A Runge-Kutta Type Boundary Value ODE Solver with Defect Control

W.H. Enright and Paul Muir

Technical Report No. 267/93
June 1993.
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract. A popular approach for the numerical solution of boundary value ODE problems involves the use of collocation methods. Such methods can be naturally implemented so as to provide a continuous approximation to the solution over the entire problem interval. On the other hand, several authors have suggested as an alternative, certain subclasses of the implicit Runge-Kutta formulas, known as mono-implicit Runge-Kutta (MIRK) formulas, which can be implemented substantially more efficiently than the collocation methods. These latter formulas do not have a natural implementation that provides a continuous approximation to the solution; rather only a discrete approximation at certain points within the problem interval is obtained. However recent work in the area of initial value problems has demonstrated the possibility of generating inexpensive interpolants for any explicit Runge-Kutta formula. These ideas have recently been extended to develop interpolants for the MIRK formulas. In this paper, we describe our investigation of the use of continuous MIRK formulas in a code for the numerical solution of boundary value ODE problems. A primary thrust of this investigation is to consider defect control, based on these interpolants, as an alternative to the standard use of global error estimates, as the basis for termination and mesh redistribution criteria.

Keywords: Runge-Kutta methods, ordinary differential equations, interpolation

AMS Subject Classifications: 65L05, 65L10

This work was supported by the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Center of Ontario. Author addresses: W.H.Enright, University of Toronto, Toronto, Ontario, Canada, M5S 1A4 (e-mail: enright@na.toronto.edu); Paul Muir, Saint Mary's University, Halifax, Nova Scotia, Canada, B3H 3C3. (e-mail: muir@husky1.stmarys.ca)

1. Introduction

A popular approach for the numerical solution of boundary value ODE (BVODE) problems involves the use of collocation formulas based on, for example, Gauss or Lobatto points. The Gauss point formulas have been implemented in the widely used code, COLSYS [Ascher, Christiansen, and Russell 1981] and its descendant, COLNEW [Bader and Ascher 1987]. These approaches have been shown to be particularly effective in the solution of stiff or singularly perturbed BVODEs [Ascher and Weiss 1983, 1984a, 1984b], [Ascher 1985, 1986], [Ascher and Bader 1986], and [Ascher and Jacobs 1989], For many numerical methods for BVODE problems, one obtains an approximation to the solution at a discrete set of mesh points which partition the problem interval. An advantage of the collocation approach is that a continuous approximation to the solution over the entire problem interval is obtained as a natural consequence of the standard implementation of these methods. For a mesh with subintervals of size h and collocation based on Gauss points with s collocation points per subinterval, the continuous solution provides an approximation with global error $O(h^s)$, while the associated discrete solution at the mesh points has a global error that is (in the abscence of order reduction phenomenon) $O(h^{2s})$, a property known as super-convergence. The continuous solution approximation can be very useful, for example, when the user requires solution information throughout the problem interval. Furthermore it can also be useful within the numerical software package, itself, for example, for error estimation, mesh refinement and redistribution, and generation of initial iterates for Newton iterations.

It is well known ([Weiss 1974]) that the collocation formulas, when applied to first order systems of ODEs, can be viewed as equivalent to certain subclasses of the implicit Runge-Kutta formulas. Recently, several authors have suggested, as an alternative to the collocation formulas, other subclasses of implicit Runge-Kutta formulas for use in the numerical solution of BVODE problems. These formulas are known as mono-implicit Runge-Kutta (MIRK) formulas [vanBokhoven 1980], [Cash and Singhal 1982a]. A recent investigation of MIRK formulas is presented in [Burrage, Chipman, and Muir 1993], where families of order 1 through 6 are characterized. An advantage of these formulas over the collocation formulas, is that the formula calculations on each subinterval can be implemented substantially more efficiently [Cash and Singhal 1982b], [Gupta 1985], [Enright and Muir 1986], [Cash 1986, 1988]. The most recent work using these formulas [Cash and Wright 1990, 1991], makes use of three specific MIRK formulas of orders 4, 6, and 8 within a software package for the numerical solution of BVODEs, called HAGRON. An early software effort, described in [Gupta 1985], involved modifying the PASVA code, [Lentini and Pererya 1977], to use MIRK formulas. The approaches used in the HAGRON code and in the modified PASVA code are fundamantally different from those we consider in this paper. Both of those codes use deferred correction to provide error estimation in order to obtain a discrete solution approximation, while we use continuous MIRK schemes to provide defect control. As well the nonlinear iteration and the mesh selection algorithms are different.

Unlike the collocation formulas, the MIRK formulas do not directly provide a continuous approximation to the solution. The idea of extending the discrete solution approximation to a continuous one has received considerable attention over the last few years, primarily in the area of initial value ODE problems. In that context, a number of authors have demonstrated the possibility of generating inexpensive interpolants for general explicit Runge-Kutta formulas; see, for example, [Enright et al. 1986], [Gladwell et al. 1987], [Owren and Zennaro 1991,1992] and [Verner 1990]. These interpolants are obtained by performing extra derivative evaluations, within the current step, thus preserving the one-step nature of the formula. The usefulness of these interpolants has been demonstrated in error estimation and defect control [Enright 1989a, 1989b] and in detection of discontinuities [Enright et al. 1988]. Recently, [Muir and Owren 1993], have developed continuous versions of the MIRK formulas. In addition to presenting a general analysis, they establish order barriers for continuous MIRK (CMIRK) schemes of orders 1 through 6, and provide characterizations of CMIRK families of these orders.

For BVODEs, the question of extending a discrete solution to obtain a continuous one has received substantially less attention. Most BVODE codes do not provide a continuous solution approximation. Examples of such codes are the PASVA code and the modified version mentioned above, some experimental codes based on special one-sided Runge-Kutta schemes [Kreiss, Nichols, and Brown 1986], [Brown and Lorenz 1988], and the HAGRON code mentioned earlier.

In this paper, we investigate the use of continuous solution approximations based on CMIRK formulas for the numerical solution of BVODEs. A primary thrust of the investigation involves the use of defect control as an alternative to the standard global error estimate control, for use in both mesh selection and accuracy control. The quality of the continuous solution is assessed by examining the defect, which is defined as the amount by which the computed continuous solution fails to satisfy the system of differential equations. It is, in a sense, a more natural choice for control than estimates of the global error, since it is the defect which arises in the analysis of the mathematical conditioning of the underlying problem where appropriate condition numbers are introduced to quantify the sensitivity of the global error to perturbations of the ODEs. See [Ascher, Mattheij, and Russell 1988, Chap. 3, Sec. 4]

This paper is organized as follows. In Section 2 we describe the use of discrete and continuous MIRK schemes in the numerical solution of BVODEs. In Section 3 we briefly discuss the class of CMIRK formulas [Muir and Owren 1993] and describe the specific continuous formulas we have chosen to implement. In Section 4, we present our investigations of implementations of a modified Newton iteration, mesh selection algorithms, and defect control using CMIRK schemes. In section 5, we demonstrate the effectiveness of a numerical software package we have written based on the ideas discussed in the previous sections of the paper. We apply this package to some families of test problems chosen from the literature, and provide corresponding results from the COLNEW code in order to demonstrate the effectiveness of our approach. We close, in Section 6, with our conclusions and a summary of our plans for future work in this area.

2. The numerical solution of BVODEs using MIRK schemes

In this section we review the basic algorithm for solving a system of BVODEs with a MIRK scheme as the underlying discretization formula and then how a continuous version of the discrete MIRK scheme is used to provide defect control and mesh selection capabilities. The basic approach is to use the MIRK formulas to determine a non-linear discrete system which can then be solved with a Newton iteration to obtain a discrete solution. Only after this solution is obtained is a CMIRK scheme employed to provide a continuous solution approximation over the problem interval for use in the computation of defect estimates, mesh redistribution, and initial guesses for subsequent Newton iterates.

We assume that the boundary value ODEs to be solved can be expressed in the general form,

$$\underline{y}'(t) = \underline{f}(t, \underline{y}(t)), \qquad t \in [a, b],$$

where $\underline{y} \in R^n$ and $\underline{f} : R \times R^n \to R^n$, and where we assume for convenience of implementation that we have separated boundary conditions,

$$g(y(a), y(b)) = \begin{pmatrix} g_0(y(a)) \\ g_1(y(b)) \end{pmatrix} = 0,$$

where $g_0 : R^n \to R^{n_0}$ and $g_1 : R^n \to R^{n_1}$ and $n_0 + n_1 = n$. (The non-separated case can be treated at the cost of doubling the size of the ABD system, although it may be possible to avoid this by adapting some of the recently developed algorithms for the parallel solution of ABD systems.)

A standard approach for employing a Runge-Kutta scheme to solve such problems can be described in terms of a two-level iteration scheme:

(0) Prior to beginning the two-level iteration we determine a suitable initial mesh and associated discrete solution approximation (often provided by the user).

(1) The first step of the upper-level iteration is the setup and solution of a discrete system, $\Phi(Y) = 0$, where Φ is the "residual function" and Y is a vector whose i-th component, y_i, is an approximation to the exact solution evaluated at the i-th point of an $N+1$ point mesh which partitions the problem interval into subintervals. The residual function has $N+1$ components, each of size n: one component associated with the boundary conditions, g_0 and g_1, and N components, ϕ_i, $i = 0, \ldots, N-1$, where ϕ_i is associated with the use of the Runge-Kutta scheme on the $(i+1)$-st subinterval. We solve this discrete system using a modified Newton iteration, which constitutes the lower-level iteration. The Jacobian of the Newton system has an almost block diagonal structure with blocks $B_0 = \dfrac{\partial g_1}{\partial y_0}$ and $B_1 = \dfrac{\partial g_1}{\partial y_N}$ at the top and bottom positions of the block diagonal, and blocks $L_i = \dfrac{\partial \phi_i}{\partial y_i}$, and $R_i = \dfrac{\partial \phi_i}{\partial y_{i+1}}$, $\quad i = 0, \ldots, N-1$, across the ith block row of the block diagonal. The blocks L_i, R_i, and the block $[B_0, B_1]^T$, are all of size n by n. When a MIRK scheme is used as the underlying discretization the ith component of the residual function takes the form,

$$\phi_i = y_{i+1} - y_i - h_i \sum_{r=1}^{s} b_r K_r, \tag{2.1}$$

where, the stages, K_r, are given by,

$$K_r = f(t_i + c_r h_i, (1 - v_r) y_i + v_r y_{i+1} + h_i \sum_{j=1}^{r-1} x_{rj} K_j), \qquad r = 1, \ldots, s. \tag{2.2}$$

Because the stages are defined explicitly in terms of the unknowns y_i and y_{i+1}, these methods can be implemented quite efficiently. The corresponding partial derivatives are also easily computed. The expressions for L_i and R_i follow simply by differentiating ϕ_i with respect to y_i and y_{i+1}, respectively. This yields

$$L_i = -I - h_i \sum_{r=1}^{s} b_r \left(\frac{\partial K_r}{\partial y_i} \right) \quad \text{and} \quad R_i = I - h_i \sum_{r=1}^{s} b_r \left(\frac{\partial K_r}{\partial y_{i+1}} \right), \tag{2.3}$$

where

$$\frac{\partial K_r}{\partial y_i} = \mathbf{J}_r \left((1-v_r)I + h_i \sum_{j=1}^{r-1} x_{rj} \frac{\partial K_j}{\partial y_i} \right)$$

and

$$\frac{\partial K_r}{\partial y_{i+1}} = \mathbf{J}_r \left(v_r I + h_i \sum_{j=1}^{r-1} x_{rj} \frac{\partial K_j}{\partial y_{i+1}} \right), \qquad r = 1, \ldots, s. \tag{2.4}$$

Here $\mathbf{J}_r$ is the evaluation of $\dfrac{\partial f(t, y(t))}{\partial y}$ at the argument of the rth stage of the MIRK scheme, as given in (2.2).

(2) The previous step uses a MIRK formula to establish a discrete system which is then solved with a Newton iteration. On convergence, this yields a discrete solution approximation, $y_i, i = 0, \ldots, N$, on the given mesh. We then use an associated CMIRK scheme to construct a C^1 continuous solution approximation over the entire problem interval. It is important to note that although the construction of the continuous approximation requires the evaluation of additional stages of the form (2.2), the associated calculations are explicit. As a result, in contrast to the situation for initial value ODEs where explicit Runge-Kutta formulas are usually used for the discrete solution, the cost of obtaining the required continuous approximation is a small part of the overall cost associated with the computation of the discrete solution on each subinterval. On each subinterval the CMIRK scheme provides a polynomial interpolant to the approximate solution over that subinterval. The interpolant is obtained using information that is local to the step. (This is particularly relevant when one wishes to consider a parallel implementation where it is important that the calculations associated with each subinterval be based on local information only.) The continuous solution approximation, which we will call $u(t)$, has the same order of accuracy as the underlying discrete solution. The defect, $\delta(t)$, is defined by

$$\delta(t) = u'(t) - f(t, u(t)), \quad a \le t \le b.$$

We sample the defect on each subinterval, to obtain an estimate of the maximum magnitude of the defect on that subinterval. This algorithm terminates if this estimate on each subinterval is less than a given user-defined tolerance.

(3) If the termination condition is not met, the relative sizes of the maximum defect estimates associated with each subinterval are examined in the mesh selection algorithm to determine how a more appropriate mesh can be constructed. The idea is to redistribute the mesh while possibly adding or deleting points so that the maximum defect estimates obtained on the new mesh will be approximately the same on each subinterval (equidistribution of the defect) and will be bounded by the user tolerance. The maximum defect estimates are employed within a monitor function which provides the basis for the equidistribution process. Based on these considerations a new mesh is determined. See [Ascher, Mattheij, and Russell 1988, Chap. 9, Sec. 1]. The algorithm is terminated, unsuccessfully, if the predicted number of meshpoints for the new mesh is too large (according to a user-defined storage restriction).

(4) When a new mesh is determined, the continuous solution approximation is used to compute an initial iterate for the next Newton iteration.

This algorithm is then repeated, beginning at step (1), using this new mesh and solution approximation.

We will discuss further details of the Newton iteration, defect estimation, and mesh selection algorithms in section 4. In the next section we employ the work of [Muir and Owren 1993] on CMIRK schemes to derive the specific CMIRK formulas which we have implemented in our code.

3. CMIRK Schemes

3.1. Overview

In the paper [Muir and Owren 1993], the class of CMIRK schemes is investigated. We now briefly summarize their work and discuss how it is applied in the computation of continuous solution approximations for BVODE problems. As mentioned in the previous section, the CMIRK scheme is employed only after the discrete solution has been obtained for the current mesh. By saving the stages from the MIRK scheme that were computed for each subinterval during the computation of the discrete solution, we can save some of the computational cost associated with the construction of the stages for the CMIRK scheme for each subinterval. For the ith subinterval, the basic form of a CMIRK scheme is a polynomial in θ, (where $\theta = \frac{t-t_i}{h_i}$),

$$u(t) = u(t_i + \theta h_i) = y_i + h_i \sum_{r=1}^{s^*} b_r(\theta) K_r, \quad 0 \le \theta \le 1, \quad t_i \le t \le t_{i+1}, \quad s^* \ge s, \tag{3.1}$$

with stages

$$K_r = f(t_i + c_r h_i, (1 - v_r) y_i + v_r y_{i+1} + h_i \sum_{j=1}^{r-1} x_{rj} K_j), \quad r = 1, \ldots, s^*. \tag{3.2}$$

Observe that the stage values, (3.2), have the same structure as for the discrete MIRK scheme, (2.2); in fact the first s stages are precisely those of the discrete MIRK scheme used during the computation of the discrete solution. The remaining $s^* - s$ stages are defined by determining appropriate sets of new coupling coefficients $x_{rj}, r = s+1, \ldots, s^*, j = 1, \ldots, r-1$. The definition of the CMIRK is completed with the determination of the s^* weight polynomials, $b_r(\theta), r = 1, \ldots, s^*$. These are polynomials of maximum degree p, where p is the order of accuracy of the discrete MIRK scheme. The extra stages and the weight polynomials are constructed to satisfy continuous order conditions up to and including order p so that on each subinterval, the CMIRK scheme satisfies

$$\max_{0 \le \theta \le 1} ||y(t_i + \theta h_i) - u(t_i + \theta h_i)|| = O(h_i^{p+1}), \tag{3.3}$$

where $y(t)$ is the exact solution to the local initial value ODE,

$$y'(t) = f(t, y(t)), \quad y(t_i) = y_i,$$

$y(t_i) = u(t_i)$, and $||\cdot||$ is any norm on R^n. For all CMIRK schemes, $u(t_i) = y_i$, and for those satisfying an additional condition known as the $C(3)$ stage order condition, $u(t_{i+1}) = y_{i+1}$.

