Saint Mary's University Mathematics and Computing Science Technical Report 2005-003

Collocation software based on a Runge-Kutta time integrator for 1-D Parabolic PDEs and Schrödinger type problems, with spatial and temporal error control *

R. Wang,[†] P. Keast,[‡] P. Muir[§]

Abstract

The BACOL software package, which employs high order methods in time and space to adaptively control spatial and temporal errors in a method-of-lines approach, has been shown to be significantly more efficient than existing codes for the accurate numerical solution of systems of parabolic PDEs in one space dimension. In BACOL, the collocation spatial discretization gives a system of differential-algebraic equations (DAEs) which is treated using the DAE solver DASSL. Since DASSL employs backward differentiation formulas (BDFs), each spatial remeshing requires an interpolation of previous solution values. In addition, for DAE systems whose Jacobians have eigenvalues on the imaginary axis, such as those arising from Schrödinger problems, DASSL performs inefficiently since the higher order BDFs are not A-stable. In this paper, we describe a new software package, BACOLR, which addresses these issues by using RADAU5, a DAE solver based on an A-stable, (one-step) implicit Runge-Kutta method. Numerical results show that BACOLR is generally more efficient than BACOL on parabolic PDEs and substantially more efficient on Schrödinger problems.

*This work was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada and by MITACS, the Mathematics of Information Technology and Complex Systems.

[†]Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, S7N 5A9, Canada. rong@cs.usask.ca

[‡]Department of Mathematics and Statistics, Dalhousie University, Halifax, Nova Scotia B3H 3J5, Canada. keast@mathstat.dal.ca.

[§]Department of Mathematics and Computing Science, Saint Mary's University, Nova Scotia B3H 3C3, Canada. muir@stmarys.ca.

Keywords: numerical software, method-of-lines, Runge-Kutta methods, differential-algebraic equations, 1-D parabolic PDEs, Schrödinger problems
AMS(MOS) subject classification: 65M20, 65M50, 65M70

1 Introduction

1.1 Overview

In [32], the recently developed BACOL code [31, 33], for the numerical solution of systems of parabolic PDEs in one space dimension, is shown to be significantly more efficient than similar available codes such as PDECOL/EPDCOL [21, 19], PDECHEB [4], D03PPF [10], TOMS371 [6], MOVCOL [16], and HP-NEW [24]. It is also able to solve problems to a higher accuracy than these other packages. This better performance is especially evident when the solution has narrow spikes or boundary layers. Based on the method-of-lines (MOL) approach, BACOL uses collocation to discretize the PDEs, leading to a system of ODEs which together with the boundary conditions form a system of differential-algebraic equations (DAEs). This system is solved using DASSL [25], a widely used DAE solver which is based on a family of multistep methods, specifically backward differentiation formulas (BDFs), whose orders vary from 1 to 5.

Because DASSL employs BDFs, two issues arise when it is used within the BACOL code. The first issue is related to the fact that a multistep method requires solution approximations from several previous time steps, evaluated at the points of the current spatial mesh. This can lead to a difficulty because BACOL has the ability to refine and redistribute the spatial mesh. As long as the spatial mesh stays fixed over a sufficient number of time steps, the previous solution values are easily available to the multistep method but whenever there is a spatial mesh redistribution or refinement, BACOL must compute solution approximations for the new mesh, at the current time and up to five previous time steps, depending on what order of BDF formula is currently being employed. The availability of these previous solution values allows DASSL to perform what is referred to as a *warm restart*, i.e., after the spatial remeshing DASSL can continue with the same order and stepsize. The alternative is for DASSL to restart with a very small stepsize and a method of order one so that no past values are required but such an approach, referred to in the literature as a *cold restart*, has been shown to be very inefficient [3]. BACOL obtains an estimate of the spatial error by computing two global solutions, of degree p and $p+1$, on the same spatial mesh. If the error is too large or is not approximately equidistributed over the subintervals of the mesh, it is necessary to change the mesh and then the two global solutions must be interpolated from the old mesh to the new one. This means that the back values computed by DASSL must also be interpolated on the new mesh. Consequently as many as twelve interpolations may be necessary. This results in considerable overhead.

A second issue with using DASSL as the time integrator in BACOL arises in

the solution of DAEs in which eigenvalues of the Jacobian are purely imaginary (such as those that arise from Schrödinger problems). In such a case it is usually impossible for DASSL to obtain a solution unless the user restricts the code so that it only uses BDFs of orders 1 or 2 because the stability regions of the higher order BDFs do not contain the imaginary axis; this order restriction results in significant inefficiency. (We note that these same issues arise with any adaptive MOL solver that employs a DAE or IVP solver based on a multistep method.)

The above two issues are addressed through the use of implicit Runge-Kutta methods, e.g., [14]. These are one-step methods and thus no solution values from previous steps are needed in the determination of the solution values for the next step. As well there exist many methods which are high order and A-stable (which implies that the stability region includes the imaginary axis). A well-known DAE solver based on a particular fifth order implicit Runge-Kutta method of Radau type is the RADAU5 package [14]. Due to the one-step nature and the A-stability of the underlying Runge-Kutta formula employed within this code, it appears to provide an attractive alternative to DASSL for use in BACOL. A drawback, however, is that in order to obtain an efficient implementation of the fifth order Radau formula, RADAU5 employs complex arithmetic. One must then consider the trade-off between this higher cost and the savings associated with being able to avoid both interpolation after a remeshing and order restrictions due to stability issues when one considers problems leading to DAE systems having Jacobians with eigenvalues on the imaginary axis.

In this paper, we describe a new software package called BACOLR which addresses the difficulties associated with the use of DASSL in BACOL. BACOLR was developed from BACOL through a substantial modification process which involved replacing DASSL with a significantly modified version of RADAU5.

This paper is organized as follows. We conclude this section with a brief overview of method-of-lines software for 1-D parabolic PDEs and the general forms of the problem classes to be considered in this paper. In section 2 we describe the new package, providing a summary of the modifications undertaken to develop BACOLR from the original BACOL code and the modifications to the RADAU5 code, followed by a brief description of the interface for the BACOLR package. Section 3 gives the main thrust of the paper which is the comparison of BACOLR and BACOL on two 1-D parabolic problems and two 1-D Schrödinger problems. Numerical results show that BACOLR is generally more efficient than BACOL on standard 1-D parabolic PDEs and substantially more efficient on 1-D Schrödinger problems. We close with a summary, some conclusions, and a discussion of future work.

1.2 Method-of-lines Software for 1-D Parabolic PDEs

In the numerical approximation to solutions of parabolic PDEs, the MOL is one of the most popular approaches. In this technique, one begins with a mesh which partitions the spatial domain. The spatial operators are then discretized with a scheme based on, for example, finite difference methods or collocation, leading to a system of initial value ordinary differential equations (IVODEs).

The boundary conditions in space may be applied directly to the approximation space, or differentiated to give IVODEs. In the latter case a system of IVODEs results, while in the former case a system of DAEs of index 1 results. The system of IVODEs or DAEs is then solved approximately using a suitable solver.

Over the last three decades several software packages have been developed for the numerical solution of systems of parabolic PDEs in one space dimension, based on the standard MOL. Early codes (e.g., PDECOL, EPDCOL, PDECHEB) made no attempt to control the spatial error or adapt the spatial mesh; they employed a fixed (but possibly non-uniform) mesh. They did, however, attempt to control the local time error through the IVODE or DAE solver which would use error estimates and adaptive strategies, such as variable stepsize and formula order.

More recent packages (e.g., D03PPF, TOMS731, MOVCOL), employ a fixed number of mesh points and use redistribution or movement of these points to try to reduce the spatial error. These are referred to as *adaptive* MOL (AMOL) codes. In addition to employing temporal adaptivity and error control through a high quality IVODE or DAE solver, these codes are able to adjust the locations of the mesh points to focus points on regions of difficult solution behaviour, typically based on a monitor function. Thus for a given discretization scheme and a given number of mesh points one obtains about as much accuracy in space as is possible. While better spatial accuracy is obtained through adaptivity, there is no capability for actually controlling the spatial error.

