A Computational Study of the Efficiency of Collocation Software for 1D Parabolic PDEs with Interpolation-based Spatial Error Estimation *

Jack Pew[†], Zhi Li[‡], Paul Muir[§]

Abstract

BACOL is a software package for the numerical solution of systems of one-dimensional parabolic partial differential equations (PDEs) that has been shown to be superior to other similar packages, especially for problems exhibiting sharp spatial layer regions where a stringent tolerance is imposed. BACOL, based on a method-of-lines algorithm, features adaptive control of a high order estimate of the spatial error. (Adaptive control of the temporal error in the numerical solution of the system of differential-algebraic equations (DAEs), arising from a B-spline Gaussian collocation spatial discretization, is provided by the underlying DAE solver, DASSL.) The spatial error estimate for the collocation solution computed by the code is obtained by computing a *second* collocation solution, which involves a substantial cost - the execution time and memory usage are almost doubled.

In this report we discuss BACOLI, a new version of BACOL that computes only one collocation solution and uses efficient interpolation-based approaches to obtain a spatial error estimate. These approaches have recently been shown to provide spatial error estimates of comparable quality to those computed by BACOL. We describe the substantial modification of the BACOL code that was required in order to obtain BACOLI and provide numerical results to compare BACOL and BACOLI. We show that BACOLI is about twice as efficient as BACOL.

Subject Classification: 65M15, 65M20, 65M70
Keywords: Efficiency, 1D Parabolic PDEs, Collocation, Spatial Error Estimation, Interpolation, Method-of-Lines.

*This work was 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.

[†]Saint Mary's University, Halifax, NS, Canada, B3H 3C3
[‡]Hong Kong Baptist University, Kowloon Tong, Kowloon
[§]Saint Mary's University, Halifax, NS, Canada, B3H 3C3, muir@smu.ca

1 Introduction

In this report, we consider systems of PDEs with *NPDE* components having the form

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

with initial conditions

$$\underline{u}(x,t_0) = \underline{u}_0(x), \qquad a \le x \le b, \tag{2}$$

and separated boundary conditions

$$\underline{b}_L\left(t, \underline{u}(a,t), \underline{u}_x(a,t)\right) = \underline{0}, \qquad \underline{b}_R\left(t, \underline{u}(b,t), \underline{u}_x(b,t)\right) = \underline{0}, \quad t \ge t_0. \tag{3}$$

The method-of-lines (MOL) approach for the numerical solution of the above problem class involves the discretization of the spatial domain using a numerical method such as a finite difference method, a finite element method, or a collocation method. (We will consider a detailed discussion of the use of a collocation method for the spatial discretization in the next section of this report.) The spatial discretization process reduces the PDEs to a system of time-dependent ordinary differential equations (ODEs). When this system of ODEs is combined with the boundary conditions, the result is a system of time-dependent differential-algebraic equations (DAEs). The DAE system is solved using high quality DAE software such as DASSL [9] or RADAU5 [20].

The computation of a numerical solution using error control means that the computation is adapted so that a high order estimate of the error in the approximate solution is less than a user-provided tolerance, i.e., an approximate solution is not returned by the code unless the error estimate satisfies the user tolerance. Control of the *temporal error* in an MOL solver is handled by the underlying DAE solver that computes the solution to the time-dependent DAE system. An MOL solver that provides *spatial error control* also computes a high order estimate of the spatial error of the approximate solution and then adapts the spatial discretization in an attempt to compute an approximate solution whose spatial error estimate is less than the user-provided tolerance. Examples of MOL codes of this type are HPNEW [23], and BACOL/BACOLR [32, 30, 31, 33].

In previous studies, [31, 33], the BACOL/BACOLR packages were shown to be comparable to and in some cases superior to other available packages for the above problem class, especially for problems exhibiting sharp spatial layer regions, with a high accuracy requirement. These packages employ adaptive B-spline collocation for the spatial discretization; BACOL uses DASSL for the solution of the DAEs resulting from the spatial discretization; BACOLR uses RADAU5 for this task. Given a spatial mesh which partitions $[a,b]$, the collocation algorithm employed by BACOL/BACOLR expresses the approximate solution at a given time as a linear combination of known spatial basis functions (B-splines)[12] - piecewise polynomials of a given degree p - with unknown

time dependent coefficients, determined from the solution of the DAE system. In BACOL/BACOLR, the spatial error estimate for the collocation solution is obtained through the computation of a second global collocation solution; the same general approach is used but the second collocation solution is expressed in terms of piecewise polynomials of degree $p + 1$. The difference between the two collocation solutions gives an estimate of the spatial error *in the lower order collocation solution*. (We will refer to the collocation solution of degree p as the lower order solution and the collocation solution of degree $p + 1$ as the higher order solution.) *The computation of a second global collocation solution in order to obtain a spatial error estimate clearly involves a major computational cost, essentially doubling the cost of the overall computation.*

There is of course a substantial body of literature on error estimation for the numerical solution of PDEs - see, e.g., [1] and [15] and references within. Other examples of related work on error estimation for PDEs include [6], [7], [29], [25, 28]. See also the recent book [8] and references within. The recent work most closely related to the spatial error estimation schemes employed in the software we discuss in this report is that of Moore [23, 24, 27, 26], in which interpolation error based spatial error estimates for 1D parabolic PDEs are discussed.

In this report we consider two approaches for the modification of BACOL that preserve its ability to compute and control a high order estimate of the spatial error of the collocation solution but avoid the computation of a second collocation solution. The two approaches involve the replacement of one of the two collocation solutions computed by BACOL with an appropriately designed interpolant of the same order:

- In one of the approaches we replace the *higher* order collocation solution with a piecewise polynomial interpolant that is based on evaluation of the lower order collocation solution at certain special points where it is known to be superconvergent; an interpolant based on these values can then be constructed so that it has the same order of accuracy as the higher order solution and hence can replace it in the computation of the spatial error estimate.

- In the other approach we replace the *lower* order collocation solution with an interpolant that is based on evaluation of the higher order collocation solution at certain points that ensure that the leading order term in the interpolation error for this interpolant is asymptotically equivalent to the leading order term in the collocation error for the lower order collocation solution. This interpolant can then replace the lower collocation solution in the computation of the spatial error estimate.

See [2, 4] for further details; a summary is provided in this report in Section 3.

One important goal of this report is to describe the development of the new code, BACOLI, that employs these new spatial error estimates. BACOLI was developed through a substantial modification of the BACOL package to allow it to use either of the above interpolation-based spatial error estimation

schemes. (Since BACOL and BACOLR employ the same spatial discretization scheme, the software modification process described in this report could also be applied, with minor modifications, to BACOLR; however, in this report we focus on BACOL.) A second important goal is to provide an extensive numerical comparison of the efficiency of the original BACOL code vs. the new BACOLI code.

This report is organized as follows. Section 2 reviews the BACOL algorithm and its spatial error estimation scheme. In Section 3, we provide a brief review of the two interpolation-based schemes for spatial error estimation in BACOLI. Section 4 provides a detailed discussion of the software modifications undertaken to develop BACOLI from BACOL. Section 5 presents and discusses numerical results for the comparison of the efficiency of BACOL vs. BACOLI. Our conclusions and suggestions for future work are provided in Section 6.

2 Overview of BACOL and its Spatial Error Estimation Scheme

Let the points, $a = x_0 < x_1 < \cdots < x_{NINT} = b$, define a spatial mesh consisting of $NINT$ subintervals partitioning the problem domain, $[a, b]$. In BACOL, the approximate solution is a C^1-continuous piecewise polynomial in x, of a given degree p ($3 \le p \le 11$) on each subinterval. This piecewise polynomial is represented as a linear combination of C^1-continuous B-spline basis functions [12] with time dependent coefficients. That is, the approximate solution, $\underline{U}(x,t)$, has the form

$$\underline{U}(x,t) = \sum_{i=1}^{NC_p} \underline{y}_{p,i}(t) B_{p,i}(x), \tag{4}$$

where $\underline{y}_{p,i}(t)$ is the (unknown) time dependent coefficient of the i-th B-spline basis function, $B_{p,i}(x)$, and $NC_p = NINT(p-1)+2$.

The unknown coefficients appearing in (4) are determined by imposing certain conditions on the approximate solution, $\underline{U}(x,t)$: we require $\underline{U}(x,t)$ to satisfy the PDEs at certain points on each subinterval (the *collocation* conditions) and the boundary conditions at $x = a$ and $x = b$. The collocation conditions have the form

$$\frac{d}{dt}\underline{U}(\xi_l, t) = \underline{f}\left(t, \xi_l, \underline{U}(\xi_l,t), \underline{U}_x(\xi_l,t), \underline{U}_{xx}(\xi_l,t)\right), \tag{5}$$

for $l = 2, \ldots, NC_p - 1$, where the collocation points are

$$\begin{aligned} \xi_l &= x_{i-1} + h_i \rho_j, \quad \text{where } \; l = 1 + (i-1)(p-1) + j, \\ &\quad \text{for } \; i = 1, \ldots, NINT, \quad j = 1, \ldots, p-1, \end{aligned} \tag{6}$$

and $\{\rho_i\}_{i=1}^{p-1}$, are the set of $p-1$ Gauss points on $[0,1]$ - see, e.g., [5]. The points, $\xi_1 = a$ and $\xi_{NC_p} = b$ are associated with requiring the approximate solution to satisfy the boundary conditions:

$$\underline{b}_L\left(t, \underline{U}(a,t), \underline{U}_x(a,t)\right) = 0, \qquad \underline{b}_R\left(t, \underline{U}(b,t), \underline{U}_x(b,t)\right) = 0. \tag{7}$$

The collocation conditions (5) are a system of ODEs (in time). These ODEs, coupled with the boundary conditions, (7), give a system of DAEs; this DAE system is treated using a modified version of the DAE solver DASSL. After DASSL has computed approximations for the $\underline{y}_{p,i}(t)$ coefficients at time t, these can be employed together with the known B-spline basis functions, $B_{p,i}(x)$, within (4), to obtain values of the approximate solution at desired x values (in $[a,b]$), at time t. The most expensive part of the DASSL execution involves linear algebra computations. The modified version of DASSL that is employed by BACOL takes into account the special almost block diagonal (ABD) [13] structure of the linear systems that arise. See [30] for further details.

The collocation solution, $\underline{U}(x,t)$, at the current time, t, is accepted if its spatial error estimate satisfies a given user tolerance. In BACOL, the spatial error estimate is obtained through the computation of a *second* (higher order) collocation solution on the same spatial mesh for the same time t. This approximate solution, $\underline{\bar{U}}(x,t)$, is a C^1-continuous, piecewise polynomial in x of degree $p+1$ on a given subinterval; it has the form

$$\underline{\bar{U}}(x,t) = \sum_{i=1}^{NC_{p+1}} \underline{y}_{p+1,i}(t) B_{p+1,i}(x), \tag{8}$$

where $\underline{y}_{p+1,i}(t)$ is the unknown time dependent coefficient for the ith B-spline basis polynomial, $B_{p+1,i}(x)$, of degree $p+1$ on each subinterval, and $NC_{p+1} = NINT \cdot p + 2$. The unknowns, $\underline{y}_{p+1,i}(t)$, are determined by requiring $\underline{\bar{U}}(x,t)$ to satisfy the PDEs at the images of the p Gauss points on $[0,1]$ mapped onto each subinterval and the boundary conditions at $x = a$ and $x = b$. This leads to a second system of DAEs whose solution gives the $\underline{y}_{p+1,i}(t)$ coefficients. In order to ensure that the two approximate solutions, $\underline{U}(x,t)$ and $\underline{\bar{U}}(x,t)$, are available at the same time t, the two DAE systems are provided to DASSL as one larger DAE system so that DASSL treats both systems of DAEs with the same time-stepping strategy. (However, the linear systems are treated separately in order to take advantage of the ABD structure of the matrices that arise.)

Based on $\underline{U}(x,t)$ and $\underline{\bar{U}}(x,t)$, BACOL computes a set of $NPDE$ spatial error estimates *over the whole spatial domain* of the form

$$E_s(t) = \sqrt{\int_a^b \left(\frac{U_s(x,t) - \bar{U}_s(x,t)}{ATOL_s + RTOL_s |U_s(x,t)|} \right)^2 dx}, \quad s = 1, \ldots, NPDE, \tag{9}$$

where $U_s(x,t)$ is the sth component of $\underline{U}(x,t)$, $\bar{U}_s(x,t)$ is the sth component of $\underline{\bar{U}}(x,t)$, and $ATOL_s$ and $RTOL_s$ are the (user-provided) absolute and relative tolerances for the sth component of the spatial error estimate. BACOL also computes a second set of $NINT$ spatial error estimates, *one for each spatial*

mesh subinterval, of the form,

$$\hat{E}_i(t) = \sqrt{\sum_{s=1}^{NPDE} \int_{x_{i-1}}^{x_i} \left(\frac{U_s(x,t) - \bar{U}_s(x,t)}{ATOL_s + RTOL_s|U_s(x,t)|} \right)^2 dx}, \quad i = 1, \ldots, NINT. \tag{10}$$

Note that $E_s(t), s = 1, \ldots, NPDE$, and $\hat{E}_i(t), i = 1, \ldots, NINT$, are estimates of the spatial error associated with the *lower* degree solution, $\underline{U}(x,t)$. At the current time, t, $\underline{U}(x,t)$ is accepted if

$$\max_{1 \le s \le NPDE} E_s(t) \le 1.$$

If this condition is satisfied, BACOL attempts to take the next time step. Otherwise, the step is rejected and BACOL uses the spatial error estimates, $\hat{E}_i(t), i = 1, \ldots, NINT$, to perform a spatial *remeshing*; that is, BACOL attempts to construct a new mesh that will have as many mesh points as necessary to yield an approximate solution whose estimated spatial error satisfies the user tolerances and that will approximately equidistribute the estimated spatial error over the subintervals of the new mesh.

The theoretical basis for the above spatial error estimates follows from standard convergence results for Gaussian collocation applied to 1D parabolic PDEs, obtained in [14] and [10]: over the entire spatial domain, the collocation solution, based on piecewise polynomials of degree p, with $p-1$ collocation points per subinterval, has a spatial error that is $O(h^{p+1})$, where h is the maximum spatial mesh subinterval size. Thus $\bar{\underline{U}}(x,t)$ is one order higher than $\underline{U}(x,t)$.

The temporal tolerance employed in DASSL is equal to the spatial tolerance and in numerical experiments performed with BACOL, this has been observed to (generally) be sufficient to insure that the temporal error does not dominate the spatial error, which in turn allows the spatial error estimates to be effective.

Once a new mesh is determined, DASSL requires current and past solution information (corresponding to several past time steps) associated with the new mesh points for both the lower order and higher order collocation solutions. (This is because DASSL employs a family of multi-step methods - the Backward Differentiation Formulas.) In BACOL this information for both collocation solutions is obtained through high order interpolation of the *higher order* solution information associated with the previous mesh. Once this interpolated information has been computed, *both* collocation solutions are propagated forward in time. This is referred to as a *warm start* since it allows the time integration to continue using the same time step size and same time integration method that was used prior to the remeshing. However, in some cases after it has encountered several rejected steps due to repeated failures of the spatial error tests, (9), BACOL will restart DASSL using only solution information from the current time. The time integration will then proceed from the current time using a low order, one-step, time stepping method and a small stepsize. Such a restart is known as a *cold start*. See [32, 30] for further details.

3 Interpolation-based Spatial Error Estimation

It is clear that the computation by BACOL of the second collocation solution represents a significant computational overhead. A key feature of the interpolation-based spatial error estimation schemes we consider in this report is that they compute only *one* collocation solution and then perform an auxiliary computation, involving an interpolant associated with the collocation solution, in order to obtain an estimate of the spatial error.

Since BACOL computes two collocation solutions, there are two natural viewpoints that can be taken regarding the computation of an interpolation-based spatial error estimate: (i) we can remove the computation of the *higher* order collocation solution and replace that solution in the spatial error estimates, (9), (10), with an interpolant of the same order, or (ii) we can remove the computation of the *lower* order solution and replace that solution in the spatial error estimates, (9), (10), with an interpolant of the same order. The algorithm for the first scheme makes use of the presence of higher order, i.e., superconvergent, values available from the lower order collocation solution and its first spatial derivative and we will refer to this scheme as the Superconvergent Interpolant (SCI) scheme. The algorithm for the second approach involves the construction of an interpolant, based on evaluations of the higher order collocation solution and its first spatial derivative, *whose interpolation error agrees asymptotically with the collocation error of the lower order collocation solution.* We will refer to this scheme as the Lower Order Interpolant (LOI) scheme.

As mentioned in the previous section, while BACOL computes and *propagates* both collocation solutions forward in time, the spatial error estimate it computes gives an estimate of the collocation error for only the lower order collocation solution. *This means that both the lower order and higher order collocation solutions are propagated based on a spatial error estimate appropriate for the lower order collocation solution.* Thus, if we focus on the lower order collocation solution computed by BACOL, we see that BACOL advances this solution based on an estimate of the spatial error appropriate for that solution. This perspective makes the SCI scheme appear to be the most natural approach since it provides an estimate of the spatial error appropriate for the collocation solution that is computed. However, if we focus on the higher order collocation solution computed by BACOL, we see that this solution is propagated based on a spatial error estimate that is appropriate for a collocation solution of one order lower. (This is similar to an approach involving Runge-Kutta formula pairs applied in the numerical integration of initial value ODEs - see, e.g., [19] - in which numerical solutions based on two Runge-Kutta formulas, differing by one order of accuracy, are computed on each time step and the difference between the two provides an estimate of the local error in the lower order solution. In *standard (ST) mode*, the lower order solution is propagated based on this estimate. However, it is natural to think about propagating the higher order solution, since it is expected to be more accurate, and in this case, the higher order solution is propagated based on a spatial error estimate that is appropriate for the lower order solution. This is known as *local extrapolation*

(LE) mode.) The original BACOL code returns the lower order collocation solution to the user and thus can be viewed as operating in ST mode. However, it would require only a small modification of the code to have it return the higher order solution and in that case the code could be viewed as operating in LE mode. The SCI scheme is similar to the scheme currently implemented in BACOL and operates in ST error control mode. The LOI scheme employs LE mode; the collocation solution is propagated based on a spatial error estimate that is appropriate for a collocation solution of one order lower.

3.1 The SCI Spatial Error Estimation Scheme

Here we provide a brief review of the SCI spatial error estimation scheme. Further details are available in [2]. Since BACOL in fact takes as input the number of collocation points per subinterval ($kcol$) rather than the degree of the piecewise polynomial on each subinterval (p), in the remainder of this report we will use $kcol$ rather than p to specify a particular order of collocation scheme; the degree of the piecewise polynomials will be $p = kcol + 1$. Since the limit on p assumed by BACOL is $3 \leq p \leq 11$, the limit on $kcol$ is $2 \leq kcol \leq 10$.

For a given input value for $kcol$, we compute the corresponding collocation solution, as described in the previous section. Then, based on evaluations of this collocation solution at certain special (superconvergent) points within $[a, b]$ (see below) we construct an interpolant that is *one order higher* than the collocation solution. The collocation solution and the SCI are then used to compute spatial error estimates, similar to (9) and (10). (The quantity, $\bar{U}_s(x, t)$, in (9) and (10) is replaced by the SCI.)

As mentioned in the previous section, standard convergence results for Gaussian collocation applied to 1D parabolic PDEs state that over the entire spatial domain, the collocation solution spatial error is $O(h^{kcol+2})$, where h is the maximum spatial mesh subinterval size. These papers also prove that at the spatial mesh points, both the collocation solution and its first spatial derivative have spatial errors that are $O(h^{2\times kcol})$. Thus the mesh point values are superconvergent when $kcol > 2$. (For $kcol = 2$, the mesh point collocation solution errors are the same order as elsewhere in the subinterval and thus the collocation solution values at the mesh points are not superconvergent in this case.)

Similar results were proved around the same time for Gaussian collocation applied to boundary value ODEs - see, e.g., [5]. However, the boundary value ODE results provide further details regarding the leading order term in the collocation error for each subinterval. These results show that (for $kcol \geq 3$) there are $kcol - 2$ points, internal to each subinterval, where the collocation error is $O(h^{kcol+3})$, one order higher than the standard collocation error. To our knowledge, the corresponding result for the PDE case has not been proved. However [3] provides experimental evidence demonstrating that, for the spatial discretization of a 1D parabolic PDE by Gaussian collocation, the same $kcol-2$ points within each spatial mesh subinterval have an order of convergence consistent with that for the BVODE case, i.e., for the collocation solution computed by BACOL that employs $kcol \geq 3$ collocation points per subinterval, there

are $kcol - 2$ points internal to each subinterval where this collocation solution (which generally has a spatial error that is $O(h^{kcol+2})$) is *superconvergent*, i.e., the spatial error is $O(h^{kcol+3})$. Thus simply evaluating this collocation solution at the mesh points and at these known special points within each subinterval and evaluating the first spatial derivative of the collocation solution at the mesh points provides the required superconvergent data.

The SCI is a C^1-continuous piecewise polynomial interpolant determined as follows. Since the interpolation error should not interfere with the overall error of the interpolated values, the polynomial that represents the SCI on each subinterval interpolates the 4 superconvergent solution and first spatial derivative mesh point values at the left and right endpoints of the subinterval, the $kcol - 2$ superconvergent solution values that are internal to each subinterval, and the two closest superconvergent values internal to each adjacent subinterval. (For the leftmost and rightmost subintervals, the SCI interpolates the two closest superconvergent solution values available in the lone adjacent subinterval.) This gives a total of $kcol + 4$ interpolation points and, from standard interpolation theory, implies that the interpolation error will be $O(h^{kcol+4})$. Thus the interpolation error is dominated by the spatial error of the superconvergent data values (i.e., the spatial error of the superconvergent data values is, asymptotically, larger than the interpolation error of the SCI), implying that the spatial error of the SCI is $O(h^{kcol+3})$, one order higher than that of the collocation solution.

Because the SCI will depend on a combination of solution and derivative values, a Hermite-Birkhoff form for the interpolant is appropriate. The paper [16] discusses the general forms for the Hermite-Birkhoff basis polynomials upon which the SCI is based and also provides results that can be used to obtain an explicit expression for the interpolation error of the Hermite-Birkhoff interpolant representing the SCI on a given subinterval; see [2]. From this expression it can be seen that, for a given subinterval, the interpolation error depends on the ratios of the size of that subinterval to the sizes of the immediately adjacent subintervals. In the case of the leftmost or rightmost subinterval, the interpolation error depends on the square of the ratio of the size of that subinterval to the size of the lone adjacent subinterval. When any of these ratios is large, the interpolation error can become large and this can lead to a substantial overestimate of the spatial error estimate by the SCI scheme. The paper [2] provides experimental results demonstrating that while the SCI scheme generally provides estimates of the spatial error that are of comparable quality to those provided by the original BACOL code, for highly nonuniform meshes, the SCI scheme can sometimes give spatial error estimates that significantly overestimate the true spatial error.

As mentioned earlier, when BACOLI uses the SCI spatial error estimation scheme, it computes a collocation solution for which the number of collocation points used per subinterval is specified by the input *kcol* value and it controls an estimate of the spatial error of that collocation solution. Thus, in this case, the code employs *ST error control mode*.

Note also, as explained above, that the SCI scheme requires $kcol \geq 3$ and

thus the range of allowed $kcol$ values for BACOLI using the SCI scheme will be $3 \leq kcol \leq 10$, while the original BACOL allows $2 \leq kcol \leq 10$.

3.2 The LOI spatial error estimation scheme

Here we provide a brief review of the LOI spatial error estimation scheme. The paper [4] provides further details. For a given input value for $kcol$, we compute the corresponding collocation solution, as described in the previous section. Then, based on evaluations of this collocation solution at selected points within $[a, b]$ (see below) we construct an interpolant that is of *one order lower* than the collocation solution. The collocation solution and the LOI are then used to compute spatial error estimates, similar to (9) and (10). (The quantity, $\bar{U}_s(x, t)$, appearing in (9) and (10), is replaced by the LOI.)

From the collocation theory outlined in the previous subsection, it is clear that evaluation of a collocation solution that is based on the use of $kcol$ collocation points per subinterval yields a solution approximation for which the spatial error is $O(h^{kcol+2})$. Thus evaluation of this collocation solution *at any point* within the subinterval yields a solution approximation for which the spatial error is $O(h^{kcol+2})$. On a given subinterval, the polynomial that represents the LOI is defined by requiring it to interpolate $kcol + 1$ values of this collocation solution or its first spatial derivative. This will yield an interpolant whose interpolation error is $O(h^{kcol+1})$. (Thus the interpolation error will dominate the collocation error; i.e., for sufficiently small h, the interpolation error will be larger than the data error associated with the collocation solution and derivative values.) The values chosen for interpolation are the collocation solution and its first spatial derivative at the endpoints of the subinterval (4 values) and $kcol - 3$ collocation solution values at points internal to the subinterval. The internal points are chosen so that *the interpolation error for the LOI on the given subinterval will be asymptotically equivalent to the collocation error for a collocation solution based on the use of* $kcol - 1$ *collocation points per subinterval.* (For a given subinterval, this turns out to require the internal interpolation points to be the internal superconvergent points of a collocation solution that uses $kcol-1$ collocation points per subinterval.) The explicit form of the leading term in the collocation error is known from the basic theory for collocation and it is then possible to choose the interpolation points so that the leading term in the interpolation error matches the leading term in the collocation error. See [4] for further discussion.

Because the polynomial interpolant representing the LOI on a given subinterval uses a combination of derivative and solution values, a Hermite-Birkhoff representation for the polynomial interpolant is again appropriate and we again use the general forms for the Hermite-Birkhoff basis polynomials given in [16]. Since all of the interpolated values are associated with points contained within the given subinterval, the corresponding interpolation error expression does not involve adjacent subinterval size ratios, as in the SCI case, and thus the spatial error estimates for the LOI scheme are not sensitive to the presence of adjacent subintervals of significantly different sizes when nonuniform meshes arise.

The paper [4] provides experimental results comparing the spatial error estimates of the original BACOL code with those computed by the SCI and LOI schemes. These results show that the three schemes generally provide error estimates of comparable accuracy (except in the case of the SCI scheme when there are adjacent subintervals differing greatly in size, as mentioned in the previous subsection.)

As mentioned earlier, when BACOLI uses the LOI scheme for spatial error estimation, it returns a collocation solution which is based on the use of $kcol$ collocation points per subinterval, with spatial error control based on a spatial error estimate that agrees asymptotically with that of a collocation solution of one lower order. This means that, in this case, the code is operating in *LE error control mode*.

BACOLI, when using the LOI scheme, will restrict $kcol$ so that $3 \leq kcol \leq 10$. This will mean that the LOI scheme will generate an error estimate corresponding to a collocation solution for which the $kcol$ range will be $2 \leq kcol \leq 9$. The collocation solution returned by BACOLI in this case will have a $kcol$ value in the range $3 \leq kcol \leq 10$.

3.3 Collocation Solutions and Spatial Error Control Modes

In this subsection, we summarize the four combinations of collocation solution/spatial error control modes reviewed earlier in this section. (Assume that the input value for the number of collocation points per subinterval is $kcol$ and let $p = kcol + 1$.):

- **BACOL with ST Error Control (BAC/ST)**: The original BACOL code returns a collocation solution of degree p, obtained by controlling a spatial error estimate *for that collocation solution*. (The spatial error estimate is obtained by comparing the collocation solution with another collocation solution of degree $p + 1$.) The allowed range for $kcol$ is $2 \leq kcol \leq 10$.

- **BACOL with LE Error Control (BAC/LE)**: A modified version of BACOL that returns a collocation solution of degree $p + 1$, obtained by controlling a spatial error estimate for a collocation solution of degree p. (The spatial error estimate is obtained by comparing the collocation solution with another collocation solution of degree p.) The allowed range for $kcol$ is $3 \leq kcol \leq 11$.