Included in [Muir and Owren 1993] is a presentation of multi-parameter families of CMIRK stages having a minimal number of stages for each order $p = 1, \ldots, 6$. Similarly, a portion of the [Burrage, Chipman and Muir 1993] paper is devoted to the presentation of multi-parameter families of (discrete) MIRK schemes having a minimal number of stages for each order $p = 1, \ldots, 6$. Throughout this paper, we will refer to CMIRK and MIRK schemes having a minimal number of stages for a given order as optimal schemes. It is of course always the case that the number of stages needed to get a discrete MIRK scheme is less than or equal to that for the associated CMIRK scheme. In this section, we will choose optimal discrete MIRK schemes from the families presented in [Burrage, Chipman, and Muir 1993] and then attempt to embed the discrete schemes in optimal CMIRK schemes. The CMIRK schemes will be determined by employing the additional stage coefficients and weight polynomials according to the characterizations given in [Muir and Owren 1993].

Since the families of schemes given in [Burrage, Chipman, and Muir 1993] and [Muir and Owren 1993] involve several parameters, there is usually substantial freedom in the choice of a particular MIRK scheme and the associated CMIRK scheme. An open problem is to determine appropriate criteria for the determination of good MIRK schemes and good CMIRK schemes for use in the numerical solution of boundary value ODEs with defect control. In the current paper, where the selection of the MIRK and CMIRK schemes is only one component of the project, we have selected schemes which are efficient, have satisfactory accuracy properties, and have been demonstrated to work well on several test problems. It should be noted that the focus of this paper is to investigate the potential advantages of *using* MIRK and CMIRK schemes in a software package for the numerical solution of boundary value ODEs, rather than to develop optimal schemes of these types. We will pursue this latter investigation in a subsequent study.

We next present the particular MIRK and CMIRK schemes which have been implemented with our code. The coefficients defining a MIRK scheme, (2.1),(2.2) are usually presented in the form of a tableau as in Table 3.1.

Table 3.1

$$
\begin{array}{c|c|ccccc}
c_1 & v_1 & 0 & 0 & \ldots & 0 & 0 \\
c_2 & v_2 & x_{21} & 0 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
c_{s-1} & v_{s-1} & x_{s-1,1} & x_{s-1,2} & \ldots & 0 & 0 \\
c_s & v_s & x_{s,1} & x_{s,2} & \ldots & x_{s,s-1} & 0 \\
\hline
 & & b_1 & b_2 & \ldots & b_{s-1} & b_s
\end{array}
$$

Later in the paper, we will use the following compact notation for the above scheme: $c = [c_1, c_2, \ldots, c_s]^T$, $v = [v_1, v_2, \ldots, v_s]^T$, $b = [b_1, b_2, \ldots, b_s]^T$, and X is the $s \times s$ matrix whose (i,j)th component is x_{ij}. For a CMIRK scheme, as given by (3.1),(3.2), the tableau has the form shown in Table 3.1 but the bottom row will consist of weight polynomials, $b_1(\theta), b_2(\theta), \ldots, b_s(\theta)$, and we will use the notation, $b(\theta) = [b_1(\theta), b_2(\theta), \ldots, b_s(\theta)]^T$. Here we present two MIRK/CMIRK schemes, one of order 4, the other of order 6.

3.2. MIRK/CMIRK schemes of 4th order

The most general form of a 4th order CMIRK scheme is given in [Muir and Owren 1993]. It uses 4 stages and has free parameters c_3, c_4, and v_4. We will attempt to embed the optimal, 3-stage, 4th order discrete MIRK scheme within the CMIRK scheme. That is, the first 3 stages of the CMIRK scheme will be the same as those of the MIRK scheme, and the values of the continuous weight polynomials at $\theta = 1$ will be the weights of the discrete scheme. If we choose the free parameter $c_3 = \frac{1}{2}$ then the first three stages of this CMIRK scheme correspond to the unique 3-stage, 4th order MIRK, and the expressions for the 4th stage and the weight polynomials become more restricted. In Table 3.2 we give the tableau for this restricted CMIRK scheme.

Table 3.2

0	0	0	0	0	0
1	1	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{-1}{8}$	0	0
c_4	v_4	$-\frac{v_4}{6}-\frac{3{c_4}^2}{2}+c_4+\frac{2{c_4}^3}{3}$	$-\frac{{c_4}^2}{2}-\frac{v_4}{6}+\frac{2{c_4}^3}{3}$	$2{c_4}^2-\frac{4{c_4}^3}{3}-\frac{2v_4}{3}$	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$

where

$$b_1(\theta) = \frac{\theta\,(6\,c_4 - 9\,\theta c_4 - 3\,\theta + 6\,\theta^2 + 4\,\theta^2 c_4 - 3\,\theta^3)}{6\,c_4}, \quad b_2(\theta) = \frac{\theta^2\,(2\,\theta - 3\,c_4 + 4\,\theta c_4 - 3\,\theta^2)}{6(c_4 - 1)},$$

$$b_3(\theta) = -\frac{2\,\theta^2\,(4\,\theta c_4 - 6\,c_4 + 4\,\theta - 3\,\theta^2)}{3(2\,c_4 - 1)}, \quad b_4(\theta) = \frac{\theta^2\,(\theta - 1)^2}{2\,c_4\,(2\,c_4 - 1)\,(c_4 - 1)},$$

with the restrictions that $c_4 \neq 0, 1, \frac{1}{2}$. Also $b_1(1) = \dfrac{1}{6}$, $b_2(1) = \dfrac{1}{6}$, $b_3(1) = \dfrac{2}{3}$, and $b_4(1) = 0$.

A careful analysis to determine suitable optimization criteria based, for example, on minimizing the error coefficients of the dominant error terms followed by a thorough numerical investigation will be needed to determine optimal values for the remaining free parameters c_4 and v_4. Here, in order to exhibit a specific CMIRK scheme, we have arbitrarily chosen $c_4 = \frac{3}{4}$ and $v_4 = \frac{27}{32}$. With these choices we get the 4th order CMIRK scheme, given in Table 3.3.

Table 3.3

0	0	0	0	0	0
1	1	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{-1}{8}$	0	0
$\frac{3}{4}$	$\frac{27}{32}$	$\frac{3}{64}$	$\frac{-9}{64}$	0	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$

where

$$b_1(\theta) = -\frac{\theta\,(2\,\theta-3)\,(2\,\theta^2-3\,\theta+2)}{6}, \quad b_2(\theta) = \frac{\theta^2\,(9-20\,\theta+12\,\theta^2)}{6},$$

$$b_3(\theta) = \frac{2\,\theta^2\,(9-14\,\theta+6\,\theta^2)}{3}, \quad b_4(\theta) = -\frac{16\,\theta^2\,(\theta-1)^2}{3}.$$

This is the 4th order CMIRK scheme, with embedded MIRK scheme we have implemented.

3.3. MIRK/CMIRK schemes of 6th order

A particular family of 6th order CMIRK schemes is given in [Muir and Owren 1993]. Although the family has 9 free parameters $c_3, c_4, c_5,\ c_6, c_7, v_7,\ c_8, v_8, x_{81}$, it does not necessarily represent all possible 8-stage, 6th order CMIRK schemes. As in the previous subsection, our plan will be to attempt to construct a particular CMIRK scheme which contains a specific, 5-stage, 6th order discrete MIRK scheme as an embedded scheme, i.e. the tableau will contain the stages of the discrete scheme, and the weights evaluated at $\theta = 1$ are the weights of the discrete scheme. If we choose the free parameters $c_3 = \frac{1}{4}$, $c_4 = \frac{3}{4}$, and $c_5 = \frac{1}{2}$ this completely defines the first 5 rows of the CMIRK scheme and the expressions for the remaining 4 stages and the weight polynomials become more restricted. The remaining free parameters for this family are $c_6, c_7, c_8,\ v_7, v_8$, and x_{81}. We give the tableau for this CMIRK scheme in Section 1 of the Appendix. Unfortunately, this family of CMIRK schemes does not include the standard optimal 5-stage, 6th order discrete MIRK scheme, often quoted in the literature. (See, for example, [Van Bokhoven 1980], Method VIII.) The 5th stage of our CMIRK family is different than Method VIII, and more significantly, the 3rd and 4th weights of our CMIRK scheme are identically zero, while the corresponding weights of Method VIII are not.

We consider whether or not our CMIRK family contains a different optimal 5-stage, 6th order discrete MIRK scheme. (The 5-stage, 6th order MIRK scheme of [Van Bokhoven 1980] is easily shown to be non-unique.) The remaining free parameters are all associated with defining the extra stages (6 through 8) required by the CMIRK scheme. We arbitrarily choose $c_6 = \frac{1}{8}$, $c_7 = \frac{7}{8}$, $v_7 = \frac{3479}{4096}$ (chosen so that $x_{76} = 0$), $c_8 = v_8 = \frac{3}{8}$, and $x_{81} = \frac{5}{24}$. This gives a CMIRK scheme with the tableau given in Table 3.4.

Table 3.4

0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
$\frac{1}{4}$	$\frac{5}{32}$	$\frac{9}{64}$	$\frac{-3}{64}$	0	0	0	0	0	0
$\frac{3}{4}$	$\frac{27}{32}$	$\frac{3}{64}$	$\frac{-9}{64}$	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{24}$	$\frac{-1}{24}$	$\frac{1}{6}$	$\frac{-1}{6}$	0	0	0	0
$\frac{1}{8}$	$\frac{617}{4096}$	$\frac{1519}{24576}$	$\frac{-385}{24576}$	$\frac{49}{1536}$	$\frac{-49}{1536}$	$\frac{-147}{2048}$	0	0	0
$\frac{7}{8}$	$\frac{3479}{4096}$	$\frac{385}{24576}$	$\frac{-1519}{24576}$	$\frac{49}{1536}$	$\frac{-49}{1536}$	$\frac{147}{2048}$	0	0	0
$\frac{3}{8}$	$\frac{3}{8}$	$\frac{5}{24}$	$\frac{-271}{1344}$	$\frac{1147}{1920}$	$\frac{-1147}{1920}$	$\frac{-37}{576}$	$\frac{-9809}{20160}$	$\frac{1567}{2880}$	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$	$b_5(\theta)$	$b_6(\theta)$	$b_7(\theta)$	$b_8(\theta)$

where the weight polynomials are

$$b_1(\theta) = -\frac{\theta\,(4665\,\theta + 5120\,\theta^5 - 17664\,\theta^4 + 23400\,\theta^3 - 14900\,\theta^2 - 630)}{630},$$

$$b_2(\theta) = \frac{\theta^2\,(63 - 580\,\theta + 1800\,\theta^2 - 2304\,\theta^3 + 1024\,\theta^4)}{210}, \quad b_3(\theta) = 0, \quad b_4(\theta) = 0,$$

$$b_5(\theta) = \frac{2\,\theta^2\,(315 - 2690\,\theta + 7140\,\theta^2 - 7296\,\theta^3 + 2560\,\theta^4)}{135},$$

$$b_6(\theta) = \frac{32\,\theta^2\,(315 - 1430\,\theta + 2595\,\theta^2 - 2112\,\theta^3 + 640\,\theta^4)}{945},$$

$$b_7(\theta) = -\frac{16\,\theta^2\,(45 - 410\,\theta + 1245\,\theta^2 - 1536\,\theta^3 + 640\,\theta^4)}{945},$$

$$b_8(\theta) = -\frac{16\,\theta^2\,(128\,\theta^2 - 128\,\theta + 21)\,(\theta - 1)^2}{45}.$$

If we evaluate these polynomials at $\theta = 1$ we get a 6th order 7-*stage* discrete MIRK scheme with 5 non-zero weights, as follows,

$$b_1 = b_2 = \frac{1}{70}, \quad b_3 = b_4 = 0, \quad b_5 = \frac{58}{135}, \quad b_6 = b_7 = \frac{256}{945}, \quad b_8 = 0.$$

The above discussion demonstrates a fundamental contrast between the 4th and the 6th order cases. In the 4th order case, the number of stages needed to get a 4th order continuous extension of a given optimal discrete MIRK is the same as the optimal number of stages for a continuous CMIRK scheme of order 4, namely 4, and the discrete scheme can be embedded in the continuous family. This follows from the observation that the continuous family of CMIRK schemes of 4th order given in [Muir and Owren 1993] is complete. In the 6th order case, the optimal number of stages for a 6th order CMIRK scheme is 8. The particular CMIRK family exhibited in [Muir and Owren 1993] is based on two additional assumptions about the stage order of some of the stages of the schemes and thus is not likely to be complete. In fact it is substantially restrictive since it must have $b_3(\theta) = b_4(\theta) = 0$. It is then impossible for this family to include any optimal discrete MIRK scheme having only 5 stages, since there would then have to be only 3 stages with non-zero weights which from consideration of the quadrature conditions, we know is not possible. In the above derivation, the result is a discrete scheme with only 5 non-zero weights but requiring 7 stages.

For the optimal 5-stage, 6th order discrete scheme of [Van Bokhoven 1980], it appears, from an analysis of the continuous order conditions, that 4 additional stages must be added in order to produce a 6th order continuous scheme. That is, the continuous extension of the given optimal 5-stage, 6th order discrete scheme requires one more stage than that required by an optimal continuous CMIRK scheme of 6th order. We thus have the choice of an optimal 8-stage CMIRK scheme with a non-optimal embedded 7-stage, 6th order MIRK scheme, or a non-optimal 9-stage CMIRK scheme with an optimal embedded 5-stage, 6th order MIRK scheme. Because an important factor in the computational costs is the number of stages in the discrete MIRK scheme, we choose the latter option for our implementation. The above discussion suggests the need for a theoretical investigation of optimal CMIRK schemes containing embedded optimal discrete MIRK schemes. We plan to study this question in a subsequent investigation.

In Table 3.5, we present a 9-stage, 6th order CMIRK scheme, which has the above mentioned 5-stage, 6th order discrete MIRK scheme embedded within it. The first 5 stages of the CMIRK scheme are the stages of this MIRK scheme. The first 5 weight polynomials evaluated at $\theta = 1$ are the weights of this MIRK scheme, while the last 4 weight polynomials are 0 when evaluated at $\theta = 1$.