Within the last few years a new generation of AMOL codes has been developed. One such code is HPNEW which is a modification of HPDASSL [23]. A related code is HPSIRK [23], which was shown to be generally less efficient than HPDASSL. The HPSIRK, HPDASSL, and HPNEW codes employ both mesh and spatial order adaptivity, and use finite-element Galerkin methods for the spatial discretization. In HPNEW and HPDASSL the resultant DAEs are treated using DASSL. HPSIRK solves the DAEs using singly-implicit Runge-Kutta methods, e.g., [14]. Another code from this class is the BACOL code, mentioned earlier, which we describe in more detail in the next section.

1.3 Problem Classes

In this paper we consider the numerical solution of two classes of problems that commonly appear in the literature.

The first class consists of systems of time-dependent parabolic PDEs in one space dimension. Such problems arise in a great many applications including chemical reaction and heat diffusion studies. We assume parabolic problems with the general form

$$\mathbf{u}_t(x,t) = \mathbf{f}(x,t,\mathbf{u}(x,t),\mathbf{u}_x(x,t),\mathbf{u}_{xx}(x,t)), \quad a \le x \le b, \quad t \ge t_0, \qquad (1)$$

with initial conditions

$$\mathbf{u}(x,t_0) = \mathbf{u}_0(x), \quad a \le x \le b,$$

and separated boundary conditions

$$\mathbf{g_a}(t, \mathbf{u}(a,t), \mathbf{u}_x(a,t)) = \mathbf{0}, \quad \mathbf{g_b}(t, \mathbf{u}(b,t), \mathbf{u}_x(b,t)) = \mathbf{0}, \quad t \geq t_0.$$

We assume that $\mathbf{u}$, $\mathbf{f}$, $\mathbf{u_0}$, $\mathbf{g_a}$, and $\mathbf{g_b}$ are vector functions with n components. We consider two problems of this type; the first is Burger's equation, which is a single nonlinear PDE; the second problem is a system of four coupled nonlinear PDEs arising from a catalytic surface reaction model. These problems are described in sections 3.1 and 3.2.

The second class consists of time dependent Schrödinger problems in one space variable. Application areas in which such problems arise include fibre optics, plasma physics, laser pulses, seismology, and quantum mechanics. The paper [12] provides an excellent survey of problems of this type and numerical methods for their treatment. In this paper we consider problems of this type that have the general form (1) with the restriction that

$$\mathbf{f}(x, t, \mathbf{u}(x,t), \mathbf{u}_x(x,t), \mathbf{u}_{xx}(x,t)) = i(S_1 \mathbf{u}_x(x,t) + S_2 \mathbf{u}_{xx}(x,t) + \mathbf{F}(x, t, \mathbf{u}(x,t))), \tag{2}$$

where $i^2 = -1$, S_1 and S_2 are n by n matrices, and the function $\mathbf{F}$ is a vector function with n components. In this paper, we consider two problems of this type, the cubic (nonlinear) Schrödinger equation and a coupled nonlinear Schrödinger system consisting of two equations. These problems are described in sections 3.3 and 3.4.

2 BACOLR

2.1 Overview

Here we provide a brief overview of the algorithms implemented in the BACOL and BACOLR packages. We describe the algorithms assuming a single PDE but the generalization to a system of PDEs is straightforward. The reader is referred to [31] for further details.

The spatial discretization is based on a partitioning of the spatial domain by a mesh of $N+1$ points, $a = x_0 < x_1 < \cdots < x_N = b$. A piecewise polynomial of degree p (where p is chosen by the user) is set up on this mesh, having continuous first derivatives at the mesh points and having $NC \equiv N(p-1)+2$ free parameters. BACOL and BACOLR use B-splines [7] as a basis for these piecewise polynomials. The solution $u(x,t)$ is then approximated by $U(x,t)$ where

$$U(x,t) = \sum_{i=1}^{NC} B_i(x) y_i(t), \tag{3}$$

and where $y_i(t)$ represents the (unknown) coefficient of the i-th B-spline basis function. These coefficients are then computed using collocation. That is, we require the piecewise polynomial in (3) to satisfy the PDE at $p-1$ Gaussian points in each of the N subintervals, and to satisfy the two boundary conditions.

This process results in a set of $NC-2$ ODEs and two algebraic equations, thereby giving a system of DAEs of index 1. (The DAE system has index 1 because a differentiation of the boundary conditions gives a pure ODE system.) Due to the type of discretization scheme and basis, this DAE system has a Jacobian which has a special structure known as almost block diagonal (ABD) [2].

In BACOL, as mentioned earlier, the time integrator used is a modified version of DASSL. The changes made are detailed in [33]. One of the major changes to DASSL involved adding a module, COLROW, [11], for the solution of the ABD linear systems which arise during the time integration. In BACOLR, as indicated earlier, the time integration is handled by the RADAU5 code, which was also modified in a number of ways, as described in the next subsection. In each code the time tolerance provided to the DAE solver is the tolerance provided by the user.

BACOL and BACOLR each compute two approximate solutions, one with degree p and one with degree $p+1$. The difference between these two solution approximations is used to obtain an estimate of the error. We require $E_s < 1$ where E_s is the error estimate associated with the sth solution component; E_s has the form

$$E_s = \sqrt{\int_a^b \left(\frac{U_s(x,t) - \bar{U}_s(x,t)}{ATOL_s + RTOL_s|U_s(x,t)|} \right)^2 dx}, \tag{4}$$

where t is the current time, $ATOL_s$ and $RTOL_s$ denote the absolute and relative tolerances for the s-th component of the PDE system, respectively, $U_s(x,t)$ is the degree p solution approximation and $\bar{U}_s(x,t)$ is the degree $p+1$ solution approximation, for the s-th PDE component. When the error at the current step fails to satisfy a given tolerance, the spatial remeshing algorithm in BACOL uses an equidistribution principle to obtain a new mesh. The new mesh may consist of the same number of points, redistributed, or a redistributed mesh with either more or fewer points. The details of the error control algorithm are discussed in [31].

Once a remeshing has been performed, as indicated earlier in this paper, BACOL performs a warm restart requiring up to 5 interpolations of past solutions values on the new mesh, for each of the two solution approximations. BACOLR, since it uses a one-step method does not need to perform these interpolations.

Further software development aspects of the BACOL project are discussed in [33], where the user interface is also explained. See also section 2.3 below for some discussion of the user interface for BACOLR.

2.2 Software Modifications

The BACOL package is based on the ODE solver DASSL, a BDF code. In section 1.1 we have discussed the motivation for creating a modified version of BACOL which replaces DASSL by the Runge-Kutta solver RADAU5 for the solution of the DAEs. This of course necessitated substantial modification of

BACOL as well as significant modification of RADAU5. We now discuss the changes made to these two packages.

- Since RADAU5 uses a (one-step) Runge-Kutta method, we do not need to save solution information for up to five previous time steps and after a remeshing we do not need to use interpolation of these past values to get new solution approximations on the new mesh. We were therefore able to simplify BACOLR by removing the code in BACOL that handled this task.

