Parallel diagonally implicit multistage integration methods for stiff ordinary differential equations

Anjana Devi Singh, BSc, GCEd, MSc (USP,Fiji).

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY

Department of Mathematics
The University of Auckland

1999

Abstract

From this large family of general linear methods, we look at a special class of methods known as the diagonally implicit multi-stage integration methods (DIMSIMs). One of the four matrices characterizing a DIMSIM, the matrix A, corresponds to the coefficient matrix in a Runge-Kutta method. Hence, the structure of A plays a central role in the implementation costs of the method. We will refer to various choices for the form of A as "types". In particular, for type 4 methods, $A = \lambda I$ and these are suitable for the parallel solution of stiff problems. A-stable type 4 methods in which the order of the method equals the number of stages, has been derived by J. C. Butcher (1996). In this thesis we consider a variable stepsize, variable order implementation of these methods as parallel solvers for stiff initial value problems. Although these methods have shown potential for the solution of stiff initial value problems they are not very efficient when compared to existing sequential solvers RADAU5 and VODE. This is probably because of the large error constants of these methods.

In order to have an efficient set of parallel methods, we derive a new set of A-stable type 4 methods which have one more stage than the order of the method. For these methods we can make the error constants very small and error estimates for stepsize control are available within each step. Hence, these methods are much simpler to implement and in a parallel implementation these methods will cost no more than the methods in which the number of stages is equal to the order. These methods have been successfully implemented in a variable stepsize, variable order code and have been used to solve some well-known stiff problems. On a test problem of dimension 400 these methods have shown speedup factors of up to 2.5. Although the present implementation is slightly slower than the sequential codes such as RADAU5 and VODE, they have shown potential as parallel solvers of stiff initial value problems.

Acknowledgements

This study has been completed in the Department of Mathematics at the University of Auckland. I am grateful to my supervisor Prof J. C. Butcher for his guidance throughout the course of my study. His enthusiasm for the subject has been an inspiration to me throughout the three years that I have spent here.

I would like to thank my advisors Dr. R. Chan and Dr. P. Sharp for their assistance. I am grateful to Dr. A. Heard and my fellow students Dr. David Chen and Dr Tina Chan, Miss Nicolette Goodwin, Ms Beth Jackson and Mr William Wright for their help.

During my study I have met many visitors with whom I have had the opportunity of discussing my work. In particular I would like to thank Dr. M. Diamantakis for all his help.

I am grateful to the University of the South Pacific, Suva, Fiji, for providing financial assistance and leave from my job to complete this study. I would like to thank the Department of Mathematics at the University of Auckland for providing some financial assistance in the form of a tutorship.

I would like to thank my husband Yogendra and children Arti and Yuvnit for their patience and help during the course of this study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is widely believed that, the only feasible means of solving computationally intensive systems of differential equations arising in science and engineering, is to use parallel computers effectively. As a result of this, the study of parallel methods for the numerical solution of ordinary differential equations, has become an active area of research. In this thesis, we look at methods which offer the potential for small scale parallelism on computers with a few processors.

1.1 Initial value problems

We consider parallelism in the numerical methods for solving initial value problems (IVPs) for ordinary differential equations (ODEs)

$$\begin{aligned} y'(x) &= f(x, y(x)), \qquad x \in [x_0, x_N], \\ y(x_0) &= y_0, \end{aligned} \tag{1.1}$$

where $y : \mathbb{R} \to \mathbb{R}^m$ and $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$. Systems of ODEs arise very frequently in mathematical modelling and in many practical cases where an exact mathematical solution cannot be found, the need for numerical methods for solving such systems arises.

The interest in parallel IVP solvers arises from the need to be able to solve problems more rapidly than is currently possible. For example, this may be because

the solution is needed in real time, as is the case of flight simulations, or it may be because the computation time required on sequential computers is so large that it affects the productivity of scientists or engineers working on complex systems. The other reason is the widespread availability of parallel computers, with massive computational potential, which provides a challenge for computational scientists to exploit this capability to solve problems more efficiently or to solve problems that are not currently feasible on sequential computers.

As outlined in [60] the IVP (1.2) can be considered too time consuming to solve if

- f is expensive to evaluate,
- the number of equations, m, in the system is large, for example in the spatial discretisation of partial differential equations,
- the interval of integration, $[x_0, x_N]$, is large,
- the IVP must be solved repeatedly, for example, in parameter fitting problems.

