iMC}(t) \end{bmatrix}.$$

The right hand side vector, $\underline{F}(t, \underline{W}(t)) = \begin{bmatrix} F_1(t, \underline{W}(t)) \\ F_2(t, \underline{W}(t)) \\ \vdots \\ F_{NC}(t, \underline{W}(t)) \end{bmatrix}$, where each $\underline{F}_i(t, \underline{W}(t))$ has MC components. The expressions for $\underline{F}_i(t, \underline{W}(t))$, $i = 1, \cdots, NC$, are given in (4.9), (4.10) and (4.11). These expressions are not in terms of $w_{ij}(t)$, but such expressions could easily be obtained by substituting (4.1) and its derivatives into the following formulas.

$$
\begin{aligned}
0 &= g_a(\gamma_1, t, U(\xi_1, \gamma_1, t), U_x(\xi_1, \gamma_1, t), U_y(\xi_1, \gamma_1, t)), \\
&\cdots \\
0 &= g_a(\gamma_{MC}, t, U(\xi_1, \gamma_{MC}, t), U_x(\xi_1, \gamma_{MC}, t), U_y(\xi_1, \gamma_{MC}, t)), \\
&- \\
0 &= g_c(\xi_2, t, U(\xi_2, \gamma_1, t), U_x(\xi_2, \gamma_1, t), U_y(\xi_2, \gamma_1, t)), \\
U_t(\xi_2, \gamma_2, t) &= f(\xi_2, \gamma_2, t, U(\xi_2, \gamma_2, t), U_x(\xi_2, \gamma_2, t), U_y(\xi_2, \gamma_2, t), \\
&U_{xx}(\xi_2, \gamma_2, t), U_{xy}(\xi_2, \gamma_2, t), U_{yy}(\xi_2, \gamma_2, t)), \\
&\cdots \\
U_t(\xi_2, \gamma_{MC-1}, t) &= f(\xi_2, \gamma_{MC-1}, t, U(\xi_2, \gamma_{MC-1}, t), U_x(\xi_2, \gamma_{MC-1}, t), \\
U_y(\xi_2, \gamma_{MC-1}, t), &U_{xx}(\xi_2, \gamma_{MC-1}, t), U_{xy}(\xi_2, \gamma_{MC-1}, t), U_{yy}(\xi_2, \gamma_{MC-1}, t)), \\
0 &= g_d(\xi_2, t, U(\xi_2, \gamma_{MC}, t), U_x(\xi_2, \gamma_{MC}, t), U_y(\xi_2, \gamma_{MC}, t)), \\
&- \\
&\cdots \\
&- \\
0 &= g_c(\xi_{NC-1}, t, U(\xi_{NC-1}, \gamma_1, t), U_x(\xi_{NC-1}, \gamma_1, t), U_y(\xi_{NC-1}, \gamma_1, t)), \\
U_t(\xi_{NC-1}, \gamma_2, t) &= f(\xi_{NC-1}, \gamma_2, t, U(\xi_{NC-1}, \gamma_2, t), U_x(\xi_{NC-1}, \gamma_2, t), U_y(\xi_{NC-1}, \gamma_2, t), \\
&U_{xx}(\xi_{NC-1}, \gamma_2, t), U_{xy}(\xi_{NC-1}, \gamma_2, t), U_{yy}(\xi_{NC-1}, \gamma_2, t)), \\
&\cdots \\
U_t(\xi_{NC-1}, \gamma_{MC-1}, t) &= f(\xi_{NC-1}, \gamma_{MC-1}, t, U(\xi_{NC-1}, \gamma_{MC-1}, t), U_x(\xi_{NC-1}, \gamma_{MC-1}, t), \\
U_y(\xi_{NC-1}, \gamma_{MC-1}, t), &U_{xx}(\xi_{NC-1}, \gamma_{MC-1}, t), U_{xy}(\xi_{NC-1}, \gamma_{MC-1}, t), U_{yy}(\xi_{NC-1}, \gamma_{MC-1}, t)), \\
0 &= g_d(\xi_{NC-1}, t, U(\xi_{NC-1}, \gamma_{MC}, t), U_x(\xi_{NC-1}, \gamma_{MC}, t), U_y(\xi_{NC-1}, \gamma_{MC}, t)), \\
&- \\
0 &= g_b(\gamma_1, t, U(\xi_{NC}, \gamma_1, t), U_x(\xi_{NC}, \gamma_1, t), U_y(\xi_{NC}, \gamma_1, t)), \\
&\cdots \\
0 &= g_b(\gamma_{MC}, t, U(\xi_{NC}, \gamma_{MC}, t), U_x(\xi_{NC}, \gamma_{MC}, t), U_y(\xi_{NC}, \gamma_{MC}, t)).
\end{aligned}
\tag{4.8}
$$

$\underline{F}_i(t, \underline{W}(t)), \quad i = 2, \cdots, NC-1$, has the form:

$$\underline{F}_i(t, \underline{W}(t)) = \begin{bmatrix} g_c(\xi_i, t, U(\xi_i, \gamma_1, t), U_x(\xi_i, \gamma_1, t), U_y(\xi_i, \gamma_1, t)) \\ f(\xi_i, \gamma_2, t, U(\xi_i, \gamma_2, t), U_x(\xi_i, \gamma_2, t), U_y(\xi_i, \gamma_2, t), \\ U_{xx}(\xi_i, \gamma_2, t), U_{xy}(\xi_i, \gamma_2, t), U_{yy}(\xi_i, \gamma_{j2}, t)) \\ \vdots \\ f(\xi_i, \gamma_{MC-1}, t, U(\xi_i, \gamma_{MC-1}, t), U_x(\xi_i, \gamma_{MC-1}, t), U_y(\xi_i, \gamma_{MC-1}, t), \\ U_{xx}(\xi_i, \gamma_{MC-1}, t), U_{xy}(\xi_i, \gamma_{MC-1}, t), U_{yy}(\xi_i, \gamma_{MC-1}, t)) \\ g_d(\xi_i, t, U(\xi_i, \gamma_{MC}, t), U_x(\xi_i, \gamma_{MC}, t), U_y(\xi_i, \gamma_{MC}, t)) \end{bmatrix}, \tag{4.9}$$

$\underline{F}_1(t, \underline{W}(t))$ has the form:

$$\begin{bmatrix} g_a(\gamma_1, t, U(\xi_1, \gamma_1, t), U_x(\xi_1, \gamma_1, t), U_y(\xi_1, \gamma_1, t)), \\ \cdots \\ g_a(\gamma_{MC}, t, U(\xi_1, \gamma_{MC}, t), U_x(\xi_1, \gamma_{MC}, t), U_y(\xi_1, \gamma_{MC}, t)) \end{bmatrix}. \tag{4.10}$$

$F_{NC}(t, \underline{W}(t))$ has the form:

$$\begin{bmatrix} g_b(\gamma_1, t, U(\xi_{NC}, \gamma_1, t), U_x(\xi_{NC}, \gamma_1, t), U_y(\xi_{NC}, \gamma_1, t)), \\ \cdots \\ g_b(\gamma_{MC}, t, U(\xi_{NC}, \gamma_{MC}, t), U_x(\xi_{NC}, \gamma_{MC}, t), U_y(\xi_{NC}, \gamma_{MC}, t)) \end{bmatrix}. \tag{4.11}$$

The matrix A appearing in (4.7) has the form shown in Figure 4.2.

In Figure 4.2, each matrix D_i has $(p-1)$ block rows and $(p+1)$ block columns, where each block is an $MC \times MC$ matrix. The structure of each block inside D_i has the form shown in Figure 4.3. From Figure 4.3, we can see that each block inside D_i is an ABD matrix. The ith subblock, R_i is equal to $a_{k,l}S_i$, $i = 1, \cdots, N$, where the

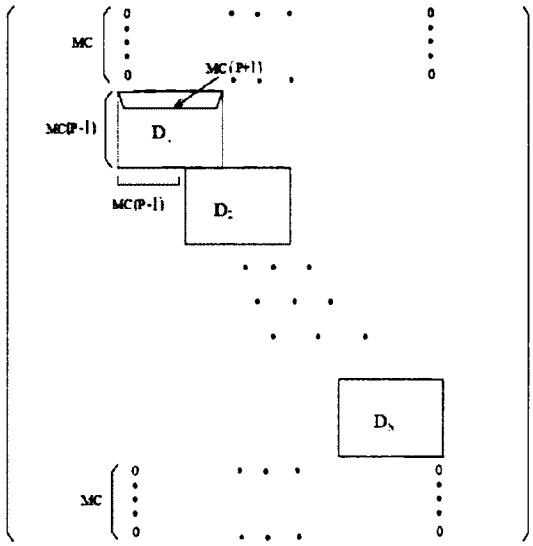

Figure 4.2: The structure of the ABD matrix A appearing in (4.7). Each block, D_i, is a matrix of size $MC(p-1)$ by $MC(p+1)$. The overlap between the D_i blocks is $2MC$, p is the degree of the polynomials used in the x domain, N is the number of subintervals in the x domain, MC is the dimension of the piecewise polynomial subspace in the y domain.

S_i has the form (2.7), with I_n replaced by 1. The matrix A, appearing in (4.7), can be written as

$$A = A_x \otimes A_y,$$

where A_x is the matrix shown in Figure 2.1, and A_y is a matrix with a similar structure to that of A_x, but for the y domain.

4.3 DASPK

The DAE system (4.7) is solved using DASPK, a high quality software package for the numerical solution of large-scale systems of DAEs. DASPK discretizes the DAE system, yielding a nonlinear system. This nonlinear system can be solved using, for example, Newton's method, and a large linear system will arise in each Newton

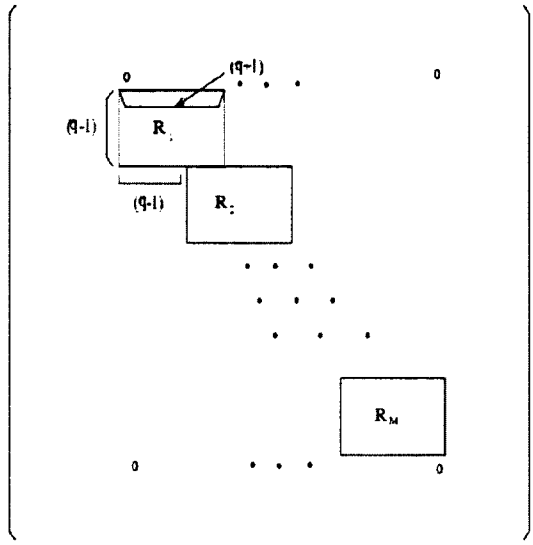

Figure 4.3: The structure of one of the block matrices inside D_i. Each block inside D_i is of size $MC \times MC$. The overlap between the R_j blocks is 2, q is the degree of the polynomials in the y domain, M is the number of subintervals in the y domain. Top and bottom are rows of zeros.

step. The linear systems arising during the numerical solution of the DAE system are too big to be solved with a direct method, so an iterative method must be used. The most significant difference between DASPK and DASSL is that DASPK uses the scaled preconditioned incomplete GMRES method (truncated GMRES) (an iterative solver for solving a linear system) [Saa81, SS86, Saa93] combined with an inexact Newton method [DES82] for the treatment of the linear and nonlinear systems that arise.

We now discuss the linear system solver in more detail. The large linear systems are solved iteratively rather than directly, as mentioned above. A banded preconditioner matrix [Saa90] is used during the solution of the linear system arising during each Newton iteration. The matrix preconditioner must try to approximate the Jacobian matrix (which has an ABD form) arising during the calculation. The preconditioner

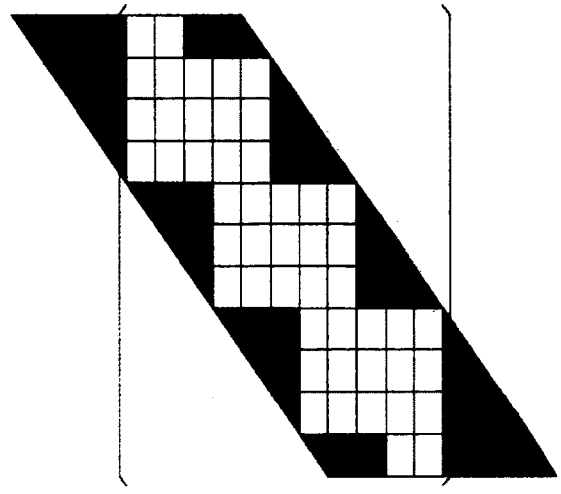

Figure 4.4: The banded preconditioner matrix employed by DASPK. The ABD structure is included with the band structure.

matrix is generated by DASPK. DASPK cannot generate a preconditioner with an ABD structure. The closest sparse matrix structure it has available for the preconditioner is a band matrix structure. Hence the preconditioner we employ has a band matrix structure (See Figure 4.4). DASPK tries to reuse this preconditioner matrix for as many time steps as possible, since the costs for building the preconditioners are high.

The linear system involving the preconditioner matrix is solved by an incomplete LU factorization [Saa03] based on routines from the SPARSKIT library [Saa90] (a FORTRAN 77 library for sparse matrix computations).

4.4 Efficient Block Matrix Algorithms

As mentioned earlier, R. D. Russell and W. Sun [RS97] suggested a fast algorithm based upon a matrix block eigenvalue decomposition for solving spline collocation matrices. We use a similar fast algorithm based upon LU decomposition with modified alternate row and column elimination with partial pivoting to take advantage of the

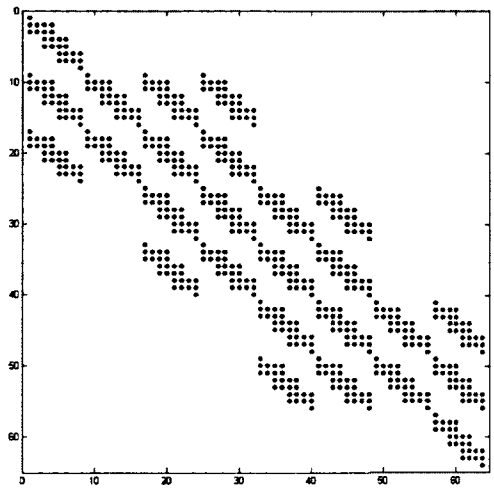

Figure 4.5: The ABD structure of the 2D B-spline projection Q in (4.12) (the degree of the polynomials in the x and y domains is 3 in each case, the number of subintervals is 3 in each case.)

structure of the matrices that arise. In addition, we also develop an efficient way to deal with the matrix multiplication for these structured matrices.

4.4.1 2D B-spline Projection

Using the tensor product approach, the B-spline approximation in 2D has the form (4.1), as mentioned earlier. The B-spline coefficients characterize the projection of the approximate solution onto the B-spline tensor product basis. We can rewrite this 2D B-spline projection in matrix form as:

$$\underline{U}(\underline{\xi}, \underline{\gamma}, t) = (M_x \otimes M_y)\underline{W}(t) = Q\underline{W}(t), \tag{4.12}$$

where $\underline{U}(\underline{\xi}, \underline{\gamma}, t)$ is the evaluation of $U(x, y, t)$ at all combinations of $\{\xi_i\}_{i=1}^{NC}$, $\{\gamma_i\}_{i=1}^{MC}$, as prescribed by the following definitions for $\underline{\xi}$ and $\underline{\gamma}$:

$$\underline{\xi} = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_{NC} \end{bmatrix} \otimes e_{MC}, \quad \underline{\gamma} = e_{NC} \otimes \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{MC} \end{bmatrix},$$

where e_{MC} is the vector of $1s$ of length MC, e_{NC} is the vector of $1s$ of length NC, M_x (for the x domain) is an $NC \times NC$ matrix having the form shown in Figure 2.2, and M_y is an $MC \times MC$ matrix having a similar structure to M_x, but for the y domain. Note $Q = M_x \otimes M_y$ has the form shown in Figure 4.5.

4.4.2 A Fast Block Matrix System Solution Algorithm

Referring to (4.12), we see that if we are given $\underline{W}(t)$, then we can obtain a vector representing an evaluation of $U(\underline{\xi}, \underline{\gamma}, t)$, the approximate solution at the collocation points, by multiplying $\underline{W}(t)$ by Q. Similarly, given $\underline{U}(\underline{\xi}, \underline{\gamma}, t)$, we can obtain $\underline{W}(t)$ by solving $Q\underline{W}(t) = \underline{U}(\underline{\xi}, \underline{\gamma}, t)$. To solve $Q\underline{W}(t) = \underline{U}(\underline{\xi}, \underline{\gamma}, t)$, we use COLROW to efficiently handle the ABD structure of the matrices M_x and M_y.

The algorithm we use is as follows. We assume that M_x and M_y are factored as:

$$M_x = L_x U_x, \quad M_y = L_y U_y,$$

where L_x, L_y are lower triangular matrices, and U_x, U_y are upper triangular matrices. (The factorization performed by COLROW is actually more complicated than what we have stated above - see [DFK83b]. It also includes row and column permutation matrices associated with alternating row and column pivoting and is based on row and column elimination. However in order to simplify the presentation we do not include these components of the factorization here. Also modifications to some of the COLROW routines were required in order to separate the back solve steps associated with upper triangular matrices from the forward solve steps associated with lower

triangular matrices, as required in the algorithm given below. As well, L_x, L_y are actually lower triangular ABD matrices and U_x, U_y are actually upper triangular ABD matrices. This means that most of the lower triangle of L_x, L_y is zero and that most of the upper triangle of U_x, U_y is zero. However, for simplicity, in the discussion below we will simply represent L_x, L_y as lower triangular matrices and U_x, U_y are upper triangular matrices.) Note that L_x, U_x are $NC \times NC$ matrices and L_y, U_y are $MC \times MC$ matrices.

Let us simplify the notation by writing (4.12) as

$$(M_x \otimes M_y)w = b, \tag{4.13}$$

where $w = \underline{W}(t)$ and $b = \underline{U}(\underline{\xi}, \underline{\gamma}, t)$. The above system can then be rewritten as

$$((L_x U_x) \otimes (L_y U_y))w = b.$$

Based on a property of the Kronecker product, we can rewrite the above system as

$$((L_x \otimes L_y)(U_x \otimes U_y))w = b.$$

This system can then be solved in 4 steps:

(1)Solve $(L_x \otimes I_{MC})\overline{v} = b$ for $\overline{v}$.

(2)Then solve $(I_{NC} \otimes L_y)v = \overline{v}$ for v.

(3)Then solve $(U_x \otimes I_{MC})\overline{w} = v$ for $\overline{w}$.

(4)Then solve $(I_{NC} \otimes U_y)w = \overline{w}$ for w,

where I_{MC} is the $MC \times MC$ identity matrix and I_{NC} is the $NC \times NC$ identity matrix. (Recall that M_x is a matrix of dimension $NC \times NC$ and M_y is a matrix of dimension $MC \times MC$.)

Note that steps (1) and (2) solve the system

$$(L_x \otimes L_y)v = b$$

for v.

To see this, substitute from step (2) into step (1); we get

$$(L_x \otimes I_{MC})(I_{NC} \otimes L_y)v = b.$$

Then again using the property of the Kronecker product, we can rewrite this equation as

$$(L_x I_{NC} \otimes I_{MC} L_y)v = b,$$

and this reduces to

$$(L_x \otimes L_y)v = b.$$

A similar argument can be used to show that steps (3) and (4) solve the linear system

$$(U_x \otimes U_y)w = v.$$

Let

$$L_x = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \cdots & & \ddots & \\ l_{NC,1} & l_{NC,2} & \cdots & l_{NC,NC} \end{bmatrix}.$$

Considering step (1) in block form, we have:

$$\begin{bmatrix} l_{11}I_{MC} & & & \\ l_{21}I_{MC} & l_{22}I_{MC} & & \\ \cdots & & \ddots & \\ l_{NC,1}I_{MC} & l_{NC,2}I_{MC} & \cdots & l_{NC,NC}I_{MC} \end{bmatrix} \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \\ \underline{v}_{NC} \end{bmatrix} = \begin{bmatrix} \underline{b}_1 \\ \underline{b}_2 \\ \vdots \\ \underline{b}_{NC} \end{bmatrix},$$

where $\overline{v}_j = \begin{bmatrix} \overline{v}_{(j-1)MC+1} \\ \overline{v}_{(j-1)MC+2} \\ \vdots \\ \overline{v}_{jMC} \end{bmatrix}$, $\underline{b}_j = \begin{bmatrix} \underline{b}_{(j-1)MC+1} \\ \underline{b}_{(j-1)MC+2} \\ \vdots \\ \underline{b}_{jMC} \end{bmatrix}$.

Since the above matrix is a block lower triangular matrix, we can solve the first left upper block, $l_{11} I_{MC} \cdot \overline{v}_1 = \underline{b}_1$ for $\overline{v}_1$, then substitute $\overline{v}_1$ into the second block row and solve for $\overline{v}_2$ and so on.

Step (2) in block form is:

$$\begin{bmatrix} L_y & & & \\ & L_y & & \\ & & \ddots & \\ & & & L_y \end{bmatrix} \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \\ \underline{v}_{NC} \end{bmatrix} = \begin{bmatrix} \overline{\underline{v}}_1 \\ \overline{\underline{v}}_2 \\ \vdots \\ \overline{\underline{v}}_{NC} \end{bmatrix}, \text{ where } \underline{v}_j = \begin{bmatrix} v_{(j-1)MC+1} \\ v_{(j-1)MC+2} \\ \vdots \\ v_{jMC} \end{bmatrix}.$$

This system can be separated into NC subsystems of the form $L_y \cdot \underline{v}_i = \overline{\underline{v}}_i, i = 1, \cdots, NC$.

In steps (3) and (4), based on the same idea, we solve upper triangular systems.

Let

$$U_x = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1,NC} \\ & c_{22} & \ddots & \vdots \\ & & \ddots & c_{(NC-1)(NC-1)} \\ & & & c_{NC,NC} \end{bmatrix}.$$

Step (3) in block form is (the components of U_x are called c_{ij}):

$$\begin{bmatrix} c_{11} I_{MC} & c_{12} I_{MC} & \cdots & c_{1,NC} I_{MC} \\ & c_{22} I_{MC} & \ddots & \vdots \\ & & \ddots & c_{(NC-1)(NC-1)} I_{MC} \\ & & & c_{NC,NC} I_{MC} \end{bmatrix} \begin{bmatrix} \overline{\underline{w}}_1 \\ \overline{\underline{w}}_2 \\ \vdots \\ \overline{\underline{w}}_{NC} \end{bmatrix} = \begin{bmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \\ \underline{v}_{NC} \end{bmatrix},$$

where $\underline{\overline{w}}_j = \begin{bmatrix} \overline{w}_{(j-1)MC+1} \\ \overline{w}_{(j-1)MC+2} \\ \vdots \\ \overline{w}_{jMC} \end{bmatrix}$, $\underline{v}_j = \begin{bmatrix} v_{(j-1)MC+1} \\ v_{(j-1)MC+2} \\ \vdots \\ v_{jMC} \end{bmatrix}$.

We first solve the right bottom block, $c_{NC,NC} I_{MC} \cdot \underline{\overline{w}}_{NC} = \underline{v}_{NC}$, for $\underline{\overline{w}}_{NC}$, and then substitute $\underline{\overline{w}}_{NC}$ back into the second last block equation, then solve for $\underline{\overline{w}}_{NC-1}$ and so on.