- At the end of each successful time step, both BACOL and BACOLR compute an error estimate based on the comparison of two global solutions, one of spatial order p and the other of spatial order $p+1$. BACOLR has a different structure than BACOL in that the spatial error estimation and control calculations are not carried out within a module called from BACOL, but rather in a sub-module of RADAU5. The reason for this is as follows. Since the interface to DASSL allows for a return after a fixed number of steps, in BACOL we ask DASSL to return to the main BACOL subroutine after every successful time step and then we estimate the spatial error within BACOL. RADAU5 does not provide the option to return after a fixed number of time steps; it returns only after reaching a specified output time, or on a signal from one of its subroutines, SOLOUT, which it calls after every successful time step. SOLOUT was originally intended to allow the user to output the numerical solution during the integration, rather than only at the output time, or to allow the user to interrupt the computation after a successful step, should a special condition arise. We have written a SOLOUT routine which employs the solution returned by RADAU5 to compute an estimate of the spatial error and decide whether or not to do a remeshing. Since RADAU5 calls this routine after every successful time step we therefore get the same functionality for this task as is available in BACOL.

- After the spatial discretization, the resultant ODEs together with the boundary conditions give an index-1 DAE system. RADAU5 is able to handle a DAE system whose index is less than or equal to 2. However, RADAU5 requires the ODEs to be given first, followed by the algebraic equations. In our case, the algebraic constraints are generated from the boundary conditions. In order to preserve the ABD structure of the linear systems, the equations must be ordered so that those associated with the left boundary come first, followed by the equations resulting from the discretization of the PDEs, followed by the boundary conditions associated with the right boundary. From an inspection of the RADAU5 code we observed that the ordering of the equations is not fundamental to the implementation. The difference in the way RADAU5 treats the variables corresponding to the ODEs and those corresponding to the index-1 algebraic equations is that the estimates of the error for the latter variables are multiplied by the time stepsize, h_{time}. We therefore provide the ODEs

and algebraic equations to RADAU5 in the order specified above which preserves the structure of the linear systems, rather than in the order expected by RADAU5, and have modified RADAU5 to perform the appropriate scaling on the algebraic constraints, which appear as the first and last groups of equations.

- RADAU5 employs a 3-stage Radau-type implicit Runge-Kutta method and is designed for DAEs of the form $M\,y' = f(t, y)$. As discussed in [14], the Newton systems which arise in the treatment of the DAE system have the form

$$\begin{pmatrix} \gamma\,M - G & 0 & 0 \\ 0 & \alpha\,M - G & -\beta\,M \\ 0 & \beta\,M & \alpha\,M - G \end{pmatrix} \begin{pmatrix} w \\ u \\ v \end{pmatrix} = \begin{pmatrix} \rho \\ \delta \\ \psi \end{pmatrix},$$

  where $\gamma = \hat{\gamma}/h_{time}$, $\alpha = \hat{\alpha}/h_{time}$, $\beta = \hat{\beta}/h_{time}$, $\hat{\gamma}$, $\hat{\alpha}$ and $\hat{\beta}$ are positive constants, and $G = \partial f/\partial y$. One can clearly split the above system into two smaller systems. The first is $(\gamma\,M - G)w = \rho$. For the second, one can transform the real system into a complex system of the form $((\alpha + i\,\beta)M - G)(u + iv) = \delta + i\,\psi$. The matrices M and G have the same ABD matrix structure. The matrix $\gamma\,M - G$ is therefore also of ABD form, as is the matrix $(\alpha + i\,\beta)M - G$. RADAU5 offers several options for the solution of the linear system, including a full matrix solver and a banded matrix solver (both in real and complex form). However, the linear system solver, COLROW [11] is significantly more efficient in treating the ABD systems than a band solver and also does not introduce fill-in of the matrix structure, which implies that no extra storage is required. We have therefore introduced a new linear system solver option within RADAU5 to allow it to employ COLROW and its complex version, COMPLEXCOLROW [18], for the efficient treatment of the Newton systems.

- It is well known that the condition number of the iteration matrix for a DAE system with index 1 is $O((h_{time})^{-1})$; see, e.g., [9]. A poorly conditioned Jacobian may lead to failure of the associated Newton iteration [33]. A scaling technique is suggested in [26] to overcome this difficulty. This technique is implemented in BACOL and we also employ it in BACOLR. We scale the subblock of the real matrix $\gamma\,M - G$ and the corresponding right hand side elements associated with the boundary conditions by γ and scale the subblock of the complex matrix $(\alpha + i\,\beta)\,M - G$ and the corresponding right hand side elements associated with the boundary conditions by $\alpha + i\,\beta$. We have verified numerically that this scaling technique significantly reduces the condition number of the Jacobian matrix.

- Unlike DASSL, RADAU5 is unable to automatically select an initial stepsize. RADAU5 does allow the user to specify an initial stepsize however, and we provide this option in the BACOLR parameter list, consistent with

a similar option in the BACOL parameter list. If the user does not provide an initial stepsize, BACOLR sets the default value to be the maximum of the relative and absolute tolerances values.

- In RADAU5, one of the input parameters is used to decide how often the Jacobian matrix should be recomputed. The default value is appropriate for small DAE systems. However, as recommended in [14], we reset this parameter to a larger value which is more appropriate when RADAU5 is applied to large systems, such as those that arise in the MOL.

2.3 The BACOLR Interface

The interface of BACOLR is similar to that of BACOL, i.e., BACOLR has the same user supplied subroutines and essentially the same parameter list as BACOL, and we therefore refer the reader to [33] for the details of the BACOL interface. In this subsection we discuss the minor differences between the interfaces of the two codes.

- If the user wishes, BACOL can return after a certain number of accepted time steps. RADAU5 does not provide this capability directly, although it could be implemented using the SOLOUT subroutine. This option has not been implemented in the current version of BACOLR.

- DASSL will normally integrate past the output time, T_{out}, and then interpolate within the final step to obtain a solution approximation at T_{out}. For some problems, it is not possible to integrate beyond some point, T_{stop}, because the equation changes at that point or is not defined beyond that point. Thus DASSL has an option which allows the user to provide the code with the point T_{stop}. RADAU5 (and thus BACOLR) does not have this option.

- The parameter IDID is the exit status flag for both BACOL and BACOLR. A negative value of IDID indicates an unsuccessful return and an error message is output. However, the meaning of a positive value of IDID is different for each code. In BACOL, IDID can have values of 1, 2, or 3, while for BACOLR, IDID can have only one positive value, namely 1, which indicates that the integration to T_{out} was successfully completed.

- In order to continue time stepping for the current problem after a successful return, for BACOL, the user should set MFLAG(1) and IDID, while for BACOLR, the user need only set MFLAG(1) = 1.

The BACOLR software is currently available for download from the website of one of the authors; see www.mscs.dal.ca/∼keast/research/pubs.html. The source code for BACOLR contains further detailed documentation in support of the package. The webpage also includes an example driver program that can be downloaded.

3 Numerical Comparison of BACOL and BACOLR

In this section we compare BACOL and BACOLR on a series of test problems chosen from the literature. These tests were performed on a 480 MHz. SPARC CPU, using SUNWspro/bin/f77 as the FORTRAN compiler. Compilation is done using the -O switch, which provides a basic level of optimization.
The notation used and the statistics collected include:

- N: the number of subintervals;
- p: the degree of the piecewise polynomial for the primary solution approximation ($3 \le p \le 11$);
- T_{out}: the time at which the numerical solution is requested;
- TOL: the user supplied tolerance (we set $ATOL_s = RTOL_s = TOL$);
- CPU: the execution time in seconds.

We use $p = 3$, $p = 6$ and $p = 9$ in our numerical experiments. The initial mesh is chosen to be uniform with $N = 10$.
At the output time, T_{out}, we calculate the absolute L^2-norm error; i.e., the difference between the approximate solution and the exact solution:

$$\|U_s(x,t) - u_s(x,t)\|_2 = \sqrt{\int_a^b (U_s(x,t) - u_s(x,t))^2 \, dx}, \tag{5}$$

where $s = 1, \ldots, NPDE$, $u_s(x,t)$ represents the s-th component of the exact solution, and $U_s(x,t)$ represents the s-th component of the approximate solution. While both BACOL and BACOLR control a blended absolute/relative estimate of the error (4), for the problems we consider in this paper, the solution components are $O(1)$ and therefore the error estimate, (4), controlled by the codes is comparable to the error we measure, (5). (For the problems considered in this paper, (4) implies that the absolute error estimate will be required to be less than about $2 \cdot TOL$.)