- **BACOLI with the SCI scheme - ST Error Control (SCI/ST)**: BACOLI returns a collocation solution of degree p, obtained by controlling a spatial error estimate *for that collocation solution*. (The spatial error estimate is obtained by comparing the collocation solution with an SCI.) The allowed range for $kcol$ is $3 \leq kcol \leq 10$.

- **BACOLI with the LOI scheme - LE Error Control (LOI/LE)**: BACOLI returns a collocation solution of degree p, obtained by controlling

a spatial error estimate for a collocation solution of degree $p-1$. (The spatial error estimate is obtained by comparing the collocation solution with an LOI whose interpolation error agrees asymptotically with the collocation error of a collocation solution of degree $p-1$.) The allowed range for *kcol* is $3 \leq kcol \leq 10$.

We mention as an aside that it would be possible to modify BACOLI so that it (i) uses the SCI scheme but operates in LE Error Control mode, or (ii) uses the LOI scheme but operates in ST Error Control mode. In case (i) the code would have to be modified to return the SCI as the approximate solution; in case (ii) the code would have to return the LOI as the approximate solution. We do not investigate this point any further in this report.

4 Modification of BACOL to obtain BACOLI

Modifications to a number of important components of the BACOL package were required in order to obtain BACOLI. We refer the reader to Figure 1 of [30] which provides a detailed structure diagram of all of the program modules that make up the BACOL package.

The modifications required to obtain BACOLI from BACOL proceeded in four phases:

1. The primary part of this phase involved the modification of ERREST, the routine that computes the spatial error estimate. New code was introduced to implement the SCI and LOI schemes; this involved writing new subroutines, SCINT and LOWINT, that are called within ERREST. *No effort was made in this phase to remove the computation of the second collocation solution.* The resultant code had the options of generating a spatial error estimate based on the computation of (i) a second collocation solution (as in the original code), (ii) the SCI scheme, or (iii) the LOI scheme. This allowed the code to be run using any of the three spatial error estimation schemes and thus the quality of the spatial error estimates and the performance of the code could be compared in these three cases. These results were reported in the papers [2] and [4], as mentioned earlier. These studies experimentally examined the accuracy of the three spatial error estimation schemes but did not investigate potential efficiency improvements. In this phase, we did not pay close attention to attempting to optimize the efficiency of the implementations of the SCINT and LOWINT routines.

2. This phase involved the important step of removing the computation of the second collocation solution. Three significant changes were required:

   - In the original BACOL code, after a warm restart, both the lower order and higher order collocation solutions on the new mesh were reinitialized using the higher order solution. Since BACOLI computes only one solution, the code had to be modified so that after a

remeshing, the reinitialization process would make use of this solution. This involved changes to the parameters passed to the SUCSTP and REINIT routines. The REINIT routine is now called only once after a remeshing. Similarly, the INIY and INIYP routines are now each called only once whenever the B-spline coefficients and their derivatives need to be initialized. Also, the COLPNT routine, which computes the collocation points for the current spatial mesh, and the DIVDIF routine, which is called to compute past time step values required by DASSL, are now called only once.

- In the original BACOL code, we recall that the two DAE systems associated with the computation of the lower and higher order collocation solutions are passed to DASSL simultaneously as one "double" (but uncoupled) DAE system in order to ensure synchronization of the time-stepping for the two collocation solutions. Thus, for BACOLI, the parameters passed to DASSL had to be modified so that only the data for the one collocation solution being computed was passed. This involved changes to the 'NEQ', 'RPAR', 'IPAR', 'RWORK', and 'IWORK' parameters of DASSL. We also had to modify the DDASLV and DDAJAC routines (called by DASSL) so that only the computations for one collocation solution are performed. In particular only one call each to the ABD matrix factorization routine, CRDCMP, and ABD linear system solve routine, CRSLVE, are required. Similarly, only one call to each of the CALJAC and CALRES routines that compute, respectively, the coefficient matrix for the Newton system and the right hand side of the Newton system that arise during the computations performed by DASSL, are required.
- Additional modification of the ERREST routine was required so that it no longer accessed information associated with two collocation solutions in order to obtain a spatial error estimate. This routine was modified so that it accessed only the information for the single available collocation solution and then used that information to generate a spatial error estimate based on either the SCI or LOI scheme, through a call to either the SCINT or LOWINT routine.
- Communication between the user's main program and BACOL or BACOLI is handled through a communication vector called 'MFLAG'. In BACOLI an extra entry was added to this vector to allow the user to select the spatial error estimation/control scheme: 'MFLAG(8) = 0' for the LOI scheme, 'MFLAG(8) = 1' for the SCI scheme. Alternatively, we can say that these choices correspond to, respectively, LE error control mode and ST error control mode.

Upon completion of these modifications we obtained a new version of BACOL (BACOLI) that no longer computed a second collocation solution and was able to use either of the two interpolation-based spatial error estimation schemes to compute a spatial error estimate which could then

be used to drive the mesh refinement/spatial error control process. The resultant version of BACOLI, for a given test problem, had performance generally comparable to that of BACOL in terms of the number of mesh-points it used, the number of remeshings it required, and the total number of time steps it took. We provide results in the next section to demonstrate this machine independent performance of BACOL vs. BACOLI.

However, preliminary machine dependent timing results showed that BACOLI, for some problems, actually ran more slowly than the original BACOL code! Thus, a third phase was undertaken, in which the efficiency of the implementations of the SCINT and LOWINT algorithms was addressed.

3. The work undertaken in this phase was primarily guided by execution time profiling of the code. We used the 'gprof' utility [18] in order to identify routines which took the significant amounts of execution time. See Figures 75 and 76 for examples of the profiling of BACOL and BACOLI. These figures were obtained using the Gprof2Dot [17] utility.

   - The execution time analysis showed that the calls from within the SCINT and LOWINT to the B-spline evaluation routines BSPVLD and BSPVLN required relatively large amounts of time. We were able to substantially improve the efficiency of the SCINT and LOWINT routines by modifying them to save the B-spline evaluations between remeshings. These values were saved in the 'RPAR' array. This change also involved modification of the ERREST routine since in the original BACOL code B-spline package routines were not called directly, but rather through the routine called VALUES. In BACOLI, we make direct calls to the B-spline routines BSPLVN and BSPLVD in order to improve the efficiency of the evaluation of the B-spline basis polynomials.
   - Further savings in execution time for the SCINT and LOWINT routines were obtained by making a similar change in the way the coefficients of the Hermite-Birkhoff polynomials representing the interpolants were handled. The Hermite-Birkhoff coefficients are saved and reused between remeshings. These coefficients are also stored in the 'RPAR' array.
   - Another efficiency gain was made by employing a Barycentric form for the Hermite-Birkhoff interpolants; see, e.g., [11] and references within. To be specific, this involved factoring common factors from the basis polynomials that appear in the representation of the Hermite-Birkhoff interpolants we employ - see [2].
   - Another efficiency gain, revealed by some experimentation with different compiler optimization settings, involved a slight modification to the EVAL routine to improve the computation of array indices.

4. This last phase focused on the SCI implementation and was motivated by an observation we made while testing the performance of BACOLI when it uses the SCI scheme.

    - As discussed earlier, the SCI scheme can overestimate the spatial error for a given subinterval when the ratio of the size of that subinterval to the size of either adjacent subinterval is large. Since it is easy to compute these ratios *we can predict when the spatial error estimate from the SCI scheme will be too large* and then attempt to address the issue. We therefore modified BACOLI by scaling the SCI spatial error estimates for each subinterval by the reciprocal of the product of the adjacent subinterval size ratios. (In the case of the leftmost and rightmost subintervals, we scale by the square of the ratio of the size of that subinterval to the size of the lone adjacent subinterval.) This scaling is motivated by the appearance of these ratios in the expressions for the SCI spatial error estimates, mentioned in Section 3.1 (and discussed in more detail in [2]). The scaling reduces the size of the SCI spatial error estimates when they overestimate the spatial error, leading to closer agreement between the true error and the estimated spatial error and thus better adaptation of the spatial mesh, which in turn improves the efficiency of the BACOLI/SCI computation.

We also made a number of small changes involving small bug fixes that became apparent during the modification process. These are documented within the source code of BACOLI.

The BACOLI parameter list is the same as that of BACOL except for two differences:

- As noted above, the 'MFLAG' vector now has an extra (eighth) component used to specify the error estimation/error control mode.
- We also generalized the interface to BACOLI to allow the user to specify the names of the routines corresponding to the PDEs and their derivatives, and the boundary conditions and their derivatives, and the initial conditions. To be specific, seven new parameters have been added to the end of the BACOLI parameter list: *f, derivf, bndxa, difbxa, bndxb, difbxb, uinit,* corresponding to the above routines.

See [30] for a detailed description of the use of BACOL, and in particular its associated online Appendix. The BACOLI webpage, where the source code for BACOLI and a number of examples are posted, is
http://cs.smu.ca/∼muir/BACOLI_Webpage.htm.

5 Numerical Comparison: BACOL vs. BACOLI

In this section we provide experimental results which allow a comparison of the efficiency of BACOL with BACOLI using the SCI scheme or the LOI scheme.

5.1 Test problems

Test Problem I:
The One Layer Burgers equation (OLBE) - see, e.g., [31]:

$$u_t = \epsilon u_{xx} - u u_x, \tag{11}$$

with boundary conditions at $x = 0$ and $x = 1$ ($t > 0$) and an initial condition at $t = 0$ ($0 \le x \le 1$) taken from the exact solution

$$u(x,t) = \frac{1}{2} - \frac{1}{2}\tanh\left(\frac{x - \frac{t}{2} - \frac{1}{4}}{4\epsilon}\right),$$

where ϵ is a problem dependent parameter. We will consider $\epsilon = 10^{-3}$ and $\epsilon = 10^{-4}$. For $\epsilon = 10^{-4}$, the exact solution is plotted in Figure 1. (This plot and all subsequent plots presented in this report were prepared using matplotlib software package [21].)

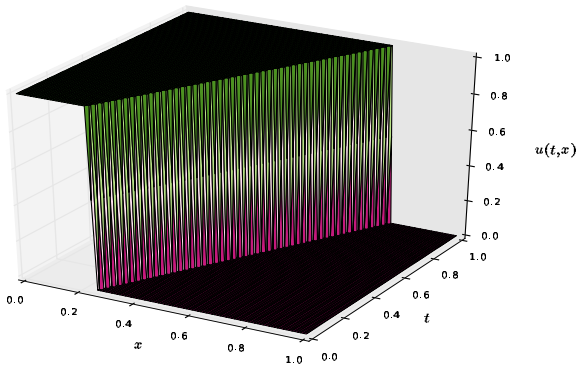

Figure 1: *Solution of the One Layer Burgers equation with* $\varepsilon = 10^{-4}$. *There is a sharp layer region at* $x \approx 0.25$ *when* $t = 0$. *As t goes from 0 to 1, the layer moves to the right and is located at* $x \approx 0.75$ *when* $t = 1$.

Test Problem II:
The Catalytic Surface Reaction Model (CSRM) - see, e.g., [31]:

$$\begin{aligned}
(u_1)_t &= -(u_1)_x + n(D_1 u_3 - A_1 u_1 \gamma) + (u_1)_{xx}/Pe_1, \\
(u_2)_t &= -(u_2)_x + n(D_2 u_4 - A_2 u_2 \gamma) + (u_2)_{xx}/Pe_1, \\
(u_3)_t &= A_1 u_1 \gamma - D_1 u_3 - R u_3 u_4 \gamma^2 + (u_3)_{xx}/Pe_2, \\
(u_4)_t &= A_2 u_2 \gamma - D_2 u_4 - R u_3 u_4 \gamma^2 + (u_4)_{xx}/Pe_2,
\end{aligned} \tag{12}$$

where $\gamma = 1 - u_3 - u_4$, and $n, r, Pe_1, Pe_2, D_1, D_2, R, A_1$, and A_2 are problem dependent parameters. The initial conditions at $t = 0$ $(0 \le x \le 1)$ are

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

and the boundary conditions at $x = 0$ and $x = 1$ $(t > 0)$ are

$$(u_1)_x(0,t) = -Pe_1(2 - r - u_1(0,t)), \quad (u_2)_x(0,t) = -Pe_1(r - u_2(0,t)),$$

$$(u_3)_x(0,t) = (u_4)_x(0,t) = 0,$$

$$(u_1)_x(1,t) = (u_2)_x(1,t) = (u_3)_x(1,t) = (u_4)_x(1,t) = 0.$$

(To our knowledge, this problem does not have a closed form solution.) With $Pe_1 = Pe_2 = 10000, D_1 = 1.5, D_2 = 1.2, R = 1000, r = 0.96, n = 1$, and $A_1 = A_2 = 30$, a plot of the approximate solution components are shown in Figures 2-5.

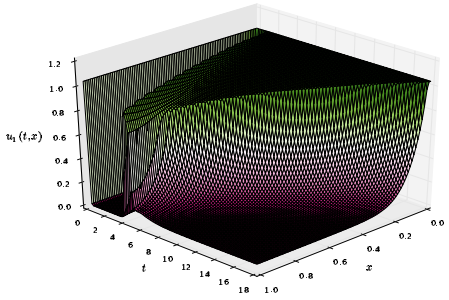

Figure 2: *Solution component, $u_1(x,t)$, for the Catalytic Surface Reaction Model.*

Test Problem III:
The Two Layer Burgers equation (TLBE) - see, e.g., [31] - has the same PDE as the One Layer Burgers equation - Test Problem I, above. However, it has different initial and boundary conditions; the boundary conditions at $x = 0$ and $x = 1$ $(t > 0)$ and the initial condition at $t = 0$ $(0 \le x \le 1)$ are taken from the exact solution,

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

where,

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

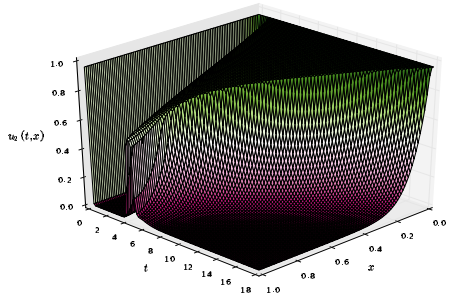

Figure 3: *Solution component, $u_2(x,t)$, for the Catalytic Surface Reaction Model.*

where ϵ is a problem dependent parameter. We will consider $\epsilon = 10^{-3}$ and $\epsilon = 10^{-4}$. A plot of the solution to this problem with $\epsilon = 10^{-4}$ is given in Figure 6.

Test Problem IV:
A PDE system consisting of 6 copies of the Two Layer Burgers equation (TLBE×6). We consider $\epsilon = 10^{-3}$ and $\epsilon = 10^{-4}$.

Test Problem V:
A PDE system consisting of 12 copies of the Two Layer Burgers equation (TLBE×12). We consider $\epsilon = 10^{-3}$ and $\epsilon = 10^{-4}$.

There are thus a total of nine test problems: I (with $\epsilon = 10^{-3}$ or $\epsilon = 10^{-4}$), II, III (with $\epsilon = 10^{-3}$ or $\epsilon = 10^{-4}$), IV (with $\epsilon = 10^{-3}$ or $\epsilon = 10^{-4}$), and V (with $\epsilon = 10^{-3}$ or $\epsilon = 10^{-4}$).

5.2 Machine Independent Efficiency Tests

In this subsection, we will compare the efficiency of BAC/ST, BAC/LE, SCI/ST, and LOI/LE with respect to machine independent measures. (These code combinations were identified in Section 3.3.)

For all tests, we will choose the input kcol value so that we are comparing computations that return collocation solutions based on the same number of collocation points per subinterval; only the spatial error estimation/control schemes are different.

This collection of tests provides machine independent results that will allow us to compare the number of subintervals used in the spatial mesh at the final time (Final Nint), the total number of accepted time steps taken (Accepted Time Steps), the total number of spatial remeshings undertaken throughout the computation (Remeshings), the total number of cold starts (Cold Starts), and the total number of factorizations and backsolves of an ABD system (calls to

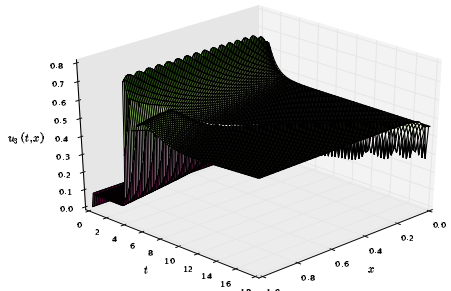

Figure 4: *Solution component,* $u_3(x,t)$*, for the Catalytic Surface Reaction Model.*

CRDCMP, CRSLVE). These quantities all contribute significantly to the total amount of work performed by the code and thus allow for good machine independent comparisons of the codes. We consider $kcol = 3, 4, 6, 8$, tolerances, $ATOL_s = RTOL_s = tol = 10^{-4}, 10^{-6}, 10^{-8}$, and all nine test problems identified in the previous subsection. (As mentioned earlier, BAC/ST can be run with $2 \leq kcol \leq 10$, BAC/LE with $3 \leq kcol \leq 11$, while the range for both SCI/ST and LOI/LE is $3 \leq kcol \leq 10$. Results for BAC/LE with $kcol = 3$ are equivalent to BAC/ST results for $kcol = 2$ so we can use the former to also examine the performance of BAC/ST with $kcol = 2$.)

Results are given in Tables 1-9. A missing table entry means that the code failed during the test run. In each case the failure occurred at the beginning of the computation when the code was unable, within 20 attempts, to adapt to a satisfactory initial spatial mesh, such that an approximate solution, satisfying the user tolerance, was obtained.

We see from these tables that the relative performances of BAC/ST, BAC/LE, SCI/ST, and LOI/LE exhibit some variation from problem to problem and tolerance to tolerance as $kcol$ varies. Considering all test cases, we observe that there are isolated cases where one code performs worse than the others in terms of one of the machine independent measures, but, generally, these are not consistent trends. The codes are generally comparable with respect to all machine independent measures, over the full range of test problems, except for the total number of ABD matrix factorizations (i.e., calls to CRDCMP) and ABD system solves (i.e., calls to CRSLVE). An observation that holds for a large majority of the tests is that the number of factorizations and solves performed by BAC/ST or BAC/LE is roughly about twice as many as performed by SCI/ST or LOI/LE.

Comparing the performance of each of the codes when $kcol = 3$ with itself using higher $kcol$ values, we generally see inferior performance (or at best com-

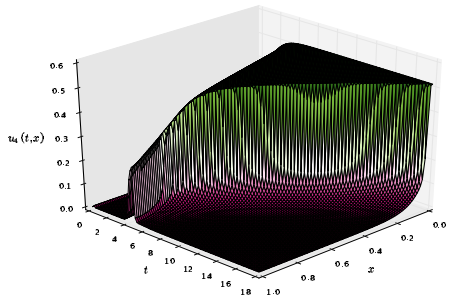

Figure 5: *Solution component,* $u_4(x,t)$*, for the Catalytic Surface Reaction Model.*

parable performance) in the $kcol = 3$ case. Similar comments apply when we consider the BAC/ST $kcol = 2$ results (obtained from the BAC/LE $kcol = 3$ results, as mentioned above). It is usually the case that when $kcol = 3$, the codes require more meshpoints than for higher $kcol$ values. The SCI/ST code has a failure rate that is significantly higher than the other codes; all of the failures that arise in this set of tests except for one are attributable to the SCI/ST code (the other failure occurs with the LOI/LE code ($kcol = 6, tol = 10^{-8}$)). Most of the SCI/ST code failures occur for the $kcol = 3$ case; the rest occur for the $kcol = 4$ case. (The reason for this poor performance by the SCI/ST code for low $kcol$ values may be related to the fact that the collocation error expression for each subinterval includes a local term (which generally dominates), as well as a global term. There may be non-local contributions to the collocation error on a given subinterval that are comparable in size to the local term and in this case the internal collocation solution values employed by the SCI scheme may not be sufficiently accurate, implying that the SCI scheme will not be sufficiently accurate to be effective in the error estimate.) Given the high proportion of failures, it appears to be the case that the SCI/ST code should not be used with $kcol = 3$. For the $kcol = 3$ case, the LOI/LE code requires significantly more remeshings than the other codes; this adds significantly to the costs for this code. (This poorer performance may be related to a poor quality approximation of the collocation error of the $kcol = 2$ collocation solution by the LOI scheme due to contributions to the error from higher order terms that are comparable in size to the leading order interpolation error term.) Given the additional costs associated with the extra remeshings, it appears that the LOI/LE code should not be run with $kcol = 3$.

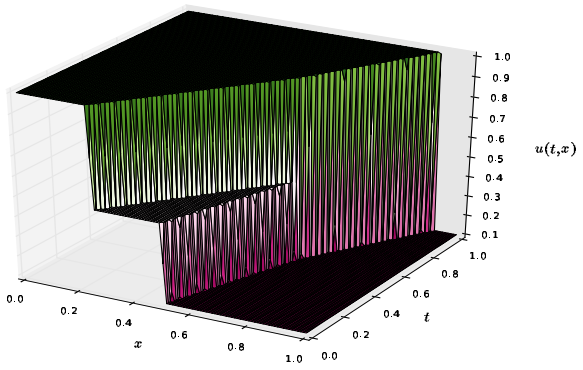

Figure 6: *Solution of Two Layer Burgers equation with* $\varepsilon = 10^{-4}$. *When* $t = 0$, *there are two sharp layers at* $x \approx 0.25$ *and* $x \approx 0.5$. *As* t *increases, these layers move to the right and merge, forming a single layer at* $x \approx 0.7$ *when* $t \approx 0.5$. *As time increases further, the single layer continues to move to the right, and is located at* $x \approx 0.9$ *when* $t = 1$.

5.3 Machine Dependent Efficiency Tests

This collection of tests provide machine dependent results that will allow us to compare execution times for BAC/ST, BAC/LE, SCI/ST, and LOI/LE. These tests were performed on an HP Compaq 6005 Pro SFF PC with a Quad core AMD Phenom II X4 B95 processor (clock speed 3000.00 MHz). The operating system was Arch Linux (3.9.4-1-ARCH); the compiler was gfortran (4.8.0) with optimization level set to -O2. (*As in the previous set of tests, we will choose the input kcol value so that we are comparing computations that return collocation solutions based on the same number of collocation points per subinterval.*)

5.3.1 Timing Tables

The first set of results give tables of timings (in seconds). Each test was run three times and the reported time is the average of the three tests. We compare the codes over a selected set of *kcol*-tolerance combinations over the nine test problems specified in Section 5.1. We consider $kcol = 3, \ldots, 10$ and tolerances, $ATOL_s = RTOL_s = tol = 10^{-4}, 10^{-6}, 10^{-8}$. Results are given in Tables 10-18. A missing table entry means that the code failed during the test run. In most of the cases in which there was a failure, the difficulty occurred at the beginning of the computation when the code was unable, within 20 attempts, to adapt to a satisfactory initial spatial mesh. In the remaining cases, all of which arise in Table 12, the failures arise because DASSL terminates its computation due to

the time stepsize becoming too small. Almost all the failures occurring in all the tables are associated with the SCI/ST code, especially for the $kcol = 3$ case; over all test cases, the LOI/LE code has only two failures and the BAC/ST and BAC/LE codes have only one failure each.

From Tables 10-18, we can make a number of general observations:

- The execution times for SCI/ST and LOI/LE are generally significantly lower than the corresponding times for BAC/ST and BAC/LE. With the exception of the LOI/LE code for $kcol = 3$, BAC/ST is slower than SCI/ST or LOI/LE, for smaller $kcol$ values, by a factor of about 1.5. For larger $kcol$ values, it is slower by a factor of almost 2.

- BAC/LE is generally about 20% faster than BAC/ST. The SCI/ST and LOI/LE codes are generally faster than BAC/LE (except for the LOI/LE, $kcol = 3$ case) but the speedup factors are somewhat smaller than for the BAC/ST code.

- SCI/ST and LOI/LE are generally of comparable speed (except for the LOI/LE, $kcol = 3$ case)

- The SCI/ST code has a higher failure rate than the other codes; the failure rate for the $kcol = 3$ case is particularly high.

- For $kcol = 3$, the LOI/LE code has poorer performance than the other codes (due to the larger number of remeshings that it performs).

Another general observation from the tables is that all codes are faster for smaller $kcol$ values (with the exception of the LOI/LE code for $kcol = 3$) when the tolerance is not too sharp. (Generally, $kcol = 3$ gives the best times for coarse or intermediate tolerance values, with the exception of the LOI/LE code for $kcol = 3$.) When $tol = 10^{-8}$, higher $kcol$ values yield the fastest performances. The most extreme case appears in Table 10, when $tol = 10^{-8}$; the LOI/LE code is fastest when $kcol = 8$. (Generally $kcol = 4$ or 5 gives the best results for the sharpest tolerance.) The codes generally run slowest for the higher $kcol$ values.

5.3.2 Timing Ratio Tables

In Tables 19 and 20, we summarize the timing results from Tables 10-18. For each reported timing result appearing as a table entry in each of the Tables 10-18, we compute the ratio of that time to the corresponding BAC/ST time. We also compute the ratio of the LOI/LE time to the corresponding SCI/ST time. In Table 19 we report the average of these ratios over all $kcol$ values and tolerances, for each problem. In Table 20 we report the average of the timing ratios over all problems and tolerances, for each $kcol$ value.

From Table 19 we observe that the $\frac{\text{BAC/LE}}{\text{BAC/ST}}$ ratio averages range from slightly over 0.90 to about 0.80. The $\frac{\text{SCI/ST}}{\text{BAC/ST}}$ and $\frac{\text{LOI/LE}}{\text{BAC/ST}}$ ratio averages range from

about 0.60 to about 0.50. The $\frac{\text{LOI/LE}}{\text{SCI/ST}}$ ratio averages range from about 1.3 to slightly less than 1.0. From Table 20, we observe that the $\frac{\text{BAC/LE}}{\text{BAC/ST}}$ ratio averages range from slightly more than 1.0 for the smallest *kcol* value to around 0.84 for the larger *kcol* values. The $\frac{\text{SCI/ST}}{\text{BAC/ST}}$ ratio averages are slightly above 0.5. For the two smallest *kcol* values the $\frac{\text{LOI/LE}}{\text{BAC/ST}}$ ratio averages are 0.92 and 0.61; for higher *kcol* values these ratio averages are slightly above 0.5. The $\frac{\text{LOI/LE}}{\text{SCI/ST}}$ ratio averages range from 1.66 for $kcol = 3$ down to about 0.95 for the larger *kcol* values. Over all problems, *kcol* values, and tolerances the ratio averages are $\frac{\text{BAC/LE}}{\text{BAC/ST}} = 0.88$, $\frac{\text{SCI/ST}}{\text{BAC/ST}} = 0.53$, $\frac{\text{LOI/LE}}{\text{BAC/ST}} = 0.58$, and $\frac{\text{LOI/LE}}{\text{SCI/ST}} = 1.06$.

5.3.3 Work vs. Accuracy Plots by Problem

The previous results compared the execution time of the four codes when they were given the same tolerance. Here we compare the execution times of the codes with respect to the accuracy they achieve. We consider the One Layer and Two Layer Burgers equations with $\epsilon = 10^{-3}$ and 10^{-4}, since both have known exact solutions. We consider *kcol* values from 3 to 10. The results were obtained by running the codes with a range of 91 tolerance values from 10^{-1} to 10^{-10}. See Figures 7-38.

These figures show that, generally, the SCI/ST and LOI/LE codes are faster than the BAC/ST and BAC/LE codes over the range of problems and *kcol* values considered. The BAC/LE code is slightly faster than the BAC/ST code and the SCI/ST and LOI/LE codes have comparable performance. The performance of the LOI/LE code for $kcol = 3$ is significantly poorer than for higher *kcol* values.