Step (4) in block form is:

$$\begin{bmatrix} U_y & & & \\ & U_y & & \\ & & \ddots & \\ & & & U_y \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{NC} \end{bmatrix} = \begin{bmatrix} \underline{\overline{w}}_1 \\ \underline{\overline{w}}_2 \\ \vdots \\ \underline{\overline{w}}_{NC} \end{bmatrix}, \text{ where } w_j = \begin{bmatrix} w_{(j-1)MC+1} \\ w_{(j-1)MC+2} \\ \vdots \\ w_{jMC} \end{bmatrix}.$$

This system can be separated into NC subsystems of the form $U_y \cdot w_i = \underline{\overline{w}}_i$, $i = 1, \cdots, NC$.

Thus the above algorithm implies that instead of solving a large $(MC \cdot NC) \times (MC \cdot NC)$ linear system (4.2), we only need to solve a sequence of $MC \times MC$ and $NC \times NC$ linear systems. This allows us to save a large amount of storage and computation.

We now take a closer look at the computational costs for this four step algorithm.

Based on the analysis given in [DFK83b] and referring to Figure 2.2, it follows that the cost of factoring an $NC \times NC$ ABD system with coefficient matrix M_x is $O(Np^3)$, while the solution step cost is $O(Np^2)$. Both M_x and M_y have to be factored and (assuming $N = M$ and $p = q$) these factorization costs will be $O(Np^3)$ for both matrices. They are factored only once so these are the total costs for the factorizations.

In step (1) of the four step algorithm, we have to perform one lower triangular solve but we are considering a block lower triangular matrix and the blocks are diagonal matrices of size $MC \times MC$. The cost for a lower triangular solve is $O(Np^2)$ but

since we are working with elements that are diagonal matrices of size $MC \times MC$, we have to multiply this cost by $MC = NC = O(Np)$. Thus the cost for step (1) is $O((Np)(Np^2)) = O(N^2p^3)$.

In step (2), we have to perform NC lower triangular solves, each of which costs $O(Np^2)$. Since $NC = O(Np)$, the total cost for step (2) is $O((Np)(Np^2)) = O(N^2p^3)$.

Similar arguments show that steps (3) and (4) also cost $O(N^2p^3)$. Thus the total cost for the four step algorithm is also $O(N^2p^3)$. This dominates the cost of the factorizations which is $O(Np^3)$ and thus the total cost for the solution of (4.13) is $O(N^2p^3)$.

If we were to solve the ABD system (4.13) simply by treating the coefficient matrix as an ABD matrix with blocks of size $O(p(MC \times MC)) = O(p(Np \times Np)) = O(Np^2 \times Np^2))$, the cost of factoring this ABD matrix with blocks of this size would be $O(N(Np^2)^3)) = O(N^4p^6)$. Thus substantial savings are achieved by using the four step algorithm even for modestly sized values of n and p.

4.4.3 Fast Block Matrix Multiplication

At certain points in the algorithm, we know the B-spline coefficients, $\underline{W}(t)$, and we need to evaluate the approximate solution at the collocation points $\underline{U}(\underline{\xi}, \underline{\gamma}, t)$. This corresponds to the computation, $(M_x \otimes M_y)w = b$, where we know w, M_x and M_y and we want to compute b, using a matrix multiplication.

Recall that both M_x and M_y have an ABD form. Thus $Q = M_x \otimes M_y$ has a block ABD form, and each subblock is also an ABD matrix. Recall that the degree of the B-splines basis polynomials in x and y are p and q respectively. The i and $(i+1)st$ subblocks of Q, related to the $N(p-1)$ collocation points in the x domain, have the form:

$$
\begin{array}{ccccccc}
\ddots & \cdots & & & & & \\
\cdots & c_{i,j}M_y & \cdots & c_{i,j+p-1}M_y & c_{i,j+p}M_y & & \\
\cdots & \cdots & \cdots & \cdots & \cdots & & \\
\cdots & c_{i+p-2,j}M_y & \cdots & c_{i+p-2,j+p-1}M_y & c_{i+p-2,j+p}M_y & & \\
 & & & c_{i+p-1,j+p-1}M_y & c_{i+p-1,j+p}M_y & \cdots & \\
 & & & \cdots & \cdots & \cdots & \\
 & & & c_{i+2p-3,j+p-1}M_y & c_{i+2p-3,j+p}M_y & \cdots & \\
 & & & & \cdots & \ddots &
\end{array}
, \tag{4.14}
$$

where $c_{i,j}$ is the first non zero element on the ith line of M_x. We note that the block column $c_{i,j}M_y, \cdots, c_{i,j+p-2}M_y$ appears in a column of a larger block that does not overlap with the block below it; we refer to this as a non-overlap block. A close look inside the $c_{i,j}M_y$ block reveals that this block has an ABD form:

$$
\begin{array}{ccccccc}
c_{i,j}d_{1,1} & & & & & & \\
c_{i,j}d_{2,1} & \cdots & c_{i,j}d_{2,1+q-1} & c_{i,j}d_{2,1+q} & & & \\
\cdots & \cdots & \cdots & \cdots & & & \\
c_{i,j}d_{2+q-2,1} & \cdots & c_{i,j}d_{2+q-2,1+q-1} & c_{i,j}d_{2+q-2,1+q} & & & \\
 & & c_{i,j}d_{2+q-1,1+q-1} & c_{i,j}d_{2+q-1,1+q} & \cdots & c_{i,j}d_{2+q-1,1+2q-1} & \\
 & & \cdots & \cdots & \cdots & \cdots & \\
 & & c_{i,j}d_{2+2q-3,1+q-1} & c_{i,j}d_{2+2q-3,1+q} & \cdots & c_{i,j}d_{2+2q-3,1+2q-1} & \\
 & & & & & & \ddots
\end{array}
, \tag{4.15}
$$

where $d_{i,j}$ are non zero elements in M_y.

We next return to the matrix form (4.14) and consider multiplication by w:

$$\begin{bmatrix} \ddots & \cdots & & & & \\ \cdots & c_{i,j}M_y & \cdots & c_{i,j+p-1}M_y & c_{i,j+p}M_y & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ \cdots & c_{i+p-2,j}M_y & \cdots & c_{i+p-2,j+p-1}M_y & c_{i+p-2,j+p}M_y & \\ & & & c_{i+p-1,j+p-1}M_y & c_{i+p-1,j+p}M_y & \cdots \\ & & & \cdots & \cdots & \cdots \\ & & & c_{i+2p-3,j+p-1}M_y & c_{i+2p-3,j+p}M_y & \cdots \\ & & & & \cdots & \ddots \end{bmatrix} \cdot \begin{bmatrix} \vdots \\ w_k \\ \vdots \\ w_{k+q-1} \\ w_{k+q} \\ \vdots \\ \\ \end{bmatrix}.$$

Let $\underline{w} = \begin{bmatrix} w_k \\ \vdots \\ w_{k+q-1} \\ w_{k+q} \end{bmatrix}$, and note that this vector is the part of the vector w which is to be multiplied by the subblocks, $c_{i,j}M_y, \cdots, c_{i+p-2,j}M_y$. The key idea is to avoid duplicate multiplications involving $M_y \cdot \underline{w}$.

Consider the 2nd row of the $c_{i,j}M_y$ block above multiplied by $\underline{w}$. Using (4.15), we obtain the expression

$$c_{i,j}d_{2,1}w_k + \cdots + c_{i,j}d_{2,1+q}w_{k+q} = c_{i,j}(d_{2,1}w_k + \cdots + d_{2,1+q}w_{k+q}).$$

Referring to the block column containing $c_{i,j}M_y, \cdots, c_{i+p-2,j}M_y$, for the 2nd row of each subblock, we see a similar calculation:

$$\begin{gathered} c_{i,j}(d_{2,1}w_k + \cdots + d_{2,q}w_{k+q-1} + d_{2,1+q}w_{k+q}) \\ \cdots \\ c_{i+p-2,j}(d_{2,1}w_k + \cdots + d_{2,q}w_{k+q-1} + d_{2,1+q}w_{k+q}) \end{gathered}.$$

Thus we can compute $d_{2,1}w_k + \cdots + d_{2,q}w_{k+q-1} + d_{2,1+q}w_{k+q}$ once and use it $p-1$ times.

For the overlap subblocks, for example:

$$c_{i,j+p-1}M_y, \cdots, c_{i+p-2,j+p-1}M_y,\ c_{i+p-1,j+p-1}M_y, \cdots, c_{i+2p-3,j+p-1}M_y,$$

we can apply the same approach described above; the only difference is that the calculation can be applied $2(p-1)$ instead of $(p-1)$ times.

4.5 Numerical experiments

In this section, we will consider three different time-dependent 2D parabolic PDE problems. We use GNU Fortran77 (GCC) 4.4.3 under Ubuntu (Kernel Linux 2.6.32-40-server) running on an 7 Intel (R) Xeon (R) CPUs (E5420 @ 2.50GHz) system for which the accessible memory is 2GB.

Recall that the degrees of the piecewise polynomial for the x-axis and the y-axis are p and q respectively. In our numerical experiments, we use $p = q$.

The following notation will be used in representing the numerical results.

KCOL: the number of collocation points per subinterval , KCOL $= p - 1$;

NINT: the number of subintervals (NINT $= N = M$);

ATOL: the absolute tolerance (used by DASPK);

RTOL: the relative tolerance (used by DASPK);

TOL: the tolerance for the nonlinear solver in DASPK;

TOUT: the output time (t_{out});

GE: the true error at a set of sample points equally distributed over the problem domain at time TOUT.

In order to obtain the convergence results, we employed a number of different choices for TOL, ATOL, and RTOL. These were chosen by trial and error so that the temporal error was smaller than the observed spatial error. We ran a number of experiments in which we gradually reduced the tolerances provided to DASPK

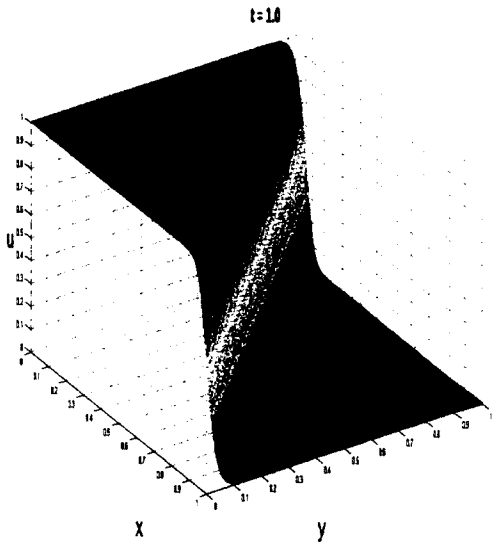

Figure 4.6: Approximate solution for Problem 1, $\varepsilon = 0.01$, KCOL=3, NINT=64.

until the spatial error was observed to be constant from one experiment to the next, indicating that the temporal error was no longer contributing significantly to the overall error.

Problem 1

We consider the 2D Burgers' equation [VB06],

$$\frac{\partial u}{\partial t} = \xi \frac{\partial^2 u}{\partial x^2} + \xi \frac{\partial^2 u}{\partial y^2} - u\frac{\partial u}{\partial x} - u\frac{\partial u}{\partial y}. \tag{4.16}$$

The problem domain is $(x, y) \in (0, 1) \times (0, 1)$, $t > 0$; the boundary and initial conditions are chosen so that the true solution is

$$u(x, y, t) = \frac{1}{1 + e^{\frac{x+y-t}{2\xi}}}.$$

We set $\xi = 0.01$. The numerical solution is plotted in Figure 4.6 (KCOL = 3, NINT

Table 4.1: Observed GE, GE ratios, and corresponding approximate convergence rates for Problem 1.

KCOL	NINT	GE	ratio	rate
3	8	9.95×10^{-2}	-	-
3	16	3.69×10^{-3}	26.96	4.75
3	32	1.16×10^{-4}	31.81	4.99
3	64	4.22×10^{-6}	27.49	4.78
4	16	5.56×10^{-4}	-	-
4	32	1.05×10^{-5}	52.95	5.73
4	64	1.90×10^{-7}	55.33	5.79
5	10	2.40×10^{-3}	-	-
5	20	2.06×10^{-5}	116.50	6.86
5	40	1.61×10^{-6}	130.38	7.03

$= 64$).

Problem 2

Another example is the problem [Wan95]

$$L_1 = (x^2+1)\frac{\partial^2}{\partial x^2} + x,$$

$$L_2 = (y^2+1)\frac{\partial^2}{\partial y^2} + y\frac{\partial}{\partial y} + y,$$

$$\frac{\partial u}{\partial t} = (L_1 + L_2)u + f(x,y,t),$$

$$(x,y,t) \in \Omega \times (0,1], \quad \Omega = (0,1) \times (0,1).$$

The boundary and initial conditions and $f(x,y,t)$ are chosen so that the true solution is

$$u = (e^{-t}+1)\sin(\pi x)\sin(\pi y).$$

The numerical solution is plotted in Figure 4.7 (KCOL = 3, NINT = 16).

Problem 3

For this problem, we consider the equation [Wan95]

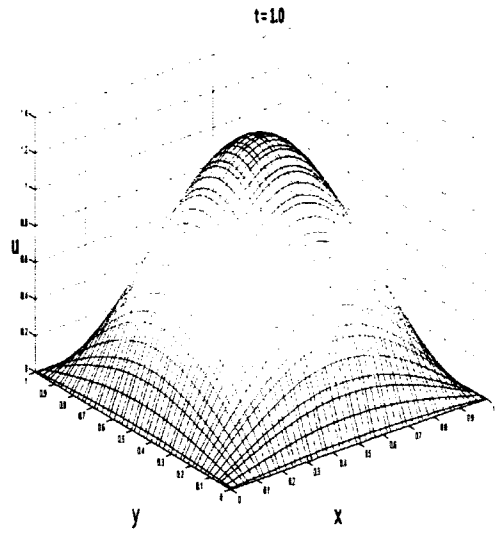

Figure 4.7: Approximate solution of Problem 2. KCOL=3, NINT=16.

$$\frac{\partial u}{\partial t} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u + f(x, y, t),$$

on $[0, 1] \times [0, 1]$.

The boundary and initial conditions and $f(x, y, t)$ are chosen so that the true solution is

$$u(x, y, t) = (e^{-t} + 1)(x^m + y^m + xy^{m-1} + 1).$$

We set $m = 6$. The numerical solution is plotted in Figure 4.8 (KCOL = 5, NINT = 20).

Problem 4

This example is a slight modification of Problem 2, in which we have added a mixed derivative operator L_3

$$L_1 = (x^2 + 1)\frac{\partial^2}{\partial x^2} + x,$$

$$L_2 = (y^2 + 1)\frac{\partial^2}{\partial y^2} + y\frac{\partial}{\partial y} + y,$$

Table 4.2: Observed GE, GE ratios, and corresponding approximate convergence rates for Problem 2.

KCOL	NINT	GE	ratio	rate
3	4	1.50×10^{-5}	-	-
3	8	4.51×10^{-7}	33.22	5.05
3	16	1.36×10^{-8}	33.23	5.05
3	32	4.14×10^{-10}	32.78	5.03
4	4	5.23×10^{-7}	-	-
4	8	8.58×10^{-9}	60.88	5.93
4	16	1.36×10^{-10}	63.26	5.98
4	32	2.13×10^{-12}	63.63	5.99
5	10	3.06×10^{-11}	-	-
5	20	1.97×10^{-13}	155.33	7.28

Table 4.3: Observed GE, GE ratios, and corresponding approximate convergence rates for Problem 3.

KCOL	NINT	GE	ratio	rate
3	32	1.99×10^{-9}	-	-
3	64	6.38×10^{-11}	31.19	4.96
4	20	1.67×10^{-10}	-	-
4	40	2.98×10^{-12}	56.04	5.81
5	15	1.71×10^{-11}	-	-
5	30	1.33×10^{-13}	128.57	7.01

$$L_3 = \frac{\partial^2}{\partial x \partial y},$$

$$\frac{\partial u}{\partial t} = (L_1 + L_2 + L_3)u + f(x, y, t),$$

$$(x, y, t) \in \Omega \times (0, 1], \quad \Omega = (0, 1) \times (0, 1).$$

The boundary and initial conditions and $f(x, y, t)$ are chosen so that the true solution is

$$u = (e^{-t} + 1)\sin(\pi x)\sin(\pi y).$$

The numerical solution is plotted in Figure 4.9 (KCOL = 3, NINT = 16).

Each of the four test problems has a known exact solution and thus it is possible to estimate the maximum GE of a given numerical solution. We next compute the GE for

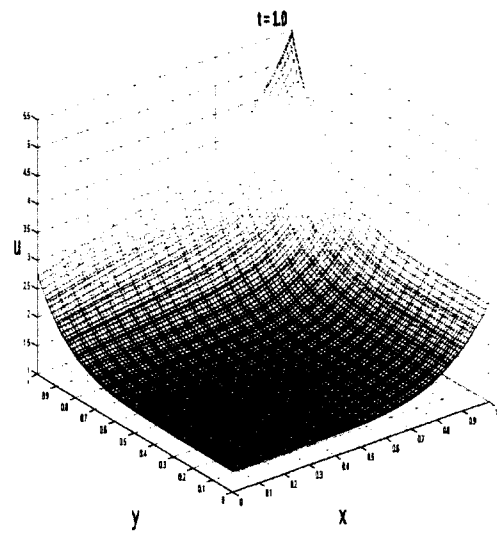

Figure 4.8: Approximate solution of Problem 3,m=6, KCOL=5, NINT=20.

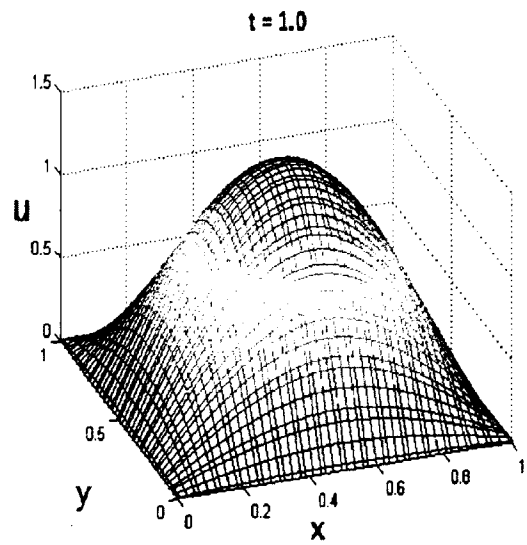

Figure 4.9: Approximate solution of Problem 4. KCOL=3, NINT=16.

Table 4.4: Observed GE, GE ratios, and corresponding approximate convergence rates for Problem 4.

KCOL	NINT	GE	ratio	rate
3	4	1.71×10^{-5}	-	-
3	8	4.85×10^{-7}	35.24	5.14
3	16	1.41×10^{-8}	34.38	5.10
3	32	4.22×10^{-10}	33.40	5.06
4	4	5.38×10^{-7}	-	-
4	8	8.65×10^{-9}	62.20	5.96
4	16	1.36×10^{-10}	63.43	5.99
4	32	2.39×10^{-12}	57.00	5.83

a number of collocation solutions computed by BACOL2D, for a variety of KCOL and NINT values, for the four test problems. By fixing KCOL and considering a sequence of meshes obtained by doubling the NINT value we can compare the observed GE and by considering the ratio of the GE of the collocation solutions obtained using this sequence of meshes, we can experimentally determine the spatial order of convergence of the collocation solution as a function of KCOL.

The convergence results for the corresponding 1D case are known from the literature, e.g., [CP76] (the rate of convergence is KCOL + 2.) Since we are using a tensor product framework, we anticipate that the corresponding result will also hold for the 2D case.

In Tables 4.1-4.4, we present, for the collocation solutions we compute at $t = 1$, the observed GE, GE ratios, and corresponding approximate convergence rates for Problems 1-4. From the four tables we observe that the expected rates of convergence (based on the known results for the 1D case) are indeed observed in the 2D case (i.e., the GE is order KCOL + 2.)

Chapter 5

BACOL2D

This chapter first describes the subroutines included in the BACOL2D software package. Then we discuss a sample program and present the corresponding user supplied subroutines. Finally, we give the structure of the BACOL2D software. BACOL2D was developed in Fortran 77 using the GNU Fortran77 (GCC) 4.4.3 compiler.

5.1 Description of the Software

This section describes the components of the BACOL2D software package.

BACOL2D. This subroutine is the main component of the software package. It performs initialization tasks such as assigning values to parameters and defining the length of the common storage arrays. It also checks the length of the work arrays which serve as temporary storage. It also calls MESHSQ and COLPNT to complete the initialization (see below for an explanation of MESHSQ and COLPNT). BACOL2D will make repeated calls to the time-integrator DDASPK to take time steps. After reaching the output time TOUT, BACOL2D will have the B-spline coefficients which can be used to calculate the values of $U(x, y, TOUT)$, where $a \leq x \leq b$, $c \leq y \leq d$. Both the absolute tolerance, ATOL, and the relative tolerance, RTOL, are set by user

as the tolerances for DASPK.

KXYV. This subroutine takes as input the B-spline coefficients, and then calculates the approximate solution at the current time, for given x and y.

COLPNT. This subroutine calculates the collocation point sequence.

MESHSQ. This subroutine calculates the spatial mesh size sequence, as well as generating the points and weights for the Gaussian quadrature rule.

BSCOE. This subroutine is called by main. It evaluates the B-spline basis (and its first and second derivatives) and stores them in ABD form.

KRONXY1. This subroutine calculates the approximate solution at the current time and at any given spatial points, x and y.

INVKXY. This subroutine will call CRSLVELB, CRSLVEL, CRSLVEUB and CRSLVEU to solve the linear system (4.13) using the algorithm described in Section 4.4.2, and they are used in steps (1)-(4) separately.

Now we mention some subroutines which are included in BACOL2D but were developed by others.

DASPK. This software package, developed by P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, was obtained from a modification of DASSL. The most significant difference between the two solvers is that DASPK uses a sparse linear system solver combined with an inexact Newton method. In Chapter 4, we described DASPK.

INTREV, BSPLVD, BSPLVN. [dB77, dB78] These routines are part of the B-spline package. We use these subroutines to generate the values of the B-spline basis functions and their derivatives.

GAULEG. This subroutine was developed by P. Keast [WKM04a]. For the Gauss-Legendre quadrature rule, it calculates the points and weights in the interval $[0, 1]$ or $[-1, 1]$.

5.2 User Supplied Subroutines

The user needs to define the PDE by providing subroutines to give various kinds of information. For given input values of x, y and t, and corresponding input values for U, U_x, U_y, U_{xy}, U_{xx}, U_{yy}, the user needs to provide the following subroutines:

F. This subroutine computes the value

$$f(x,y,t,U(x,y,t),U_x(x,y,t),U_y(x,y,t),U_{xx}(x,y,t),U_{xy}(x,y,t),U_{yy}(x,y,t)),$$

representing the right-hand side of the problem PDE.

UINIT. This subroutine is used to provide the initial values $U(x,y,t_0)$ for any given x, y.

RES. This subroutine computes the residual of the DAE system (4.7) to be solved by DASPK(The residual of (4.7) is $A(\underline{W}(t))_t - \underline{F}(t,\underline{W}(t))$.). In this subroutine, the user needs to describe the boundary conditions.