Table 3.5

$$
\begin{array}{c|c|ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{5}{32} & \frac{9}{64} & \frac{-3}{64} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{3}{4} & \frac{27}{32} & \frac{3}{64} & \frac{-9}{64} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & \frac{5}{24} & \frac{-5}{24} & \frac{2}{3} & \frac{-2}{3} & 0 & 0 & 0 & 0 & 0 \\
\frac{7}{16} & \frac{7}{16} & \frac{1547}{32768} & \frac{-1225}{32768} & \frac{749}{4096} & \frac{-287}{2048} & \frac{-861}{16384} & 0 & 0 & 0 & 0 \\
\frac{1}{8} & \frac{1}{8} & \frac{83}{1536} & \frac{-13}{384} & \frac{283}{1536} & \frac{-167}{1536} & \frac{-49}{512} & 0 & 0 & 0 & 0 \\
\frac{9}{16} & \frac{9}{16} & \frac{1225}{32768} & \frac{-1547}{32768} & \frac{287}{2048} & \frac{-749}{4096} & \frac{861}{16384} & 0 & 0 & 0 & 0 \\
\frac{3}{8} & \frac{3}{8} & \frac{233}{3456} & \frac{-19}{1152} & 0 & 0 & 0 & \frac{-5}{72} & \frac{7}{72} & \frac{-17}{216} & 0 \\
\hline
 & & b_1(\theta) & b_2(\theta) & b_3(\theta) & b_4(\theta) & b_5(\theta) & b_6(\theta) & b_7(\theta) & b_8(\theta) & b_9(\theta)
\end{array}
$$

where

$$b_1(\theta) = \frac{\theta\,(334530 - 1287315\,\theta - 1662100\,\theta^2 + 10043400\,\theta^3 - 11467776\,\theta^4 + 4065280\,\theta^5)}{334530},$$

$$b_2(\theta) = \frac{\theta^2\,(3083913 - 25341580\,\theta + 62657400\,\theta^2 - 62874624\,\theta^3 + 22657024\,\theta^4)}{2341710},$$

$$b_3(\theta) = -\frac{16\,\theta^2\,(8505 - 69430\,\theta + 168780\,\theta^2 - 164352\,\theta^3 + 56320\,\theta^4)}{7965},$$

$$b_4(\theta) = -\frac{16\,\theta^2\,(8505 - 69430\,\theta + 168780\,\theta^2 - 164352\,\theta^3 + 56320\,\theta^4)}{7965},$$

$$b_5(\theta) = -\frac{2\,\theta^2\,(8505 - 69430\,\theta + 168780\,\theta^2 - 164352\,\theta^3 + 56320\,\theta^4)}{2655},$$

$$b_6(\theta) = -\frac{1024\,\theta^2\,(45568\,\theta^2 - 45856\,\theta + 7533)\,(\theta - 1)^2}{33453},$$

$$b_7(\theta) = \frac{128\,\theta^2\,(128\,\theta^2 - 128\,\theta + 21)\,(\theta - 1)^2}{15},$$

$$b_8(\theta) = \frac{1024\,\theta^2\,(134144\,\theta^2 - 132128\,\theta + 21609)\,(\theta - 1)^2}{234171}, \quad b_9(\theta) = -\frac{16384\,\theta^3\,(\theta - 1)^3}{441}.$$

Evaluating these weight polynomials at $\theta = 1$ gives

$$b_1(1) = b_2(1) = \frac{7}{90}, b_3(1) = b_4(1) = \frac{32}{90}, b_5(1) = \frac{12}{90}, b_6(1) = b_7(1) = b_8(1) = b_9(1) = 0.$$

These 5 non-zero weights plus the first 5 stages from Table 3.5 define the optimal 5-stage, 6th order MIRK scheme, embedded within the 9-stage, 6th order CMIRK scheme, which we have implemented.

4. Practical Aspects of MIRK Schemes in a Defect-Controlled BVODE Code

In this section we discuss the development, implementation, and analysis of algorithms for various parts of a software package for the numerical solution of BVODEs, based on MIRK schemes and defect control, for which the general algorithm was given in Section 2. We will include a presentation of the results of an exploration of several of the heuristic strategies employed within the code. We will discuss the modified Newton iteration used to solve the non-linear systems arising from the discretization of the ODE system, the details of the implementation of defect control (including estimation of the defect through defect sampling), and the results of our investigation of some of the controlling parameters in the mesh refinement and redistribution algorithm.

4.1. The Modified Newton Iteration

4.1.1. Damped Newton and fixed Jacobian iterations

To solve the system, $\Phi(Y) = 0$, described in section 2, we have employed a modified Newton iteration, modelled on that discussed in [Ascher, Mattheij, and Russell 1988, Chap. 8]. This includes the implementation of a damped Newton iteration, and a fixed Jacobian iteration, as well as a scheme for switching between these two modes of iteration. The switching scheme involves monitoring the rate of decrease of a function known as the "natural criterion function", $||J^{-1}(Y^{(q)})\Phi(Y^{(q+1)})||$, where $Y^{(q)}$ is the qth iterate, $\Phi(Y)$ is the residual function defined in section 2, and $J(Y)$ is the Jacobian of $\Phi(Y)$. The code begins with a full Newton step and continues with damped Newton steps until it has taken a step with the damping factor equal to 1 (i.e. a full Newton step) and there has been a sufficient decrease in the size of the natural criterion function. When these conditions arise the code switches to fixed Jacobian steps, continuing these as long as sufficient decreases in the value of the natural criterion function continue. Our numerical experiments indicate that the computational costs of fixed Jacobian steps are relatively negligible compared to the costs of setting up and factoring the full Jacobian. Thus we have found it worthwhile to be fairly liberal in allowing the code to attempt to take fixed Jacobian steps.

In addition to monitoring the size of the natural criterion function to check for a sufficient rate of convergence, we have also found it useful to use the size of the natural criterion function to allow early termination of divergent or slowly converging iterations. When the natural criterion function value exceeds a certain threshold value this usually signifies that the corresponding Newton correction is also large, and we terminate the Newton iteration since a large Newton correction indicates that the current iterate is not close to the true solution of the current discrete system. In the context of solving BVODE problems, it may be preferable to employ the option of restarting with a new discrete system based on a finer mesh. An open question here is to determine a suitable value for the threshold value bounding the size of the natural criterion function values. A somewhat related point concerns determination of the appropriate upper limit on the maximum number of iterations that should be allowed for each Newton iteration. Even if the natural criterion function values remain bounded the rate of convergence may be sufficiently slow that convergence will not be achieved for many steps. Also divergent iterations can be detected in this way. As mentioned above, a reasonable strategy may be to set the maximum number of iterations limit fairly low, with the idea that it would be better to start over on a finer mesh than to continue on the current mesh.

An additional consideration, mentioned in [AMR 1988], is to have the size of the system contribute to deciding when to switch to a fixed Jacobian iteration: for smaller systems which usually arise during the initial iterations (where the mesh is coarse and may not be well adapted to the solution behavior) it may be better to simply employ full or damped Newton iterations, and avoid any consideration of fixed Jacobian steps until the systems become larger and more substantial savings can be expected. We have not implemented this idea, but are considering it for further testing to evaluate its importance. Another interesting aspect of the efficient implementation of this algorithm has been the need for careful data management to ensure that no duplicate calculations are performed. Some of the quantities computed by the natural criterion function to assess the quality of the most recent iterate can be used in the next Newton step, even when the current step is rejected.

We conducted a numerical evaluation of the damped Newton iteration compared with a simpler approach where a full Newton iteration was used for every step. As expected, for very difficult discrete systems where we used poor initial estimates of the solution, the damped Newton iteration achieved convergence in cases where the full Newton iteration did not. This usually occurred during the early part of the execution of the code where the mesh was coarse and the solution information from the previous mesh was nonexistent or not relevant. However for somewhat less difficult discrete systems, again on coarse meshes with poor initial information for the solution, the full Newton iteration sometimes achieved convergence when the damped iteration gave up too soon. Thus it is not clear that the damped Newton iteration will *always* behave in a superior fashion. Furthermore, when solving the nonlinear systems arising in BVODEs, it may not be a good strategy to employ a damped Newton iteration and devote substantial computational effort to obtaining a converged solution to a discrete system derived from a coarse mesh. Rather, it may be better to just restart on a finer mesh where convergence may be obtained more quickly. Of course, as is noted in [Ascher, Mattheij, and Russell, 1988], the primary benefit of the damped Newton iteration in this context may be that it allows early termination of a divergent or too slowly converging iteration.

4.1.2. Approximate Jacobian evaluations

Here we make note of an important point associated with the evaluation of the Jacobian matrix arising during the damped Newton iteration. The Jacobian, $\frac{\partial \Phi}{\partial Y}$, as described in section 2, has an almost block diagonal structure, with blocks based on the partial derivatives of the components of Φ associated with each subinterval. The expressions for the blocks are given in (2.3) and (2.4). The literature for MIRK schemes for both initial and boundary value ODEs suggests that (2.4) can be approximated by evaluating $\frac{\partial f}{\partial y}$ at only one point on each subinterval, with this one value being used in place of each instance of J_r arising in (2.4). We implemented this suggestion in the module which constructs the Jacobian blocks, and compared the performance with that obtained using a similar module where s Jacobian evaluations were done for each subinterval, as in (2.4). We found major differences in performance between the two approaches, especially when applied to difficult nonlinear problems. When using single Jacobian evaluations per subinterval approach, we found some savings in the cost of computing the Jacobian blocks, but the resultant degradation in the performance of the Newton iteration more than offset these savings. Not only was the rate of convergence for the single Jacobian evaluation approach much slower, but also the likelyhood of achieving convergence was much lower as well. This suggests that on some problems it is critical to incur the extra cost of using exact Jacobian values on each subinterval, in order to achieve a lower overall cost.

4.1.3. Solution and mesh continuation over the discrete problem sequence

We conducted a set of experiments in which we investigated the standard approach used in the numerical solution of boundary value problems in which the code begins with a coarse mesh and possibly poor solution approximation, sets up and solves a discrete system based on this information, and then proceeds to construct a more appropriate mesh and estimate of the solution for the next discrete system, (based on the solution information from the current step). When the code begins execution under such circumstances, it was often difficult to obtain convergence of the Newton iteration. Furthermore, in those instances where convergence was obtained on the initial mesh (after many iterations, and possibly using a sequence of damped steps), and the resulting continuous solution used for the initial guess for the next discrete system (based on a mesh obtained by halving the previous one), we often observed that the Newton iteration for the new discrete problem would often not converge at all. In fact, we sometimes observed that we could get better convergence results by ignoring the computed solution from the previous coarse mesh, and simply providing the original initial solution approximation for the refined mesh. The difficulty here is that the solution to the discrete system on a coarse mesh may not be closely related to the solution for the discrete system on the refined mesh, so it is not particularly helpful to provide the computed solution from the coarse system as the initial solution estimate for the refined system. The implication for strategies for monitoring the Newton iteration is that, if we observe convergence difficulty for a discrete system on a coarse mesh, it may not be worthwhile to invest much computational effort in finding a solution to this system. A better strategy might be to simply "fail" on that coarse system, and restart with a refined mesh and a larger discrete system. We found that by monitoring the size of the defect we could generally detect whether the solution from a coarse discrete system would be useful as the initial solution estimate for the next refined system. A large defect signaled that the solution to the coarse system would probably not be useful. We are presently conducting further investigation of an optimal choice for the maximum size of an acceptable defect.

A related consideration here is that the appropriate threshold value for the size of the defect of the numerical solution may be different depending on how we intend to use the defect. We have found that in some cases, where the defect is of intermediate size, it may be worthwhile to use the defect estimates for each subinterval as input for the mesh redistribution algorithm but it is not appropriate to use the corresponding numerical solution as the initial guess for the Newton iterate for the discrete system derived from the new mesh. This suggests that two thresholds may be needed, one for acceptance of the defect for use in the mesh redistribution process, and another for acceptance of the numerical solution for use as the initial Newton iterate. We are currently investigating the determination of appropriate threshold values.

4.1.4. The tolerance for the Newton iteration

A final issue in this area is to determine the appropriate size for the tolerance to be applied to the Newton correction vector at each step of the Newton iteration in order to determine convergence of the iteration. It is clear that the Newton tolerance should be less than the user-defined tolerance in order to ensure that the discretization errors will dominate in the determination of the defect, and thus in the mesh redistribution process. We have found through numerical experiment that setting the Newton tolerance to be $\frac{1}{100}$ that of the user-defined tolerance removes any effects of the error in the Newton iterate from the rest of the calculation.

Another aspect of this issue concerns the use of a blended relative/absolute tolerance rather than an absolute tolerance for the Newton correction. We have been able to achieve slight improvements in the efficiency of the Newton iteration, with no negative performance effects, by using the following relative/absolute tolerance on the Newton corrections. Denote the j-th component of the approximate solution, y_i at t_i by $y_{i,j}$ and the corresponding Newton correction by $\Delta y_{i,j}$. Then we apply the Newton tolerance, *newtol* as follows,

$$|\Delta y_{i,j}| \leq (|y_{i,j}| + 1)\ newtol.$$

When some of the solution values are relatively large, this is a much more reasonable termination criterion than the usual absolute tolerance, and leads to significant savings in the overall computation. This is the same criterion that is implemented in COLSYS and COLNEW, as well.

4.2. Defect Estimation

The defect, as given in section 2, is a continuous function over the problem interval and gives us a measure of the quality of the computed solution. For our termination criterion, we would like to know the maximum value of the defect over the entire problem interval. It should be noted that the defect estimate can be computed for a given solution on a given mesh and thus termination can occur without having to compare the solution with that obtained on another mesh. This can be contrasted with the scheme used in COLSYS where termination occurs only when the current solution can be compared with the previously computed solution obtained on a mesh which is exactly half the current one. (That is, the current mesh is obtained by halving each subinterval of the previous mesh.) The defect is also used for mesh refinement and redistribution purposes where it is important to know the maximum size of the defect over each subinterval. We will discuss mesh selection in the next subsection.

To avoid excessive computational costs, we would like to be able to obtain an estimate of the maximum defect on each subinterval by sampling the defect at only a few points. In an analogous setting, COLSYS samples the global error at 2 points per subinterval and it has been suggested in [Ascher and Jacobs 1989] that sampling at the s collocation points of each subinterval can be advantageous. In the initial value ODE setting, defect estimation is typically done using 1 sample point per step. In order to implement the 1-point sampling scheme for defect estimation, in our BVODE software, we have employed the following approach to determine where that sample point should be, based on the ideas in [Enright 1991]. Since the continuous solution approximations we use are based on continuous Runge-Kutta schemes (3.1),(3.2), it is possible to express the defect in terms of the coefficients of these schemes. On the ith subinterval, the defect is denoted by $\delta_i(t)$, and is given by

$$\delta_i(t) = \left(\sum_{j=1}^{N_p} {q_j}'(\theta) F_j \right) h_i^p + O(h_i^{p+1}),$$

where p is the order of the Runge-Kutta scheme, N_p is the number of elementary differentials, F_j, (related to the number of rooted trees with p nodes, see [Butcher 1987]), and the $q_j(\theta)$'s are polynomials in θ whose coefficients depend on the coefficients and weight polynomials of the continuous Runge-Kutta scheme. Thus the size of the defect is determined by the coefficient of the h_i^p term in the above expression, for h_i sufficiently small. We introduce a further approximation by considering only the term that remains when linear ODE problems are considered. In this case the only non-zero $q_j(\theta)$, which we write without a subscript, is

$$q(\theta) = b^T(\theta)(X + vb(1)^T)^{p-2}c - \frac{\theta^p}{p!}.$$

For a particular continuous MIRK scheme we can examine the behavior of $q'(\theta)$ over $0 \le \theta \le 1$ and determine where it reaches it maximum. We will then choose the defect sample point, θ^*, at this location. For the 4th order continuous MIRK scheme given in the previous section, the above analysis led us to choose $\theta^* \approx 0.226$. For the 6th order CMIRK given at the end of the previous section, we found that $\theta^* \approx 0.716$. The estimate of the maximum defect for each subinterval is defined to be the max-norm of the defect vector (of length n) generated by sampling the exact defect expression at this single sample point. We will refer to this as the *absolute* defect estimate.

As we saw in the previous subsection for Newton iterations, it is more practical to consider using a relative/absolute definition of the defect for use in termination criteria and in mesh redistribution. The use of the *scaled* defect, $\bar{\delta}_i(t)$, on each subinterval, defined by,

$$\bar{\delta}_i(t) \equiv \frac{\delta_i(t)}{(|f(t, u(t))| + 1)}$$

has proven, in our numerical experiments, to lead to substantial savings in the computational costs. Since the defect is a perturbation of the original ODE system, it is clearly more appropriate to consider the size of the defect relative to the size of f, when f is large.

We performed numerical experiments to determine the reliability of the above approach to defect sampling. Using the family of test problems, Swirling Flow III, [Ascher, Mattheij, Russell, 1980, Pg. 23], (see the Appendix, Section 2, for the corrected form of this problem), we assessed the reliability of three defect estimation schemes. We considered several problems from this family, each characterized by a particular value of the parameter value, *epsilon*. We considered a sequence of meshes and used a tolerance of 10^{-6}. We examined the reliability of estimation of the maximum absolute defect using 1-point sampling, estimation of the maximum scaled defect using 1-point sampling, and estimation of the maximum scaled defect using 2-point sampling, with the second sample point chosen to be $1 - \theta^*$. We compared the estimate for each subinterval with a "true" value obtained by sampling the defect at 11 points on the subinterval to get a good estimate of the true maximum value. We computed the ratio of the estimated maximum to the true maximum and defined a deception to occur when the estimate of the maximum was less than $\frac{1}{5}$ of the true maximum. We then computed the percentage of deceptions with respect to the total number of estimates performed over all discrete problems considered by the code in arriving at the final solution. We used the BVODE code to solve each problem beginning with an initial uniform mesh of 5 subintervals, and and initial guess consisting of straight lines through the boundary conditions. The code then worked through a sequence of discrete problems, choosing new meshes based on defect information, until it obtained a solution with a satisfactory defect or until it found it could not find a solution using fewer than the maximum number of subintervals we allowed, namely 300. For each discrete problem for which a solution was obtained, the code computed defect estimates according to one of the above schemes: absolute defect with 1-point sampling, scaled defect with 1-point sampling, or scaled defect with 2-point sampling. (Note that when applied as the termination criterion, the absolute defect requirement is more severe than the scaled defect requirement, so a BVODE code would have to work much harder in the former case. The point of this particular experiment is to examine the reliability of the estimation schemes and rather than compare their efficiency within the BVODE software.) We used the defect to control both termination and mesh redistribution. In Tables 4.2.1 and 4.2.2 we report the percentage of deceptions for the the 4th and 6th order MIRK/CMIRK schemes, for each of the three estimation schemes which we will identify as follows: I: Absolute Defect, 1-point sampling; II: Scaled Defect, 1-point sampling; III: Scaled Defect, 2-point sampling. Also, note that a '*' indicates that the problem could not be solved using within a limit of 300 subintervals; we will refer to this as a failure.

