Modifications to a Runge-Kutta Type Software Package for the Numerical Solution of Boundary Value Ordinary Differential Equations

By

Hui Xu

A thesis submitted in partial fulfillment of

the requirements for the degree of

Master of Applied Science (in Computer Science)

Saint Mary's University

Halifax, Nova Scotia

Submitted August 31, 2004

Your file Votre référence
ISBN: 0-612-96135-4
Our file Notre référence
ISBN: 0-612-96135-4

Certification

Name: Hui Xu

Degree: Master of Science in Applied Science

Title of Thesis: Modifications to a Runge-Kutta Type Software Package for the Numerical Solution of Boundary Value Ordinary Differential Equations

Examining Committee:

Dr. William E. Jones, Acting Dean of Graduate Studies

Dr. David H. S. Richardson, Program Co-ordinator

Dr. Pat Keast, External Examiner
Dalhousie University

Dr. Paul Muir, Senior Supervisor

Dr. Walt Finden, Supervisory Committee

Dr. Milt Chew, Supervisory Committee

Date Certified: August 31, 2004

To My Family

Contents

1 Introduction **2**

2 Review of Standard Techniques and Software **10**

2.1 Introduction 10

2.2 Numerical Methods for BVODEs 11

2.2.1 Shooting/Multiple Shooting 11

2.2.2 Collocation 12

2.2.3 Finite Difference Methods 13

2.2.4 Runge-Kutta Schemes 13

2.2.5 Summary 16

2.3 Numerical Software 17

2.3.1 Shooting Software 17

2.3.2 COLSYS - Software Based on Collocation 18

2.3.3 TWPBVP - Software Based on Deferred Corrections 18

2.3.4 MIRKDC - Software Based on Defect Control 19

2.3.5 Comparison 20

3 Test Problems **21**

3.1 Test Problems 21

3.2 Computer / Compiler Information 23

4 Modification of MIRKDC (I)— Software Modifications Based on Existing Modules and Formulas **24**

4.1 Computational Derivative Approximation 25

4.1.1 Introduction 25

4.1.2 Description of the Software Modification 25

4.1.3 Results and Discussion 26

4.2 Analytic Derivative Assessment 29

4.2.1 Introduction 29

4.2.2 Description of Software Modification 29

4.2.3 Results and Discussion 30

4.3 Problem Sensitivity (Conditioning) Assessment 31

4.3.1 Introduction 31

4.3.2 Description of the Software Modification 37

4.3.3 Results and Discussion 38

4.4 Improved Runge-Kutta Methods 39

4.4.1 Introduction 39

4.4.2 Results and Discussion 42

4.5 Preliminary Global Error Indicator 42

4.5.1 Introduction and Description of Software Modification 42

4.5.2 Results and Discussion 44

5 Modification of MIRKDC (II) — Design and Analysis of Defect Control Strategies **49**

5.1 Introduction 49

5.2 Description of the Software Modification 50

5.3 Continuous Runge-Kutta Schemes 52

5.3.1 A Continuous Runge-Kutta Scheme of 2nd Order 52

5.3.2 A New Continuous 4th Order Scheme 54

5.3.3 A New Continuous 6th Order Scheme 58

5.4 Numerical Experiments and Results 63

5.4.1 Experimental Location of Maximum Defect 63

5.4.2 Comparison of Relaxed Defect Control, Strict Defect Control and Safe-Guarded Strict Defect Control 73

5.5 Conclusions 81

6 Conclusions and Future Work **84**

6.1 Conclusions 84

6.2 Future Work 85

List of Tables

2.1 Tableau of an s-stage MIRK scheme 16

4.1 Absolute maximum difference between analytic and approximate Jacobian on TP1 - TP5, method = 4, Nsub = 10. 28

4.2 CPU time for $dfsub$ and $dfsub\_diff$ for TP1 and TP2; method = 4, tol = 10^{-3}, Nsub= 10, $\epsilon = 0.04$ for TP1, $\epsilon = 1.4$ for TP2; time in seconds. . . 28

4.3 CPU time for overall MIRKDC computation with $dfsub\_diff$ and $dfsub$ on TP1 and TP2; time in seconds. 28

4.4 Execution times for TP1 for COLROW and BSPCNDMAX/CRSLVE. . . 38

4.5 Execution times for overall MIRKDC computation with and without conditioning constant estimate for TP1, TP3, and TP4, method = 4, tol = 10^{-6}, $\epsilon = 0.04$, $\gamma = 1.4$, initial Nsub = 2. 38

4.6 Tableau for 3-stage, 4th order, stage order 3, MIRK scheme. 40

4.7 Tableau for original 4-stage, 4th order, stage order 3, CMIRK scheme. . . 41

4.8 Tableau for new 4-stage, 4th order, stage order 3 CMIRK scheme. 41

4.9 Comparison of MIRKDC mesh sizes and defect estimates for TP1, tol = 10^{-9}, method = 4, ϵ = 0.001, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8. 43

4.10 Comparison of MIRKDC mesh sizes and defect estimates for TP2, tol = 10^{-9}, method = 4, ϵ = 0.1, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8. 44

4.11 Comparison of MIRKDC mesh sizes and defect estimates for TP1, tol = 10^{-9}, method = 4, ϵ = 0.0001, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8. 45

4.12 Comparison of MIRKDC mesh sizes and defect estimates for TP2, tol = 10^{-9}, method = 4, ϵ = 0.01, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8. 46

4.13 Comparison of CPU times for MIRKDC global error estimation. 48

5.1 The locations of the maximum defect for different numbers of subintervals, method = 4, tol = 10^{-9}, TP1 - TP3. 52

5.2 The locations of the maximum defect for different numbers of subintervals, method = 4, tol = 10^{-9}, for TP4 - TP6. 53

5.3 General form for the tableau of 5-stage, 4th order, stage order 3 CMIRK scheme with the discrete 3-stage, 4th order, stage order 3 MIRK scheme of Table 4.6 embedded. 54

5.4 Tableau of 5-stage, 4th order, stage order 3 CMIRK scheme which gives an asymptotically correct defect estimate. 56

5.5 Tableau of 9-stage, 6th order, stage order 3 CMIRK scheme which gives an asymptotically correct defect estimate. 59

5.6 Comparison of MIRKDC execution sequences for TP1, ϵ= 0.04; tol = 10^{-9}, method = 4; tol = 10^{-12}, method = 6; NI: the number of full Newton Iterations. 77

5.7 CPU time (seconds) for defect control strategies for various initial Nsub values; method = 4, tol = 10^{-9}, for TP1, $\epsilon = 0.04$. 79

5.8 CPU time (seconds) for defect control strategies for various initial Nsub values; method = 6, tol = 10^{-9}, for TP1, $\epsilon = 0.04$. 79

5.9 Comparison of defect control strategies; TP1 with $\epsilon = 0.04$, tol = 10^{-9}; EX1: exact maximum defect - original interpolant; EX2: relaxed defect control - original interpolant; EX3: safe-guarded strict defect control - new interpolant. 82

5.10 Comparison of relaxed defect control and safe-guarded strict defect control for TP1, $\epsilon = 0.04$, tol = 10^{-4}, method = 2. 83

List of Figures

4.1 Comparison of ratio of estimate error and true error of solution component 1 on TP7, method = 4, tol = 10^{-6}. 47

4.2 Comparison of ratio of estimate error and true error of solution component 2 on TP7, method = 4, tol = 10^{-6}. 47

5.1 Location of maximum defect on each subinterval for six test problems with Nsub = 10, method = 2, tol = 10^{-9}. 64

5.2 Location of maximum defect on each subinterval for six test problems with Nsub = 50, method = 2, tol = 10^{-9}. 65

5.3 Location of maximum defect on each subinterval for six test problems with Nsub = 100, method = 2, tol = 10^{-9}. 65

5.4 Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the old 4th order method, tol = 10^{-9}. 66

5.5 Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the old 4th order method, tol = 10^{-9}. 67

5.6 Location of maximum defect on each subinterval for six test problems with Nsub = 300 for the old 4th order method, tol = 10^{-9}. 67

5.7 Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the new 4th order method, tol = 10^{-9}. 68

5.8 Location of maximum defect on each subinterval for six test problems with Nsub = 50 for the new 4th order method, tol = 10^{-9}. 68

5.9 Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the new 4th order method, tol = 10^{-9}. 69

5.10 Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the old 6th order method, tol = 10^{-9}. 70

5.11 Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the old 6th order method, tol = 10^{-9}. 70

5.12 Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the new 6th order method, tol = 10^{-9}. 71

5.13 Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the new 6th order method, tol = 10^{-9}. 71

5.14 Location of maximum defect on each subinterval for six test problems with Nsub = 300 for the new 6th order method, tol = 10^{-9}. 72

5.15 Locations of maximum defects for TP6, uniform mesh, Nsub = 10, method order 4. 74

5.16 Locations of maximum defects for TP6, nonuniform mesh, Nsub = 40, method order 4. 74

5.17 Locations of maximum defects for TP6, nonuniform mesh, Nsub = 156, method order 4. 75

5.18 Locations of maximum defects for TP6, nonuniform mesh, Nsub = 205, method order 4. 75

ACKNOWLEDGEMENTS

I express my sincere gratitude to all those who gave me the opportunity to complete this thesis.

First of all, I would like to sincerely thank my supervisor, Dr. Paul Muir. He provided a motivating, enthusiastic, and critical atmosphere during the many discussions we had. He opened the world of Numerical Analysis to me. It was a great pleasure for me to conduct this thesis under his supervision.

Thanks also to my external thesis examiner, Dr. Patrick Keast (Dalhousie University), for his valuable comments and suggestions. I also would like to thank internal examiners, Dr. Walt Finden (Saint Mary's University) and Dr. Milton Chew (Saint Mary's University), for their critical review and useful suggestions on this thesis.

During my graduate study at Saint Mary's University, my colleagues and professors from the Mathematics and Computing Science Department gave me lots of support and help. I really appreciated them.

Finally, I would like to give my special thanks to my husband, my son, and my parents for their constant encouragement and love which enabled me to complete this work.

Chapter 1

Introduction

A differential equation describes how something changes. It allows us to model phenomena that change continuously over time or space. Mathematical models based on differential equations are widely used in the sciences to attempt to describe the real world. These models are essential tools from the molecular level in the analysis of chemical reactions, to the cosmic level in the motion of planets or comets.

Computer modeling of complex phenomena now plays an important role in all areas of science and engineering. It is usually the case that such models are based upon complicated systems of differential equations. Furthermore, the complexity of these systems implies that they cannot be treated with traditional analytic approaches, and therefore sophisticated, robust software packages for the approximate numerical solution of these systems must be employed.

An ordinary differential equation (ODE) is an equation involving a function of one independent variable and its derivatives. There are two main types: initial value

ODEs (IVODEs) and boundary value ODEs (BVODEs). An IVODE consists of a system of differential equations with solution information specified at one initial point. A simple IVODE would have the form:

$$y'(t) = f(t, y(t)), \quad t \geq a, \tag{1.1}$$

$$y(a) = \alpha, \tag{1.2}$$

where y and f are vectors and a is the initial point and α is a given constant vector. A BVODE consists of a system of differential equations on a given interval with conditions on the solution at two or more points. We wish to find the solution of the BVODE on this interval. A simple and common form for a two-point BVODE is [1]:

$$y'(t) = f(t, y(t)), \tag{1.3}$$

$$g(y(a), y(b)) = 0, \tag{1.4}$$

where y, f, g, and 0 are vectors and a and b are known endpoints. Equation (1.3) subject to (1.4) is a first order system (which means that only first derivatives appear). In general, if the ODE is pth order, the boundary conditions may involve derivatives of the unknown function up to the $(p-1)$st.

This thesis describes software development and modification associated with the enhancement of the MIRKDC [18] software package. This Fortran 77 package is used for the numerical solution of systems of first order, nonlinear, BVODEs, with separated boundary conditions. Given a mesh of points which partitions the problem interval [a,b], it employs mono-implicit Runge-Kutta methods [18] for the discretization of the ODEs and monitors the quality of the numerical solution using defect

control. The discrete systems are solved by modified Newton iterations and extensive use of adaptive mesh refinement is employed, based on equidistribution of the defect.

This thesis work has involved two phases. Phase one is concerned with the modification of the MIRKDC software package in order to incorporate a number of performance enhancements including analytic derivative assessment, computational derivative approximation, problem sensitivity (i.e., conditioning) assessment, and the introduction of an auxiliary global error indicator. In each of these modifications, we build upon well established theoretical developments, numerical results, and robust software components already available, and the focus is in the introduction of these features into the existing MIRKDC package in a well-structured and well-documented manner. Numerical results to demonstrate the impact of these enhancements will be presented.

The second phase examines new approaches for control of the defect. The defect of a numerical solution is the amount by which that solution fails to satisfy the ODE system. For example, suppose that $u(t)$ is the approximate solution to (1.3), (1.4). Then the defect, $\delta(t)$, is defined as follows:

$$\delta(t) = u'(t) - f(t, u(t)).$$

That is, we substitute the approximate solution $u(t)$ into the differential equation $y'(t) = f(t, y(t))$ in place of $y(t)$ and see how well $u(t)$ satisfies the differential equation by subtracting the right hand side from the left hand side. The difference is called the defect.

This second phase of this thesis work involves an investigation of new approaches

for the defect control strategy employed within MIRKDC for the estimation of the maximum defect on each subinterval of the mesh which subdivides the problem interval. The main goal is to try to compute an approximate solution so that the corresponding defect is less than some user defined tolerance, over the whole problem interval [a,b]. We investigate a new class of continuous solution approximations which lead to *an asymptotically correct estimate* of the maximum defect, determined by a single evaluation of the defect on each subinterval. This is potentially a significant improvement over the previous strategy employed in MIRKDC, which *estimated* the maximum defect by choosing the largest of several samples of the defect within each subinterval.

In [16], it is pointed out that, for the design and implementation of ODE software over the last few decades, there are two frameworks where such software is now widely used. One of these frameworks is generally referred to under the title of Problem Solving Environments (PSEs), e.g. MATLAB [32] or MAPLE [30]. In PSEs, scientists and engineers are often able to solve their problem without having much knowledge of the underlying numerical algorithms. This convenience associated with the PSE is often offset by a loss in efficiency within the numerical computation. The second major framework, called General Scientific Computing (GSC), provides more efficiency, but there is a cost to the user in terms of ease of use. In the GSC environment, the user is able to access high quality, robust, efficient numerical algorithms through software modules available from various libraries, e.g. IMSL [39], NAG [33], and Netlib [37], but the user must be able to write driver programs in a high level programming

language, such as Fortran or C, in order to call these modules. The MIRKDC package, to be considered in this thesis, falls into this class, and therefore, throughout this thesis, one important assessment of the modifications we undertake will be a measure of their contribution to the overall efficiency of the computation, as measured in terms of execution or CPU time and of the complexity added to the interface.

Recent efforts, e.g. Shampine et al. [26], have sought to bridge the gap between these two environments by employing an updated version of the Fortran programming language known as Fortran 90/95. In this language, one is able to dramatically simplify the interface to a sophisticated software module by employing various new language features such as optional parameters, structured data objects, and dynamic memory allocation. The user can call such a module using a simple driving program template, and generally interact with the module in a fairly simple way, while largely enjoying the significant advantages in overall computational efficiency associated with a GSC environment.

This thesis involves modification of the MIRKDC software package in order to incorporate a number of significant performance enhancements including:

1. *Computational derivative approximation* — In the current version of MIRKDC, the user must supply several subroutines including $fsub$, $gsub$, $dfsub$, and $dgsub$. The first two subroutines describe the ODEs and boundary conditions respectively; the latter subroutines describe the partial derivatives of the ODEs and boundary conditions. When the ODEs are complicated, it is usually difficult for the user to supply correct $dfsub$ and $dgsub$ subroutines. By modifying MIRKDC

to allow it to provide numerical approximations to the partial derivatives, we can avoid asking the user to supply the *dfsub* and *dgsub* routines and can significantly improve the ease-of-use of the package. (We should note that the use of numerical derivative approximation does add to the overall computational costs.)

2. *Analytic derivative assessment* — When the ODEs are complicated, the development of a correct Jacobian subroutine is an error prone process. When the user does provide the *dfsub* and *dgsub* routines, the purpose of analytic derivative assessment is to let MIRKDC attempt to check these routines to see if they correspond to the partial derivatives of the expressions given in the *fsub* and *gsub* routines. This check is done by comparing the results from the *dfsub* and *dgsub* routines with numerically generated partial derivatives.

3. *Problem sensitivity (i.e., conditioning) assessment* — The MIRKDC solver attempts to compute a numerical solution whose defect is less than a user prescribed tolerance. It can be shown - see, e.g. [25], that the global error, i.e. the difference between the true solution and the approximation solution, is bounded by the product of the defect and the *conditioning constant* for the BVODE, which gives a measure of the sensitivity of the BVODE to slight changes in the problem definition. When the conditioning constant is large, the problem is ill-conditioned and a small defect will not imply a small error. In the extreme case when the product of the conditioning constant and the defect of the computed solution is greater than one, the error can be very large and the solution may

have no correct digits; in this case the computed solution is sometimes called a *pseudosolution.* In this modification to MIRKDC we added the capability for estimating the conditioning constant of the BVODE. This estimate can be returned to alert the user to the fact that the BVODE is ill-conditioned, implying the possibility of a loss of accuracy in the solution, and in the extreme case, the possibility of a pseudosolution.

4. *New Runge-Kutta methods* — As mentioned previously, the discretization of the ODEs is performed in MIRKDC using Runge-Kutta methods. In [21] new optimal mono-implicit Runge-Kutta methods with interpolants are derived. In this thesis work, we added a new interpolant of order 4 to MIRKDC and compared the performance with the original method.

