Numerical Algorithms (2019) 81:1505–1519
https://doi.org/10.1007/s11075-019-00738-2

ORIGINAL PAPER

Error control Gaussian collocation software for boundary value ODEs and 1D time-dependent PDEs

Mark Adams[1] · Connor Tannahill[1] · Paul Muir[1]

Received: 27 June 2018 / Accepted: 27 May 2019 / Published online: 6 June 2019

Abstract
Error control software packages based on Gaussian collocation have been widely used for the numerical solution of boundary value ODEs (BVODEs) and 1D parabolic time-dependent PDEs (1D PDEs) for several decades. These robust and efficient packages are among the best available for these problem classes. In this paper, we survey error control Gaussian collocation software for BVODEs and 1D PDEs and provide an overview of recent work involving the development of two new packages, one for each problem class. The first is an updated version of the well-known COLSYS/COLNEW package for BVODEs. The second is the newest member of the BACOL family of software packages for 1D PDEs. We briefly review the underlying numerical algorithms employed in these packages and then provide numerical results that show the superiority of the new packages compared to previously released packages from each software class.

Keywords Boundary value ordinary differential equations · Partial differential equations · Collocation · Error estimation · Error control · Efficiency

Mathematics Subject Classification (2010) 65L06 · 65L10 · 65L80 · 65M20 · 65M70

1 Introduction

For boundary value ODEs (BVODEs) and 1D parabolic time-dependent PDEs (1D PDEs), error control software based on Gaussian collocation algorithms has been widely used for several decades. In this paper, we survey error control Gaussian collocation software for BVODEs and 1D PDEs and provide a brief overview of

✉ Paul Muir
muir@smu.ca

1 Saint Mary's University, Halifax, NS B3H 3C3, Canada

recent work involving the development of two new error control Gaussian collocation packages, one for each problem class.

Error control is an essential element of all high-quality numerical software. Error control software returns a numerical solution for which an associated error estimate satisfies a user-prescribed tolerance. Such software has two important advantages: (i) the user can have reasonable confidence that the numerical solution has an error that is consistent with the requested tolerance, and (ii) the cost of the computation will be consistent with the requested tolerance.

In this paper, we assume systems of BVODEs having the general form

$$\mathbf{u}'(x) = \mathbf{f}\left(x, \mathbf{u}(x)\right), \quad a < x < b, \quad \mathbf{u} : \Re \to \Re^n, \quad \mathbf{f} : \Re \times \Re^n \to \Re^n, \tag{1}$$

with separated boundary conditions,

$$\mathbf{b}_L\left(\mathbf{u}(a)\right) = \mathbf{0}_L, \quad \mathbf{b}_R\left(\mathbf{u}(b)\right) = \mathbf{0}_R, \quad \mathbf{b}_L : \Re^n \to \Re^{n_L}, \mathbf{b}_R : \Re^n \to \Re^{n_R}, \tag{2}$$

where $\mathbf{0}_L \in \Re^{n_L}$, $\mathbf{0}_R \in \Re^{n_R}$, and $n_L + n_R = n$.

For 1D PDEs, we will assume systems having the general form,

$$\mathbf{u}_t(x, t) = \mathbf{f}\left(x, t, \mathbf{u}(x, t), \mathbf{u}_x(x, t), \mathbf{u}_{xx}(x, t)\right), \quad a < x < b, \quad t \geq t_0, \tag{3}$$

where $\mathbf{u} : \Re \times \Re \to \Re^n$ and $\mathbf{f} : \Re \times \Re \times \Re^n \times \Re^n \times \Re^n \to \Re^n$, with separated boundary conditions,

$$\mathbf{b}_L\left(t, \mathbf{u}(a, t), \mathbf{u}_x(a, t)\right) = \mathbf{0}, \quad \mathbf{b}_R\left(t, \mathbf{u}(b, t), \mathbf{u}_x(b, t)\right) = \mathbf{0}, \quad t \geq t_0, \tag{4}$$

where $\mathbf{b}_L, \mathbf{b}_R : \Re \times \Re^n \times \Re^n \to \Re^n$, $\mathbf{0} \in \Re^n$, and initial conditions,

$$\mathbf{u}(x, t_0) = \mathbf{u}_0(x), \qquad a < x < b, \quad \mathbf{u_0} : \Re \to \Re^n. \tag{5}$$

For systems of BVODEs, the earliest package to implement Gaussian collocation within an error control framework is COLSYS [4], which was released about four decades ago. The computation is based on a mesh of points that partitions $[a, b]$ and the approximate solution is represented in terms of a B-spline basis [11]. The system of nonlinear equations arising from the collocation process together with the boundary conditions is solved using a Newton-type algorithm to obtain the B-spline coefficients. Control of an estimate of the error of the approximate solution is provided through adaptive mesh refinement based on equidistribution of estimates of the global error for each subinterval. About three decades ago, COLNEW [7], an update of COLSYS, was released; the primary modifications were the replacement of the B-spine basis with a monomial basis and corresponding changes to the linear algebra algorithms. Both COLSYS and COLNEW are able to directly treat a generalization of (1) and (2), known as a mixed order system (see [5] for further details); however, in this paper, we consider only the standard first-order system form (1) and (2). We will discuss the algorithms employed in COLSYS/COLNEW in more detail in the next section.

In subsequent years, a number of packages, obtained through modifications of COLNEW, have been released; these include COLMOD [10], which updates the mesh refinement algorithm and introduces automatic parameter continuation, and COLDAE [6], which extends the problem class to include BVODEs with coupled algebraic constraints. More recent work has seen interfaces to COLNEW developed in Scilab [22], Python [24], and R [23].