5.3.4 Work vs. Accuracy Plots by Code

Figures 39-42 provide plots (four per figure) comparing each code over the range of *kcol* values on the One Layer and Two Layer Burgers equations with $\epsilon = 10^{-3}$ and 10^{-4}. That is, for each problem, each code is compared with itself over a range of *kcol* values. These plots also include a table recording, for each *kcol* value, the number of failures, the number of times when the actual error in the returned final solution was greater than the requested tolerance, and the number of times when the returned final solution was greater than 10 times the requested tolerance.

From these figures, a general observation is that, for low accuracy, the codes are more efficient when *kcol* is small, while for high accuracy, a higher choice of *kcol* provides the most efficient computation. For higher accuracy, low *kcol* values lead to substantially poorer results than do higher *kcol* values. The LOI/LE code with $kcol = 3$ is especially inefficient for high accuracy computations.

From the embedded tables that show the number of failures and number of times that the tolerance is exceeded, we see that *the BAC/ST, BAC/LE, and SCI/ST codes are more reliable (i.e., the actual error is within the requested tolerance) for higher kcol values.* The SCI/ST code exhibits substantially more

failures for the $kcol = 3$ case. The LOI/LE code is relatively more reliable than the other codes across all $kcol$ values. These trends are more pronounced for the more difficult problems (in which $\epsilon = 10^{-4}$). All codes have more failures and the actual error exceeds the tolerance more often for low $kcol$ values but reliability is improved in all codes for larger $kcol$ values. The SCI/ST code shows a number of instances when the actual error exceeds 10 times the tolerance; this never happens with the other codes.

5.3.5 Relative Work vs. Accuracy Plots by Problem

These plots are obtained from the data that was used to generate the work-accuracy plots discussed earlier. Rather than report execution times, these plots consider the execution times of the BAC/LE, SCI/ST, and LOI/LE codes relative to a corresponding BAC/ST code execution time. The data was analyzed and the plots developed as follows. We describe this process first for the BAC/LE data.

- We first perform a linear fit to the log of the error vs. log of time data associated with the BAC/ST code in order to obtain a continuous representation of the baseline BAC/ST data.
- Then, for each (error,time) ordered pair from the BAC/LE data set, we use the linear fit associated with the BAC/ST data to obtain a corresponding time estimate for the BAC/ST code (i.e., an estimate of how much time the BAC/ST code would take to compute a solution with the same error as the BAC/LE code).
- We then compute the ratio of the actual BAC/LE time to this estimated time. This yields a set of ordered pairs of the form (error, time ratio) that we can associate with the BAC/LE code.
- Finally we fit a line to the log of the error and the time ratios and plot this line on a semi-log scale.

This process is repeated for the SCI/ST data and the LOI/LE data. For reference the BAC/ST line is also shown in these plots.

Figures 43-74 show that, generally, the SCI/ST and LOI/LE codes are substantially faster (almost 50% faster in some cases) than the BAC/ST code, and that the BAC/LE code is generally somewhat faster than the BAC/ST code, although the difference between the two codes diminishes as the accuracy increases. (In fact, Figure 52 shows that for intermediate to high accuracies, BAC/LE is substantially slower than BAC/ST.) The performance of the LOI/LE code for $kcol = 3$ also departs from the general pattern described above. While it is faster than the other codes for low accuracy, it loses this advantage for higher accuracy.

6 Summary, Conclusions and Future Work

6.1 Summary and Conclusions

This report describes the modifications that were applied to the BACOL software package in order to obtain the new package, BACOLI, that replaces the expensive computation of a second collocation solution with the construction of a low cost interpolant for use in the spatial error estimate. The interpolants are described in detail in [2] and [4]. The former paper describes an algorithm (the SCI scheme) in which the higher order collocation solution is replaced by a superconvergent piecewise polynomial interpolant; error control is standard (ST) mode in this case: the spatial error estimate is appropriate for the collocation solution that is returned to the user. The latter paper describes an algorithm (the LOI scheme) in which the lower order collocation solution is replaced by a lower order interpolant whose interpolation error agrees asymptotically with the collocation error of the lower order collocation solution; error control in this case is local extrapolation (LE) mode: the spatial error estimate is for a collocation solution that is of one lower order than that which is returned to the user. The new code BACOLI provides an option for the selection of either of these spatial error estimation/error control schemes. The choice of the SCI scheme specifies ST error control; the choice of the LOI scheme specifies LE error control mode.

This report also provides extensive numerical results that compare the original BACOL code with BACOLI running in each of the above modes. For each test, the input value for *kcol*, the number of collocation points per subinterval, is chosen so that a collocation solution of the same order is returned. Machine independent and machine dependent test results are provided for several test problems. The test results show that the BACOLI code (using either the SCI or LOI scheme) is substantially faster than BACOL; this is mainly due to the avoidance of the computation of two collocation solutions performed by BACOL. In some cases, BACOLI is shown to be about twice as fast as BACOL.

Comparing BACOL and BACOLI in ST or LE error control mode, the test results showed that all codes are generally faster for lower *kcol* values when the tolerance is coarse to intermediate (with the exception of BACOLI/LE with $kcol = 3$.) When the tolerance is sharp, higher *kcol* values, e.g. $kcol = 4$, 5, or 6, lead to faster computations. On the other hand, the BACOL code in either error control mode and the BACOLI code in ST error control mode do a better job of returning solutions for which the actual error meets the tolerance when larger *kcol* values are employed. BACOLI in LE error control mode, however, provides more reliable control of the error, across all tolerance ranges.

When $kcol = 3$, both the SCI/ST code (due to a high failure rate) and the LOI/LE code (due to a large number of remeshings) exhibit poor performance.

6.2 Future Work

There are a number of directions for future work. In earlier work, BACOLR was shown to be superior to BACOL; thus it would appear to be worthwhile to

investigate applying the ideas discussed in this report to BACOLR. Other ideas for future work include the development a Fortran95 version of BACOLI and a parallel version of BACOLI. Another possibility is to investigate a spatial error estimation scheme that would involve computing *both* of the interpolation-based spatial error estimates (perhaps some sharing of the intermediate computations can occur) and comparing them. It is possible that the two spatial error estimates could be used to guide an increase or decrease *kcol* in a remeshing while allowing a warm start. This would mean that BACOLI would then have a *p*-refinement capability. A related comment is that, since the numerical results presented here show that better efficiency for problems with coarse or intermediate level tolerances is obtained when smaller *kcol* values are used, it may also be appropriate to have BACOLI select the *kcol* value based on the user tolerance rather than leaving the choice of *kcol* up to the user. It is interesting to compare the results on the current paper with those in [31]. That paper considers, for BACOL, $p = 3$ and $p = 6$ (which correspond to $kcol = 2$ and $kcol = 5$) and the results of that paper show that $kcol = 2$ is better for very coarse tolerances (10^{-2} or 10^{-3}) but for sharper tolerances $kcol = 5$ is comparable or better.

Another significant direction for future work is the generalization of the approach to two-dimensional problems. Some preliminary work in this direction has been reported in [22].

In addition to the above future work projects, a number of other issues associated with BACOLI might also be investigated:

- When the SCI spatial error estimates are scaled to adjust for large mesh ratios (as discussed in Section 4), the performance of the SCI/ST version of BACOLI is improved. Currently, BACOLI uses the *product* of the adjacent subinterval ratios for internal intervals (and the square of the lone adjacent subinterval ratio for boundary subintervals) but another reasonable alternative would employ, for the internal subintervals, the *maximum* of the adjacent subinterval ratios. A more careful investigation of the interpolation error expression associated with the SCI scheme may be worthwhile.

- The L2-norm used in the error estimates could be replaced with a max norm. This would remove the need to use a quadrature formula to compute the integral that arises in the L2-norm. It may be that the knowledge we have of the specific polynomials that represent the approximate solution and interpolant on each subinterval could be used to select the best point to sample the spatial error estimate on each subinterval.

- There is an issue in the original BACOL with respect to underestimation of the spatial error in the sharp layer regions. This is also an issue for the SCI and LOI estimates. Further investigation of the error estimation schemes and the mesh refinement strategy is required.

References

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

[2] T. Arsenault, T. Smith, and P.H. Muir. Superconvergent interpolants for efficient spatial error estimation in 1d pde collocation solvers. *Can. Appl. Math. Q.*, 17:409–431, 2009.

[3] T. Arsenault, T. Smith, P.H. Muir, and P. Keast. Efficient interpolation-based error estimation for 1d time-dependent pde collocation codes. *Saint Mary's University, Dept. of Mathematics and Computing Science Technical Report Series,http://cs.smu.ca/tech_reports*, 2011.

[4] T. Arsenault, T. Smith, P.H. Muir, and J. Pew. Asymptotically correct interpolation-based spatial error estimation for 1d pde solvers. *Can. Appl. Math. Q.*, to appear, 2013.

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

[6] I. Babuška, M. Feistauer, and P. Šolín. On one approach to a posteriori error estimates for evolution problems solved by the method of lines. *Numer. Math.*, 89(2):225–256, 2001.

[7] I. Babuška and S. Ohnimus. A posteriori error estimation for the semidiscrete finite element method of parabolic differential equations. *Comput. Methods Appl. Mech. Engrg.*, 190(35-36):4691–4712, 2001.

[8] Ivo Babuška, John R. Whiteman, and Theofanis Strouboulis. *Finite elements*. Oxford University Press, Oxford, 2011.

[9] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

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

[11] R.M. Corless, A. Shakoori, D.A. Aruliah, and L. Gonzalez-Vega. Barycentric Hermite interpolants for event location in initial-value problems. *JNAIAM J. Numer. Anal. Ind. Appl. Math.*, 3(1-2):1–16, 2008.

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

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

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

[15] D.J. Estep, M.G. Larson, and R.D. Williams. Estimating the error of numerical solutions of systems of reaction-diffusion equations. *Mem. Amer. Math. Soc.*, 146(696):viii+109, 2000.

[16] W.F. Finden. An error term and uniqueness for Hermite-Birkhoff interpolation involving only function values and/or first derivative values. *J. Comput. Appl. Math.*, 212(1):1–15, 2008.

[17] J.R. Fonseca. http://code.google.com/p/jrfonseca/wiki/Gprof2Dot.

[18] GNU gprof. http://sourceware.org/binutils/docs/gprof/.

[19] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.

[20] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1996.

[21] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[22] Z. Li and P.H. Muir. B-spline Gaussian collocation software for two-dimensional parabolic pdes. *to appear in Adv. Appl. Math. Mech.*, 2013.

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

[24] P.K. Moore. Applications of Lobatto polynomials to an adaptive finite element method: a posteriori error estimates for *hp*-adaptivity and grid-to-grid interpolation. *Numer. Math.*, 94(2):367–401, 2003.

[25] P.K. Moore. Implicit interpolation error-based error estimation for reaction-diffusion equations in two space dimensions. *Comput. Methods Appl. Mech. Engrg.*, 192(39-40):4379–4401, 2003.

[26] P.K. Moore. An implicit interpolation error-based error estimation strategy for *hp*-adaptivity in one space dimension. *J. Numer. Math.*, 12(2):143–167, 2004.

[27] P.K. Moore. Interpolation error-based a posteriori error estimation for *hp*-refinement using first and second derivative jumps. *Appl. Numer. Math.*, 48(1):63–82, 2004.

[28] P.K. Moore. Effects of basis selection and h-refinement on error estimator reliability and solution efficiency for high-order methods in three space dimensions. *Int. J. Numer. Anal. Model.*, 3(1):21–51, 2006.

[29] T. Strouboulis, I. Babuška, and D.K. Datta. Guaranteed a posteriori error estimation for fully discrete solutions of parabolic problems. *Internat. J. Numer. Methods Engrg.*, 56(9):1243–1259, 2003.

[30] R. Wang, P. Keast, and P.H. Muir. BACOL: B-spline Adaptive COLlocation software for 1-D parabolic PDEs. *ACM Trans. Math. Software*, 30(4):454–470, 2004.

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

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

[33] R. Wang, P. Keast, and P.H. Muir. Algorithm 874: BACOLR—spatial and temporal error control software for PDEs based on high-order adaptive collocation. *ACM Trans. Math. Software*, 34(3):Art. 15, 28, 2008.

tol	10^{-4}	10^{-6}	10^{-8}
kcol	3		
BAC/ST	15, 1152, 83 (0) [380, 3984]	23, 2526, 117 (0) [564, 7866]	55, 6203, 110 (0) [754, 17086]
BAC/LE	16, 1144, 122 (1) [554, 4408]	42, 2471, 109 (0) [496, 7102]	111, 5708, 137 (0) [602, 13846]
SCI/ST	14, 1285, 71 (0) [188, 2151]	— —	54, 6970, 135 (0) [1123, 10541]
LOI/LE	17, 1102, 131 (0) [280, 2218]	45, 2197, 150 (0) [329, 3504]	124, 5715, 468 (0) [963, 9905]
kcol	4		
BAC/ST	14, 1160, 66 (0) [310, 3896]	16, 3058, 127 (0) [682, 9546]	32, 6578, 131 (1) [944, 18894]
BAC/LE	15, 1152, 83 (0) [380, 3984]	23, 2526, 117 (0) [564, 7866]	55, 6203, 110 (0) [754, 17086]
SCI/ST	13, 1196, 66 (0) [163, 1982]	18, 2932, 115 (0) [524, 4783]	35, 6509, 129 (0) [1200, 9981]
LOI/LE	15, 1153, 91 (0) [203, 2021]	23, 2462, 143 (0) [325, 4000]	50, 5845, 151 (0) [416, 8144]
kcol	6		
BAC/ST	10, 1479, 49 (0) [376, 4810]	14, 3159, 76 (2) [542, 9108]	15, 6644, 163 (1) [2256, 20770]
BAC/LE	14, 1122, 63 (0) [288, 3730]	15, 3045, 87 (0) [538, 8918]	22, 6192, 160 (0) [1396, 18438]
SCI/ST	12, 1283, 51 (0) [128, 2031]	14, 3132, 70 (0) [436, 4864]	19, 6279, 126 (0) [1120, 9658]
LOI/LE	11, 1253, 66 (0) [154, 2042]	15, 3145, 110 (1) [390, 4912]	22, 6637, 164 (2) [686, 9718]
kcol	8		
BAC/ST	10, 1530, 45 (0) [302, 5008]	15, 2763, 63 (0) [378, 7814]	13, 6447, 108 (1) [2128, 19302]
BAC/LE	10, 1617, 47 (0) [372, 5168]	15, 2751, 65 (0) [462, 7832]	14, 5947, 111 (1) [1838, 17494]
SCI/ST	10, 1715, 42 (0) [219, 2740]	14, 3223, 62 (0) [440, 4887]	14, 6771, 103 (0) [1071, 9938]
LOI/LE	10, 1366, 56 (0) [150, 2214]	15, 2861, 68 (0) [198, 3989]	15, 6381, 128 (0) [1049, 9709]

Table 1: *Machine independent results for the One Layer Burgers equation with* $\epsilon = 10^{-3}$. *We consider kcol* $= 3, 4, 6, 8$ *and tol* $= 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol		3	
BAC/ST	13, 13321, 753 (2) [4054, 46242]	22, 33345, 1188 (2) [6182, 101576]	46, 68868, 1245 (0) [10758, 191580]
BAC/LE	17, 12475, 907 (1) [4186, 45014]	37, 27789, 1074 (0) [4862, 84942]	88, 70200, 1247 (0) [7956, 196708]
SCI/ST	16, 13127, 741 (2) [2370, 23726]	— —	— —
LOI/LE	20, 10870, 1213 (2) [2551, 21540]	40, 24481, 1414 (0) [3002, 38537]	116, 45139, 3998 (1) [8061, 81210]
kcol		4	
BAC/ST	14, 13148, 585 (2) [3324, 43788]	18, 33814, 1215 (4) [12196, 112460]	27, 60633, 1322 (3) [15030, 178116]
BAC/LE	13, 13321, 753 (2) [4054, 46242]	22, 33345, 1188 (2) [6182, 101576]	46, 68868, 1245 (0) [10758, 191580]
SCI/ST	14, 13819, 655 (0) [2445, 24439]	— —	— —
LOI/LE	15, 18120, 997 (12) [2385, 23825]	23, 29705, 1161 (2) [2821, 45541]	48, 65420, 1301 (1) [4308, 94426]
kcol		6	
BAC/ST	13, 14012, 473 (0) [2868, 45246]	15, 32398, 675 (5) [8526, 97292]	18, 78698, 1473 (5) [27404, 223812]
BAC/LE	15, 12112, 558 (0) [2762, 39984]	15, 34594, 1117 (4) [13030, 115922]	25, 64011, 1411 (3) [21898, 198946]
SCI/ST	15, 14141, 429 (0) [1397, 22721]	14, 36481, 967 (1) [7231, 59718]	22, 67213, 1169 (1) [11332, 101223]
LOI/LE	15, 12780, 637 (4) [1781, 21623]	15, 56124, 1215 (7) [6576, 58230]	22, 77296, 1372 (11) [9142, 92835]
kcol		8	
BAC/ST	14, 14096, 452 (1) [2922, 45800]	14, 34498, 535 (1) [8694, 103136]	14, 67504, 921 (1) [20058, 195896]
BAC/LE	13, 15843, 416 (0) [3442, 50752]	14, 33700, 588 (4) [8868, 101146]	15, 74574, 1491 (3) [29160, 237118]
SCI/ST	14, 15041, 386 (0) [1434, 23982]	15, 36965, 579 (0) [5518, 55984]	15, 71911, 1312 (2) [12869, 109570]
LOI/LE	14, 12835, 548 (0) [1437, 21268]	15, 33800, 633 (0) [4713, 51899]	15, 93326, 1670 (23) [15500, 119843]

Table 2: *Machine independent results for the One Layer Burgers equation with* $\epsilon = 10^{-4}$. *We consider kcol* $= 3, 4, 6, 8$ *and tol* $= 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol		3	
BAC/ST	19, 3537, 282 (3) [1296, 12796]	50, 14089, 430 (6) [2122, 34638]	81, 25690, 581 (3) [3010, 69910]
BAC/LE	32, 3307, 311 (4) [1352, 12306]	86, 11704, 611 (4) [2716, 36896]	215, 24412, 647 (4) [3154, 65636]
SCI/ST	19, 5691, 173 (0) [478, 8591]	43, 11876, 332 (0) [890, 17254]	92, 24745, 413 (0) [1547, 33524]
LOI/LE	50, 3669, 432 (3) [931, 7386]	97, 11665, 755 (4) [1655, 19207]	262, 24625, 1221 (8) [2652, 36259]
kcol		4	
BAC/ST	15, 4715, 256 (2) [1248, 15770]	33, 12815, 421 (0) [2128, 38080]	55, 25475, 555 (0) [3040, 72150]
BAC/LE	19, 3537, 282 (3) [1296, 12796]	50, 14089, 430 (6) [2122, 34638]	81, 25690, 581 (3) [3010, 69910]
SCI/ST	15, 6708, 167 (0) [500, 9804]	29, 12655, 327 (0) [975, 18571]	50, 27507, 454 (0) [2107, 37984]
LOI/LE	19, 4276, 240 (3) [574, 7097]	40, 12352, 394 (8) [1040, 17831]	88, 23947, 427 (0) [1166, 31314]
kcol		6	
BAC/ST	14, 7260, 226 (3) [1548, 22650]	22, 13964, 375 (0) [3358, 43428]	27, 27773, 504 (0) [5280, 80144]
BAC/LE	14, 5801, 230 (2) [1316, 18636]	24, 12554, 391 (0) [2332, 38204]	34, 25636, 490 (1) [3558, 72622]
SCI/ST	16, 7505, 116 (0) [496, 10724]	20, 14881, 328 (0) [1684, 22580]	26, 29066, 425 (0) [2720, 41327]
LOI/LE	14, 5822, 179 (0) [542, 9003]	22, 19282, 389 (5) [1218, 18657]	33, 25331, 451 (0) [1522, 35250]
kcol		8	
BAC/ST	13, 12816, 183 (7) [2232, 30304]	16, 18277, 425 (1) [6078, 58248]	22, 32963, 530 (0) [8482, 97036]
BAC/LE	13, 8500, 200 (0) [1784, 26134]	18, 16415, 438 (3) [5388, 52938]	23, 31197, 531 (3) [7628, 91156]
SCI/ST	15, 9863, 136 (0) [792, 14213]	17, 17572, 301 (0) [2214, 26395]	17, 31947, 438 (0) [3700, 45985]
LOI/LE	15, 8196, 152 (5) [805, 12054]	15, 16060, 390 (1) [2560, 25747]	20, 29630, 518 (1) [3319, 43516]

Table 3: *Machine independent results for the Catalytic Surface Reaction Model. We consider kcol* $= 3, 4, 6, 8$ *and tol* $= 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol	3		
BAC/ST	14, 892, 68 (0) [306, 3108]	22, 1866, 96 (0) [448, 5638]	53, 4656, 91 (0) [542, 12398]
BAC/LE	15, 833, 86 (0) [378, 3180]	39, 1918, 93 (0) [406, 5546]	120, 3300, 91 (0) [412, 8244]
SCI/ST	— —	— —	— —
LOI/LE	18, 792, 88 (0) [190, 1560]	42, 1688, 121 (0) [259, 2679]	134, 3384, 414 (0) [852, 7088]
kcol	4		
BAC/ST	15, 865, 56 (0) [258, 2902]	15, 2068, 92 (0) [482, 6408]	30, 4620, 100 (0) [624, 12784]
BAC/LE	14, 892, 68 (0) [306, 3108]	22, 1866, 96 (0) [448, 5638]	53, 4656, 91 (0) [542, 12398]
SCI/ST	15, 868, 55 (0) [129, 1444]	19, 2230, 72 (0) [265, 3375]	34, 4777, 90 (0) [685, 6978]
LOI/LE	13, 883, 75 (0) [165, 1586]	22, 1838, 104 (0) [234, 2841]	48, 4087, 104 (0) [262, 5293]
kcol	6		
BAC/ST	13, 963, 47 (0) [234, 3094]	15, 2272, 76 (0) [444, 6886]	15, 5103, 120 (0) [1282, 15176]
BAC/LE	15, 899, 48 (0) [220, 2868]	15, 2256, 75 (0) [414, 6514]	20, 5021, 115 (0) [846, 14248]
SCI/ST	13, 940, 38 (0) [93, 1478]	15, 2398, 75 (0) [366, 3777]	18, 4639, 97 (0) [749, 7036]
LOI/LE	15, 887, 52 (0) [119, 1441]	15, 2184, 83 (0) [223, 3311]	21, 4732, 105 (0) [375, 6730]
kcol	8		
BAC/ST	15, 891, 34 (0) [160, 2804]	15, 2262, 49 (0) [382, 6456]	15, 5014, 80 (0) [1396, 14664]
BAC/LE	13, 985, 42 (0) [206, 3132]	15, 2223, 58 (0) [418, 6402]	14, 4760, 99 (0) [1422, 14260]
SCI/ST	11, 1124, 37 (0) [135, 1801]	14, 2558, 52 (0) [314, 3758]	15, 5298, 86 (0) [840, 7943]
LOI/LE	12, 964, 45 (0) [106, 1570]	15, 2267, 65 (0) [234, 3391]	15, 4922, 107 (0) [677, 7295]

Table 4: *Machine independent results for the Two Layer Burgers equation with* $\epsilon = 10^{-3}$. *We consider kcol* $= 3, 4, 6, 8$ *and tol* $= 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol	3		
BAC/ST	15, 871, 64 (0) [282, 2972]	22, 1846, 92 (0) [426, 5528]	50, 4509, 102 (0) [618, 12506]
BAC/LE	15, 834, 87 (0) [388, 3214]	39, 1918, 93 (0) [406, 5542]	120, 3300, 93 (0) [420, 8280]
SCI/ST	14, 882, 80 (0) [197, 1629]	— —	— —
LOI/LE	17, 756, 93 (0) [199, 1541]	45, 1561, 119 (0) [260, 2559]	136, 3384, 418 (0) [860, 7125]
kcol	4		
BAC/ST	13, 906, 59 (0) [276, 3096]	14, 2156, 97 (0) [490, 6634]	28, 4520, 98 (0) [644, 12754]
BAC/LE	15, 871, 64 (0) [282, 2972]	22, 1846, 92 (0) [426, 5528]	50, 4509, 102 (0) [618, 12506]
SCI/ST	15, 870, 54 (1) [129, 1445]	18, 2168, 78 (0) [279, 3347]	35, 4621, 82 (0) [625, 6622]
LOI/LE	15, 854, 74 (1) [166, 1538]	24, 1793, 94 (0) [213, 2709]	51, 4125, 112 (0) [272, 5430]
kcol	6		
BAC/ST	13, 951, 46 (0) [228, 3114]	15, 2183, 70 (0) [436, 6490]	15, 4849, 113 (0) [1272, 14588]
BAC/LE	15, 884, 51 (0) [232, 2890]	15, 2159, 72 (0) [420, 6352]	20, 4803, 112 (0) [872, 13888]
SCI/ST	13, 945, 36 (0) [85, 1434]	11, 2329, 65 (0) [295, 3522]	17, 4882, 116 (0) [881, 7758]
LOI/LE	15, 884, 54 (0) [125, 1448]	15, 2215, 80 (0) [227, 3349]	22, 4816, 120 (0) [410, 6862]
kcol	8		
BAC/ST	15, 917, 37 (0) [188, 2788]	15, 2229, 51 (0) [418, 6310]	14, 4940, 86 (0) [1520, 14766]
BAC/LE	14, 952, 42 (0) [214, 3028]	14, 2443, 64 (0) [524, 7160]	14, 4894, 100 (0) [1404, 14630]
SCI/ST	11, 1141, 33 (0) [114, 1779]	14, 2496, 51 (0) [325, 3749]	14, 5468, 93 (0) [901, 8290]
LOI/LE	11, 1012, 44 (0) [120, 1650]	15, 2177, 64 (1) [219, 3185]	15, 4879, 109 (1) [647, 7248]

Table 5: *Machine independent results for the Two Layer Burgers equation*×6 *with* $\epsilon = 10^{-3}$. *We consider kcol* $= 3, 4, 6, 8$ *and tol* $= 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol	3		
BAC/ST	14, 855, 63 (0) [288, 2940]	20, 1883, 101 (0) [474, 5842]	46, 4438, 93 (0) [550, 11870]
BAC/LE	15, 833, 87 (0) [382, 3198]	39, 1918, 93 (0) [406, 5544]	125, 3365, 91 (0) [410, 8344]
SCI/ST	— —	— —	— —
LOI/LE	19, 760, 95 (0) [203, 1581]	44, 1738, 122 (0) [265, 2766]	136, 3384, 411 (0) [846, 7062]
kcol	4		
BAC/ST	15, 882, 54 (0) [246, 2932]	14, 2137, 95 (0) [482, 6520]	28, 4781, 103 (0) [642, 13404]
BAC/LE	14, 855, 63 (0) [288, 2940]	20, 1883, 101 (0) [474, 5842]	46, 4438, 93 (0) [550, 11870]
SCI/ST	14, 901, 51 (0) [126, 1481]	20, 2215, 73 (0) [247, 3336]	31, 4869, 90 (0) [695, 7125]
LOI/LE	15, 844, 72 (0) [158, 1504]	24, 1841, 99 (0) [228, 2817]	52, 4105, 106 (0) [267, 5385]
kcol	6		
BAC/ST	15, 934, 46 (0) [224, 3020]	15, 2272, 69 (0) [434, 6752]	15, 4905, 114 (0) [1272, 14746]
BAC/LE	14, 884, 49 (0) [226, 2858]	14, 2149, 77 (0) [476, 6548]	20, 4844, 113 (0) [864, 13938]
SCI/ST	13, 926, 38 (0) [92, 1460]	14, 2357, 73 (0) [382, 3748]	18, 4731, 105 (0) [803, 7288]
LOI/LE	15, 825, 51 (0) [117, 1373]	15, 2207, 78 (0) [219, 3290]	20, 4906, 110 (0) [411, 6957]
kcol	8		
BAC/ST	15, 891, 36 (0) [176, 2832]	14, 2265, 54 (0) [488, 6696]	14, 4954, 88 (0) [1572, 14882]
BAC/LE	13, 1008, 42 (0) [198, 3210]	15, 2164, 56 (0) [378, 6296]	14, 5036, 101 (0) [1454, 15042]
SCI/ST	11, 1123, 35 (0) [115, 1784]	15, 2409, 50 (0) [233, 3472]	15, 5285, 78 (0) [777, 7747]
LOI/LE	12, 1000, 46 (0) [129, 1654]	14, 2351, 66 (0) [247, 3429]	14, 5037, 102 (0) [721, 7533]