Table 4.2.1 : 4th order MIRK/CMIRK scheme : Percentage of Deceptions

$\epsilon \Rightarrow$	0.01	0.005	0.001	0.0005	0.0001	Average
I	8.5	4.7	$*$	$*$	$*$	6.3
II	17.6	11.8	2.3	0.6	$*$	4.5
III	0.3	0.9	0.8	0.3	$*$	0.6

Table 4.2.2 : 6th order MIRK/CMIRK scheme : Percentage of Deceptions

$\epsilon \Rightarrow$	0.01	0.005	0.001	0.0005	0.0001	Average
I	0.0	0.0	0.8	0.2	$*$	0.3
II	0.0	0.8	2.4	0.8	0.8	1.0
III	0.0	0.0	1.2	1.2	0.2	0.5

It is clear from Tables 4.2.1 and 4.2.2, that 2-point sampling can yield substantial improvements over 1-point sampling, particularly in the 4th order case. It is worthwhile considering how much extra this more robust scheme costs in terms of efficiency. The specific cost of 2-point sampling is about twice that of one point sampling but the contribution to the overall cost is less clear. We performed some timings of schemes II and III for the 4th and 6th order MIRK/CMIRK schemes on a SPARC station IPC. The results are given in Table 4.2.3. The table entry is the overall cost for solving the problem in seconds.

Table 4.2.3 : Execution Times for Defect Sampling Schemes

$\epsilon \Rightarrow$	0.01	0.005	0.001	0.0005	0.0001	Average
II, 4th	5.9	9.7	26.2	31.2	*	73.0
III, 4th	6.2	10.7	24.2	23.8	*	64.9
II, 6th	4.5	6.9	12.8	21.2	81.1	126.5
III, 6th	5.2	8.1	13.4	13.9	110.4	151.0

While computing the times for the above table, we also monitored the execution times specifically associated with the defect estimation module and found, as expected, that the execution times for that module were approximately doubled when 2-point sampling was used instead of 1-point sampling. It is interesting to note, in Table 4.2.3, however, that the overall execution costs for the BVODE code are roughly the same. This occurs because the availability of a sharper estimate of the defect profile over the mesh subintervals allows the mesh redistribution algorithm to obtain meshes that are better suited to the underlying solution behavior, thus leading to a final solution more quickly.

In conclusion, 1-point sampling for a scaled defect provides a reasonably robust estimation scheme. Sampling at 2 points provides substantially more reliability, and overall, the costs are not substantially increased.

4.3. Mesh Selection

We refer the reader to [AMR 1988, Chap. 9] for an overview of mesh selection algorithms for boundary value codes. The mesh selection algorithm we have implemented is modelled on that given in [AMR 1998, Pg 369-70] but it uses scaled defect estimates rather than global error estimates as the basis for the equidistribution process. The mesh selection process has as its kernel a mesh redistribution algorithm which, when given the defect estimate for each subinterval of the current mesh, produces a new mesh which approximately equidistributes that estimate over the subintervals of the new mesh. The other major part of the mesh selection process is an algorithm which examines the degree of non-uniformity of the defect estimate over the subintervals of the current mesh to establish whether or not a redistribution of the mesh points could be expected to be of much help. Both these strategies depend on some parameters whose values can have a significant effect on the overall performance of the code.

One of these parameters, which we will call ρ, is used to monitor the ratio of the maximum to the average value of a quantity which depends on the relative size of the scaled defect,

$$\frac{\max_i h_i \left(\frac{|\bar{\delta}_i|}{TOL}\right)^{\frac{1}{p}}}{\frac{1}{N}\sum_{i=1}^{N} h_i \left(\frac{|\bar{\delta}_i|}{TOL}\right)^{\frac{1}{p}}}, \tag{4.1}$$

where p is the order of the Runge-Kutta scheme, TOL is the user defined tolerance, δ_i is the defect estimate for the ith subinterval, and h_i is the width of the ith subinterval. Mesh refinement is performed only if the above ratio is greater than ρ. When the ratio is less than or equal to ρ, the new mesh is obtained simply by halving each subinterval of the current mesh. In COLSYS, the value used for ρ is 2. This means that if all the subintervals are about the same size then the maximum value of this quantity must be 2^p times the average value before a redistribution is performed. The result is that for COLSYS, redistributions are infrequent. We have profiled our code and noted that the execution costs associated with the mesh refinement algorithm are small compared to some other parts of the code, so efficiency concerns over execution costs directly associated with the mesh selection modules are not particularly relevant, and redistribution can be considered more often, if desired. Later in this section we will present some results describing the values we considered for this parameter.

Another parameter that arises within the mesh selection algorithm predicts the number of points a new mesh would need in order to lead to a solution having a defect satisfying the user-requested tolerance. We have performed some numerical investigations of this parameter and have found that it usually provides good predictions for the number of points which will be needed in the final mesh. A difficulty with the approach described here is that the mesh redistribution is only invoked if the distribution of the defect estimates is sufficiently non-uniform. That is, if the distribution is fairly uniform, the mesh redistribution algorithm is never invoked. This is unfortunate because there are times when the distribution of mesh points over the problem interval is satisfactory but the resultant solution is not satisfactory simply because the mesh is not fine enough to lead to a solution which meets the tolerance. We would like to be able to switch to a mesh with the same distribution but with more points. We can control the frequency with which mesh redistribution takes place by choosing a smaller value for ρ. For example, choosing $\rho = 1$ implies that the mesh will be redistributed with possibly new points added or taken away, every time the check is performed.

We have also experimented with applying a "safety factor" to the parameter determining the predicted number of points for a new mesh. We multiply the prediction parameter by a safety factor slightly bigger than 1 with the reasoning that it would be preferable to perform the next set of computations using a slightly larger number of meshpoints than necessary rather than fail on the current mesh because the number of meshpoints was slightly too small to provide the required accuracy. Shortly, we will present some results comparing the performances associated with several choices of the safety factor parameter.

The final two parameters control the width of the window of possible values for the new number of mesh points, compared to the current number of mesh points. In COLSYS the number of points in the new mesh can be no greater than the number of points in the current mesh, and no smaller than half the number of points in the current mesh. We have experimented with the factors controlling how much smaller or larger the number of new meshpoints should be allowed to change with respect to the old mesh, and report the results below.

In the Table 4.3.1, we summarize the results of our investigations of these parameters. We again used the Swirling Flow III problem family from [AMR 1988, Pg. 23], with a tolerance value of 10^{-6}. The 6th order MIRK/CMIRK scheme was used. A uniform mesh of 5 subintervals was provided as the initial mesh for each problem. The code worked through a sequence of discrete problems based on meshes determined using the mesh redistribution algorithm for each of the mesh algorithm parameters. Termination occured when a solution with a scaled defect satisfying the tolerance was obtained, or when the code determined that more than 300 subintervals would be needed to obtain a solution.

The process for computing the solution for a particular problem usually requires the code to solve a sequence of discrete problems. For each problem, we noted the number of meshes used, the size of each, and the number of Newton iterations performed to get convergence. One important unit of computational cost is the work per subinterval involved in setting up and factoring the Newton systems. Thus, in Table 4.3.1, for each value of ϵ, and each set of mesh algorithm parameter choices, we record the total number of subinterval computations treated by the code. This cost measure is $\sum_{j=1}^{q} N_j I_j$, where q is the number of different meshes used to solve the problem, N_j is the number of mesh intervals associated with the jth mesh, and I_j is the number of full or damped Newton iterations (i.e. Jacobian setup and factorizations) performed on the discrete problem derived from that mesh. The safety factor is sf and w is the window for the new mesh sizes. If the current mesh has N subintervals, then $w = [\alpha, \beta]$ means the new mesh must have at least αN subintervals, and at most βN subintervals. We will define a failure to occur when a given problem cannot be solved using no more than a maximum number of subintervals, chosen for this experiment to be 300. In the event of a failure, we doubled the number of subintervals in the final mesh and contributed this value to the overall cost for that combination of mesh algorithm parameters. We have summed up the above cost measures over all discrete problems and ϵ values in order to arrive at an overall cost for each parameter combination.

Table 4.3.1 : Number of subinterval computations per problem

$sf \Downarrow \;\; \rho, w \Rightarrow$	$2.0, [\frac{1}{2}, 2]$	$2.0, [\frac{1}{2}, 4]$	$1.5, [\frac{1}{2}, 2]$	$1.5, [\frac{1}{2}, 4]$	$1.0, [\frac{1}{2}, 2]$	$1.0, [\frac{1}{2}, 4]$
1.1	5999	5999	5856	5725	5462	5373
1.2	5989	5998	6302	6209	6033	5373
1.3	5944	5974	5672	5847	5459	5427

We see from this table that savings in overall costs of about 10% are possible for particular choices of the mesh parameters. In particular, choosing $\rho = 1.0$ and $w = [\frac{1}{2}, 4]$ appears to give the best results overall, independent of the choice of safety factor. The results for $\rho = 1.0$ and $w = [\frac{1}{2}, 2]$ with $sf = 1.1$ or 1.3, are not much worse.

Another important consideration here is the frequency of occurences of failure. There are no failures in the above table for any problem with $\rho = 1.5$ or 1.0, while 7 failures occurred over the two cases with $\rho = 2.0$. This latter choice for ρ severely restricts the ability of the code to adapt the mesh to the solution profile because the mesh redistribution algorithm is invoked only when the defect profile is very non-uniform. In contrast, the choice of $\rho = 1.0$ forces the code to perform a mesh redistribution after every solved discrete problem, even when the defect profile is not particularly non-uniform. This allows the code to introduce more points into a mesh in order to meet accuracy requirements, more quickly than is possible if only mesh doubling is allowed.

5. Overall performance analysis

5.1. Overview

In this section we present some results on the performance of our BVODE solver which use MIRK formulas with Defect Control, and which we will call MIRKDC. We will present a preliminary demonstration of the performance of this code on a two families of test problems chosen from the literature. We will also present performance results for the COLNEW code. It is not the purpose of this paper to compare these codes. Comparison of boundary value software is very difficult under any circumstances. The fact that these two codes are attempting to produce solutions that satisfy different criterion (i.e. defect control vs. global error control) makes a detailed comparison impossible. Also, COLNEW is able to handle directly mixed order BVODE systems, while MIRKDC, because it uses standard Runge-Kutta formulas, can only handle first order BVODE systems. Our point in presenting results from COLNEW is to demonstrate that the performance exhibited by our code MIRKDC is comparable to that of state-of-the-art software in this area.

We shall assess the performance of a code according to several criteria. For each test problem, we will first present what we will call a "performance profile". This profile will trace the sequence of discrete problems that each code considered in its attempt to arrive at a final solution. The performance profile will consist of a list of ordered pairs, (n, q), one pair for each discrete system, where n is the number of subintervals in the mesh from which the discrete system was derived, and q is the number of full Newton steps (also the number of Jacobian evaluations and factorizations). Such a profile gives a concise yet reliable indication of the difficulty encountered by the code in attempting to solve the given problem. In addition to the profile, our assessment of the performance of each code on a given problem will include the CPU time (obtained on a Micro-VAX 3800), the global error of the computed solution (measured in a relative/absolute sense), and the defect of the solution (again measured in a relative/absolute sense, as explained in section 4.2). The global error will be obtained by comparing the computed solution, SOL_C, with a highly accurate numerical solution SOL_{ACC}, obtained by COLNEW. The relative/absolute global error, GE_{REL} is given by

$$GE_{REL} \equiv \frac{||SOL_C - SOL_{ACC}||}{(||SOL_{ACC}|| + 1)}.$$

In order to have the codes perform approximately the same task for each test problem, we have chosen to ask each code to return a solution for which the global error (in the relative/absolute sense mentioned above) is within a certain tolerance. Since COLNEW is designed to control the global error in this relative/absolute sense, it is straightforward to make this specification to COLNEW - one simply sets the tolerances for each solution component. On the other hand MIRKDC controls the defect (in a relative/absolute sense) so a slightly different strategy is employed. For each test problem, using MIRKDC, we select a constant tolerance to be applied to the components of the defect that will yield a solution whose global error in a relative/absolute sense is within the desired tolerance. The tolerance that must be applied for the defect will usually not be the same as the tolerance for the global error, and the relationship between these values is problem and method dependent. For each test problem we experimentally determined a tolerance for the defect that would lead to a solution whose global error satisfied the requested global error tolerance. Thus the codes will both produce solutions with global errors satisfying a given tolerance but the manner in which they do this will be fundamentally different.

Since MIRKDC can only handle first order systems, our set of test problems will be presented as first order systems. The importance of this restriction has not been fully discussed in the current literature. The ability of a code to handle mixed order systems directly has fundamental implications for the efficiency of that code on such problems. Whenever a mixed order system must be converted to a first order system, the result is a substantial increase in the sizes of the blocks of the almost block diagonal systems arising in the Newton matrices. Since the factorization costs for these matricies depend on the cube of the size of these blocks, any increase in this size can lead to substantial increases in the computational cost of a BVODE solver employing them. Although the larger blocks in the almost block diagonal systems arising from the conversion of a mixed order system to a first order system do exhibit some structure which perhaps could be exploited by some specially designed factorization software, no such software currently exists, and BVODE codes rely upon more general purpose almost block diagonal system software to perform the factorizations. This is an important area for future work in the use of Runge-Kutta methods for BVODEs. In the Appendix, Section 3, we give the results of a numerical experiment in which we compare the performance of COLNEW on mixed and first order system versions of a BVODE system to demonstrate the significant differences. As an alternative to designing special purpose software to handle the Jacobian blocks from first order system, an alternative approach which we plan to investigate involves developing extensions of the MIRK and CMIRK formulas for the direct treatment of higher order ODE systems (analogous to the situation with initial value ODE problems where Runge-Kutta-Nystrom formulas have been available for some time.)

5.2. Numerical results for problem 1

The first problem for which we present results is the Swirling Flow III problem from [AMR 1988, pg. 23], (with the correction as noted in the Appendix, Section 2) which we used as a test problem family in section 4. We consider two tolerance values. Our goal here is to have both codes produce solutions which, in the usual relative/absolute sense, have global errors of less than 10^{-6} or 10^{-8}. Thus for each set of tests, we simply provided COLNEW with a tolerance of 10^{-6} and 10^{-8} for each component. For MIRKDC, for each problem (i.e. value of ϵ) we list the tolerances values that were supplied in order to obtain solutions with the desired global errors. We provide results for both 4th and 6th order methods. A '*' indicates that the Newton iteration for a discrete problem did not converge. A failure is indicated (by the entry $FAIL$) if a code signaled that it required more than 2000 subintervals to solve the problem. For MIRKDC, we use 2-point defect sampling, and set the mesh selection parameters of section 4.3 as $\rho = 1.0$, $sf = 1.3$, $w = [\frac{1}{2}, 4]$. Also, the threshold strategies for monitoring the size of the defect, as discussed in Section 4, are not used here since the appropriate values for the thresholds are currently under investigation.