For 1D PDEs, PDECOL [18], released about four decades ago, is the earliest Gaussian collocation package to provide any form of error control. PDECOL assumes a mesh that partitions $[a, b]$ and represents the approximate solution using a B-spline basis for the spatial dependence with the time dependence represented in the coefficients of the B-spline basis functions. The Gaussian collocation process discretizes the PDEs, yielding a system of ODEs. The user must provide the time derivatives of the boundary conditions; these are a set of ODEs. These ODEs, together with those arising from the collocation process, are solved using an ODE solver [18], which provides control of the temporal error. PDECOL does not provide control of the spatial error.

Almost three decades ago, a modification of PDECOL, called EPDCOL [16] was developed. EPDCOL replaces the banded linear system solver employed by PDECOL with an almost block diagonal (ABD) solver, COLROW [12], which can efficiently solve the ABD linear systems that arise from the B-spline collocation spatial discretization. In [16], EPDCOL was shown to be about twice as fast as PDECOL. EPDCOL also does not provide control of the spatial error.

The next 1D PDE package implementing Gaussian collocation, BACOL [26, 28], was developed about 15 years ago. This package features control of both the spatial and temporal error. It uses the same representation for the approximate solution that is used in PDECOL and EPDCOL. However, the boundary conditions are treated directly and, these equations, together with the ODEs arising from the collocation discretization, represent a system of index-1 differential algebraic equations (DAEs). The temporal error–controlled computation of the B-spline coefficients is performed by DASSL [9], which is based on backward differentiation formulas (BDFs). Spatial error control is provided through spatial mesh refinement based on equidistribution of computed spatial error estimates. In a comparison with several other packages for 1D PDEs, BACOL was shown to provide superior performance, especially for problems with solutions exhibiting sharp moving layers and for sharp tolerances [27].

Approximately a decade ago, BACOLR [25], a modification of BACOL, was developed. In BACOLR, DASSL is replaced with RADAU5 [15], a DAE solver which is based on a fifth-order implicit Runge-Kutta method of Radau IIA type. Numerical comparisons of BACOL and BACOLR show that the two codes perform similarly on standard test problems and that BACOLR has much superior performance on problems for which the stability of the higher order BDFs is an issue. The stability regions of the higher order BDFs do not include the imaginary axis and thus problems that lead to DAEs having Jacobians with eigenvalues near the imaginary axis, for example, Schrödinger type problems, cannot be treated using the higher order BDFs. The paper [25] shows that BACOL fails on problems of this type unless DASSL is restricted to using only lower order BDFs, in which case the efficiency of the computation is substantially degraded.

The recently released package, BACOLI [20], is a modification of BACOL that improves on the efficiency of the BACOL spatial error estimation algorithm by replacing it with a pair of new interpolation–based spatial error estimation schemes. In [20], BACOLI is shown to be approximately twice as efficient as BACOL. We describe the algorithms employed in BACOL, BACOLR, and BACOLI in more detail in Section 3.

The two new error control Gaussian collocation packages introduced in this paper are called COLNEW95 and BACOLRI. COLNEW95 is a modification of COLNEW that updates it to Fortran 95 while introducing several new features that substantially improve the efficiency of the computation. The primary modifications are the introduction of a superconvergent interpolant to augment the collocation solution and new error estimation and mesh refinement algorithms. BACOLRI is the newest member of the BACOL family of error control 1D PDE solvers. It updates BACOLR to use the new interpolation–based spatial error estimation schemes that were introduced in BACOLI.

The results of this work will lead to improved error control Gaussian collocation software for BVODEs and 1D PDEs and will contribute to ongoing work in the development of error control Gaussian collocation software for 2D PDEs [17].

This paper is organized as follows. In Sections 2 and 3, we review the algorithms implemented in COLSYS/COLNEW and BACOL, BACOLR, and BACOLI, respectively. Sections 4 and 5, respectively, give brief overviews of COLNEW95 and BACOLRI, along with some numerical comparisons that demonstrate the superiority of these two new solvers. We close, in Section 6, with our conclusions and suggestions for future work.

2 Error control Gaussian collocation software for BVODEs: COLSYS/COLNEW

Let $\{x_i\}_{i=0}^{N}$ be a set of mesh points partitioning $[a, b]$ with,

$$a = x_0 < x_1 < \ldots < x_{N-1} < x_N = b.$$

Assume a C^0-continuous, degree p, B-spline basis defined on this mesh. In COLSYS, the approximate solution, $\mathbf{U}(x) : \Re \rightarrow \Re^n$, to (1) and (2), is expressed in the form,

$$\mathbf{U}(x) = \sum_{i=1}^{N_p} \mathbf{c}_{p,i} B_{p,i}(x), \tag{6}$$

where $\mathbf{c}_{p,i} \in \Re^n$ is an unknown vector coefficient for $B_{p,i}(x)$, the ith B-spline basis function of degree p (associated with the above mesh and continuity conditions), and $N_p = Np + 1$. As mentioned earlier, the primary difference between COLSYS and COLNEW is that in COLNEW the B-spline basis is replaced with a piecewise monomial basis; see [7] for further details.