Numerical methods for solving the initial value problem typically compute approximations to $y(x_n), \quad n = 1, 2, \ldots, N$ on a mesh $x_0 < x_1 < \cdots < x_N$, where the approximation, $y(x_n)$, depends on one or more of the approximations at the previous steps. Methods which use only one approximation from the past step to calculate the present approximation are known as one-step methods, the most well known of which are the Runge-Kutta methods, although two-step Runge-Kutta methods also exist. Multistep methods use more than one past approximation in calculating the present approximation. These methods are basically sequential in nature, as approximations to the solution are calculated in a step-by-step manner and as such they offer little scope for the use of parallel computers effectively. However, several attempts have been made to adapt existing methods or to develop new methods to exploit parallelism and these have been classified by Gear [47] into two main categories:

- parallelism across the steps,
- parallelism across the method.

Parallelism across steps, as defined in [3], denotes a process in which the solution at different points is calculated simultaneously. Methods which can handle parallelism across steps are suited for parallel computers which have a very large number of processors. Such methods can be used for solving very large problems such as the spatial discretisation of mult-dimensional partial differential equations, with the aim of solving the partial differential equations very accurately. These problems can easily exceed the capacity of the largest supercomputers. At present parallelism of this type is exploited in the spatial domain decomposition of partial differential equations where portions of the spatial domain are assigned to the different processors and parallelism is obtained through the linear algebraic systems associated with the implicit ODE techniques.

Parallelism across a method refers to the use of parallelism inherently available within a method. For example, this could include the parallel evaluation of the stages in a multistage method. This approach is effective for problems with complex function evaluation and particularly for real-time applications for example problems involving active control in which input must be evaluated in time to adjust a strategy as in spacecraft course adjustment or in chemical processing. Realistic chemical reactions and material processing problems involve intricate rate equations and equations of state. Multi-disciplinary applications may involve combined physical, chemical and biological interactions. Such problems may be difficult even in low dimensional situations and parallelising the stages in a multistage method offers a good opportunity to reduce the computational cost. Parallelism of this sort is the main focus of this thesis and we refer to such methods as the parallel methods.

Before we examine some of the parallel methods that have been so far developed for the solution of the IVPs, we introduce some concepts which are necessary to the study of numerical methods for the solution of ordinary differential equations. In particular we look at the existence of unique solutions, the convergence and stability of IVPs. Furthermore, we briefly outline the main numerical methods for the solution of the initial value problems.

Many ODEs that occur naturally are of order greater than one while most of

the numerical methods in existence are designed for solving first order problems only. However, all higher order problems can easily be converted to first order ones by increasing the dimension of the problem. The non-autonomous initial value problem (1.1) can be converted to autonomous form, in which f is does not depend on x explicitly,

$$y'(x) = f(y(x)), \quad y(x_0) = y_0, \quad f : \mathbb{R}^m \to \mathbb{R}^m, \tag{1.2}$$

by adding the differential equation, $y' = 1$, and the corresponding initial condition, $y(x_0) = x_0$, to the system. This increases the dimension of the problem by one.

The existence of a unique solution to the initial value problem is guaranteed by the following theorem.

Theorem 1.1 *Consider the initial value problem (1.1) and let $f(x, y)$ be defined and continuous for all (x, y) in the region D defined by $a \leq x \leq b$, $-\infty < y^q < \infty$, $q = 1, 2, \ldots, m$, where a and b are finite, $y = [y^1, y^2, \ldots, y^m]$, and let there exist a constant L, called the Lipschitz constant, such that*

$$\|f(x, y) - f(x, y^*)\| \leq L\|y - y^*\| \tag{1.3}$$

holds for every $(x, y), (x, y^*) \in D$. Then for any $y_0 \in \mathbb{R}^m$, there exists a unique solution $y(x)$ of the problem (1.1), where $y(x)$ is continuous and differentiable for all $(x, y) \in D$.