We now present our test problems and computational results. We plot the exact L^2-norm error, (5), at T_{out} against the CPU time for each code. There is one data set for each of the three p values, for each code. For each data set we plot a curve marked with a unique symbol, as indicated by the legend included in the figure; for example, the $*$ represents BACOLR, $p = 9$. The nine symbols appearing along each curve correspond to the nine tolerance values considered for each code and p value; the tolerance values we employ are $\{10^{-q}\}_{q=2}^{10}$. Thus, for example, the third symbol from the left appearing on each plotted curve represents the error and CPU time we observed when the given tolerance was 10^{-4}. Using this information, we can then also observe from the figures how close each code comes to meeting the prescribed tolerance.

3.1 Burgers' Equation

This parabolic equation has the form

$$u_t = -uu_x + \epsilon u_{xx}, \qquad 0 < x < 1, \quad t > 0, \quad \epsilon > 0.$$

The initial condition and boundary conditions are chosen so that the exact solution is given by

$$u(x,t) = \frac{0.1e^{-A} + 0.5e^{-B} + e^{-C}}{e^{-A} + e^{-B} + e^{-C}},$$

where

$$A = \frac{0.05}{\epsilon}(x - 0.5 + 4.95t), \quad B = \frac{0.25}{\epsilon}(x - 0.5 + 0.75t), \quad C = \frac{0.5}{\epsilon}(x - 0.375).$$

This problem is taken from [1, 5]. The exact solution is plotted in Figure 1 for $\epsilon = 10^{-4}$, $0 \le t \le 1$, $0 \le x \le 1$.

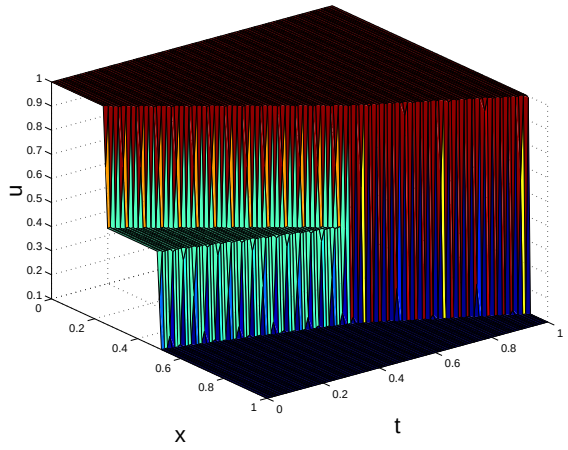

Figure 1: $u(x,t)$ for Burgers' equation, $\epsilon = 10^{-4}$.

The solution begins with two wavefronts. They move from left to right and merge to form one wavefront. As mentioned in [22], the thickness of the wavefronts is $O(\epsilon)$. The codes will have to quickly adapt the initial uniform mesh to the presence of the two distinct wavefronts and then as time advances, the codes will have to adapt the meshes to track the movement of these wavefronts as they merge. Figure 2 shows the performance of the codes (with various p values) for Burgers' equation with $\epsilon = 10^{-4}$. The L^2-norm error, (5), at $T_{out} = 1$, is plotted against the CPU time, both in logarithmic scales.

We make the following observations.

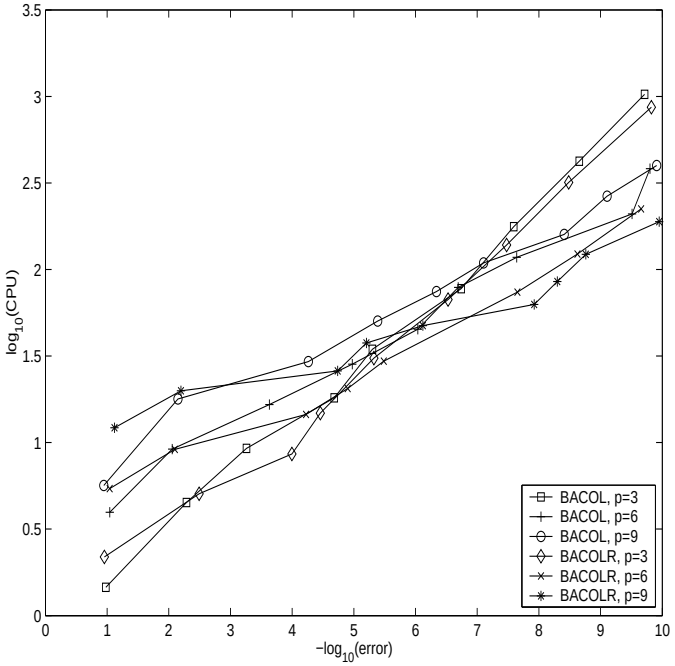

Figure 2: L^2-norm error vs. CPU time for Burgers' equation, $\epsilon = 10^{-4}$.

- When we choose $p = 3$, BACOL and BACOLR have almost identical performance. However when we choose $p = 6$ or $p = 9$, BACOLR is about twice as fast as BACOL. Our examination of the details of the computation shows that this difference in performance is primarily attributable to the time integrators.

- We observe that, for a given tolerance, both codes are able to obtain an approximate solution whose error is consistent with the tolerance. For example from Figure 2, we can see that when BACOL and BACOLR are given a tolerance of 10^{-10} the observed error is approximately 10^{-10}.

- While we were conducting this set of experiments we observed that both BACOL and BACOLR employ spatial meshes that are quite similar in the number of points employed and in the distribution of those points. The meshes generated by the codes successfully track the moving wavefronts.

3.2 A Catalytic Surface Reaction Model

The second parabolic problem is taken from [23]; it represents a reaction-diffusion-convection system for modelling a catalytic surface reaction. We include this problem because it allows us to investigate code performance on a

system of PDEs exhibiting challenging solution behaviour. The system has the form

$$\begin{aligned}
(u_1)_t &= -(u_1)_x + n(D_1 u_3 - A_1 u_1(1-u_3-u_4)) + \frac{1}{Pe_1}(u_1)_{xx}, \\
(u_2)_t &= -(u_2)_x + n(D_2 u_4 - A_2 u_2(1-u_3-u_4)) + \frac{1}{Pe_1}(u_2)_{xx}, \\
(u_3)_t &= A_1 u_1(1-u_3-u_4) - D_1 u_3 - R u_3 u_4 (1-u_3-u_4)^2 + \frac{1}{Pe_2}(u_3)_{xx}, \\
(u_4)_t &= A_2 u_2(1-u_3-u_4) - D_2 u_4 - R u_3 u_4 (1-u_3-u_4)^2 + \frac{1}{Pe_2}(u_4)_{xx},
\end{aligned}$$

where $0 < x < 1$ and $t > 0$, with initial conditions

$$u_1(x,0) = 2 - r, \quad u_2(x,0) = r, \quad u_3(x,0) = u_4(x,0) = 0, \qquad 0 < x < 1,$$

and boundary conditions

$$\frac{1}{Pe_1}(u_1)_x(0,t) = -(2-r-u_1), \quad \frac{1}{Pe_1}(u_2)_x(0,t) = -(r-u_2), \qquad t > 0,$$

$$(u_3)_x(0,t) = (u_4)_x(0,t) = 0, \qquad t > 0,$$

$$(u_1)_x(1,t) = (u_2)_x(1,t) = (u_3)_x(1,t) = (u_4)_x(1,t) = 0, \qquad t > 0,$$

where $u_1(x,t)$ and $u_2(x,t)$ are nondimensionalized concentrations, $u_3(x,t)$ and $u_4(x,t)$ are coverage of adsorbed reactants on the catalytic wall, Pe_1 and Pe_2 are Peclet numbers, and D_1, D_2, R, A_1 and A_2 are Damkohler numbers. We choose $A_1 = A_2 = 30$, $D_1 = 1.5$, $D_2 = 1.2$, $R = 1000$, $r = 0.96$, $n = 1$ and $Pe_1 = Pe_2 = 100$.