5. *Preliminary auxiliary global error indicator* — While the primary mode for MIRKDC is defect control, i.e., the computation of a numerical solution whose defect satisfies a given tolerance, it might be considered useful to also provide a low cost estimate of the global error in the final computed solution. The global error is the difference between the approximate solution and the true solution. That is, if $y(t)$ is the true solution to (1.3), (1.4) and $u(t)$ is an approximate solution to (1.3), (1.4), then the global error is simply $y(t) - u(t)$, the difference between them. In this modification, once MIRKDC has obtained a final approximate solution, we compute a global error estimate by computing a second global solution on a new mesh and then comparing it with the original final solution. This provides a high quality global error estimate but

the additional costs are significant. However this approach provides a baseline for future work in the development of lower cost global error estimates.

6. *Defect control improvements* — An investigation of new approaches for the defect control strategy employed in MIRKDC is presented. In the current version of MIRKDC, *relaxed defect control*, in which we sample the defect at several points on each subinterval and then select the largest of these samples as an estimate of maximum defect, is used to monitor the quality of a numerical solution. We have investigated another defect control strategy called *strict defect control* in this thesis. Strict defect control samples the defect at one point per subinterval *using a special interpolant* and is guaranteed to give *an asymptotically correct estimate* of maximum defect when the subinterval size is sufficiently small. In this thesis, we derive special interpolants of orders 4 and 6, that lead to asymptotically correct defects.

This thesis is organized as follows. In chapter 2, we review standard numerical methods for BVODEs. Some software packages for the numerical solution of the BVODEs are also discussed. In chapter 3, we provide all the test problems which will be used in this thesis. Chapter 4 presents all the modifications to MIRKDC we have considered except that involving improvement of the defect control strategy. Chapter 5 describes the modifications we implemented for defect control and gives an analysis of our experimental results. Chapter 6 gives our conclusions and suggestions for future work.

Chapter 2

Review of Standard Techniques and Software

2.1 Introduction

Most approaches for the numerical solution of BVODEs use a mesh to partition the problem interval $[a, b]$, into several subintervals. The mesh points are usually chosen in an adaptive way to control some estimate of the error. On each subinterval one employs a numerical method which discretizes the ODEs; the resultant set of equations together with the boundary conditions leads to a large system of nonlinear equations, which is usually solved using a modified Newton's method. Some form of error estimate is also obtained through an auxiliary computation. Once an approximate solution and error estimate are obtained, the error is assessed to see if it satisfies the given user tolerance. If not, the error distribution over the problem interval is

used to guide a mesh redistribution algorithm. Once the new mesh is obtained, the above process is repeated.

2.2 Numerical Methods for BVODEs

2.2.1 Shooting/Multiple Shooting

One of the most popular approaches for the numerical solution of BVODEs is the simple shooting method. It is a simple, intuitive method that builds on the IVODE approach; shooting is a straightforward extension of initial value techniques. Essentially, one shoots trajectories of the same ordinary differential equation, with estimated initial values employed at the left end of the problem interval, until one hits the given boundary values at the right end of the problem interval.

Consider the equation (1.3); we denote by $y(x) \equiv y(x;c)$, the vector solution of the ODE which satisfies the initial (or left end point) condition $y(a;c) = c$. We can then write $h(c) \equiv g\left(y(a;c), y(b;c)\right) = g(c, y(b;c)) = 0$, from the boundary conditions. This gives a set of nonlinear equations for the unknown initial conditions c.

The advantages of this approach are simplicity and the ability to make use of the excellent initial value ODE software [3]. Difficulties in shooting are that the conditioning of each shooting step depends on the conditioning of the IVODE, not on the conditioning of the BVODE. It is well known that simple shooting is unstable, i.e., errors can grow in an unbounded fashion [1].

An improvement on simple shooting is called multiple shooting. In this approach,

one uses a set of mesh points or shooting points to partition the problem interval, and then on each subinterval one sets up and solves a local initial value problem with an estimated initial condition employed at the left end of each subinterval. This gives a larger set of unknowns, but the interval over which each IVODEs needs to be solved is smaller, making each of the local IVODEs easier to solve. Requiring the local initial value solutions to match at the internal mesh points and to satisfy the boundary conditions leads to a large system of nonlinear equations. The shooting points are chosen according to the difficulty of the local initial value problem, so that a problem of comparable difficulty is solved on each subinterval. Multiple shooting can be ineffective for singularly perturbed problems because of the possible presence of rapidly increasing solution modes that can not be dealt with using an initial value solver.

2.2.2 Collocation

Another popular technique for solving BVODEs is collocation. This approach begins by selecting a set of basis functions, usually piecewise polynomials, defined on a mesh of points which partition the problem interval. Continuity of the basis functions at the mesh points is usually imposed. The collocation approach requires the solution approximation to satisfy the differential equation at a set of collocation points defined over the problem interval. The resultant equations, called collocation conditions, together with the continuity and boundary conditions, give a large system of nonlinear equations. Important aspects of this approach are the choice of appropriate basis

functions and collocation points. These have been discussed in many papers, see, e.g., [4],[13]. One standard choice is to use B-splines [6] for the basis functions and Gauss points for the collocation points.

2.2.3 Finite Difference Methods

In the text book [1], the basic steps for the use of a finite difference method for the numerical solution of a BVODE are as follows:

1. For a given mesh π: $a = x_1 < x_2 < \cdots < x_N < x_{N+1} = b$, define approximate solution values $y_j \approx y(x_j)$.

2. Form a set of equations for the approximate solution values by replacing derivatives with finite difference quotients in the differential equations and boundary conditions, e.g. $y'(x_j) \approx \frac{y_{j+1}-y_j}{x_{j+1}-x_j}$.

3. Solve this set of equations together with the boundary conditions for the approximate solution values.

2.2.4 Runge-Kutta Schemes

Explicit Runge-Kutta (ERK) schemes were originally proposed for the numerical solution of IVODEs by Runge and then were further developed by Heun and Kutta - see, e.g., [7]. Implicit Runge-Kutta methods were later recognized as appropriate for stiff initial value differential equations [7]. Runge-Kutta schemes have also been considered for use in the numerical solution of BVODEs for some time.

Again assuming a mesh and discrete approximation solution values as in section 2.2.3, on each subinterval we replace the ODE, $y'(t) = f(t, y(t))$, using a Runge-Kutta scheme which discretizes the ODE at t_i, the general form being

$$\frac{y_{i+1} - y_i}{h} = \sum_{r=1}^{s} b_r K_r, \quad \text{or} \quad y_{i+1} = y_i + h \sum_{r=1}^{s} b_r K_r,$$

where $K_r = f\left(t_i + c_r h, y_i + h \sum_{j=1}^{s} a_{rj} K_j\right)$, $r = 1, ..., s$, $h = t_{i+1} - t_i$, and b_r, c_r and a_{rj} are the *coefficients* of the Runge-Kutta scheme. For example, the classical fourth-order explicit Runge-Kutta method is expressed as follows [7],

$$\frac{y_{i+1} - y_i}{h} = \frac{1}{6}K_1 + \frac{1}{3}(K_2 + K_3) + \frac{1}{6}K_4,$$

$$K_1 = f(t_i, y_i), \quad K_2 = f(t_i + \frac{1}{2}h, y_i + \frac{1}{2}hK_1),$$

$$K_3 = f(t_i + \frac{h}{2}, y_i + \frac{1}{2}hK_2), \quad K_4 = f(t_i + h, y_i + hK_3),$$

and we have $s = 4; b_1 = b_4 = \frac{1}{6}, b_2 = b_3 = \frac{1}{3}; c_1 = 0, c_2 = c_3 = \frac{1}{2}, c_4 = 1; a_{11} = a_{12} = a_{13} = a_{14} = 0, a_{21} = \frac{1}{2}, a_{22} = a_{23} = a_{24} = 0, a_{32} = \frac{1}{2}, a_{31} = a_{33} = a_{34} = 0, a_{43} = 1, a_{41} = a_{42} = a_{44} = 0$. These methods represent generalizations of the finite difference methods and for $y'(t) = f(t, y(t))$ also include the collocation schemes.

More recent work [11],[17],[21] has included studies of a subclass of the implicit RK schemes called mono-implicit Runge-Kutta (MIRK) schemes [18], which have been found to be appropriate for BVODEs. The scheme is defined by the number of stages, s, the coefficients, $[v_r]_{r=1}^{s}$ and $[x_{rj}]_{j=1,r=1}^{r-1,s}$, and the weights $[b_r]_{r=1}^{s}$. The abscissa, $[c_r]_{r=1}^{s}$, are defined by the condition that $c_r = v_r + \sum_{j=1}^{r-1} x_{rj}$. We also define the following notation to be used later in this thesis:

$$\underline{c} = (c_1, c_2, \cdots, c_s)^T, \quad \underline{v} = (v_1, v_2, \cdots, v_s)^T,$$

$$\underline{b} = (b_1, b_2, \cdots, b_s)^T,$$

and

$$X = \begin{pmatrix} 0 & 0 & \cdots & \cdots & 0 \\ x_{21} & 0 & \cdots & \cdots & 0 \\ . & . & \cdots & \cdots & 0 \\ . & . & \cdots & \cdots & 0 \\ x_{s1} & x_{s2} & \cdots & x_{s,s-1} & 0 \end{pmatrix}.$$

The coefficients of a MIRK scheme are usually presented in a tableau [7] of the form given in Table 2.1. The general form of the MIRK scheme, which relates the solution approximation at t_i to that at t_{i+1}, is

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

where

$$K_r = f\left(t_i + c_r h, (1 - v_r) y_i + v_r y_{i+1} + h \sum_{j=1}^{r-1} x_{rj} K_j\right).$$

The application of the Runge-Kutta method on each subinterval together with the boundary conditions, leads to a large system of nonlinear equations. In the survey paper [10], the use of implicit Runge-Kutta methods for singularly perturbed problems is discussed, and it is shown that some classes of implicit Runge-Kutta formulae are both stable and accurate for such problems.

c_1	v_1	0	0	$\cdots$	$\cdots$	0
c_2	v_2	x_{21}	0	$\cdots$	$\cdots$	0
.	.	.	.	$\cdots$	$\cdots$	.
.	.	.	.	$\cdots$	$\cdots$	.
.	.	.	.	$\cdots$	$\cdots$	.
c_s	v_s	x_{s1}	x_{s2}	$\cdots$	$x_{s,s-1}$	0
		b_1	b_2	$\cdots$	$\cdots$	b_s

Table 2.1: Tableau of an s-stage MIRK scheme

2.2.5 Summary

1. The simple shooting method is straightforward to implement and it is good for relatively easy problems that may need to be solved many times. But it is unstable and the improved version, multiple shooting, can be ineffective for singularly perturbed problems.

2. By using basis functions which are high-degree splines (piecewise polynomials) the collocation approach can be superior to the finite difference approach [24].

3. The attraction of the finite difference scheme approach is its simplicity. The disadvantage of this approach is that the one step methods have only second-order accuracy at most. This is not very efficient for many applications. One approach to extend the finite difference methods is to define families of higher order one-step methods such as implicit Runge-Kutta schemes. Another exten-

sion is to define higher order finite difference schemes over several subintervals.

2.3 Numerical Software

Since the underlying mathematical theory for initial value problems is much better understood than for the boundary value case, and since the former problem class is much smaller than the latter, the state of software development for IVODEs is well ahead of that for BVODEs. For IVODEs, software based on multi-step and Runge-Kutta methods has been available for over 40 years. Thus, many IVODEs can be solved by existing codes with reasonable efficiency and reliability. On the other hand, the codes for BVODEs, although in wide use, are still less well developed.

2.3.1 Shooting Software

From our previous discussion, we know that the simple shooting method is based on an IVODE solver and a nonlinear equation solver. Thus, one couples a program module for solving nonlinear equations with a module that solves the corresponding IVODEs. Examples of IVODE solvers include RKSUITE [36] and CVODE [28] while an example of a nonlinear equation solver is KINSOL [29].

Multiple shooting has been implemented in many packages, some of which are part of standard libraries for numerical software. For example, MUSL [38] based on the multiple shooting method is designed for nonstiff linear BVODEs, while MUSN [38] also based on the multiple shooting method, is designed for nonstiff nonlinear BVODEs.

2.3.2 COLSYS - Software Based on Collocation

COLSYS (COLlocation for SYStems)[2] and COLNEW (a later version)[5] are designed to solve mixed order systems of nonlinear BVODEs. The method of spline collocation at Gaussian points is implemented using a B-spline basis in COLSYS and a monomial spline basis in COLNEW. A damped Newton's method is employed for the nonlinear iteration. The mesh refinement procedure in COLSYS automatically adapts the subinterval distribution to accommodate the solution behaviour based on a global error estimate for the continuous approximation. Approximate solutions are computed on a sequence of automatically selected meshes until a set of user-specified tolerances is satisfied.

2.3.3 TWPBVP - Software Based on Deferred Corrections

A-stable, symmetric mono-implicit Runge-Kutta (MIRK) schemes have been employed in a software package for the numerical solution of BVODEs, called TWPBVP [12]. TWPBVP is designed for the numerical solution of first order systems of nonlinear BVODEs. TWPBVP uses a deferred correction method based on mono-implicit Runge-Kutta formulas of orders 4, 6 and 8. The deferred correction approach involves first computing a 4th order solution approximation using the 4th order formula and then using the 6th order formula to generate a "correction" to the 4th order solution to make it 6th order. One then repeats this process using the 8th order formula to correct the 6th order solution to make it 8th order. The code controls a global error estimate of the solution approximations at the mesh points. Its original version did

not include an option for a continuous solution approximation. However more recent work [9] has provided TWPBVP with continuous solution approximations for graphical purposes only, i.e., a lower order interpolant is employed and no attempt is made to control the error in the continuous solution approximation.

2.3.4 MIRKDC - Software Based on Defect Control

MIRK software based on defect control called MIRKDC employs continuous MIRK (CMIRK) schemes to provide C^1 continuous approximate solutions.

The basic algorithm employed in MIRKDC uses MIRK formulas to discretize the ODE system; the resulting equations, together with the boundary conditions, give a nonlinear system for the solution approximations at the mesh points. Once this solution is obtained, a CMIRK scheme is used to provide a C^1 solution approximation over each subinterval for use in the computation of defect estimates, mesh redistribution, and initial guesses for subsequent Newton iterates. For the discrete schemes, Enright and Muir [18] employ symmetric MIRK schemes of orders two, four, and six. Because it is efficient to reuse the stages from the MIRK scheme within the CMIRK scheme, they embedded the MIRK scheme within the CMIRK scheme. In MIRKDC, to solve the nonlinear system, a combination of damped Newton iterations and fixed Jacobian iterations, with a scheme for switching between the two, is used. The Jacobian matrices arising from the nonlinear system have a special sparsity structure called almost block diagonal, and a software package, called COLROW [14], designed to handle this kind of structure, is employed.

Both the termination criterion for the overall computation and the mesh selection algorithm require an estimate of the maximum value of the defect on each subinterval. In MIRKDC, this is estimated by sampling the defect at several points within each subinterval. The MIRKDC algorithm controls defect estimates rather than global error estimates. The estimate of the maximum defect is required to satisfy a user provided tolerance.

2.3.5 Comparison

1. An advantage of the MIRK formulas over the collocation formulas is that, because the methods are being used in the BVODE context, the calculations on each subinterval are explicit and therefore can be implemented more efficiently.

2. For the MIRKDC software the major new aspects are the use of continuous MIRK formulas and the use of defect estimation rather than global error estimation for accuracy control and mesh selection.

3. The COLSYS/COLNEW software has proven to be competitive with the other robust software for solving BVODEs and to be particularly effective for difficult problems.

Chapter 3

Test Problems

3.1 Test Problems

We will make use of seven test problems in this thesis. Unless otherwise stated, no closed form solution is available.

The first test problem (TP1) is a nonlinear fluid flow problem from [1], page 23, Example 1.20:

$$\epsilon f''''(t) + f(t)f'''(t) + g(t)g'(t) = 0, \quad \epsilon g''(t) + f(t)g'(t) - f'(t)g(t) = 0, \tag{3.1}$$

with boundary conditions

$f(0) = f(1) = f'(0) = f'(1) = 0, \quad g(0) = -1, \quad g(1) = 1$; ϵ is a parameter.

The second problem (TP2) is a shock wave problem from [1], page 21, Example

1.17:

$$\epsilon A(t)u(t)u''(t) - \left[\frac{1}{2} + \frac{\gamma}{2} - \epsilon A'(t)\right] u(t)u'(t) + \frac{u'(t)}{u(t)} + \frac{A'(t)}{A(t)}\left[1 - \left(\frac{\gamma - 1}{2}\right)(u(t))^2\right] = 0, \tag{3.2}$$

where $A(t) = 1 + t^2$, $\gamma = 0.04$, and ϵ is a parameter. The boundary conditions are: $u(0) = 0.9129$, $u(1) = 0.3750$.

The third problem (TP3) is a fluid flow problem from [1], page 22, Example 1.18:

$$f'''(t) = \gamma^2 - 2f(t)f''(t) + (f'(t))^2 - (g(t))^2, \quad g''(t) = 2f'(t)g(t) - 2f(t)g'(t), \tag{3.3}$$

with $f(0) = f'(0) = 0$, $g(0) = 1, f'(\infty) = 0$, $g(\infty) = \gamma$; γ is a parameter.

The fourth problem (TP4) is from [31], test problem 17:

$$y''(t) = \frac{-3\epsilon y(t)}{(\epsilon + t^2)^2}, \tag{3.4}$$

with $y(0.1) = -y(-0.1) = \frac{0.1}{\sqrt{\epsilon + 0.01}}$; ϵ is a parameter. The true solution is $y(t) = \frac{t}{\sqrt{\epsilon + t^2}}$.