When the conditions of this theorem are satisfied, the initial value problem (1.1) is said to be well posed. We will assume that the initial value problems to be solved by the numerical methods are well posed. If $f(x, y)$ is differentiable with respect to y, then using the mean value theorem we have

$$f(x, u) - f(x, v) = \frac{\bar{\partial} f(x, \zeta)}{\partial y}(u - v),$$

where the notation implies that each row of the Jacobian, $\frac{\partial f}{\partial y}$, is evaluated at different mean values of the second argument, all of which are internal points on the line segment in $\mathbb{R}^{m+1}$ from (x, u) to (x, v). It follows that (1.3) can be satisfied by choosing the Lipschitz constant to be

$$L = \sup_{(x,y)\in D} \left\| \frac{\partial f(x, y)}{\partial y} \right\|.$$

1.2 Convergence of method and stability of the IVP

In order to obtain a numerical solution of (1.1), the interval $[x_0, x_N]$ is subdivided into a set of discrete points $\{x_n\}$, $n = 0, 1, \ldots, N$ and approximations y_n to the exact solution $y(x_n)$ are calculated at each of these points. These approximations are calculated sequentially by solving a set of finite difference equation at each step, starting from x_0 and moving towards x_N, where $x_n = x_{n-1} + h_{n-1}$, $n = 1, 2, \ldots, N$. The finite difference equations usually involve approximations to the exact solution as well as approximations to the derivatives at these points and h_n is the steplength at step number n. If we hold the steplength constant and let $h_n = h$, then $x_n = x_0 + nh$. An obvious property required for any numerical method is that as h becomes smaller the numerical approximations, y_n, become more accurate. Considering an arbitrary point $x \in [x_0, x_N]$, where $x = x_0 + nh$, we expect that as $h \to 0$, which means that $n \to \infty$, the approximate solution y_n converges to the exact solution $y(x_n)$. To use a numerical method, for example a k step method, we need to somehow select starting points in the solution sequence

$$\begin{aligned} y_0 &\approx y(x_0), \\ y_1 &\approx y(x_0 + h) = y(x_1), \\ &\vdots \\ y_{k-1} &\approx y(x_0 + (k-1)h) = y(x_{k-1}), \end{aligned}$$

where we are dealing with the initial value problem with $y(x_0)$ given, but where y_0 does not denote this initial value. Assume that these are generated by algorithms that guarantee that

$$\|y_i - y(x_i)\| \to 0 \quad \text{as} \quad h \to 0 \quad \text{for} \quad i = 0, 1, \ldots, k-1.$$

On this understanding we introduce the following definition of convergence.

Definition 1.2 *A numerical method is said to be convergent if, for all initial value*

problems satisfying the hypothesis of Theorem 1.1, we have that

$$\max_{0\leq n\leq N} \|y_n - y(x_n)\| \to 0 \qquad as \quad h \to 0.$$

In the computation of a numerical solution, discretization and roundoff errors cannot be avoided. Consequently, when any numerical method is used, it is necessary that any small errors that are introduced die out or remain bounded. It is possible that the initial value problem is such that when small errors are introduced, these errors do not die out. Here we are looking at a concept of stability of the initial value problem called *total stability*. To define total stability we consider the initial value problem

$$\begin{aligned} z'(x) &= f(x,z) + \delta(x), \qquad x_0 \leq x \leq x_N, \\ z(x_0) &= y_0 + \delta, \end{aligned}$$

which can be considered as a perturbation of the initial value problem

$$\begin{aligned} y'(x) &= f(x,y), \qquad x_0 \leq x \leq x_N, \\ y(x_0) &= y_0. \end{aligned}$$

Definition 1.3 *Let $(\delta(x), \delta)$ and $(\delta^*(x), \delta^*)$ be any two perturbations of the initial value problem (1.1) and let $z(x)$ and $z^*(x)$ be the resulting perturbed solutions. Then, if there exists a positive constant S such that, for all $x \in [x_0, x_N]$,*

$$\|z(x) - z^*(x)\| \leq S\epsilon,$$

for $\epsilon > 0$, whenever

$$\|\delta(x) - \delta^*(x)\| \leq \epsilon \qquad and \qquad \|\delta - \delta^*\| \leq \epsilon,$$

then the initial value problem (1.1) is said to be totally stable or well posed.