5.3 Sample Program

We use the Problem 1: the 2D Burgers' equation, to show that these user-supplied subroutines are usually easily constructed. (See Appendix B)

5.4 Structure of BACOL2D

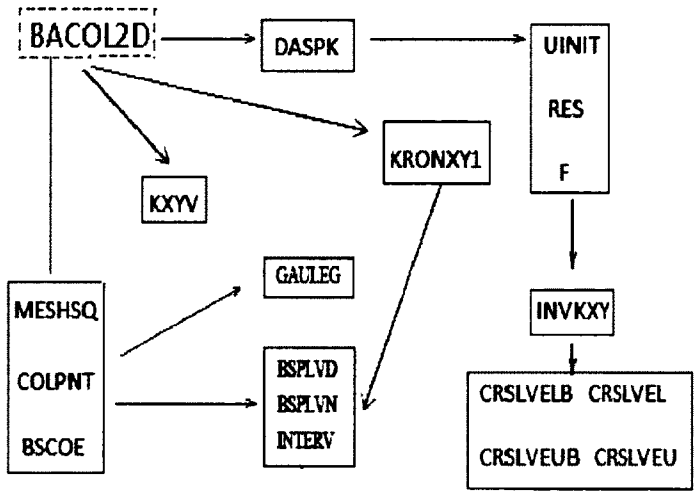

Figure 5.1: Software structure of BACOL2D

Chapter 6

Interpolation-based Error Estimation

6.1 1D interpolation-based error estimation

As we discussed earlier, in addition to the solution approximation, $U(x,t)$, BACOL computes a second global collocation solution, $\overline{U}(x,t)$, with a degree $p+1$ piecewise polynomial on the same spatial mesh at the same time t. The difference between the higher order solution and the lower order solution gives an approximate spatial error. Based on the error estimation, BACOL can construct a new spatial mesh if necessary.

In this chapter, we first describe two interpolation based approaches, recently developed for use in BACOL, that remove the need to compute a second collocation solution in order to obtain an error estimate. We then discuss some preliminary work that (experimentally) establishes the existence of certain points in the 2D spatial domain where the 2D collocation solutions exhibit superconvergence, i.e., "extra" accuracy, that may be useful for generating a low cost spatial error estimate.

6.1.1 1D Superconvergent Interpolant (SCI)-based Spatial Error Estimation

This approach is described in more detail in [ASM09].

Instead of computing the second higher order solution, $\overline{U}(x,t)$, the new version of BACOL (called BACOLI) uses a local interpolant to replace $\overline{U}(x,t)$ in the error estimation. That is, only the lower order solution approximation, $U(x,t)$ of degree p, is computed and propagated forward in time for every time step. This cuts the computational work approximately in half [ASM09].

The collocation solution at certain points such as mesh points is superconvergent, because Gaussian collocation is employed as the spatial discretization scheme [CP76, DD74]. Also, it turns out that there are a number of special points within each subinterval that are superconvergent. The existence of these points within the spatial domain of a 1D PDE was suggested by some theoretical results for BVODEs.

Applying a $(p-1)-$point Gaussian collocation method to a standard BVODE, based on appropriate assumptions, the following results regarding the collocation error are known [AMR95, ASM09]. (Here u is the exact solution, U is the collocation solution, x_i is the ith mesh point, $h = \max\limits_{1\le i\le N}\{h_i\}$, $h_i = x_i - x_{i-1}$, N is the number of subintervals.)

(1) At the mesh points:

$$| u(x_i) - U(x_i) | = O(h^{2(p-1)}), \quad | u'(x_i) - U'(x_i) | = O(h^{2(p-1)}), \quad i = 1, \cdots, N.$$

(2) At the nonmesh points:

$$u^{(j)}(x) - U^{(j)}(x) = u^{(p+1)}(x_{i-1}) P_p^{(j)}\left(\frac{x - x_{i-1}}{h_i}\right) h_i^{p+1-j} + O(h_i^{p+2-j}) + O(h^{2(p-1)}) = O(h^{p+1-j}), \tag{6.1}$$

where $x \in (x_{i-1}, x_i)$, $i = 1, \cdots, N$, $j = 0, \ldots, p$, and

$$P_p(\xi) = \frac{1}{(p-1)!} \int_0^{\xi} (t - \xi) \prod_{r=1}^{p-1} (t - \rho_r) dt, \tag{6.2}$$

and $u^{(j)}(x)$ is the jth derivative of $u(x)$ and $U^{(j)}(x)$ is the jth derivative of $U(x)$. Thus at the roots of the polynomial, $P_p(\xi)$ $(p \geqslant 2)$, the collocation solution will have a higher order of convergence.

The existence of these superconvergent points within each subinterval of the spatial domain of a 1D PDE was experimentally verified in [ASMK11].

BACOLI constructs a superconvergent interpolant for each subinterval. The interpolant is a C^1-continuous, piecewise polynomial that uses the superconvergent solution and derivative values at the endpoints of the subinterval, the superconvergent solution values within the current subinterval, and the two closest superconvergent solution values from the adjacent subintervals. (The interpolant uses $p + 3$ interpolant values so that the interpolant error will be smaller than the data error of the interpolated collocation solution values.)

6.1.2 1D Lower Order Interpolant (LOI)-based Spatial Error Estimation

This approach is described in detail in [ASMP12].

In contrast to the SCI, the LOI computes only the higher order collocation solution $\overline{U}(x,t)$, and uses an interpolant to approximate the lower order solution. That is, only one collocation solution approximation, $\overline{U}(x,t)$, of degree $p+1$, is computed and propagated forward in time. The basic idea is to develop an interpolant for which the leading order term in the interpolation error is equal to the leading order term in the collocation error of the lower order collocation solution[Moo01].

This requires the LOI to interpolate the solution and derivative values at the

endpoints and the points corresponding to the superconvergent solution values of the lower order solution internal to each subinterval. In this case, fewer interpolation points are used and the interpolation error dominates the data error.

6.2 Extension to 2D

It may be possible to extend the 1D interpolation based error estimation to the 2D case. The SCI approach will require the existence of superconvergent points within the 2D domain. In this section we consider numerical experiments that give us some evidence of the existence of such points.

In order to compute the superconvergent points in 2D, we use (6.1), (6.2) to suggest the locations of the superconvergent points in x and y domain separately, and use these values as coordinates to look for superconvergent points in 2D.

We consider the test problem,

$$L_1 = (x^2+1)\frac{\partial^2}{\partial x^2} + x, \quad L_2 = (y^2+1)\frac{\partial^2}{\partial y^2} + y\frac{\partial}{\partial y} + y,$$

$$\frac{\partial u}{\partial t} = (L_1 + L_2)u + f(x,y,t), \quad (x,y,t) \in \Omega \times (0,1], \Omega = (0,1) \times (0,1),$$

with the boundary and initial conditions chosen so that the true solution is

$$u = (e^{-t}+1)\sin(\pi x)\sin(\pi y).$$

We calculate collocation solution errors across the entire problem domain (GE), at the mesh points (ME), and at the superconvergent points (SE) internal to each rectangle of the spatial domain. We also calculate the error for the spatial derivatives of the collocation solution at the mesh points.

In the following tables,

- KCOL: the number of collocation points in each subinterval in both the x and y domains (KCOL=$p-1$=$q-1$).

- NINT: the number of subintervals in both x and y domain (NINT=M=N).
- The time integration is from 0 to 1.
- We consider error ratios as NINT is doubled.
- We set ATOL=RTOL= 2.0E-13 for DASPK, and TOL= 5.0E-14 for the nonlinear system solver in DASPK.

In the 1D case the expected GE rate=KCOL+2, ME rate=2KCOL and SE rate=KCOL+3 [ASM09, CP76, DD74], so in the 2D case we expect the same convergence rates, because we are using a tensor product formulation. We also expect the convergence rates for the spatial derivative errors at the mesh points to be the same as in the 1D case, 2KCOL.

Table 6.1: Expected GE rate = KCOL+2=5, ME rate = 2KCOL =6 and SE rate = KCOL+3=6 (KCOL=3)

NINT	GE	ratio	rate	SE	ratio	rate
8	4.51×10^{-7}	-	-	9.86×10^{-8}	-	-
16	1.36×10^{-8}	33.23	5.05	1.57×10^{-9}	62.89	5.97
32	4.14×10^{-10}	32.78	5.03	2.50×10^{-11}	62.64	5.97
	ME	ratio	rate	ME(U_x)	ratio	rate
	9.83×10^{-9}	-	-	1.28×10^{-7}	-	-
	1.61×10^{-10}	61.22	5.94	2.05×10^{-9}	62.50	5.97
	2.45×10^{-12}	65.61	6.04	3.10×10^{-11}	65.91	6.04
	ME(U_y)	ratio	rate			
	1.49×10^{-7}	-	-			
	2.36×10^{-9}	63.09	5.96			
	3.57×10^{-11}	66.08	6.05			

From Tables 6.1, 6.2, we can observe the existence of superconvergence at the internal superconvergent points (SE) and at the mesh points (ME), compared to the GE and associated observed rate of convergence at an arbitrary point in the spatial domain. The presence of such superconvergence values may be useful for error

Table 6.2: Expected GE rate = KCOL+2=6, ME rate = 2KCOL =8 and SE rate = KCOL+3=7 (KCOL=4)

NINT	GE	ratio	rate	SE	ratio	rate
3	2.76×10^{-6}	-	-	3.31×10^{-7}	-	-
6	4.77×10^{-8}	57.90	5.86	2.72×10^{-9}	121.66	6.93
12	7.61×10^{-10}	62.61	5.97	2.09×10^{-11}	129.95	7.02
	ME	ratio	rate	ME(U_x)	ratio	rate
	5.77×10^{-8}	-	-	1.66×10^{-7}	-	-
	2.64×10^{-10}	219.27	7.78	7.01×10^{-10}	236.22	7.88
	9.81×10^{-13}	687.41	9.43	2.10×10^{-12}	334.27	8.38
	ME(U_y)	ratio	rate			
	2.57×10^{-7}	-	-			
	1.81×10^{-9}	141.88	7.15			
	1.22×10^{-11}	148.89	7.22			

estimation in the 2D case. Note that while we have observed the existence of the internal superconvergent points in the 2D case, this result has not been proven in the literature.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

7.1.1 MOS

The MOS can be efficient for problems for which layer regions appear only in either the x or y dimension since we use a non-adaptive spatial discretization in one of the two dimensions. BACOL performs adaptive mesh refinement in the other dimension. The MOS requires that one pair of boundary conditions be differentiated; this can affect the accuracy of the numerical solution.

However, an improvement of the MOS algorithm could be possible if we were to make a modification of BACOL to allow it to accept extra algebraic equations; then none of the boundary conditions would need to be differentiated and this would improve the accuracy of the numerical solution. If BACOL were to be modified in this way and run with no spatial adaptivity, then the B-spline collocation version of the MOS and 2D B-spline collocation algorithm would be mathematically equivalent.

7.1.2 2D B-spline Collocation

Direct application of 2D B-spline collocation appears to be an interesting approach for the numerical solution of 2D parabolic PDEs because there is no need to make the restrictive assumptions regarding the boundary conditions that we have made in the MOS algorithm, and we do not need to differentiate the boundary conditions. In this thesis we have implemented a static (non-adaptive) 2D collocation algorithm and used it to successfully solve several test problems. In addition, we have observed the existence of superconvergent points that may be useful for error estimation.

Since we are using a tensor product framework to extend the B-spline collocation algorithm to 2D, the number of collocation points per subrectangle is $(p-1)(q-1)$ (assuming that the polynomials in the x and y domain have degree p and q respectively). While our 2D B-spline collocation algorithm usually produces an accurate solution, it needs substantially more computer time than in the 1D case. However, we introduced an algorithm to do a fast linear projection and its inverse, based on the tensor product form of the two ABD matrices and their LU decompositions. The amount of computation and memory is substantially reduced.

From our experience with the two algorithms considered in this thesis, i.e., the MOS and the 2D B-spline collocation algorithm, it appears that the latter approach is significantly better than the former approach (as it is now implemented). We have observed that the 2D B-spline collocation algorithm is able to solve much more difficult problems than the MOS algorithm. The parameter ξ which controls the difficulty of the 2D Burgers equation had to be set to a fairly large value (0.1 or 0.25) in order for the MOS algorithm to be able to solve the problem whereas the 2D B-spline collocation algorithm could solve the same problem with a significantly smaller value for ξ (0.01).

7.2 Future Work

Since the 2D B-spline collocation algorithm appears to be superior to the MOS algorithm, we limit our comments regarding future work to the former. Since PDEs frequently have large variations occurring over small regions of the physical domain, the next step will be effectively solving such problems by introducing mesh adaptivity to put more mesh points in the layer regions. Approaches that have been employed in MMM may be useful here. Such approaches use a MMPDE to transfer the PDE from the physical domain to a computational domain where tensor product B-spline collocation can be easily applied.

Bibliography

[AAF01] S. Adjerid, M. Aiffa, and J. E. Flaherty. Hierarchical finite element bases for triangular and tetrahedral elements. *Computer Methods in Applied Mechanics and Engineering*, 190(22–23):2925–2941, 2001.

[ACR81] U. Ascher, J. Christiansen, and R. D. Russell. Collocation software for boundary-value odes. *ACM Trans. Math. Softw.*, 7(2):209–222, 1981.

[AFMW92] S. Adjerid, J. E. Flaherty, P. K. Moore, and Y. J. Wang. High-order adaptive methods for parabolic systems. *Phys. D*, 60(1-4):94–111, 1992.

[AM02] S. Adjerid and T. C. Massey. A posteriori discontinuous finite element error estimation for two-dimensional hyperbolic problems. *Comput. Methods Appl. Mech. Engrg.*, 191(51-52):5877–5897, 2002.

[AMR95] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. *Numerical solution of boundary value problems for ordinary differential equations*, volume 13 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1995.

[AO00] M. Ainsworth and J. T. Oden. *A posteriori error estimation in finite element analysis*. Pure and Applied Mathematics (New York). Wiley-Interscience [John Wiley & Sons], New York, 2000.

[ASM09] T. Arsenault, T. Smith, and P. H. Muir. Superconvergent interpolants for efficient spatial error estimation in 1D PDE collocation solvers. *Can. Appl. Math. Q.*, 17(3):409–431, 2009.

[ASMK11] T. Arsenault, T. Smith, P.H. Muir, and P. Keast. Efficient interpolation-based error estimation for 1d time-dependent pde collocation codes. Technical Report 2011_001, Department of Mathematics and Computing Science, Saint Mary's University, Halifax, NS, 2011.

[ASMP12] T. Arsenault, T. Smith, P. H. Muir, and J. Pew. Asymptotically correct interpolation-based estimation in 1D PDE collocation solvers. *to appear in Can. Appl. Math. Q.*, 2012.

[Bak77] M. Bakker. *Software for semi-discretization of time dependent partial differential equations in one space variable.* Mathematisch Centrum, Afdeling Numerieke Wiskunde. Stichting Mathematisch Centrum, 1977.

[BER95] R. Beck, B. Erdmann, and R. Roitzsch. Kaskade 3.0 - an object-oriented adaptive finite element code. In *International workshop*, pages 105–124. Birkhaeuser, 1995.

[BFP+98] M. Berzins, R. Fairlie, S. V. Pennington, J. M. Ware, and L. E. Scales. Sprint2d: adaptive software for pdes. *ACM Trans. Math. Softw.*, 24(4):475–499, 1998.

[BHK07] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II—a general-purpose object-oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 2007.

[BHP94] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM J. Sci. Comput.*, 15(6):1467–1488, 1994.

[BL90] G. Birkhoff and R. E. Lynch. Iterative methods for large linear systems. chapter ELLPACK and ITPACK as research tools for solving elliptic problems, pages 41–63. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[BL97] A. M. Bruaset and H. P. Langtangen. A comprehensive set of tools for solving partial differential equations; diffpack. In M. Dhlen and A. Tveito, editors, *Numerical Methods and Software Tools in Industrial Mathematics*, pages 61–90. Birkhauser, Boston, 1997.

[BM02] E. Bertolazzi and G. Manzini. Algorithm 817: P2mesh: generic object-oriented interface between 2-d unstructured meshes and fem/fvm-based pde solvers. *ACM Trans. Math. Softw.*, 28(1):101–132, 2002.

[BPPW97] M. Berzins, S. V. Pennington, P. R. Pratt, and J. M. Ware. Modern software tools for scientific computing. chapter SPRINT2D software for convection dominated PDEs, pages 63–80. Birkhauser Boston Inc., Cambridge, MA, USA, 1997.

[BS73] C. de Boor and B. Swartz. Collocation at gaussian points. *SIAM Journal on Numerical Analysis*, 10(4):582–606, 1973.

[BTV96] J. G. Blom, R. A. Trompert, and J. G. Verwer. Algorithm 758: Vlugr2: a vectorizable adaptive-grid solver for pdes in 2d. *ACM Trans. Math. Softw.*, 22(3):302–328, 1996.

[BV96] J. G. Blom and J. G. Verwer. Algorithm 759: Vlugr3: a vectorizable adaptive-grid solver for pdes in 3d part ii. code description. *ACM Trans. Math. Softw.*, 22(3):329–347, 1996.

[Chr94] C. C. Christara. Quadratic spline collocation methods for elliptic partial differential equations. *BIT*, 34(1):33–61, 1994.

[CHR99] W. Cao, W. Huang, and R. D. Russell. A study of monitor functions for two-dimensional adaptive mesh generation. *SIAM J. Sci. Comput.*, 20(6):1978–1994 (electronic), 1999.

[CP76] J. H. Cerutti and S. V. Parter. Collocation methods for parabolic partial differential equations in one space dimension. *Numer. Math.*, 26(3):227–254, 1976.

[dB77] C. de Boor. Package for calculating with B-splines. *SIAM J. Numer. Anal.*, 14(3):441–472, 1977.

[dB78] C. de Boor. *A practical guide to splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1978.

[DD74] Jr. J. Douglas and T. Dupont. *Collocation methods for parabolic equations in a single space variable*. Lecture Notes in Mathematics, Vol. 385. Springer-Verlag, Berlin, 1974.

[DES82] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–408, 1982.

[DFK83a] J. C. Díaz, G. Fairweather, and P. Keast. Algorithm 603. COLROW and ARCECO: FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination. *ACM Trans. Math. Software*, 9(3):376–380, 1983.

[DFK83b] J. C. Díaz, G. Fairweather, and P. Keast. FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination. *ACM Trans. Math. Software*, 9(3):358–375, 1983.

[DMPP86] L. M. Delves, A. McKerrell, S. A. Peters, and C. Phillips. Performance of GEM2 on the ELLPACK problem population. *Internat. J. Numer. Methods Engrg.*, 23(2):229–238, 1986.

[DR87] W. R. Dyksen and C. J. Ribbens. Interactive ELLPACK: an interactive problem-solving environment for elliptic partial differential equations. *ACM Trans. Math. Software*, 13(2):113–132, 1987.

[dSF93] E. de Sturler and D. R. Fokkema. Nested krylov methods and preserving the orthogonality. *Sixth Copper Mountain Conference on Multigrid Methods, NASA Conference Publication 3224, Part 1*, pages 111–125, 1993.

[FL03] R. Fazio and R. J. LeVeque. Moving-mesh methods for one-dimensional hyperbolic problems using CLAWPACK. *Comput. Math. Appl.*, 45(1-3):273–298, 2003.

[Gea71] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1971.

[Goc02] M.S. Gockenbach. *Partial Differential Equations: Analytical and Numerical Methods.* Number v. 1 in Partial Differential Equations: Analytical and Numerical Methods. Society for Industrial and Applied Mathematics, 2002.

[Hin76] A.C. Hindmarsh. *Preliminary Documentation of GEARIB: Solution of implicit system of ordinary differential equations with banded with banded Jacobian.* Lawrence Livermore Laboratory, 1976.

[HMR85a] E. N. Houstis, W. F. Mitchell, and J. R. Rice. Algorithm 637: Gencol: collocation of general domains with bicubic hermite polynomials. *ACM Trans. Math. Softw.*, 11(4):413–415, 1985.

[HMR85b] E. N. Houstis, W. F. Mitchell, and J. R. Rice. Algorithm 638: Intcol and hermcol: collocation on rectangular domains with bicubic hermite polynomials. *ACM Trans. Math. Softw.*, 11(4):416–418, 1985.

[HMR85c] E. N. Houstis, W. F. Mitchell, and J. R. Rice. Collocation software for second-order elliptic partial differential equations. *ACM Trans. Math. Softw.*, 11(4):379–412, 1985.

[HR96] W. Huang and R. D. Russell. A moving collocation method for solving time dependent partial differential equations. *Appl. Numer. Math.*, 20(1-2):101–116, 1996.

[HR98a] W. Huang and R. D. Russell. A high-dimensional moving mesh strategy. *Appl. Numer. Math.*, 26(1-2):63–76, 1998.

[HR98b] W. Huang and R. D. Russell. Moving mesh strategy based on a gradient flow equation for two-dimensional problems. *SIAM J. Sci. Comput.*, 1998.

[HR11] W. Huang and R. D. Russell. *Adaptive moving mesh methods*, volume 174 of *Applied Mathematical Sciences.* Springer, New York, 2011.

[HS94] W. Huang and D. M. Sloan. A simple adaptive grid method in two dimensions. *SIAM J. Sci. Comput.*, 15(4):776–797, 1994.

[Hua01] W. Huang. Practical aspects of formulation and solution of moving mesh partial differential equations. *J. Comput. Phys.*, 171(2):753–775, 2001.

[HV03] W. Hundsdorfer and J. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2003.

[HVR88] E. N. Houstis, E. A. Vavalis, and J. R. Rice. Convergence of $O(h^4)$ cubic spline collocation methods for elliptic partial differential equations. *SIAM J. Numer. Anal.*, 25(1):54–74, 1988.

[HW93] E. Hairer and G. Wanner. *Solving Ordinary Differential EquationsII: Stiff and differential-algebraic problems*. Springer Series in Computational Mathematics. Springer, 1993.

[KM91] P. Keast and P. H. Muir. Algorithm 688:EPDCOL: a more efficient PDECOL code. *ACM Trans. Math. Softw.*, 17(2):153–166, 1991.

[Lan03] H. P. Langtangen. *Computational partial differential equations*, volume 1 of *Texts in Computational Science and Engineering*. Springer-Verlag, Berlin, second edition, 2003.

[LeV96] R. J. LeVeque. Clawpack—a software package for solving multidimensional conservation laws. In *Hyperbolic problems: theory, numerics, applications (Stony Brook, NY, 1994)*, pages 188–197. World Sci. Publ., River Edge, NJ, 1996.

[LeV07] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Classics in Applied Mathematics Classics in Applied Mathemat. Society for Industrial and Applied Mathematics, 2007.

[mol] http://reference.wolfram.com/legacy/v52/builtinfunctions/advanceddocumentation/differentialequations/ndsolve/partialdifferentialequations/thenumericalmethodoflines/introduction.html.

[Moo95] P. K. Moore. Comparison of adaptive methods for one-dimensional parabolic systems. *Applied Numerical Mathematics*, 16(4):471 – 488, 1995.