The fifth problem (TP5) is from [31], test problem 31:

$$y'(t) = \sin(\theta(t)), \quad \theta'(t) = M(t), \quad \epsilon M'(t) = -Q(t),$$

$$\epsilon Q'(t) = (y(t) - 1)\cos(\theta(t)) - M(t)\left[\sec(\theta(t)) + \epsilon Q(t)\tan(\theta(t))\right], \tag{3.5}$$

with $y(0) = y(1) = 0$, $M(0) = M(1) = 0$.

The sixth problem (TP6) is from [31], test problem 32:

$$y''''(t) = R\left(y'(t)y''(t) - y(t)y'''(t)\right), \tag{3.6}$$

with $y(0) = y'(0) = 0$, $y(1) = 1$, $y'(1) = 0$; R is a parameter.

The seventh problem (TP7) is from [31], test problem 20:

$$\epsilon y''(t) + (y'(t))^2 = 1, \tag{3.7}$$

with $y(0) = 1 + \epsilon \ln(\cosh(\frac{-0.745}{\epsilon}))$, $y(1) = 1 + \epsilon \ln(\cosh(\frac{0.255}{\epsilon}))$; ϵ is a parameter. The true solution is $y(t) = 1 + \epsilon \ln(\cosh(\frac{t-0.745}{\epsilon}))$.

3.2 Computer / Compiler Information

The computer which we used for CPU time testing has a 296.0 MHz SPARC-based CPU; the floating-point controller is Sun-4. The Fortran compiler is Sun WorkShop 6, update 2, FORTRAN 77, 5.3. The computer which we used to do the other experiments is an HP ProLiant DL380G2 with two 1.40GHz Pentium III processors. The compiler is gnu Fortran (gcc) 3.3.3.

Chapter 4

Modification of MIRKDC (I)— Software Modifications Based on Existing Modules and Formulas

In this chapter we describe modifications to the MIRKDC package which are based on previously developed software modules and formulas and involve straightforward implementations within the MIRKDC package. The emphasis for this portion of the thesis work is on the software engineering effort associated with modifying MIRKDC to add the new component, taking care to pay attention to the user interface, memory management issues, documentation, etc. As well, we have focused on analyzing the impact of each modification, especially in terms of its contribution to the overall execution time of the MIRKDC package. We consider modifications associated with computational derivative approximation, analytic derivative assessment, prob-

lem sensitivity assessment, new Runge-Kutta methods, and a preliminary global error estimate.

4.1 Computational Derivative Approximation

4.1.1 Introduction

For the ODE system, $y'(t) = f(t, y(t))$, MIRKDC requires the Jacobian (matrix) of partial derivatives of $f(t, y(t))$. Therefore, in addition to providing a subroutine which computes $f(t, y(t))$, the user also has to provide a subroutine which computes the Jacobian matrix. A similar situation holds for the boundary condition subroutines. When $f(t, y(t))$ is complicated, the development of a correct Jacobian subroutine is an error prone process for the user. However, the Jacobian can be approximated. One possibility is to use divided differencing to approximate the components of the Jacobian matrix. A basic formula for the approximation of the derivative of a given function, $f(x)$, is the simple forward divided difference formula which gives $f'(x) \approx (f(x + h) - f(x))/h$, where h is some carefully chosen increment, e.g., $h \approx \sqrt{eps}$, where *eps* is machine epsilon.

4.1.2 Description of the Software Modification

First, we chose an existing Fortran subroutine $fdjac1$ from the MINPACK software library (available at netlib [37]) which calculates a finite difference approximation to the Jacobian matrix, evaluated at $y(t)$, for a given vector function, $f(y(t))$, which

can be evaluated through a subroutine call. Since the parameter list for the function call in $fdjac1$ is different from that of the MIRKDC function, $fsub$, which evaluates $f(t, y(t))$, we modified $fdjac1$, to employ the $fsub$ parameter list, calling the new subroutine $fdjac1\_fsub$. To the parameter list for $fdjac1\_fsub$, we also added the independent variable t in order to be able to compute the approximate Jacobian of a non-autonomous ODE system, i.e., a system of the form $y'(t) = f(t, y(t))$ rather than $y'(t) = f(y(t))$. This does not add extra derivative computations because MIRKDC does not need the derivative of f with respect to t. We also wrote a new subroutine called $dfsub\_diff$, which can replace the call to the user defined $dfsub$ which provides the analytic Jacobian of $f(t, y(t))$. The $dfsub\_diff$ subroutine computes the approximate Jacobian by calling the $fdjac1\_fsub$ routine, mentioned above. We also modified the main MIRKDC subroutine by adding the parameter $approx\_jac$, which the user should set to be zero if $dfsub$ is provided, and otherwise, set to be one. We have also developed similar subroutines called $fdjac1\_gsub$ and $dgsub\_diff$, which compute the approximate Jacobian matrix for the boundary conditions.

4.1.3 Results and Discussion

For the numerical experiments we ran both versions of MIRKDC on several of the test problems, as indicated below. After making the above modifications, we did some testing of the $dfsub\_diff$ and $dgsub\_diff$ routines.

We now present and discuss numerical results for five test problems. In MIRKDC there is a parameter named 'method' which stands for the method order. We chose

method = 4, tolerance = 10^{-6}, the initial number of subintervals, Nsub = 10, ϵ = 0.04 for TP1, TP4 and TP5, $\epsilon = 1.4$ for TP2; and $\gamma = 1.4$ for TP3. The results given in Table 4.1 reveal that the differences in the entries of the approximate Jacobians computed by $dfsub\_diff$ and $dgsub\_diff$ range from 10^{-6} to 10^{-8}, except the linear test problem TP4. These differences are not significant; the differences in the resulting numerical solutions to the BVODEs are only about 10^{-12}(not shown in this thesis.)

Becasue the execution time for the $dgsub\_diff$ subroutine is too small to be timed, we focused only on timing $dfsub\_diff$. We chose both a non-autonomous ODE system (TP2) and an autonomous ODE system (TP1) for our test problems. Timing the cost of calling $dfsub\_diff$ and the overall cost for the main MIRKDC subroutine on TP1 and TP2 were done separately. Results are given in Table 4.2 and Table 4.3. We chose method = 4, tolerance = 10^{-3}, number of subintervals, Nsub = 10, and ϵ = 0.04 for TP1, $\epsilon = 1.4$ for TP2. For TP1, the call to $dfsub\_diff$ is about 4 times more than the call to $dfsub$ but the overall execution cost for MIRKDC when divided difference Jacobians are used is only about 5 percent more than MIRKDC when an analytic Jacobian is provided. For TP2, the call to $dfsub\_diff$ is about 3 times more than the call to $dfsub$ and the divided difference approximation of the Jacobian adds about 4 percent to the overall cost.

	TP1	TP2	TP3	TP4	TP5
max difference	5.96×10^{-7}	1.23×10^{-6}	2.72×10^{-8}	0	2.38×10^{-7}

Table 4.1: Absolute maximum difference between analytic and approximate Jacobian on TP1 - TP5, method = 4, Nsub = 10.

CPU time	TP1	TP2
$dfsub$	0.44×10^{-6}	0.22×10^{-5}
$dfsub\_diff$	1.97×10^{-6}	0.62×10^{-5}

Table 4.2: CPU time for $dfsub$ and $dfsub\_diff$ for TP1 and TP2; method = 4, tol = 10^{-3}, Nsub= 10, $\epsilon = 0.04$ for TP1, $\epsilon = 1.4$ for TP2; time in seconds.

		$TP1$			$TP2$	
ε	0.004	0.004	0.0004	0.04	0.04	0.04
Tol	10^{-6}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	10^{-11}
$MIRKDC$ *with* $dfsub$	0.28	1.22	2.98	0.06	0.24	0.40
$MIRKDC$ *with* $dfsub\_diff$	0.29	1.27	3.08	0.06	0.24	0.46

Table 4.3: CPU time for overall MIRKDC computation with $dfsub\_diff$ and $dfsub$ on TP1 and TP2; time in seconds.

4.2 Analytic Derivative Assessment

4.2.1 Introduction

For the ODE system, $y'(t) = f(t, y(t))$, we recall from the previous section that one of the options available to the user of the MIRKDC package is to provide a subroutine which gives the Jacobian of partial derivatives of $f(t, y(t))$. However it is common for users to make mistakes in the development of this subroutine. The idea we discuss in this section is that, prior to using the Jacobian subroutine provided by the user, MIRKDC should attempt to check if this subroutine is correct or not. Our basic approach is to compute finite difference approximations to the Jacobian (based on evaluations of $f(t, y(t))$ at the points of the initial mesh, using an initial solution approximation, and then compare these with evaluations of the analytic Jacobian provided by the user.

4.2.2 Description of Software Modification

We wrote a new subroutine, *dfsub_check*, which compares analytic and approximate Jacobian values. Its purpose is to compute the maximum difference between the analytic Jacobian computed by the user-supplied subroutine *dfsub* and the divided difference Jacobian computed by the MIRKDC subroutine *dfsub_diff*. If the maximum relative difference is greater than 1%, then the *dfsub_check* routine gives an output message which warns the user that the user-supplied subroutine, *dfsub*, may not be correct. The subroutine also will identify which entries of the Jacobian may

have errors. The final action of this subroutine is to set a flag which causes MIRKDC to terminate when an incorrect Jacobian has been encountered. We also developed a similar subroutine called *dgsub_check*, which checks the Jacobian matrix generated by the user-supplied subroutine *dgsub* for the boundary conditions.

We modified the main MIRKDC subroutine to add calls to the *dfsub_check* and *dgsub_check* subroutine. If $approx\_jac = 0$ (which implies that the user will provide *dfsub* and *dgsub*), then the main MIRKDC subroutine automatically calls *dfsub_check* and *dgsub_check* to perform a check on *dfsub* and *dgsub*.

4.2.3 Results and Discussion

We employed test problem TP1 with method = 4, tolerance = 10^{-3}, initial Nsub = 10, $\epsilon = 0.04$ and $approx\_jac = 0$. We introduced several deliberate errors within the *dfsub* and *dgsub* routines. Test results (not included in this thesis) show that *dfsub_check* and *dgsub_check* are able to detect typical errors in *dfsub* and *dgsub* routines. If there are errors in *dfsub* and/or *dgsub*, the *dfsub_check* and *dgsub_check* routines will indicate which components of the user defined subroutine may contain errors by outputting the row number and column number for each component.

4.3 Problem Sensitivity (Conditioning) Assessment

4.3.1 Introduction

As stated earlier, MIRKDC controls the defect or residual of the numerical solution. This is the amount by which the numerical solution fails to satisfy the ODEs and boundary conditions. An advantage of considering the defect is that it allows one to adopt a *backward error analysis* viewpoint for the error in the numerical approximation. When computation and control of the defect are coupled with an estimate of the sensitivity or conditioning of the BVODE, as we describe in this section, a backward error analysis provides us three kinds of information:

1. the knowledge that we have a numerical solution whose computed defect is less than the given user tolerance,

2. an indication of the sensitivity of the problem to small changes in its definition (i.e., an estimate of the magnitude of the conditioning constant).

3. an upper bound on the magnitude of the global error (equal to the product of the maximum defect and the conditioning constant).

The second of these is particularly important as it gives the application expert, who is using the MIRKDC software, an indication of the sensitivity of the solution to the accuracy of the application dependent parameters present in the mathematical model. That is, based on the size of the conditioning constant estimate, the user can tell how much small changes in these parameters will affect the solution.

If $u(t)$ is the numerical solution to (1.3), (1.4), then $\delta(t)$, the defect of $u(t)$ with respect to the ODEs, is given by

$$\delta(t) = u'(t) - f(t, u(t)),$$

and the defects, δ_a, and δ_b, of $u(a)$ and $u(b)$ with respect to the boundary conditions, are given by

$$\delta_a = g_a(u(a)), \quad \delta_b = g_b(u(b)).$$

The above equations can be rewritten as

$$u'(t) = f(t, u(t)) + \delta(t), \tag{4.1}$$

and

$$g_a(u(a)) - \delta_a = 0, \quad g_b(u(b)) - \delta_b = 0. \tag{4.2}$$

Equations (4.1) and (4.2) are important because they show us that $u(t)$, the *numerical* solution of (1.3), (1.4), is the *exact* solution to the "nearby" problem (4.1), (4.2). This perturbed problem differs from the original problem by the amounts $\delta(t)$, δ_a, and δ_b, which are usually small; in the case of MIRKDC these quantities are required to be less than the user tolerance.

However, even when $\delta(t)$, δ_a, and δ_b are small, the size of the difference between $u(t)$ and the true solution $y(t)$, i.e., the global error, depends on the conditioning of the problem. A well-conditioned problem can be described as follows: if there is a small change in f, g_a, or g_b arising in the BVODE problem definition, $y'(t) = f(t, y(t))$, $g_a(y(a)) = 0$, $g_b(y(b)) = 0$, then it will produce only a small change in the solution $y(t)$. Otherwise, it is called an ill-conditioned problem. Thus, if a stable, accurate numerical

method is applied to solve a well-conditioned BVODE, the numerical solution will be accurate; i.e., the difference between the numerical solution and the true solution will be small. The conditioning of a BVODE is characterized in terms of a *conditioning constant.*

We consider this in more detail; the following is based on [1] and [25]. Let us consider the linear BVODE,

$$y'(t) = A(t)y(t) + q(t), \quad a < t < b, \quad B_a y(\mathrm{a}) + \mathrm{B_b y(b)} = \beta, \tag{4.3}$$

with smooth coefficients $A(t) \in R^{n\times n}$, $q(t) \in R^n$, $y(t) \in R^n$, and $\beta \in R^n$, B_a,$B_b \in R^{n\times n}$. Under certain assumptions, the solution can be represented as [[1], pg. 111-112],

$$y(t) = Y(t)Q^{-1}\beta + \int_a^b G(t,z)q(z)dz, \tag{4.4}$$

where $Y(t)$ is a fundamental solution defined by

$$Y'(t) = A(t)Y(t), \quad a < t < b, \quad Y(a) = I,$$

$Q = B_a Y(a) + B_b Y(b)$, and $G(t,z)$ is the $n \times n$ Green's function [[1], pg. 94-95], given by

$$G(t,z) = \begin{cases} \Phi(t)B_a\Phi(a)\Phi^{-1}(z), & z \le t, \\ -\Phi(t)B_b\Phi(b)\Phi^{-1}(z), & z > t, \end{cases}$$

where $\Phi(t) = Y(t)Q^{-1}$.

Let $u(t)$ be the numerical solution to (4.3). In this case, the defects $\delta(t)$ and δ_{ab} are given by

$$\delta(t) \equiv u'(t) - (A(t)u(t) + q(t))$$

and

$$\delta_{ab} \equiv B_a u(a) + B_b u(b) - \beta.$$

If we rewrite these two equations, we observe that the approximate solution $u(t)$ satisfies

$$u'(t) = A(t)u(t) + q(t) + \delta(t), \quad B_a u(a) + B_b u(b) = \beta + \delta_{ab}. \tag{4.5}$$

That is $u(t)$ *is the exact solution to (4.5).* Of course $y(t)$ satisfies

$$y'(t) = A(t)y(t) + q(t), \quad B_a y(a) + B_b y(b) = \beta, \tag{4.6}$$

and we have seen that $y(t)$ is given by expression (4.4). Since $u(t)$ is the exact solution of equation (4.5), we can also use (4.4) to write

$$u(t) = Y(t)Q^{-1}(\beta + \delta_{ab}) + \int_a^b G(t,z)(q(z) + \delta(z))dz. \tag{4.7}$$

Therefore, from (4.4) and (4.7) we have

$$u(t) - y(t) = Y(t)Q^{-1}\delta_{ab} + \int_a^b G(t,z)\delta(z)dz. \tag{4.8}$$

Taking $\|\cdot\|_\infty$ norms on each side of equation (4.8) and using standard norm inequalities, we get

$$\begin{aligned}
\max_{a\le t\le b} \|u(t) - y(t)\|_\infty &= \max_{a\le t\le b} \|Y(t)Q^{-1}\delta_{ab} + \int_a^b G(t,z)\delta(z)dz\|_\infty \\
&\le \max_{a\le t\le b} \|Y(t)Q^{-1}\|_\infty \|\delta_{ab}\|_\infty + \max_{a\le t\le b} \int_a^b \|G(t,z)\|_\infty \|\delta(z)\|_\infty dz \\
&\le \max_{a\le t\le b} \|Y(t)Q^{-1}\|_\infty \|\delta_{ab}\|_\infty + \max_{a\le t\le b} \|\delta(t)\|_\infty \int_a^b \|G(t,z)\|_\infty dz \\
&\le \max\left(\max_{a\le t\le b} \|\delta(t)\|_\infty, \|\delta_{ab}\|_\infty\right) \max_{a\le t\le b} \left(\|Y(t)Q^{-1}\|_\infty + \int_a^b \|G(t,z)\|_\infty dz\right)
\end{aligned}$$

$$= \kappa \cdot \max\left(\max_{a\le t\le b} \|\delta(t)\|_\infty, \|\delta_{ab}\|_\infty\right)$$

where

$$\kappa \equiv \max_{a\le t\le b}\left(\int_a^b \|G(t,z)\|_\infty dz + \|Y(t)Q^{-1}\|_\infty\right). \tag{4.9}$$

Thus $e(t) \equiv u(t) - y(t)$ satisfies

$$\max_{a\le t\le b} \|e(t)\|_\infty \le \kappa \cdot \max\left(\max_{a\le t\le b} \|\delta(t)\|_\infty, \|\delta_{ab}\|_\infty\right). \tag{4.10}$$

Equation (4.10) says that the error is bounded by the conditioning constant, κ, times the maximum of the defects of the ODE and the boundary condition. A given BVODE is well-conditioned if κ is of moderate size.