Let $\left\{\rho_j\right\}_{j=1}^{p}$ be the points of the p-point Gauss-Legendre quadrature rule mapped onto [0, 1] and let $h_i = x_i - x_{i-1}$, $i = 1, \ldots, N$. The collocation points, ξ_m, $m = 1, \ldots, Np$, are given by,

$$\xi_m = x_{i-1} + h_i \rho_j, \quad m = (i-1)p + j, \quad j = 1, \ldots, p, \quad i = 1, \ldots, N. \tag{7}$$

The collocation equations are obtained by requiring that (6) satisfy (1) at the collocation points (7); this gives,

$$\mathbf{U}(\xi_m) = \mathbf{f}\left(\xi_m, \mathbf{U}(\xi_m)\right), \quad m = 1, \ldots, Np. \tag{8}$$

The approximate solution, (6), is also required to satisfy (2); this gives,

$$\mathbf{b}_L\left(\mathbf{U}(a)\right) = \mathbf{0}_L, \qquad \mathbf{b}_R\left(\mathbf{U}(b)\right) = \mathbf{0}_R. \tag{9}$$

The system, (8) and (9), is solved using a modified Newton iteration (see [5] for further details) in order to obtain $\mathbf{c}_{p,i}$, $i = 1, \ldots, N_p$.

Standard theory (see, e.g., [5]) for the Gaussian collocation solution of (1) states that the global error at $\{x_i\}_{i=0}^N$ satisfies,

$$\mathbf{U}(x_i) - \mathbf{u}(x_i) \sim O\left(h^{2p}\right), \quad i = 0, \ldots, N, \tag{10}$$

while the global error at a non-mesh point, $x \neq \{x_i\}_{i=0}^N$, satisfies,

$$\mathbf{U}(x) - \mathbf{u}(x) \sim O\left(h^{p+1}\right), \quad x \in [a, b].$$

For $p > 1$, the collocation solution is thus superconvergent at $\{x_i\}_{i=0}^N$.

The global error estimates, one for each mesh subinterval, are obtained using Richardson extrapolation (RE). This process requires that a collocation solution obtained on a mesh of N subintervals be subtracted from a collocation solution (of the same degree) computed on a mesh of $2N$ subintervals, where the second mesh is obtained by halving each subinterval of the first mesh. If the error estimates do not satisfy the tolerance, then the code will generate a new mesh and compute another collocation solution based on this new mesh. If the distribution of the error estimates over the subintervals is sufficiently non-uniform, then the error estimates are used as the basis for an equidistribution algorithm that can change the number of mesh points and their locations. Otherwise, the new mesh is obtained simply by halving each subinterval of the current mesh.

Since the decision to perform an equidistribution requires a fairly substantial non-uniform distribution of the error estimates, it is common for the next mesh to be obtained by halving each subinterval of the current mesh. This improves the efficiency of the RE error estimation computation because the code can use RE to compute an error estimate for the current collocation solution without having to compute a new collocation solution on a new mesh formed by halving each subinterval of the current mesh. While this is an advantage for the error estimate computation, a disadvantage is that the meshes are typically not particularly well adapted to the error estimates, since the equidistribution algorithm is only rarely invoked.

3 Error control Gaussian collocation software for 1D PDEs: BACOL, BACOLR, BACOLI

In this section, we will assume the same mesh, $\{x_i\}_{i=0}^N$, that was defined in the previous section but now will assume a C^1-continuous, degree p, B-spline basis defined on this mesh. In BACOL, BACOLR, and BACOLI, the approximate solution, $\mathbf{U}(x,t) : \Re \times \Re \to \Re^n$, is expressed in the form,

$$\mathbf{U}(x,t) = \sum_{i=1}^{N_p} \mathbf{c}_{p,i}(t) B_{p,i}(x), \tag{11}$$

where $\mathbf{c}_{p,i}(t) : \Re \to \Re^n$ is an unknown time–dependent vector coefficient of $B_{p,i}(x)$, the ith B-spline basis function (associated with the spatial mesh and spatial continuity conditions indicated above), and, in this case, $N_p = N(p-1) + 2$.

Let $\{\rho_j\}_{j=1}^{p-1}$ be the points of the $(p-1)$-point Gauss-Legendre quadrature rule mapped onto [0, 1]. The collocation points, ξ_m, $m = 2, \ldots, N(p-1)+1$, are given by,

$$\xi_m = x_{i-1} + h_i \rho_j, \quad m = (i-1)(p-1)+j+1, \quad j = 1, \ldots, p-1, \quad i = 1, \ldots, N. \tag{12}$$

For a given t, the coefficients, $\mathbf{c}_{p,i}(t)$, are determined by requiring that $\mathbf{U}(x,t)$ satisfy (3) at the collocation points (12). The collocation equations are,

$$\mathbf{U}_t(\xi_m, t) = \mathbf{f}\left(\xi_m, t, \mathbf{U}(\xi_m, t), \mathbf{U}_x(\xi_m, t), \mathbf{U}_{xx}(\xi_m, t)\right), m = 2, \ldots, N(p-1)+1. \tag{13}$$

The approximate solution, $\mathbf{U}(x,t)$, is also required to satisfy the boundary conditions; this gives,

$$\mathbf{b}_L\left(t, \mathbf{U}(a,t), \mathbf{U}_x(a,t)\right) = \mathbf{0}, \qquad \mathbf{b}_R\left(t, \mathbf{U}(b,t), \mathbf{U}_x(b,t)\right) = \mathbf{0}. \tag{14}$$

Standard theory states that the spatial error of $\mathbf{U}(x,t)$ is $O\left(h^{p+1}\right)$, where h is the spatial mesh spacing; see, e.g., [20] and references within.