Table 6: *Machine independent results for the Two Layer Burgers equation*$\times 12$ *with* $\epsilon = 10^{-3}$. *We consider* $kcol = 3, 4, 6, 8$ *and* $tol = 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol		3	
BAC/ST	13, 9178, 602 (3) [2848, 32226]	22, 21422, 904 (0) [4116, 65290]	43, 45483, 947 (0) [6754, 127594]
BAC/LE	18, 8841, 809 (2) [3478, 33640]	36, 19283, 1017 (0) [4384, 60834]	84, 44775, 1063 (0) [5700, 122680]
SCI/ST	— —	27, 29909, 750 (3) [3140, 35920]	51, 47229, 801 (0) [6354, 68648]
LOI/LE	19, 7978, 969 (0) [1984, 16370]	40, 16843, 1202 (0) [2459, 27173]	105, 38520, 3361 (2) [6767, 68697]
kcol		4	
BAC/ST	15, 9426, 521 (4) [2510, 32096]	17, 23743, 954 (4) [6638, 76484]	28, 46574, 1039 (3) [8866, 134230]
BAC/LE	13, 9178, 602 (3) [2848, 32226]	22, 21422, 904 (0) [4116, 65290]	43, 45483, 947 (0) [6754, 127594]
SCI/ST	15, 9678, 489 (0) [1348, 16543]	— —	33, 48678, 781 (0) [6607, 70499]
LOI/LE	15, 9090, 775 (2) [1715, 17141]	22, 20116, 954 (0) [2125, 30923]	43, 46580, 1090 (1) [3104, 65542]
kcol		6	
BAC/ST	15, 10782, 432 (11) [2418, 34582]	14, 26090, 796 (0) [8498, 84682]	16, 79020, 1159 (4) [18122, 163304]
BAC/LE	15, 13009, 481 (5) [2344, 32586]	15, 25207, 943 (0) [8502, 84048]	20, 47812, 1045 (2) [12044, 141438]
SCI/ST	14, 10938, 402 (0) [1250, 17857]	14, 26095, 718 (0) [4561, 41787]	19, 54477, 905 (3) [8332, 78009]
LOI/LE	15, 9792, 547 (2) [1321, 16744]	15, 24572, 971 (7) [4211, 40997]	22, 48019, 1079 (5) [5474, 70210]
kcol		8	
BAC/ST	14, 12410, 316 (0) [2716, 38194]	15, 27032, 608 (3) [8148, 83486]	15, 52275, 874 (0) [16988, 158874]
BAC/LE	13, 12995, 329 (7) [3016, 39850]	15, 26849, 672 (0) [8356, 84464]	15, 78789, 1166 (8) [20350, 175078]
SCI/ST	14, 12513, 362 (0) [1635, 20163]	14, 29192, 615 (0) [4728, 45133]	15, 81994, 975 (2) [9682, 84781]
LOI/LE	14, 12062, 424 (1) [1652, 19852]	15, 42373, 743 (23) [4460, 42732]	16, 63511, 1231 (13) [10352, 89036]

Table 7: *Machine independent results for the Two Layer Burgers equation with* $\epsilon = 10^{-4}$. *We consider kcol* $= 3, 4, 6, 8$ *and tol* $= 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol		3	
BAC/ST	15, 9365, 634 (5) [3048, 33282]	23, 21616, 914 (3) [4188, 65854]	46, 48641, 937 (0) [6480, 138236]
BAC/LE	17, 8884, 796 (2) [3470, 33842]	36, 19203, 982 (0) [4214, 59722]	101, 44323, 991 (0) [4912, 117634]
SCI/ST	14, 9319, 660 (1) [1651, 17262]	— —	47, 47607, 827 (2) [6724, 69671]
LOI/LE	21, 8257, 940 (0) [1928, 16450]	38, 16466, 1179 (0) [2424, 26883]	103, 38196, 3247 (0) [6526, 67435]
kcol		4	
BAC/ST	15, 9541, 541 (4) [2576, 32460]	16, 23484, 986 (6) [6852, 76320]	28, 46528, 1012 (3) [8860, 133872]
BAC/LE	15, 9365, 634 (5) [3048, 33282]	23, 21616, 914 (3) [4188, 65854]	46, 48641, 937 (0) [6480, 138236]
SCI/ST	14, 9743, 510 (1) [1406, 16804]	19, 23215, 752 (0) [3644, 37581]	30, 48715, 757 (0) [6612, 70440]
LOI/LE	15, 9132, 754 (6) [1720, 16958]	24, 19792, 937 (1) [2114, 30650]	48, 43528, 1016 (1) [2793, 60359]
kcol		6	
BAC/ST	15, 10969, 440 (6) [2812, 36024]	15, 25450, 757 (3) [7806, 81182]	18, 52028, 1135 (3) [18394, 164570]
BAC/LE	14, 10341, 471 (5) [2346, 34086]	15, 24356, 836 (3) [7422, 78698]	21, 48205, 1080 (3) [12766, 143966]
SCI/ST	15, 12749, 439 (2) [1220, 17230]	14, 26075, 704 (0) [4527, 41851]	18, 84396, 955 (6) [8827, 79550]
LOI/LE	15, 10305, 537 (3) [1435, 17606]	15, 24430, 962 (6) [4187, 40909]	21, 47887, 1072 (9) [5420, 69248]
kcol		8	
BAC/ST	14, 16712, 329 (3) [2976, 40288]	14, 28390, 628 (3) [9058, 88188]	14, 64637, 927 (4) [18240, 164180]
BAC/LE	15, 11871, 377 (8) [2800, 37192]	15, 26501, 663 (0) [8230, 83514]	15, 55285, 1127 (4) [19986, 174726]
SCI/ST	15, 11478, 384 (0) [1528, 18778]	15, 28076, 586 (0) [4380, 43172]	16, 55252, 946 (1) [9467, 83336]
LOI/LE	14, 11704, 422 (0) [1541, 19211]	13, 33795, 777 (10) [4843, 44302]	15, 95550, 1276 (17) [10630, 89062]

Table 8: *Machine independent results for the Two Layer Burgers equation*×6 *with* $\epsilon = 10^{-4}$. *We consider* $kcol = 3, 4, 6, 8$ *and* $tol = 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

tol	10^{-4}	10^{-6}	10^{-8}
kcol		3	
BAC/ST	13, 9312, 642 (4) [3004, 33194]	22, 21352, 898 (3) [4060, 64710]	45, 49204, 898 (0) [6030, 136218]
BAC/LE	17, 8820, 805 (2) [3462, 33634]	33, 20529, 1010 (0) [4348, 63970]	89, 44747, 1003 (0) [5564, 123658]
SCI/ST	15, 9306, 663 (3) [1683, 17278]	22, 22881, 752 (0) [3306, 36669]	49, 47962, 871 (2) [6945, 70867]
LOI/LE	20, 8195, 980 (0) [2011, 16689]	42, 16162, 1201 (0) [2450, 26339]	101, 38904, 3247 (0) [6529, 68168]
kcol		4	
BAC/ST	15, 9705, 527 (7) [2572, 32586]	17, 23482, 974 (4) [7100, 76832]	27, 47309, 1032 (3) [8764, 136006]
BAC/LE	13, 9312, 642 (4) [3004, 33194]	22, 21352, 898 (3) [4060, 64710]	45, 49204, 898 (0) [6030, 136218]
SCI/ST	15, 9663, 473 (3) [1271, 16324]	— —	33, 49073, 793 (0) [6717, 71234]
LOI/LE	13, 9714, 804 (9) [1839, 17430]	25, 23038, 930 (3) [2098, 30532]	46, 45034, 1043 (2) [2992, 63230]
kcol		6	
BAC/ST	13, 11694, 378 (4) [2846, 37344]	15, 29451, 789 (2) [8710, 85112]	17, 82610, 1182 (5) [18554, 163600]
BAC/LE	15, 9710, 481 (3) [2242, 32186]	14, 25059, 898 (3) [8102, 82074]	21, 48503, 1069 (3) [12646, 144238]
SCI/ST	15, 10282, 440 (0) [1180, 16984]	15, 26471, 730 (0) [4683, 42506]	20, 51961, 930 (0) [8750, 79625]
LOI/LE	13, 10469, 554 (3) [1365, 17280]	15, 24484, 1016 (8) [4398, 41349]	— —
kcol		8	
BAC/ST	15, 16449, 354 (3) [2654, 36104]	15, 40531, 617 (2) [8424, 85848]	14, 54871, 950 (0) [18726, 167840]
BAC/LE	15, 13997, 375 (8) [2500, 35796]	15, 26589, 692 (0) [8464, 84318]	14, 61185, 1139 (4) [19914, 173086]
SCI/ST	14, 12371, 351 (1) [1570, 19749]	14, 28385, 590 (0) [4455, 43840]	14, 54571, 949 (0) [9457, 82689]
LOI/LE	14, 11555, 433 (0) [1563, 19209]	15, 26753, 728 (11) [4351, 42578]	14, 67267, 1249 (13) [10343, 88006]

Table 9: *Machine independent results for the Two Layer Burgers equation*×12 *with* $\epsilon = 10^{-4}$. *We consider kcol* $= 3, 4, 6, 8$ *and tol* $= 10^{-4}, 10^{-6}, 10^{-8}$. *Table entries are of the form Final Nint, Accepted Time Steps, Remeshings (Cold Starts) [Calls to CRDCMP, CRSLVE].*

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.08	0.09	0.12	0.12	0.15	0.18	0.22	0.27
BAC/LE	0.07	0.08	0.09	0.12	0.12	0.15	0.18	0.22
SCI/ST	0.04	0.05	0.06	0.07	0.08	0.10	0.11	0.14
LOI/LE	0.05	0.05	0.06	0.06	0.08	0.09	0.11	0.14
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.22	0.25	0.27	0.30	0.36	0.42	0.50	0.55
BAC/LE	0.26	0.22	0.25	0.27	0.30	0.36	0.42	0.50
SCI/ST	—	0.15	0.17	0.17	0.19	0.23	0.27	0.31
LOI/LE	0.19	0.15	0.17	0.17	0.19	0.22	0.25	0.29
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	1.05	0.87	0.76	0.79	0.77	0.83	0.93	1.07
BAC/LE	1.35	1.05	0.87	0.76	0.79	0.77	0.83	0.93
SCI/ST	0.64	0.51	0.44	0.46	0.50	0.53	0.54	0.60
LOI/LE	1.43	0.60	0.53	0.52	0.51	0.49	0.52	0.57

Table 10: *Machine dependent timings (in seconds), One Layer Burgers' equation,* $\epsilon = 10^{-3}$*,* $kcol = 3, \ldots, 10$*,* $tol = 10^{-4}, 10^{-6}, 10^{-8}$*.*

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.84	0.96	1.21	1.43	1.82	2.24	2.56	3.27
BAC/LE	0.69	0.84	0.96	1.21	1.43	1.82	2.24	2.56
SCI/ST	0.48	0.58	0.65	0.81	0.96	1.21	1.46	1.90
LOI/LE	0.55	0.58	0.66	0.80	0.92	1.14	1.41	1.67
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	2.82	2.95	3.27	3.34	4.02	4.65	5.62	6.84
BAC/LE	2.61	2.82	2.95	3.27	3.34	4.02	4.65	5.62
SCI/ST	—	—	1.89	2.23	2.30	2.78	3.21	3.80
LOI/LE	1.79	1.70	1.94	2.21	2.28	2.52	2.92	3.46
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	10.07	7.69	8.26	9.18	10.25	9.71	10.59	12.10
BAC/LE	13.49	10.07	7.69	8.26	9.18	10.25	9.71	10.59
SCI/ST	—	—	4.72	5.07	5.48	6.36	6.60	9.07
LOI/LE	10.31	6.40	5.14	5.10	5.87	6.65	6.16	6.66

Table 11: *Machine dependent timings (in seconds), One Layer Burgers' equation,* $\epsilon = 10^{-4}$*,* $kcol = 3, \ldots, 10$*,* $tol = 10^{-4}, 10^{-6}, 10^{-8}$*.*

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.03	0.04	0.05	0.06	0.07	0.09	0.10	0.11
BAC/LE	0.02	0.03	0.04	0.05	0.06	0.07	0.09	0.10
SCI/ST	0.01	0.02	0.02	0.02	0.03	0.04	0.05	0.07
LOI/LE	0.02	0.02	0.02	0.04	0.03	0.05	0.05	0.06
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.12	0.14	0.17	0.21	0.24	0.30	0.39	0.42
BAC/LE	0.14	0.12	0.14	0.17	0.21	0.24	0.30	0.39
SCI/ST	0.05	0.06	—	0.08	0.11	0.13	0.17	—
LOI/LE	0.09	0.10	0.09	0.09	0.11	0.13	0.14	0.19
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.50	0.46	0.54	0.53	0.65	0.77	—	0.97
BAC/LE	0.80	0.50	0.46	0.54	0.53	0.65	0.77	—
SCI/ST	0.26	0.23	0.26	0.27	0.30	0.37	0.45	0.50
LOI/LE	0.59	0.38	0.30	0.29	0.30	0.39	0.38	—

Table 12: *Machine dependent timings (in seconds), Catalytic Surface Reaction Model,* $kcol = 3, \ldots, 10$*,* $tol = 10^{-4}, 10^{-6}, 10^{-8}$*.*

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.05	0.06	0.08	0.09	0.11	0.14	0.16	0.18
BAC/LE	0.05	0.05	0.06	0.08	0.09	0.11	0.14	0.16
SCI/ST	—	0.03	0.04	0.04	0.05	0.06	0.08	0.10
LOI/LE	0.03	0.03	0.04	0.05	0.05	0.06	0.07	0.11
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.15	0.16	0.18	0.21	0.23	0.27	0.32	0.36
BAC/LE	0.20	0.15	0.16	0.18	0.21	0.23	0.27	0.32
SCI/ST	—	0.10	0.11	0.13	0.14	0.16	0.18	0.20
LOI/LE	0.15	0.10	0.11	0.12	0.13	0.16	0.17	0.18
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.72	0.57	0.53	0.55	0.55	0.63	0.68	0.80
BAC/LE	0.89	0.72	0.57	0.53	0.55	0.55	0.63	0.68
SCI/ST	—	0.35	0.30	0.32	0.34	0.38	0.39	0.44
LOI/LE	1.16	0.44	0.36	0.33	0.35	0.37	0.38	0.45

Table 13: *Machine dependent timings (in seconds), Two Layer Burgers' equation,* $\epsilon = 10^{-3}$*,* $kcol = 3, \ldots, 10$*,* $tol = 10^{-4}, 10^{-6}, 10^{-8}$*.*

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.33	0.40	0.57	0.66	0.86	1.12	1.31	1.53
BAC/LE	0.26	0.33	0.40	0.57	0.66	0.86	1.12	1.31
SCI/ST	0.17	0.20	0.23	0.28	0.36	0.45	0.64	0.72
LOI/LE	0.19	0.22	0.26	0.34	0.35	0.43	0.52	0.77
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.96	1.11	1.28	1.60	2.08	2.36	2.91	3.68
BAC/LE	1.15	0.96	1.11	1.28	1.60	2.08	2.36	2.91
SCI/ST	—	0.63	0.78	0.88	1.07	1.31	1.53	1.84
LOI/LE	0.81	0.63	0.69	0.79	0.93	1.13	1.36	1.59
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	4.71	4.02	4.08	4.66	5.44	6.41	7.50	8.31
BAC/LE	5.34	4.71	4.02	4.08	4.66	5.44	6.41	7.50
SCI/ST	—	2.41	2.34	2.89	3.06	3.71	3.98	4.62
LOI/LE	7.03	2.82	2.45	2.51	2.81	3.21	3.88	4.42

Table 14: *Machine dependent timings (in seconds), Two Layer Burgers' equation*$\times 6$*,* $\epsilon = 10^{-3}$*,* $kcol = 3, \ldots, 10$*,* $tol = 10^{-4}, 10^{-6}, 10^{-8}$*.*

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	1.09	1.51	1.95	2.74	3.18	4.43	5.46	6.81
BAC/LE	0.89	1.09	1.51	1.95	2.74	3.18	4.43	5.46
SCI/ST	—	0.61	0.81	1.01	1.29	1.72	2.45	2.98
LOI/LE	0.67	0.72	0.95	1.28	1.37	1.91	2.47	3.35
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	3.31	3.83	4.61	5.84	6.81	9.83	12.15	15.66
BAC/LE	3.58	3.31	3.83	4.61	5.84	6.81	9.83	12.15
SCI/ST	—	1.99	2.63	3.57	4.18	4.64	6.97	8.43
LOI/LE	2.70	2.21	2.38	2.84	3.39	4.56	5.21	6.91
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	14.00	12.73	13.71	17.50	22.92	29.10	35.48	37.28
BAC/LE	17.12	14.00	12.73	13.71	17.50	22.92	29.10	35.48
SCI/ST	—	8.42	8.17	10.62	12.43	14.38	16.19	20.89
LOI/LE	24.24	8.25	7.45	8.41	10.84	13.30	15.84	18.99

Table 15: *Machine dependent timings (in seconds), Two Layer Burgers' equation*$\times 12$, $\epsilon = 10^{-3}$, $kcol = 3, \ldots, 10$, $tol = 10^{-4}, 10^{-6}, 10^{-8}$.

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	0.58	0.72	0.90	1.11	1.35	1.56	1.98	2.35
BAC/LE	0.51	0.58	0.72	0.90	1.11	1.35	1.56	1.98
SCI/ST	—	0.38	0.45	0.57	0.71	0.87	1.05	1.21
LOI/LE	0.40	0.42	0.48	0.57	0.73	0.89	1.12	1.22
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	1.77	2.03	2.43	2.79	3.27	3.78	4.70	5.69
BAC/LE	1.91	1.77	2.03	2.43	2.79	3.27	3.78	4.70
SCI/ST	1.18	—	1.33	1.53	1.82	2.17	2.55	3.01
LOI/LE	1.33	1.17	1.35	1.54	1.84	2.10	2.50	2.87
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	6.66	5.89	5.89	6.65	7.53	7.75	8.90	10.46
BAC/LE	9.81	6.66	5.89	5.89	6.65	7.53	7.75	8.90
SCI/ST	4.26	3.63	3.51	3.81	4.16	4.68	5.13	5.64
LOI/LE	9.61	4.47	3.96	3.82	4.34	5.12	5.14	5.68

Table 16: *Machine dependent timings (in seconds), Two Layer Burgers' equation*, $\epsilon = 10^{-4}$, $kcol = 3, \ldots, 10$, $tol = 10^{-4}, 10^{-6}, 10^{-8}$.

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	3.68	4.84	6.41	9.23	12.01	14.56	18.21	22.08
BAC/LE	2.82	3.68	4.84	6.41	9.23	12.01	14.56	18.21
SCI/ST	1.87	2.34	2.87	3.82	5.33	6.69	7.91	10.26
LOI/LE	2.14	2.52	—	4.12	5.47	6.60	8.86	9.81
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	11.20	14.92	18.51	23.83	30.94	39.78	48.22	61.64
BAC/LE	10.82	11.20	14.92	18.51	23.83	30.94	39.78	48.22
SCI/ST	—	7.87	9.71	12.29	15.52	19.33	23.82	29.92
LOI/LE	7.11	7.20	9.14	11.77	14.61	19.31	24.02	27.71
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	48.25	43.71	50.59	65.73	81.97	87.70	103.05	120.42
BAC/LE	58.79	48.25	43.71	50.59	65.73	81.97	87.70	103.05
SCI/ST	25.79	25.49	27.44	33.45	38.49	45.53	53.05	60.16
LOI/LE	54.96	28.09	28.14	29.27	39.64	47.81	54.39	63.63

Table 17: *Machine dependent timings (in seconds), Two Layer Burgers' equation*$\times 6$, $\epsilon = 10^{-4}$, $kcol = 3, \ldots, 10$, $tol = 10^{-4}, 10^{-6}, 10^{-8}$.

$tol = 10^{-4}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	12.51	16.89	22.05	32.94	43.22	56.11	78.78	97.31
BAC/LE	9.11	12.51	16.89	22.05	32.94	43.22	56.11	78.78
SCI/ST	5.79	7.18	9.80	13.55	19.22	26.79	32.78	40.17
LOI/LE	7.27	8.77	11.25	14.54	21.08	26.59	36.28	39.95
$tol = 10^{-6}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	35.82	52.98	72.42	99.14	128.22	169.66	226.70	274.55
BAC/LE	33.56	35.82	52.98	72.42	99.14	128.22	169.66	226.70
SCI/ST	21.56	—	34.86	48.95	61.62	81.14	105.12	134.48
LOI/LE	23.92	23.17	31.90	47.00	58.21	77.42	106.24	131.37
$tol = 10^{-8}/kcol =$	3	4	5	6	7	8	9	10
BAC/ST	146.34	145.06	182.29	268.81	337.51	405.95	494.13	593.82
BAC/LE	165.03	146.34	145.06	182.29	268.81	337.51	405.95	494.13
SCI/ST	83.51	86.47	97.39	128.54	157.00	189.48	233.16	284.21
LOI/LE	171.56	85.32	88.62	—	155.66	203.16	238.82	294.21

Table 18: *Machine dependent timings (in seconds), Two Layer Burgers' equation*$\times 12$, $\epsilon = 10^{-4}$, $kcol = 3, \ldots, 10$, $tol = 10^{-4}, 10^{-6}, 10^{-8}$.

	Timing Ratio Averages for Nine Test Problems			
Problem	$\frac{\text{BAC/LE}}{\text{BAC/ST}}$	$\frac{\text{SCI/ST}}{\text{BAC/ST}}$	$\frac{\text{LOI/LE}}{\text{BAC/ST}}$	$\frac{\text{LOI/LE}}{\text{SCI/ST}}$
OLBE, $\epsilon = 10^{-3}$	0.93	0.57	0.62	1.06
OLBE, $\epsilon = 10^{-4}$	0.91	0.59	0.60	0.98
TLBE, $\epsilon = 10^{-3}$	0.94	0.56	0.64	1.02
TLBE, $\epsilon = 10^{-4}$	0.90	0.56	0.62	1.10
CSRM	0.90	0.46	0.57	1.29
TLBE×6, $\epsilon = 10^{-3}$	0.88	0.52	0.56	0.98
TLBE×6, $\epsilon = 10^{-4}$	0.84	0.49	0.53	1.06
TLBE×12, $\epsilon = 10^{-3}$	0.84	0.51	0.56	0.97
TLBE×12, $\epsilon = 10^{-4}$	0.79	0.48	0.52	1.08

Table 19: *Averages of ratios of timings for each problem (see Section 5.1) over* $kcol = 3, \ldots, 10$ *and* $tol = 10^{-4}, 10^{-6}, 10^{-8}$. *Overall averages over all nine problems are* $\frac{BAC/LE}{BAC/ST} = 0.88$, $\frac{SCI/ST}{BAC/ST} = 0.53$, $\frac{LOI/LE}{BAC/ST} = 0.58$, *and* $\frac{LOI/LE}{SCI/ST} = 1.06$

	Timing Ratio Averages for $kcol = 3, \ldots, 10$			
kcol	$\frac{\text{BAC/LE}}{\text{BAC/ST}}$	$\frac{\text{SCI/ST}}{\text{BAC/ST}}$	$\frac{\text{LOI/LE}}{\text{BAC/ST}}$	$\frac{\text{LOI/LE}}{\text{SCI/ST}}$
3	1.07	0.53	0.92	1.66
4	0.94	0.54	0.61	1.15
5	0.86	0.53	0.55	1.05
6	0.84	0.52	0.53	1.03
7	0.84	0.51	0.51	0.99
8	0.84	0.52	0.51	0.99
9	0.83	0.52	0.50	0.95
10	0.85	0.53	0.50	0.96

Table 20: *Averages of ratios of timings for* $kcol = 3, \ldots, 10$ *over all test problems (see Section 5.1) and* $tol = 10^{-4}, 10^{-6}, 10^{-8}$. *Overall averages over all eight kcol values are* $\frac{BAC/LE}{BAC/ST} = 0.88$, $\frac{SCI/ST}{BAC/ST} = 0.53$, $\frac{LOI/LE}{BAC/ST} = 0.58$, *and* $\frac{LOI/LE}{SCI/ST} = 1.06$