While (4.9) does give us an expression for the conditioning constant, we need a more practical way of estimating κ. From [[1], pg. 202], let

$$M^{-1} \equiv \begin{pmatrix} G(t_1,t_2) & \cdots & G(t_1,t_{N+1}) & Y(t_1)Q^{-1} \\ \vdots & & \ddots & \vdots \\ G(t_{N+1},t_2) & \cdots & G(t_{N+1},t_{N+1}) & Y(t_{N+1})Q^{-1} \end{pmatrix},$$

where $\{t_j\}_{j=1}^{N+1}$ is a mesh of $N+1$ points which partitions the problem interval [a,b] (with $t_1 = a$ and $t_{N+1} = b$). When a Runge-Kutta method is used to discretize the linear ODE system (4.3), the resultant equations, together with the boundary conditions, form a linear system with coefficient matrix

$$A = \begin{pmatrix} S_1 & R_1 & & & \\ & S_2 & R_2 & & \\ & & \ddots & & \\ & & & S_N & R_N \\ B_a & & & & B_b \end{pmatrix},$$

where S_j, R_j are $n \times n$ matrices. Also as in [[1],pg. 202], let D be the block diagonal matrix given by

$$D = \begin{pmatrix} R_1^{-1} & & & & \\ & R_2^{-1} & & & \\ & & \ddots & & \\ & & & R_N^{-1} & \\ & & & & I \end{pmatrix}.$$

Define $h_i = t_{i+1} - t_i$ and $h = \max_{i=1}^{N} h_i$. Then Theorem 5.38 of [1] states that

$$\|M^{-1}D\|_\infty = \max_{1 \le i \le N+1} \left(\sum_{j=1}^{N} h_j \|G(t_i, t_{j+1})\|_\infty + \|Y(t_i)Q^{-1}\|_\infty + O(h) \right).$$

Assuming sufficiently small h and approximating the sum by an integral, we have

$$\|M^{-1}D\|_\infty \approx \max_{a \le t \le b} \left(\|Y(t)Q^{-1}\|_\infty + \int_a^b \|G(t,z)\|_\infty dz \right).$$

Finally, Theorem 5.38 of [1] also states that for a sufficiently fine mesh, $\|M^{-1}D\|_\infty \approx \|A^{-1}\|_\infty$. This gives

$$\|A^{-1}\|_\infty \approx \max_{a \le t \le b} \left(\|Y(t)Q^{-1}\|_\infty + \int_a^b \|G(t,z)\|_\infty dz \right),$$

and we note that the right hand side of this equation is κ. Thus, we have $\|A^{-1}\|_\infty \approx \kappa$.

The matrix A arises in the computation of the numerical solution; we need to setup and solve linear systems having A as the coefficient matrix, and during this solution process it is possible to estimate $\|A^{-1}\|_\infty$, as we describe in the next subsection. The algorithm described in [25] also takes into account that the error and the defect are in fact scaled, but we do not include these details here.

4.3.2 Description of the Software Modification

The original MIRKDC used COLROW [14] for the linear equations which arise in the computation. In the new version of MIRKDC, we employ a slightly updated version of the COLROW [19] subroutine, and also a modification of the subroutine BSPCND [20] called BSPCNDMAX [22]. The new COLROW has a parameter called 'job' which can be used to specify which of $Ax = b$ or $A^T x = b$ is to be solved. BSPCND uses this feature to compute an estimate of the condition number ($cond(A) = \|A\|\|A^{-1}\|$) in the 1-norm. BSPCNDMAX is a modification of this which uses the infinity or max norm.

In the BSPCNDMAX routine, $\|A\|_\infty$ is computed by calling the ABDNRMMAX [19] routine. Then, a factorization of A is performed using the subroutine CRDCMP [19] from COLROW, which decomposes the matrix A using modified alternate row and column elimination with partial pivoting. Then, an estimate of $\|A^{-1}\|_\infty$ is obtained by using the DONEST [19] and CRSLVE [19] subroutines. The DONEST subroutine estimates the norm of a matrix. CRSLVE will solve a linear system once A is decomposed. We modified BSPCNDMAX so that it does not compute $\|A\|_\infty$ and $cond(A) = \|A\|_\infty\|A^{-1}\|_\infty$ but rather only computes and returns an estimate of $\|A^{-1}\|_\infty$, since this is all that is needed for the estimate of κ, as explained in the previous section.

We added a new parameter, called *cond_check*, to the parameter list of MIRKDC. If *cond_check* $= 0$, MIRKDC will not compute the conditioning constant estimate. If *cond_check* $= 1$, MIRKDC will compute an estimate of κ.

CPU Time	$Nsub = 100$	$Nsub = 500$	$Nsub = 1000$
calling COLROW	0.02999	0.149999	0.319999
calling BSPCNDMAX and *CRSLVE*	0.0499991	0.259999	0.529999

Table 4.4: Execution times for TP1 for COLROW and BSPCNDMAX/CRSLVE.

CPU Time	$TP1$	$TP3$	$TP4$
without κ *estimate*	0.10	0.04	0.01
with κ *estimate*	0.13	0.07	0.01

Table 4.5: Execution times for overall MIRKDC computation with and without conditioning constant estimate for TP1, TP3, and TP4, method = 4, tol = 10^{-6}, $\epsilon = 0.04$, $\gamma = 1.4$, initial Nsub = 2.

4.3.3 Results and Discussion

Using TP1, we did some tests of the CPU time for calls to the BSPCNDMAX, CRSLVE, and COLROW subroutines. We chose method = 4, tolerance = 10^{-6} and $\epsilon = 0.04$, initial Nsub = 100, 500, and 1000. We also did tests to measure the CPU time for the overall execution of MIRKDC on TP1, TP3 and TP4, using the conditioning constant estimation option.

The test results are shown in Table 4.4 and Table 4.5. From Table 4.4, we see that the CPU time for calling BSPCNDMAX and CRSLVE together is approximately 1.5 to 2 times more than that of calling COLROW alone. Table 4.5 shows that for TP4 there is no additional cost for the κ estimate, for TP1 there is about a 30% cost, and

for TP3 the cost is about 75%.

In the new version of MIRKDC we compute an estimate of $\|A^{-1}\|_\infty$ whenever we set up and factor a new A matrix. This makes estimates of κ available throughout the computation and these estimates could be used to monitor and guide the computation. A much less expensive alternative would be to only compute the estimate of $\|A^{-1}\|_\infty$ after an acceptable solution has been obtained. This would involve a few extra back solves involving the final A matrix and would add very little to the overall cost, while providing the user with an estimate of κ for the BVODE.

4.4 Improved Runge-Kutta Methods

4.4.1 Introduction

Recall that the general form of a discrete MIRK scheme is as given in (2.1), with the coefficients usually represented in the tableau given in Table 2.1. The general form of a CMIRK scheme on the subinterval $[t_i, t_{i+1}]$ is

$$u(t_i + \theta h_i) = y_i + h_i \sum_{r=1}^{s^*} b_r(\theta) K_r,$$

where $K_r = f(t_i + c_r h_i, (1 - v_r)y_i + v_r y_{i+1} + h \sum_{j=1}^{r-1} x_{rj} K_j)$, and $b_r(\theta)$ is a polynomial in θ, $\theta \in [0, 1]$, and s^* is the number of stages. The general form of the corresponding tableau is

$\underline{c}$	$\underline{v}$	X
		$\underline{b}(\theta)^T$

where $\underline{b}(\theta) = [b_1(\theta), b_2(\theta), \cdots, b_{s^*}(\theta)]^T$. A pth order method has a local error that is $O(h^{p+1})$. A MIRK or CMIRK method with stage order q has coefficients which satisfy

$$X\underline{c}^{l-1} + \frac{\underline{v}}{l} = \frac{\underline{c}^l}{l}, \quad l = 1, \cdots, q.$$

In the original MIRKDC software, the discrete 3-stage, 4th order, stage order 3, MIRK scheme has the tableau that is shown in Table 4.6. This scheme can be embedded in a 4-stage, stage order 3, CMIRK scheme which provides the interpolant for the discrete solution values. The tableau for this continuous scheme is given in Table 4.7, where

$$b_1(\theta) = -\frac{\theta(2\theta-3)(3\theta^2-3\theta+2)}{6}, \quad b_2(\theta) = -\frac{\theta^2(12\theta^2-20\theta+9)}{6},$$
$$b_3(\theta) = \frac{2\theta^2(6\theta^2-14\theta+9)}{3}, \quad b_4(\theta) = -\frac{16\theta^2(\theta-1)^2}{3}.$$

0	0	0	0	0
1	1	0	0	0
1/2	1/2	1/8	−1/8	0
		1/6	1/6	2/3

Table 4.6: Tableau for 3-stage, 4th order, stage order 3, MIRK scheme.

In the paper[21], Muir derived a new optimized 4th order CMIRK scheme, containing the discrete scheme of Table 4.6. It has the tableau shown in Table 4.8. where

$$b_1(\theta) = -\frac{\theta(3\theta-4)(5\theta^2-6\theta+3)}{12}, \quad b_2(\theta) = \frac{\theta^2(5\theta^2-6\theta+2)}{6},$$
$$b_3(\theta) = -\frac{2\theta^2(3\theta-2)(5\theta-6)}{3}, \quad b_4(\theta) = \frac{125\theta^2(\theta-1)^2}{12}.$$

0	0	0	0	0	0
1	1	0	0	0	0
1/2	1/2	1/8	−1/8	0	0
3/4	27/32	3/64	−9/64	0	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$

Table 4.7: Tableau for original 4-stage, 4th order, stage order 3, CMIRK scheme.

0	0	0	0	0	0
1	1	0	0	0	0
1/2	1/2	1/8	−1/8	0	0
2/5	2/5	17/125	−13/125	−4/125	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$

Table 4.8: Tableau for new 4-stage, 4th order, stage order 3 CMIRK scheme.

In the new version of MIRKDC, we replaced the original CMIRK scheme of Table 4.7 with the new CMIRK scheme of Table 4.8.

Muir[21] also describes an optimized 5-stage discrete, sixth order, stage order 3, embedded in a 6th order, 8-stage CMIRK scheme and has reported test results for these schemes which showed that the new schemes led to significant improvements in overall performance. These had already been added to MIRKDC; we simply modified

the code to make these formulas the default 6th order schemes.

4.4.2 Results and Discussion

The purpose of our experiments is to investigate the impact of the new 4th order interpolant. We did some testing using the two test problems TP1 and TP2. The testing was done with a tolerance of 10^{-9}, ϵ equal to 0.001 and 0.0001 for TP1, ϵ equal to 0.1 and 0.01 for TP2 ($\gamma = 1.4$), and a uniform initial mesh of 2 subintervals. The results are given in Table 4.9 and Table 4.11 for TP1, and Table 4.10 and Table 4.12 for TP2.

From Tables 4.9 - Table 4.12, we see that the improved interpolant leads to a defect of comparable size using fewer mesh points. The improvement is more significant for the more difficult problems. The fact that the code uses fewer mesh points leads to significant savings in execution time.

4.5 Preliminary Global Error Indicator

4.5.1 Introduction and Description of Software Modification

The primary approach used in MIRKDC for monitoring the quality of the numerical solution is defect control. MIRKDC will normally return a solution whose defect satisfies the user tolerance. After this computation is completed, it might be considered useful to also have a rough, low cost indicator of the global error of the numerical solution.

Old Interpolant		New Interpolant	
Nsub	Def Est	Nsub	Def Est
2	6.89	2	2.03
8	11.49	8	1.21
16	$7.64 * 10^{-2}$	16	$4.72 * 10^{-2}$
64	$3.96 * 10^{-4}$	64	$9.86 * 10^{-4}$
256	$5.70 * 10^{-9}$	256	$5.53 * 10^{-7}$
844	$3.80 * 10^{-10}$	699	$7.57 * 10^{-9}$
1069	$7.73 * 10^{-10}$	853	$6.32 * 10^{-10}$

Table 4.9: Comparison of MIRKDC mesh sizes and defect estimates for TP1, tol = 10^{-9}, method = 4, $\epsilon = 0.001$, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8.

Over the last 20 years, there have been a number of papers, see, e.g.,[8], [23],[27] which have considered the development of global error estimation techniques. The paper [27] classified, described and compared 13 ways to estimate global error. We note that almost all of these techniques are applied only to initial value problems. In the text book [1], global error estimation strategies for BVODEs are discussed in section 9.3.

In the new version of MIRKDC, the preliminary technique we use for obtaining a global error estimate involves calculating a second numerical solution to the same problem using a doubled mesh, i.e., we repeat the final calculation using a mesh in

Old Interpolant		New Interpolant	
Nsub	Def Est	Nsub	Def Est
2	2.44	2	1.05
4	$5.41 * 10^{-2}$	4	$1.86 * 10^{-2}$
16	$5.38 * 10^{-4}$	16	$4.07 * 10^{-4}$
64	$9.19 * 10^{-7}$	64	$5.76 * 10^{-7}$
256	$2.45 * 10^{-9}$	256	$1.56 * 10^{-9}$
375	$4.23 * 10^{-10}$	340	$3.88 * 10^{-10}$

Table 4.10: Comparison of MIRKDC mesh sizes and defect estimates for TP2, tol = 10^{-9}, method = 4, $\epsilon = 0.1$, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8.

which each subinterval is split in half. We compare the numerical solutions from the original mesh and the doubled mesh at the mesh points of the original mesh to get a global error estimate for the original numerical solution. In order to compare this estimate with the true error, we choose a test problem, TP7, having a closed form solution.

4.5.2 Results and Discussion

We employ a tolerance set of 10^{-6}. We use an absolute measure of the error $|Y_{Nsub} - Y_{2*Nsub}|$, where Y_{Nsub} is the numerical solution from the original mesh, Y_{2*Nsub} is the numerical solution from the doubled mesh. In Figures 4.1 - 4.2, we plot the errors

Old Interpolant		New Interpolant	
Nsub	Def Est	Nsub	Def Est
2	18.84	2	15.56
32	2.11	32	0.43
64	$5.96 * 10^{-2}$	64	$2.66 * 10^{-2}$
256	$3.75 * 10^{-5}$	256	$3.63 * 10^{-5}$
1024	$8.38 * 10^{-9}$	1017	$3.22 * 10^{-8}$
1638	$2.05 * 10^{-9}$	1334	$1.29 * 10^{-9}$
1801	$2.02 * 10^{-9}$	1467	$1.09 * 10^{-9}$
1981	$2.85 * 10^{-10}$	1613	$4.32 * 10^{-10}$

Table 4.11: Comparison of MIRKDC mesh sizes and defect estimates for TP1, tol = 10^{-9}, method = 4, $\epsilon = 0.0001$, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8.

for solution components y_1 and y_2. From the results, we see that the error estimate is within a factor of 10 of the true error.

In order to examine the cost of computing this global error estimation in MIRKDC, we measured the CPU time for MIRKDC with and without the global error indicator. The results, shown in Table 4.13, indicate that the CPU time for computing the global error estimate increases the cost of the computation by a factor of 2 or 3. For large Nsub the factor will be bigger.

While the global error estimates in this test are of reasonable quality, the cost of

Old Interpolant		New Interpolant	
Nsub	Def Est	Nsub	Def Est
2	–	2	–
4	–	4	–
8	–	8	–
16	–	16	–
32	–	32	–
64	$2.95 * 10^{-2}$	64	$1.27 * 10^{-2}$
256	$4.64 * 10^{-6}$	256	$6.38 * 10^{-6}$
929	$1.32 * 10^{-8}$	818	$7.74 * 10^{-9}$
1205	$9.67 * 10^{-10}$	1035	$1.49 * 10^{-9}$
		1138	$3.07 * 10^{-10}$

Table 4.12: Comparison of MIRKDC mesh sizes and defect estimates for TP2, tol = 10^{-9}, method = 4, $\epsilon = 0.01$, for old interpolant from original MIRKDC (Table 4.7) and new interpolant from Table 4.8.

obtaining them is too high. However, these results can serve as a baseline for future work in which the tradeoffs between the quality of the global error estimate and its cost are explored.

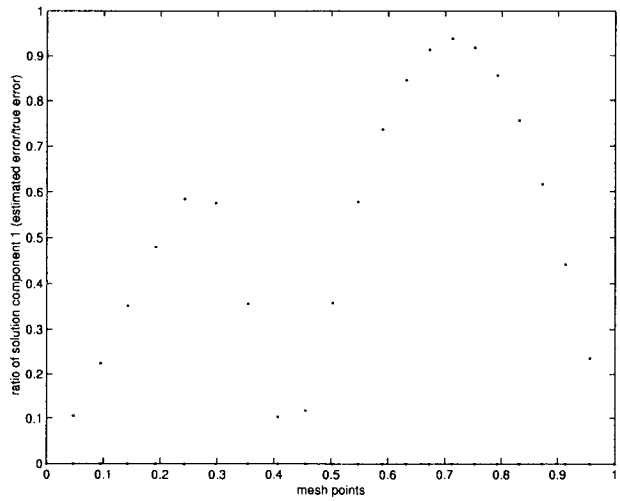

Figure 4.1: Comparison of ratio of estimate error and true error of solution component 1 on TP7, method = 4, tol = 10^{-6}.

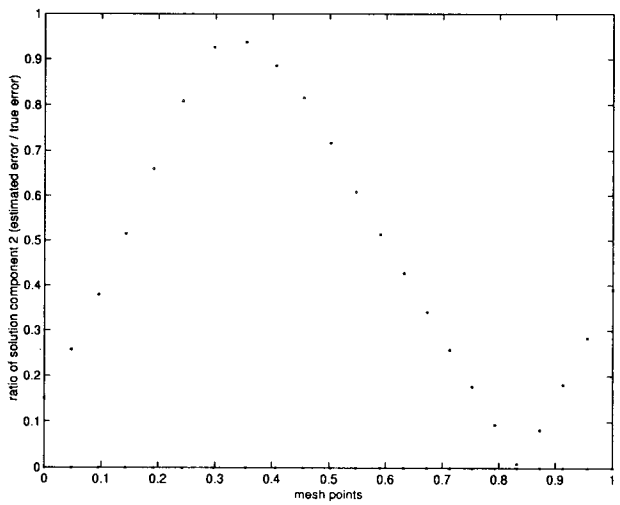

Figure 4.2: Comparison of ratio of estimate error and true error of solution component 2 on TP7, method = 4, tol = 10^{-6}.