As mentioned earlier, the system, (13) and (14), is solved using either DASSL or RADAU5, in order to perform a temporal error–controlled computation to obtain the B-spline coefficients. After each accepted time step, BACOL and BACOLR compute an estimate of the spatial error for (11). If this error estimate does not satisfy the tolerance, the numerical solution is rejected, and a spatial remeshing is performed, based on the principle of equidistributing the spatial error estimates over the spatial mesh subintervals. Both the number of mesh points and their locations may be changed in order to adapt to the magnitude (with respect to the user-specified tolerance) and distribution of the subinterval spatial error estimates over $[a, b]$. See [28] for further details. *The spatial error estimates are obtained by computing a second approximate solution,* $\overline{\mathbf{U}}(x,t)$, *on the same spatial mesh and at the same time* t, *but based on B-splines of degree* $p+1$, *and this essentially doubles the overall cost.*

This inefficiency is addressed in BACOLI which replaces the computation of $\overline{\mathbf{U}}(x,t)$ with the more efficient computation of a piecewise polynomial interpolant to $\mathbf{U}(x,t)$. There are two options for this interpolant. One, the *superconvergent interpolant (SCI)* [2], is based on interpolating $\mathbf{U}(x,t)$ and $\mathbf{U}_x(x,t)$ at the mesh points and $\mathbf{U}(x,t)$ at certain other points where it is superconvergent in space, i.e., the spatial error is at least one order higher than at an arbitrary point in the spatial domain. The resultant interpolant is of one order of spatial accuracy higher than $\mathbf{U}(x,t)$ and replaces $\overline{\mathbf{U}}(x,t)$ in the computation of the spatial error estimates for $\mathbf{U}(x,t)$. Since this scheme returns $\mathbf{U}(x,t)$ and controls a spatial error estimate for $\mathbf{U}(x,t)$, we will refer to this as standard (ST) error control.

The second type of interpolant, the *lower order interpolant (LOI)* [3], is based on interpolating $\mathbf{U}(x,t)$ and $\mathbf{U}_x(x,t)$ at the mesh points and $\mathbf{U}(x,t)$ at certain other points such that the resulting interpolant has an interpolation error that is asymptotically equivalent to the error of a collocation solution *of one order lower* than $\mathbf{U}(x,t)$. Then the difference between this interpolant and $\mathbf{U}(x,t)$ is used as an estimate of the spatial error of $\mathbf{U}(x,t)$. Since this scheme controls a spatial error estimate *for a collocation solution that is of one lower order than the returned approximate solution*, $\mathbf{U}(x,t)$, we will refer to this as *local extrapolation (LE) error control*. This type of control is similar to what is done in the context of Runge-Kutta formula pairs for the numerical solution of initial value ODEs [14]. BACOLI has been shown to be approximately twice as fast as BACOL [20].

4 Overview of new error control Gaussian collocation software for BVODEs: COLNEW95

In this section, we briefly describe the modifications that were made to COLNEW to obtain COLNEW95 and then provide some numerical results. Further details are provided in [1]. The modifications proceeded in three phases:

(Phase 0) The initial phase involved a modification of COLNEW to update it from its original Fortran 77 implementation to a Fortran 95 implementation. This led to major simplifications in the calling sequence of the solver. COLNEW95 employs a variety of Fortran 95 language features such as module environments, derived types, optional arguments, and dynamic memory allocation, and also computes divided difference Jacobians.

(Phase 1) The second phase of modifications involved augmenting the collocation solution, (6) (which, as mentioned earlier, has a global error that is $O\left(h^{p+1}\right)$) with a low-cost piecewise polynomial superconvergent interpolant [13], which has a global error that is $O\left(h^{2p}\right)$, the same order as that of the mesh point solution values. This interpolant, which we will refer to as the SCI-BV, is constructed by using the framework of continuous mono-implicit Runge-Kutta methods [19] and takes advantage of the fact that a collocation method is equivalent to a Runge-Kutta method [5].

In [13], the SCI-BV is used in a post-processing step to improve the accuracy of the final collocation solution returned by COLNEW. The issue with this use of the SCI-BV is that COLNEW is still required to perform a computation such that the estimated global error of the *collocation* solution is less than the user tolerance. The SCI-BV then provides a second solution approximation, *which has a global error that is much smaller than the user has requested.*

In this phase, we modified COLNEW to introduce the SCI-BV *within the code* rather than as a post-processing step. The SCI-BV is constructed after each intermediate collocation solution is computed and the difference between the collocation solution and the SCI-BV is used to estimate the error of the collocation solution. This

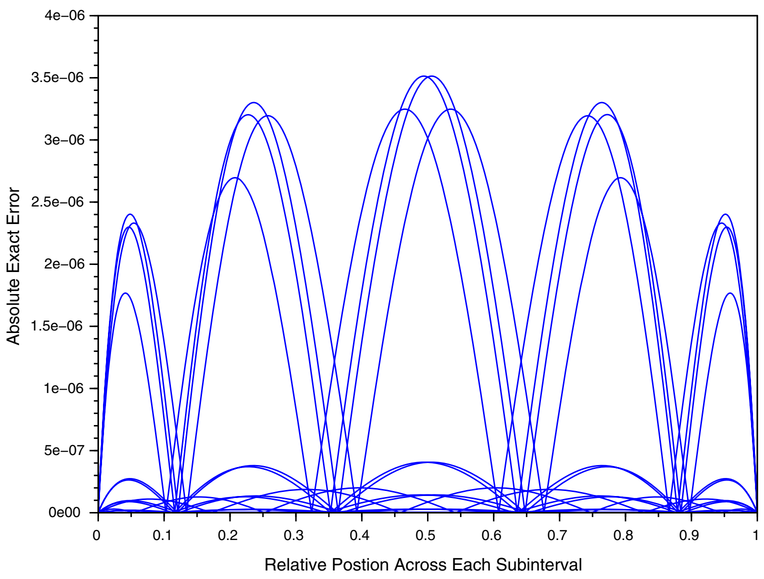

Fig. 1 Absolute error of collocation solution on each subinterval, mapped onto [0,1], for test problem (15)

means that RE is not required and thus it is not necessary to bias the mesh refinement algorithm so that it primarily forms each new mesh by halving each subinterval of the current mesh. We therefore also modified the mesh refinement algorithm so that it always uses equidistribution; this allows the meshes to be better adapted to the error estimates throughout the computation. Note that the collocation solution is still the primary solution that is returned to the user.