[Moo01] P. K. Moore. Interpolation error-based a posteriori error estimation for two-point boundary value problems and parabolic equations in one space dimension. *Numer. Math.*, 90(1):149–177, 2001.

[MRB05] R. M.M. Mattheij, S. W. Rienstra, and J.H.M.T.T. Boonkkamp. *Partial Differential Equations: Modeling, Analysis, Computation*. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics, 2005.

[MS79] N. K. Madsen and R. F. Sincovec. Algorithm 540: PDECOL, general collocation software for partial differential equations [d3]. *ACM Trans. Math. Softw.*, 5(3):326–351, 1979.

[Ng05] K. S. Ng. *Spline collocation on adaptive grids and non-rectangular regions*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, ON, 2005.

[Pet83] L. R. Petzold. A description of DASSL: a differential/algebraic system solver. In *Scientific computing (Montreal, Que., 1982)*, IMACS Trans. Sci. Comput., I, pages 65–68. IMACS, New Brunswick, NJ, 1983.

[PW95] I. Patlashenko and T. Weller. Two-dimensional spline collocation method for nonlinear analysis of laminated panels. *Computers Structures*, 57(1):131–139, 1995.

[RS97] R. D. Russell and W. Sun. Spline collocation differentiation matrices. *SIAM J. Numer. Anal.*, 34(6):2274–2287, 1997.

[Saa81] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comp.*, 37(155):105–126, 1981.

[Saa90] Y. Saad. Sparskit: a basic tool-kit for sparse matrix computations. Technical Report RIACS9020, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, CA, 1990.

[Saa93] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[Saa03] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[Sch91] W. E. Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations*. Academic Press, 1991.

[SS86] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[Sun01] W. Sun. B-spline collocation methods for elasticity problems. In *Scientific computing and applications (Kananaskis, AB, 2000)*, volume 7 of *Adv. Comput. Theory Pract.*, pages 133–141. Nova Sci. Publ., Huntington, NY, 2001.

[VB06] A. C. Velivelli and K. M. Bryden. Parallel performance and accuracy of lattice boltzmann and traditional finite difference methods for solving the unsteady two-dimensional burger's equation. *Physica A: Statistical Mechanics and its Applications*, 362(1):139–145, 2006.

[vdV92] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[Ver80a] J. G. Verwer. Algorithm 553: M3rk, an explicit time integrator for semidiscrete parabolic equations. *ACM Trans. Math. Softw.*, 6(2):236–239, 1980.

[Ver80b] J. G. Verwer. An implementation of a class of stabilized explicit methods for the time integration of parabolic equations. *ACM Trans. Math. Softw.*, 6(2):188–205, 1980.

[VvZ07] T. Vejchodský, P. Šolín, and M. Zítka. Modular hp-fem system hermes and its application to maxwell's equations. *Math. Comput. Simul.*, 76(1-3):223–228, 2007.

[VWSS01] A. Vande Wouwer, Ph. Saucez, and W. E. Schiesser, editors. *Adaptive method of lines.* Chapman & Hall/CRC, Boca Raton, FL, 2001.

[Wan95] Y. Wang. *A Parallel Collocation Method for Two Dimensional Linear Parabolic Separable Partial Differential Equations.* PhD thesis, Department of Mathematics and Statistics, Dalhousie University, Halifax, NS, 1995.

[WKM04a] R. Wang, P. Keast, and P. H. Muir. BACOL: B-spline adaptive collocation software for 1-d parabolic pdes. *ACM Trans. Math. Softw.*, 30(4):454–470, 2004.

[WKM04b] R. Wang, P. Keast, and P. H. Muir. A comparison of adaptive software for 1D parabolic PDEs. *J. Comput. Appl. Math.*, 169(1):127–150, 2004.

[WKM04c] R. Wang, P. Keast, and P. H. Muir. A high-order global spatially adaptive collocation method for 1-d parabolic pdes. *Appl. Numer. Math.*, 50(2):239–260, 2004.

[WKM08] R. Wang, P. Keast, and P. H. Muir. Algorithm 874: BACOLR: spatial and temporal error control software for pdes based on high-order adaptive collocation. *ACM Trans. Math. Softw.*, 34(3):15:1–15:28, 2008.

[WMKL93] S. Wendel, H. Maisch, H. Karl, and G. Lehner. Two-dimensional b-spline finite elements and their application to the computation of solitons. *Electrical Engineering (Archiv fur Elektrotechnik)*, 76:427–435, 1993.

Appendix A

Source Code

A.1 A Finite Difference based MOS Scheme

These are the user-supplied subroutines for the 2D Burgers' equation.

```
      SUBROUTINE F(T, X, U, UX, UXX, FVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THIS SUBROUTINE DEFINES THE RIGHT HAND SIDE VECTOR OF THE
C       NPDE DIMENSIONAL PARABOLIC PARTIAL DIFFERENTIAL EQUATION
C                          UT = F(T, X, U, UX, UXX).
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
C
```

```
      DOUBLE PRECISION      X
C                           THE CURRENT SPATIAL COORDINATE.
C
      DOUBLE PRECISION      U(NPDE)
C                           U(1:NPDE) IS THE APPROXIMATION OF THE
C                           SOLUTION AT THE POINT (T,X).
C
      DOUBLE PRECISION      UX(NPDE)
C                           UX(1:NPDE) IS THE APPROXIMATION OF THE
C                           SPATIAL DERIVATIVE OF THE SOLUTION AT
C                           THE POINT (T,X).
C
      DOUBLE PRECISION      UXX(NPDE)
C                           UXX(1:NPDE) IS THE APPROXIMATION OF THE
C                           SECOND SPATIAL DERIVATIVE OF THE
C                           SOLUTION AT THE POINT (T,X).
C
      DOUBLE PRECISION      U0
C                           U0 IS THE EXACT
C                           SOLUTION AT THE POINT (Y0,T,X).
C                           (Y0=0.D0)
      DOUBLE PRECISION      UNPDE_1
C                           UNPDE_1 IS THE EXACT
C                           SOLUTION AT THE POINT (Y(NPDE+1),T,X).
C                           (Y(NPDE+1)=1.D0)
C
C OUTPUT:
      DOUBLE PRECISION      FVAL(NPDE)
C                           FVAL(1:NPDE) IS THE RIGHT HAND SIDE
C                           VECTOR F(T, X, U, UX, UXX) OF THE PDE.
      DOUBLE PRECISION      COEFF, H
      COMMON /TWOD/         COEFF, H
```

```
      INTEGER                 I
C-----------------------------------------------------------------------
C     SOLUTION AT THE POINT (Y0,T,X). (Y0=0.D0)
      U0 = 1.0D0 / (1.0D0+DEXP((X-T) / (2*COEFF)))

C     SOLUTION AT THE POINT (Y(NPDE+1),T,X). (Y(NPDE+1)=1.D0)
      UNPDE_1 = 1.0D0 / (1.0D0+DEXP((X+1.0D0-T) / (2*COEFF)))

      FVAL(1) = COEFF * UXX(1) - U(1) * UX(1)
     &          + COEFF / (H*H) * (U(2)-2*U(1)+U0)
     &          - U(1) * (U(2)-U0) / (2*H)

      DO 10 I = 2, NPDE-1

C     CENTRAL FINITE DIFFERENCE, SO THREE PIONTS ARE NON-ZERO
      FVAL(I) = COEFF * UXX(I) - U(I) * UX(I)
     &          + COEFF / (H*H) * (U(I+1)-2*U(I)+U(I-1))
     &          - U(I)*(U(I+1)-U(I-1))/(2*H)

   10 CONTINUE

      FVAL(NPDE) = COEFF * UXX(NPDE) - U(NPDE) * UX(NPDE)
     &          + COEFF / (H*H) * (UNPDE_1-2*U(NPDE)+U(NPDE-1))
     &          - U(NPDE)*(UNPDE_1-U(NPDE-1))/(2*H)
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE DERIVF(T, X, U, UX, UXX, DFDU, DFDUX, DFDUXX, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THIS SUBROUTINE IS USED TO DEFINE THE INFORMATION ABOUT THE
```

```
C       PDE REQUIRED TO FORM THE ANALYTIC JACOBIAN MATRIX FOR THE DAE
C       OR ODE SYSTEM. ASSUMING THE PDE IS OF THE FORM
C                              UT = F(T, X, U, UX, UXX)
C       THIS ROUTINE RETURNS THE JACOBIANS D(F)/D(U), D(F)/D(UX), AND
C       D(F)/D(UXX).
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        X
C                               THE CURRENT SPATIAL COORDINATE.
C
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE APPROXIMATION OF THE
C                               SOLUTION AT THE POINT (T,X).
C
        DOUBLE PRECISION        UX(NPDE)
C                               UX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SPATIAL DERIVATIVE OF THE SOLUTION AT
C                               THE POINT (T,X).
C
        DOUBLE PRECISION        UXX(NPDE)
C                               UXX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SECOND SPATIAL DERIVATIVE OF THE
C                               SOLUTION AT THE POINT (T,X).
C
```

```
C OUTPUT:
      DOUBLE PRECISION      DFDU(NPDE,NPDE)
C                           DFDU(I,J) IS THE PARTIAL DERIVATIVE
C                           OF THE I-TH COMPONENT OF THE VECTOR F
C                           WITH RESPECT TO THE J-TH COMPONENT
C                           OF THE UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION      DFDUX(NPDE,NPDE)
C                           DFDUX(I,J) IS THE PARTIAL DERIVATIVE
C                           OF THE I-TH COMPONENT OF THE VECTOR F
C                           WITH RESPECT TO THE J-TH COMPONENT
C                           OF THE SPATIAL DERIVATIVE OF THE
C                           UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION      DFDUXX(NPDE,NPDE)
C                           DFDUXX(I,J) IS THE PARTIAL DERIVATIVE
C                           OF THE I-TH COMPONENT OF THE VECTOR F
C                           WITH RESPECT TO THE J-TH COMPONENT
C                           OF THE SECOND SPATIAL DERIVATIVE OF THE
C                           UNKNOWN FUNCTION U.
      DOUBLE PRECISION      COEFF, H
      COMMON /TWOD/         COEFF, H
      INTEGER               I, J
C-----------------------------------------------------------------------
C     CENTERED FINITE DIFFERENCE,
C     THE FRIST ROW HAS TWO NON-ZERO ELEMENTS
      DFDU(1,1) = -UX(1) - COEFF / (H*H)* 2
     &            - (U(2) - DEXP((X-T) / COEFF))/ (2*H)
      DFDU(1,2) = COEFF / (H*H) - U(1) / (2*H)

      DO 10 J = 3, NPDE
         DFDU(1,J) = 0.D0
```

```
 10   CONTINUE

      DO 20 I = 2, NPDE-1
         DO 11 J = 1, I-2
            DFDU(I,J) = 0.D0
 11      CONTINUE

C     CENTERED FINITE DIFFERENCE, SO THREE PIONTS ARE NON-ZERO
         DFDU(I,I-1) = COEFF / (H*H) + U(I) / (2*H)
         DFDU(I,I) = -UX(I) - COEFF / (H*H)* 2 - (U(I+1)-U(I-1)) / (2*H)
         DFDU(I,I+1) = COEFF / (H*H) - U(I) / (2*H)

         DO 12 J =I+2, NPDE
            DFDU(I,J) = 0.D0
 12      CONTINUE
 20   CONTINUE

C     CENTERED FINITE DIFFERENCE, THE LAST ROW HAS TWO NON-ZERO ELEMENTS
      DO 30 J = 1, NPDE-2
            DFDU(NPDE,J) = 0.D0
 30   CONTINUE

      DFDU(NPDE,NPDE-1) = COEFF / (H*H) + U(NPDE) / (2*H)
      DFDU(NPDE,NPDE) = -UX(NPDE) - COEFF / (H*H)* 2
     &                  - (DEXP((X-T) / COEFF) - U(NPDE-1)) / (2*H)

      DO 40 I = 1, NPDE
         DO 31 J = 1, NPDE
            IF (J .EQ. I) THEN
               DFDUX(I,J) = -U(I)
            ELSE
               DFDUX(I,J) = 0.D0
```

```
            ENDIF
 31       CONTINUE
 40    CONTINUE

       DO 50 I = 1, NPDE
          DO 41 J = 1, NPDE
             IF (J .EQ. I) THEN
                DFDUXX(I,J) = COEFF
             ELSE
                DFDUXX(I,J) = 0.D0
             ENDIF
 41       CONTINUE
 50    CONTINUE

       RETURN
       END
C-----------------------------------------------------------------------
       SUBROUTINE BNDXA(T, U, UX, BVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C        THE SUBROUTINE IS USED TO DEFINE THE BOUNDARY CONDITIONS AT THE
C        LEFT SPATIAL END POINT X = XA.
C                                B(T, U, UX) = 0
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
         INTEGER                 NPDE
C                                THE NUMBER OF PDES IN THE SYSTEM.
C
         DOUBLE PRECISION        T
C                                THE CURRENT TIME COORDINATE.
```

```
C
      DOUBLE PRECISION        U(NPDE)
C                             U(1:NPDE) IS THE APPROXIMATION OF THE
C                             SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION        UX(NPDE)
C                             UX(1:NPDE) IS THE APPROXIMATION OF THE
C                             SPATIAL DERIVATIVE OF THE SOLUTION AT
C                             THE POINT (T,XA).
C OUTPUT:
      DOUBLE PRECISION        BVAL(NPDE)
C                             BVAL(1:NPDE) IS THE BOUNDARY CONTIDITION
C                             AT THE LEFT BOUNDARY POINT.
      DOUBLE PRECISION        COEFF, H
      COMMON /TWOD/           COEFF, H
      INTEGER                 I
C-----------------------------------------------------------------------

      DO 10 I = 1, NPDE
         BVAL(I) = (1.0D0 + DEXP((I*H-T) / (2*COEFF))) * U(I) - 1.0D0
   10 CONTINUE

      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BNDXB(T, U, UX, BVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE BOUNDARY CONDITIONS AT THE
C       RIGHT SPATIAL END POINT X = XB.
C                             B(T, U, UX) = 0
C
```

```
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
      INTEGER                 NPDE
C                             THE NUMBER OF PDES IN THE SYSTEM.
C
      DOUBLE PRECISION        T
C                             THE CURRENT TIME COORDINATE.
C
      DOUBLE PRECISION        U(NPDE)
C                             U(1:NPDE) IS THE APPROXIMATION OF THE
C                             SOLUTION AT THE POINT (T,XB).
C
      DOUBLE PRECISION        UX(NPDE)
C                             UX(1:NPDE) IS THE APPROXIMATION OF THE
C                             SPATIAL DERIVATIVE OF THE SOLUTION AT
C                             THE POINT (T,XB).
C
C OUTPUT:
      DOUBLE PRECISION        BVAL(NPDE)
C                             BVAL(1:NPDE) IS THE BOUNDARY CONTIDITION
C                             AT THE RIGHT BOUNDARY POINT.
      DOUBLE PRECISION        COEFF, H
      COMMON /TWOD/           COEFF, H
      INTEGER                 I
C-----------------------------------------------------------------------
      DO 10 I = 1, NPDE
      BVAL(I) = (1.0D0 + DEXP((1.0D0+I*H-T) / (2*COEFF))) * U(I) - 1.0D0
   10 CONTINUE

      RETURN
      END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE DIFBXA(T, U, UX, DBDU, DBDUX, DBDT, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE DIFFERENTIATED BOUNDARY
C       CONDITIONS AT THE LEFT SPATIAL END POINT X = XA. FOR THE
C       BOUNDARY CONDITION EQUATION
C                                   B(T, U, UX) = 0
C       THE PARTIAL DERIVATIVES DB/DU, DB/DUX, AND DB/DT ARE SUPPLIED
C       BY THIS ROUTINE.
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
       INTEGER                 NPDE
C                              THE NUMBER OF PDES IN THE SYSTEM.
C
       DOUBLE PRECISION        T
C                              THE CURRENT TIME COORDINATE.
C
       DOUBLE PRECISION        U(NPDE)
C                              U(1:NPDE) IS THE APPROXIMATION OF THE
C                              SOLUTION AT THE POINT (T,X).
C
       DOUBLE PRECISION        UX(NPDE)
C                              UX(1:NPDE) IS THE APPROXIMATION OF THE
C                              SPATIAL DERIVATIVE OF THE SOLUTION AT
C                              THE POINT (T,X).
C
C OUTPUT:
       DOUBLE PRECISION        DBDU(NPDE,NPDE)
C                              DBDU(I,J) IS THE PARTIAL DERIVATIVE
```

```
C                                 OF THE I-TH COMPONENT OF THE VECTOR B
C                                 WITH RESPECT TO THE J-TH COMPONENT
C                                 OF THE UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION            DBDUX(NPDE,NPDE)
C                                 DBDUX(I,J) IS THE PARTIAL DERIVATIVE
C                                 OF THE I-TH COMPONENT OF THE VECTOR B
C                                 WITH RESPECT TO THE J-TH COMPONENT
C                                 OF THE SPATIAL DERIVATIVE OF THE
C                                 UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION            DBDT(NPDE)
C                                 DBDT(I) IS THE PARTIAL DERIVATIVE
C                                 OF THE I-TH COMPONENT OF THE VECTOR B
C                                 WITH RESPECT TO TIME T.
      DOUBLE PRECISION            COEFF, H
      COMMON /TWOD/               COEFF, H
      INTEGER                     I, J
C-----------------------------------------------------------------------
      DO 10 I = 1, NPDE
         DO 1 J = 1, NPDE
            IF (J .EQ. I) THEN
               DBDU(I,J) = 1.0D0 + DEXP((I*H-T) / (2*COEFF))
            ELSE
               DBDU(I,J) = 0.D0
            ENDIF
 1       CONTINUE
 10   CONTINUE

      DO 20 I = 1, NPDE
         DO 11 J = 1, NPDE
               DBDUX(I,J) = 0.D0
```

```
 11      CONTINUE
 20   CONTINUE

      DO 30 I = 1, NPDE
         DBDT(I) = 1.0D0/(2*COEFF) * DEXP((I*H-T) / (2*COEFF)) * U(I)
 30   CONTINUE

      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE DIFBXB(T, U, UX, DBDU, DBDUX, DBDT, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C        THE SUBROUTINE IS USED TO DEFINE THE DIFFERENTIATED BOUNDARY
C        CONDITIONS AT THE RIGHT SPATIAL END POINT 1 = XB. FOR THE
C        BOUNDARY CONDITION EQUATION
C                                   B(T, U, UX) = 0
C        THE PARTIAL DERIVATIVES DB/DU, DB/DUX, AND DB/DT ARE SUPPLIED
C        BY THIS ROUTINE.
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
         INTEGER                 NPDE
C                                THE NUMBER OF PDES IN THE SYSTEM.
C
         DOUBLE PRECISION        T
C                                THE CURRENT TIME COORDINATE.
C
         DOUBLE PRECISION        U(NPDE)
C                                U(1:NPDE) IS THE APPROXIMATION OF THE
C                                SOLUTION AT THE POINT (T,X).
```

```
C
      DOUBLE PRECISION        UX(NPDE)
C                             UX(1:NPDE) IS THE APPROXIMATION OF THE
C                             SPATIAL DERIVATIVE OF THE SOLUTION AT
C                             THE POINT (T,X).
C
C OUTPUT:
      DOUBLE PRECISION        DBDU(NPDE,NPDE)
C                             DBDU(I,J) IS THE PARTIAL DERIVATIVE
C                             OF THE I-TH COMPONENT OF THE VECTOR B
C                             WITH RESPECT TO THE J-TH COMPONENT
C                             OF THE UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION        DBDUX(NPDE,NPDE)
C                             DBDUX(I,J) IS THE PARTIAL DERIVATIVE
C                             OF THE I-TH COMPONENT OF THE VECTOR B
C                             WITH RESPECT TO THE J-TH COMPONENT
C                             OF THE SPATIAL DERIVATIVE OF THE
C                             UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION        DBDT(NPDE)
C                             DBDT(I) IS THE PARTIAL DERIVATIVE
C                             OF THE I-TH COMPONENT OF THE VECTOR B
C                             WITH RESPECT TO TIME T.
      DOUBLE PRECISION        COEFF, H
      COMMON /TWOD/           COEFF, H
      INTEGER                 I, J
C-----------------------------------------------------------------------
      DO 10 I = 1, NPDE
         DO 1 J = 1, NPDE
            IF (J .EQ. I) THEN
               DBDU(I,J) = 1.0D0 + DEXP((1.0D0+I*H-T) / (2*COEFF))
```

```
            ELSE
               DBDU(I,J) = 0.D0
            ENDIF
 1       CONTINUE
 10   CONTINUE

      DO 20 I = 1, NPDE
         DO 11 J = 1, NPDE
               DBDUX(I,J) = 0.D0
 11      CONTINUE
 20   CONTINUE

      DO 30 I = 1, NPDE
         DBDT(I) = 1.0D0/(2.0*COEFF) *
     *           DEXP((1.0D0+I*H-T) / (2.0*COEFF)) * U(I)
 30   CONTINUE

      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE UINIT(X, U, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C        THIS SUBROUTINE IS USED TO RETURN THE NPDE-VECTOR OF INITIAL
C        CONDITIONS OF THE UNKNOWN FUNCTION AT THE INITIAL TIME T = T0
C        AT THE SPATIAL COORDINATE X.
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        DOUBLE PRECISION        X
C                               THE SPATIAL COORDINATE.
```

```
C
      INTEGER                 NPDE
C                             THE NUMBER OF PDES IN THE SYSTEM.
C
C OUTPUT:
      DOUBLE PRECISION        U(NPDE)
C                             U(1:NPDE) IS VECTOR OF INITIAL VALUES OF
C                             THE UNKNOWN FUNCTION AT T = T0 AND THE
C                             GIVEN VALUE OF X.
      DOUBLE PRECISION        COEFF, H
      COMMON /TWOD/           COEFF, H
      INTEGER                 I
C-----------------------------------------------------------------------
C
C     ASSIGN U(1:NPDE) THE INITIAL VALUES OF U(T0,X).

      DO 10 I = 1, NPDE
         U(I) = 1.0D0 / (1.0D0+DEXP((X+I*H) / (2*COEFF)))
   10 CONTINUE

      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE TRUU(T, X, U, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C     THIS FUNCTION PROVIDES THE EXACT SOLUTION OF THE PDE.
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
      INTEGER                 NPDE
C                             THE NUMBER OF PDES IN THE SYSTEM.
```

```
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        X
C                               THE CURRENT SPATIAL COORDINATE.
C
C OUTPUT:
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE EXACT SOLUTION AT THE
C                               POINT (T,X).
C-----------------------------------------------------------------------
        DOUBLE PRECISION        COEFF, H
        COMMON /TWOD/           COEFF, H
        INTEGER                 I
C-----------------------------------------------------------------------
      DO 10 I = 1, NPDE
         U(I) = 1.0D0 / (1.0D0+DEXP((X+I*H-T) / (2*COEFF)))
   10 CONTINUE

      RETURN
      END
```