(method, tol, Nsub, ϵ)	With global error indicator	Without global error indicator
$(4, 10^{-6}, 10, 0.5)$	0.03	0.01
$(4, 10^{-6}, 20, 0.06)$	0.11	0.04
$(6, 10^{-9}, 20, 0.5)$	0.04	0.02
$(6, 10^{-9}, 20, 0.06)$	0.17	0.07

Table 4.13: Comparison of CPU times for MIRKDC global error estimation.

Chapter 5

Modification of MIRKDC (II) — Design and Analysis of Defect Control Strategies

5.1 Introduction

In this chapter we consider techniques which can lead to improved control of the defect of the approximate solution to a BVODE system. The approach is based on considering special kinds of interpolants for the representation of the continuous solution approximation.

The paper[15] considers specially constructed continuous solution approximations which lead to defects having the special form,

$$\delta(t) = \varphi_p'(\theta)\kappa_p h^p + \kappa_{p+1}(\theta)h^{p+1} + O(h^{p+2}), \tag{5.1}$$

where $\theta = (t - t_{i-1})/h$, and $\varphi_p(\theta)$ is a polynomial in θ whose coefficients depend on the method but not on the problem and the step size h. κ_p depends only on the test problem, and p is the order of the RK method. In this case, the defect is said to be *asymptotically correct* because as $h \to 0$, the maximum of the defect on each subinterval is located at the maximum of the polynomial $\varphi_p'(\theta)$, which can be determined ahead of time. In order for the defect estimate to be asymptotically correct, the h^p term must be significantly bigger than the h^{p+1} term. If $\kappa_{p+1}(\theta)$ is comparable in size to $\varphi_p'(\theta)\kappa_p$ and h is not sufficiently small, then we cannot in practice expect the maximum defect to be at the maximum of $\varphi_p'(\theta)$.

We can sample the defect at a set of points in the subinterval and use the maximum of these samples as an estimate of maximum defect. This is called 'relaxed defect control'; it is reliable and effective for many practical problems. However, as explained above, we can employ one-point sampling of the defect provided we employ a special kind of interpolant, which leads to an asymptotically correct form for the defect. This is called 'strict defect control'.

5.2 Description of the Software Modification

We have designed new interpolants (based on CMIRK schemes) that lead to asymptotically correct defects, implemented them as an option within MIRKDC, and then performed experiments to test our new schemes to see if they in fact give asymptotically correct defects and to see what the implications are for the execution time.

It turns out that the second order continuous scheme already employed in MIRKDC

leads to an asymptotically correct defect. We will consider this in more detail shortly. We have added new schemes for orders 4 and 6. We have also added an option called 'defect_control' to the parameter list for the new version of MIRKDC to let the user choose the type of defect control: 'relaxed defect control', or a modification of 'strict defect control' which we call 'safe-guarded strict defect control', to be described momentarily. When the user selects 'relaxed defect control', we choose two-point sampling for the defect estimate using the 4th and 6th order continuous schemes discussed in the previous chapter. When the user selects 'safe-guarded strict defect control', we will use new 4th and 6th order continuous Runge-Kutta schemes which yield a defect with the form given in (5.1). The basic idea for 'safe-guarded strict defect control' is that the step size h should be small enough so that we can use one-point sampling; otherwise we will employ extra sample points in order to estimate the maximum defect on each subinterval. If we are not in the asymptotic region because the step size h is not small enough (we have experimentally determined a threshold of 0.01 for the methods of orders 2 and 4, and a threshold of 0.002 for the method of order 6), the new version of MIRKDC will do two extra samples (at $\theta = 0.55$ and $\theta = 0.47$ for the 2nd order method, at $\theta = 0.54$ and $\theta = 0.56$ for the 4th order method, and at $\theta = 0.22$ and $\theta = 0.81$ for the 6th order method) determined through numerical experiments (see Tables 5.1 and 5.2). Otherwise, we will only do one point sampling at $\theta = 0.5$ for the 2nd order method, at $\theta = 0.5453$ for the 4th order method and at $\theta = 0.5$ for the 6th order method. These are the points where the maximums of the $\varphi_2'(\theta)$, $\varphi_4'(\theta)$, and $\varphi_6'(\theta)$ polynomials occur. (We will consider the details shortly.)

Initial Nsub	TP1	TP2	TP3
100	$0.54 - 0.55$	0.55	0.55
50	$0.54 - 0.55$	$0.54 - 0.55$	0.55
40	$0.54 - 0.55$	$0.55 - 0.56$	0.55
30	$0.54 - 0.55$	$0.54 - 0.57$	$0.54 - 0.55$
20	$0.53 - 0.56$	$0.53 - 0.58$	$0.54 - 0.55$
10	$0.42 - 0.66$	$0.55 - 0.59$	$0.54 - 0.55$
8	$0.44 - 0.64$	$0.55 - 0.6$	$0.54 - 0.55$

Table 5.1: The locations of the maximum defect for different numbers of subintervals, method = 4, tol = 10^{-9}, TP1 - TP3.

5.3 Continuous Runge-Kutta Schemes

5.3.1 A Continuous Runge-Kutta Scheme of 2nd Order

The continuous scheme for second order has the form

$$u(t) = u(t_i + \theta h_i) = y_i + h_i \left(b_1(\theta) K_1 + b_2(\theta) K_2 \right),$$

where

$$K_1 = f(t_i, y_i), \quad K_2 = f(t_{i+1}, y_{i+1}), \quad \text{and} \quad b_1(\theta) = \theta(1 - \frac{\theta}{2}), \quad b_2(\theta) = \frac{\theta^2}{2}.$$

This scheme has coefficients which satisfy

$$\underline{c} = X\underline{e} + \underline{v} \quad \text{and} \quad \frac{\underline{c}^2}{2} = X\underline{c} + \frac{\underline{v}}{2},$$

Initial Nsub	TP4	TP5	TP6
100	$0.54 - 0.55$	$0.54 - 0.56$	$0.54 - 0.55$
50	$0.54 - 0.55$	$0.54 - 0.55$	$0.54 - 0.55$
40	$0.54 - 0.55$	$0.54 - 0.55$	$0.54 - 0.55$
30	$0.53 - 0.56$	$0.53 - 0.57$	$0.54 - 0.57$
20	$0.51 - 0.57$	$0.52 - 0.57$	$0.53 - 0.58$
10	$0.49 - 0.93$	$0.42 - 0.69$	$0.29 - 0.59$
8	$0.48 - 0.93$	$0.43 - 0.67$	$0.54 - 0.60$

Table 5.2: The locations of the maximum defect for different numbers of subintervals, method = 4, tol = 10^{-9}, for TP4 - TP6.

where $\underline{e} = (1, 1, \cdots, 1)^T$, and it therefore has stage order 2. Letting $y(t)$ be the true solution, we know from standard theory for Runge-Kutta methods, [7], that a local error expansion for this scheme has the form

$$y(t) - u(t) = C_3 \left(\underline{b}^T(\theta)\underline{c}^2 - \frac{\theta^3}{3} \right) h^3 + O(h^4),$$

where h is the size of the step or subinterval,

$$\underline{b}^T(\theta) = \begin{pmatrix} b_1(\theta) \\ \\ b_2(\theta) \end{pmatrix}, \quad \underline{c} = \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix},$$

and C_3 is problem dependent but not dependent on θ. Substituting the above values into $\phi_2(\theta) = \underline{b}^T(\theta)\underline{c}^2 - \frac{\theta^3}{3}$ gives $\phi_2(\theta) = \frac{\theta^2}{2} - \frac{\theta^3}{3}$ and then $\phi_2'(\theta) = \theta - \theta^2$, with a maximum at $\theta = \frac{1}{2}$. We will explain further in the next section how the above form of the error leads to the specific form of the defect in (5.1).

5.3.2 A New Continuous 4th Order Scheme

A standard continuous Runge-Kutta scheme for fourth order requires 4 stages. A continuous Runge-Kutta scheme which gives a maximum defect having the special form will be presented here; it uses 5 stages. The tableau of coefficients for this scheme which we denote by CMIRK543 is shown in Table 5.3. It has the 3-stage discrete 4th order MIRK scheme of Table 4.6 embedded within it. The continuous solution approximation $u(t)$ based on this CMIRK scheme has the form

$$u(t) = y_i + h_i \sum_{r=1}^{5} b_r(\theta) K_r. \tag{5.2}$$

We now discuss the derivation of this scheme. Let

0	0	0	0	0	0	0
1	1	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$	$-\frac{1}{8}$	0	0	0
c_4	v_4	x_{41}	x_{42}	x_{43}	0	0
c_5	v_5	x_{51}	x_{52}	x_{53}	x_{54}	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$	$b_5(\theta)$

Table 5.3: General form for the tableau of 5-stage, 4th order, stage order 3 CMIRK scheme with the discrete 3-stage, 4th order, stage order 3 MIRK scheme of Table 4.6 embedded.

$$\underline{c} = (0, 1, 1/2, c_4, c_5)^T, \quad \underline{v} = (0, 1, 1/2, v_4, v_5)^T,$$

$$\underline{e} = (1, 1, 1, 1, 1)^T, \quad \underline{b}(\theta) = (b_1(\theta), b_2(\theta), b_3(\theta), b_4(\theta), b_5(\theta))^T,$$

and

$$X = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & -\frac{1}{8} & 0 & 0 & 0 \\ x_{41} & x_{42} & x_{43} & 0 & 0 \\ x_{51} & x_{52} & x_{53} & x_{54} & 0 \end{pmatrix}.$$

The embedded discrete method is said to have stage order 3. This means that

$$X\underline{e} + \underline{v} = \underline{c}, \quad X\underline{c} + \frac{\underline{v}}{2} = \frac{\underline{c}^2}{2}, \quad X\underline{c}^2 + \frac{\underline{v}}{3} = \frac{\underline{c}^3}{3}.$$

Here $\underline{c}^n$ means that the every element of $\underline{c}$ should be taken to the nth power. We will also require the extra stages 4 and 5 to satisfy these stage order conditions. Requiring that these stage order conditions be satisfied allows us to fix $x_{41}, x_{42}, x_{43}, x_{52}, x_{53}, x_{54}$ in terms of the free variables $c_4, c_5, v_4, v_5, x_{51}$.

The 4th order continuous order conditions for a Runge-Kutta method with stage order 3 are

$$\underline{b}^T\underline{e} = \theta, \quad \underline{b}^T(\theta)\underline{c} = \frac{\theta^2}{2}, \quad \underline{b}^T(\theta)\underline{c}^2 = \frac{\theta^3}{3}, \quad \underline{b}^T(\theta)\underline{c}^3 = \frac{\theta^4}{4}.$$

The two order conditions for 5th order are

$$\underline{b}^T(\theta)\underline{c}^4 = \frac{\theta^5}{5}, \quad \underline{b}^T(\theta)\left(X\underline{c}^3 + \frac{\underline{v}}{4}\right) = \frac{\theta^5}{20}.$$

By satisfying the first 5th order condition above in addition to all the 4th order conditions, we get five sets of equations for the coefficients of the five weight polynomial $b_1(\theta), \cdots, b_5(\theta)$. This leaves us with the 5 free coefficients $c_4, c_5, v_4, v_5, x_{51}$. We arbitrarily chose: $c_4 = v_4 = \frac{1}{10}$, $c_5 = v_5 = \frac{9}{10}$, $x_{51} = \frac{3}{40}$. (A more optimal choice of

these free parameters could lead to a better method with smaller error coefficients but this is left for future work.) This leads to the tableau of coefficients for the CMIRK543 scheme shown in Table 5.4. The weight polynomials are

$$b_1(\theta) = \frac{\theta}{54}(8\theta - 9)(30\theta^3 - 60\theta^2 + 37\theta - 6), \quad b_2(\theta) = \frac{\theta^2}{54}(-27 + 236\theta - 450\theta^2 + 240\theta^3),$$

$$b_3(\theta) = \frac{\theta^2}{24}(-27 + 218\theta - 300\theta^2 + 120\theta^3), \quad b_4(\theta) = -\frac{125}{432}\theta^2(-27 + 74\theta - 72\theta^2 + 24\theta^3),$$

$$b_5(\theta) = -\frac{125}{432}\theta^2(-3 + 26\theta - 48\theta^2 + 24\theta^3).$$

0	0	0	0	0	0	0
1	1	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$	$-\frac{1}{8}$	0	0	0
$\frac{1}{10}$	$\frac{1}{10}$	$\frac{69}{1000}$	$-\frac{21}{1000}$	$-\frac{6}{125}$	0	0
$\frac{9}{10}$	$\frac{9}{10}$	$\frac{3}{40}$	$-\frac{3}{40}$	$\frac{3}{40}$	$-\frac{3}{40}$	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$	$b_5(\theta)$

Table 5.4: Tableau of 5-stage, 4th order, stage order 3 CMIRK scheme which gives an asymptotically correct defect estimate.

We conclude this section by considering how this continuous scheme leads to a defect that has the form (5.1).(The basic idea applies generally to pth order schemes as well.) Suppose that $y(t)$ is the true solution of the BVODE. Then we know that (according to standard theory for Runge-Kutta methods, [7],)

$$y(t) - u(t) = C_1\left(\underline{b}^T(\theta)e - \theta\right)h + C_2\left(\underline{b}^T(\theta)\underline{c} - \frac{\theta^2}{2}\right)h^2$$

$$+C_3\left(\underline{b}^T(\theta)\underline{c}^2-\frac{\theta^3}{3}\right)h^3+C_4\left(\underline{b}^T(\theta)\underline{c}^3-\frac{\theta^4}{4}\right)h^4$$
$$+\left(C_{51}\left(\underline{b}^T(\theta)-\frac{\theta^5}{5}\right)+C_{52}\left(\underline{b}^T(\theta)\left(X\underline{c}^3+\frac{\underline{v}}{4}\right)-\frac{\theta^5}{20}\right)\right)h^5+O(h^6),$$

where h is the subinterval size. For a fourth order method the coefficients of h, h^2, h^3, and h^4 are zero. We have explained above that we also chose the coefficients of our scheme such that $\left(\underline{b}^T(\theta)-\frac{\theta^5}{5}\right)=0$. This leaves

$$y(t)-u(t)=\phi_4(\theta)K_5h^5+O(h^6), \tag{5.3}$$

where $\phi_4(\theta)\equiv\underline{b}^T(\theta)(X\underline{c}^3+\frac{\underline{v}}{4})-\frac{\theta^5}{20}$ and $K_5=C_{52}$.

Recall that $t=t_i+\theta h$ which implies $\theta=\frac{t-t_i}{h}$. We therefore have that $\frac{d}{dt}=\frac{d}{d\theta}\frac{d\theta}{dt}=\frac{1}{h}\frac{d}{d\theta}$. Taking derivatives in equation (5.3) with respect to t, we get

$$\frac{d}{dt}\left(y(t)-u(t)\right)=y'(t)-u'(t)=\frac{d}{dt}\left(\phi_4(\theta)K_5h^5+O(h^5)\right)=K_5h^5\frac{d}{dt}\phi_4(\theta)+O(h^5),$$

$$=K_5h^5\frac{1}{h}\frac{d}{d\theta}\phi_4(\theta)+O(h^5)=\phi_4'(\theta)K_5h^4+O(h^5),$$

where $'\equiv\frac{d}{d\theta}$ on the right hand side of the above equation.

Thus

$$y'(t)-u'(t)=\phi_4'(\theta)K_5h^4+O(h^5).$$

The defect, $\delta(t)=u'(t)-f(t,u(t))$, can be rewritten as follows (recall that $y'(t)-f(t,y(t))=0$ because $y(t)$ is the true solution):

$$\delta(t)=u'(t)-f(t,u(t))-(y'(t)-f(t,y(t)))=u'(t)-y'(t)-(f(t,u(t))-f(t,y(t))).$$

Assuming f satisfies a Lipschitz condition with constant L, then

$$|f(t,u(t)))-f(t,y(t))|\leq L|u(t)-y(t)|=O(h^5).$$

Thus

$$\delta(t) = u'(t) - y'(t) + O(h^5) = \phi_4'(\theta)K_5h^4 + O(h^5), \tag{5.4}$$

where we recall that $\phi_4(\theta) = \underline{b}^T(\theta)(X\underline{c}^3 + \frac{v}{4}) - \frac{\theta^5}{20}$. From this we have that

$$\phi_4'(\theta) = -\frac{1}{640}\theta(\theta - 1)(100\theta^2 - 124\theta - 3),$$

and the maximum value of the defect, when the stepsize h is sufficiently small, is at the maximum of $\phi_4'(\theta)$ for $\theta = [0, 1]$ which turns out to be about 0.5453.