Another modification performed during this phase is based on the observation that the collocation solution on each subinterval is known to have a leading-order error term that is a multiple of a single polynomial [5]. Figure 1 shows the absolute error for the test problem,

$$\left(\epsilon + x^2\right) y'' + 4xy' + 2y = 0, \quad y(-1) = y(1) = 1/(1+\epsilon), \tag{15}$$

on each subinterval of a mesh that was used in the computation of the collocation solution; the errors on each subinterval are mapped onto [0,1] and plotted on the same graph. We choose $\epsilon = 0.1$, $p = 5$, and a tolerance of 10^{-5}. We see that the error has the same form on every subinterval and in particular that the maximum error occurs at the midpoint of each subinterval. We can therefore obtain the maximum value of the error estimate on each subinterval by evaluating the difference between the collocation solution and the SCI-BV at this point on each subinterval.

We compare the performance of COLNEW and COLNEW95_Phase1 on a test BVODE system obtained by using the transverse-method-of-lines (see, e.g., [5]), together with a fixed time step backward Euler method, to discretize the temporal domain of the 1D PDE:

$$z_t = z_{xx} - zz_x + \cos(\omega x) + t\omega^2 \cos(\omega x) - t^2 \cos(\omega x)\sin(\omega x), \tag{16}$$

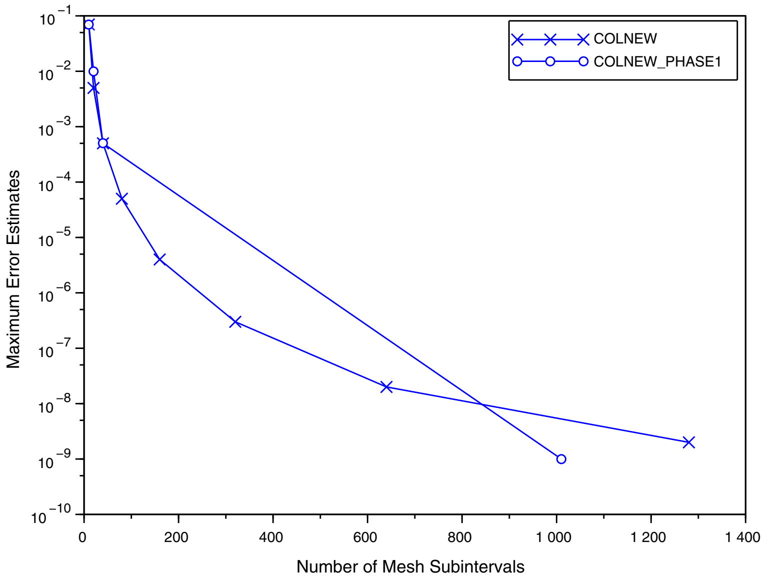

Fig. 2 Number of Mesh Subintervals vs. Maximum Error Estimate, COLNEW and COLNEW95_Phase1, 1D PDE test problem (16), p=5

where

$$z(0,t) = t, \quad z(1,t) = t\cos(\omega), \quad z(x,0) = 0, \quad \omega = 10, \quad t_{end} = 1.$$

The time step is chosen to give a system of 50 second-order BVODEs, which are then converted to a system of 100 first-order BVODEs. The tolerance is 10^{-10} and $p = 5$. In Fig. 2, we show results obtained by running COLNEW and COLNEW95_Phase1 on this test problem. We see that COLNEW95_Phase1 converges much more quickly to an acceptable numerical solution and that final mesh it uses has fewer subintervals than the final mesh employed by COLNEW. The CPU time for COLNEW95_Phase1 is 209 s while that for COLNEW is 474 s.

(Phase 2) The last step in the development of COLNEW95 involved computing the SCI-BV immediately after each collocation solution is computed but in this phase, *it is the SCI-BV that is returned to the user as the primary solution.* The error estimate for the SCI-BV is obtained by employing an algorithm called the HO scheme [8]. It makes use of a collocation method and its corresponding SCI-BV that is of one higher order than that which was used to compute the collocation solution upon which the primary SCI-BV is based. The HO scheme has a low computational cost; most of the work is done during the computation of the original collocation solution. In Fig. 3, we show results which compare COLNEW95_Phase1 with COLNEW95_Phase2. We see that COLNEW95_Phase2 converges to a mesh that leads to an acceptable numerical solution faster than COLNEW95_Phase1 and that it employs a final mesh that has substantially fewer subintervals than that employed by COLNEW95_Phase1. The