A.2 A B-spline Gaussian Collocation based MOS Scheme

These are the user-supplied subroutines for the 2D Burgers' equation.

```
C-----------------------------------------------------------------------
C DRIVING PROGRAM FOR THE SYSTEM OF NPDE EQUATIONS DESCRIBED IN THE
c PAPER.
C U = 1.0D0 / (1.0D0+DEXP((X+Y-T) / (2*COEFF)))
C-----------------------------------------------------------------------
```

```
C CONSTANTS:
      DOUBLE PRECISION      ZERO
      PARAMETER             (ZERO = 0.0D0)
C
      DOUBLE PRECISION      NEGONE
      PARAMETER             (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
      INTEGER               NCONTI
      PARAMETER             (NCONTI = 2)
C                           NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                           AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER               YKCOL
C                           YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                          TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                          WHICH IS EQUAL TO THE DEGREE OF THE
C                          PIECEWISE POLYNOMIALS MINUS ONE.
C                             1 < YKCOL < 11.
      PARAMETER             (YKCOL = 3)
C
      INTEGER               YNINT
      PARAMETER             (YNINT = 4)
C                           YNINT IS THE NUMBER OF SUBINTERVALS
C                           DEFINED BY THE SPATIAL MESH Y.
C
      INTEGER               YNCPTS
      PARAMETER             (YNCPTS=(YKCOL*YNINT+NCONTI))
C                           YNCPTS IS THE NUMBER
C                           OF COLLOCATION POINTS.
C
```

```
      INTEGER                 NPDE
C                             NUMBER OF PDES
C                             =THE NUMBER OF COLLOCATION POINTS
C                             IN Y DOMAIN
      PARAMETER               (NPDE = YKCOL*YNINT+NCONTI)
C
      INTEGER                 YNELS
      PARAMETER               (YNELS=YKCOL*(YKCOL+NCONTI))
C                             THE NUMBER OF ELEMENTS IN ONE
C                             COLLOCATION BLOCK OF WORK.
C
C-----------------------------------------------------------------------
C     Y DIMENSION
      DOUBLE PRECISION        YA
C                             THE LEFT BOUNDARY POINT
      PARAMETER               (YA = 0.0D0)
      DOUBLE PRECISION        YB
C                             THE RIGHT BOUNDARY POINT
      PARAMETER               (YB = 1.0D0)
C
      DOUBLE PRECISION        Y(YNINT+1)
C                             Y IS THE SPATIAL MESH WHICH DIVIDES THE
C                             INTERVAL [Y_A,Y_B] AS: Y_A = Y(1) <
C                             Y(2) < Y(3) < ... < Y(NINT+1) = Y_B.
C
      DOUBLE PRECISION        YBS(YNCPTS+YKCOL+NCONTI)
C                             THE BREAKPOINT SEQUENCE.
C                             YBS(I)=X(1), I=1, KCOL+NCONTI;
C                             YBS((I-1)*KCOL+NCONTI+J)=Y(I),
C                                   I=2, NINT;  J=1, KCOL
C                           YBS(YNCPTS+I)=Y(NINT+1), I=1,KCOL+NCONTI.
C
```

```
      DOUBLE PRECISION      YH(YNINT)
C                           YH IS THE MESH STEP SIZE SEQUENCE
C                           (B-SPLINE).
      DOUBLE PRECISION      EXCOL(YNINT*(YKCOL+3))
C                           EXCOL IS THE COLLOCATION POINT SEQUENCE
C                           WHICH IS USED FOR ERROR ESTIMATE.
C
      DOUBLE PRECISION      EWTS(YNINT*(YKCOL+3))
C                           EWTS IS THE GAUSSIAN WEIGHT SEQUENCE
C                           WHICH IS USED FOR ERROR ESTIMATE.
C
C-----------------------------------------------------------------------
C     X DIMENSION
      INTEGER               XKCOL
C                           KCOL IS THE NUMBER OF COLLOCATION POINTS
C                           TO BE USED IN EACH SUBINTERVAL, WHICH IS
C                           EQUAL TO THE DEGREE OF THE PIECEWISE
C                           POLYNOMIALS MINUS ONE.
C                           1 < KCOL < 11.
      PARAMETER             (XKCOL = 5)
C
      INTEGER               XNINT
      PARAMETER             (XNINT = 4)
C                           NINT IS THE NUMBER OF SUBINTERVALS
C                           DEFINED BY THE SPATIAL MESH X.
C
C-----------------------------------------------------------------------
C     X DIMENSION
      INTEGER               NINTMX
C                           MAXIMAL NUMBER OF INTEVALS ALLOWED
      PARAMETER             (NINTMX = 1000)
C
```

```
      INTEGER                 MAXVEC
C                             THE DIMENSION OF THE VECTOR OF
C                             BSPLINE COEFFICIENTS
      PARAMETER               (MAXVEC = NPDE*(NINTMX*XKCOL+2))
C
      INTEGER                 NXOUT
      PARAMETER               (NXOUT = 21)
C                             NXOUT IS THE NUMER OF A SET OF SPATIAL
C                             POINTS FOR OUTPUT
C
      INTEGER                 NUOUT
C                             THE DIMENSION OF UOUT
      PARAMETER               (NUOUT = NPDE*NXOUT)
C
      INTEGER                 LRP
C                             SEE THE COMMENT FOR RPAR
      PARAMETER               (LRP = 134+NINTMX*(35+35*XKCOL+31*NPDE
     +                              +38*NPDE*XKCOL+8*XKCOL*XKCOL)
     +                              +14*XKCOL+79*NPDE+NPDE*NPDE*(21
     +                              +4*NINTMX*XKCOL*XKCOL
     +                              +12*NINTMX*XKCOL+6*NINTMX))
C
      INTEGER                 LIP
C                             SEE THE COMMENT FOR IPAR
      PARAMETER               (LIP = 115+NPDE*((2*XKCOL+1)*NINTMX+4))
C
C
      DOUBLE PRECISION        RPAR(LRP)
C                             RPAR IS A FLOATING POINT WORK ARRAY
C                             OF SIZE LRP.
C
      INTEGER                 IPAR(LIP)
```

```
C                                IPAR IS AN INTEGER WORK ARRAY
C                                OF SIZE LIP.
C
      INTEGER                    LENWRK
C                                THE DIMENSION OF ARRAY WORK WHEN WE
C                                CALL VALUES
      PARAMETER                  (LENWRK =(XKCOL+2)+XKCOL*(NINTMX+1)+4)
C
C-----------------------------------------------------------------------
C     X DIMENSION
      DOUBLE PRECISION           XA
C                                THE LEFT BOUNDARY POINT
      PARAMETER                  (XA = 0.0D0)
      DOUBLE PRECISION           XB
C                                THE RIGHT BOUNDARY POINT
      PARAMETER                  (XB = 1.0D0)
C
      DOUBLE PRECISION           X(NINTMX+1)
C                                X IS THE SPATIAL MESH WHICH DIVIDES THE
C                                INTERVAL [X_A,X_B] AS: X_A = X(1) <
C                                X(2) < X(3) < ... < X(NINT+1) = X_B.
C
      DOUBLE PRECISION           W(MAXVEC)
C                                ON SUCCESSFUL RETURN FROM BACOL, Y IS
C                                THE VECTOR OF BSPLINE
C                                COEFFICIENTS AT THE CURRENT TIME T0.
C
C-----------------------------------------------------------------------
      DOUBLE PRECISION           T0
C                                T0 < TOUT IS THE INITIAL TIME.
C
      DOUBLE PRECISION           TOUT
```

```
C                                TOUT IS THE DESIRED FINAL OUTPUT TIME.
C
      DOUBLE PRECISION           ATOL(NPDE)
C                                ATOL IS THE ABSOLUTE ERROR TOLERANCE
C                                REQUEST AND IS A SCALAR QUANTITY IF
C                                MFLAG(2) = 0.
C
      DOUBLE PRECISION           RTOL(NPDE)
C                                RTOL IS THE RELATIVE ERROR TOLERANCE
C                                REQUEST AND IS A SCALAR QUANTITY IF
C                                MFLAG(2) = 0.
C
      INTEGER                    MFLAG(7)
C                                THIS VECTOR OF USER INPUT DETERMINES
C                                THE INTERACTION OF BACOL WITH DASSL.

C-----------------------------------------------------------------------

      INTEGER                    IDID
C                                IDID IS THE BACOL EXIT STATUS FLAG
C                                WHICH IS BASED ON THE EXIT STATUS FROM
C                                DASSL ON ERROR CHECKING PERFORMED BY
C                                BACOL ON INITIALIZATION.
C-----------------------------------------------------------------------

      DOUBLE PRECISION           EXACTU(NPDE)
C                                EXACT SOLUTION AT CERTAIN POINT
      DOUBLE PRECISION           UOUT(NUOUT)
C                                THE APPROXIMATION SOLUTIONS AT A SET
C                                OF POINTS
      DOUBLE PRECISION           VALWRK(LENWRK)
C                                VALWRK IS A WORK ARRAY IN VALUES
```

```
      DOUBLE PRECISION      XOUT(NXOUT)
C                           XOUT IS A SET OF SPATIAL POINTS FOR
C                           OUTPUT
C
      DOUBLE PRECISION      UBAROUTM(NPDE,NXOUT)
C                           THE APPROXIMATION SOLUTIONS AT A SET
C                           OF POINTS(MATRIX FORM)
      DOUBLE PRECISION      UOUTM(NPDE,NXOUT)
C                           THE APPROXIMATION SOLUTIONS AT A SET
C                           OF POINTS(MATRIX FORM)
      DOUBLE PRECISION      TRY(NPDE,NXOUT)
C                           THE APPROXIMATION SOLUTIONS AT A SET
C                           OF POINTS(MATRIX FORM)
      DOUBLE PRECISION      EXACTUM(NPDE,NXOUT)
C                           THE EXACT SOLUTIONS AT A SET
C                           OF POINTS(MATRIX FORM)
      DOUBLE PRECISION      ERRORM(NPDE,NXOUT)
C                           THE ERROR MATRIXM = EXACTUM - UOUTM
      DOUBLE PRECISION      ERROR1NORM
C                           1-NORM OF THE ERROR MATRIX
      DOUBLE PRECISION      ERRORINFNORM
C                           INFINITE-NORM OF THE ERROR MATRIX
      DOUBLE PRECISION      ERRORMAXNORM
C                           MAX-NORM OF THE ERROR MATRIX
C
C-----------------------------------------------------------------------
C
C     WORK STORAGE:
      DOUBLE PRECISION      WORK1((YKCOL+3)*(YKCOL+3))
C                           WORK IS A FLOATING POINT WORK STORAGE
C                           ARRAY.
C
```

```
      DOUBLE PRECISION      WORK22(YKCOL*YKCOL)
C                           WORK IS A FLOATING POINT WORK STORAGE
C                           ARRAY.
C
      INTEGER               ICFLAG
C                           THIS IS THE STATUS FLAG FROM COLROW
C                           WHICH IS CALLED BY CRDCMP.
C                           ICFLAG =  0, INDICATES NON-SINGULARITY.
C                           ICFLAG = -1, INDICATES SINGULARITY.
C                           ICFLAG =  1, INDICATES INVALID INPUT.
C
      DOUBLE PRECISION      YFBASIS(*)
C                           BASIS FUNCTION VALUES AT THE COLLOCATION
C                           POINTS. FBASIS(K,J,I) CONTAINS THE
C                           VALUES OF THE (J-1)ST DERIVATIVE
C                           (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                           FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                           I-TH COLLOCATION POINT.
C
      DOUBLE PRECISION      COEFF
C                           COEFF IS THE COEFFCIENT OF UXX IN THE
C                           BURGERS' EQUATION
C
C     INTEGER               IPIVOT(NPDE)
      INTEGER               IPIVOT(*)
C                           PIVOTING INFORMATION FROM THE
C                           FACTORIZATION OF THE TEMPORARY MATRIX.
C
      DOUBLE PRECISION      YABD(*)
C                           ABD
C
      INTEGER               IABDTP
```

```
C                                WORK(IABDTP) CONTAINS A COPY OF THE TOP
C                                BLOCK WHICH IS REQUIRED SINCE CRDCMP
C                                OVERWRITES THE INPUT COLLOCATION MATRIX.
C
      INTEGER                    IABDBK
C                                WORK(IABDBK) CONTAINS A COPY OF ABDBLK
C                                WHICH IS REQUIRED SINCE CRDCMP
C                                OVERWRITES THE INPUT COLLOCATION MATRIX.
C
      INTEGER                    IABDBT
C                                WORK(IABDBT) CONTAINS A COPY OF THE
C                                BOTTOM BLOCK WHICH IS REQUIRED SINCE
C                                CRDCMP OVERWRITES THE INPUT COLLOCATION
C                                MATRIX.
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER                    KCOLY
C                                YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                              TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                                WHICH IS EQUAL TO THE DEGREE OF THE
C                                PIECEWISE POLYNOMIALS MINUS ONE.
C                                 1 < YKCOL < 11.
C      PARAMETER                  (KCOLY = YKCOL)
C
      INTEGER                    NINTY
C      PARAMETER                  (NINTY = YNINT)
C                                YNINT IS THE NUMBER OF SUBINTERVALS
C                                DEFINED BY THE SPATIAL MESH Y.
C
C      DOUBLE PRECISION           YCOL(YNCPTS)
      DOUBLE PRECISION           YCOL(*)
C                                THE SEQUENCE OF COLLOCATION POINTS ON
```

```
C                                THE INTERVAL [Y_A, Y_B].
C
C-----------------------------------------------------------------------
C
      COMMON /BURGER/            COEFF
      COMMON /YBSPLINE/          NINTY, KCOLY, YCOL(442)
      COMMON /ABDLU/             YABD(8804),
&                                IPIVOT(442), IABDTP, IABDBK, IABDBT
      COMMON /YBCOEFF/           YFBASIS(29172)

C-----------------------------------------------------------------------
C LOCAL VARIABLES:
C-----------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                    I
      INTEGER                    J
      INTEGER                    II
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C SUBROUTINES CALLED:
C                                BACOL
C                                VALUES
C                                TRUU
C-----------------------------------------------------------------------
C HAS TO DO THIS, BECAUSE OF THE USE OF COMMON BLOCK PASS DATA
C COULD NOT BE USED AS VARIABLE
      KCOLY = YKCOL
      NINTY = YNINT
C     SET THE POINTERS INTO THE FLOATING POINT WORK ARRAY.
      IABDTP = 1
      IABDBK = IABDTP + NCONTI
```

```
      IABDBT = IABDBK + YNINT*YKCOL*(YKCOL+NCONTI)

C-----------------------------------------------------------------------
C     INITIALIZE ABDBLK, THE TOP BLOCK AND THE BOTTOM BLOCK TO ZERO.
      DO 11 I = 1, YNINT*YKCOL*(YKCOL+NCONTI)+2*NCONTI
         YABD(I) = ZERO
   11 CONTINUE

      DO 12 I = 1, (YKCOL+NCONTI)*3*YNCPTS
         YFBASIS(I) = ZERO
   12 CONTINUE

C-----------------------------------------------------------------------
C     SET THE REMAINING INPUT PARAMETERS.
      T0 = 0.0D0
      TOUT = 1.0D+0
      ATOL(1) = 1.D-3
      ATOL(NPDE) = 1.D-3
      DO I = 2, NPDE-1
         ATOL(I) = 1.D-3
      END DO
      DO I = 1, NPDE
         RTOL(I) = ATOL(I)
      END DO
      COEFF = 2.5D-1
C
C     DEFINE THE MESH BASED ON A UNIFORM STEP SIZE.
      Y(1) = YA
      DO 10 I = 2, YNINT
         Y(I) = YA + ((I-1) * (YB - YA)) / YNINT
   10 CONTINUE
      Y(YNINT+1) = YB
```

```
      CALL MESHSQ(YKCOL, YNINT, Y, WORK1, YH,
     &            EXCOL, EWTS)

      CALL COLPNT(YKCOL, YNINT, YNCPTS, Y, YH,
     &            WORK22, YCOL,
     &            YBS)
C-----------------------------------------------------------------------
C     DISCRETIZE Y DOMAIN (USE PARAMETER TO PASS THE 1D ARRAY YFBASIS
C     TO 3D FBASIS, AND GET YABD(3 PARTS))

      CALL BSPLINECOEFF(YKCOL,YNINT,YCOL,YBS,YFBASIS)

C     LU DECOMPOSITION OF THE MATRIX.
      CALL CRDCMP(YNCPTS,YABD(IABDTP),1,2*1,YABD(IABDBK),YKCOL,
     &            (YKCOL+NCONTI),YNINT,YABD(IABDBT),1,IPIVOT,
     &            ICFLAG)
      IF (ICFLAG .NE. 0) GOTO 9999

C-----------------------------------------------------------------------
C     USE BACOL

C     DEFINE THE MESH BASED ON A UNIFORM STEP SIZE.
      X(1) = XA
      DO 20 I = 2, XNINT
         X(I) = XA + ((I-1) * (XB - XA)) / XNINT
   20 CONTINUE
      X(XNINT+1) = XB

C     INITIALIZE THE MFLAG VECTOR.
      DO 21 I = 1, 7
         MFLAG(I) = 0
```

```
   21 CONTINUE

      MFLAG(2) = 1

C     INITIALIZE IPAR AND RPAR.
      DO I = 1, LIP
         IPAR(I) = 0
      END DO
      DO I = 1, LRP
         RPAR(I) = 0.0D0
      END DO

C-----------------------------------------------------------------------
      OPEN(7, FILE='2DBURGER.TEXT', ACCESS='SEQUENTIAL',
     &     FORM='FORMATTED')
      WRITE(7,'(/A)') 'THE INPUT IS  '
      WRITE(7,'(/A, I3, A, I4, 2(A, E8.2))') 'XKCOL =', XKCOL,
     &  ', XNINT =', XNINT, ', ATOL(1) =',
     &  ATOL(1), ', RTOL(1) =', RTOL(1)
      WRITE(7,'(/A, E8.2)') 'TOUT = ', TOUT

      CALL BACOL(T0, TOUT, ATOL, RTOL, NPDE, XKCOL, NINTMX, XNINT, X,
     &           MFLAG, RPAR, LRP, IPAR, LIP, W, IDID)

C     CHECK FOR AN ERROR FROM BACOL.
      WRITE(7,'(/A, I5)') 'IDID =', IDID
      IF (IDID .LT. 2) GOTO 100

C     OUTPUT APPROXIMATE SOLUTION
      XOUT(1) = XA
      DO 30 I = 2, NXOUT-1
         XOUT(I) = XA + DBLE(I - 1) * (XB - XA)/DBLE(NXOUT-1)
```

```
   30 CONTINUE
      XOUT(NXOUT) = XB

      CALL VALUES(XKCOL, XOUT, XNINT, X, NPDE, NXOUT, 0,
     &            UOUT, W, VALWRK)
      WRITE(7,'(/A)') 'THE OUTPUT IS   '
      WRITE(7,'(/A, I3, A, I4)') 'XKCOL =', XKCOL, ', XNINT =', XNINT
      WRITE(7,'(/A, 4A)') '      XOUT     ', '      UOUT(1)     ',
     &                    '    UOUT(2)     ', '    UOUT(3)     ',
     &                    '    UOUT(4)'
      DO 40 I = 1, NXOUT

         II = (I - 1) * NPDE
C     REFORM UOUT INTO NPDE*NXOUT MATRIX
         DO 41 J = 1, NPDE
            UBAROUTM(J,I) = UOUT(II+J)
   41    CONTINUE

C    DO THE PROJECTION
C    98 FORMAT(E14.6, ADVANCE = NO)
   40 CONTINUE
      DO 42 I = 1, NXOUT
         CALL YEVAL1(KCOLY,NINTY,YFBASIS,UBAROUTM(1,I),
     &               UBAROUTM(1,I), UOUTM(1,I),TRY(1,I))
   42 CONTINUE

      DO 43 I = 1, NXOUT
      WRITE(7,'(13E14.6)') XOUT(I), (UOUTM(J,I),J=1, NPDE)
   43 CONTINUE

C     OUTPUT EXACT SOLUTION.
      WRITE(7,'(/A, 1A)') '        '
```

```
      WRITE(7,'(/A, 4A)') '     XOUT     ', '    EXACTU(1)    ',
     &                    '   EXACTU(2)   ', '   EXACTU(3)     ',
     &                    '   EXACTU(4)'

      DO 50 I = 1, NXOUT
         CALL TRUU(TOUT, XOUT(I), EXACTU, NPDE)
C     STORE EXACTU INTO NPDE*NXOUT MATRIX
         DO 51 J = 1, NPDE
            EXACTUM(J,I) = EXACTU(J)
   51    CONTINUE

         WRITE(7,'(13E14.6)') XOUT(I), (EXACTUM(J,I),J=1, NPDE)

   50 CONTINUE

C     COMPUTES 1-NORM OF THE ERROR MATRIX
      DO 60 I = 1, NXOUT
         DO 61 J = 1, NPDE
            ERRORM(J,I) = UOUTM(J,I)-EXACTUM(J,I)
   61    CONTINUE
   60 CONTINUE

C     OUTPUT THE ERROR MATRIX.
      WRITE(7,'(/A, 1A)') '         '
      WRITE(7,'(/A, 4A)') '     XOUT     ', '    ERRORM(1)    ',
     &                    '   ERRORM(2)   ', '   ERRORM(3)     ',
     &                    '   ERRORM(4)'

      DO 70 I = 1, NXOUT
         WRITE(7,'(13E14.6)') XOUT(I), (ERRORM(J,I),J=1, NPDE)
   70 CONTINUE
```

```
      CLOSE(7)

      GOTO 9999

  100 CONTINUE
      WRITE(6,'(A)')  'CANNOT PROCEED DUE TO ERROR FROM BACOL.'

 9999 STOP
      END
C-----------------------------------------------------------------------
CC-------END OF  MAIN-------------------------------------------------


      SUBROUTINE BSPLINECOEFF(YKCOL,YNINT,YCOL,YBS,FBASIS)
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
        DOUBLE PRECISION        ZERO
        PARAMETER               (ZERO = 0.0D0)
C
        DOUBLE PRECISION        NEGONE
        PARAMETER               (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
        INTEGER                 NCONTI
        PARAMETER               (NCONTI = 2)
C                               NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                               AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C       INPUT:
C       Y DIMENSION
```

```
      INTEGER              YKCOL
C                          YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                          TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                          WHICH IS EQUAL TO THE DEGREE OF THE
C                          PIECEWISE POLYNOMIALS MINUS ONE.
C                             1 < YKCOL < 11.
C       PARAMETER          (YKCOL = 2)
C       PARAMETER          (YKCOL = WORK2(IYKCOL))
C
      INTEGER              YNINT
C       PARAMETER          (YNINT = WORK2(IYNINT))
C       PARAMETER          (YNINT = 2)
C                          YNINT IS THE NUMBER OF SUBINTERVALS
C                          DEFINED BY THE SPATIAL MESH Y.
C-----------------------------------------------------------------------
      INTEGER              YNCPTS
      PARAMETER            (YNCPTS=(YKCOL*YNINT+NCONTI))
C                          YNCPTS IS THE NUMBER
C                          OF COLLOCATION POINTS.
C
      INTEGER              NPDE
C                          NUMBER OF PDES
C                          =THE NUMBER OF COLLOCATION POINTS
C                          IN Y DOMAIN
      PARAMETER            (NPDE = YKCOL*YNINT+NCONTI)
C
      INTEGER              YNELS
      PARAMETER            (YNELS=YKCOL*(YKCOL+NCONTI))
C                          THE NUMBER OF ELEMENTS IN ONE
C                          COLLOCATION BLOCK OF WORK.
C
C-----------------------------------------------------------------------
```

```
C       Y DIMENSION
C      INPUT:
        DOUBLE PRECISION        YCOL(YNCPTS)
C        DOUBLE PRECISION        YCOL(*)
C                                THE SEQUENCE OF COLLOCATION POINTS ON
C                                THE INTERVAL [Y_A, Y_B].
C
        DOUBLE PRECISION        YBS(YNCPTS+YKCOL+NCONTI)
C                                THE BREAKPOINT SEQUENCE.
C                                YBS(I)=X(1), I=1, KCOL+NCONTI;
C                                YBS((I-1)*KCOL+NCONTI+J)=Y(I),
C                                      I=2, NINT;   J=1, KCOL
C                               YBS(YNCPTS+I)=Y(NINT+1), I=1,KCOL+NCONTI.
C
C-----------------------------------------------------------------------
C
C       OUTPUT:
        DOUBLE PRECISION        FBASIS(YKCOL+NCONTI, 3, YNCPTS)
C                                BASIS FUNCTION VALUES AT THE COLLOCATION
C                                POINTS. FBASIS(K,J,I) CONTAINS THE
C                                VALUES OF THE (J-1)ST DERIVATIVE
C                                (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                                FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                                I-TH COLLOCATION POINT.
C-----------------------------------------------------------------------
C
C        INTEGER                 IPIVOT(NPDE)
        INTEGER                 IPIVOT(*)
C                                PIVOTING INFORMATION FROM THE
C                                FACTORIZATION OF THE TEMPORARY MATRIX.
C
C
```

```
C       DOUBLE PRECISION        YABD(YNINT*YNELS+2*NCONTI)
        DOUBLE PRECISION        YABD(*)
C                               ABD
C
        INTEGER                 IABDTP
C                               WORK(IABDTP) CONTAINS A COPY OF THE TOP
C                               BLOCK WHICH IS REQUIRED SINCE CRDCMP
C                               OVERWRITES THE INPUT COLLOCATION MATRIX.
C
        INTEGER                 IABDBK
C                               WORK(IABDBK) CONTAINS A COPY OF ABDBLK
C                               WHICH IS REQUIRED SINCE CRDCMP
C                               OVERWRITES THE INPUT COLLOCATION MATRIX.
C
        INTEGER                 IABDBT
C                               WORK(IABDBT) CONTAINS A COPY OF THE
C                               BOTTOM BLOCK WHICH IS REQUIRED SINCE
C                               CRDCMP OVERWRITES THE INPUT COLLOCATION
C                               MATRIX.
C-------------------------------------------------------------------------
C-------------------------------------------------------------------------
        COMMON /ABDLU/              YABD(8804),
&                                   IPIVOT(442), IABDTP, IABDBK, IABDBT

C-------------------------------------------------------------------------
C LOCAL VARIABLES:
        INTEGER                 ILEFT
C                               BREAKPOINT INFORMATION.
C
C-------------------------------------------------------------------------
C LOOP INDICES:
        INTEGER                 I
```

```
      INTEGER                 J
      INTEGER                 L
      INTEGER                 II
      INTEGER                 LL
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C SUBROUTINES CALLED:
C                                 BSPLVD
C
C-----------------------------------------------------------------------

C***END PROLOGUE  BSPLINECOEFF
C     BSPLVD IS CALLED TO COMPUTE THE COMPONENTS OF FBASIS(K,I,J)
C     ASSOCIATED THE FIRST COLLOCATION POINT. NOW ILEFT = KCOL + NCONTI.
      CALL BSPLVD(YBS,YKCOL+NCONTI,YCOL(1),YKCOL+NCONTI,FBASIS(1,1,1),3)