Table 5.2.1 : COLNEW on Problem 1 : 4th order, tol $= 10^{-6}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	$(5,2)$	$(5,2)$	$(5,3)$	$(5,*)$	$(5,*)$	$(5,*)(160,1)$	$(5,*)(110,1)$
	$(10,1)$	$(10,1)$	$(10,1)$	$(10,5)$	$(10,6)$	$(10,9)(320,1)$	$(10,*)(220,1)$
		$(20,1)$	$(20,1)$	$(20,1)$	$(10,1)$	$(10,2)$	$(20,*)(440,1)$
		$(40,1)$	$(40,1)$	$(40,1)$	$(20,1)$	$(20,2)$	$(40,10)$
			$(80,1)$	$(80,1)$	$(40,1)$	$(20,1)$	$(40,1)$
				$(58,1)$	$(80,1)$	$(40,1)$	$(40,1)$
				$(116,1)$	$(71,1)$	$(40,1)$	$(80,1)$
					$(142,1)$	$(80,1)$	$(160,1)$
Time(sec)	0.078	0.313	0.645	1.415	1.697	2.963	6.869
Error	$8.3 \cdot 10^{-7}$	$2.3 \cdot 10^{-7}$	$6.2 \cdot 10^{-8}$	$1.6 \cdot 10^{-7}$	$1.8 \cdot 10^{-7}$	$1.2 \cdot 10^{-7}$	$2.4 \cdot 10^{-8}$
Defect	$1.0 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$	$9.4 \cdot 10^{-4}$	$2.0 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$	$2.1 \cdot 10^{-3}$

Table 5.2.2 : MIRKDC on Problem 1 : 4th order, error $= 10^{-6}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
$tol \Rightarrow$	$6 \cdot 10^{-7}$	$8 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	$4 \cdot 10^{-6}$	$3 \cdot 10^{-7}$
Profile	$(5,1)$	$(5,1)$	$(5,1)$	$(5,2)$	$(5,*)$	$(5,*)$	$(5,5)$
	$(16,1)$	$(20,1)$	$(20,1)$	$(20,2)$	$(10,2)$	$(10,4)$	$(20,*)$
	$(21,1)$	$(29,1)$	$(45,1)$	$(80,1)$	$(40,1)$	$(40,*)$	$(40,5)$
			$52,1)$	$(111,1)$	$(104,1)$	$(80,4)$	$(160,1)$
					$(125,1)$	$(153,1)$	$(323,1)$
						$(175,1)$	$(373,1)$
						$(192,1)$	
Time(sec)	0.059	0.078	0.169	0.358	0.468	1.468	1.667
Error	$3.7 \cdot 10^{-7}$	$9.6 \cdot 10^{-7}$	$7.6 \cdot 10^{-7}$	$7.2 \cdot 10^{-7}$	$8.9 \cdot 10^{-7}$	$8.4 \cdot 10^{-7}$	$8.0 \cdot 10^{-7}$
Defect	$3.5 \cdot 10^{-7}$	$4.3 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$3.0 \cdot 10^{-6}$	$4.5 \cdot 10^{-6}$	$9.7 \cdot 10^{-6}$	$2.0 \cdot 10^{-6}$

Table 5.2.3 : COLNEW on Problem 1 : 4th order, $\text{tol} = 10^{-8}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	$(5,2)$	$(5,3)$	$(5,3)$	$(5,*)$	$(5,*)$	$(5,*)(320,1)$	$(5,*)(320,1)$
	$(10,1)$	$(10,1)$	$(10,1)$	$(10,6)$	$(10,7)$	$(10,9)(640,1)$	$(10,*)(640,1)$
	$(20,1)$	$(20,1)$	$(20,1)$	$(20,1)$	$(10,1)$	$(10,2)(325,1)$	$(20,*)$
	$(40,1)$	$(40,1)$	$(40,1)$	$(40,1)$	$(20,1)$	$(20,2)(650,1)$	$(40,10)$
		$(80,1)$	$(80,1)$	$(80,1)$	$(40,1)$	$(20,1)$	$(40,2)$
		$(160,1)$	$(160,1)$	$(80,1)$	$(80,1)$	$(40,1)$	$(40,2)$
				$(160,1)$	$(80,1)$	$(40,1)$	$(80,2)$
				$(320,1)$	$(160,1)$	$(80,1)$	$(160,1)$
				$(320,1)$	$(320,1)$	$(160,1)$	$(160,1)$
Time(sec)	0.28	1.13	1.15	3.00	3.13	9.73	9.26
Error	$2.5 \cdot 10^{-9}$	$8.6 \cdot 10^{-10}$	$4.1 \cdot 10^{-9}$	$2.2 \cdot 10^{-9}$	$6.3 \cdot 10^{-9}$	$1.0 \cdot 10^{-9}$	$5.4 \cdot 10^{-9}$
Defect	$1.8 \cdot 10^{-6}$	$3.4 \cdot 10^{-6}$	$1.9 \cdot 10^{-5}$	$4.6 \cdot 10^{-5}$	$1.8 \cdot 10^{-4}$	$2.3 \cdot 10^{-4}$	$6.8 \cdot 10^{-4}$

Table 5.2.4 : MIRKDC on Problem 1 : 4th order, $\text{error} = 10^{-8}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
$tol \Rightarrow$	$7 \cdot 10^{-9}$	$8 \cdot 10^{-8}$	$1 \cdot 10^{-8}$	$3 \cdot 10^{-8}$	$3 \cdot 10^{-8}$	$3 \cdot 10^{-8}$	$3 \cdot 10^{-9}$
Profile	$(5,1)$	$(5,1)$	$(5,1)$	$(5,3)$	$(5,5)$	$(5,*)$	$(5,5)$
	$(20,1)$	$(20,1)$	$(20,1)$	$(20,2)$	$(10,2)$	$(10,4)$	$(20,*)$
	$(52,1)$	$(71,1)$	$(80,1)$	$(80,1)$	$(40,1)$	$(40,*)$	$(40,5)$
		$(92,1)$	$(168,1)$	$(256,1)$	$(160,1)$	$(80,4)$	$(160,1)$
				$(323,1)$	$(341,1)$	$(320,1)$	$(640,1)$
					$(397,1)$	$(536,1)$	$(1071,1)$
						$(595,1)$	$(1186,1)$
Time(sec)	0.106	0.259	0.393	1.000	1.390	2.821	4.852
Error	$1.0 \cdot 10^{-8}$	$9.5 \cdot 10^{-9}$	$7.0 \cdot 10^{-9}$	$9.9 \cdot 10^{-9}$	$9.1 \cdot 10^{-9}$	$9.0 \cdot 10^{-9}$	$9.1 \cdot 10^{-9}$
Defect	$9.3 \cdot 10^{-9}$	$5.3 \cdot 10^{-8}$	$1.3 \cdot 10^{-8}$	$2.7 \cdot 10^{-8}$	$2.4 \cdot 10^{-8}$	$2.8 \cdot 10^{-8}$	$3.4 \cdot 10^{-9}$

Table 5.2.5 : COLNEW on Problem 1 : 6th order, $\text{tol} = 10^{-6}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	$(5,2)$	$(5,3)$	$(5,3)$	$(5,*)$	$(5,*)$	$(5,*)$	$(5,*)$
	$(10,1)$	$(10,1)$	$(10,1)$	$(10,5)$	$(10,5)$	$(10,9)$	$(10,11)$
			$(20,1)$	$(20,1)$	$(20,1)$	$(10,1)$	$(10,1)$
				$(40,1)$	$(40,1)$	$(20,1)$	$(20,1)$
					$(80,1)$	$(20,1)$	$(20,1)$
						$(40,1)$	$(40,1)$
						$(80,1)$	$(80,1)$
Time(sec)	0.178	0.183	0.410	1.053	1.875	2.557	2.646
Error	$2.2 \cdot 10^{-10}$	$9.6 \cdot 10^{-8}$	$1.1 \cdot 10^{-8}$	$3.6 \cdot 10^{-8}$	$5.0 \cdot 10^{-9}$	$2.3 \cdot 10^{-8}$	$1.3 \cdot 10^{-7}$
Defect	$4.3 \cdot 10^{-8}$	$1.3 \cdot 10^{-5}$	$6.4 \cdot 10^{-6}$	$9.4 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$	$4.1 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$

Table 5.2.6 : MIRKDC on Problem 1 : 6th order, $\text{error} = 10^{-6}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
$tol \Rightarrow$	$1 \cdot 10^{-4}$	$6 \cdot 10^{-5}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$9 \cdot 10^{-6}$	$5 \cdot 10^{-6}$	$7 \cdot 10^{-6}$
Profile	$(5,1)$	$(5,1)$	$(5,1)$	$(5,2)$	$(5,3)$	$(5,14)$	$(5,11)$
		$(10,1)$	$(12,1)$	$(20,1)$	$(20,*)$	$(20,*)$	$(20,*)$
		$(12,1)$	$(15,1)$	$(38,1)$	$(40,3)$	$(40,4)$	$(40,5)$
				$(42,1)$	$(49,1)$	$(74,1)$	$(82,1)$
					$(53,1)$	$(82,1)$	$(90,1)$
						$(90,1)$	
Time(sec)	0.015	0.074	0.090	0.310	0.933	1.232	1.118
Error	$6.1 \cdot 10^{-7}$	$7.0 \cdot 10^{-7}$	$9.5 \cdot 10^{-7}$	$5.7 \cdot 10^{-7}$	$6.6 \cdot 10^{-7}$	$4.9 \cdot 10^{-7}$	$9.4 \cdot 10^{-7}$
Defect	$4.5 \cdot 10^{-6}$	$1.1 \cdot 10^{-5}$	$2.6 \cdot 10^{-5}$	$2.6 \cdot 10^{-6}$	$1.8 \cdot 10^{-5}$	$3.7 \cdot 10^{-6}$	$8.1 \cdot 10^{-6}$

Table 5.2.7 : COLNEW on Problem 1 : 6th order, tol $= 10^{-8}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	$(5,2)$	$(5,3)$	$(5,3)$	$(5,*)$	$(5,*)$	$(5,*)$	$(5,*)$
	$(10,1)$	$(10,1)$	$(10,1)$	$(10,6)$	$(10,6)$	$(10,10)$	$(10,11)$
		$(20,1)$	$(20,1)$	$(20,1)$	$(20,1)$	$(10,1)$	$(10,*)$
			$(40,1)$	$(40,1)$	$(40,1)$	$(20,1)$	$(20,3)$
				$(80,1)$	$(80,1)$	$(20,1)$	$(20,1)$
				$(160,1)$	$(160,1)$	$(40,1)$	$(40,1)$
						$(80,1)$	$(80,1)$
						$(160,1)$	$(160,1)$
Time(sec)	0.169	0.348	0.778	1.955	3.589	4.212	4.702
Error	$2.2 \cdot 10^{-10}$	$1.4 \cdot 10^{-9}$	$1.6 \cdot 10^{-10}$	$5.0 \cdot 10^{-10}$	$8.7 \cdot 10^{-11}$	$5.2 \cdot 10^{-10}$	$2.5 \cdot 10^{-9}$
Defect	$4.3 \cdot 10^{-8}$	$4.8 \cdot 10^{-7}$	$2.3 \cdot 10^{-7}$	$3.5 \cdot 10^{-6}$	$1.3 \cdot 10^{-6}$	$1.5 \cdot 10^{-5}$	$1.0 \cdot 10^{-4}$

Table 5.2.8 : MIRKDC on Problem 1 : 6th order, error $= 10^{-8}$

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
$tol \Rightarrow$	$5 \cdot 10^{-7}$	$8 \cdot 10^{-7}$	$6 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$7 \cdot 10^{-8}$	$7 \cdot 10^{-8}$
Profile	$(5,1)$	$(5,1)$	$(5,1)$	$(5,3)$	$(5,3)$	$(5,14)$	$(5,11)$
	$(9,1)$	$(18,1)$	$(20,1)$	$(20,1)$	$(20,*)$	$(20,*)$	$(20,*)$
		$(23,1)$	$(32,1)$	$(72,1)$	$(40,3)$	$(40,4)$	$(40,5)$
				$(87,1)$	$(92,1)$	$(135,1)$	$(150,1)$
					$(104,1)$	$(162,1)$	$(183,1)$
						$(178,1)$	$(201,1)$
Time(sec)	0.039	0.125	0.161	0.541	1.178	1.886	2.160
Error	$6.9 \cdot 10^{-9}$	$7.2 \cdot 10^{-9}$	$8.6 \cdot 10^{-9}$	$8.5 \cdot 10^{-9}$	$8.6 \cdot 10^{-9}$	$8.1 \cdot 10^{-9}$	$8.8 \cdot 10^{-9}$
Defect	$8.4 \cdot 10^{-8}$	$2.1 \cdot 10^{-7}$	$2.9 \cdot 10^{-7}$	$3.6 \cdot 10^{-8}$	$3.6 \cdot 10^{-8}$	$2.0 \cdot 10^{-8}$	$3.6 \cdot 10^{-8}$

5.3. Numerical results for problem 2

The second problem for which we present results is the Swirling Flow I problem from [AMR 1988, Pg. 22] (see the Appendix, Section 2.) All other aspects of the testing procedure and the software configurations are the same as for the previous section.

Table 5.3.1 : COLNEW on Problem 2 : 4th order, $\mathrm{tol} = 10^{-6}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	$(5,7)$	$(5,5)$	$(5,6)$	$(5,10)$	$(5,14)(80,1)$	$(5,13)(40,1)$
	$(10,1)$	$(10,2)$	$(10,2)$	$(10,3)$	$(10,7)(160,1)$	$(10,8)(80,1)$
	$(10,1)$	$(10,1)$	$(10,1)$	$(10,2)$	$(10,2)(320,1)$	$(10,6)(160,1)$
	$(20,1)$	$(20,1)$	$(20,1)$	$(20,1)$	$(20,1)$	$(10,2)(320,1)$
	$(40,1)$	$(40,1)$	$(40,1)$	$(40,1)$	$(20,1)$	$(20,1)(160,1)$
		$(80,1)$	$(80,1)$	$(80,1)$	$(20,1)$	$(20,1)(320,1)$
		$(160,1)$	$(160,1)$	$(160,1)$	$(40,1)$	$(20,1)$
				$(320,1)$	$(40,1)$	$(40,1)$
Time(sec)	0.304	0.988	1.015	1.998	2.360	3.763
Error	$2.4 \cdot 10^{-7}$	$4.7 \cdot 10^{-8}$	$6.0 \cdot 10^{-7}$	$4.8 \cdot 10^{-8}$	$8.1 \cdot 10^{-8}$	$5.2 \cdot 10^{-8}$
Defect	$7.5 \cdot 10^{-5}$	$1.9 \cdot 10^{-4}$	$1.6 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	$8.2 \cdot 10^{-3}$

Table 5.3.2 : MIRKDC on Problem 2 : 4th order, $\mathrm{error} = 10^{-6}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
$tol \Rightarrow$	$5 \cdot 10^{-7}$	$1 \cdot 10^{-6}$	$6 \cdot 10^{-7}$	$3 \cdot 10^{-7}$	$6 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
Profile	$(5,5)$	$(5,3)$	$(5,5)$	$(5,6)$	$(5,*)$	$(5,*)$
	$(20,1)$	$(20,1)$	$(20,2)$	$(20,2)$	$(10,6)$	$(10,7)$
	$(36,1)$	$(80,1)$	$(80,1)$	$(80,1)$	$(40,2)$	$(40,3)$
		$(111,1)$	$(163,1)$	$(234,1)$	$(160,1)$	$(160,1)$
				$(282,1)$	$(282,1)$	$(459,1)$
					$(313,1)$	$(554,1)$
					$(334,1)$	$(609,1)$
Time(sec)	0.151	0.730	0.623	0.884	2.303	2.161
Error	$9.9 \cdot 10^{-7}$	$7.1 \cdot 10^{-7}$	$8.9 \cdot 10^{-7}$	$7.9 \cdot 10^{-7}$	$7.9 \cdot 10^{-7}$	$7.0 \cdot 10^{-7}$
Defect	$6.0 \cdot 10^{-7}$	$9.4 \cdot 10^{-7}$	$1.2 \cdot 10^{-6}$	$3.1 \cdot 10^{-7}$	$5.3 \cdot 10^{-7}$	$1.6 \cdot 10^{-7}$