5.3.3 A New Continuous 6th Order Scheme

In the new version of MIRKDC, the discrete 5-stage, 6th order, MIRK scheme is embedded in an 8-stage, 6th order CMIRK scheme - see Muir[21]. Here we present a 9-stage, 6th order CMIRK scheme, containing the discrete 5-stage, 6th order MIRK scheme, which leads to a defect which is asymptotically correct. The coefficients of this scheme are shown in Table 5.5, where

$x_{31} = \frac{1}{14} + \frac{\sqrt{21}}{98}, x_{32} = \frac{-1}{14} + \frac{\sqrt{21}}{98}, x_{41} = \frac{1}{14} - \frac{\sqrt{21}}{98}, x_{42} = \frac{-1}{14} - \frac{\sqrt{21}}{98},$

$x_{51} = \frac{-5}{128}, x_{52} = \frac{5}{128}, x_{53} = \frac{7\sqrt{21}}{128}, x_{53} = \frac{-7\sqrt{21}}{128},$

$x_{61} = \frac{9\sqrt{7}}{1960} + \frac{3}{112}, x_{62} = \frac{9\sqrt{7}}{1960} - \frac{3}{112},$

$x_{63} = \frac{11\sqrt{7}}{840} + \frac{3\sqrt{21}}{112}, x_{64} = \frac{11\sqrt{7}}{840} - \frac{3\sqrt{21}}{112}, x_{65} = \frac{-26\sqrt{7}}{735},$

$x_{71} = \frac{-9\sqrt{7}}{1960} + \frac{3}{112}, x_{72} = \frac{-9\sqrt{7}}{1960} - \frac{3}{112},$

$x_{73} = \frac{-11\sqrt{7}}{840} + \frac{3\sqrt{21}}{112}, x_{74} = \frac{-11\sqrt{7}}{840} - \frac{3\sqrt{21}}{112}, x_{75} = \frac{26\sqrt{7}}{735},$

$x_{81} = \frac{729\sqrt{21}}{500000} + \frac{764}{15625}, x_{82} = \frac{-28}{15625} + \frac{729\sqrt{21}}{500000},$

$x_{84} = \frac{-5103\sqrt{21}}{250000}, x_{85} = \frac{-834}{15625} - \frac{729\sqrt{21}}{31250},$

0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
$\frac{1}{2}+\frac{\sqrt{21}}{14}$	$\frac{1}{2}-\frac{8\sqrt{21}}{98}$	x_{31}	x_{32}	0	0	0	0	0	0	0
$\frac{1}{2}-\frac{\sqrt{21}}{14}$	$\frac{1}{2}+\frac{8\sqrt{21}}{98}$	x_{41}	x_{42}	0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	x_{51}	x_{52}	x_{53}	x_{54}	0	0	0	0	0
$\frac{1}{2}-\frac{\sqrt{7}}{14}$	$\frac{1}{2}-\frac{\sqrt{7}}{14}$	x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	0	0	0	0
$\frac{1}{2}+\frac{\sqrt{7}}{14}$	$\frac{1}{2}+\frac{\sqrt{7}}{14}$	x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	0	0	0	0
$\frac{1}{10}$	$\frac{1}{10}$	x_{81}	x_{82}	0	x_{84}	x_{85}	x_{86}	x_{87}	0	0
$\frac{9}{10}$	$\frac{9}{10}$	x_{91}	x_{92}	0	0	x_{95}	x_{96}	x_{97}	x_{98}	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$	$b_5(\theta)$	$b_6(\theta)$	$b_7(\theta)$	$b_8(\theta)$	$b_9(\theta)$

Table 5.5: Tableau of 9-stage, 6th order, stage order 3 CMIRK scheme which gives an asymptotically correct defect estimate.

$x_{86} = \frac{49}{15625} - \frac{5481\sqrt{7}}{500000} + \frac{5103\sqrt{21}}{250000}, x_{87} = \frac{49}{15625} + \frac{5481\sqrt{7}}{500000} + \frac{5103\sqrt{21}}{250000},$

$x_{91} = \frac{-8581}{500000}, x_{92} = \frac{-27931}{500000}, x_{95} = \frac{17607}{125000},$

$x_{96} = \frac{-62377}{90625} - \frac{1323\sqrt{7}}{580000}, x_{97} = \frac{-62377}{90625} + \frac{1323\sqrt{7}}{580000}, x_{98} = \frac{81}{1160},$

and where

$b_1(\theta) = -\frac{\theta}{4860}(40635\theta - 133610\theta^2 + 202540\theta^3 - 141948\theta^4 + 35000\theta^5 + 2000\theta^6 - 4860),$

$b_2(\theta) = -\frac{\theta^2}{4860}(-2565 + 27070\theta - 87800\theta^2 + 110052\theta^3 - 49000\theta^4 + 2000\theta^5),$

$b_3(\theta) = -\frac{49}{540}\theta^2(-135 + 1510\theta - 5630\theta^2 + 9250\theta^3 - 7000\theta^4 + 2000\theta^5),$

$b_4(\theta) = -\frac{49}{540}\theta^2(-135 + 1510\theta - 5630\theta^2 + 9250\theta^3 - 7000\theta^4 + 2000\theta^5),$

$b_5(\theta) = -\frac{16}{135}\theta^2(-135 + 1510\theta - 5630\theta^2 + 9250\theta^3 - 7000\theta^4 + 2000\theta^5),$

$b_6(\theta) = \frac{49}{4698}\theta^2(\theta-1)^2(220000\theta^3 - 330000\theta^2 - 600\sqrt{7}\theta^2 + 14132\theta + 600\sqrt{7}\theta - 1566 - 81\sqrt{7}),$

$b_7(\theta) = \frac{49}{4698}\theta^2(\theta-1)^2(22000\theta^3-33000\theta^2+600\sqrt{7}\theta^2+14132\theta-600\sqrt{7}\theta-1566+81\sqrt{7})$,

$b_8(\theta) = \frac{3125}{14094}(\theta-1)^2\theta^2(27+76\theta-396\theta^2+320\theta^3)$,

$b_9(\theta) = \frac{3125}{14094}(\theta-1)^2\theta^2(-27+244\theta-564\theta^2+320\theta^3)$.

We next consider the derivation of this scheme. We use a Hermite-Birkhoff interpolation approach. The general form for the continuous solution approximation on the ith subinterval is

$$u(t) = u(t_i+\theta h_i) = d_0(\theta)y_i + d_1(\theta)y_{i+1}$$

$$+h_i\left(b_1(\theta)K_1 + b_2(\theta)K_2 + b_6(\theta)K_6 + b_7(\theta)K_7 + b_8(\theta)K_8 + b_9(\theta)K_9\right), \qquad (5.5)$$

where $K_j = f(t_i + c_j h, \hat{y}_j), j = 1,2,6,7,8,9$, with $c_1 = 0, \hat{y}_1 = y_i, c_2 = 1, \hat{y}_2 = y_{i+1}$; the other c_j and $\hat{y}_j$ values are to be determined.

We consider the local error associated with each of the terms in (5.5). Since we are looking at local error, y_i and $K_1 = f(t_i, y_i)$ are considered to be exact. The right end point value, y_{i+1}, for a 6th order method has local error $O(h_i^7)$ as does $K_2 = f(t_{i+1}, y_{i+1})$ (assuming a Lipschitz condition on f). We note that the factor of h_i multiplying the sum of K_j terms implies that the contribution to the error in $u(t)$ in (5.5) from K_2 will then be $O(h_i^8)$. With a similar argument we see that we want the remaining K_j terms, $K_6, \cdots, K_9$, to also have a local error that is $O(h_i^7)$ so that they will also contribute an $O(h_i^8)$ error to $u(t)$. This will leave only the y_{i+1} term contributing an $O(h_i^7)$ term to the error.

The main task is to construct four new stages, K_6, K_7, K_8, K_9, each of which is based on a corresponding argument $\hat{y}_6, \hat{y}_7, \hat{y}_8, \hat{y}_9$. Those arguments have to have a local error that is $O(h_i^7)$. This is done by requiring $\hat{y}_6, \hat{y}_7, \hat{y}_8, \hat{y}_9$, to satisfy the stage

order 6 conditions. Then on subinterval $[t_i, t_{i+1}]$, $u(t)$ will be based on 7 pieces of $O(h_i^8)$ data, and one piece of $O(h_i^7)$ data, namely y_{i+1}. Since the interpolation error will be $O(h_i^8)$, the only contribution to the leading term in the defect will come from the term $d_1(\theta)y_{i+1}$. The value $\hat{y}_6$ has the general form

$$\hat{y}_6 \equiv (1 - v_6)y_i + v_6 y_{i+1} + h_i \sum_{j=1}^{5} x_{6j} K_j,$$

and $\hat{y}_7, \hat{y}_8$, and $\hat{y}_9$ are given by similar expressions.

For the new four stages, we must apply the stage order 6 conditions,

$$X\underline{c}^{l-1} + \frac{\underline{v}}{l} = \frac{\underline{c}^l}{l}, \quad l = 1, \cdots, 6.$$

We can apply these 6 equations to $\hat{y}_6$ to solve for $c_6, x_{61}, x_{62}, x_{63}, x_{64}, x_{65}$ in terms of v_6. Similarly, we can apply the stage order 6 conditions to $\hat{y}_7$ to solve $x_{7j} (j = 1, \cdots, 6)$ in terms of c_7, v_7; to $\hat{y}_8$ to solve x_{8j} $(j = 1, \cdots, 7, j \neq 3)$ in terms of c_8, v_8, x_{83}; and to $\hat{y}_9$ to solve $x_{9j} (j = 1, \cdots, 8, j \neq 3$ and $j \neq 4)$ in terms of c_9, v_9, x_{93}, x_{94}. Finally we make some arbitrary assignments: $v_6 = \frac{1}{2} - \frac{\sqrt{7}}{14}$, $c_7 = v_7 = \frac{1}{2} + \frac{\sqrt{7}}{14}, c_8 = v_8 = \frac{1}{10}, c_9 = v_9 = \frac{9}{10}, x_{83} = x_{93} = x_{94} = 0$. (A different choice of the values could lead to a method with a smaller truncation error but this is left for future work.)

Having constructed the four extra stages, we now consider the weight polynomials, $d_0(\theta), d_1(\theta), b_1(\theta), b_2(\theta), b_6(\theta), b_7(\theta), b_8(\theta), b_9(\theta)$, which are degree 7 polynomials. The Hermit-Birkhoff interpolant (5.5) interpolates the y_i value at 0 and the y_{i+1} value at 1, and its derivative will interpolate K_1 at $c_1 = 0$, K_2 at $c_2 = 1$, K_6 at c_6, K_7 at c_7, K_8 at c_8, and K_9 at c_9.

Each of these interpolation conditions leads to conditions on the weight poly-

nomials $d_0(\theta)$, $d_1(\theta)$, $b_1(\theta)$, $b_2(\theta)$, $b_6(\theta)$, $b_7(\theta)$, $b_8(\theta)$, $b_9(\theta)$. For example the first interpolation condition,

$$u(t_i) = u(t_i + 0h_i) = d_0(0)y_i + d_1(0)y_{i+1}$$

$$+h_i\left(b_1(0)K_1 + b_2(0)K_2 + b_6(0)K_6 + b_7(0)K_7 + b_8(0)K_8 + b_9(0)K_9\right) = y_i,$$

implies $d_0(0) = 1, d_1(0) = 0, b_1(0) = b_2(0) = b_6(0) = b_7(0) = b_8(0) = b_9(0) = 0$. Applying all eight interpolation conditions similarly leads to a total of eight conditions on each of the weight polynomials. These conditions turn out to be sufficient to uniquely specify each of the degree 7 polynomials. By comparing powers of θ and the corresponding coefficients of the weight polynomials, the interpolation conditions reduce to a matrix system of the form

$$MC = I,$$

where the columns of C give the coefficients of the weight polynomials, and where

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2c_6 & 3c_6^2 & 4c_6^3 & 5c_6^4 & 6c_6^5 & 7c_6^6 \\ 0 & 1 & 2c_7 & 3c_7^2 & 4c_7^3 & 5c_7^4 & 6c_7^5 & 7c_7^6 \\ 0 & 1 & 2c_8 & 3c_8^2 & 4c_8^3 & 5c_8^4 & 6c_8^5 & 7c_8^6 \\ 0 & 1 & 2c_9 & 3c_9^2 & 4c_9^3 & 5c_9^4 & 6c_9^5 & 7c_9^6 \end{pmatrix},$$

It then follows that the columns of C are the columns of M^{-1}; we can compute these explicitly using Maple [30], and from these we get the coefficients which define $d_0(\theta), d_1(\theta), b_1(\theta), b_2(\theta), b_6(\theta), b_7(\theta), b_8(\theta), b_9(\theta)$. The final step is to substitute in (5.5) for y_{i+1} using the discrete 6th order MIRK scheme to convert (5.5) to the standard CMIRK form similar to that given in (5.2) but with 9 stages.

5.4 Numerical Experiments and Results

In order to obtain a good estimate of the actual location of the maximum defect on each subinterval, we added some (temporary) code to the new version of MIRKDC to perform 100-point sampling of the defect on each subinterval.

5.4.1 Experimental Location of Maximum Defect

2nd order method

The 2nd order schemes which are employed in the new version of MIRKDC are the same as in the original MIRKDC. For order 2, we chose the number of subintervals, Nsub = 10, 50, 100 and the tolerance to be 10^{-9}, for all six test problems. The results are shown in Figures 5.1 - 5.3. From these results, we see that when the step size h is sufficiently small, the location of maximum defect will be about 0.5. The results further show that the 2nd order continuous method in the original version of MIRKDC gives an asymptotically correct defect. This is supported by the theoretical analysis of this scheme, given earlier in this chapter.

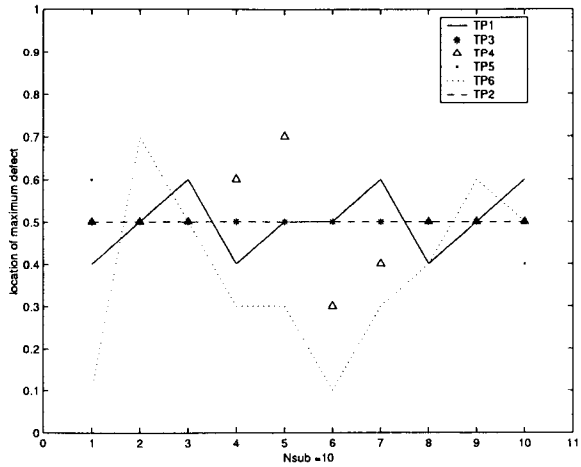

Figure 5.1: Location of maximum defect on each subinterval for six test problems with Nsub = 10, method = 2, tol = 10^{-9}.

We note that for the sixth test problem (TP6), the true solution near the left boundary exhibits a sharp boundary layer; that is, the solution derivatives in this region are very large. In this case the $O(h^{p+1})$ term in (5.1) is not dominated by the $O(h^p)$ term and we can see from Figures 5.2 - 5.3 that the location of maximum defect is not at 0.5 for the first few subintervals, at the left end of the domain.

During a standard computation, as MIRKDC adapts its mesh, it would place more subintervals in this layer using smaller subintervals where appropriate leading to smaller h values, which would restore the dominance of the $O(h^p)$ term in the defect, and give the maximum defect at 0.5.

4th order method

First, we tested the 4th order continuous scheme which we implemented as described in section 4.4 (which we will call the old 4th order scheme) with all six test problems

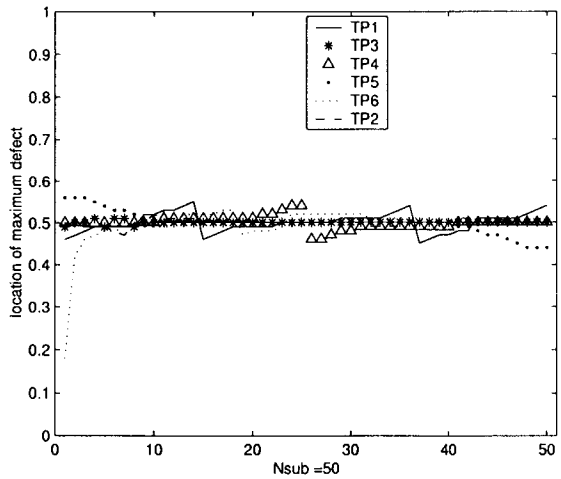

Figure 5.2: Location of maximum defect on each subinterval for six test problems with Nsub = 50, method = 2, tol = 10^{-9}.

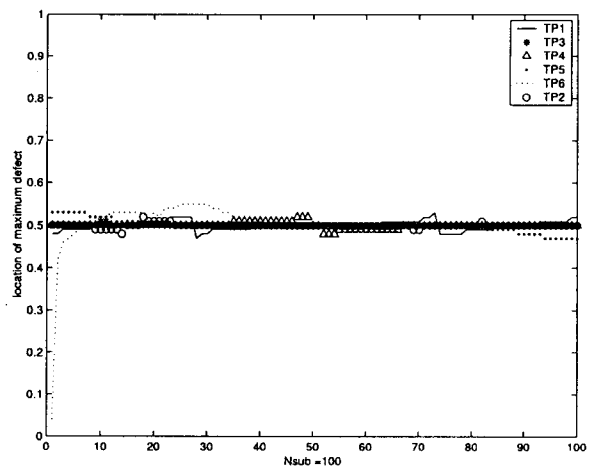

Figure 5.3: Location of maximum defect on each subinterval for six test problems with Nsub = 100, method = 2, tol = 10^{-9}.

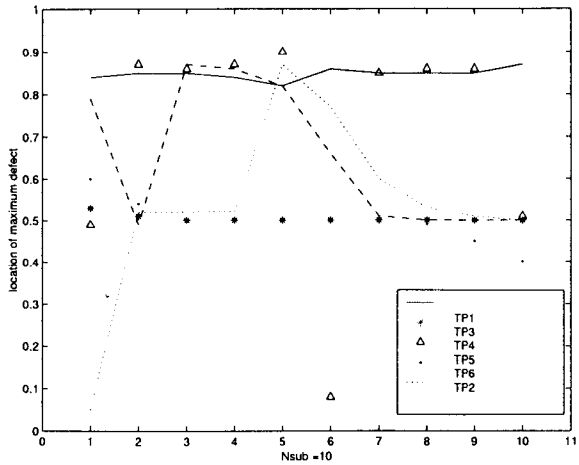

Figure 5.4: Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the old 4th order method, tol = 10^{-9}.

for Nsub = 10, 100, 300, and tol = 10^{-9}. The results are shown in Figures 5.4 - 5.6. Secondly, we tested the new 4th order continuous scheme described in section 5.3.1 with all six test problems for Nsub = 10, 50, 100, and tol = 10^{-9}. Figures 5.7 - 5.9 show the results.