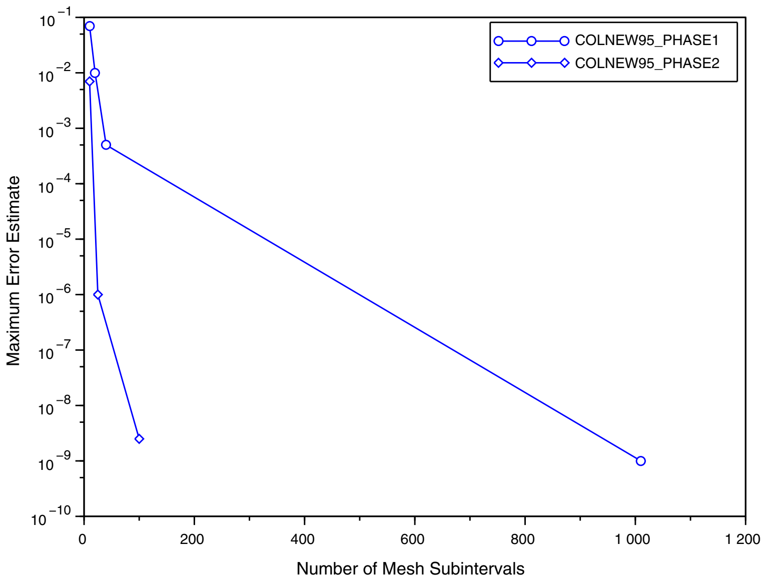

Fig. 3 Number of Subintervals vs. Maximum Error Estimate, COLNEW95_Phase1 and COLNEW95_Phase2, 1D PDE test problem (16), p=5.

CPU time used by COLNEW95_Phase2 is 124 s, while that for COLNEW95_Phase1 is 209 s. COLNEW95_Phase2 represents the current version of COLNEW95.

5 Overview of new error control Gaussian collocation software for 1D PDEs: BACOLRI

The release of BACOLRI completes a four-step software development project that started with BACOL, followed with BACOLR, and more recently saw the release of BACOLI. The following diagram shows the relationships among the four codes.

$$\begin{tikzcd} BACOL \arrow[r, Rightarrow] \arrow[d, Rightarrow] & BACOLR \arrow[d, Rightarrow] \\ BACOLI \arrow[r, Rightarrow] & BACOLRI \end{tikzcd}$$

As mentioned earlier, BACOLR was developed as a modification of BACOL to replace DASSL with RADAU5, addressing the stability issues associated with the use of DASSL on certain types of PDEs, and BACOLI was developed from BACOL to address efficiency issues associated with the computation of the spatial error estimates. BACOLRI follows from both BACOLR and BACOLI; it uses RADAU5 for the time integration and replaces the expensive computation of the second approximate solution in BACOLR with the interpolation-based spatial error estimation schemes introduced in BACOLI. The development of BACOLRI involved a substantial modification of BACOLR. Further details regarding this modification and the results of an extensive numerical testing process are described in [21].

As mentioned earlier, BACOLI has two error control modes, ST (standard) error control, based on the SCI scheme, and LE (Local Extrapolation) error control, based on the LOI scheme. Because BACOLRI also implements both the SCI and LOI schemes, it also has the ST and LE error control options. The earlier package, BACOLR, runs in ST error control mode; but with a simple modification, it is possible to have BACOLR return the higher order approximate solution and in this case it is employing LE error control mode. We introduce the following notation to summarize the software package and error control combinations that are discussed in this paper.

- (BACOLI/ST): BACOLI using the SCI spatial error estimation scheme in ST spatial error control mode,
- (BACOLI/LE): BACOLI using the LOI spatial error estimation scheme in LE spatial error control mode,
- (BACOLRI/ST): BACOLRI using the SCI spatial error estimation scheme in ST spatial error control mode,
- (BACOLRI/LE): BACOLRI using the LOI spatial error estimation scheme in LE spatial error control mode,
- (BACOLR/ST): BACOLR in ST spatial error control mode,
- (BACOLR/LE): BACOLR in LE spatial error control mode.

In the first set of tests, we compare BACOLR/ST and BACOLR/LE with BACOLRI/ST and BACOLRI/LE on a standard test problem, which we call the two-layer Burger's equation:

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

with boundary and initial conditions taken from the exact solution,

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

where

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

ϵ is a problem-dependent parameter that we set to 10^{-4}, and $t_{end} = 1$. We choose $p = 6$ and consider a range of tolerances uniformly distributed (on a log scale) over $10^{-2}, \ldots, 10^{-10}$.

In Fig. 4, we plot CPU times for BACOLR/LE, BACOLRI/ST, and BACOLRI/LE *relative to the corresponding CPU times for BACOLR/ST.* We also plot the best fit line to each data set to provide some indication of the overall trends of the data sets. Note that the horizontal line associated with a relative error of 1.0 corresponds to the BACOLR/ST code. We see that BACOLRI/ST and BACOLRI/LE both have costs that are approximately 40–60% of those of BACOLR/ST and BACOLR/LE, over the range of tolerances considered.