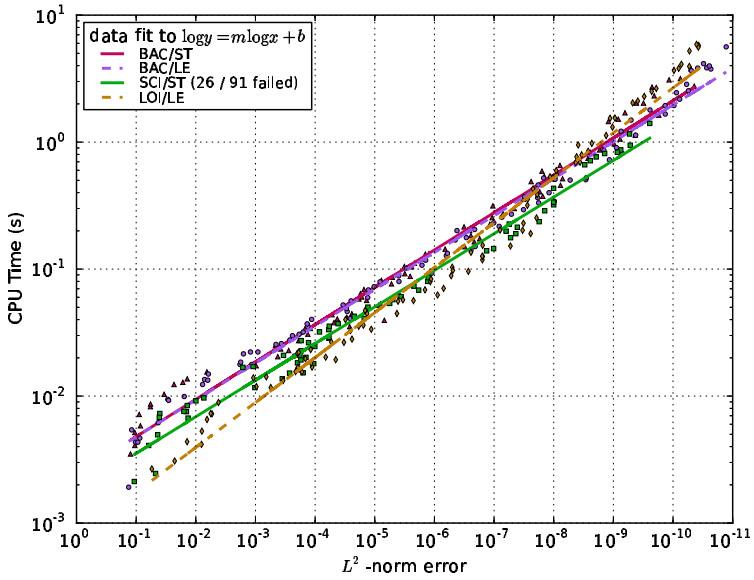

Figure 7: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 3$

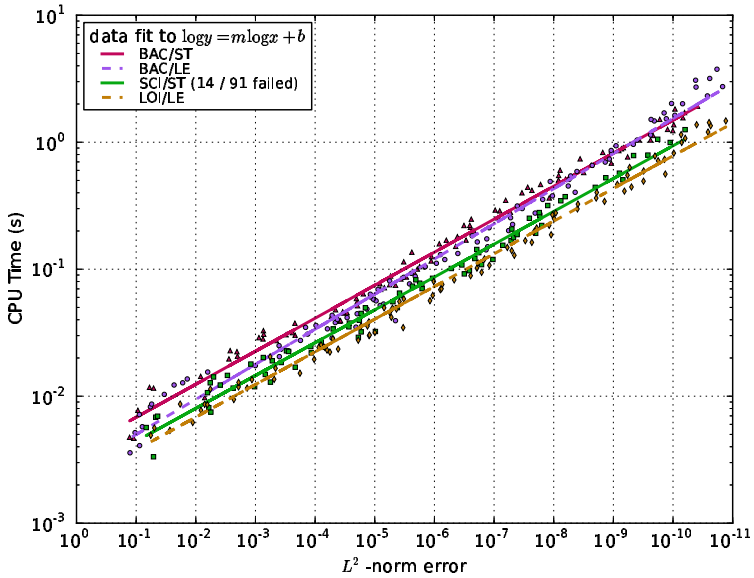

Figure 8: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 4$

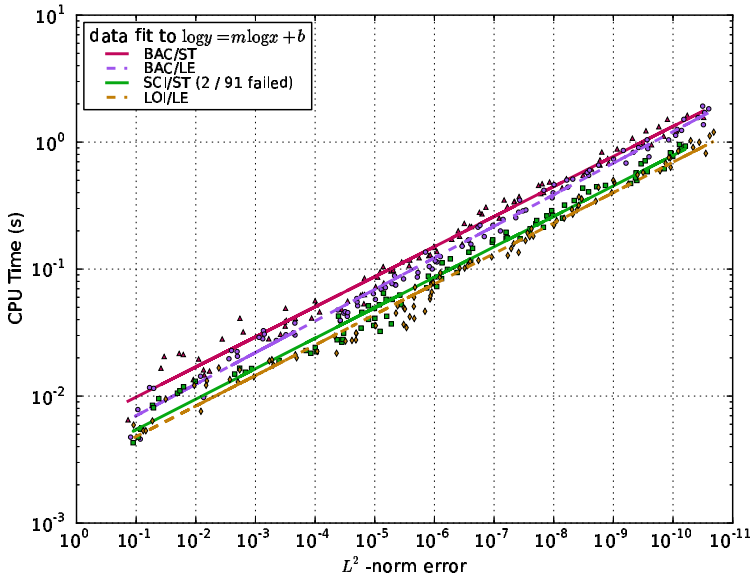

Figure 9: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 5$

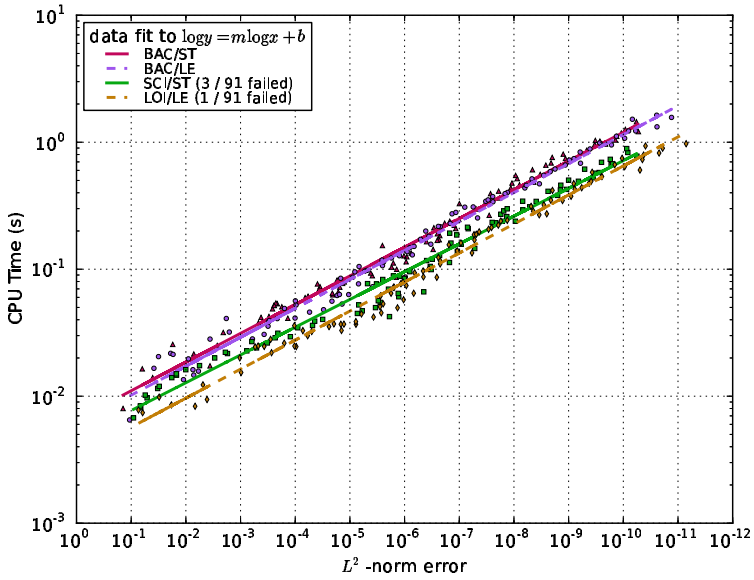

Figure 10: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 6$

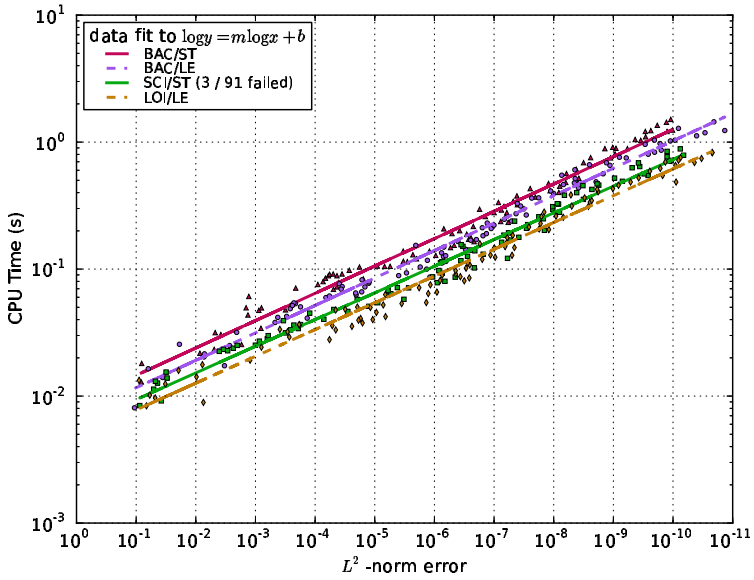

Figure 11: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 7$

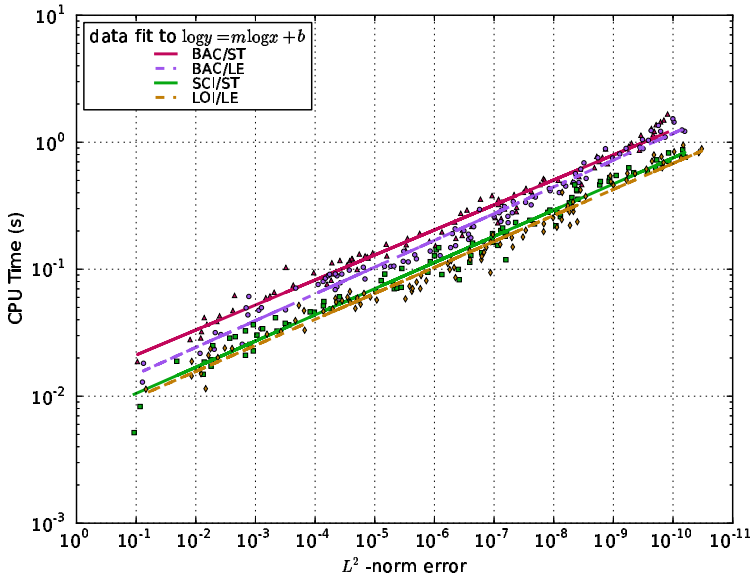

Figure 12: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 8$

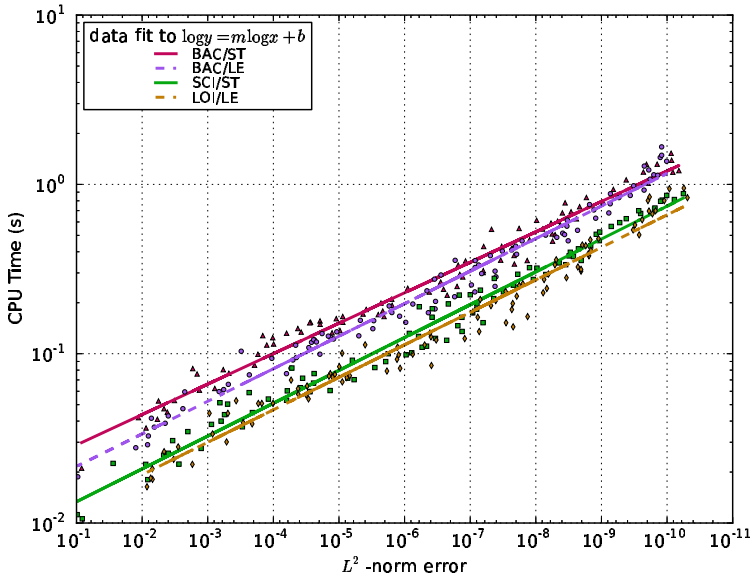

Figure 13: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 9$

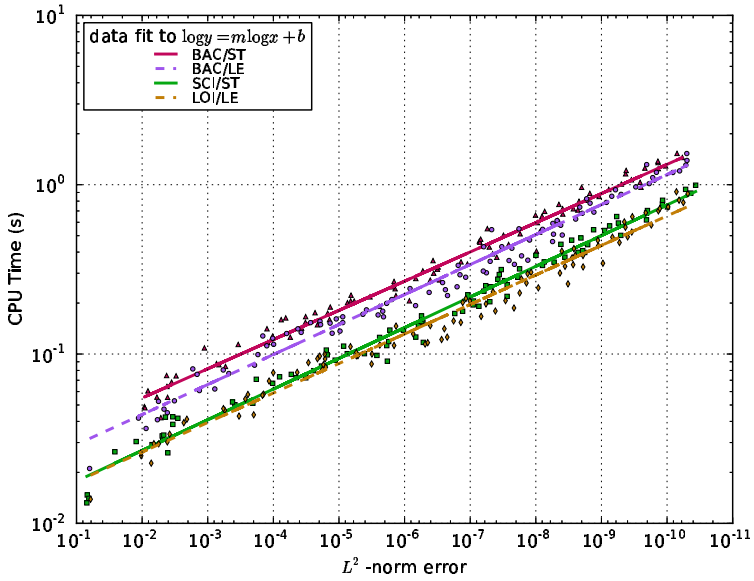

Figure 14: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 10$

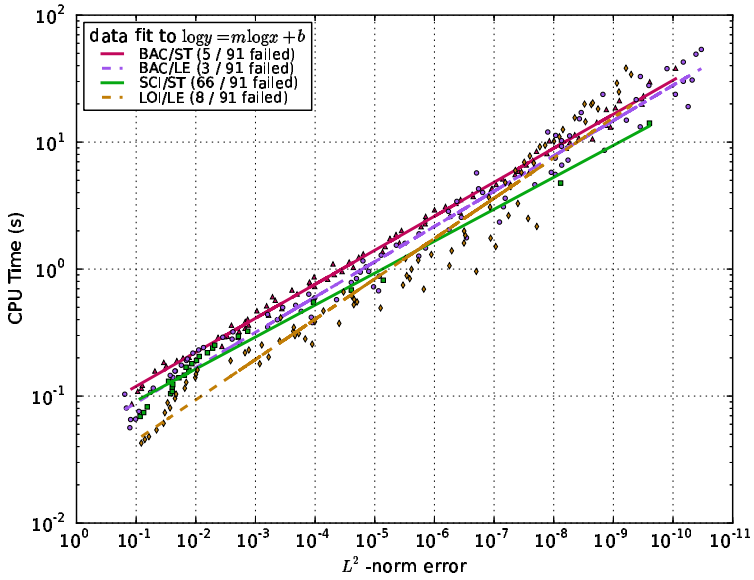

Figure 15: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 3$

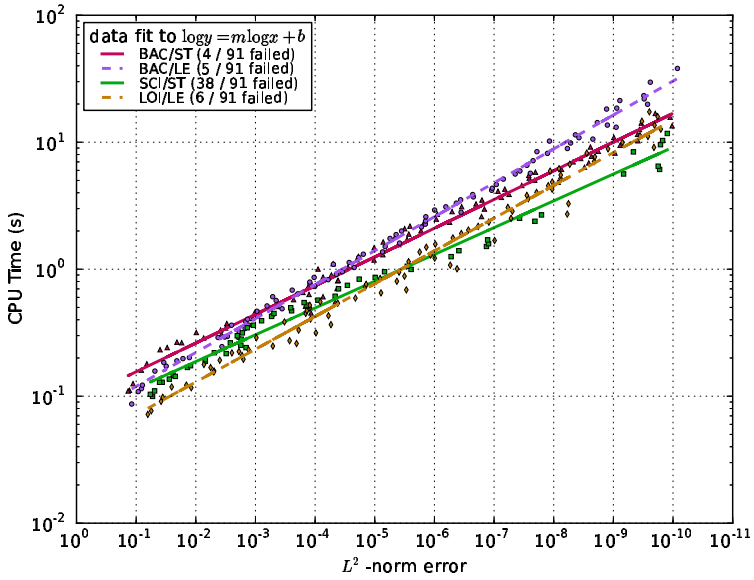

Figure 16: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 4$

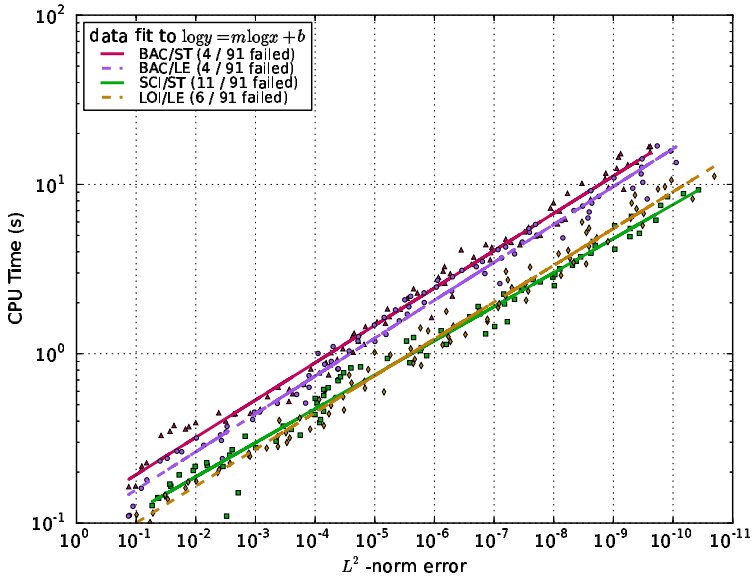

Figure 17: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 5$

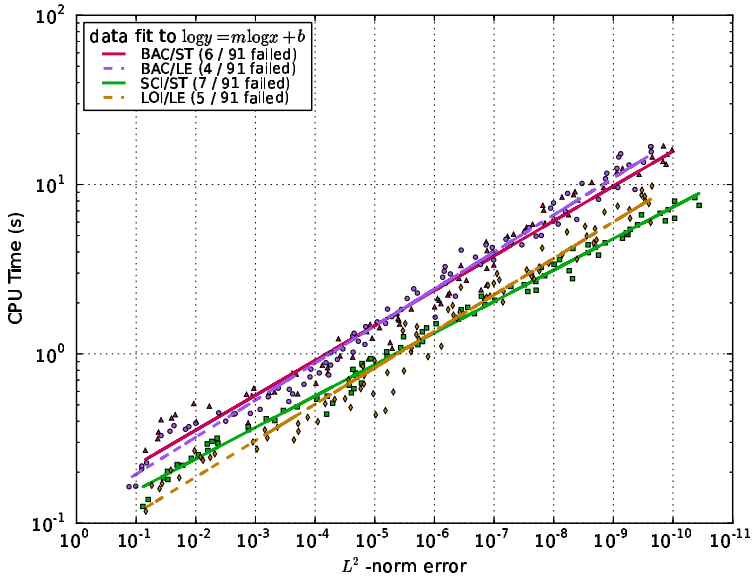

Figure 18: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 6$

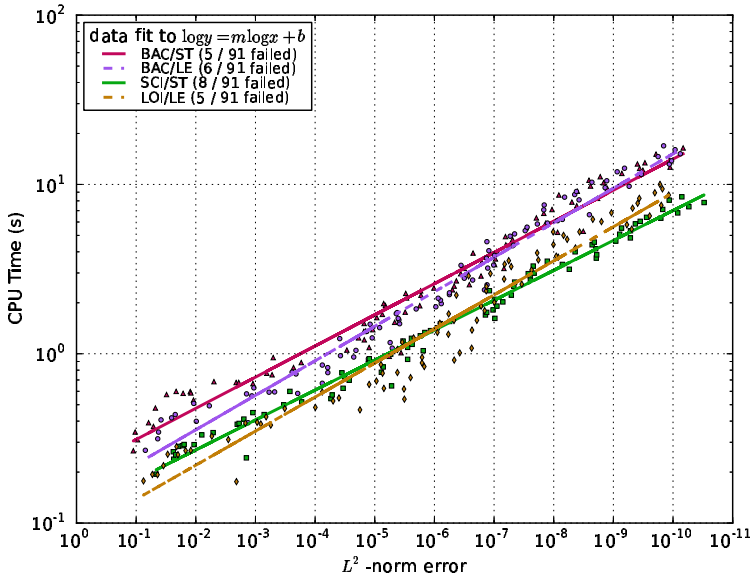

Figure 19: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 7$

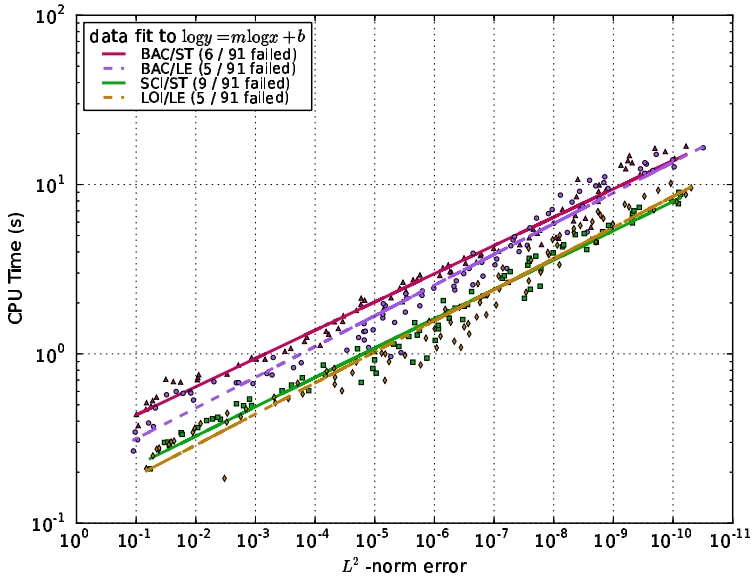

Figure 20: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 8$

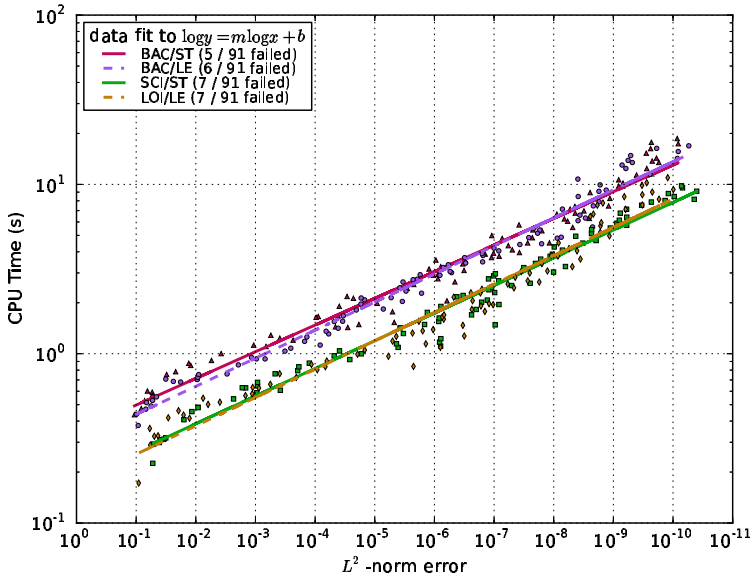

Figure 21: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 9$

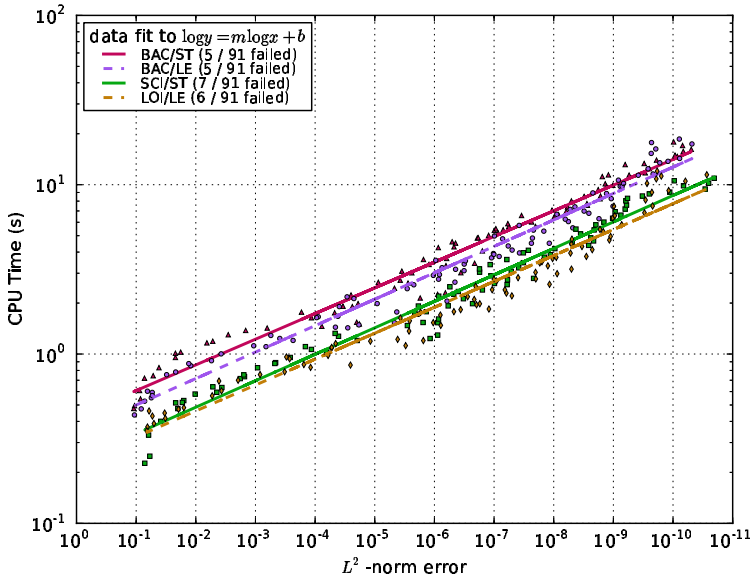

Figure 22: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 10$

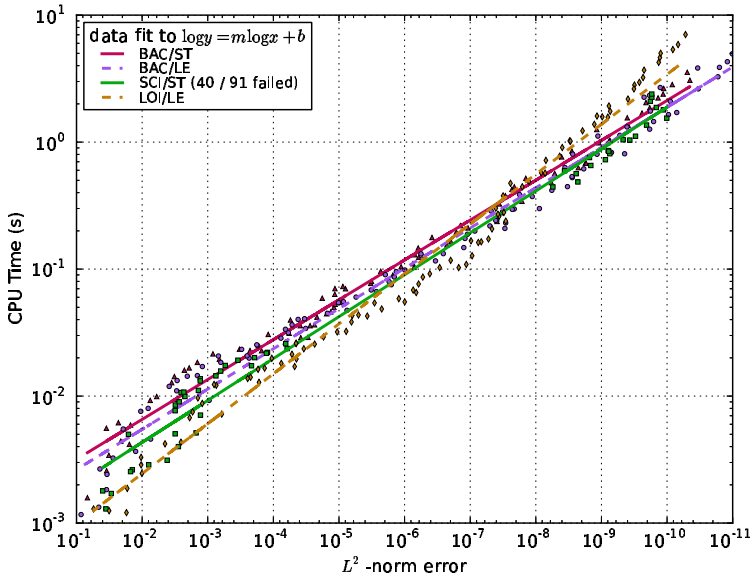

Figure 23: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 3$

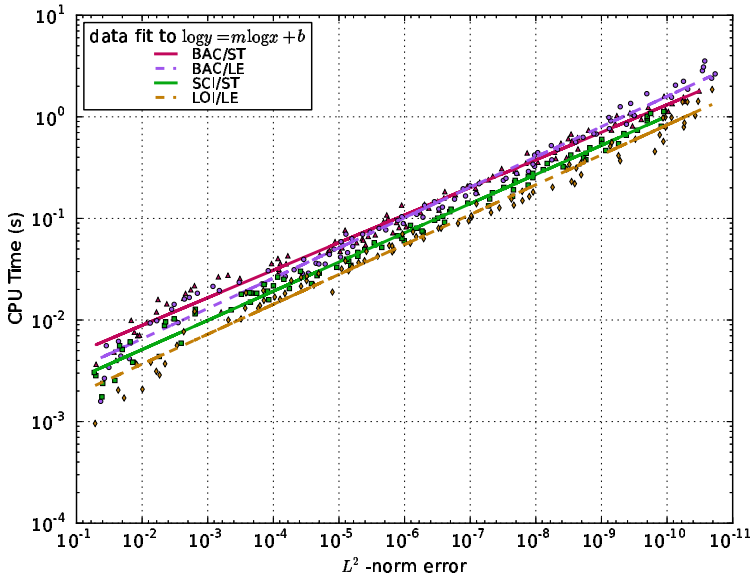

Figure 24: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 4$

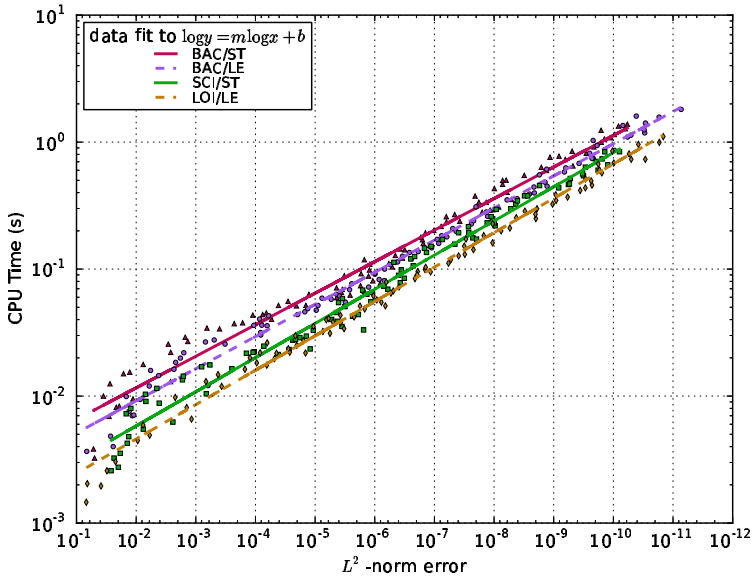

Figure 25: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 5$