C     MAKEING USE OF THE FACT THAT ONLY THE FIRST BSPLINE HAS A NONZERO
C     VALUE AT THE LEFT END POINT, SET UP THE TOP BLOCK IN WORK.
      YABD(IABDTP) = FBASIS(1,1,1)

C-----------------------------------------------------------------------
C     THE NINT BLOCKS AT THE MIDDLE OF THE MATRIX WILL NOW BE SET UP.
      DO 80 I = 1, YNINT

C     MAKE USE OF THE FACT THAT THERE ARE KCOL COLLOCATION POINTS IN
C     EACH SUBINTERVAL TO FIND THE VALUE OF ILEFT.
         ILEFT = YKCOL + NCONTI + (I - 1) * YKCOL

         DO 70 J = 1, YKCOL

C     II IS THE POSITION IN YCOL OF THE J-TH COLLOCATION POINT OF THE
```

```
C       I-TH SUBINTERVAL.
C       II IS THE POSITION IN THE UBAR VECTOR WHERE THE VALUES FOR THE
C       RIGHT HAND SIDE OF THE INITIAL CONDITIONS, EVALUATED AT THE II-TH
C       COLLOCATION POINT ARE STORED.
              II = (I-1) * YKCOL + 1 + J
               CALL BSPLVD(YBS,YKCOL+NCONTI,YCOL(II),ILEFT,
     &                    FBASIS(1,1,II),3)
              DO 60 L = 1, YKCOL + NCONTI

C       GENERATE THE SUBBLOCK IN ABDBLK CORRESPONDING TO THE II-TH
C       COLLOCATION POINT.
C
                   LL = (L-1)*YKCOL + (I-1)*YNELS + (J-1)

                   YABD(IABDBK+LL) = FBASIS(L,1,II)

   60           CONTINUE
   70        CONTINUE
   80 CONTINUE

C-------------------------------------------------------------------------
C       NOW, SET UP THE BOTTOM BLOCK, USING THE FACT THAT ONLY THE
C       LAST BSPLINE BASIS FUNCTION IS NON-ZERO AT THE RIGHT END POINT.
C       SIMULTANEOUSLY, SET UP THE CORRESPONDING PART OF THE RIGHT HAND
C       SIDE.
C
       CALL BSPLVD(YBS,YKCOL+NCONTI,YCOL(YNCPTS),YNCPTS,
     &                    FBASIS(1,1,YNCPTS),3)

       YABD(IABDBT+1) = FBASIS(YKCOL+NCONTI,1,YNCPTS)

  998 RETURN
```

```
      END
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE BSPLINECOEFF---------------------------------

      SUBROUTINE DERIVF(T, X, UBAR, UBARX, UBARXX, DFDUBAR,
     &                  DFDUBARX, DFDUBARXX, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THIS SUBROUTINE IS USED TO DEFINE THE INFORMATION ABOUT THE
C       PDE REQUIRED TO FORM THE ANALYTIC JACOBIAN MATRIX FOR THE DAE
C       OR ODE SYSTEM. ASSUMING THE PDE IS OF THE FORM
C                          UT = F(T, X, UBAR, UBARX, UBARXX)
C       THIS ROUTINE RETURNS THE JACOBIANS D(F)/D(UBAR), D(F)/D(UBARX),
C       AND D(F)/D(UBARXX).
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
      DOUBLE PRECISION      ZERO
      PARAMETER             (ZERO = 0.0D0)
C
      DOUBLE PRECISION      NEGONE
      PARAMETER             (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
      INTEGER               NCONTI
      PARAMETER             (NCONTI = 2)
C                           NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                           AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
```

```
      INTEGER                 NPDE
C                             THE NUMBER OF PDES IN THE SYSTEM.
C
      DOUBLE PRECISION        T
C                             THE CURRENT TIME COORDINATE.
C
      DOUBLE PRECISION        X
C                             THE CURRENT SPATIAL COORDINATE.
C
      DOUBLE PRECISION        UBAR(NPDE)
C                             UBAR(1:NPDE) IS THE APPROXIMATION OF THE
C                             SOLUTION AT THE POINT (T,X).
C
      DOUBLE PRECISION        UBARX(NPDE)
C                             UBARX(1:NPDE) IS THE APPROXIMATION OF
C                             THE SPATIAL DERIVATIVE OF THE SOLUTION
C                             AT THE POINT (T,X).
C
      DOUBLE PRECISION        UBARXX(NPDE)
C                             UBARXX(1:NPDE) IS THE APPROXIMATION OF
C                             THE SECOND SPATIAL DERIVATIVE OF THE
C                             SOLUTION AT THE POINT (T,X).
C
C OUTPUT:
      DOUBLE PRECISION        DFDUBAR(NPDE,NPDE)
C                             DFDUBAR(I,J) IS THE PARTIAL DERIVATIVE
C                             OF THE I-TH COMPONENT OF THE VECTOR F
C                             WITH RESPECT TO THE J-TH COMPONENT
C                             OF THE UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION        DFDUBARX(NPDE,NPDE)
C                             DFDUBARX(I,J) IS THE PARTIAL DERIVATIVE
```

```
C                                   OF THE I-TH COMPONENT OF THE VECTOR F
C                                   WITH RESPECT TO THE J-TH COMPONENT
C                                   OF THE SPATIAL DERIVATIVE OF THE
C                                   UNKNOWN FUNCTION U.
C
      DOUBLE PRECISION              DFDUBARXX(NPDE,NPDE)
C                                   DFDUBARXX(I,J) IS THE PARTIAL DERIVATIVE
C                                   OF THE I-TH COMPONENT OF THE VECTOR F
C                                   WITH RESPECT TO THE J-TH COMPONENT
C                                   OF THE SECOND SPATIAL DERIVATIVE OF THE
C                                   UNKNOWN FUNCTION U.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C LOCAL:
      DOUBLE PRECISION              U(442)
C                                   UBAR IS THE APPROXIMATION OF THE
C                                   SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION              UX(442)
C                                   UBARX IS THE APPROXIMATION OF THE
C                                   SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                                   THE POINT (T,XA).
C
      DOUBLE PRECISION              UY(442)
C                                   UBARX IS THE APPROXIMATION OF THE
C                                   SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                                   THE POINT (T,XA).
C
C     LOCAL:
      INTEGER                       YNCPTS
C      PARAMETER                    (YNCPTS=(YKCOL*YNINT+NCONTI))
C                                   YNCPTS=(YKCOL*YNINT+NCONTI) IS THE
```

```
C                                NUMBER OF COLLOCATION POINTS.
C
      INTEGER                    ICPT
C                                THE INDEX OF THE COLLOCATION POINT.
C
C-----------------------------------------------------------------------
      DOUBLE PRECISION           YFBASIS(*)
C                                BASIS FUNCTION VALUES AT THE COLLOCATION
C                                POINTS. FBASIS(K,J,I) CONTAINS THE
C                                VALUES OF THE (J-1)ST DERIVATIVE
C                                (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                                FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                                I-TH COLLOCATION POINT.
C
      DOUBLE PRECISION           COEFF
C                                COEFF IS THE COEFFCIENT OF UXX IN THE
C                                BURGERS' EQUATION
C
C     INTEGER                     IPIVOT(NPDE)
      INTEGER                    IPIVOT(*)
C                                PIVOTING INFORMATION FROM THE
C                                FACTORIZATION OF THE TEMPORARY MATRIX.
C
C
C     DOUBLE PRECISION            YABD(YNINT*YNELS+2*NCONTI)
      DOUBLE PRECISION           YABD(*)
C                                ABD
C
      INTEGER                    IABDTP
C                                WORK(IABDTP) CONTAINS A COPY OF THE TOP
C                                BLOCK WHICH IS REQUIRED SINCE CRDCMP
C                                OVERWRITES THE INPUT COLLOCATION MATRIX.
```

```
C
      INTEGER                 IABDBK
C                             WORK(IABDBK) CONTAINS A COPY OF ABDBLK
C                             WHICH IS REQUIRED SINCE CRDCMP
C                             OVERWRITES THE INPUT COLLOCATION MATRIX.
C
      INTEGER                 IABDBT
C                             WORK(IABDBT) CONTAINS A COPY OF THE
C                             BOTTOM BLOCK WHICH IS REQUIRED SINCE
C                             CRDCMP OVERWRITES THE INPUT COLLOCATION
C                             MATRIX.
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER                 KCOLY
C                             YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                             TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                             WHICH IS EQUAL TO THE DEGREE OF THE
C                             PIECEWISE POLYNOMIALS MINUS ONE.
C                                1 < YKCOL < 11.
C      PARAMETER                (KCOLY = YKCOL)
C
      INTEGER                 NINTY
C      PARAMETER                (NINTY = YNINT)
C                             YNINT IS THE NUMBER OF SUBINTERVALS
C                             DEFINED BY THE SPATIAL MESH Y.
C
C      DOUBLE PRECISION         YCOL(YNCPTS)
      DOUBLE PRECISION        YCOL(*)
C                             THE SEQUENCE OF COLLOCATION POINTS ON
C                             THE INTERVAL [Y_A, Y_B].
C
C-----------------------------------------------------------------------
```

```
C
      COMMON /BURGER/          COEFF
      COMMON /YBSPLINE/        NINTY, KCOLY, YCOL(442)
      COMMON /ABDLU/           YABD(8804),
&                              IPIVOT(442), IABDTP, IABDBK, IABDBT
      COMMON /YBCOEFF/         YFBASIS(29172)

C-------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                  I
      INTEGER                  J
      INTEGER                  K
      INTEGER                  IJ
      INTEGER                  M
C-------------------------------------------------------------------
C***END PROLOGUE  DERIVF

      YNCPTS=KCOLY*NINTY+NCONTI

      DO 10 I = 1, YNCPTS
         DO 11 J = 1,YNCPTS
            DFDUBAR(I,J) = ZERO
            DFDUBARX(I,J) = ZERO
            DFDUBARXX(I,J) = ZERO
   11    CONTINUE
   10 CONTINUE

C     CALCULATE U UX
      CALL YEVAL3(KCOLY,NINTY,YFBASIS,UBAR,UBARX, U,UX,UY)
C
      ICPT = 1
      K = 0
```

```
C       PIONTER
        IJ = (ICPT-1)*(KCOLY+NCONTI)*3
        DO 12 M = 1,KCOLY+NCONTI
           DFDUBAR(ICPT,K+M) = COEFF*YFBASIS(IJ+M+2*(KCOLY+NCONTI))
     &             - YFBASIS(IJ+M)*UX(ICPT)
     &             - YFBASIS(IJ+M)*UY(ICPT)
     &             - U(ICPT)*YFBASIS(IJ+M+(KCOLY+NCONTI))

           DFDUBARX(ICPT,K+M) = - U(ICPT)*YFBASIS(IJ+M)

           DFDUBARXX(ICPT,K+M) = COEFF*YFBASIS(IJ+M)
    12 CONTINUE

C       ICPT = 2, ... , YNCPTS-1
        DO 20 I = 1,NINTY
           K = (I-1)*KCOLY

           DO 30 J = 1,KCOLY
              ICPT = K + J + 1
C       PIONTER
              IJ = (ICPT-1)*(KCOLY+NCONTI)*3

              DO 40 M = 1, KCOLY + NCONTI
                 DFDUBAR(ICPT,K+M) = COEFF*YFBASIS(IJ+M+2*(KCOLY+NCONTI))
     &                   - YFBASIS(IJ+M)*UX(ICPT)
     &                   - YFBASIS(IJ+M)*UY(ICPT)
     &                   - U(ICPT)*YFBASIS(IJ+M+(KCOLY+NCONTI))

                 DFDUBARX(ICPT,K+M) = - U(ICPT)*YFBASIS(IJ+M)

                 DFDUBARXX(ICPT,K+M) = COEFF*YFBASIS(IJ+M)
    40        CONTINUE
```

```
   30     CONTINUE
   20 CONTINUE

      ICPT = YNCPTS
      K = (NINTY-1)*KCOLY
C     PIONTER
      IJ = (ICPT-1)*(KCOLY+NCONTI)*3
      DO 50 M = 1,KCOLY+NCONTI
         DFDUBAR(ICPT,K+M) = COEFF*YFBASIS(IJ+M+2*(KCOLY+NCONTI))
     &             - YFBASIS(IJ+M)*UX(ICPT)
     &             - YFBASIS(IJ+M)*UY(ICPT)
     &             - U(ICPT)*YFBASIS(IJ+M+(KCOLY+NCONTI))

         DFDUBARX(ICPT,K+M) = - U(ICPT)*YFBASIS(IJ+M)

         DFDUBARXX(ICPT,K+M) = COEFF*YFBASIS(IJ+M)
   50 CONTINUE

C-----------------------------------------------------------------------
C     SOLVE THE LINEAR SYSTEM.
      DO 60 I = 1, NPDE
         CALL CRSLVE(YABD(IABDTP),1,2*1,YABD(IABDBK),KCOLY*1,
     &                (KCOLY+NCONTI)*1,NINTY,YABD(IABDBT),1,
     &                IPIVOT,DFDUBAR(1,I),0)
       IF (ICFLAG .NE. 0) GOTO 999

         CALL CRSLVE(YABD(IABDTP),1,2*1,YABD(IABDBK),KCOLY*1,
     &                (KCOLY+NCONTI)*1,NINTY,YABD(IABDBT),1,
     &                IPIVOT,DFDUBARX(1,I),0)
       IF (ICFLAG .NE. 0) GOTO 999

         CALL CRSLVE(YABD(IABDTP),1,2*1,YABD(IABDBK),KCOLY*1,
```

```
     &                    (KCOLY+NCONTI)*1,NINTY,YABD(IABDBT),1,
     &                    IPIVOT,DFDUBARXX(1,I),0)
      IF (ICFLAG .NE. 0) GOTO 999
   60 CONTINUE
  999 RETURN
      END
C-------END OF SUBROUTINE DERIVF-----------------------------------------------

      SUBROUTINE DIFBXA(T, UBAR, UBARX, DBDUBAR, DBDUBARX, DBDT, NPDE)
C-------------------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE DIFFERENTIATED BOUNDARY
C       CONDITIONS AT THE LEFT SPATIAL END POINT X = XA. FOR THE
C       BOUNDARY CONDITION EQUATION
C                                   B(T, UBAR, UBARX) = 0
C       THE PARTIAL DERIVATIVES DB/DUBAR, DB/DUBARX, AND DB/DT ARE
C       SUPPLIED BY THIS ROUTINE.
C
C-------------------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-------------------------------------------------------------------------------
C CONSTANTS:
        DOUBLE PRECISION        ZERO
        PARAMETER               (ZERO = 0.0D0)
C
        DOUBLE PRECISION        NEGONE
        PARAMETER               (NEGONE = -1.0D0)
C
C-------------------------------------------------------------------------------
        INTEGER                 NCONTI
        PARAMETER               (NCONTI = 2)
C                               NCONTI CONTINUITY CONDITIONS ARE IMPOSED
```

```
C                                   AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
        INTEGER                     NPDE
C                                   THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION            T
C                                   THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION            UBAR(NPDE)
C                                   UBAR(1:NPDE) IS THE APPROXIMATION OF THE
C                                   SOLUTION AT THE POINT (T,X).
C
        DOUBLE PRECISION            UBARX(NPDE)
C                                   UBARX(1:NPDE) IS THE APPROXIMATION OF
C                                   THE SPATIAL DERIVATIVE OF THE SOLUTION
C                                   AT THE POINT (T,X).
C
C OUTPUT:
        DOUBLE PRECISION            DBDUBAR(NPDE,NPDE)
C                                   DBDUBAR(I,J) IS THE PARTIAL DERIVATIVE
C                                   OF THE I-TH COMPONENT OF THE VECTOR B
C                                   WITH RESPECT TO THE J-TH COMPONENT
C                                   OF THE UNKNOWN FUNCTION UBAR.
C
        DOUBLE PRECISION            DBDUBARX(NPDE,NPDE)
C                                   DBDUBARX(I,J) IS THE PARTIAL DERIVATIVE
C                                   OF THE I-TH COMPONENT OF THE VECTOR B
C                                   WITH RESPECT TO THE J-TH COMPONENT
C                                   OF THE SPATIAL DERIVATIVE OF THE
C                                   UNKNOWN FUNCTION UBAR.
C
```

```
      DOUBLE PRECISION      DBDT(NPDE)
C                           DBDT(I) IS THE PARTIAL DERIVATIVE
C                           OF THE I-TH COMPONENT OF THE VECTOR B
C                           WITH RESPECT TO TIME T.
C-----------------------------------------------------------------
C LOCAL:
      DOUBLE PRECISION      U(442)
C                           UBAR IS THE APPROXIMATION OF THE
C                           SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION      UX(442)
C                           UBARX IS THE APPROXIMATION OF THE
C                           SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                           THE POINT (T,XA).
C
C     LOCAL:
      INTEGER               YNCPTS
C      PARAMETER            (YNCPTS=(YKCOL*YNINT+NCONTI))
C                           YNCPTS=(YKCOL*YNINT+NCONTI) IS THE
C                           NUMBER OF COLLOCATION POINTS.
C
      INTEGER               ICPT
C                           THE INDEX OF THE COLLOCATION POINT.
C
      DOUBLE PRECISION      TEMP
C                           UBAR IS THE APPROXIMATION OF THE
C                           SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION      ALPHA
C                           UBARX IS THE APPROXIMATION OF THE
C                           SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                           THE POINT (T,XA).
```

```
C-----------------------------------------------------------------------
      DOUBLE PRECISION      YFBASIS(*)
C                           BASIS FUNCTION VALUES AT THE COLLOCATION
C                           POINTS. FBASIS(K,J,I) CONTAINS THE
C                           VALUES OF THE (J-1)ST DERIVATIVE
C                           (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                           FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                           I-TH COLLOCATION POINT.
C
      DOUBLE PRECISION      COEFF
C                           COEFF IS THE COEFFCIENT OF UXX IN THE
C                           BURGERS' EQUATION
C
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER               KCOLY
C                           YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                           TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                           WHICH IS EQUAL TO THE DEGREE OF THE
C                           PIECEWISE POLYNOMIALS MINUS ONE.
C                              1 < YKCOL < 11.
C      PARAMETER              (KCOLY = YKCOL)
C
      INTEGER               NINTY
C      PARAMETER              (NINTY = YNINT)
C                           YNINT IS THE NUMBER OF SUBINTERVALS
C                           DEFINED BY THE SPATIAL MESH Y.
C
C      DOUBLE PRECISION       YCOL(YNCPTS)
      DOUBLE PRECISION      YCOL(*)
C                           THE SEQUENCE OF COLLOCATION POINTS ON
C                           THE INTERVAL [Y_A, Y_B].
```

```
C
C-----------------------------------------------------------------------
C
      COMMON /BURGER/          COEFF
      COMMON /YBSPLINE/        NINTY, KCOLY, YCOL(442)
      COMMON /YBCOEFF/         YFBASIS(29172)

C-----------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                  I
      INTEGER                  J
      INTEGER                  K
      INTEGER                  IJ
      INTEGER                  M
C-----------------------------------------------------------------------
C***END PROLOGUE  DIFBXA
      YNCPTS=KCOLY*NINTY+NCONTI

      DO 10 I = 1, YNCPTS
         DO 11 J = 1,YNCPTS
            DBDUBAR(I,J) = ZERO
            DBDUBARX(I,J) = ZERO
   11    CONTINUE
         DBDT(I) = ZERO
   10 CONTINUE

C     CALCULATE U UX
      CALL YEVAL1(KCOLY,NINTY,YFBASIS,UBAR,UBARX, U,UX)
C
      ICPT = 1
      TEMP = DEXP((YCOL(ICPT)-T) / (2.0D0*COEFF))
      ALPHA = 1.0D0 + TEMP
```

```
      K = 0
C     PIONTER
      IJ = (ICPT-1)*(KCOLY+NCONTI)*3
      DO 12 M = 1,KCOLY+NCONTI
         DBDUBAR(ICPT,K+M) = ALPHA*YFBASIS(IJ+M)
   12 CONTINUE

      DBDT(ICPT) = 1.0D0/(2.0D0*COEFF)*TEMP*U(ICPT)

C     ICPT = 2, ... , YNCPTS-1
      DO 20 I = 1,NINTY
         K = (I-1)*KCOLY

         DO 30 J = 1,KCOLY
            ICPT = K + J + 1

            TEMP = DEXP((YCOL(ICPT)-T) / (2.0D0*COEFF))
            ALPHA = 1.0D0 + TEMP
C     PIONTER
            IJ = (ICPT-1)*(KCOLY+NCONTI)*3

            DO 40 M = 1, KCOLY + NCONTI
               DBDUBAR(ICPT,K+M) = ALPHA*YFBASIS(IJ+M)
   40       CONTINUE

            DBDT(ICPT) = 1.0D0/(2.0D0*COEFF)*TEMP*U(ICPT)
   30    CONTINUE
   20 CONTINUE

      ICPT = YNCPTS
      TEMP = DEXP((YCOL(ICPT)-T) / (2.0D0*COEFF))
      ALPHA = 1.0D0 + TEMP
```

```
      K = (NINTY-1)*KCOLY
C     PIONTER
      IJ = (ICPT-1)*(KCOLY+NCONTI)*3
      DO 50 M = 1,KCOLY+NCONTI
         DBDUBAR(ICPT,K+M) = ALPHA*YFBASIS(IJ+M)
   50 CONTINUE

      DBDT(ICPT) = 1.0D0/(2.0D0*COEFF)*TEMP*U(ICPT)

      RETURN
      END
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE DIFBXA----------------------------------------
      SUBROUTINE DIFBXB(T, UBAR, UBARX, DBDUBAR, DBDUBARX, DBDT, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE DIFFERENTIATED BOUNDARY
C       CONDITIONS AT THE RIGHT SPATIAL END POINT X = XB. FOR THE
C       BOUNDARY CONDITION EQUATION
C                                  B(T, UBAR, UBARX) = 0
C       THE PARTIAL DERIVATIVES DB/DUBAR, DB/DUBARX, AND DB/DT ARE
C       SUPPLIED BY THIS ROUTINE.
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
      DOUBLE PRECISION      ZERO
      PARAMETER             (ZERO = 0.0D0)
C
      DOUBLE PRECISION      NEGONE
      PARAMETER             (NEGONE = -1.0D0)
```

```
C
C-----------------------------------------------------------------------
      INTEGER                NCONTI
      PARAMETER              (NCONTI = 2)
C                            NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                            AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
      INTEGER                NPDE
C                            THE NUMBER OF PDES IN THE SYSTEM.
C
      DOUBLE PRECISION       T
C                            THE CURRENT TIME COORDINATE.
C
      DOUBLE PRECISION       UBAR(NPDE)
C                            UBAR(1:NPDE) IS THE APPROXIMATION OF THE
C                            SOLUTION AT THE POINT (T,X).
C
      DOUBLE PRECISION       UBARX(NPDE)
C                            UBARX(1:NPDE) IS THE APPROXIMATION OF THE
C                            SPATIAL DERIVATIVE OF THE SOLUTION AT
C                            THE POINT (T,X).
C
C OUTPUT:
      DOUBLE PRECISION       DBDUBAR(NPDE,NPDE)
C                            DBDUBAR(I,J) IS THE PARTIAL DERIVATIVE
C                            OF THE I-TH COMPONENT OF THE VECTOR B
C                            WITH RESPECT TO THE J-TH COMPONENT
C                            OF THE UNKNOWN FUNCTION UBAR.
C
      DOUBLE PRECISION       DBDUBARX(NPDE,NPDE)
C                            DBDUBARX(I,J) IS THE PARTIAL DERIVATIVE
```

```
C                                   OF THE I-TH COMPONENT OF THE VECTOR B
C                                   WITH RESPECT TO THE J-TH COMPONENT
C                                   OF THE SPATIAL DERIVATIVE OF THE
C                                   UNKNOWN FUNCTION UBAR.
C
      DOUBLE PRECISION              DBDT(NPDE)
C                                   DBDT(I) IS THE PARTIAL DERIVATIVE
C                                   OF THE I-TH COMPONENT OF THE VECTOR B
C                                   WITH RESPECT TO TIME T.
C-----------------------------------------------------------------------
C LOCAL:
C  YCOL(YNCPTS)   YNCPTS = YNINT*(YKCOL+NCONTI)+NCONTI
C  YCOL( 442   )  YNCPTS = 20    *(    20   +   2    )+2)
      DOUBLE PRECISION              U(442)
C                                   UBAR IS THE APPROXIMATION OF THE
C                                   SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION              UX(442)
C                                   UBARX IS THE APPROXIMATION OF THE
C                                   SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                                   THE POINT (T,XA).
C
C     LOCAL:
      INTEGER                       YNCPTS
C      PARAMETER                    (YNCPTS=(YKCOL*YNINT+NCONTI))
C                                  YNCPTS=(YKCOL*YNINT+NCONTI) IS THE NUMBER
C                                   OF COLLOCATION POINTS.
C
      INTEGER                       ICPT
C                                   THE INDEX OF THE COLLOCATION POINT.
C
      DOUBLE PRECISION              TEMP
```

```
C                                   UBAR IS THE APPROXIMATION OF THE
C                                   SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION              ALPHA
C                                   UBARX IS THE APPROXIMATION OF THE
C                                   SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                                   THE POINT (T,XA).
C-----------------------------------------------------------------------
      DOUBLE PRECISION              YFBASIS(*)
C                                   BASIS FUNCTION VALUES AT THE COLLOCATION
C                                   POINTS. FBASIS(K,J,I) CONTAINS THE
C                                   VALUES OF THE (J-1)ST DERIVATIVE
C                                   (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                                   FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                                   I-TH COLLOCATION POINT.
C
      DOUBLE PRECISION              COEFF
C                                   COEFF IS THE COEFFCIENT OF UXX IN THE
C                                   BURGERS' EQUATION
C
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER                       KCOLY
C                                   YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                                   TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                                   WHICH IS EQUAL TO THE DEGREE OF THE
C                                   PIECEWISE POLYNOMIALS MINUS ONE.
C                                      1 < YKCOL < 11.
C
      INTEGER                       NINTY
C      PARAMETER                    (NINTY = YNINT)
C                                   YNINT IS THE NUMBER OF SUBINTERVALS
```

```
C                                   DEFINED BY THE SPATIAL MESH Y.
C
      DOUBLE PRECISION              YCOL(*)
C                                   THE SEQUENCE OF COLLOCATION POINTS ON