The value of S can be as large as necessary, as long as it is finite. It can be shown that when the initial value problem satisfies the assumptions of Theorem

1.1 then it is totally stable. However, there are problems which are totally stable but *ill-conditioned* with respect to the numerical computation. To illustrate this consider the differential equation

$$y' = 100y - 101e^{-x}, \qquad y(0) = 1,$$

which has solution $y(x) = e^{-x}$. The perturbed problem

$$\hat{y}' = 100\hat{y} - 101e^{-x}, \qquad \hat{y}(0) = 1 + \delta,$$

has solution $\hat{y}(x) = \delta e^{100x} + e^{-x}$. Since,

$$|y(x) - \hat{y}(x)| = \delta e^{100x},$$

the error grows as e^{100x} and it is quite clear that this will grow rapidly, no matter how accurate the numerical method is, and the computed solution will very quickly depart from the exact solution. Problems such as this are said to be ill-conditioned, although they are rarely found in practice.

1.3 Stability of the numerical method

In addition to the stability of the initial value problem, we require a numerical method used to solve it to be stable. Any numerical method applied to an initial value problem will introduce errors due to discretization and round-off. If the numerical method is to approximate the solution to the initial value problem then it must not be over-sensitive to these errors and, in particular, these errors should not increase. This requirement is known as *zero-stability* and we give the following definition from Lambert [62].

Definition 1.4 *Let* $\{\delta_n, n = 0, 1, \ldots, N\}$ *and* $\{\delta_n^*, n = 0, 1, \ldots, N\}$ *be any two perturbations of the difference method used for the numerical solution of (1.1), and let* $\{z_n, n = 0, 1, \ldots, N\}$ *and* $\{z_n^*, n = 0, 1, \ldots, N\}$ *be the resulting perturbed solutions. Then, if there exist constants* S *and* h_0 *such that, for all* $h \in (0, h_0]$,

$$\|z_n - z_n^*\| \leq S\epsilon, \qquad 0 \leq n \leq N,$$

given $\epsilon > 0$*, whenever*

$$\|\delta_n - \delta_n^*\| \leq \epsilon, \qquad 0 \leq n \leq N,$$

we say that the difference method is zero-stable.

Since zero-stability requires the inequalities to hold for values of $h \in (0, h_0]$, it is therefore concerned with the behaviour of the difference equations as $h \to 0$. In solving an initial value problem using any numerical method, the most dominant type of error is the discretization error. This is the error introduced by replacing the infinite process of integration by the finite process of solving difference equations. Zero-stability ensures that these errors will remain bounded.

In the use of numerical methods, the behaviour of the computed solutions when h is held constant as $N \to \infty$, is also important. This leads to the concepts of *absolute stability* and the *region of absolute stability*. Absolute stability is referred to as stability in short. The study of stability is dependent on the numerical method and the problem. In order to examine the stability of a numerical method, we usually use the scalar test equation

$$y' = qy, \qquad q \in \mathbb{C}, \qquad \mathrm{Re}(q) < 0. \tag{1.4}$$

A numerical method must be stable in a region of the product hq which should be as extensive as possible.

Definition 1.5 *The region of absolute stability of a method is the set of values of* hq *for which a perturbation in a single value* y_n *will produce a change in subsequent values which does not increase from step to step.*

For example, we consider the region of absolute stability of the explicit Euler method,

$$y_{n+1} = y_n + hf(x_n, y_n). \tag{1.5}$$

When this is applied to the test equation (1.4) , we obtain

$$y_{n+1} = (1 + qh)y_n = (1 + qh)^{n+1}y(0). \tag{1.6}$$

When $\mathrm{Re}(q) < 0$, the exact solution of (1.4), $y(x_0)e^{qx}$, decays monotonically. It is clear that the numerical solution (1.6) will decay in a similar way only if $|1 + qh| < 1$. This represents the interior of a circle centred at $(-1, 0)$ on the complex plane and this is the region of absolute stability of the explicit Euler method. If qh lies inside this circle then the solution, y_n, satisfies $y_n \to 0$ as $x \to \infty$, but not otherwise. When $\mathrm{Re}(q) \ll 0$, this requirement imposes severe restrictions on the size of h that can be used to satisfactorily solve the problem. Hence, the explicit Euler method is not suitable for solving such problems as it has a *bounded region of absolute stability.*