The exact solution is unknown for this problem. We therefore solve this problem with a very sharp tolerance ($TOL = 10^{-13}$) using BACOL in order to obtain a sufficiently accurate solution approximation which serves as the "exact" solution for use in (5). High-precision numerical approximations for the solution components are shown in Figures 3–4. (Note that in order to better display the solution behavior, we have reversed the x and t axes orientation from that used in Figure 1.)

While it is somewhat difficult to fully perceive detailed solution behavior from these figures, one can observe that there is significant solution variability in both time and space. The codes will need to immediately adapt the initial mesh to respond to the boundary layers in the initial solution configuration, and then will have to frequently adjust these meshes to react to changing solution behaviour as time advances.

Figure 5 shows the performance of the codes for this problem. The L^2-norm error is computed at $T_{out} = 18$. We first calculate the L^2-norm error for each PDE component, and then the maximum L^2-norm error over all PDE components is plotted.

We make the following observations:

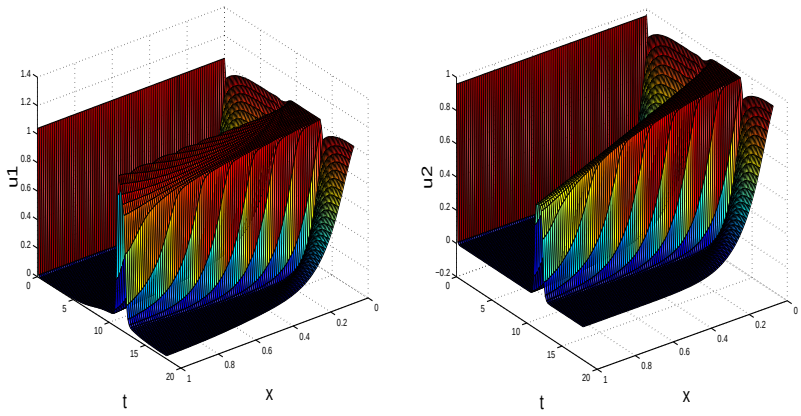

Figure 3: $u_1(x,t)$, $u_2(x,t)$ for catalytic surface reaction model.

- For a given tolerance, the solutions returned by BACOL are not as accurate as those returned by BACOLR. For example, when we consider $p = 3$ and $TOL = 10^{-7}$, the maximum L^2-norm error for BACOLR is about 4×10^{-7} while for BACOL it is about 2×10^{-5}. Similarly, for all choices of p, and a tolerance of 10^{-10}, BACOL returns a solution whose error is only about 10^{-7}. *The poorer performance by BACOL occurs because DASSL does not compute a solution to the DAEs whose error satisfies the requested time tolerance.* We observed experimentally that by giving DASSL a sharper time tolerance, which forces it to return a solution with a smaller time error, we could get BACOL to return a solution that meets the requested TOL value.

- For errors in the range from approximately 10^{-1} to 10^{-7}, for $p = 3$, BACOLR is a little more efficient than BACOL. For $p = 6$, the codes are comparable, except for very coarse tolerances, where BACOL is faster. For $p = 9$, BACOLR is slightly slower than BACOL.

- We again observed that BACOL and BACOLR employ similar spatial meshes which quickly adapt to the boundary layers that are present at the initial time and then are able to place points in appropriate places to respond to significant changes in solution behaviour.

3.3 The 1-D Nonlinear Cubic Schrödinger Equation

The one-dimensional nonlinear cubic Schrödinger equation is given by

$$u_t = i\left(u_{xx} + q|u|^2 u\right), \qquad -\infty < x < \infty, \qquad t > 0, \tag{6}$$

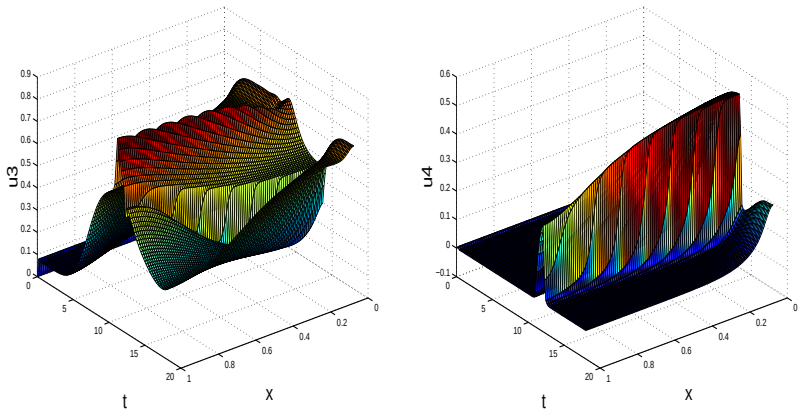

Figure 4: $u_3(x,t)$, $u_4(x,t)$ for catalytic surface reaction model.

with the initial condition

$$u(x,0) = g(x), \qquad -\infty < x < \infty, \tag{7}$$

where $i^2 = -1$, q is a positive constant, $g(x)$ is a complex function, and $|g(x)| \to 0$ as $|x| \to \infty$. This problem is taken from [27].

Since this problem is defined over the entire real line, in order to compute the approximate solution numerically, we restrict x to a finite interval $[a,b]$ with a and b chosen so that $u(x,t)$ is negligible for x outside $[a,b]$. By imposing Dirichlet boundary conditions at $x = a$ and $x = b$, we convert the initial value problem (6) into an initial-boundary value problem of the form

$$u_t = i\,(u_{xx} + q|u|^2 u), \qquad a \le x \le b, \quad t \ge 0, \tag{8}$$

$$u(x,0) = g(x), \qquad a \le x \le b, \tag{9}$$

$$u(a,t) = u(b,t) = 0, \qquad t > 0. \tag{10}$$

In our numerical experiments we choose $q = 8$. The function $g(x)$ appearing in the initial condition is chosen so that the exact solution is given by

$$u(x,t) = e^{it}\mathrm{sech}(x)\left[\frac{1 + \frac{3}{4}\mathrm{sech}^2(x)(e^{8it} - 1)}{1 - \frac{3}{4}\mathrm{sech}^4(x)\sin^2(4t)}\right].$$

The spatial interval used is $[-40, 40]$. The exact solution is plotted in Figure 6 for $-40 \le x \le 40$, $0 \le t \le 2.5$. (Note that in order to better display the solution behavior, we have reversed the x and t axes orientation from that used in Figure 1.)

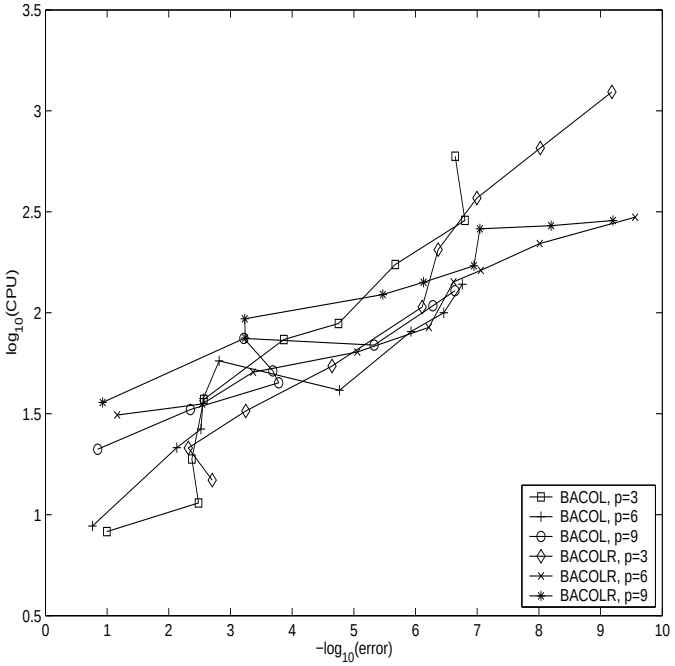

Figure 5: Maximum L^2-norm error vs. CPU time for catalytic surface reaction model.