C                                   THE INTERVAL [Y_A, Y_B].
C
C-----------------------------------------------------------------------
C
      COMMON /BURGER/               COEFF

      COMMON /YBSPLINE/             NINTY, KCOLY, YCOL(442)

      COMMON /YBCOEFF/              YFBASIS(29172)

C-----------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                       I
      INTEGER                       J
      INTEGER                       K
      INTEGER                       IJ
      INTEGER                       M
C-----------------------------------------------------------------------
C***END PROLOGUE  DIFBXB
                .
      YNCPTS=KCOLY*NINTY+NCONTI

      DO 10 I = 1, YNCPTS
         DO 11 J = 1,YNCPTS
            DBDUBAR(I,J) = ZERO
            DBDUBARX(I,J) = ZERO
   11    CONTINUE
         DBDT(I) = ZERO
```

```
   10 CONTINUE

C     CALCULATE U UX
      CALL YEVAL1(KCOLY,NINTY,YFBASIS,UBAR,UBARX,  U,UX)
C
      ICPT = 1
      TEMP = DEXP((1.0D0+YCOL(ICPT)-T) / (2.0D0*COEFF))
      ALPHA = 1.0D0 + TEMP
      K = 0
C     PIONTER
      IJ = (ICPT-1)*(KCOLY+NCONTI)*3
      DO 12 M = 1,KCOLY+NCONTI
         DBDUBAR(ICPT,K+M) = ALPHA*YFBASIS(IJ+M)
   12 CONTINUE

      DBDT(ICPT) = 1.0D0/(2.0D0*COEFF)*TEMP*U(ICPT)

C     ICPT = 2, ... , YNCPTS-1
      DO 20 I = 1,NINTY
         K = (I-1)*KCOLY

         DO 30 J = 1,KCOLY
            ICPT = K + J + 1

            TEMP = DEXP((1.0D0+YCOL(ICPT)-T) / (2.0D0*COEFF))
            ALPHA = 1.0D0 + TEMP
C     PIONTER
            IJ = (ICPT-1)*(KCOLY+NCONTI)*3

            DO 40 M = 1, KCOLY + NCONTI
               DBDUBAR(ICPT,K+M) = ALPHA*YFBASIS(IJ+M)
   40       CONTINUE
```

```
            DBDT(ICPT) = 1.0D0/(2.0D0*COEFF)*TEMP*U(ICPT)
   30     CONTINUE
   20 CONTINUE

      ICPT = YNCPTS
      TEMP = DEXP((1.0D0+YCOL(ICPT)-T) / (2.0D0*COEFF))
      ALPHA = 1.0D0 + TEMP
      K = (NINTY-1)*KCOLY
C     PIONTER
      IJ = (ICPT-1)*(KCOLY+NCONTI)*3
      DO 50 M = 1,KCOLY+NCONTI
         DBDUBAR(ICPT,K+M) = ALPHA*YFBASIS(IJ+M)
   50 CONTINUE

      DBDT(ICPT) = 1.0D0/(2.0D0*COEFF)*TEMP*U(ICPT)

      RETURN
      END
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE DIFBXB----------------------------------------
      SUBROUTINE BNDXA(T, UBAR, UBARX, BVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE BOUNDARY CONDITIONS AT THE
C       LEFT SPATIAL END POINT X = XA.
C                              B(T, UBAR, UBARX) = 0
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
```

```
      DOUBLE PRECISION      ZERO
      PARAMETER             (ZERO = 0.0D0)
C
      DOUBLE PRECISION      NEGONE
      PARAMETER             (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
      INTEGER               NCONTI
      PARAMETER             (NCONTI = 2)
C                           NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                           AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
      INTEGER               NPDE
C                           YNCPTS THE NUMBER OF PDES IN THE SYSTEM.
C
      DOUBLE PRECISION      T
C                           THE CURRENT TIME COORDINATE.
C
      DOUBLE PRECISION      UBAR(NPDE)
C                           UBAR(1:NPDE) IS THE APPROXIMATION OF THE
C                           SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION      UBARX(NPDE)
C                          UBARX(1:NPDE) IS THE APPROXIMATION OF THE
C                           SPATIAL DERIVATIVE OF THE SOLUTION AT
C                           THE POINT (T,XA).
C
C OUTPUT:
      DOUBLE PRECISION      BVAL(NPDE)
C                           BVAL(1:NPDE) IS THE BOUNDARY CONTIDITION
C                           AT THE LEFT BOUNDARY POINT.
```

```
C LOCAL:
      DOUBLE PRECISION        U(442)
C                             UBAR IS THE APPROXIMATION OF THE
C                             SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION        UX(442)
C                             UBARX IS THE APPROXIMATION OF THE
C                             SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                             THE POINT (T,XA).
C-----------------------------------------------------------------------
C
      DOUBLE PRECISION        YFBASIS(*)
C                             BASIS FUNCTION VALUES AT THE COLLOCATION
C                             POINTS. FBASIS(K,J,I) CONTAINS THE
C                             VALUES OF THE (J-1)ST DERIVATIVE
C                             (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                             FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                             I-TH COLLOCATION POINT.
C
      DOUBLE PRECISION        COEFF
C                             COEFF IS THE COEFFCIENT OF UXX IN THE
C                             BURGERS' EQUATION
C
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER                 KCOLY
C                             YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                             TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                             WHICH IS EQUAL TO THE DEGREE OF THE
C                             PIECEWISE POLYNOMIALS MINUS ONE.
C                                1 < YKCOL < 11.
C      PARAMETER                (KCOLY = YKCOL)
```

```
C
      INTEGER                  NINTY
C      PARAMETER               (NINTY = YNINT)
C                              YNINT IS THE NUMBER OF SUBINTERVALS
C                              DEFINED BY THE SPATIAL MESH Y.
C
C      DOUBLE PRECISION         YCOL(YNCPTS)
      DOUBLE PRECISION         YCOL(*)
C                              THE SEQUENCE OF COLLOCATION POINTS ON
C                              THE INTERVAL [Y_A, Y_B].
C
C-----------------------------------------------------------------------
C
      COMMON /BURGER/          COEFF
      COMMON /YBSPLINE/        NINTY, KCOLY, YCOL(442)
      COMMON /YBCOEFF/         YFBASIS(29172)
C-----------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                  I
C
C-----------------------------------------------------------------------
C***END PROLOGUE  BNDXA
C
      CALL YEVAL1(KCOLY,NINTY,YFBASIS,UBAR,UBARX, U,UX)

      DO 10 I = 1, NPDE
         BVAL(I) = (1.0D0 + DEXP((YCOL(I)-T) / (2.0D0*COEFF))) * U(I)
     &             - 1.0D0
   10 CONTINUE

      RETURN
      END
```

```
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE BNDXA-----------------------------------------
C-----------------------------------------------------------------------
      SUBROUTINE BNDXB(T, UBAR, UBARX, BVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE BOUNDARY CONDITIONS AT THE
C       RIGHT SPATIAL END POINT X = XB.
C                              B(T, UBAR, UBARX) = 0
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
        DOUBLE PRECISION        ZERO
        PARAMETER               (ZERO = 0.0D0)
C
        DOUBLE PRECISION        NEGONE
        PARAMETER               (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
        INTEGER                 NCONTI
        PARAMETER               (NCONTI = 2)
C                               NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                               AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
        INTEGER                 NPDE
C                               YNCPTS THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
```

```
C
      DOUBLE PRECISION      UBAR(NPDE)
C                           UBAR(1:NPDE) IS THE APPROXIMATION OF THE
C                           SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION      UBARX(NPDE)
C                          UBARX(1:NPDE) IS THE APPROXIMATION OF THE
C                           SPATIAL DERIVATIVE OF THE SOLUTION AT
C                           THE POINT (T,XA).
C
C OUTPUT:
      DOUBLE PRECISION      BVAL(NPDE)
C                           BVAL(1:NPDE) IS THE BOUNDARY CONTIDITION
C                           AT THE LEFT BOUNDARY POINT.
C LOCAL:
      DOUBLE PRECISION      U(442)
C                           UBAR IS THE APPROXIMATION OF THE
C                           SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION      UX(442)
C                           UBARX IS THE APPROXIMATION OF THE
C                           SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                           THE POINT (T,XA).
C-------------------------------------------------------------------
C
      DOUBLE PRECISION      YFBASIS(*)
C                           BASIS FUNCTION VALUES AT THE COLLOCATION
C                           POINTS. FBASIS(K,J,I) CONTAINS THE
C                           VALUES OF THE (J-1)ST DERIVATIVE
C                           (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                           FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                           I-TH COLLOCATION POINT.
```

```
C
      DOUBLE PRECISION        COEFF
C                             COEFF IS THE COEFFCIENT OF UXX IN THE
C                             BURGERS' EQUATION
C
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER                 KCOLY
C                             YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                             TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                             WHICH IS EQUAL TO THE DEGREE OF THE
C                             PIECEWISE POLYNOMIALS MINUS ONE.
C                                1 < YKCOL < 11.
C      PARAMETER                (KCOLY = YKCOL)
C
      INTEGER                 NINTY
C      PARAMETER                (NINTY = YNINT)
C                             YNINT IS THE NUMBER OF SUBINTERVALS
C                             DEFINED BY THE SPATIAL MESH Y.
C
C      DOUBLE PRECISION         YCOL(YNCPTS)
      DOUBLE PRECISION        YCOL(*)
C                             THE SEQUENCE OF COLLOCATION POINTS ON
C                             THE INTERVAL [Y_A, Y_B].
C
C-----------------------------------------------------------------------
C
      COMMON /BURGER/           COEFF
      COMMON /YBSPLINE/         NINTY, KCOLY, YCOL(442)
      COMMON /YBCOEFF/          YFBASIS(29172)
C-----------------------------------------------------------------------
C LOOP INDICES:
```

```
      INTEGER                  I
C
C-----------------------------------------------------------------------
C***END PROLOGUE  BNDXB
C
      CALL YEVAL1(KCOLY,NINTY,YFBASIS,UBAR,UBARX, U,UX)
C-----------------------------------------------------------------------
C     THE SEQUENCE OF COLLOCATION POINTS ON
C     THE INTERVAL [Y_A, Y_B].
      DO 10 I = 1, NPDE
         BVAL(I) = (1.0D0 + DEXP((1.0D0+YCOL(I)-T) / (2.0D0*COEFF)))
     &             * U(I) - 1.0D0
   10 CONTINUE

      RETURN
      END
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE BNDXB-----------------------------------------
      SUBROUTINE F(T, X, UBAR, UBARX, UBARXX, FVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THIS SUBROUTINE DEFINES THE RIGHT HAND SIDE VECTOR OF THE
C       NPDE DIMENSIONAL PARABOLIC PARTIAL DIFFERENTIAL EQUATION
C                          UT = F(T, X, UBAR, UBARX, UBARXX).
C                          UT = F(T, X, U, UX, UXX).
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
      DOUBLE PRECISION         ZERO
      PARAMETER                (ZERO = 0.0D0)
```

```
C
      DOUBLE PRECISION      NEGONE
      PARAMETER             (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
      INTEGER               NCONTI
      PARAMETER             (NCONTI = 2)
C                           NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                           AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
      INTEGER               NPDE
C                           THE NUMBER OF PDES IN THE SYSTEM.
C
      DOUBLE PRECISION      T
C                           THE CURRENT TIME COORDINATE.
C
      DOUBLE PRECISION      X
C                           THE CURRENT SPATIAL COORDINATE.
C
      DOUBLE PRECISION      UBAR(NPDE)
C                           U(1:NPDE) IS THE APPROXIMATION OF THE
C                           SOLUTION AT THE POINT (T,X).
C
      DOUBLE PRECISION      UBARX(NPDE)
C                           UX(1:NPDE) IS THE APPROXIMATION OF THE
C                           SPATIAL DERIVATIVE OF THE SOLUTION AT
C                           THE POINT (T,X).
C
      DOUBLE PRECISION      UBARXX(NPDE)
C                           UXX(1:NPDE) IS THE APPROXIMATION OF THE
C                           SECOND SPATIAL DERIVATIVE OF THE
```

```
C                                   SOLUTION AT THE POINT (T,X).
C
C OUTPUT:
      DOUBLE PRECISION              FVAL(NPDE)
C                                   FVAL(1:NPDE) IS THE RIGHT HAND SIDE
C                                   VECTOR F(T, X, U, UX, UXX) OF THE PDE.
C-----------------------------------------------------------------------
C LOCAL:
      DOUBLE PRECISION              U(442)
C                                   UBAR IS THE APPROXIMATION OF THE
C                                   SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION              UX(442)
C                                   UBARX IS THE APPROXIMATION OF THE
C                                   SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                                   THE POINT (T,XA).
C
      DOUBLE PRECISION              UXX(442)
C                                   UBAR IS THE APPROXIMATION OF THE
C                                   SOLUTION AT THE POINT (T,XA).
C
      DOUBLE PRECISION              UY(442)
C                                   UBARX IS THE APPROXIMATION OF THE
C                                   SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                                   THE POINT (T,XA).
C
      DOUBLE PRECISION              UYY(442)
C                                   UBARX IS THE APPROXIMATION OF THE
C                                   SPATIAL DERIVATIVE X OF THE SOLUTION AT
C                                   THE POINT (T,XA).
C
C-----------------------------------------------------------------------
```

```
      DOUBLE PRECISION      YFBASIS(*)
C                           BASIS FUNCTION VALUES AT THE COLLOCATION
C                           POINTS. FBASIS(K,J,I) CONTAINS THE
C                           VALUES OF THE (J-1)ST DERIVATIVE
C                           (J=1,2,3) OF THE K-TH NON-ZERO BASIS
C                           FUNCTION (K=1,...,KCOL+NCONTI) AT THE
C                           I-TH COLLOCATION POINT.
C
      DOUBLE PRECISION      COEFF
C                           COEFF IS THE COEFFCIENT OF UXX IN THE
C                           BURGERS' EQUATION
C
C      INTEGER               IPIVOT(NPDE)
      INTEGER               IPIVOT(*)
C                           PIVOTING INFORMATION FROM THE
C                           FACTORIZATION OF THE TEMPORARY MATRIX.
C
C
      DOUBLE PRECISION      YABD(*)
C                           ABD
C
      INTEGER               IABDTP
C                           WORK(IABDTP) CONTAINS A COPY OF THE TOP
C                           BLOCK WHICH IS REQUIRED SINCE CRDCMP
C                           OVERWRITES THE INPUT COLLOCATION MATRIX.
C
      INTEGER               IABDBK
C                           WORK(IABDBK) CONTAINS A COPY OF ABDBLK
C                           WHICH IS REQUIRED SINCE CRDCMP
C                           OVERWRITES THE INPUT COLLOCATION MATRIX.
C
      INTEGER               IABDBT
```

```
C                                WORK(IABDBT) CONTAINS A COPY OF THE
C                                BOTTOM BLOCK WHICH IS REQUIRED SINCE
C                                CRDCMP OVERWRITES THE INPUT COLLOCATION
C                                MATRIX.
C-----------------------------------------------------------------------
C       Y DIMENSION
        INTEGER                  KCOLY
C                                YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                                TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                                WHICH IS EQUAL TO THE DEGREE OF THE E
C                                PIECEWIS POLYNOMIALS MINUS ONE.
C                                   1 < YKCOL < 11.
C        PARAMETER               (KCOLY = YKCOL)
C
        INTEGER                  NINTY
C        PARAMETER               (NINTY = YNINT)
C                                YNINT IS THE NUMBER OF SUBINTERVALS
C                                DEFINED BY THE SPATIAL MESH Y.
C
C       DOUBLE PRECISION          YCOL(YNCPTS)
        DOUBLE PRECISION         YCOL(*)
C                                THE SEQUENCE OF COLLOCATION POINTS ON
C                                THE INTERVAL [Y_A, Y_B].
C
        DOUBLE PRECISION         TEMP
C-----------------------------------------------------------------------
C
        COMMON /BURGER/          COEFF
        COMMON /YBSPLINE/        NINTY, KCOLY, YCOL(442)
        COMMON /ABDLU/           YABD(8804),
&                                IPIVOT(442), IABDTP, IABDBK, IABDBT
        COMMON /YBCOEFF/         YFBASIS(29172)
```

```
C-----------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                  I
C-----------------------------------------------------------------------
CC***END PROLOGUE  F
C     CALCULATE U UX
      CALL YEVAL1(KCOLY,NINTY,YFBASIS,UBAR,UBARX,  U,UX)
C
C     CALCULATE UXX UY UYY
      CALL YEVAL2(KCOLY,NINTY,YFBASIS,UBAR,UBARXX,  UXX,UY,UYY)

C     ASSUMING PDES HOLD ON GC = U(X,0,T)  GD = U(X,1,T)
      DO 10 I = 1, NPDE
         FVAL(I) = COEFF * UXX(I) + COEFF * UYY(I)
     &             - U(I) * UX(I) - U(I) * UY(I)
   10 CONTINUE
C-----------------------------------------------------------------------
         CALL CRSLVE(YABD(IABDTP),1,2*1,YABD(IABDBK),KCOLY*1,
     &               (KCOLY+NCONTI)*1,NINTY,YABD(IABDBT),1,
     &               IPIVOT,FVAL,0)

      RETURN
      END
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE F-------------------------------------------
      SUBROUTINE UINIT(X, U, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C        THIS SUBROUTINE IS USED TO RETURN THE NPDE-VECTOR OF INITIAL
C        CONDITIONS OF THE UNKNOWN FUNCTION AT THE INITIAL TIME T = T0
C        AT THE SPATIAL COORDINATE X.
```

```
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
      DOUBLE PRECISION      ZERO
      PARAMETER             (ZERO = 0.0D0)
C
      DOUBLE PRECISION      NEGONE
      PARAMETER             (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
      INTEGER               NCONTI
      PARAMETER             (NCONTI = 2)
C                           NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                           AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
      DOUBLE PRECISION      X
C                           THE SPATIAL COORDINATE.
C
      INTEGER               NPDE
C                           THE NUMBER OF PDES IN THE SYSTEM.
C
C OUTPUT:
      DOUBLE PRECISION      U(NPDE)
C                           U(1:NPDE) IS VECTOR OF INITIAL VALUES OF
C                           THE UNKNOWN FUNCTION AT T = T0 AND THE
C                           GIVEN VALUE OF X.
C
C-----------------------------------------------------------------------
      DOUBLE PRECISION      COEFF
```

```
C                                  COEFF IS THE COEFFCIENT OF UXX IN THE
C                                  BURGERS' EQUATION
C
      INTEGER                      IPIVOT(*)
C                                  PIVOTING INFORMATION FROM THE
C                                  FACTORIZATION OF THE TEMPORARY MATRIX.
C
      DOUBLE PRECISION             YABD(*)
C                                  ABD
C
      INTEGER                      IABDTP
C                                  WORK(IABDTP) CONTAINS A COPY OF THE TOP
C                                  BLOCK WHICH IS REQUIRED SINCE CRDCMP
C                                  OVERWRITES THE INPUT COLLOCATION MATRIX.
C
      INTEGER                      IABDBK
C                                  WORK(IABDBK) CONTAINS A COPY OF ABDBLK
C                                  WHICH IS REQUIRED SINCE CRDCMP
C                                  OVERWRITES THE INPUT COLLOCATION MATRIX.
C
      INTEGER                      IABDBT
C                                  WORK(IABDBT) CONTAINS A COPY OF THE
C                                  BOTTOM BLOCK WHICH IS REQUIRED SINCE
C                                  CRDCMP OVERWRITES THE INPUT COLLOCATION
C                                  MATRIX.
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER                    KCOLY
C                                YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                                TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                                WHICH IS EQUAL TO THE DEGREE OF THE
C                                PIECEWISE POLYNOMIALS MINUS ONE.
```

```
C                                   1 < YKCOL < 11.
C         PARAMETER                 (KCOLY = YKCOL)
C
          INTEGER                   NINTY
C         PARAMETER                 (NINTY = YNINT)
C                                   YNINT IS THE NUMBER OF SUBINTERVALS
C                                   DEFINED BY THE SPATIAL MESH Y.
C
C         DOUBLE PRECISION          YCOL(YNCPTS)
          DOUBLE PRECISION          YCOL(*)
C                                   THE SEQUENCE OF COLLOCATION POINTS ON
C                                   THE INTERVAL [Y_A, Y_B].
C
C-----------------------------------------------------------------------
C
          COMMON /BURGER/           COEFF
          COMMON /YBSPLINE/         NINTY, KCOLY, YCOL(442)
          COMMON /ABDLU/            YABD(8804),
&                                   IPIVOT(442), IABDTP, IABDBK, IABDBT

C-----------------------------------------------------------------------
C LOOP INDICES:
          INTEGER                   I
C-----------------------------------------------------------------------
CC***END PROLOGUE  UINIT

C     THE SEQUENCE OF COLLOCATION POINTS ON
C     THE INTERVAL [Y_A, Y_B].

      DO 10 I = 1, NPDE
         U(I) = 1.0D0 / (1.0D0+DEXP((X+YCOL(I)) / (2*COEFF)))
   10 CONTINUE
```

```
      CALL CRSLVE(YABD(IABDTP),1,2*1,YABD(IABDBK),KCOLY*1,
     &                  (KCOLY+NCONTI)*1,NINTY,YABD(IABDBT),1,
     &                  IPIVOT,U,0)

      RETURN
      END
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE UINIT-----------------------------------------
      SUBROUTINE TRUU(T, X, U, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C     THIS FUNCTION PROVIDES THE EXACT SOLUTION OF THE PDE.
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C-----------------------------------------------------------------------
C CONSTANTS:
      DOUBLE PRECISION      ZERO
      PARAMETER             (ZERO = 0.0D0)
C
      DOUBLE PRECISION      NEGONE
      PARAMETER             (NEGONE = -1.0D0)
C
C-----------------------------------------------------------------------
      INTEGER               NCONTI
      PARAMETER             (NCONTI = 2)
C                           NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                           AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C INPUT:
      INTEGER               NPDE
C                           THE NUMBER OF PDES IN THE SYSTEM.
```

```
C
      DOUBLE PRECISION        T
C                             THE CURRENT TIME COORDINATE.
C
      DOUBLE PRECISION        X
C                             THE CURRENT SPATIAL COORDINATE.
C
C OUTPUT:
      DOUBLE PRECISION        U(NPDE)
C                             U(1:NPDE) IS THE EXACT SOLUTION AT THE
C                             POINT (T,X).
C-----------------------------------------------------------------------
      DOUBLE PRECISION        COEFF
C                             COEFF IS THE COEFFCIENT OF UXX IN THE
C                             BURGERS' EQUATION
C
C-----------------------------------------------------------------------
C     Y DIMENSION
      INTEGER                 KCOLY
C                             YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                             TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                             WHICH IS EQUAL TO THE DEGREE OF THE
C                             PIECEWISE POLYNOMIALS MINUS ONE.
C                                1 < YKCOL < 11.
C
      INTEGER                 NINTY
C      PARAMETER              (NINTY = YNINT)
C                             YNINT IS THE NUMBER OF SUBINTERVALS
C                             DEFINED BY THE SPATIAL MESH Y.
C
      DOUBLE PRECISION        YCOL(*)
C                             THE SEQUENCE OF COLLOCATION POINTS ON
```

```
C                                THE INTERVAL [Y_A, Y_B].
C
C-----------------------------------------------------------------------
C
      COMMON /BURGER/          COEFF
      COMMON /YBSPLINE/        NINTY, KCOLY, YCOL(442)

C-----------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                  I
C-----------------------------------------------------------------------
C***END PROLOGUE  TRUU
      DO 10 I = 1, NPDE
         U(I) = 1.0D0 / (1.0D0+DEXP((X+YCOL(I)-T) / (2*COEFF)))
   10 CONTINUE

      RETURN
      END
C-----------------------------------------------------------------------
C-------END OF SUBROUTINE TRUU------------------------------------------
```