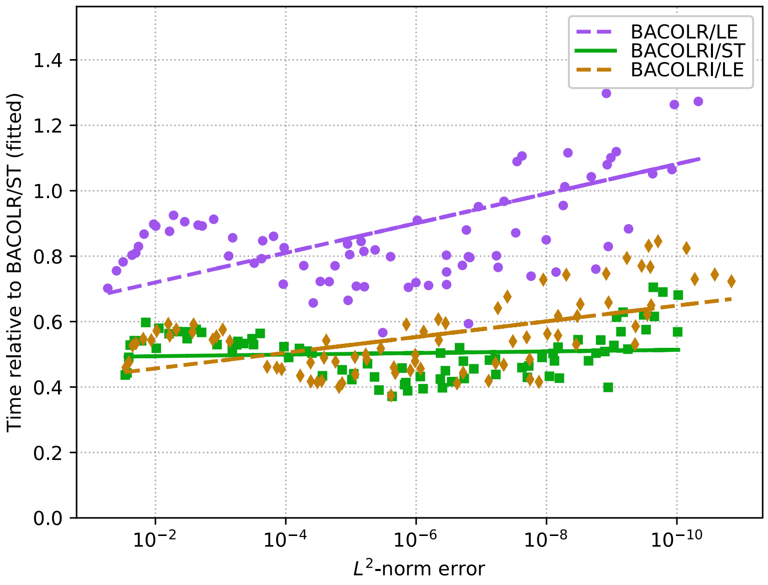

Fig. 4 L^2-norm error vs. CPU times for BACOLR/LE, BACOLRI/ST, and BACOLRI/LE *relative to the corresponding CPU times for BACOLR/ST*, for the Two Layer Burger's Equation, $\epsilon = 10^{-4}$, $t_{end} = 1$, $p = 6$

In the second set of tests, we compare BACOLI/ST and BACOLI/LE with BACOLRI/ST and BACOLRI/LE on a problem we call the coupled nonlinear Schrödinger system:

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

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

where $-30 < x < 90$ and $t > 0$, with boundary conditions given by,

$$(u_1)_x(-30,t) = (u_2)_x(-30,t) = 0, \quad (u_1)_x(90,t) = (u_2)_x(90,t) = 0.$$

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 exact solutions are,

$$u_1(x,t) = \sqrt{\frac{2\kappa}{1+\rho}}\,\text{sech}\left(\sqrt{2\kappa}(x-\phi t)\right) e^{i\left((\phi-\eta)x - \left(\frac{\phi^2-\eta^2}{2} - \kappa\right)t\right)},$$

$$u_2(x,t) = \sqrt{\frac{2\kappa}{1+\rho}}\,\text{sech}\left(\sqrt{2\kappa}(x-\phi t)\right) e^{i\left((\phi+\eta)x - \left(\frac{\phi^2-\eta^2}{2} - \kappa\right)t\right)},$$

where $\phi = 1$, $\eta = 0.5$, $\rho = 2/3$, and $\kappa = 1$. We choose $t_{end} = 1$. We again choose $p = 6$ and the same range of tolerances over $10^{-2}, \ldots, 10^{-10}$.

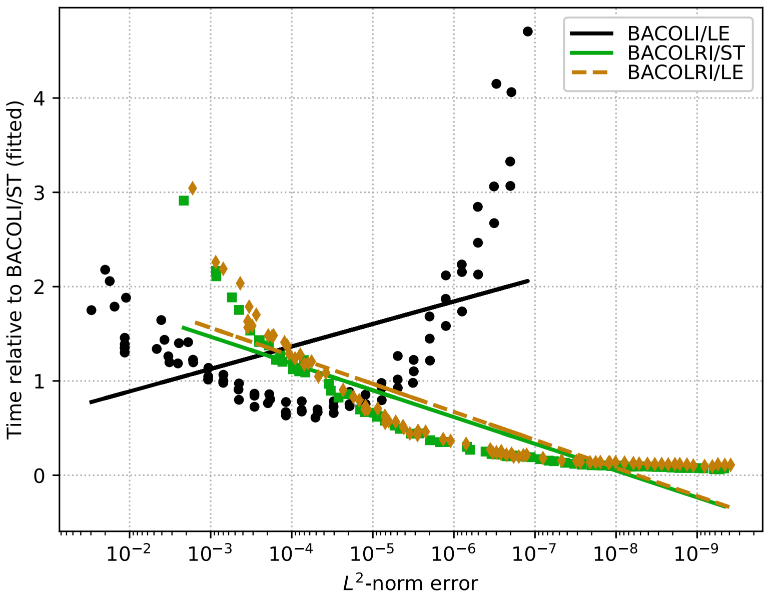

Fig. 5 L^2-norm error vs. CPU times for BACOLI/LE, BACOLRI/ST, and BACOLRI/LE *relative to the corresponding CPU times for BACOLI/ST*, for the Coupled Nonlinear Schrödinger System, $t_{end} = 1$, $p = 6$

BACOLI is not able to solve this problem unless we restrict DASSL to use only BDFs of orders 1 or 2 due to the stability issues that arise for the BDFs when applied to problems of this type, as mentioned earlier in this paper. Since it uses RADAU5 rather than DASSL, these stability issues do not arise for BACOLRI and the package is able to solve this problem in a straightforward fashion. After setting, BACOLI to use DASSL restricted to the BDFs of orders 1 and 2, we obtain the results given in Fig. 5, where we plot CPU times for BACOLI/LE, BACOLRI/ST, and BACOLRI/LE relative to the corresponding CPU times for BACOLI/ST. We see that, particularly for medium to sharp tolerances, BACOLRI/ST and BACOLRI/LE are both much faster than either BACOLI/ST or BACOLI/LE.

6 Conclusions and future work

Error control Gaussian collocation solvers for BVODEs and 1D PDEs have been widely used for several decades. The new solvers, COLNEW95 and BACOLRI, provide significant improvements over earlier solvers from their respective families.

Future work associated with COLNEW95 will involve extensive testing and analysis to optimize its performance. As well, the current version of COLNEW95 is able to compute approximate solutions, based on the corresponding SCI-BVs, that are of orders 2, 4, or 6. However COLNEW can return approximate solutions of orders 2, . . . , 8. We therefore plan to extend COLNEW95 to allow it to be able to return approximate solutions of order 8. This will involve deriving a new 8th order SCI-BV, extending the original development undertaken in [13].