We observe for this problem that the codes will have to quickly adapt the uniform initial mesh to focus points near the origin. However, after this is accomplished, no further adaptation of the mesh for later times will be necessary.

Standard numerical approaches for the spatial discretization of the 1-D nonlinear cubic Schrödinger equation include finite difference, finite element, and orthogonal spline collocation methods; see, e.g., [28, 27, 30]. These spatial discretizations have been reasonably successful, particularly when coupled with user-provided graded meshes which focus points near the origin. However, despite many studies of the time integration schemes in the literature (e.g., [15, 29]), none of these schemes have proven to be entirely satisfactory. While there have been a plethora of methods proposed for the numerical solution of Schrödinger problems - see, e.g., [12] - to our knowledge this research has not led to the development of a robust software package for the treatment of these problems.

For simpler systems of this type and for simple spatial discretization schemes it has been proven that the resultant DAE systems have Jacobians which have eigenvalues on the imaginary axis, and it appears that this property carries over to more general Schrödinger problems. This means that one needs to consider time integration schemes whose stability regions include the imaginary axis. As

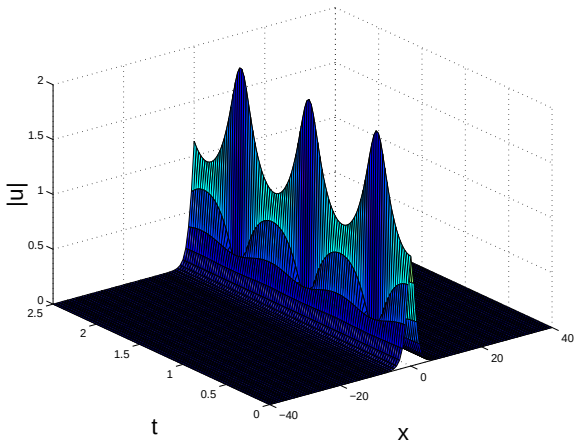

Figure 6: $|u(x,t)|$ for cubic Schrödinger equation.

mentioned earlier, higher order BDF methods (of order greater than 2), such as those employed by DASSL, will not perform well; see, e.g., [14]. On the other hand, RADAU5 employs a fifth-order Runge-Kutta scheme which is A-stable, implying that its stability region does include the imaginary axis.

In order to treat this problem using the BACOL and BACOLR codes, we first decompose the complex function $u(x,t)$ into its real and imaginary parts $v(x,t)$ and $w(x,t)$, respectively. Equations (8)–(10) become

$$
\begin{aligned}
v_t &= -w_{xx} - 8(v^2+w^2)w, \qquad -40 \le x \le 40, \quad t \ge 0, \\
w_t &= v_{xx} + 8(v^2+w^2)v, \qquad -40 \le x \le 40, \quad t \ge 0, \\
v(x,0) &= \mathrm{sech}(x), \quad w(x,0) = 0, \qquad -40 \le x \le 40, \\
v(-40,t) &= v(40,t) = w(-40,t) = w(40,t) = 0, \qquad t > 0.
\end{aligned}
$$

The L^2-norm error of the complex function $u(x,t) = v(x,t) + i\,w(x,t)$, is defined as

$$
\begin{aligned}
\|U(x,t) - u(x,t)\|_2 &= \sqrt{\int_a^b (U(x,t) - u(x,t))^2 \, dx} \\
&= \sqrt{\int_a^b ((V(x,t) - v(x,t))^2 + (W(x,t) - w(x,t))^2) \, dx},
\end{aligned}
$$

where $U(x,t) = V(x,t) + i\,W(x,t)$ is the approximate solution.

In our initial experiments on this problem, we found that BACOL was unable to obtain a solution. This is expected because, as mentioned above, the stability regions of the BDF methods whose orders are greater than or equal to 3 do not include the imaginary axis. Therefore, in order to continue with this experiment,

we employed an option within DASSL that allows one to restrict the BDF methods that the code uses to be of at most order 2.

Figure 7 shows the performance of the two codes for the cubic Schrödinger equation at $T_{out} = 2.5$, with BACOL using DASSL in an order restricted mode.

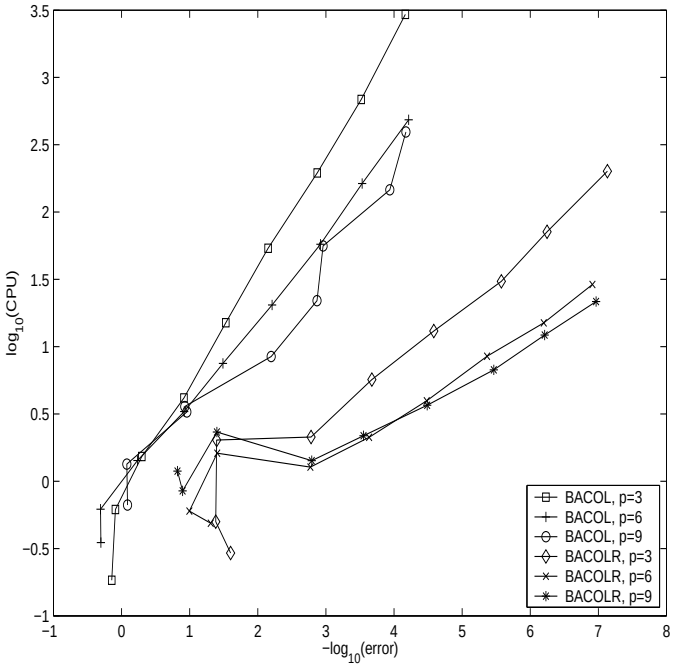

Figure 7: L^2-norm error vs. CPU time for cubic Schrödinger equation; BACOL is run with DASSL in order restricted mode.

We make the following observations:

- BACOL fails on this problem unless it is run with DASSL in a order restricted mode (orders 1 and 2 only).
- BACOLR is substantially more efficient than BACOL for this problem. This is because of the advantage of the higher order time integration scheme of RADAU5 over the lower order ones of the order restricted DASSL.
- We note that for a given tolerance, BACOLR is able to generate a much more accurate solution than BACOL. For example, we can see that for a tolerance of 10^{-10} BACOL gives a solution whose error is about 10^{-4} while BACOLR gives a solution whose error is about 10^{-7}. We note that neither code is able to meet the requested tolerance; *in each case this is*

due to the fact that the time integrator is not able to return a solution to the DAEs which satisfies the given time tolerance. As for the previous problem, we observed that we could get both codes to return solutions that satisfy the requested TOL value by setting the time tolerance to be sharper than the spatial tolerance. This forces both DAE solvers to return solutions that have errors that are in fact less than the requested time tolerance.

- As in previous experiments, we observed that BACOL and BACOLR employ similar spatial meshes. The codes quickly adapt the meshes to place a sufficient number of points near the origin and then maintain this mesh for the remainder of the computation.