Table 5.3.3 : COLNEW on Problem 2 : 4th order, tol $= 10^{-8}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	$(5,7)$	$(5,6)$	$(5,7)$	$(5,11)$	$(5,14)(320,1)$	$(5,13)(320,1)$
	$(10,1)$	$(10,2)$	$(10,2)$	$(10,3)$	$(10,7)(320,1)$	$(10,8)(320,1)$
	$(10,1)$	$(10,1)$	$(10,2)$	$(10,2)$	$(10,3)(640,1)$	$(10,7)(320,1)$
	$(20,1)$	$(20,1)$	$(20,1)$	$(20,1)$	$(20,2)$	$(10,2)(640,1)$
	$(40,1)$	$(40,1)$	$(40,1)$	$(40,1)$	$(20,1)$	$(20,*)$
	$(80,1)$	$(80,1)$	$(80,1)$	$(80,1)$	$(20,1)$	$(40,3)$
	$(160,1)$	$(160,1)$	$(160,1)$	$(160,1)$	$(40,1)$	$(40,1)$
		$(320,1)$	$(320,1)$	$(320,1)$	$(80,1)$	$(40,1)$
			$(640,1)$	$(268,1)$	$(80,1)$	$(80,1)$
				$(536,1)$	$(160,1)$	$(160,1)$
Time(sec)	0.887	1.948	3.706	4.176	5.096	6.423
Error	$9.1 \cdot 10^{-10}$	$3.7 \cdot 10^{-9}$	$2.1 \cdot 10^{-9}$	$1.5 \cdot 10^{-9}$	$2.0 \cdot 10^{-9}$	$3.3 \cdot 10^{-9}$
Defect	$1.3 \cdot 10^{-6}$	$2.4 \cdot 10^{-5}$	$2.5 \cdot 10^{-5}$	$5.3 \cdot 10^{-5}$	$1.9 \cdot 10^{-4}$	$6.7 \cdot 10^{-4}$

Table 5.3.4 : MIRKDC on Problem 2 : 4th order, error $= 10^{-8}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
$tol \Rightarrow$	$4 \cdot 10^{-9}$	$1 \cdot 10^{-8}$	$7 \cdot 10^{-9}$	$3 \cdot 10^{-9}$	$5 \cdot 10^{-9}$	$9 \cdot 10^{-10}$
Profile	$(5,5)$	$(5,3)$	$(5,5)$	$(5,6)$	$(5,*)$	$(5,*)$
	$(20,1)$	$(20,1)$	$(20,2)$	$(20,2)$	$(10,6)$	$(10,7)$
	$(80,1)$	$(80,1)$	$(80,1)$	$(80,1)$	$(40,2)$	$(40,3)$
	$(124,1)$	$(278,1)$	$(320,1)$	$(320,1)$	$(160,1)$	$(160,1)$
		$(356,1)$	$(516,1$	$(752,1)$	$(640,1)$	$(640,1)$
				$(888,1)$	$(949,1)$	$(1508,1)$
					$(1043,1)$	$(1783,1)$
						$(1961,1)$
Time(sec)	0.235	1.053	1.044	1.713	3.351	3.233
Error	$9.0 \cdot 10^{-9}$	$6.6 \cdot 10^{-9}$	$9.2 \cdot 10^{-9}$	$8.3 \cdot 10^{-9}$	$9.2 \cdot 10^{-9}$	$6.4 \cdot 10^{-9}$
Defect	$3.6 \cdot 10^{-9}$	$8.0 \cdot 10^{-9}$	$7.9 \cdot 10^{-9}$	$2.5 \cdot 10^{-9}$	$3.9 \cdot 10^{-9}$	$9.0 \cdot 10^{-10}$

Table 5.3.5 : COLNEW on Problem 2 : 6th order, tol $= 10^{-6}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	$(5,7)$	$(5,5)$	$(5,7)$	$(5,11)$	$(5,19)$	$(5,*)$
	$(10,1)$	$(10,1)$	$(10,1)$	$(10,2)$	$(10,2)$	$(10,*)$
	$(20,1)$	$(20,1)$	$(20,1)$	$(20,1)$	$(10,1)$	$(20,*)$
		$(40,1)$	$(14,1)$	$(40,1)$	$(20,1)$	$(40,*)$
			$(28,1)$	$(80,1)$	$(40,1)$	$(80,*)$
					$(21,1)$	$(160,*)$
					$(42,1)$	$(320,*)$
						$(640,*)$
						$(1280,*)$
Time(sec)	0.433	0.644	0.760	1.524	1.671	*FAIL*
Error	$7.4 \cdot 10^{-8}$	$1.2 \cdot 10^{-7}$	$1.1 \cdot 10^{-7}$	$6.1 \cdot 10^{-9}$	$1.7 \cdot 10^{-7}$	–
Defect	$9.6 \cdot 10^{-6}$	$2.5 \cdot 10^{-5}$	$4.4 \cdot 10^{-4}$	$4.0 \cdot 10^{-4}$	$3.6 \cdot 10^{-3}$	–

Table 5.3.6 : MIRKDC on Problem 2 : 6th order, error $= 10^{-6}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
$tol \Rightarrow$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$5 \cdot 10^{-6}$
Profile	$(5,5)$	$(5,9)$	$(5,*)$	$(5,*)$	$(5,*)$	$(5,*)$
	$(12,1)$	$(20,13)$	$(10,5)$	$(10,*)$	$(10,*)$	$(10,*)$
	$(14,1)$	$(35,1)$	$(34,1)$	$(20,5)$	$(20,*)$	$(20,*)$
		$(38,1)$	$(41,1)$	$(52,1)$	$(40,20)$	$(40,12)$
				$(61,1)$	$(63,1)$	$(89,1)$
				$(67,1)$	$(69,1)$	$(97,1)$
					$(75,1)$	$(106,1)$
						$(116,1)$
Time(sec)	0.106	0.624	0.318	0.901	2.013	1.791
Error	$4.7 \cdot 10^{-7}$	$2.0 \cdot 10^{-7}$	$8.5 \cdot 10^{-7}$	$4.0 \cdot 10^{-7}$	$6.5 \cdot 10^{-7}$	$6.0 \cdot 10^{-7}$
Defect	$2.4 \cdot 10^{-6}$	$2.8 \cdot 10^{-6}$	$1.9 \cdot 10^{-5}$	$3.8 \cdot 10^{-6}$	$2.0 \cdot 10^{-5}$	$3.2 \cdot 10^{-6}$

Table 5.3.7 : COLNEW on Problem 2 : 6th order, $\mathrm{tol} = 10^{-8}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	$(5,7)$	$(5,5)$	$(5,7)$	$(5,12)$	$(5,19)$	$(5,*)$
	$(10,1)$	$(10,1)$	$(10,1)$	$(10,2)$	$(10,3)$	$(10,*)$
	$(20,1)$	$(20,1)$	$(20,1)$	$(10,1)$	$(10,2)$	$(20,*)$
	$(10,1)$	$(40,1)$	$(20,1)$	$(20,1)$	$(20,1)$	$(40,*)$
	$(20,1)$	$(23,1)$	$(40,1)$	$(40,1)$	$(40,1)$	$(80,*)$
		$(46,1)$	$(80,1)$	$(80,1)$	$(40,1)$	$(160,*)$
				$(160,1)$	$(80,1)$	$(320,*)$
						$(640,*)$
						$(1280,*)$
Time(sec)	0.541	1.039	1.354	2.592	2.460	*FAIL*
Error	$1.0 \cdot 10^{-9}$	$1.5 \cdot 10^{-9}$	$5.5 \cdot 10^{-10}$	$1.1 \cdot 10^{-10}$	$2.6 \cdot 10^{-9}$	–
Defect	$3.3 \cdot 10^{-7}$	$2.3 \cdot 10^{-6}$	$6.0 \cdot 10^{-6}$	$1.8 \cdot 10^{-5}$	$2.0 \cdot 10^{-4}$	–

Table 5.3.8 : MIRKDC on Problem 2 : 6th order, $\mathrm{error} = 10^{-8}$

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
$tol \Rightarrow$	$2 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$2 \cdot 10^{-7}$	$5 \cdot 10^{-8}$
Profile	$(5,5)$	$(5,9)$	$(5,*)$	$(5,*)$	$(5,*)$	$(5,*)$
	$(20,1)$	$(20,13)$	$(10,5)$	$(10,*)$	$(10,*)$	$(10,*)$
	$(25,1)$	$(53,1)$	$(40,1)$	$(20,5)$	$(20,*)$	$(20,*)$
		$(61,1)$	$(75,1)$	$(80,1)$	$(40,20)$	$(40,12)$
			$(82,1)$	$(125,1)$	$(121,1)$	$(160,1)$
				$(137,1)$	$(143,1)$	$(201,1)$
					$(157,1)$	$(221,1)$
						$(243,1)$
Time(sec)	0.152	0.756	0.627	1.224	2.586	2.880
Error	$8.5 \cdot 10^{-9}$	$9.1 \cdot 10^{-9}$	$9.4 \cdot 10^{-9}$	$4.6 \cdot 10^{-9}$	$8.9 \cdot 10^{-9}$	$6.7 \cdot 10^{-9}$
Defect	$7.0 \cdot 10^{-8}$	$1.6 \cdot 10^{-7}$	$1.3 \cdot 10^{-7}$	$2.7 \cdot 10^{-8}$	$9.2 \cdot 10^{-8}$	$3.3 \cdot 10^{-8}$

5.4. Discussion of results

These results demonstrate that the observed performance of MIRKDC can be comparable to that of COLNEW over a range of accuracy requests and problem difficulties. The mesh selection strategy and stopping criterion of COLNEW are conservative and force an iteration on a final mesh of N subintervals (where N is the number of subintervals reported in the performance profiles). This final iteration, required primarily to obtain a reliable error estimate (after a convergent solution on N/2 subintervals has been determined), can result in a solution that is more accurate than requested at a cost that is higher than necessary. This behaviour leads to costs being very sensitive to small changes in the accuracy request (or to any other problem parameter) and therefore makes direct comparison of results very difficult. This must be kept in mind when interpreting our numerical results. The error control strategy and mesh selection strategy of MIRKDC are more flexible than that of COLNEW (allowing a more finely adapted selection of an appropriate mesh) and leads to a generally smoother relationship between the cost and requested accuracy.

An overview of the relative performance of the two methods on the two families of test problems is provided in figures 1-8 (see the Appendix, Section 4). For problem 1 the seven levels of difficulty (associated with the x-axis of figures 1-4) correspond to the seven values of 'ϵ' reported in the detailed performance profiles of tables 5.2.1-5.2.8. For problem 2 the six levels of difficulty (associated with the x-axis of figures 5-8) correspond to the six values of 'γ' reported in the detailed performance profiles of tables 5.3.1-5.3.8.

From these figures (especially those associated with problem 1) one can observe that MIRKDC is at least as efficient and exhibits smoother and more predictable behaviour as the problem become more difficult. For problem 2 this trend is also observable but there are also a few instances (for both methods) where the cost is unexpectedly large or no results are available. By studying the corresponding performance profiles we see that the former behaviour is usually the result of a large number of Newton iterations associated with a coarse-mesh approximate solution, while the latter behaviour is the result of a Newton iteration failure at all attempted meshes. In practical applications these two related difficulties are not uncommon and special method-independent techniques are often used to overcome them. For example, when the Newton iteration fails to converge for any mesh one can obtain more accurate initial guesses by employing continuaton with respect to the problem parameter that quantifies the level of difficulty.

Note that the rather large defects observed in the performance profiles associated with COLNEW should not be interpreted as a deficiency. This code was not designed to control this quantity and a different interpolating strategy would lead to a much improved relationship between the size of the defect and the requested accuracy. This statistic is only reported to confirm that MIRKDC is controlling the defect in the way that is intended.

6. Conclusions and Future Work

In this paper we have discussed an alternative approach for the numerical solution of BVODE problems. The major new aspects are the use of MIRK formulas rather than collocation formulas and the use of defect control rather than global error estimation for accuracy control and mesh selection. The algorithms for this alternate approach have been implemented in a software package called MIRKDC which, in some preliminary testing has demonstrated performance comparable to that a widely used collocation formula based software package, COLNEW. The results are sufficiently encouraging to indicate that further work in the development of the MIRKDC package is worthwhile. Further testing of the package on a larger set of test problems is now proceeding.

There are several aspects of the new approach and software that require substantial further investigation. Foremost among these is the need to explore the space of available MIRK formulas to select specific optimal formulas of each order. A related question involves a theoretical analysis of the development of continuous MIRK formulas containing optimal embedded discrete MIRK formulas. We believe that the order barriers in such a case may be different than those of [Muir and Owren 1993], where the restriction of an embedded optimal discrete formula is not included. Subsequent to the characterization of optimal CMIRK formulas with optimal embedded discrete MIRK formulas, a study to explore the parameter space of the CMIRK families to discover appropriate specific formulas is needed.

Additional areas of investigation include examining some of the parameters which control the Newton iteration for the solution of the discrete non-linear problems arising from Runge-Kutta discretization. Some of these have been explored elsewhere, and some in section 4 of this paper, but further work is needed. The overall efficiency and robustness of a BVODE code is very much dependent on the quality of the Newton iteration algorithm and associated controlling parameters. Other areas where further investigation of controlling parameters is required is in the mesh redistribution algorithm, and in the software controlling continuation of solution and mesh over the discrete problem sequences, as indicated in section 4.1.3. Finally, a number of questions remain to be answered concerning the use of defect control in the numerical solution of BVODE problems. A variety of defect estimation schemes are available, and the most appropriate choice for the BVODE context remains to be established.

7. References

U. Ascher, *Two families of symmetric difference schemes for singular perturbation problems,*, in Numerical Boundary Value ODEs, U. Ascher and R.D. Russell (eds.), Birkhauser, Boston (1985).

U. Ascher, *Collocation for two-point boundary value problems revisited*, SIAM J. Numer. Anal. **23** (1986), 596–609.

U. Ascher and G. Bader, *Stability of collocation at Gaussian points*, SIAM J. Numer. Anal. **23** (1986), 412–422.

U. Ascher, J. Christiansen, and R.D. Russell, *Collocation software for boundary value ODEs*, Trans. on Math. Software **7** (1981), 209–222.

U. Ascher, R.M.M. Mattheij, and R. Russell, "Numerical solution of boundary value problems for ordinary differential equations," Prentice Hall, New Jersey, 1988.

U. Ascher and S. Jacobs, *On collocation implementation for singularly perturbed two-point problems*, SIAM J. Sci. Stat. Comput. **10** (1989), 533–549.

U. Ascher and R. Weiss, *Collocation for singular perturbation problems I: first order systems with constant coefficients*, SIAM J. Numer. Anal. **20** (1983), 537–557.

U. Ascher and R. Weiss, *Collocation for singular perturbation problems II: linear first order systems without turning points*, Math. Comp. **43** (1984a), 157–187.

U. Ascher and R. Weiss, *Collocation for singular perturbation problems III: nonlinear problems without turning points*, SIAM J. Sci. Stat. Comp. **5** (1984b), 811–829.

G. Bader and U. Ascher, *A new basis implementation for a mixed order boundary value ode solver*, SIAM J. Sci. Stat. Comp. **8** (July 1987), 483–500.

G.K. Batchelor, *A note on a class of solutions of the Navier-Stokes equations representing rotationally symmetric flow*, Quart. J. Mech. Appl. Math. **4** (1951), 29–41.

W. M. G. van Bokhoven, *Efficient higher order implicit one-step methods for integration of stiff differential equations*, BIT **20** (1980), 34–43.

D.L. Brown and J. Lorenz, *A high-order method for stiff boundary value problems with turning points*, SIAM J. Sci. Stat. Comp. **8** (1987), 790–808.

K. Burrage, F. Chipman, and P. Muir, *Order Results for Mono-Implicit Runge-Kutta Methods*, submitted to SIAM J. Numer. Anal. (1993).

J. C. Butcher, "The Numerical Analysis of Ordinary Differential Equations," John Wiley & Sons, Toronto, 1987.

J. R. Cash, *On the numerical integration of nonlinear two-point boundary value problems using iterated deferred corrections, Part 1: A survey and comparison of some one-step formulae*, Comput. Math. Appl. **12a** (1986), 1029–1048.

J. R. Cash, *On the numerical integration of nonlinear two-point boundary value problems using iterated deferred corrections, Part 2: The development and analysis of highly stable deferred correction formulae*, SIAM J. Numer. Anal. **25** (1988), 862–882.

J. R. Cash and A. Singhal, *Mono-implicit Runge-Kutta formulae for the numerical integration of stiff differential systems*, IMA J. Numer. Anal. **2** (1982a), 211–227.

J. R. Cash and A. Singhal, *High order methods for the numerical solution of two-point boundary value problems*, BIT **22** (1982b), 184–199.