From Figures 5.4 - 5.6, we see that old 4th order scheme does not lead to an asymptotically correct estimate; even with h fairly small, the position of the maximum defect is not consistently in the same location over all problems and subintervals. From Figures 5.7 - 5.9, we see that new 4th order scheme does give an asymptotically correct estimate and that the location of maximum defect is about $0.54 - 0.55$. This is consistent with the theory shown earlier in this chapter which predicts that the location of the maximum defect will be at about 0.5453.

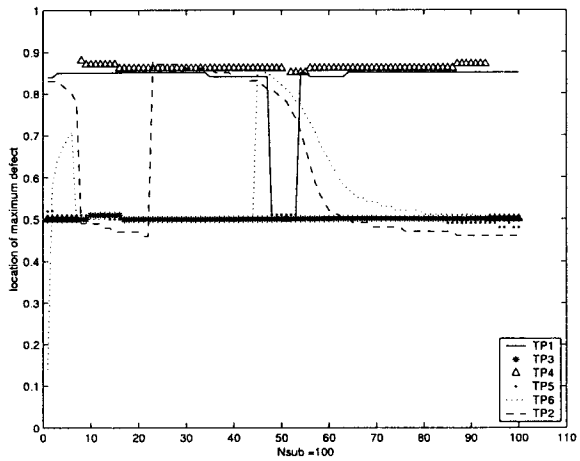

Figure 5.5: Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the old 4th order method, tol = 10^{-9}.

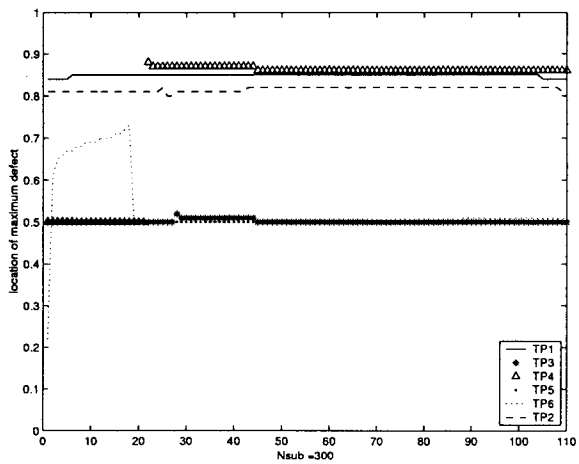

Figure 5.6: Location of maximum defect on each subinterval for six test problems with Nsub = 300 for the old 4th order method, tol = 10^{-9}.

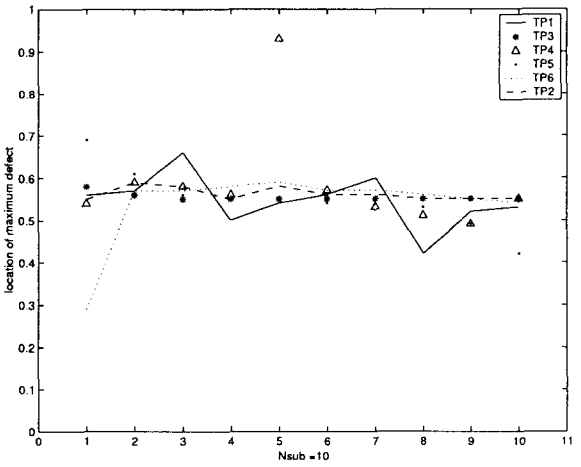

Figure 5.7: Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the new 4th order method, tol = 10^{-9}.

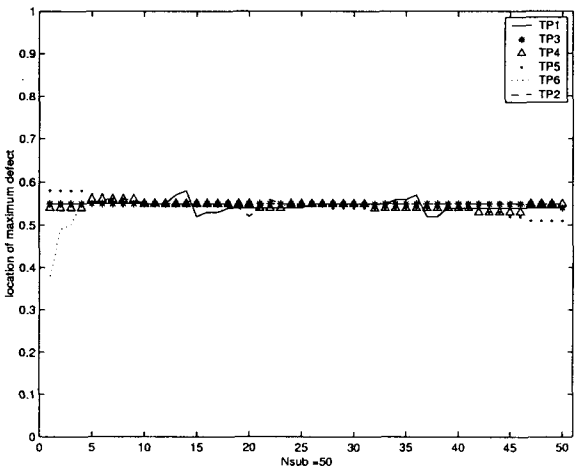

Figure 5.8: Location of maximum defect on each subinterval for six test problems with Nsub = 50 for the new 4th order method, tol = 10^{-9}.

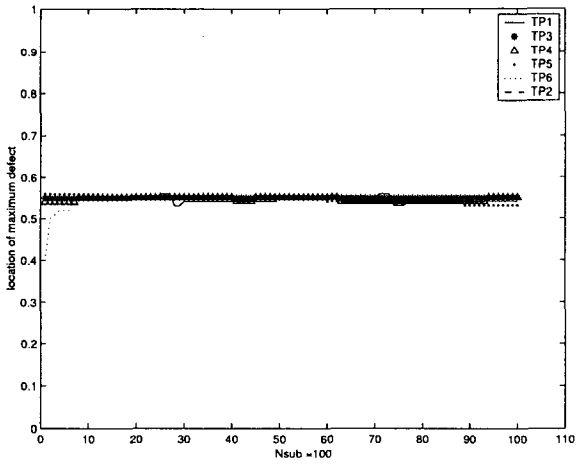

Figure 5.9: Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the new 4th order method, tol = 10^{-9}.

6th order method

First, we tested the 6th order continuous scheme, described in section 4.4, (which we will call the old 6th order scheme) with all six test problems for Nsub = 10, 100, and tol = 10^{-9}. Secondly, we tested the new 6th order scheme, described in section 5.3.2, with all six test problems for Nsub = 10, 100, 300, and tol = 10^{-9}. Figures 5.10 - 5.11 show the results for the old 6th order method; Figures 5.12 - 5.14 show the results for the new 6th order method.

From Figures 5.10 - 5.11, we see that old 6th order scheme does not give an asymptotically correct estimate. From Figures 5.12 - 5.13, we see that new 6th order scheme is giving a defect for which the maximum is either at 0.5 or at about 0.8. This tells us that for this value of $h = O(\frac{1}{Nsub})$, there are two terms dominating the error. The location of a maximum defect at 0.50 is consistent with the theoretical analysis, shown earlier in this chapter. In Figure 5.14, we consider a smaller value of

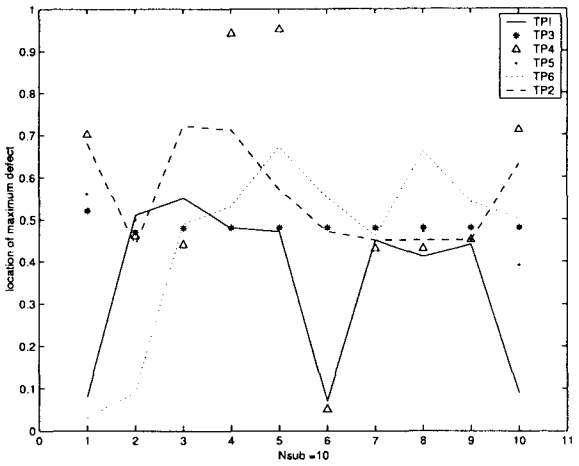

Figure 5.10: Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the old 6th order method, tol = 10^{-9}.

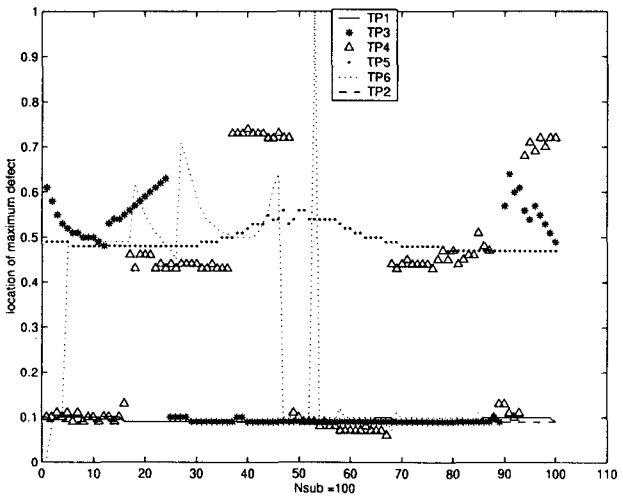

Figure 5.11: Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the old 6th order method, tol = 10^{-9}.

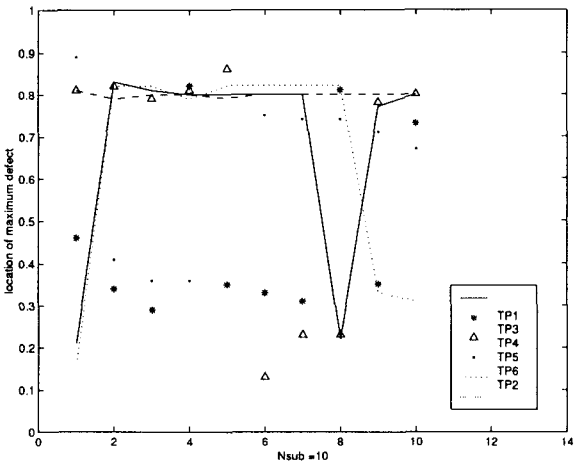

Figure 5.12: Location of maximum defect on each subinterval for six test problems with Nsub = 10 for the new 6th order method, tol = 10^{-9}.

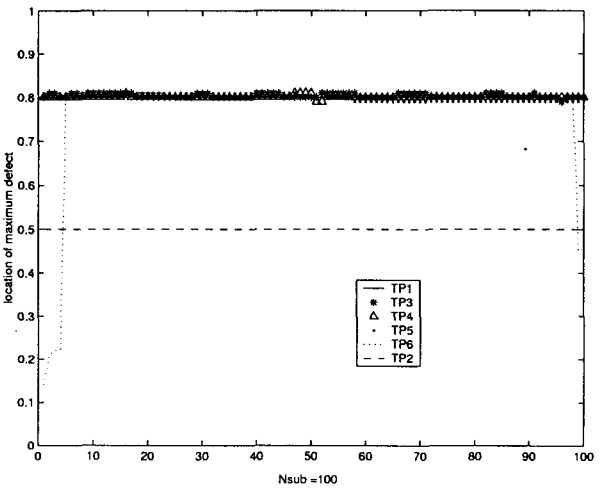

Figure 5.13: Location of maximum defect on each subinterval for six test problems with Nsub = 100 for the new 6th order method, tol = 10^{-9}.

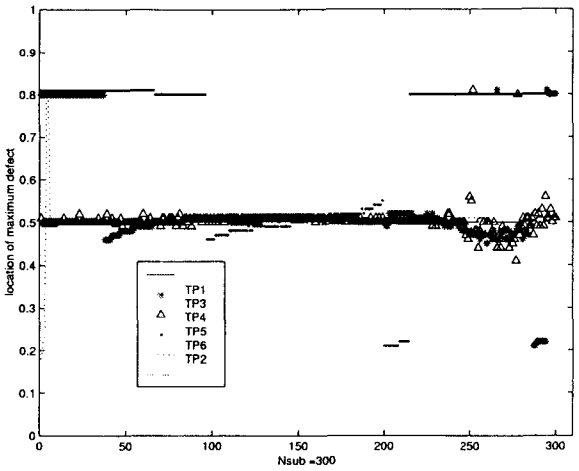

Figure 5.14: Location of maximum defect on each subinterval for six test problems with Nsub = 300 for the new 6th order method, tol = 10^{-9}.

h and can see that the second term, whose maximum defect was at 0.8, is no longer as significant.

These experiments raise an important point. Even when h is not small enough for one term to dominate, only a small number of terms contribute significantly to the error, each contributing its own defect maximum location. Thus a more robust defect estimation strategy is to conduct a small number of additional defect samples at the points corresponding to the locations of maximum defects of the other contributing terms. This is the basis for what we call "safe-guarded strict defect control".

In fact for this case we would be happier with an interpolant for which a single term dominates the error for h values that do not need to be as small as the ones we see in the experiments reported here. A strategy which is likely to produce a more satisfactory interpolant would base the interpolant on a CMIRK scheme which could have all but one of the order conditions for 7th order satisfied. We employed

an equivalent strategy for the 4th order case. This is left for future work.

Asymptotically Correct Defects on Non-uniform Meshes

In the previous experiments in this section, we considered only uniform meshes. In this experiment we tested the new 4th order continous method in a standard computation in which MIRKDC employs a sequence of non-uniform meshes to obtain a numerical solution to within the user tolerance. We solved TP1 with the new 4th order method, with a tolerance of 10^{-9}, and the initial number of subintervals equal to 10. We are interested in the "solution profile", which is given by a sequence of ordered pairs (Nsub, NI), where Nsub is the number of subintervals and NI is the number of Newton iterations MIRKDC uses to find the solution to the discretized boundary value ODE system. For this problem, the solution profile is (10,6), (40, 2), (156,1), (205,1). Figures 5.15 - 5.18 show the relative locations of the maximum defects within each subinterval. These results show that the location of maximum defect is at approximately 0.54, even when the total number of subintervals is only 40.

5.4.2 Comparison of Relaxed Defect Control, Strict Defect Control and Safe-Guarded Strict Defect Control

As mentioned earlier, we added a new option called *defect_control* to the MIRKDC parameter list. If *defect_control* = 0, MIRKDC will choose 'relaxed defect control' (standard two-point sampling). If *defect_control* = 1, MIRKDC will choose 'safe-

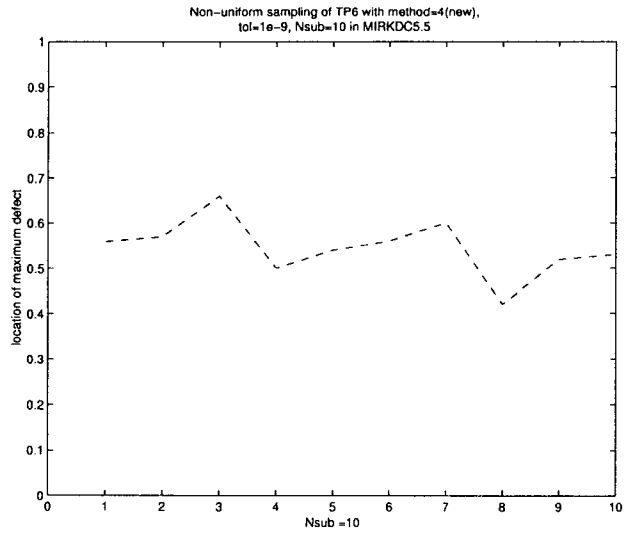

Figure 5.15: Locations of maximum defects for TP6, uniform mesh, Nsub = 10, method order 4.

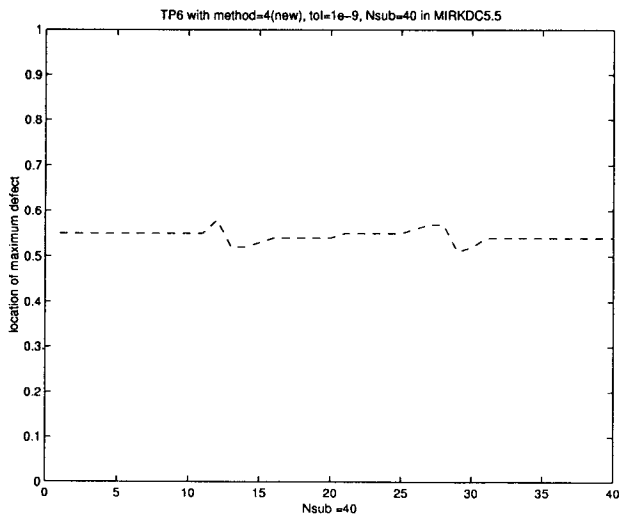

Figure 5.16: Locations of maximum defects for TP6, nonuniform mesh, Nsub = 40, method order 4.

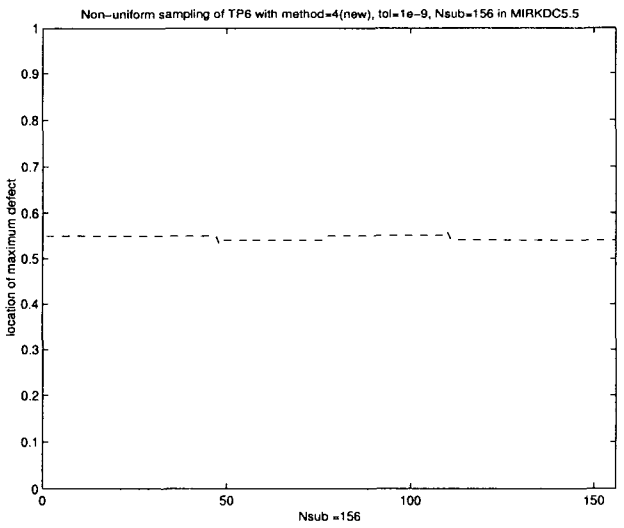

Figure 5.17: Locations of maximum defects for TP6, nonuniform mesh, Nsub = 156, method order 4.

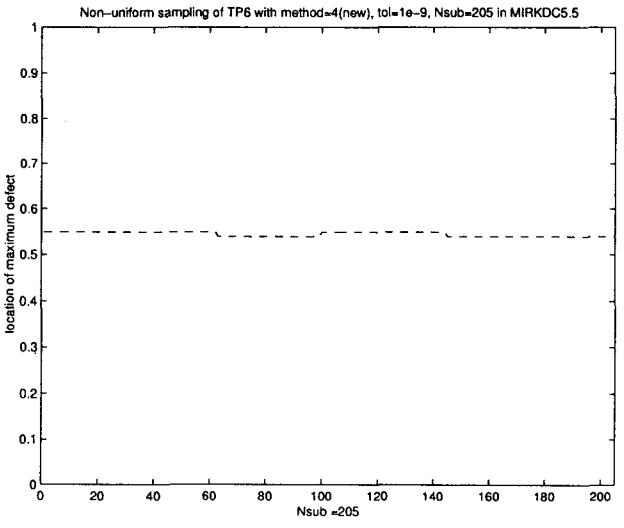

Figure 5.18: Locations of maximum defects for TP6, nonuniform mesh, Nsub = 205, method order 4.

guarded strict defect control'.