3.4 A Coupled Nonlinear Schrödinger System

This system is given by

$$\begin{aligned}(u_1)_t &= i\left(\frac{1}{2}(u_1)_{xx} + \eta\,(u_1)_x + (|u_1|^2 + \rho\,|u_2|^2)\,u_1\right),\\ (u_2)_t &= i\left(\frac{1}{2}(u_2)_{xx} - \eta\,(u_2)_x + (\rho\,|u_1|^2 + |u_2|^2)\,u_2\right),\end{aligned}$$

where $i^2 = -1$ and η and ρ are positive constants. The boundary conditions are

$$(u_1)_x(a,t) = (u_2)_x(a,t) = 0, \quad (u_1)_x(b,t) = (u_2)_x(b,t) = 0, \quad t > 0,$$

where $a = -30$ and $b = 90$, again chosen so that the solution values outside this interval are negligible. The initial conditions are

$$u_1(x,0) = g_1(x), \quad u_2(x,0) = g_2(x), \quad a \le x \le b,$$

where $g_1(x)$ and $g_2(x)$ are chosen so that the modulus for each of the exact solutions is a soliton and the exact solutions are given by

$$\begin{aligned}u_1(x,t) &= \sqrt{\frac{2\kappa}{1+\rho}}\,\mathrm{sech}\left(\sqrt{2\kappa}(x-\phi t)\right)e^{i((\phi-\eta)x-(\frac{\phi^2-\eta^2}{2}-\kappa)t)},\\ u_2(x,t) &= \sqrt{\frac{2\kappa}{1+\rho}}\,\mathrm{sech}\left(\sqrt{2\kappa}(x-\phi t)\right)e^{i((\phi+\eta)x-(\frac{\phi^2-\eta^2}{2}-\kappa)t)},\end{aligned}$$

where κ is a constant and ϕ represents the speed of the soliton. In our numerical experiments we choose $\phi = 1$, $\eta = 1/2$, $\rho = 2/3$ and $\kappa = 1$. The modulus of $u_1(x,t)$ is plotted at $t = 0, 10, 20, \ldots, 50$ in Figure 8.

We chose this problem because the location of the sharp peak moves with time thus providing more of a challenge to the mesh selection algorithms of the codes. We observe that it will be necessary for the codes to quickly adapt the initial uniform mesh to place points in the region of the initial peak and then, for

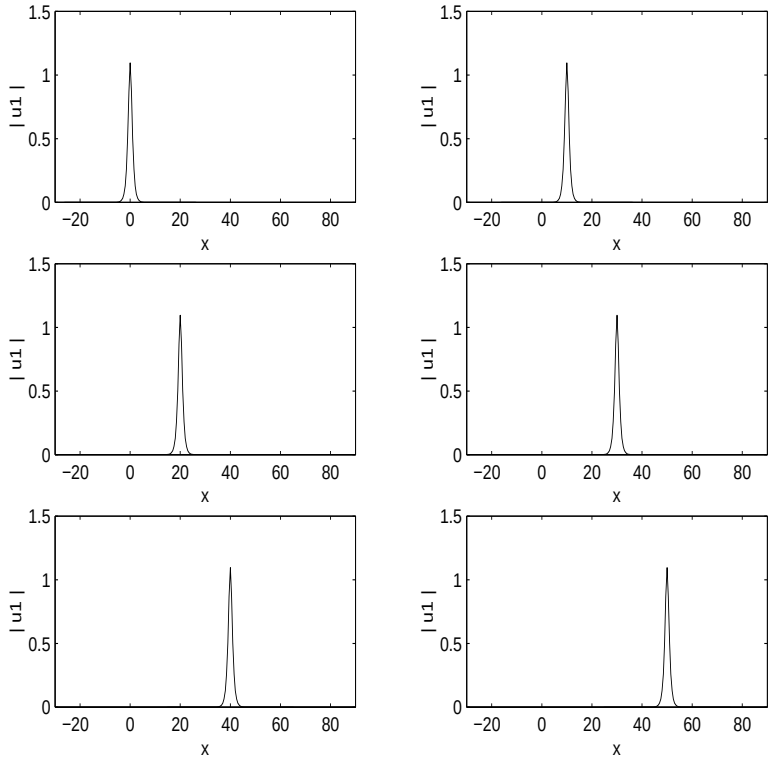

Figure 8: $|u_1(x,t)|$ for coupled Schrödinger system.

later times, these points will have to move to track the motion of the peak across the domain. As we did for the previous problem, we first decompose $u_1(x,t)$ and $u_2(x,t)$ into their real and complex parts, obtaining a four component system.

This problem is taken from [17], where the authors treat it by using non-adaptive methods, i.e. a fixed spatial mesh and fixed time steps. Neumann boundary conditions are employed and second order central differences are used for the spatial discretization. The implicit mid-point rule is used for the time integration of the resultant IVODE system. Figure 9 shows the performance of BACOL and BACOLR for this problem at $T_{out} = 5$.

We have the following observations:

- As was the case for the previous Schrödinger problem, BACOL cannot handle this problem unless it is run with DASSL in an order restricted mode (orders 1 and 2 only).
- We observe that BACOLR is substantially more efficient than BACOL for all p values and for errors less than about 10^{-2}.

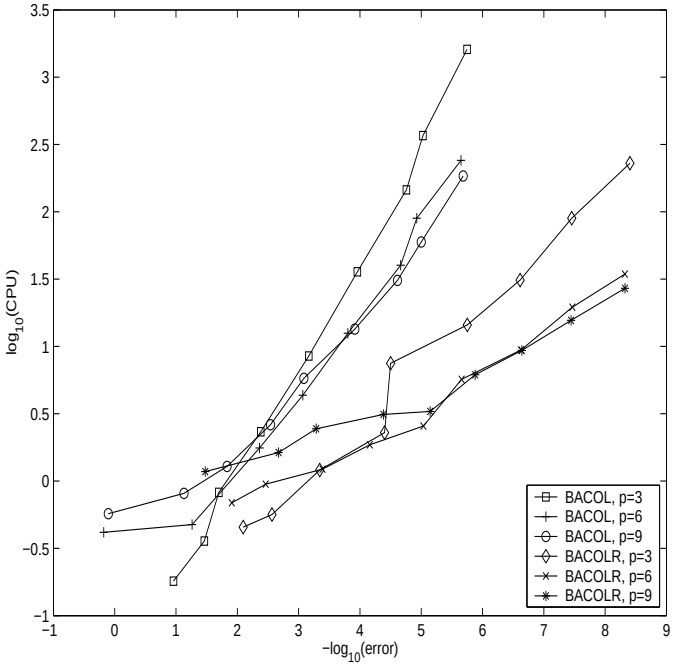

Figure 9: Maximum L^2-norm error vs. CPU time for coupled Schrödinger system. BACOL is run with DASSL in order restricted mode.

- For a given tolerance, BACOLR is able to generate a much more accurate solution than BACOL. For example, when the tolerance is 10^{-10} BACOL computes a solution whose error is about 10^{-6} while BACOLR returns a solution whose error is about 5×10^{-9}. The amount by which each code fails to meet the requested tolerance is attributable to the fact that its time integrator is not able to meet the time tolerance. This was verified by running BACOL and BACOLR with sharper time tolerances provided to their respective DAE solvers and observing that the codes were then able to return solutions which met the requested TOL value.

- For this problem BACOLR is able to quickly adapt the initial mesh to the location of the peak and then is able to move points to track the location of the peak across the domain. BACOL is not able to handle this problem as well and tends to use meshes that adapt to some extent to the presence of the peak as it moves across the domain but have too many points.

4 Summary, Conclusions and Future Work

BACOLR was obtained from BACOL through a substantial modification to replace DASSL with RADAU5; this process also involved a substantial modification of the RADAU5 package, including the addition of a capability for the efficient treatment of ABD linear systems. Extensive testing on parabolic problems shows that BACOLR is generally more efficient than BACOL. This advantage is greater for Schrödinger problems where BACOL is forced to use DASSL in an order restricted mode which significantly reduces its efficiency.

We observed that for a single parabolic PDE, both BACOL and BACOLR are able to satisfy a requested tolerance. This is consistent with other extensive numerical testing we have performed; we have found that when we consider a single parabolic equation both codes are able to obtain an approximate solution whose error is consistent with the tolerance.