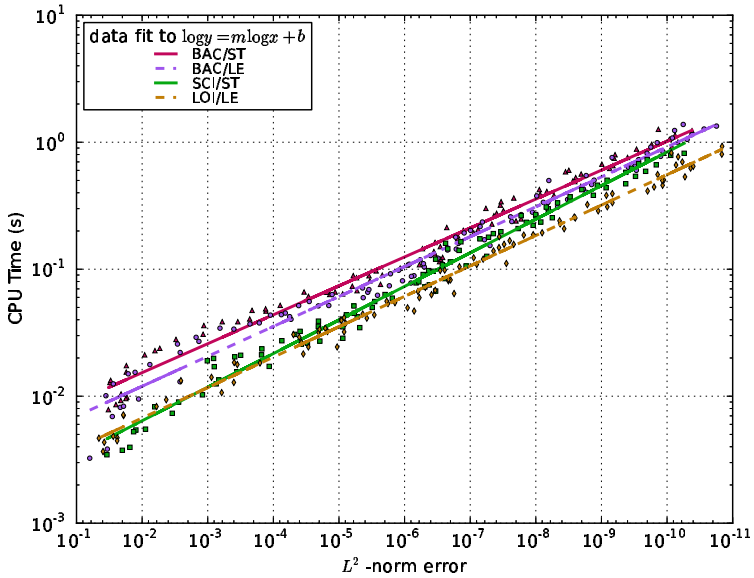

Figure 26: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 6$

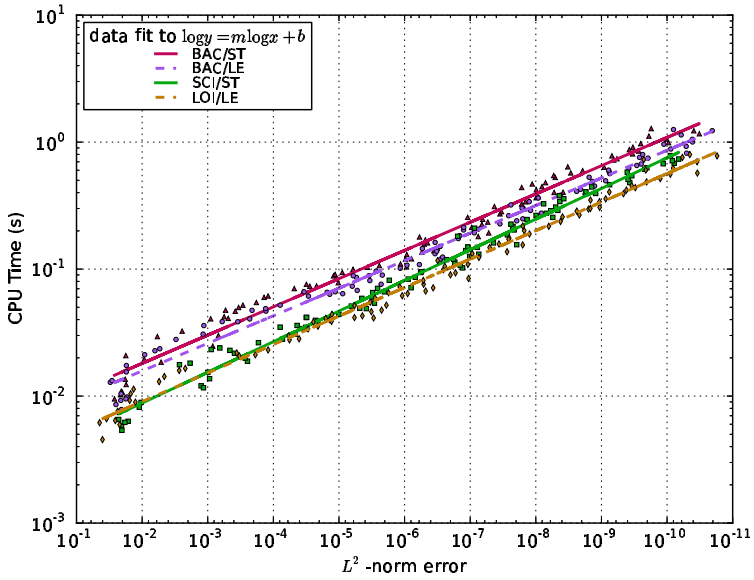

Figure 27: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 7$

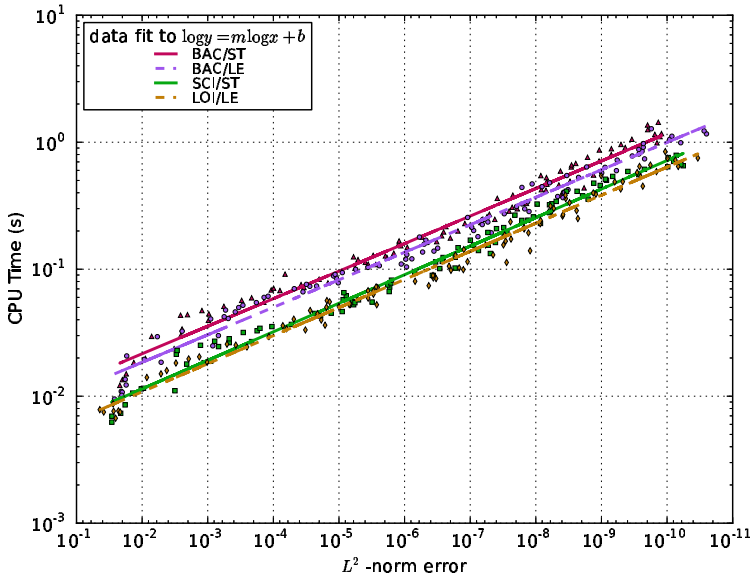

Figure 28: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 8$

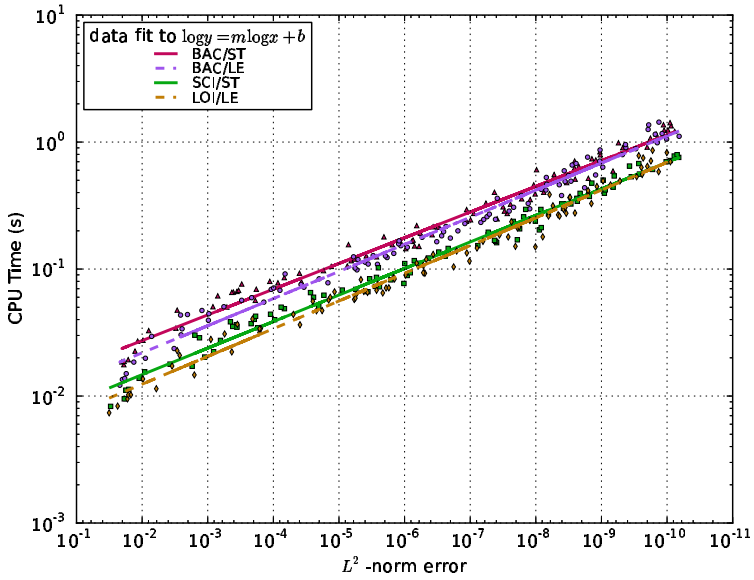

Figure 29: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 9$

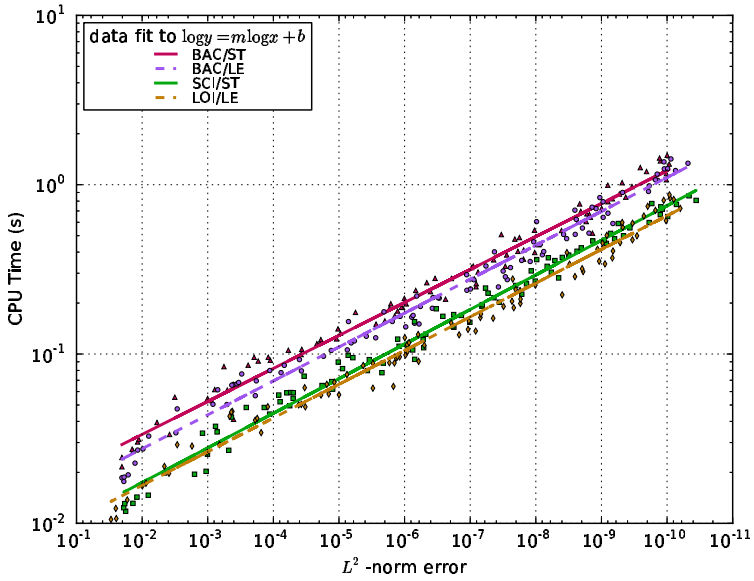

Figure 30: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}, kcol = 10$

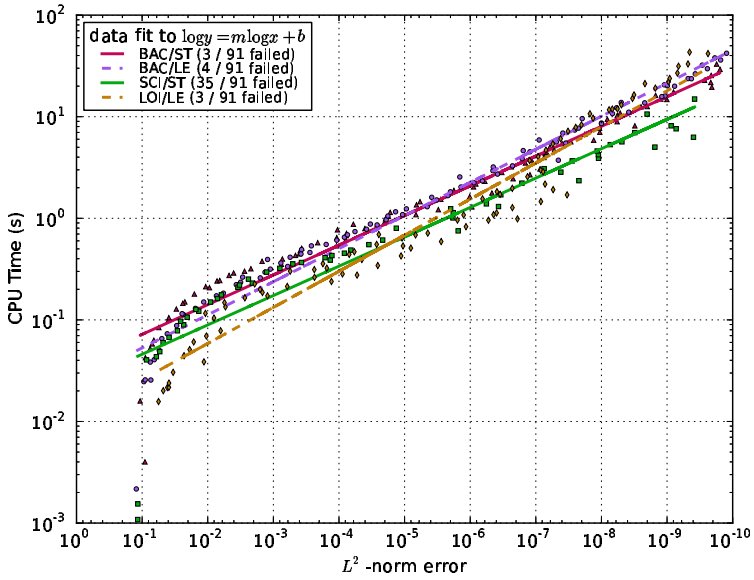

Figure 31: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 3$

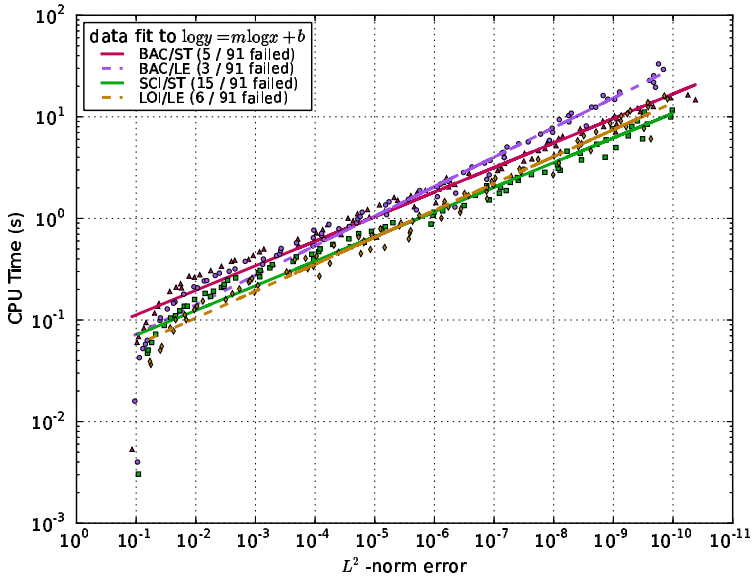

Figure 32: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 4$

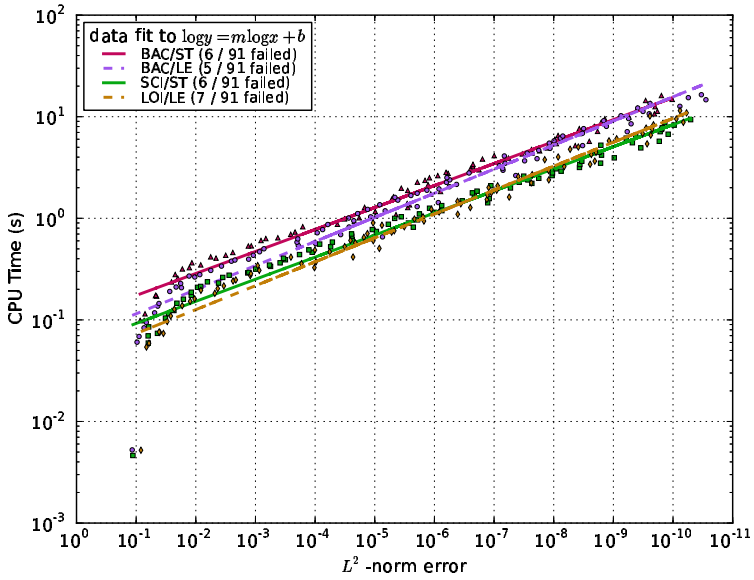

Figure 33: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 5$

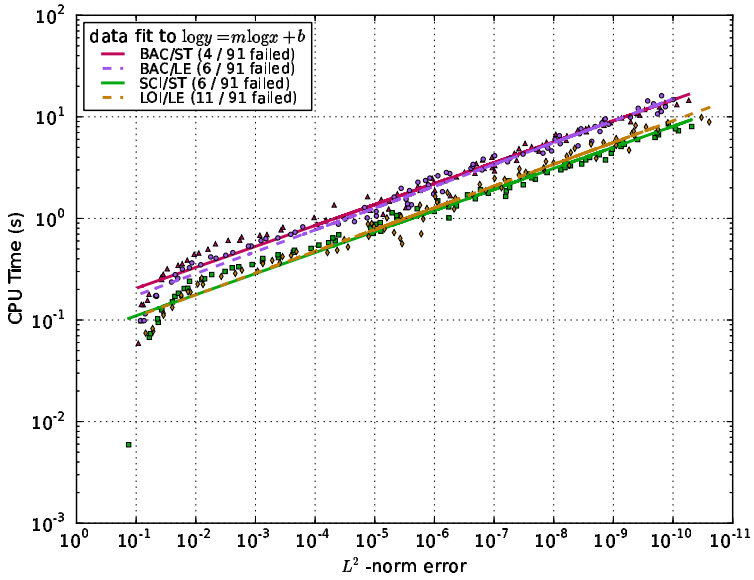

Figure 34: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 6$

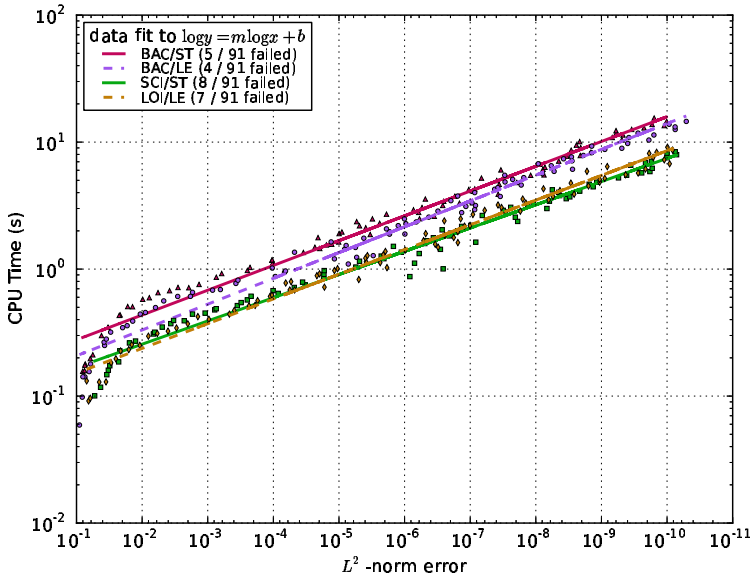

Figure 35: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 7$

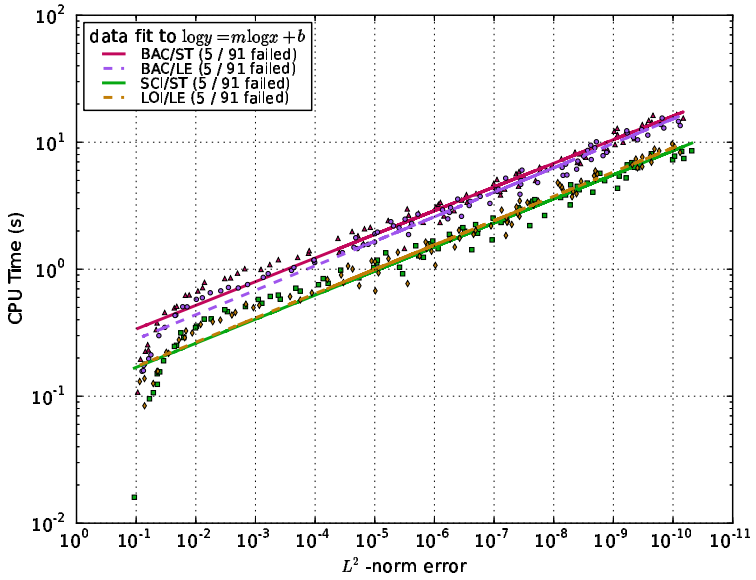

Figure 36: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 8$

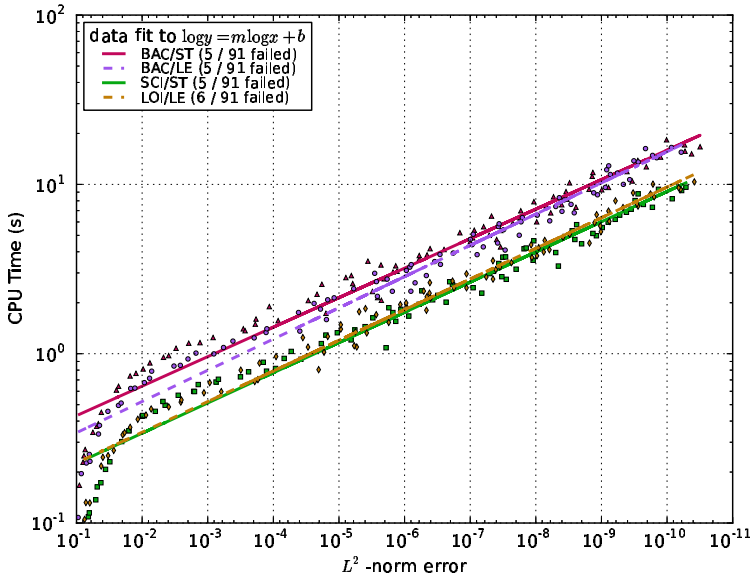

Figure 37: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 9$

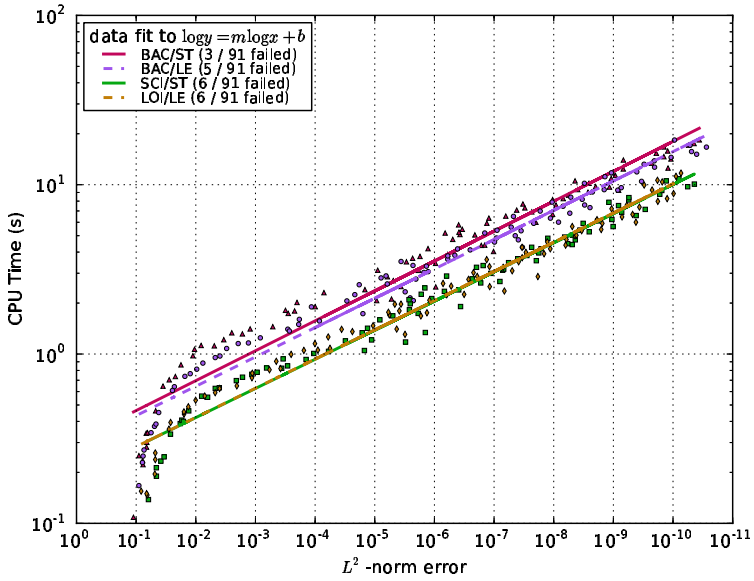

Figure 38: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}, kcol = 10$

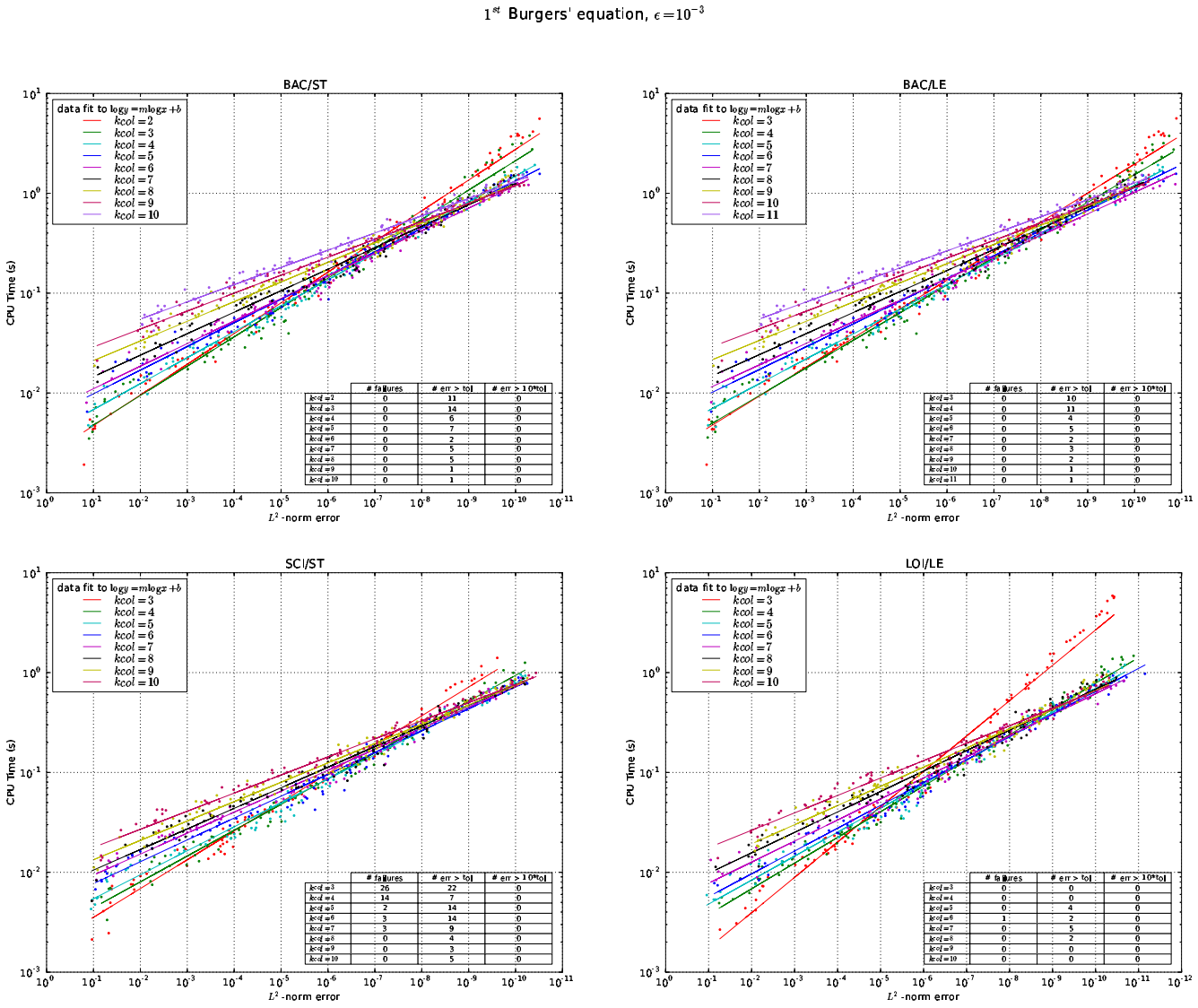

Figure 39: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-3}$; one plot for each code BAC/ST, BAC/LE, SCI/ST, LOI/LE for $kcol = 3, \ldots, 10$ (also $kcol = 2$ for BAC/ST)

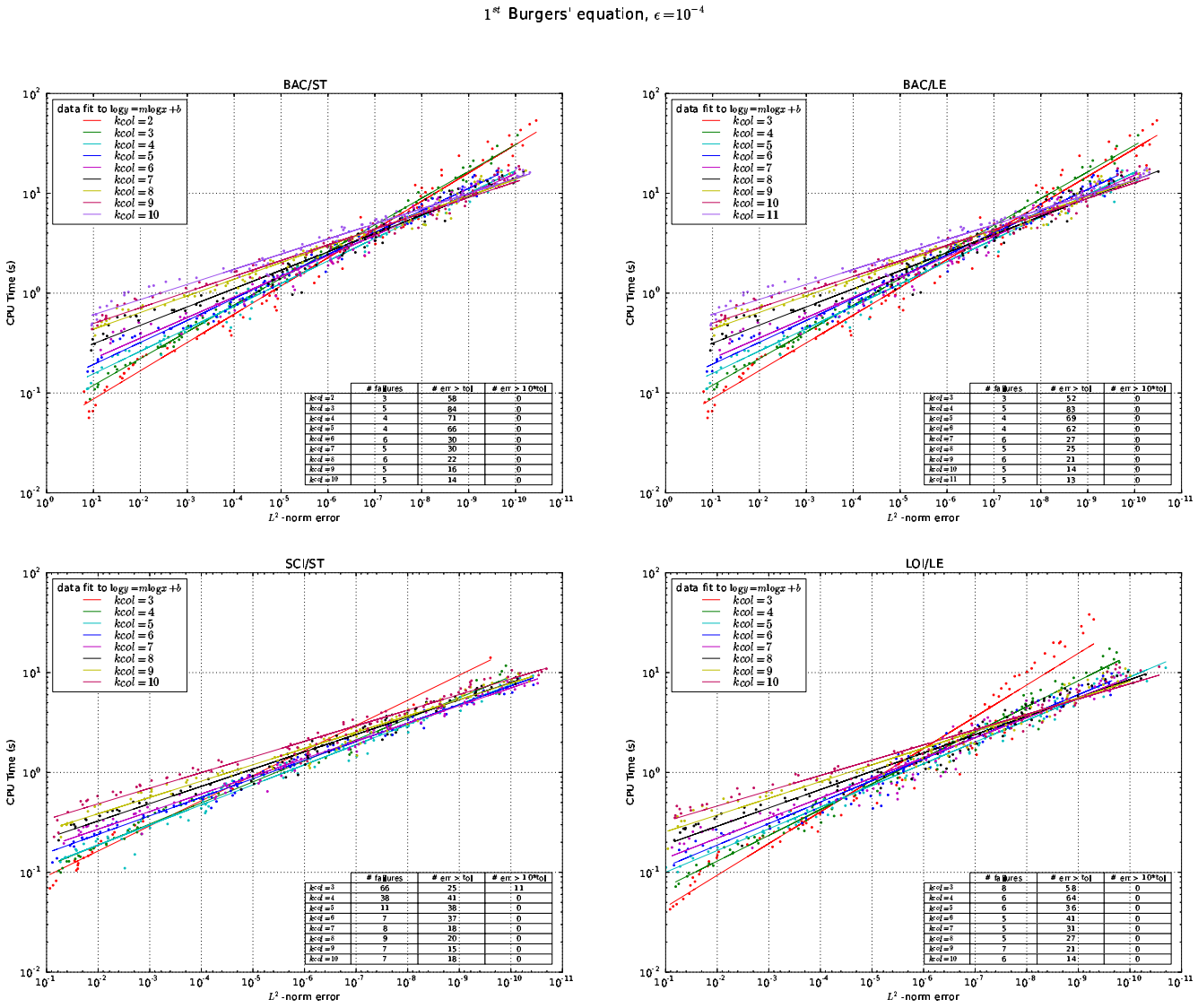

Figure 40: Work vs. Accuracy: One Layer Burgers equation, $\epsilon = 10^{-4}$; one plot for each code BAC/ST, BAC/LE, SCI/ST, LOI/LE for $kcol = 3, \ldots, 10$ (also $kcol = 2$ for BAC/ST)

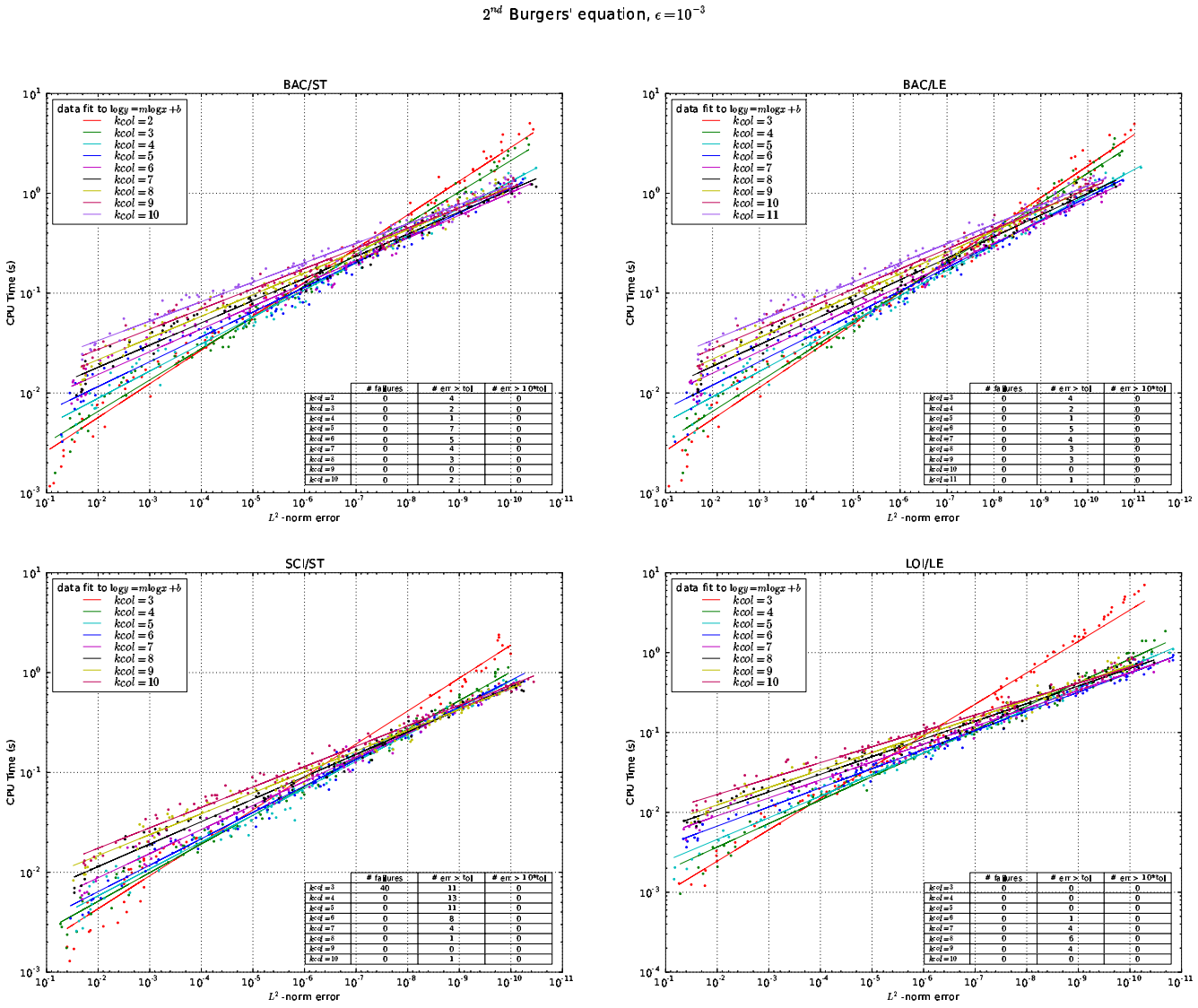

Figure 41: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-3}$; one plot for each code BAC/ST, BAC/LE, SCI/ST, LOI/LE for $kcol = 3, \ldots, 10$ (also $kcol = 2$ for BAC/ST)