J. R. Cash and M. H. Wright, *Implementation issues in solving nonlinear equations for two-point boundary value problems*, Computing **45** (1990), 17–37.

J. R. Cash and M. H. Wright, *A deferred correction method for nonlinear two-point boundary value problems: implementation and numerical evaluation*, SIAM J. Sci. Stat. Comput. **12 pages 971–989** (1991).

W. H. Enright, *A new error-control for initial value solvers*, Appl. Math. and Comp. **31** (1989a), 288–301.

W. H. Enright, *Analysis of error control strategies for continuous Runge-Kutta methods*, SIAM J. Numer. Anal. **26** (1989b), 588–599.

W.H. Enright, K.R. Jackson, S.P. Nørsett, and P.G. Thomsen, *Interpolants for Runge-Kutta formulas*, ACM Trans. Math. Softw. **12** (1986), 193–218.

W.H. Enright, K.R. Jackson, S.P. Nørsett, and P.G. Thomsen, *Effective solution of discontinuous IVPs using a Runge-Kutta formula pair with interpolants*, Appl. Math. and Comp. **27** (1988), 313–335.

W. H. Enright and P. H. Muir, *Efficient classes of Runge-Kutta methods for two-point boundary value problems*, Computing **37** (1986), 315–334.

I. Gladwell, L. F. Shampine, L. S. Baca, and R. W. Brankin, *Practical aspects of interpolation in Runge-Kutta codes*, SIAM J. Sci. Stat. Comp. **8** (1987), 322–341.

S. Gupta, *An adaptive boundary value Runge-Kutta solver for first order boundary value problems*, SIAM J. Numer. Anal. **22** (1985), 114–126.

H.-O. Kreiss, N. K. Nichols, and D. L. Brown, *Numerical methods for stiff two-point boundary value problems*, SIAM J. Numer. Anal. **23** (1986), 325–368.

M. Lentini and V. Pereyra, *An adaptive finite difference solver for nonlinear two-point boundary value problems with mild boundary layers*, SIAM J. Numer. Anal. **14** (1977), 91–111.

P.H. Muir and B. Owren, *Order barriers and characterizations of continuous mono-implicit Runge-Kutta schemes*, to appear Math. Comp. (1993).

B. Owren and M. Zennaro, *Order barriers for continuous explicit Runge-Kutta methods*, Math. Comp. **56**

(1991), 645–661.

B. Owren and M. Zennaro, *Derivation of optimal continuous explicit Runge-Kutta methods*, to appear in SIAM J. Stat. Sci. Comp. (1992).

J. Verner, *Differentiable interpolants for high-order Runge-Kutta methods,*, Technical Report 1990-9, Queen's University (1990).

R. Weiss, *The application of implicit Runge-Kutta and collocation methods to boundary value problems*, Math. Comp. **28** (1974), 449–464.

8. Appendix

8.1. MIRK/CMIRK schemes of 6th order

A family of 6th order CMIRK schemes from [Muir and Owren 1991] with $c_3 = \frac{1}{4}$, $c_4 = \frac{3}{4}$, and $c_5 = \frac{1}{2}$, and free parameters c_6, c_7, c_8, v_7, v_8, and x_{81}:

$$
\begin{array}{c|c|cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{5}{32} & \frac{9}{64} & \frac{-3}{64} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{3}{4} & \frac{27}{32} & \frac{3}{64} & \frac{-9}{64} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{24} & \frac{-1}{24} & \frac{1}{6} & \frac{-1}{6} & 0 & 0 & 0 & 0 \\
c_6 & v_6 & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & 0 & 0 & 0 \\
c_7 & v_7 & x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & 0 & 0 \\
c_8 & v_8 & x_{81} & x_{82} & x_{83} & x_{84} & x_{85} & x_{86} & x_{87} & 0 \\
\hline
 & & b_1(\theta) & b_2(\theta) & b_3(\theta) & b_4(\theta) & b_5(\theta) & b_6(\theta) & b_7(\theta) & b_8(\theta)
\end{array}
$$

where for the 6th stage we have,

$$
v_6 = -\left(50\,c_6 - 60\,c_6^2 + 24\,c_6^3 - 15\right)c_6^2, \quad x_{61} = \frac{c_6\left(12\,c_6^2 - 10\,c_6 + 3\right)(c_6 - 1)^2}{3},
$$

$$
x_{62} = \frac{c_6^2\,(c_6 - 1)\left(12\,c_6^2 - 14\,c_6 + 5\right)}{3}, \quad x_{63} = \frac{8\,c_6^2\,(c_6 - 1)^2}{3}, \quad x_{64} = -\frac{8\,c_6^2\,(c_6 - 1)^2}{3},
$$

$$
x_{65} = 8\,c_6^2\,(2\,c_6 - 1)\,(c_6 - 1)^2\,.
$$

For the 7th stage,

$$x_{71} = \frac{-\alpha_1}{90(2\,c_6-1)c_6}, \quad x_{72} = \frac{\alpha_2}{90(2\,c_6-1)(c_6-1)},$$

$$x_{73} = \frac{240\,c_6c_7^4+240\,c_7^2c_6-480\,c_6c_7^3-8\,v_7-160\,c_7^3+360\,c_7^4-192\,c_7^5}{90\,c_6-45}$$

$$x_{74} = -\frac{240\,c_7^2c_6-480\,c_6c_7^3+240\,c_6c_7^4-160\,c_7^3+360\,c_7^4-8\,v_7-192\,c_7^5}{90\,c_6-45},$$

$$x_{75} = -\frac{8\,v_7-48\,c_7^5+120\,c_7^4-40\,c_6v_7-80\,c_6c_7^3+80\,c_6^2c_7^3+40\,c_6^2v_7-120\,c_6^2c_7^2+120\,c_7^2c_6-80\,c_7^3}{15\,(2\,c_6-1)^2},$$

$$x_{76} = \frac{v_7+24\,c_7^5-60\,c_7^4+50\,c_7^3-15\,c_7^2}{(30\,c_6-30)\,c_6\,(2\,c_6-1)^2},$$

where

$$\alpha_1 = 3\,v_7+72\,c_7^5-45\,c_7^2-180\,c_7^4-23\,c_6v_7-100\,c_6c_7^3+360\,c_6c_7^4-600\,c_6^2c_7^3+$$

$$30\,c_6^2v_7+240\,c_6^2c_7^4+90\,c_7c_6-192\,c_7^5c_6+510\,c_6^2c_7^2-135\,c_7^2c_6-180\,c_6^2c_7+150\,c_7^3,$$

and

$$\alpha_2 = 120\,c_7^5-10\,v_7-180\,c_7^4+37\,c_6v_7+140\,c_6c_7^3+120\,c_6c_7^4-360\,c_6^2c_7^3-30\,c_6^2v_7$$

$$+240\,c_6^2c_7^4-192\,c_7^5c_6+150\,c_6^2c_7^2-105\,c_7^2c_6+70\,c_7^3.$$

For the 8th stage,

$$x_{82} = \frac{\beta_2}{(30\,c_7-30)\,(c_6-1)\,(-6\,c_6-6\,c_7+16\,c_7c_6+3)}, \quad x_{83} = \frac{\beta_3}{-90\,c_6-90\,c_7+240\,c_7c_6+45},$$

$$x_{84} = \frac{-\beta_4}{-90\,c_6-90\,c_7+240\,c_7c_6+45}, \quad x_{85} = \frac{-\beta_5}{(30\,c_6-15)\,(2\,c_7-1)\,(-6\,c_6-6\,c_7+16\,c_7c_6+3)},$$

$$x_{86} = \frac{-\beta_6}{(60\,c_6-30)\,(c_6-c_7)\,(-6\,c_6-6\,c_7+16\,c_7c_6+3)\,(c_6-1)},$$

$$x_{87} = \frac{\beta_7}{(60\,c_7-30)\,(c_6-c_7)\,(-6\,c_6-6\,c_7+16\,c_7c_6+3)\,(c_7-1)},$$

where

$$\beta_2 = 160\,c_6^2v_8c_7-300\,c_6c_7^2c_8-10\,v_8+70\,c_8^3-180\,c_8^4+120\,c_8^5-360\,c_6c_8^3c_7-174\,c_6c_7v_8-60\,c_6c_7c_8^2-210\,c_6x_{81}c_7$$

$$+210\,c_6c_8c_7-105\,c_7c_8^2+140\,c_7c_8^3+120\,c_7c_8^4-105\,c_6c_8^2+120\,c_6c_8^4+140\,c_6c_8^3+37\,c_7v_8+37\,c_6v_8+150\,c_7^2c_8^2-30\,c_6^2v_8$$

$$-30\,c_7^2v_8-360\,c_7^2c_8^3-192\,c_8^5c_6-192\,c_8^5c_7+240\,c_7^2c_8^4+240\,c_6^2c_8^4-360\,c_6^2c_8^3+150\,c_6^2c_8^2-960\,c_6^2c_7^2c_8^2+160\,c_7^2v_8c_6$$

$$-480\,c_7^2c_8^4c_6+320\,c_7^2c_8^3c_6+300\,c_6^2c_7x_{81}+320\,c_6^2c_8^3c_7-480\,c_6^2c_8^4c_7-480\,c_6^2c_7^2x_{81}-160\,c_6^2c_7^2v_8+384\,c_8^5c_6c_7$$

$$+300\,c_6c_7^2x_{81}+300\,c_6^2c_7c_8^2+640\,c_6^2c_8^3c_7^2-300\,c_6^2c_8c_7+300\,c_7^2c_8^2c_6+480\,c_6^2c_8c_7^2,$$

$$\beta_3=320\,c_6c_8^3c_7-80\,c_6c_7v_8-720\,c_6c_7c_8^2-480\,c_6x_{81}c_7-240\,c_7c_8^2+480\,c_7c_8^3+480\,c_6c_8c_7-240\,c_7c_8^4+8\,v_8-240\,c_6c_8^2$$

$$-240\,c_6c_8^4+160\,c_8^3+480\,c_6c_8^3-360\,c_8^4+192\,c_8^5,$$

$$\beta_4=320\,c_6c_8^3c_7-80\,c_6c_7v_8-720\,c_6c_7c_8^2-480\,c_6x_{81}c_7-240\,c_7c_8^2+480\,c_7c_8^3+480\,c_6c_8c_7-240\,c_7c_8^4+8\,v_8-240\,c_6c_8^2$$

$$-240\,c_6c_8^4+160\,c_8^3+480\,c_6c_8^3-360\,c_8^4+192\,c_8^5$$

$$\beta_5=720\,c_6c_7^2c_8-680\,c_6^2v_8c_7+24\,v_8-240\,c_8^3+360\,c_8^4-144\,c_8^5+240\,c_6c_8^3c_7+672\,c_6c_7v_8+720\,c_6x_{81}c_7-720\,c_6c_8c_7$$

$$+360\,c_7c_8^2-240\,c_7c_8^3+360\,c_6c_8^2-240\,c_6c_8^3-120\,c_7v_8-120\,c_6v_8-360\,c_7^2c_8^2+120\,c_6^2v_8+120\,c_7^2v_8+240\,c_7^2c_8^3$$

$$+240\,c_6^2c_8^3-360\,c_6^2c_8^2-1920\,c_6^2c_7^2c_8^2-680\,c_7^2v_8c_6-960\,c_7^2c_8^4c_6+1280\,c_7^2c_8^3c_6-720\,c_6^2c_7x_{81}$$

$$+1280\,c_6^2c_8^3c_7-960\,c_6^2c_8^4c_7+640\,c_6^2c_7^2v_8+768\,c_8^5c_6c_7-720\,c_6c_7^2x_{81}-360\,c_6^2c_7c_8^2$$

$$+1280\,c_6^2c_8^3c_7^2+720\,c_6^2c_8c_7-360\,c_7^2c_8^2c_6-960\,c_8^4c_6c_7$$

$$\beta_6=360\,c_7c_8^4-23\,c_7v_8+3\,v_8+72\,c_8^5+150\,c_8^3-180\,c_8^4-100\,c_7c_8^3-135\,c_7c_8^2+510\,c_7^2c_8^2+30\,c_7^2v_8-600\,c_7^2c_8^3$$

$$-192\,c_8^5c_7+240\,c_7^2c_8^4+90\,c_7c_8-90\,c_7x_{81}-45\,c_8^2+180\,c_7^2x_{81}-180\,c_8c_7^2$$

$$\beta_7=360\,c_6c_8^4+3\,v_8+72\,c_8^5+150\,c_8^3-23\,c_6v_8-180\,c_8^4-100\,c_6c_8^3-135\,c_6c_8^2+30\,c_6^2v_8-192\,c_8^5c_6+240\,c_6^2c_8^4$$

$$-600\,c_6^2c_8^3+510\,c_6^2c_8^2-90\,c_6x_{81}+90\,c_6c_8-45\,c_8^2+180\,c_6^2x_{81}-180\,c_6^2c_8.$$

The weight polynomials are

$$b_1(\theta)=\frac{-\eta_1}{60\,c_7c_6c_8}\quad b_2(\theta)=\frac{-\eta_2}{(60\,c_7-60)\,(c_6-1)\,(c_8-1)},\quad b_3(\theta)=0,\quad b_4(\theta)=0,$$

$$b_5(\theta)=\frac{\eta_5}{(30\,c_8-15)\,(2\,c_7-1)\,(2\,c_6-1)},\quad b_6(\theta)=\frac{\eta_6}{60\,c_6\,(c_6-c_8)\,(c_6-c_7)\,(c_6-1)\,(2\,c_6-1)},$$

$$b_7(\theta)=\frac{-\eta_7}{60\,c_7\,(c_7-c_8)\,(c_7-1)\,(c_6-c_7)\,(2\,c_7-1)},\quad b_8(\theta)=\frac{\eta_8}{(60\,c_8-60)\,(c_7-c_8)\,(c_6-c_8)\,(2\,c_8-1)\,c_8},$$

where

$$\eta_1=\theta(20\,\theta^5-60\,c_7c_6c_8-36\,\theta^4+15\,\theta^3+45\,\theta^3c_6+45\,\theta^3c_7+45\,\theta^3c_8-24\,\theta^4c_6-24\,\theta^4c_7-24\,\theta^4c_8-20\,\theta^2c_6-20\,\theta^2c_7$$

$$-20\,\theta^2c_8-60\,\theta^2c_7c_6-60\,\theta^2c_6c_8-60\,\theta^2c_7c_8+30\,\theta^3c_7c_6+30\,\theta^3c_6c_8+30\,\theta^3c_7c_8$$

$$+30\,\theta c_7c_6+30\,\theta c_6c_8+30\,\theta c_7c_8+90\,\theta c_7c_6c_8-40\,\theta^2c_7c_6c_8)$$

$$\eta_2=\theta^2(30\,c_7c_6c_8-20\,\theta c_7c_6-20\,\theta c_6c_8-20\,\theta c_7c_8-40\,\theta c_7c_6c_8+15\,\theta^2c_6+15\,\theta^2c_8+15\,\theta^2c_7+30\,\theta^2c_6c_8$$

$$+30\,\theta^2 c_7 c_6 + 30\,\theta^2 c_7 c_8 - 12\,\theta^3 - 24\,\theta^3 c_6 - 24\,\theta^3 c_8 - 24\,\theta^3 c_7 + 20\,\theta^4),$$

$$\eta_5 = 8\,\theta^2(30\,c_7c_6c_8 - 20\,\theta c_7c_6c_8 - 20\,\theta c_7c_6 - 20\,\theta c_6c_8 - 20\,\theta c_7c_8 + 15\,\theta^2 c_7c_6 + 15\,\theta^2 c_6c_8 + 15\,\theta^2 c_6$$

$$+15\,\theta^2 c_7c_8 + 15\,\theta^2 c_7 + 15\,\theta^2 c_8 - 12\,\theta^3 c_6 - 12\,\theta^3 c_7 - 12\,\theta^3 c_8 - 12\,\theta^3 + 10\,\theta^4),$$

$$\eta_6 = \theta^2(45\,\theta^2 c_8 - 60\,\theta c_7c_8 + 45\,\theta^2 c_7 + 30\,\theta^2 c_7c_8 - 36\,\theta^3 - 24\,\theta^3 c_8 - 24\,\theta^3 c_7 + 20\,\theta^4 + 30\,c_7c_8 - 20\,\theta c_7 - 20\,\theta c_8 + 15\,\theta^2),$$

$$\eta_7 = \theta^2(45\,\theta^2 c_6 - 60\,\theta c_6c_8 + 45\,\theta^2 c_8 + 30\,\theta^2 c_6c_8 - 36\,\theta^3 - 24\,\theta^3 c_6 - 24\,\theta^3 c_8 + 20\,\theta^4 + 30\,c_6c_8 - 20\,\theta c_6 - 20\,\theta c_8 + 15\,\theta^2),$$

$$\eta_8 = \theta^2(45\,\theta^2 c_6 - 60\,\theta c_7c_6 + 45\,\theta^2 c_7 + 30\,\theta^2 c_7c_6 - 36\,\theta^3 - 24\,\theta^3 c_6 - 24\,\theta^3 c_7 + 20\,\theta^4 + 30\,c_7c_6 - 20\,\theta c_6 - 20\,\theta c_7 + 15\,\theta^2),$$

8.2. Detailed descriptions of the test problems

The swirling flow III family of test problems is given in [AMR 1988, Pg. 23]. The version of the problem given there is slightly incorrect. The corrected version of the problem, which has been studied extensively in a large number of papers for some time, can be found, in a slightly modified form, in, for example, [Batchelor 1951], and has the following form,

$$\epsilon f'''' + ff''' + gg' = 0, \qquad \epsilon g'' + fg' - f'g = 0,$$

with

$$f(0) = f(1) = f'(0) = f'(1) = 0, \quad g(0) = \Omega_0, \quad g(1) = \Omega_1.$$

In this paper we consider the following first order system form,

$$y_1' = y_2, \quad \epsilon y_2' = y_1y_4 - y_2y_3, \quad y_3' = y_4, y_4' = y_5, \quad y_5' = y_6, \quad y_6' = -y_3y_6 - y_1y_2,$$

$$y_3(0) = y_4(0) = y_3(1) = y_4(1) = 0, \quad y_1(0) = \Omega_0, \quad y_1(1) = \Omega_1.$$

No closed form solution exists in general. For our testing, we chose $\Omega_0 = 1$ and $\Omega_1 = -1$, and consider values of the parameter $\epsilon = 1.0$, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005. The initial guesses for the solution are chosen to be straight lines through the boundary conditions.