Future work involving BACOLRI will center on further testing and analysis to improve its performance. The introduction of algorithms for the automatic selection of *p*, the degree of the B-spline basis, and the type of error control, SCI/ST, or LOI/LE, is also planned.

As well, the results from both these projects will impact on future development of error control Gaussian collocation software for 2D PDEs.

Funding information This work was financially supported by the Mathematics of Information Technology and Complex Systems Network, the Natural Sciences and Engineering Research Council of Canada, and Saint Mary's University.

References

1. Adams, M., Muir, P.H.: Gaussian collocation error control software with superconvergent interpolants for boundary value ODEs. In: preparation (2019)
2. Arsenault, T., Smith, T., Muir, P.H.: Superconvergent interpolants for efficient spatial error estimation in 1D PDE, collocation solvers. Can. Appl. Math Q. **17**, 409–431 (2009)
3. Arsenault, T., Smith, T., Muir, P.H., Pew, J.: Asymptotically correct interpolation-based spatial error estimation for 1D PDE, solvers. Can. Appl. Math Q. **20**, 307–328 (2012)
4. Ascher, U.M., Christiansen, J., Russell, R.D.: Collocation software for boundary value ODEs. ACM Trans. Math Softw. **7**, 209–222 (1981)
5. Ascher, U.M., Mattheij, R.M.M., Russell, R.D.: Numerical solution of boundary value problems for ordinary differential equations, volume 13 of Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (1995)
6. Ascher, U.M., Spiteri, R.J.: Collocation software for boundary value differential-algebraic equations. SIAM J. Sci Comput. **15**(4), 938–952 (1994)
7. Bader, G., Ascher, U.M.: A new basis implementation for a mixed order boundary value ODE solver. SIAM J. Sci. Stat Comput. **8**(4), 483–500 (1987)
8. Boisvert, J.J., Muir, P.H., Spiteri, R.J.: A Runge-Kutta BVODE solver with global error and defect control. ACM Trans. Math. Software **39**(2), Art. 11, 22 (2013)
9. Brenan, K.E., Campbell, S.L., Petzold, L.R.: Numerical solution of initial-value problems in differential-algebraic equations, vol. 14 of classics in applied mathematics. Society for industrial and applied mathematics (SIAM), Philadelphia, PA (1996)
10. Cash, J.R., Moore, G., Wright, R.W.: An automatic continuation strategy for the solution of singularly perturbed linear two-point boundary value problems. J Comput. Phys. **122**(2), 266–279 (1995)
11. de Boor, C.: A practical guide to splines, volume 27 of applied mathematical sciences. Springer-Verlag, New York (2001). revised edition
12. Díaz, J.C., Fairweather, G., Keast, P.: 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. Softw. **9**(3), 376–380 (1983)
13. Enright, W.H., Muir, P.H.: Superconvergent interpolants for the collocation solution of boundary value ordinary differential equations. SIAM J. Sci. Comput. **21**(1), 227–254 (1999)
14. Hairer, E., Nørsett, S.P., Wanner, G.: Solving ordinary differential equations. I, volume 8 of Springer series in computational mathematics, 2nd edn. Springer, Berlin (1993)
15. Hairer, E., Wanner, G.: Solving ordinary differential equations. II, volume 14 of Springer series in computational mathematics, 2nd edn. Springer, Berlin (1996)
16. Keast, P., Muir, P.H.: Algorithm 688: EPDCOL: a more efficient PDECOL code. ACM Trans. Math. Softw. **17**(2), 153–166 (1991)
17. Li, Z., Muir, P.H.: B-spline Gaussian collocation software for two-dimensional parabolic PDEs. Adv. Appl. Math Mech. **5**, 528–547 (2013)

18. Madsen, N.K., Sincovec, R.F.: Algorithm 540: PDECOL, general collocation software for partial differential equations. ACM Trans Math Softw **5**(3), 326–351 (1979)
19. Muir, P.H., Owren, B.: Order barriers and characterizations for continuous mono-implicit Runge-Kutta schemes. Math Comp **61**(204), 675–699 (1993)
20. Pew, J., Li, Z., Muir, P.H.: Algorithm 962:, BACOLI: B-spline adaptive collocation software for PDEs with interpolation-based spatial error control. ACM Trans Math Softw **42**(3), 25:1–25:17 (2016)
21. Pew, J., Tannahill, C., Murtha, T., Muir, P.H.: Gaussian collocation Runge-Kutta PDE software with interpolation-based error control. In: preparation (2019)
22. Scilab. bvode, bvodeS. Scilab Enterprises, 143 bis rue Yves Le Coz, 78000 Versailles, France
23. Soetaert, K., Cash, J.R., Mazzia, F.: Solving differential equations in R. Springer, New York (2012)
24. Virtanen, P.: scikits.bvp1lg 0.2.8., https://pv.github.io/scikits.bvp1lg/
25. Wang, R., Keast, P., Muir, P.H.: 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)
26. Wang, R., Keast, P., Muir, P.H.: BACOL: B-spline adaptive COLlocation software for 1D parabolic PDEs. ACM Trans Math. Softw. **30**(4), 454–470 (2004)
27. Wang, R., Keast, P., Muir, P.H.: A comparison of adaptive software for 1D, parabolic PDEs. J. Comput. Appl Math. **169**(1), 127–150 (2004)
28. Wang, R., Keast, P., Muir, P.H.: A high-order global spatially adaptive collocation method for 1-D, parabolic PDEs. Appl. Numer Math. **50**(2), 239–260 (2004)