Appendix B

Source Code (A B-spline Gaussian Collocation for 2D Time-dependent Parabolic PDE)

These are the user-supplied subroutines for the 2D Burgers' equation.

```
C MODULE BSPLINE_GLOBAL U = 1.0D0 / (1.0D0+DEXP((X+Y-T) / (2*COEFF)))
C-----------------------------------------------------------------------
      SUBROUTINE UINIT (W, WPRIME, RPAR, IPAR)
C THIS ROUTINE COMPUTES AND LOADS THE VECTOR OF INITIAL VALUES.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION W(*), WPRIME(*), RPAR(4), IPAR(34)
      DOUBLE PRECISION      PI
C       INTRINSIC COS,SIN
      PARAMETER(PI=3.14159265)

C-----------------------------------------------------------------------
C CONSTANTS:
        INTEGER                 NCONTI
```

```
      PARAMETER             (NCONTI = 2)
C                            NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                            AT THE INTERNAL MESH POINTS.
      INTEGER               NPDE
C                            NUMBER OF PDES
      PARAMETER             (NPDE = 1)

C-----------------------------------------------------------------------
C     X DEMENSION
      INTEGER               XKCOL
C                          XKCOL IS THE NUMBER OF COLLOCATION POINTS
C                            TO BE USED IN EACH SUBINTERVAL, WHICH IS
C                            EQUAL TO THE DEGREE OF THE PIECEWISE
C                            POLYNOMIALS MINUS ONE.

      INTEGER               XNINT
C                            XNINT IS THE NUMBER OF SUBINTERVALS
C                            DEFINED BY THE SPATIAL MESH X.
C
      INTEGER               XNCPTS
C                            XNCPTS IS THE NUMBER
C                            OF COLLOCATION POINTS.
C-----------------------------------------------------------------------
C     Y DEMENSION
      INTEGER           YKCOL
C                       YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                       TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                       WHICH IS EQUAL TO THE DEGREE OF THE
C                       PIECEWISE POLYNOMIALS MINUS ONE.

      INTEGER               YNINT
C                            YNINT IS THE NUMBER OF SUBINTERVALS
```

```
C                                   DEFINED BY THE SPATIAL MESH Y.
C
      INTEGER                       YNCPTS
C                                   YNCPTS IS THE NUMBER
C                                   OF COLLOCATION POINTS.

      INTEGER                       NEQ
C                                   NEQ IS
C                                   THE NUMBER OF BSPLINES
C                                   COEFFICIENTS (OR DAES).
C-----------------------------------------------------------------------
C     COMMON
      INTEGER                       KCOLX
C                                   KCOL IS THE NUMBER OF COLLOCATION POINTS
C                                   TO BE USED IN EACH SUBINTERVAL, WHICH IS
C                                   EQUAL TO THE DEGREE OF THE PIECEWISE
C                                   POLYNOMIALS MINUS ONE.

      INTEGER                       NINTX
C                                   NINTX IS THE NUMBER OF SUBINTERVALS
C                                   DEFINED BY THE SPATIAL MESH X.
C

      INTEGER                     KCOLY
C                                 KCOLY IS THE NUMBER OF COLLOCATION POINTS
C                                 TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                                 WHICH IS EQUAL TO THE DEGREE OF THE
C                                 PIECEWISE POLYNOMIALS MINUS ONE.

      INTEGER                       NINTY
C                                   YNINT IS THE NUMBER OF SUBINTERVALS
C                                   DEFINED BY THE SPATIAL MESH Y.
```

```
C
      DOUBLE PRECISION      XCOL(*)
C                           THE SEQUENCE OF COLLOCATION POINTS ON
C                           THE INTERVAL [Y_A, Y_B].

      DOUBLE PRECISION      YCOL(*)
C                           THE SEQUENCE OF COLLOCATION POINTS ON
C                           THE INTERVAL [Y_A, Y_B].

      DOUBLE PRECISION      COEFF
C                           COEFF IS THE COEFFCIENT OF UXX IN THE
C                           BURGERS' EQUATION

C LOOP INDICES:
      INTEGER               I
      INTEGER               J
      INTEGER               K
C-----------------------------------------------------------------------
      COMMON /BSPLINE/      KCOLX, NINTX, KCOLY, NINTY

      COMMON /COLP/         XCOL(402), YCOL(402), COEFF

C-----------------------------------------------------------------------
      XKCOL = KCOLX
      XNINT = NINTX
      YKCOL = KCOLY
      YNINT = NINTY
      XNCPTS=(XKCOL*XNINT+NCONTI)
      YNCPTS=(YKCOL*YNINT+NCONTI)
      NEQ=YNCPTS*XNCPTS

      DO 11 I = 1, NEQ
```

```
            W(K) = 0.0D0
            WPRIME(K) = 0.0D0
 11       CONTINUE

       DO 20 I = 1, XNCPTS
         DO 10 J = 1, YNCPTS
           K = (I-1)*YNCPTS+J
           TEMP = DEXP((XCOL(I)+YCOL(J)) / (2.0D0*COEFF))
           IF (REAL(1.0D0/TEMP) .NE. 0.) THEN
             W(K)= 1.0D0 / (1.0D0+TEMP)
             WPRIME(K)=1.0D0/(2.0D0*COEFF)*TEMP /
     &                  ((1.0D0+TEMP)*(1.0D0+TEMP))
           ENDIF
 10        CONTINUE
 20      CONTINUE

  999 RETURN

C------------  END OF SUBROUTINE UINIT  ------------------------------------
       END
       SUBROUTINE TRUU(T, X, XNPTS, Y, YNPTS, UTRUE)
C
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       DIMENSION UTRUE(*)

       DOUBLE PRECISION      PI
       INTRINSIC COS,SIN
       PARAMETER(PI=3.14159265)
C-------------------------------------------------------------------------
C CONSTANTS:
         INTEGER                NCONTI
         PARAMETER              (NCONTI = 2)
```

```
C                               NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                               AT THE INTERNAL MESH POINTS.
C-----------------------------------------------------------------------
C       X DEMENSION
        INTEGER                 XNPTS
C                               XNPTS IS THE NUMBER OF POINTS

        DOUBLE PRECISION        X(XNPTS)
C                               X IS THE SPATIAL POINTS
C
C-----------------------------------------------------------------------
C       Y DEMENSION
        INTEGER                 YNPTS
C                               YNPTS IS THE NUMBER OF POINTS

        DOUBLE PRECISION        Y(YNPTS)
C                               Y IS THE SPATIAL POINTS
C-----------------------------------------------------------------------
C COMMON:
        DOUBLE PRECISION        XCOL(*)
C                               THE SEQUENCE OF COLLOCATION POINTS ON
C                               THE INTERVAL [X_A, X_B].

        DOUBLE PRECISION        YCOL(*)
C                               THE SEQUENCE OF COLLOCATION POINTS ON
C                               THE INTERVAL [Y_C, Y_D].

        DOUBLE PRECISION        COEFF
C                               COEFF IS THE COEFFCIENT OF UXX IN THE
C                               BURGERS' EQUATION
C-----------------------------------------------------------------------
C LOOP INDICES:
```

```
      INTEGER                 I
      INTEGER                 J
      INTEGER                 K
C-----------------------------------------------------------------------
      COMMON /COLP/           XCOL(402), YCOL(402), COEFF
C-----------------------------------------------------------------------
C LOAD U INTO DELTA, IN ORDER TO SET BOUNDARY VALUES.
      DO 10 I = 1,XNPTS*YNPTS
 10     UTRUE(I) = 0.0D0

C LOOP OVER ALL POINTS, AND LOAD RESIDUAL VALUES.
      DO 20 I = 1, XNPTS
        DO 11 J = 1, YNPTS
          K = (I-1)*YNPTS+J
          TEMP = DEXP((X(I)+Y(J)-T) / (2*COEFF))
          UTRUE(K)= 1.0D0 / (1.0D0+TEMP)
 11       CONTINUE
 20     CONTINUE

      RETURN
C-------------- END OF SUBROUTINE TRUU ---------------------------------
      END

      SUBROUTINE RESH (T, W, WPRIME, CJ, DELTA, IRES, RPAR, IPAR)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      DIMENSION W(*), WPRIME(*), DELTA(*), RPAR(4),IPAR(34)

      DOUBLE PRECISION        PI
      INTRINSIC COS,SIN
      PARAMETER(PI=3.14159265)
```

```
C-------------------------------------------------------------------
C CONSTANTS:
      INTEGER              NCONTI
      PARAMETER            (NCONTI = 2)
C                          NCONTI CONTINUITY CONDITIONS ARE IMPOSED
C                          AT THE INTERNAL MESH POINTS.

      INTEGER              LENGTH
      PARAMETER            (LENGTH = 161604)
C-------------------------------------------------------------------
C     X DEMENSION
      INTEGER              XKCOL
C                          KCOL IS THE NUMBER OF COLLOCATION POINTS
C                          TO BE USED IN EACH SUBINTERVAL, WHICH IS
C                          EQUAL TO THE DEGREE OF THE PIECEWISE
C                          POLYNOMIALS MINUS ONE.

      INTEGER              XNINT
C     PARAMETER            (XNINT = 2)
C                          NINT IS THE NUMBER OF SUBINTERVALS
C                          DEFINED BY THE SPATIAL MESH X.
C
      INTEGER              XNCPTS)
C                          XNCPTS IS THE NUMBER
C                          OF COLLOCATION POINTS.
C-------------------------------------------------------------------
C     Y DEMENSION
      INTEGER              YKCOL
C                          YKCOL IS THE NUMBER OF COLLOCATION POINTS
C                          TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                          WHICH IS EQUAL TO THE DEGREE OF THE
C                          PIECEWISE POLYNOMIALS MINUS ONE.
```

```
      INTEGER                 YNINT
C                             YNINT IS THE NUMBER OF SUBINTERVALS
C                             DEFINED BY THE SPATIAL MESH Y.
C
      INTEGER                 YNCPTS
C                             YNCPTS IS THE NUMBER
C                             OF COLLOCATION POINTS.
C-----------------------------------------------------------------------
C LOCAL
      DOUBLE PRECISION        UXX(LENGTH) ,UX(LENGTH)
      DOUBLE PRECISION        UYY(LENGTH) ,UY(LENGTH)
      DOUBLE PRECISION        U(LENGTH) ,UPRIME(LENGTH)

      INTEGER                 NEQ
C                             NEQ IS
C                             THE NUMBER OF BSPLINES
C                             COEFFICIENTS (OR DAES).
C-----------------------------------------------------------------------
C     COMMON
      INTEGER                 KCOLX
C                            KCOLX IS THE NUMBER OF COLLOCATION POINTS
C                             TO BE USED IN EACH SUBINTERVAL, WHICH IS
C                             EQUAL TO THE DEGREE OF THE PIECEWISE
C                             POLYNOMIALS MINUS ONE.

      INTEGER                 NINTX
C                             NINTX IS THE NUMBER OF SUBINTERVALS
C                             DEFINED BY THE SPATIAL MESH X.
C
      INTEGER                KCOLY
C                            KCOLY IS THE NUMBER OF COLLOCATION POINTS
```

```
C                              TO BE USED IN EACH SUBINTERVAL IN Y DOMAIN,
C                              WHICH IS EQUAL TO THE DEGREE OF THE
C                              PIECEWISE POLYNOMIALS MINUS ONE.

      INTEGER                  NINTY
C                              NINTY IS THE NUMBER OF SUBINTERVALS
C                              DEFINED BY THE SPATIAL MESH Y.
C
      DOUBLE PRECISION         XCOL(*)
C                              THE SEQUENCE OF COLLOCATION POINTS ON
C                              THE INTERVAL [X_A, X_B].

      DOUBLE PRECISION         YCOL(*)
C                              THE SEQUENCE OF COLLOCATION POINTS ON
C                              THE INTERVAL [Y_C, Y_D].

      DOUBLE PRECISION         COEFF
C                              COEFF IS THE COEFFCIENT OF UXX IN THE
C                              BURGERS' EQUATION

      DOUBLE PRECISION         XFBASIS(*), YFBASIS(*)
C-----------------------------------------------------------------------
C LOOP INDICES:
      INTEGER                  I
      INTEGER                  J
      INTEGER                  IJ
      INTEGER                  K
C-----------------------------------------------------------------------
      COMMON /BSPLINE/         KCOLX, NINTX, KCOLY, NINTY
      COMMON /COLP/            XCOL(402), YCOL(402), COEFF
      COMMON /FBASIS/          XFBASIS(26532),YFBASIS(26532)
C-----------------------------------------------------------------------
```

```
C
      XKCOL = KCOLX
      XNINT = NINTX
      YKCOL = KCOLY
      YNINT = NINTY
      XNCPTS=(XKCOL*XNINT+NCONTI)
      YNCPTS=(YKCOL*YNINT+NCONTI)
      NEQ=YNCPTS*XNCPTS
C
C LOAD U INTO DELTA, IN ORDER TO SET BOUNDARY VALUES.
      DO 10 I = 1,NEQ
         DELTA(I) = 0.0
 10   CONTINUE

      CALL KRONXY(XKCOL, XNINT, XFBASIS, YKCOL, YNINT, YFBASIS,
     &                   W, WPRIME, U, UPRIME,
     &                   UX, UXX, UY, UYY)

C     HERE DELTA = F
      DO 20 I = 1, XNCPTS
        DO 11 J = 1, YNCPTS
          K = (I-1)*YNCPTS+J
          DELTA(K) = -U(K)*UX(K)-U(K)*UY(K)
     &               +COEFF*(UXX(K)+UYY(K))
 11       CONTINUE
 20     CONTINUE

      DO 70 I = 1,NEQ
         DELTA(I) = UPRIME(I) - DELTA(I)
 70     CONTINUE
C LOAD UPRIME ON THE BOUNDARY INTO DELTA.
C     TP
```

```
C       X = A (XCOL(1) = 0)
        DO 30 I = 1,YNCPTS
           TEMP = DEXP((YCOL(I)-T) / (2.0D0*COEFF))
           DELTA(I) = U(I) - 1.0D0/(1.0D0 + TEMP)
           DELTA(I) = DELTA(I) * CJ
 30      CONTINUE

C       BK
        DO 50 I = 1,XNINT
          DO 40 J = 1,XKCOL
             IJ = (I-1)*XKCOL + J + 1
C       Y = C (YCOL(1) = 0)
             K = YNCPTS + (I-1)*XKCOL*YNCPTS + (J-1)*YNCPTS + 1
             TEMP = DEXP((XCOL(IJ)-T) / (2.0D0*COEFF))
             DELTA(K) = U(K) - 1.0D0/(1.0D0 + TEMP)
             DELTA(K) = DELTA(K) * CJ
C       Y = D (YCOL(YNCPTS) = 1)
             K = YNCPTS + (I-1)*XKCOL*YNCPTS + J*YNCPTS
             TEMP = DEXP((1.0 + XCOL(IJ)-T) / (2.0D0*COEFF))
             DELTA(K) = U(K) - 1.0D0/(1.0D0 + TEMP)
             DELTA(K) = DELTA(K) * CJ
 40        CONTINUE
 50      CONTINUE

C       BT
C       X = B (XCOL(XNCPTS) = 1)
        K = YNCPTS + XNINT*XKCOL*YNCPTS
        DO 60 I = 1,YNCPTS
           TEMP = DEXP((1.0 + YCOL(I)-T) / (2.0D0*COEFF))
           DELTA(K+I) = U(K+I) - 1.0D0/(1.0D0 + TEMP)
           DELTA(K+I) = DELTA(K+I) * CJ
 60        CONTINUE
```

```
C
      RETURN
C------------  END OF SUBROUTINE RESH  ------------------------------------
      END
```