The swirling flow I family of test problems is given in [AMR 1988, Pg. 22], and has the following first order system form,

$$y_1' = y_2, \quad y_2' = y_3, \quad y_3' = \gamma^2 - 2y_1y_3 + y_2^2 - y_4^2, y_4' = y_5, \quad y_5' = 2y_2y_4 - 2y_1y_5,$$

with

$$y_1(0) = y_2(0) = 0, \quad y_4(0) = 1, \quad y_2(\infty) = 0, \quad y_4(\infty) = \gamma.$$

No closed form solution exists in general. For our testing, we chose $L = 10.0$. The values of γ we consider are 0.0, 5.0, 10.0, 20.0, 50.0, 100.0, and the initial conditions are straight lines through the boundary conditions.

8.3. Mixed vs. First Order BVODE systems

In the Table 7.3.1, we show the results from comparing the performance of COLNEW applied to a mixed order BVODE system with COLNEW applied to the equivalent first order BVODE system, using collocation polynomials off order 4 and 6 (which we will call CO for collocation order.). We used the swirling flow III problem family as the set of test problems. We expressed the family in its original form, as above, with a 4th order equation and a 2nd order equation, and also in terms of an equivalant first order system form with 6 first order equations. We requested a tolerance on every component of 10^{-8}, and imposed an upper bound of 300 subintervals. Note that a '*' indicates that the code failed to converge using no more than 300 subintervals.

Table 7.3.1 : COLNEW on a Mixed and First Order system

$CO \Rightarrow$	4th	4th	6th	6th
$\epsilon \Downarrow$	Mixed Order	First Order	Mixed Order	First Order
1.0	0.12	0.28	0.05	0.16
0.1	0.49	1.13	0.11	0.39
0.05	0.48	1.15	0.22	0.72
0.01	1.34	3.31	0.48	1.82
0.005	1.38	3.38	0.94	3.31
0.001	*	*	1.09	3.93
0.0005	*	*	*	4.59

Note that for the last value of $\epsilon = 0.0005$, the first order system version of the problem could be solved by COLNEW within 300 subintervals, while the mixed order system version could not. In general, substantial savings in execution time are obtained by applying COLNEW directly to a mixed order system.

8.4. Graphs for Tables 5.1.1-5.1.8 and 5.2.1-5.2.8

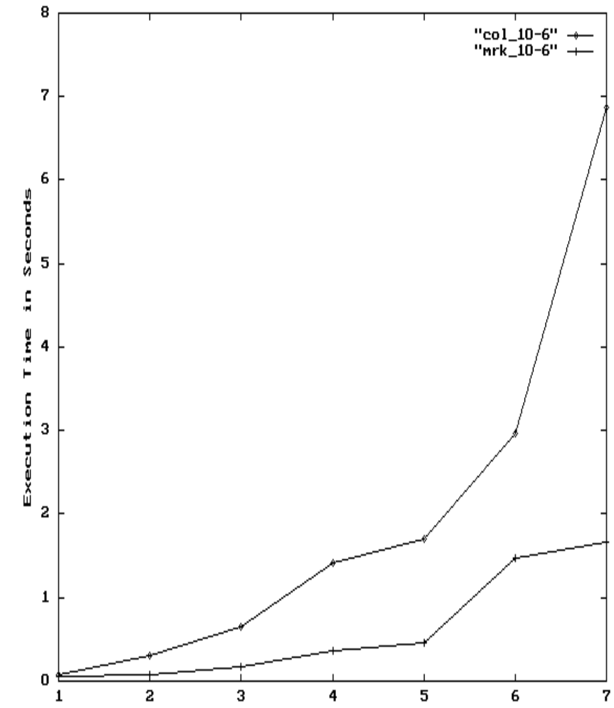

Figure 1: COLNEW and MIRKDC, Problem 1, 4th order, Err=10-6

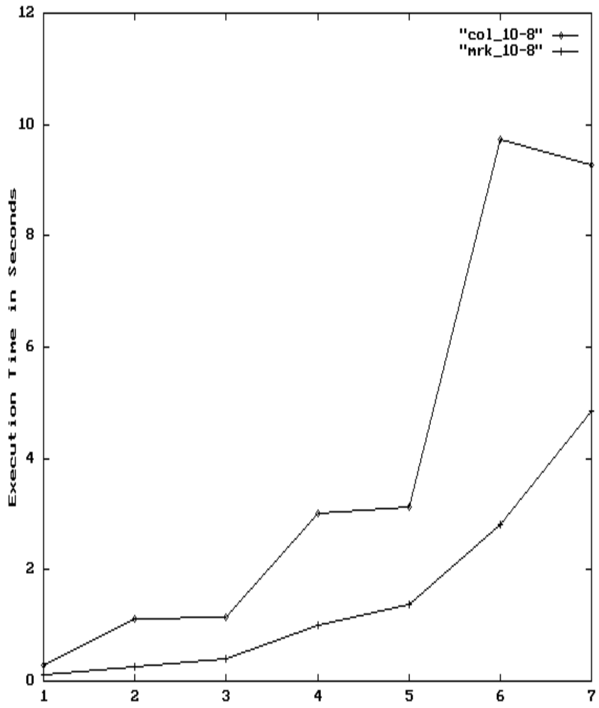

Figure 2: COLNEW and MIRKDC, Problem 1, 4th order, Err=10-8

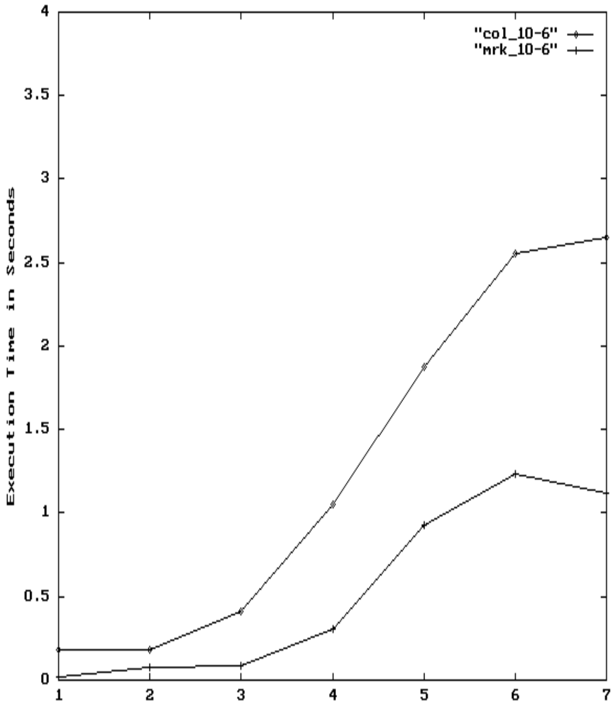

Figure 3: COLNEW and MIRKDC, Problem 1, 6th order, Err=10-6

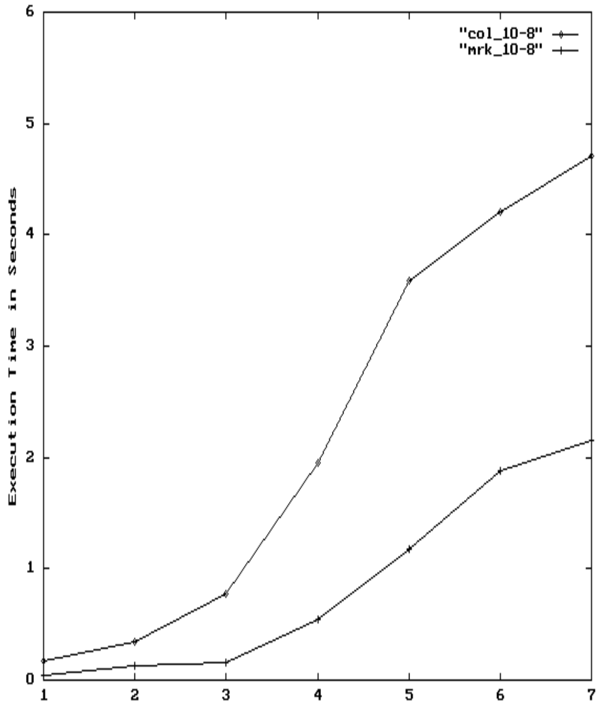

Figure 4: COLNEW and MIRKDC, Problem 1, 6th order, Err=10-8

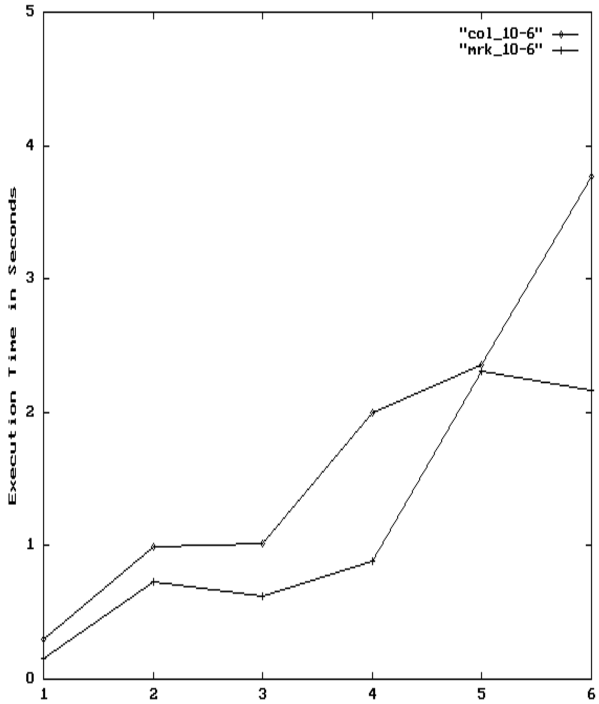

Figure 5: COLNEW and MIRKDC, Problem 2, 4th order, Err=10-6

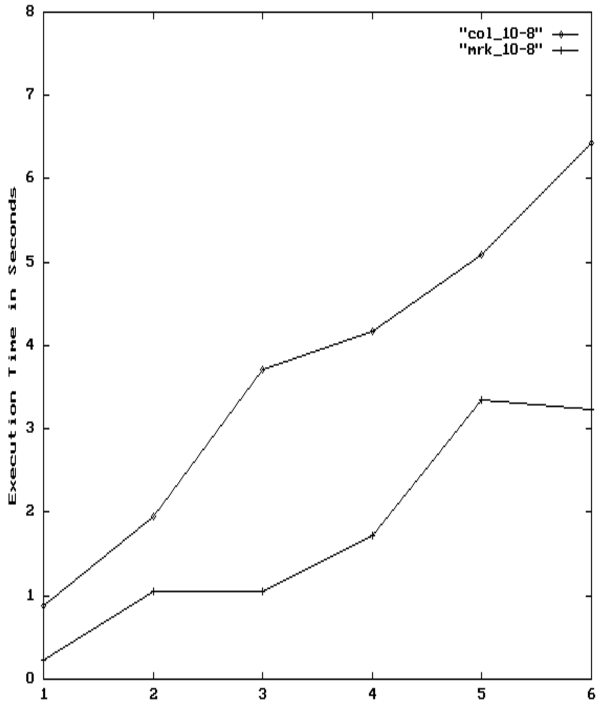

Figure 6: COLNEW and MIRKDC, Problem 2, 4th order, Err=10-8

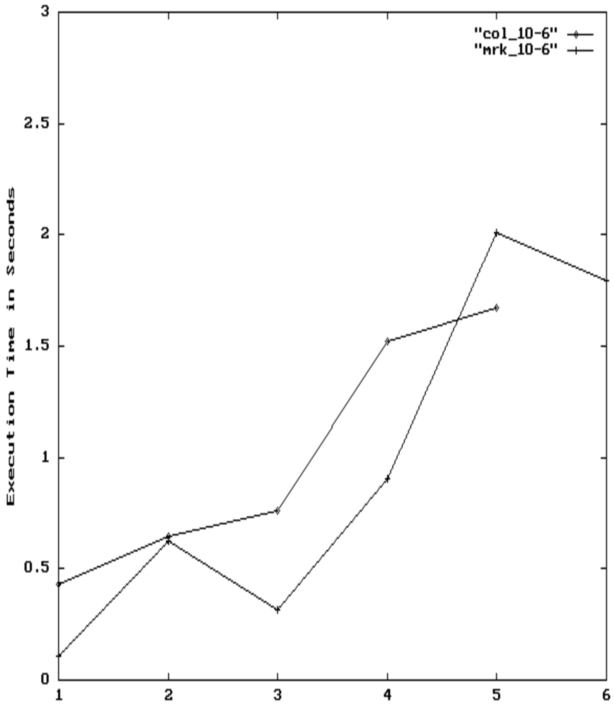

Figure 7: COLNEW and MIRKDC, Problem 2, 6th order, Err=10-6

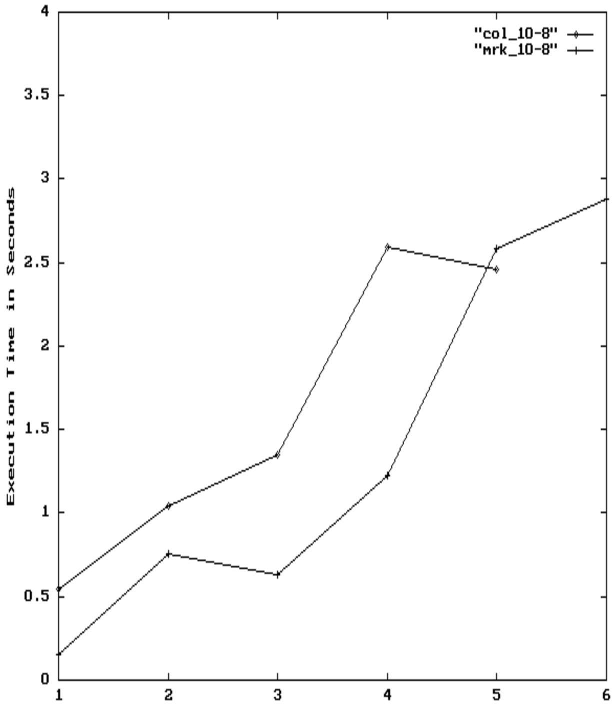

Figure 8: COLNEW and MIRKDC, Problem 2, 6th order, Err=10-8