Suppose that $\delta_{old}(t)$ is the defect estimate computed by using two-point sampling and the continuous solution $u_{old}(t)$ based on the CMIRK scheme described in section 4.4. It is generally the case that $\delta_{old}(t)$ will underestimate the true maximum defect. Suppose that $\delta_{new}(t)$ is the defect estimate computed by using one-point sampling and the continuous solution $u_{new}(t)$ based on the CMIRK scheme described in section 5.3. We will usually get a correct estimate of the maximum defect, when h is sufficiently small. If $u_{new}(t)$ is significantly more accurate than $u_{old}(t)$, then we have

$$|y(t) - u_{old}(t)| = \kappa_p^{old}(\theta)h^{p+1} + O(h^{p+2}), \quad \text{and}$$

$$|y(t) - u_{new}(t)| = \psi_p(\theta)\kappa_p^{new}h^{p+1} + O(h^{p+2}),$$

with

$$|\psi_p(\theta)\kappa_p^{new}| \ll |\kappa_p^{old}(\theta)|.$$

In the above, $t = t_i + \theta h$ which implies $\theta = \frac{t-t_i}{h}$. Then,

$$\delta_{old}(t) = (\kappa_p^{old}(\theta))'h^p + O(h^{p+1}), \quad \text{and as in (5.4):}$$

$$\delta_{new}(t) = \psi_p'(\theta)\kappa_p^{new}h^p + O(h^{p+1}),$$

and we have

$$|\psi_p'(\theta)\kappa_p^{new}| \ll |(\kappa_p^{old}(\theta))'|,$$

which implies $|\delta_{new}| < |\delta_{old}|$.

On the other hand, if $u_{new}(t)$ is not significantly more accurate than $u_{old}(t)$ (which is the usual case), then we will normally have $|\delta_{new}(t)| > |\delta_{old}(t)|$ since $\delta_{new}(t)$ is a

better estimate of the maximum defect than $\delta_{old}(t)$. Since $|\delta_{new}(t)| > |\delta_{old}(t)|$, it will usually take more subintervals and a finer mesh to satisfy the defect tolerance.

That is, one gets a better solution in the sense that the corresponding defect is more likely to satisfy the user tolerance, but the cost of obtaining this solution is greater, (a) because the new method will require more subintervals, and (b) because the new method will employ more stages per step. In this section we consider a test which examines the accuracy of the defect reported by MIRKDC using (i) the original two-point sampling and (ii) the new one-point sampling. The results for (i) and (ii) are given in Table 5.6. We employ an initial uniform mesh of 100 subintervals.

	Order 4			Order 6		
	Nsub	NI	Def. Est.	Nsub	NI	Def. Est.
original	100	1	$8.5 * 10^{-8}$	100	1	$2.6 * 10^{-12}$
	237	1	$9.3 * 10^{-10}$	125	1	$2.4 * 10^{-13}$
new	100	1	$3.0 * 10^{-8}$	100	1	$6.0 * 10^{-11}$
	202	1	$7.0 * 10^{-10}$	196	1	$9.4 * 10^{-13}$

Table 5.6: Comparison of MIRKDC execution sequences for TP1, ϵ= 0.04; tol = 10^{-9}, method = 4; tol = 10^{-12}, method = 6; NI: the number of full Newton Iterations.

From Table 5.6, we see that, for the methods of order 4, the new scheme with one-point sampling leads to smaller defect estimates, using fewer mesh points. This is an unusual result. The original method underestimates the maximum defect. The new method gives the correct maximum defect. However the estimate from the new

method is smaller than the estimate given by the original method. This happens because, as mentioned above, the new method employs a more accurate interpolant leading to a defect which has a true maximum that is in fact smaller than the true maximum defect for the original method. On the other hand, for the 6th order methods, the original scheme with two-point sampling appears to do better. However, it is underestimating the true maximum defect and is thus returning a solution which may have a larger defect than that of the new method.

We also conducted some CPU time testing for the three different defect control strategies, 'relaxed defect control', 'strict defect control', and 'safe-guarded strict defect control'. For 'relaxed defect control' we use two sample points per subinterval. For 'strict defect control' we use 1-point sampling per subinterval. For 'safe-guarded strict defect control', we first check the subinterval size h; if it is less than the threshold value we use 1-point sampling, otherwise we sample at three points. The CPU time each case for methods of order 4 and 6, respectively, is shown in Tables 5.7 and 5.8. From Table 5.7, we can observe that 'strict defect control' and 'safe-guarded strict defect control' both work faster than 'relaxed defect control' when the methods are of order 4. Table 5.8 shows us that for the methods of order 6, 'relaxed defect control' works faster when h is bigger, and that 'strict defect control' and 'safe-guarded strict defect control' are faster when h is smaller.

Initial Nsub	10	50	100	150
Relaxed defect control	0.33	0.42	0.29	0.39
Strict defect control	0.33	0.38	0.26	0.33
Safe-guarded strict defect control	0.33	0.36	0.26	0.33

Table 5.7: CPU time (seconds) for defect control strategies for various initial Nsub values; method = 4, tol = 10^{-9}, for TP1, $\epsilon = 0.04$.

Initial Nsub	10	50	100	300	500	1000
Relaxed defect control	0.06	0.07	0.18	0.52	0.85	1.73
Strict defect control	0.16	0.17	0.19	0.49	0.85	1.69
Safe-guarded strict defect control	0.17	0.18	0.19	0.52	0.88	1.65

Table 5.8: CPU time (seconds) for defect control strategies for various initial Nsub values; method = 6, tol = 10^{-9}, for TP1, $\epsilon = 0.04$.

Comparison of Solution Profiles for Defect Control Strategies

In this subsection, we study the solution profiles for a standard computation with MIRKDC with the 'defect_control' strategies described earlier. We will experimentally determine the true maximum defect value on each subinterval. The initial number of subintervals is 10. The methods are of orders 4 and 6. The tolerance is 10^{-9} and the test problem is TP1 with $\epsilon = 0.04$. In the first experiment (EX1), we use the original continuous scheme from section 4.4 and then find the true maximum defect on each subinterval, which MIRKDC will then employ instead of the estimate normally obtained from two-point sampling. The idea is to see how much of a dif-

ference it makes to provide MIRKDC with the true maximum defect rather than the underestimated maximum defect provided by two-point sampling.

The second experiment (EX2) uses the original relaxed defect control strategy. The third experiment (EX3) uses safe-guarded strict defect control. All results are given in Table 5.9.

From the results of EX1 and EX2 on method order 4, we see that there is not much difference between the computation using the true maximum defects and that using relaxed defect control. On the other hand, we see from EX3 that the more accurate interpolant associated with the safe-guarded strict defect control gives smaller defects and pays for itself, since it uses fewer subintervals to obtain the final solution.

For the 6th order case, there is again not much difference between the use of the original method with the two-point sampling estimate of the maximum defect and with the correct maximum defect. The use of the asymptotically correct defect requires more subintervals and a larger computation because the underlying interpolant is not substantially more accurate than the original and the corresponding improved estimate of the maximum defect is therefore larger, implying that a finer mesh having more subintervals is required.

We also compared the solution profiles of the method of order 2 with relaxed defect control and safe-guarded strict defect control. The interpolant of method order 2 is the same for both defect control strategies. The test problem is TP1; tolerance = 10^{-4}, the initial Nsub is 10, and $\epsilon = 0.04$. The results are shown in Table 5.10. Two observations can be made. One is that when both defect control strategies use

the same mesh points, the defect obtained from safe-guarded strict defect control is bigger than the one from relaxed defect control. This shows that safe-guarded strict defect control does a better job of estimating the maximum defect than does relaxed defect control. Another observation is that safe-guarded strict defect control forces MIRKDC to use more mesh points. Because safe-guarded strict defect control returns a bigger defect, more mesh points are needed to provide a finer partition of the problem interval in order to get a solution whose maximum defect is less than the user tolerance.

5.5 Conclusions

In this chapter we see that while relaxed defect control can be faster, it does not control the defect as well as the other strategies. On the other hand, safe-guarded strict defect control costs more per subinterval but can give us an asymptotically correct defect and the cost over the whole computation is about the same as for relaxed defect control.

	Order 4		Order 6	
	(Nsub,NI)	Def. Est.	(Nsub,NI)	Def. Est.
EX1	$(10, 6)$	$6.5 * 10^{-4}$	$(10, 6)$	$2.2 * 10^{-6}$
	$(40, 2)$	$1.8 * 10^{-6}$	$(35, 1)$	$9.1 * 10^{-10}$
	$(160, 1)$	$5.3 * 10^{-9}$		
	$(269, 1)$	$5.1 * 10^{-10}$		
EX2	$(10, 6)$	$5.0 * 10^{-4}$	$(10, 6)$	$1.4 * 10^{-6}$
	$(40, 2)$	$1.6 * 10^{-6}$	$(34, 1)$	$9.7 * 10^{-10}$
	$(160, 1)$	$4.5 * 10^{-9}$		
	$(261, 1)$	$4.9 * 10^{-10}$		
EX3	$(10, 6)$	$2.4 * 10^{-4}$	$(10, 6)$	$1.9 * 10^{-5}$
	$(40, 2)$	$5.4 * 10^{-7}$	$(40, 1)$	$1.8 * 10^{-8}$
	$(160, 1)$	$1.8 * 10^{-9}$	$(72, 1)$	$6.5 * 10^{-10}$
	$(222, 1)$	$4.1 * 10^{-10}$		

Table 5.9: Comparison of defect control strategies; TP1 with $\epsilon = 0.04$, tol $= 10^{-9}$; EX1: exact maximum defect - original interpolant; EX2: relaxed defect control - original interpolant; EX3: safe-guarded strict defect control - new interpolant.

	Order 2	
	Nsub	Def. Est.
relaxed defect control	10	$1.3 * 10^{-1}$
	20	$4.7 * 10^{-2}$
	80	$1.8 * 10^{-3}$
	211	$1.6 * 10^{-4}$
	290	$6.9 * 10^{-5}$
safe-guarded strict defect control	10	$1.5 * 10^{-1}$
	20	$5.4 * 10^{-2}$
	80	$2.4 * 10^{-3}$
	230	$1.9 * 10^{-4}$
	327	$7.3 * 10^{-5}$

Table 5.10: Comparison of relaxed defect control and safe-guarded strict defect control for TP1, $\epsilon = 0.04$, tol $= 10^{-4}$, method $= 2$.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have discussed six new features which have been added to the new version of MIRKDC. Based on a computational derivative approximation, we have provided an option in which MIRKDC can compute approximate Jacobian matrices. This can be very convenient when the BVODE system is complicated. We have added the capability for analytic derivative assessment, which can allow MIRKDC to make sure that the Jacobian matrix subroutines supplied by the user are correct.

We have added an option for problem sensitivity (conditioning) assessment, which allows MIRKDC to provide an estimate for the conditioning constant. If the estimated conditioning constant is large, then this is an indication that the problem is ill-conditioned. Thus, when MIRKDC returns a solution and a larger conditioning constant estimate, the user should be wary of the accuracy of the solution.

We introduced a new CMIRK formula for order 4, and demonstrated that it leads to significant improvements in the performance of MIRKDC. We also designed and analyzed new defect control strategies. Our analysis shows that relaxed defect control sometimes works faster, but it cannot control the defect as well. On the other hand, safe-guarded strict defect control gives a better control of the defect but costs more on each subinterval, although not more overall.

We also performed some preliminary investigation for the computation of a global error estimate. Our approach yielded a good estimate but the costs are too high. However, the results do provide a good baseline for future research.

6.2 Future Work

Some possible further work following from this thesis includes:

- Reducing the cost of computing the estimate of κ. In the new version of MIRKDC, the computation of the estimate of κ is done for each new matrix that is constructed during the computation of the numerical solution. We might only compute the estimate of κ after an acceptable solution has been obtained.

- Adding an improved interpolant of order 6 which gives an asymptotically correct defect, and is also more accurate than the standard 6th order interpolant.

- Further investigation of low cost global error estimation strategies.

Bibliography

* access date here means the date when we searched
  and got the information from the website.

[1] U. M. Ascher, R. M. M. Mattheij and R. D. Russell, Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, SIAM, Philadelphia, USA, 1995.

[2] U. M. Ascher, J. Christiansen and R. D. Russell, Collocation software for boundary-value ODEs, ACM Trans. Math. Softw., 7, 209-222, 1981.

[3] U. M. Ascher and L. R., Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, Philadelphia, USA, 1998.

[4] U. M. Ascher and R. D. Russell, Evaluation of B-splines for solving systems of boundary value problems, Tech. Rep., 77-14, Dept. of Comp. Sci., University of British Columbia, 1977.

[5] G. Bader and U. M. Ascher, A new basis implementation for a mixed order boundary value ODE solver, SIAM J. Sci. Stat. Comp., 8, 483-500, 1987.

[6] C. De. Boor, A practical guide to B-splines, Springer-Verlag, New York, 1978.

[7] J. C. Butcher, The Numerical Analysis of Ordinary Differential Equations, Wiley, Chichester, 1987.

[8] M. Calvo, D. J. Higham, J. I. Montijano and L. R'andez, Global error estimation with adaptive explicit Runge-Kutta methods, IMA J. Numer. Anal., 16, 47-63, 1996.

[9] J. R. Cash, D. R. Moore, N. Sumarti and M. V. Daele, A highly stable deferred correction scheme with interpolant for systems of nonlinear two-point boundary value problems, J. Comput. Appl. Math. 155, 339-358, 2003.

[10] J. R. Cash, Runge-Kutta methods for the solution of stiff two-point boundary value problems. Appl. Numer. Math., 22, 165-177, 1996.

[11] J. R. Cash and A. Snghal, Mono-implicit Runge-Kutta formulae for the numerical integration of stiff systems. IMA, J. Numer. Anal., 2, 211-217, 1982.

[12] 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. Statist. Comput., 12, 971-989, 1991.

[13] J. Christiansen and R. D. Russell, Error analysis for spline collocation methods with application to knot selection, Math. Comput., 32, 415-419, 1978.

[14] J. C. Diaz, G. Fairweather, and P. Keast, Fortran packages for solving certain almost block diagonal linear system by modified alternate row and column elimination, ACM Trans. Math. Softw., 9, 358-375, 1983.

[15] W. H. Enright, Continuous numerical methods for ODEs with defect control, J. Appl. Comput. Math., 125, 159-170, 2001.

[16] W. H. Enright, The design and implementation of usable ODE software, Numer. Alg., 31, 125-137, 2002.

[17] W. H. Enright and P. H. Muir, Efficient classes of Runge-Kutta methods for two-point boundary value problems, Computing, 37, 315-334, 1986.

[18] W. H. Enright and P. H. Muir, Runge-Kutta software with defect control for boundary value ODEs, SIAM J. Sci. Comput., 17, 479-497, 1996.

[19] P. Keast, COLROW, http://www.mscs.dal.ca/~ keast/research/leq/, 1992.

[20] P. Keast and R. Affleck, BSPCND, software for estimation of the 1-norm condition number of an almost block diagonal matrix, http://www.mscs.dal.ca/~ keast/, 1997.

[21] P. H. Muir, Optimal discrete and continuous mono-implicit Runge-Kutta schemes for BVODEs, Adv. Comput. Math., 10, 135-167, 1999.

[22] J. Patterson, P. H. Muir and P. Keast, BSPCNDMAX, software for estimation of the max norm condition number of an almost block diagonal matrix, http://www.mscs.dal.ca/~ keast/, 2001.

[23] P. J. Peterson, Global error estimation using defect correction techniques for explicit Runge-Kutta methods, Tech. Rep. No. 192, Dept. of Comput. Sci., University of Toronto, 1986.

[24] R. D. Russell, A comparison of collocation and finite differences for two-point boundary value problems. SIAM, J. Numer. Anal., 14, 19-39, 1977.

[25] L. F. Shampine and and P. H. Muir, Estimating conditioning of BVPs for ODEs, to appear in Comput. Math. Appl., Special Issue on the Numerical Analysis of Ordinary Differential Equations, 2004.

[26] L. F. Shampine and S. Thompson. A Friendly Fortran DDE Solver, A preprint of a paper for Volterra 2004, The Third International Conference on the Numerical Solution of Volterra and Delay Equations, 2004.

[27] R. D. Skeel, Thirteen ways to estimate global error, Numer. Math., 48, 1-20, 1986.

[28] http://www.llnl.gov/CASC/nsde/pubs/toms_cvodes_with_covers.pdf, access date: 2004-06-17.

[29] http://www.llnl.gov/CASC/sundials, access date: 2004-06-21.

[30] http://www.maplesoft.com/, access date: 2004-06-03.

[31] http://www.ma.ic.ac.uk/~jcash/BVP_software/readme.php, access date: 2004-06-25.

[32] http://www.mathworks.com/products/matlab, access date: 2004-06-02.

[33] http://www.nag.co.uk/numeric/numerical_libraries.asp, access date: 2004-06-02.

[34] http://www.netlib.org/, access date: 2004-06-02.

[35] http://www.netlib.org/ode/index.html, access date: 2004-06-25.

[36] http://www.netlib.org/ode/rksuite, access date: 2004-06-17.

[37] http://www.netlib.org, access date: 2004-06-25.

[38] http://www.netlib.org/ode/index.html, access date: 2004-06-25.

[39] http://www.vni.com/products/imsl/index.html, access date: 2004-06-03.