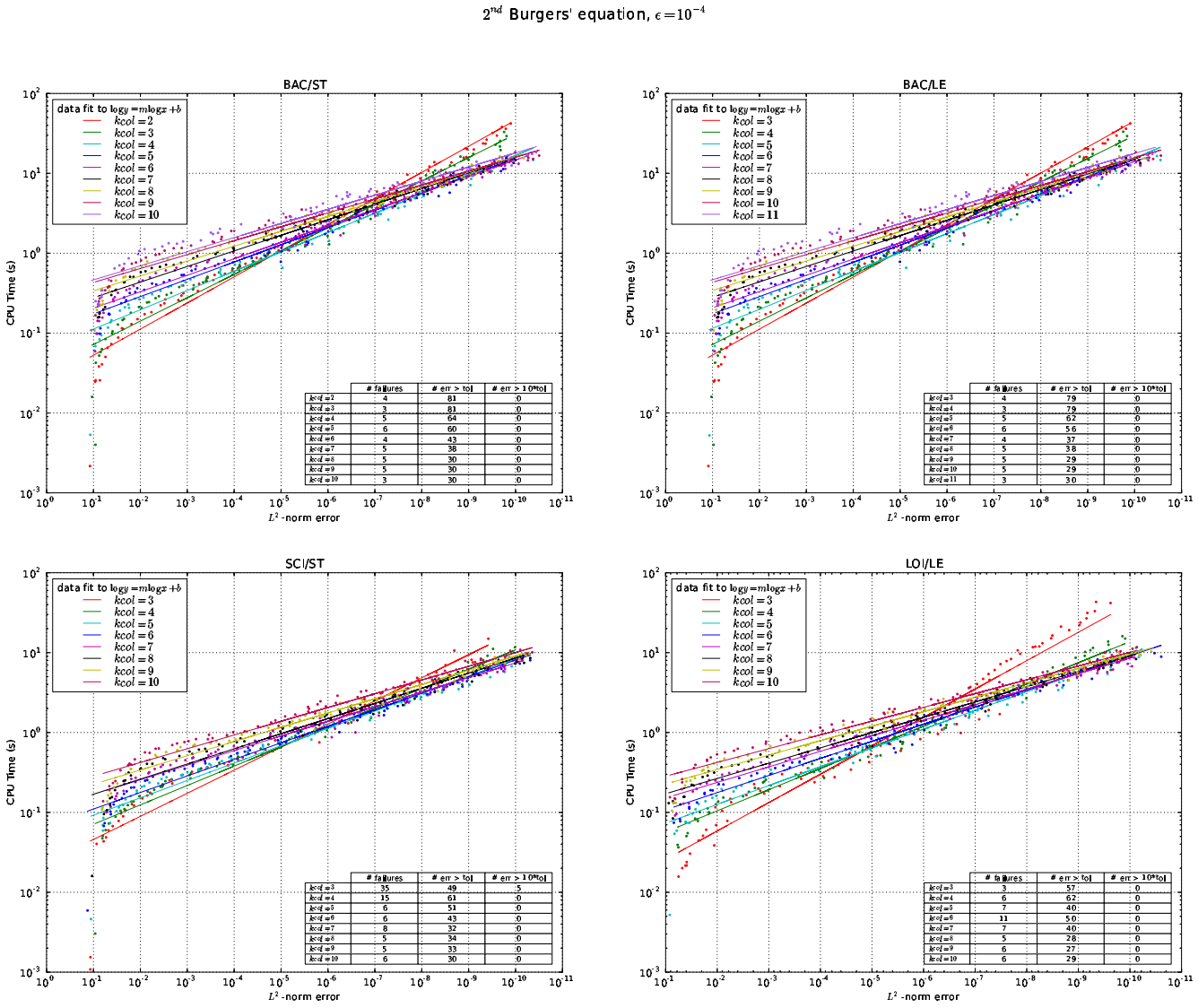

Figure 42: Work vs. Accuracy: Two Layer Burgers equation, $\epsilon = 10^{-4}$; one plot for each code BAC/ST, BAC/LE, SCI/ST, LOI/LE for $kcol = 3, \ldots, 10$ (also $kcol = 2$ for BAC/ST)

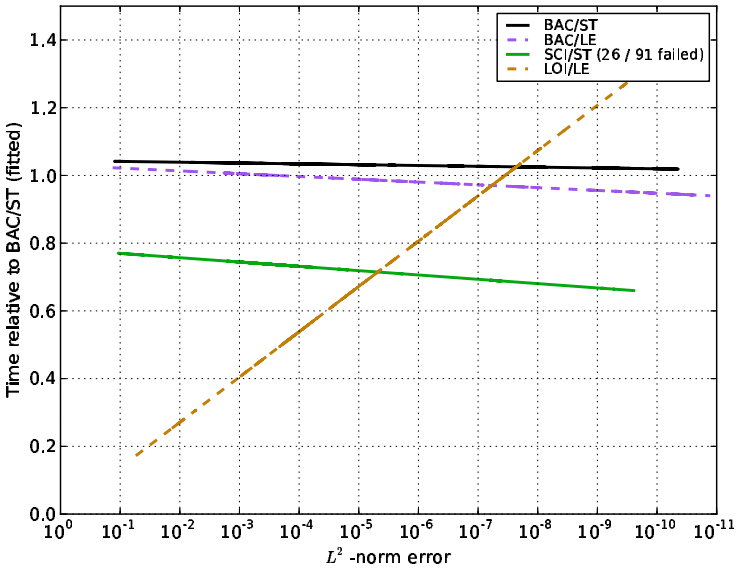

Figure 43: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 3$

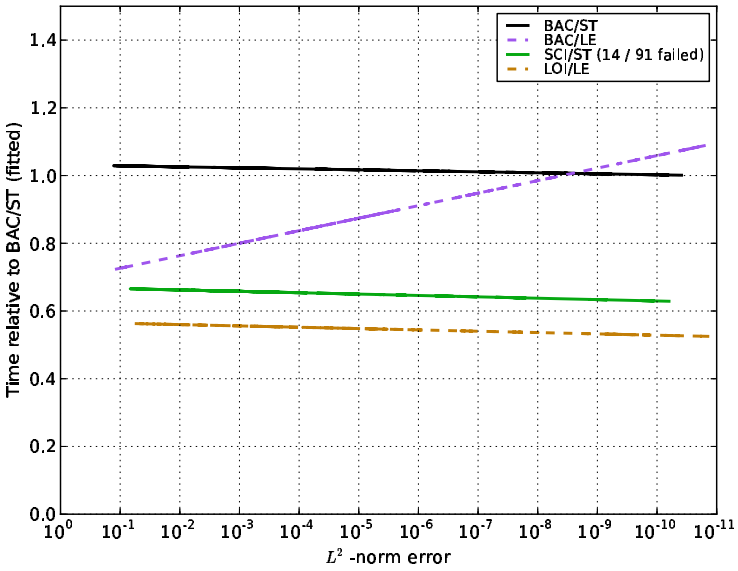

Figure 44: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 4$

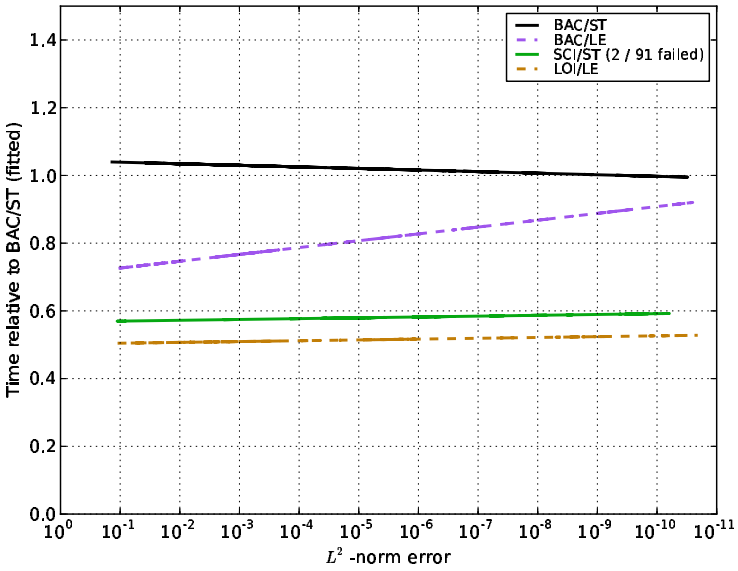

Figure 45: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 5$

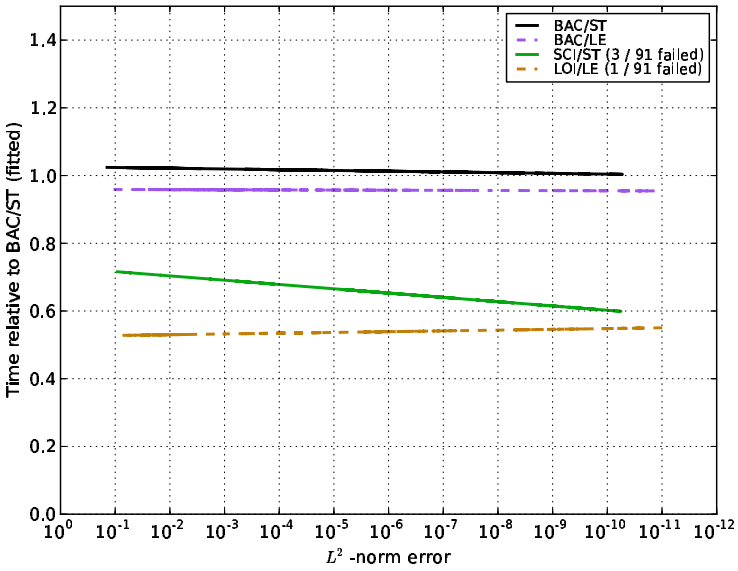

Figure 46: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 6$

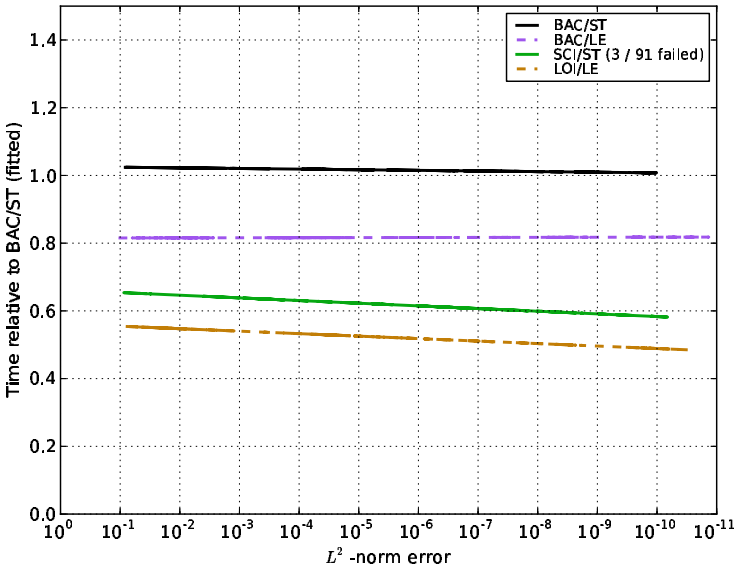

Figure 47: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 7$

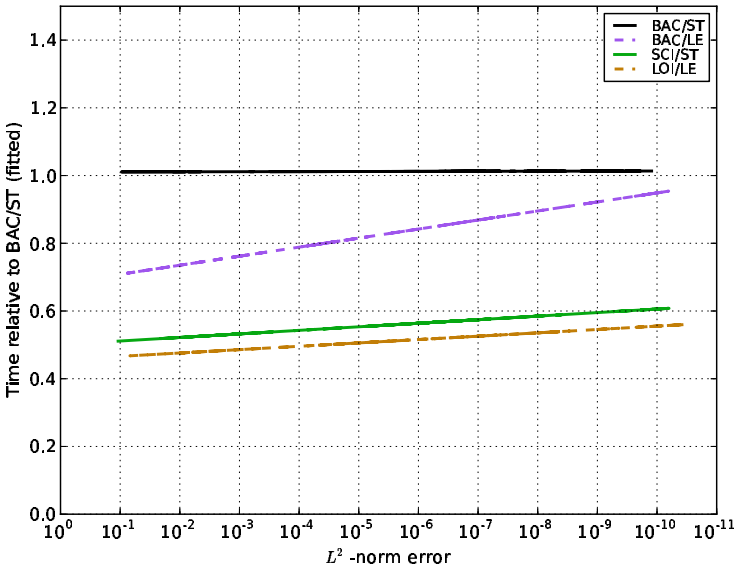

Figure 48: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 8$

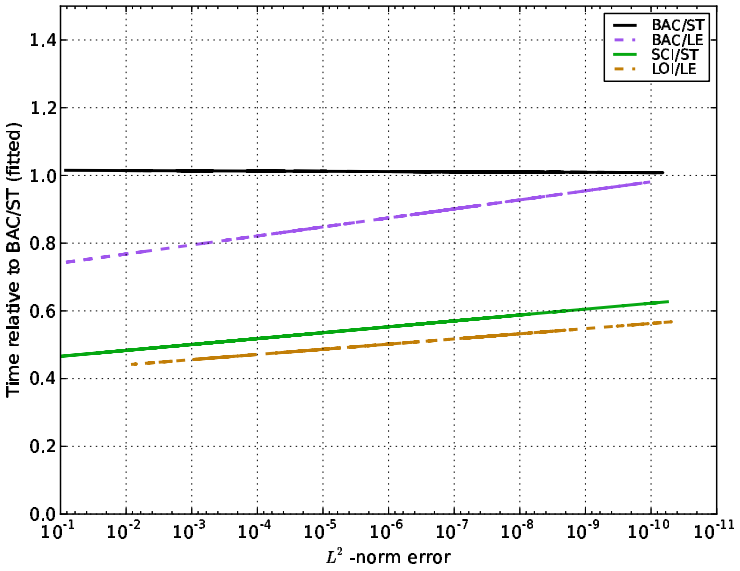

Figure 49: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 9$

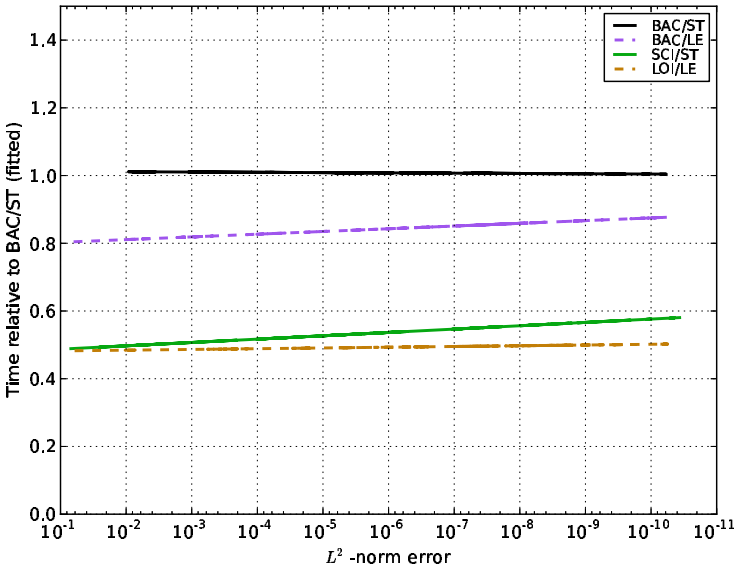

Figure 50: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 10$

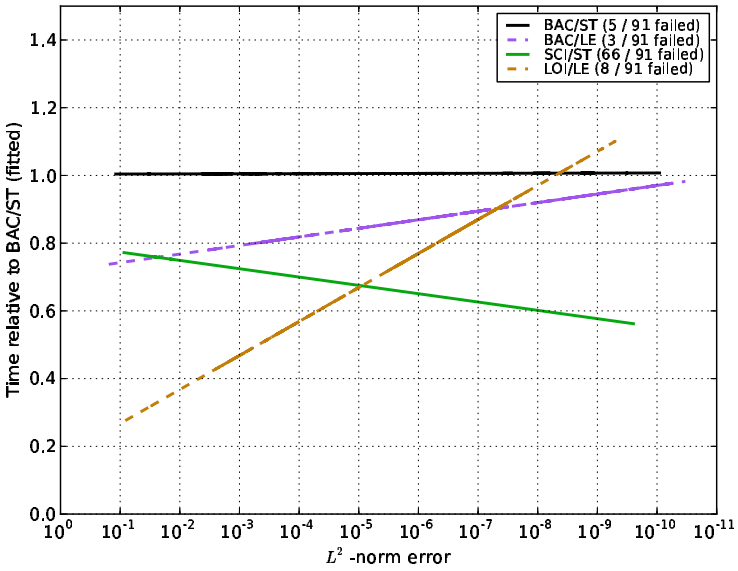

Figure 51: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 3$

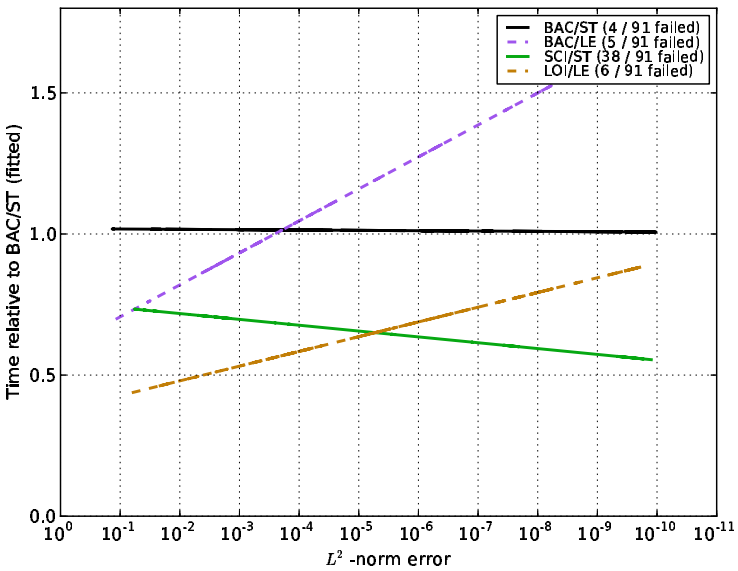

Figure 52: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 4$

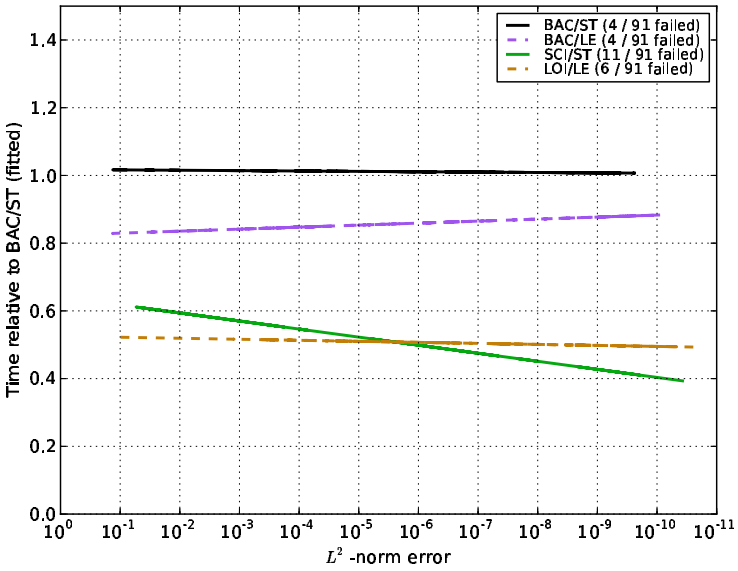

Figure 53: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 5$

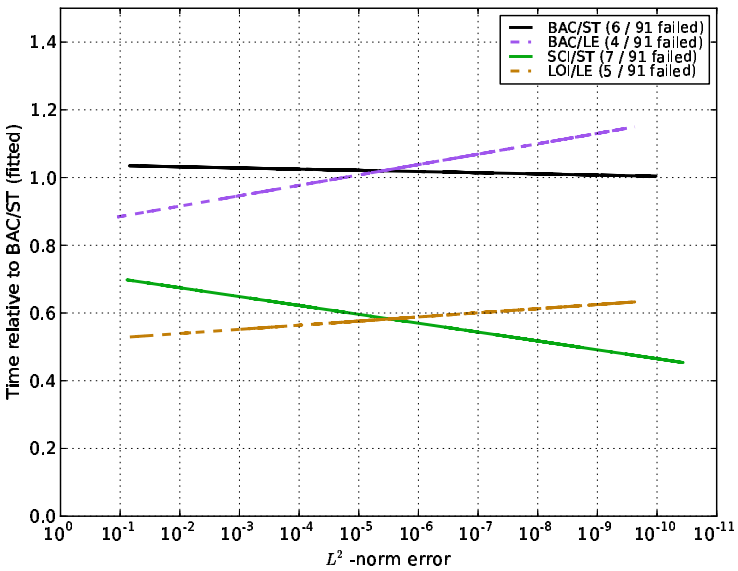

Figure 54: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 6$

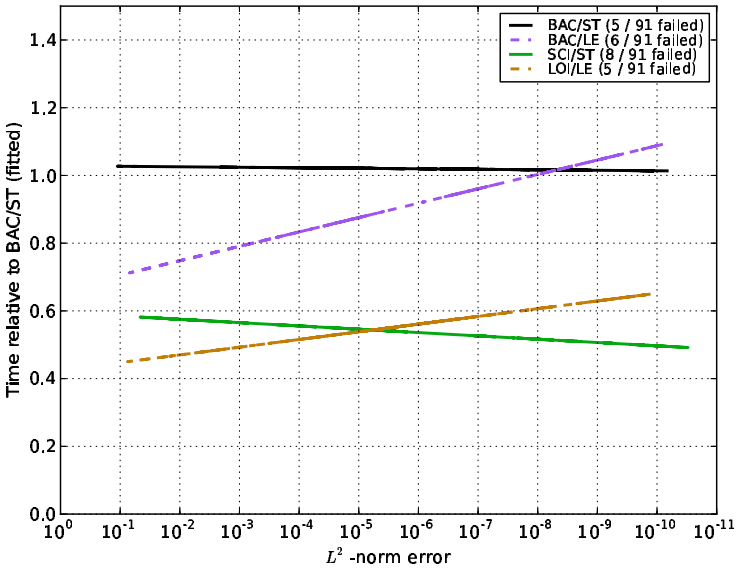

Figure 55: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 7$

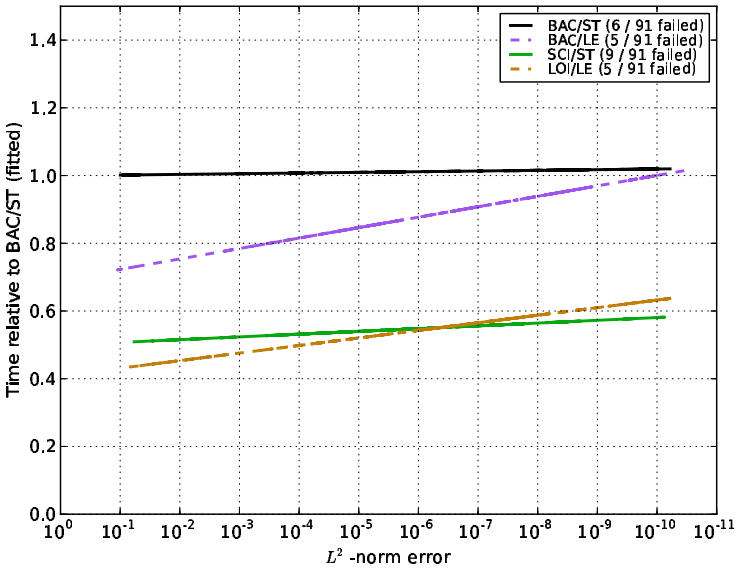

Figure 56: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 8$

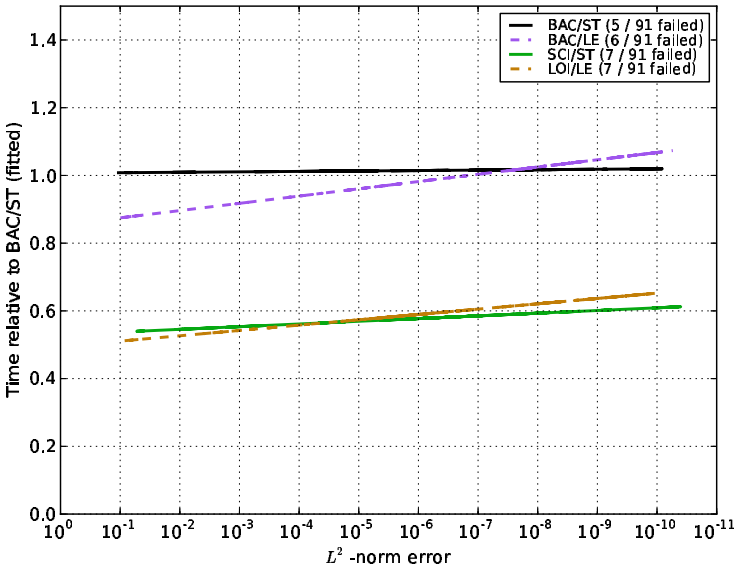

Figure 57: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 9$

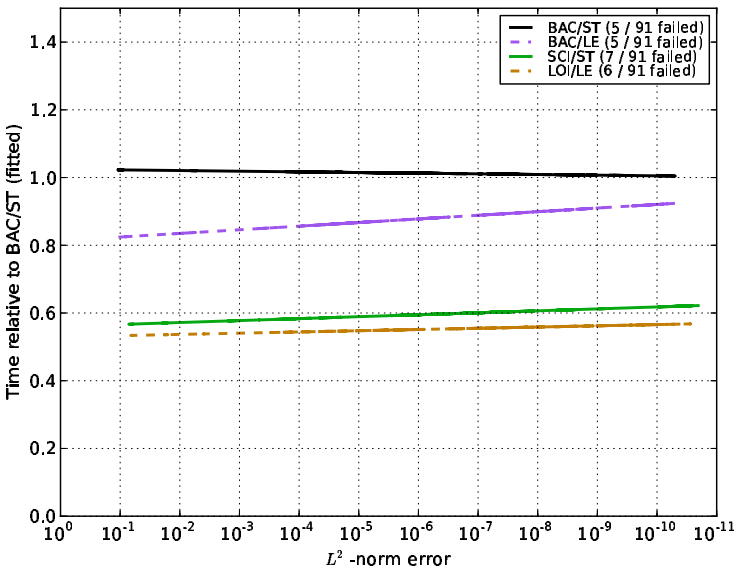

Figure 58: Rel. Work-Accuracy: One Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 10$