For the system of parabolic PDEs considered in this paper, we have noted that BACOLR is able to meet a requested tolerance but BACOL is generally not. This is because RADAU5 returns a solution to the DAEs which satisfies the requested time tolerance while DASSL does not. We have also conducted extensive numerical testing using other codes [32]. For example, when we apply HPNEW to the first and second test problems in this paper we see similar performance; the code usually returns a solution whose error satisfies the requested tolerance for a single equation but is not able to do so for systems. When we employ EPDCOL, the fixed-mesh software package mentioned previously, which employs collocation in space and BDF methods for the time integration to solve systems of equations, we also found that it is unable to obtain an approximate solution whose error is close to the requested time tolerance *no matter how many points we use on the spatial domain*. In each of these cases the initial value solver is unable to compute a solution whose error is close to the given tolerance. We have also observed this phenomenon for other parabolic PDE systems. We can thus make the important observation that RADAU5 seems to be more effective in controlling the temporal error for discretizations of PDE *systems*.

For Schrödinger systems we saw that both BACOL and BACOLR (to a lesser extent) have difficulty in returning solutions whose error is less than the given tolerance. In each case this is because the DAE solver is not able to return a solution that satisfies the given time tolerance. We have found that for all of the problems considered in this paper we can address this difficulty and get BACOL and BACOLR to satisfy requested tolerances, *provided we set the time tolerance sufficiently small compared to the spatial tolerance*. This forces the DAE solver to compute the solution to higher accuracy and it then returns a solution whose error is in fact less than the spatial tolerance. However, in some cases this significantly increases the cost of the computation. (For a different class of problems, treated by the MOL, the paper [20] investigates similar behavior for the code DASPK [8], which is derived from DASSL.)

Future work involves further investigation of the relationship between the spatial tolerance and the time tolerance. Further investigation of the behaviour

of time integrators applied to DAE systems whose Jacobians have pure imaginary eigenvalues is needed. Another possibility for future work follows from the observation that RADAU5 has been generalized to a variable order code, RADAU, based on Radau-type implicit Runge-Kutta methods of orders 5, 9, and 13 [13]. It would be interesting to modify BACOLR to replace RADAU5 with this variable order time integrator.

The computation of a second global solution for the error estimate represents a significant cost in BACOL and BACOLR and we are currently considering alternative low cost error estimates. Generalization of the BACOLR code to handle two-dimensional problems would also be interesting.

Acknowledgments

The authors would like to thank Graeme Fairweather for helpful discussions regarding the numerical solution of Schrödinger problems and for his comments on an early draft of this paper.

References

[1] S. Adjerid, M. Aiffa and J.E. Flaherty, High-order finite element methods for singularly-perturbed elliptic and parabolic problems, SIAM J. Appl. Math., 55 (1995), 520–543.

[2] P. Amodio, J.R. Cash, G. Roussos, R.W. Wright, G. Fairweather, I. Gladwell, G.L. Kraut, and M. Paprzycki, Almost block diagonal linear systems: sequential and parallel solution techniques, and applications, Numer. Linear Algebra Appl., 7 (2000), 275–317.

[3] M. Berzins, P.J. Capon and P.K. Jimack, On spatial adaptivity and interpolation when using the method of lines, Appl. Num. Math., 26 (1998), 117–133.

[4] M. Berzins and P.M. Dew, Algorithm 690: Chebyshev polynomial software for elliptic-parabolic systems of PDEs, ACM Trans. Math. Softw., 17 (1991), 178–206.

[5] J.G. Blom, J.M. Sanz-Serna and J.G. Verwer, On simple moving grid methods for one-dimensional evolutionary partial differential equations, J. Comput. Phys., 74 (1988), 191–213.

[6] J.G. Blom and P.A. Zegeling, Algorithm 731: A moving-grid interface for systems of one-dimensional time-dependent partial differential equations, ACM Trans. Math. Softw., 20 (1994), 194–214.

[7] C. de Boor, Package for calculating with B-Splines, SIAM J. Numer. Anal., 14 (1977), 441–472.

[8] 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 (1994), 1467–1488.

[9] K.E. Brenan, S.L. Campbell and L.R. Petzold, Numerical solution of initial-value problems in differential-algebraic equations, SIAM, Philadelphia, 1996.

[10] D03PPF, NAG Fortran library, Mark 16A. The Numerical Algorithms Group, Ltd. Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, U.K.

[11] J.C. Diaz, 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. Softw., 9 (1983), 358–375.

[12] G. Fairweather and M. Khebchareon, Numerical methods for Schrödinger-type problems, in Trends in Industrial and Applied Mathematics, A.H. Siddiqi and M. Kocvara (eds.), Kluwer Academic Publishers, Netherlands, 2002, 219–250.

[13] E. Hairer, RADAU, software based on implicit Runge-Kutta methods (Radau IIA type), of variable order (switches automatically between orders 5, 9, and 13) for problems of the form My'=f(x,y) with possibly singular matrix M, unpublished software, http://www.unige.ch/∼hairer/prog/stiff/radau.f

[14] E. Hairer and G. Wanner, Solving ordinary differential equations II Stiff and differential-algebraic problems, Second edition, Springer Series in Computational Mathematics, 14, Springer-Verlag, Berlin, 1996.

[15] B.M Herbst, J.L. Morris and A.R. Mitchell, Numerical experience with the nonlinear Schrödinger equation, J. Comput. Phys., 60 (1985), 282–305.

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

[17] M.S. Ismail and T.R. Taha, Numerical simulation of coupled nonlinear Schrödinger equation, Math. Comput. Simulation, 56 (2001), 547–562.

[18] P. Keast, a complex version of the COLROW package, unpublished software, http://www.mscs.dal.ca/∼keast/research/pubs.html.

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

[20] C.T. Kelley, C.T. Miller, and M.D. Tocci, Termination of the Newton/Chord iterations and the method of lines, SIAM J. Sci. Comput., 19 (1998), 280–290.

[21] N.K. Madsen and R.F. Sincovec, Algorithm 540. PDECOL, General collocation software for partial differential equations, ACM Trans. Math. Softw., 5 (1979), 326–351.

[22] K. Miller and R. N. Miller, Moving finite elements. I, SIAM J. Numer. Anal., 18 (1981), 1019–1032.

[23] P.K. Moore, Comparison of adaptive methods for one dimensional parabolic systems, Appl. Numer. Math., 16 (1995), 471–488.

[24] 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 (2001), 149–177.

[25] L.R. Petzold, A description of DASSL: a differential/algebraic system solver, Sandia Labs, Livermore, CA, 1982.

[26] L.R. Petzold and P. Lötstedt, Numerical solution of nonlinear differential equations with algebraic constraints II: practical implications, SIAM J. Sci. Stat. Comput., 7 (1986), 720–733.

[27] M.P. Robinson and G. Fairweather, Orthogonal spline collocation methods for Schrödinger-type equations in one space variable, Numer. Math., 68 (1994), 355–376.

[28] M.P. Robinson, G. Fairweather and B.M Herbst, On the numerical solution of the cubic Schrödinger equation in one space variable, J. Comput. Phys. 104 (1993), 277–284.

[29] J.M. Sanz-Serna and J.G. Verwer, Conservative and nonconservative schemes for the solutions of the nonlinear Schrödinger equation, IMA J. Numer. Anal. 6 (1986), 25–42.

[30] E.H. Twizell, A.G. Bratsos and J.C. Newby, A finite-difference method for solving the cubic Schrödinger equation, Math. Comput. Simulation, 43 (1997), 67–75.

[31] 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 (2004), 239–260.

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

[33] R. Wang, P. Keast and P.H. Muir, BACOL: B-spline Adaptive COLlocation software for 1-D parabolic PDEs, ACM Trans. Math. Softw., 30 (2004), 454–470.