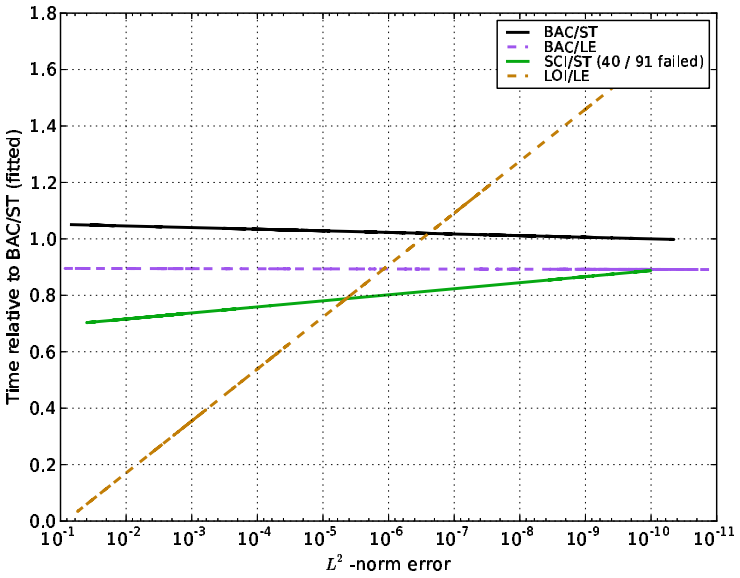

Figure 59: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 3$

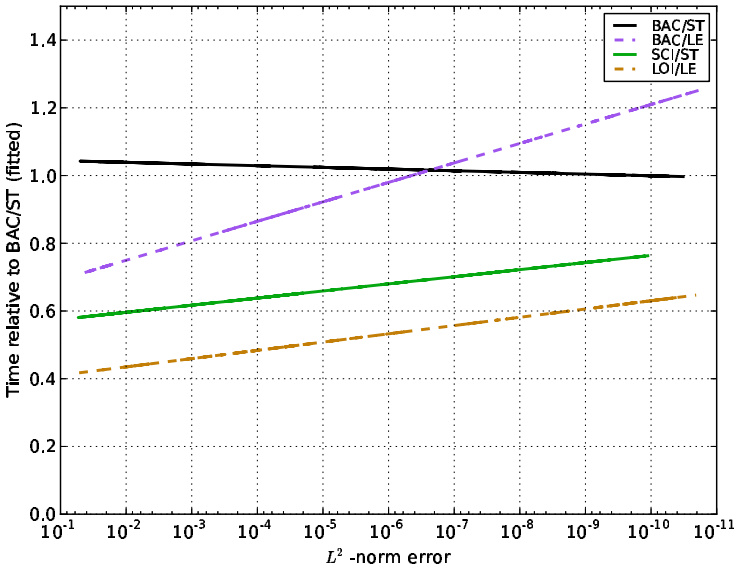

Figure 60: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 4$

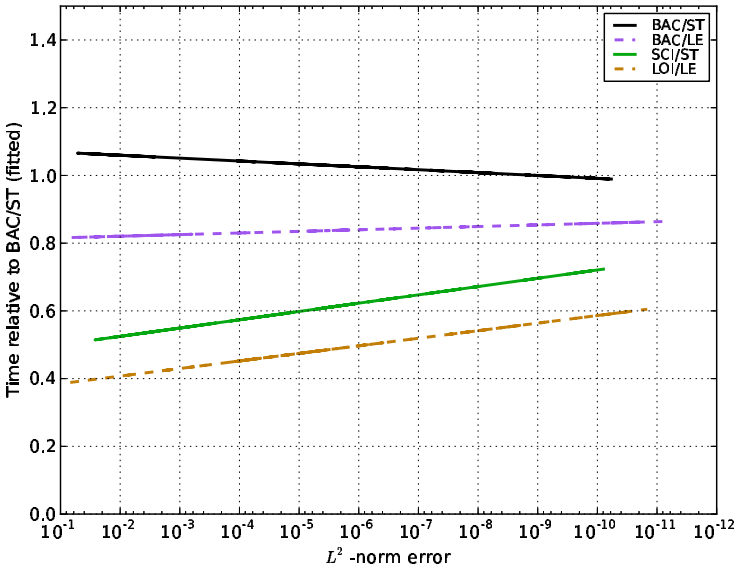

Figure 61: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 5$

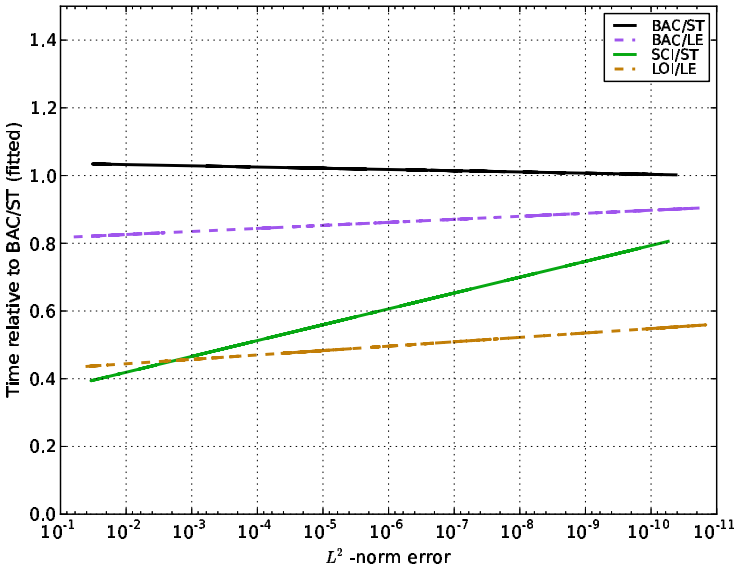

Figure 62: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 6$

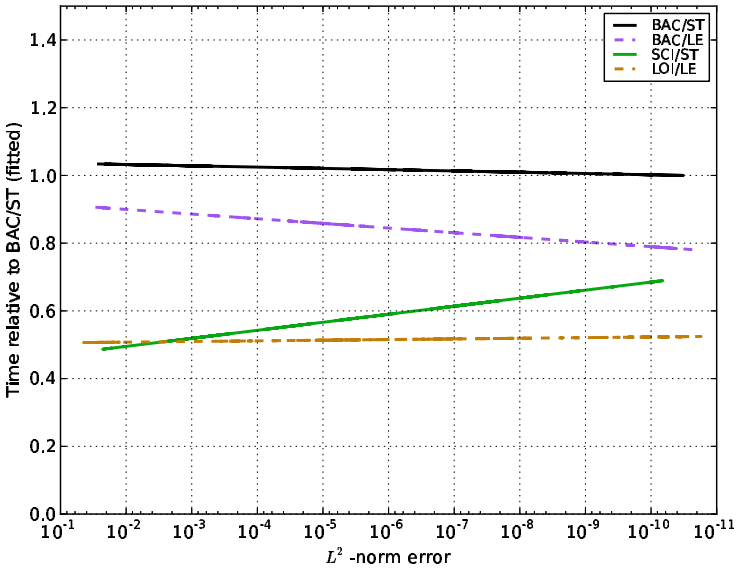

Figure 63: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 7$

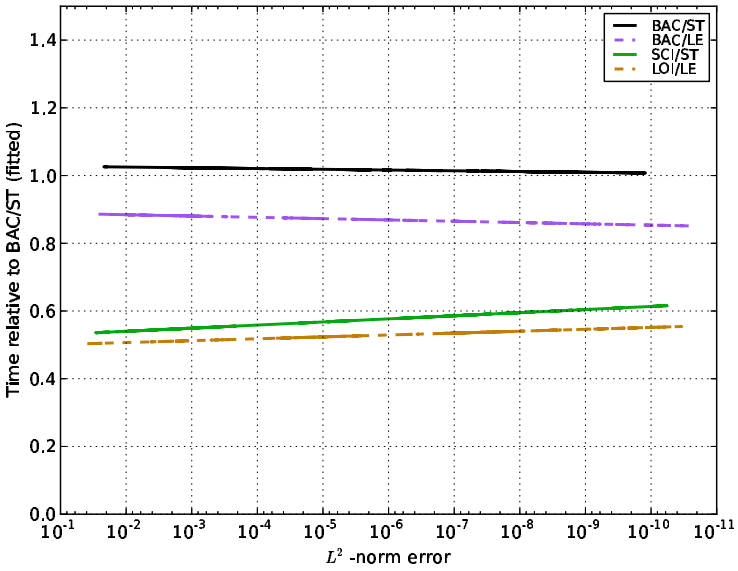

Figure 64: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 8$

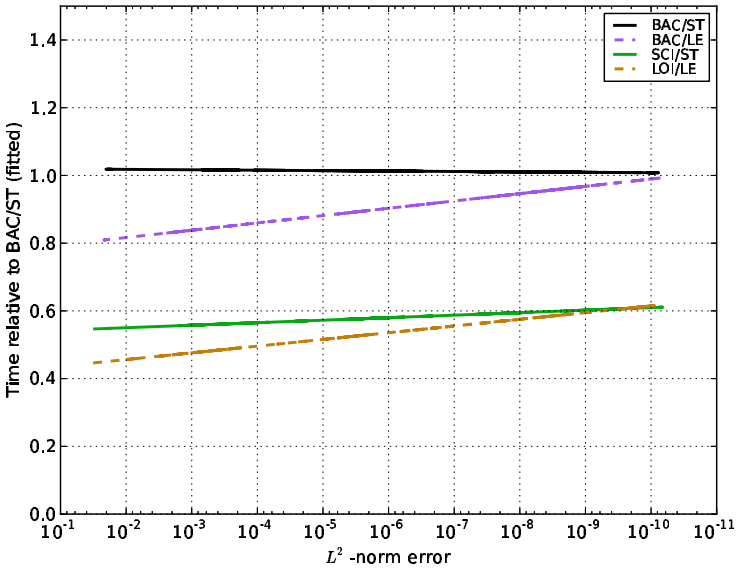

Figure 65: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 9$

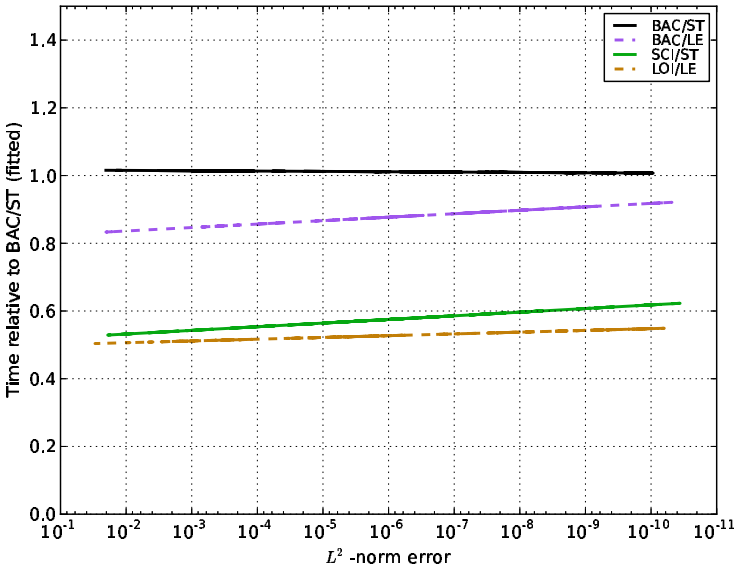

Figure 66: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-3}, kcol = 10$

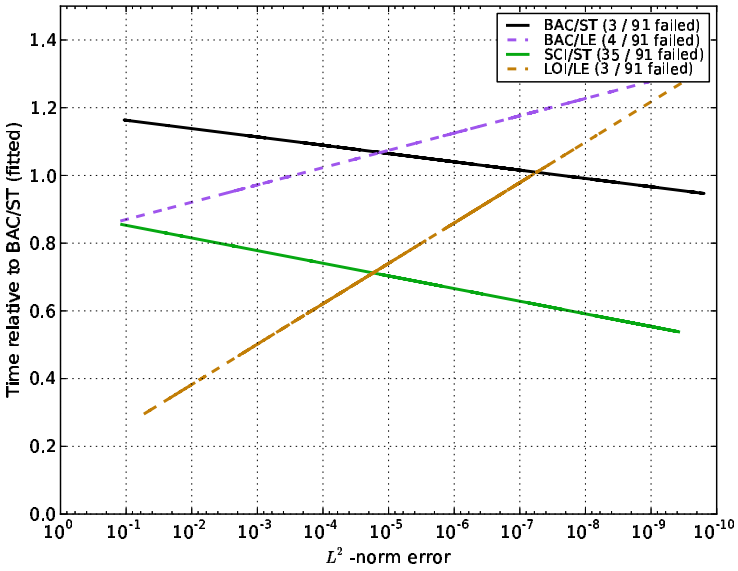

Figure 67: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 3$

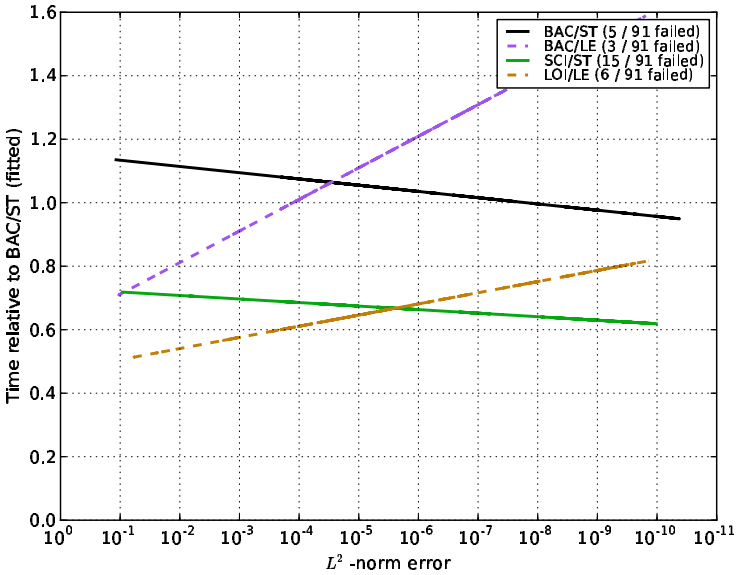

Figure 68: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 4$

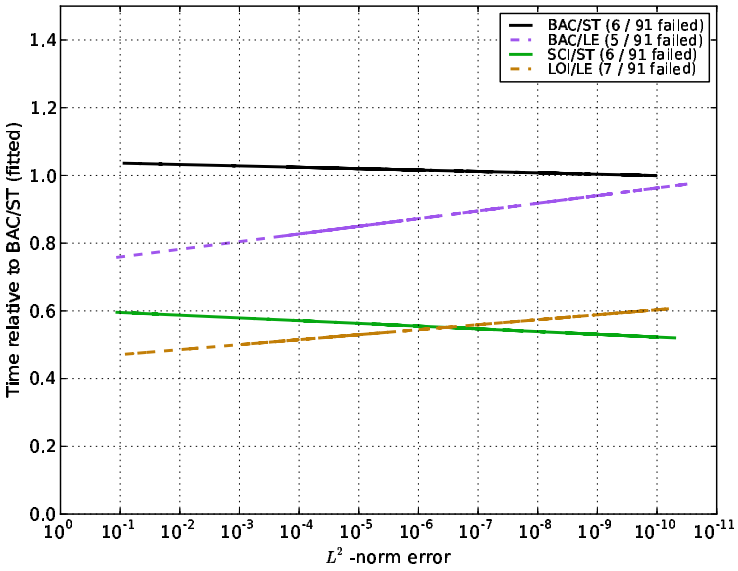

Figure 69: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 5$

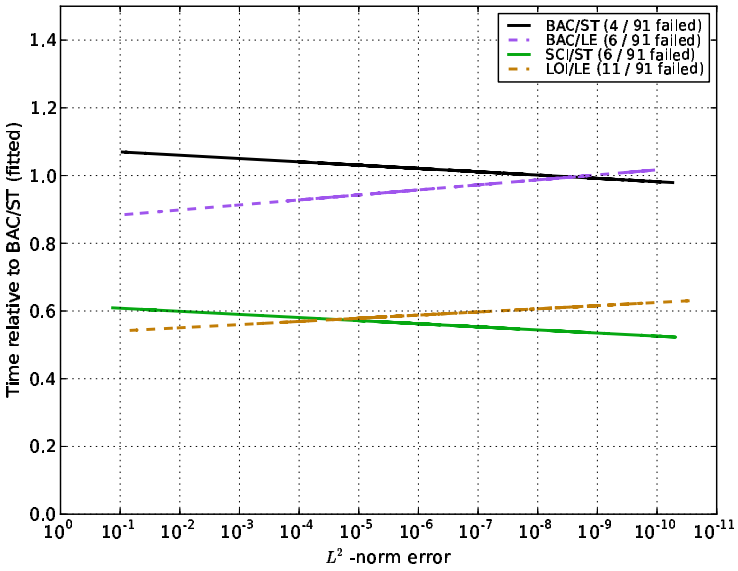

Figure 70: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 6$

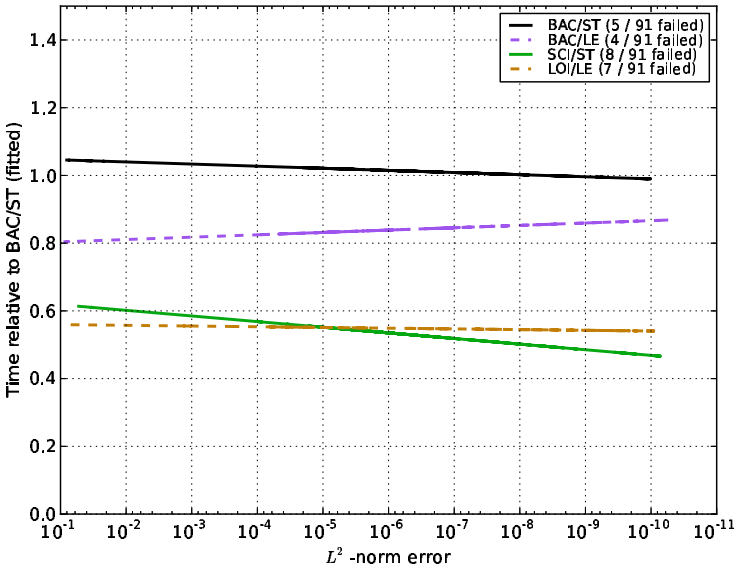

Figure 71: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 7$

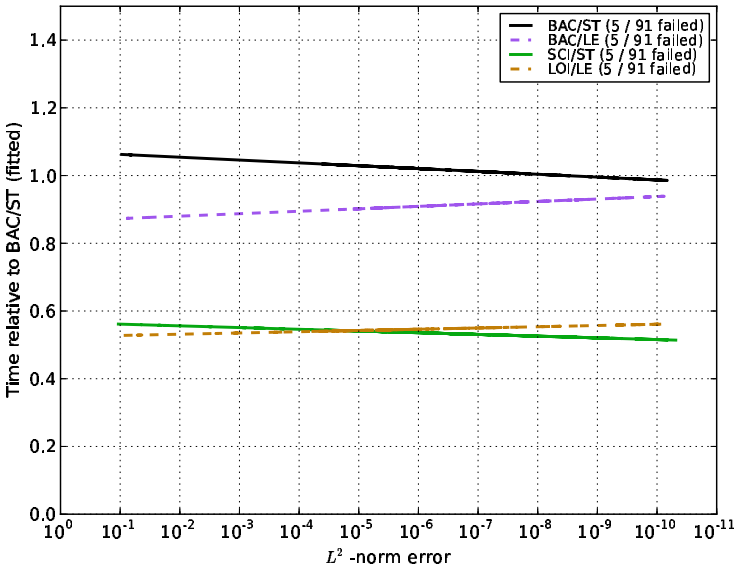

Figure 72: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 8$

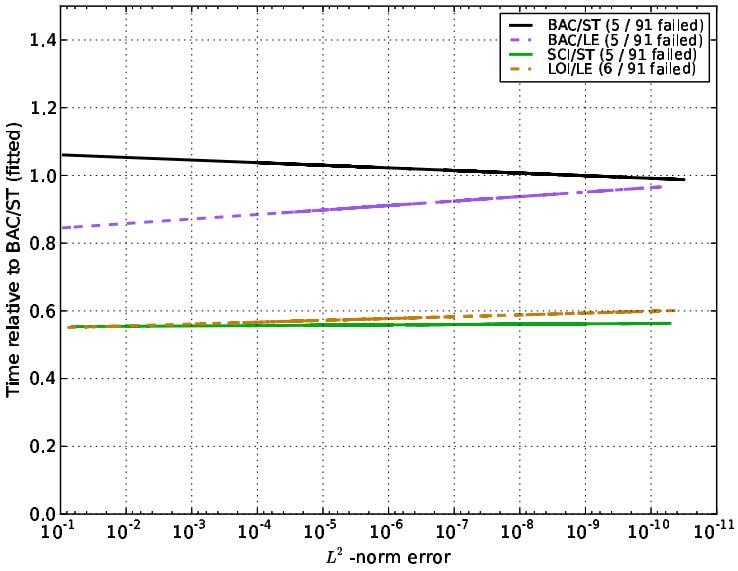

Figure 73: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 9$

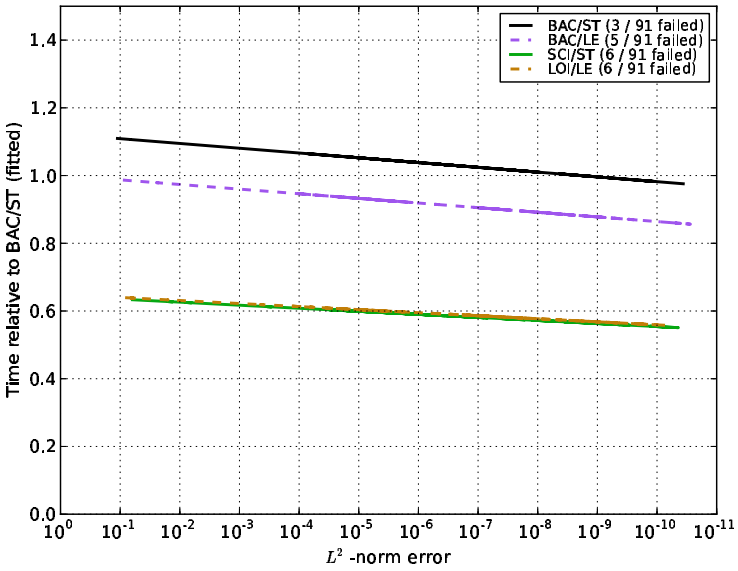

Figure 74: Rel. Work-Accuracy: Two Layer Burgers eqn, $\epsilon = 10^{-4}, kcol = 10$

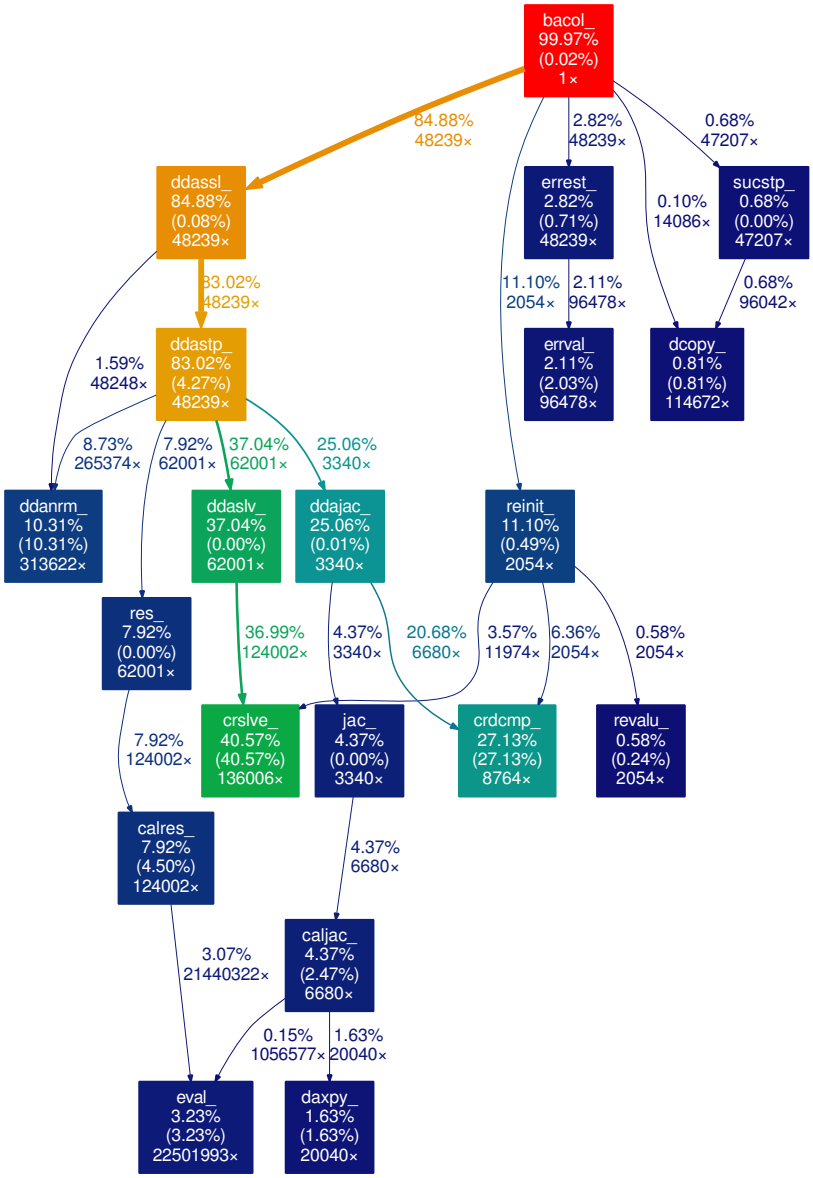

Figure 75: Graphical view of profiling results for BACOL (Standard Error Control Mode) for Two Layer Burgers Equation×12 with $\epsilon = 10^{-4}$ and tolerance $= 10^{-8}$. This graph includes one box for each component of BACOL which represents a significant percentage of the overall execution time. Within each box is listed: the name of the software component, the percentage of overall execution time (including time spent by its children) attributable to the component, the percentage of the overall execution time spent specifically within the component, and the number of times the component is called.

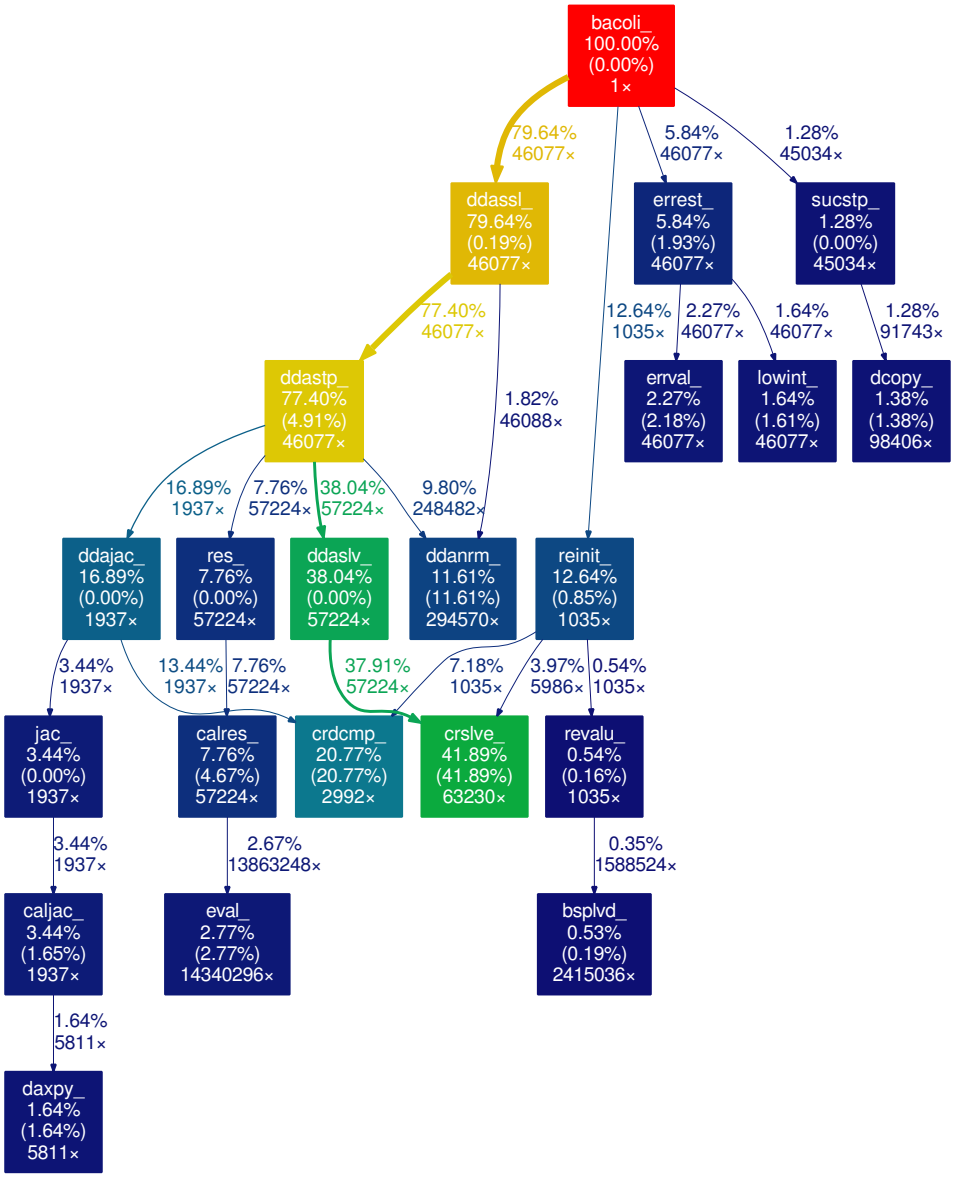

Figure 76: Graphical view of profiling results for BACOLI-LOI (Local Extrapolation Error Control Mode) for Two Layer Burgers Equation×12 with $\epsilon = 10^{-4}$ and tolerance $= 10^{-8}$. This graph includes one box for each component of BACOL which represents a significant percentage of the overall execution time. Within each box is listed: the name of the software component, the percentage of overall execution time (including time spent by its children) attributable to the component, the percentage of the overall execution time spent specifically within the component, and the number of times the component is called.