Since we are examining the stability of numerical methods for solving an initial value problem of dimension m, it is appropriate that we consider the linear test equation, $y' = Ay$, which is a linear homogeneous system with constant coefficients. However, this system can be easily decoupled and transformed to a system of independent equations of the form (1.4). Hence, the stability of a numerical method when applied to $y' = Ay$ can be examined by checking if the method is stable for each of the equations of the transformed system for the chosen value of h. Since the test equation (1.4) is linear this type of stability is referred to as *linear stability.*

In the case of nonlinear initial value problems the concept of frozen Jacobian may be used. If $y' = f(x, y)$ represents a nonlinear system and $z(x) = y(x) + \delta(y(x))$ is a perturbed solution then

$$z' = f(x, z) \qquad or \qquad y' + (\delta y)' = f(x, y + \delta y).$$

When perturbations are small, the above system can be approximated by

$$y' + (\delta y)' = f(x, y) + \frac{\partial f}{\partial y}\delta y.$$

If we assume that the Jacobian, $\frac{\partial f}{\partial y}$, is locally constant or frozen, then, small perturbations of y approximately satisfy the linear system

$$(\delta y)' = A\delta y,$$

where A is the frozen Jacobian matrix. This implies that the linear scalar test equation (1.4) can now be used to examine the stability of the original nonlinear

equation. Although this idea of frozen Jacobians can give some insight into the stability of nonlinear problems [70], it can give misleading results. It is possible to find nonlinear systems in which the eigenvalues of the frozen Jacobian have negative real parts but the solution is non-decaying. It is also possible to find nonlinear problems in which the eigenvalues of the frozen Jacobian are positive but the solution is decaying [62]. This implies that one cannot rely entirely on the arguments from linear stability theory to generalise the behaviour of nonlinear problems. Some concepts of nonlinear stability will be discussed in Chapter 2.

1.4 Numerical methods for the IVP

The development of numerical methods for the solution of initial value problems is a very active area. Many new numerical methods are developed every year. However, all of the methods can be placed in one of the existing classes, *Runge-Kutta methods*, *linear multistep methods*, *Rosenbrock methods*, or *general linear methods*. Runge-Kutta methods and linear multistep methods are amongst the most successfully implemented numerical methods. General linear methods were introduced by Butcher [12] to provide a unified framework for the analysis of all numerical methods for ordinary differential equations. These methods combine the essential multivalue nature of the linear multistep methods with the multistage nature of Runge-Kutta methods. Hence, this formulation includes linear multistep methods, Runge-Kutta methods and various generalisations such as hybrid methods. In the following sections we briefly outline these methods.

1.4.1 General linear methods

General linear methods are also known as multivalue methods. A general linear method for solving the initial value problem (1.1) can be written as

$$Y_i = h\sum_{j=1}^{s} a_{ij} f(x_n + c_j h, Y_j) + \sum_{j=1}^{r} u_{ij} y_j^{[n]}, \quad i = 1, \ldots, s, \tag{1.7}$$

$$y_i^{[n+1]} = h\sum_{j=1}^{s} b_{ij} f(x_n + c_j h, Y_j) + \sum_{j=1}^{r} v_{ij} y_j^{[n]}, \quad i = 1, \ldots, r, \tag{1.8}$$

and more compactly, using $A = [a_{ij}]$, $B = [b_{ij}]$, $U = [u_{ij}]$ and $V = [v_{ij}]$, in the form

$$\begin{bmatrix} Y \\ y^{[n+1]} \end{bmatrix} = \begin{bmatrix} A \otimes I_m & U \otimes I_m \\ B \otimes I_m & V \otimes I_m \end{bmatrix} \begin{bmatrix} hF(Y) \\ y^{[n]} \end{bmatrix}, \tag{1.9}$$

where

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \quad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_r^{[n]} \end{bmatrix}, \quad F(Y) = \begin{bmatrix} f(x_n + c_1 h, Y_1) \\ f(x_n + c_2 h, Y_2) \\ \vdots \\ f(x_n + c_s h, Y_s) \end{bmatrix},$$

and $\otimes$ represents the Kronecker product which is defined by

$$P \otimes W = \begin{bmatrix} p_{11}W & p_{12}W & \cdots & p_{1l}W \\ p_{21}W & p_{22}W & \cdots & p_{2l}W \\ \vdots & \vdots & & \vdots \\ p_{k1}W & p_{k2}W & \cdots & p_{kl}W \end{bmatrix} \in \mathbb{R}^{k\mu \times l\nu}$$

for any matrices $P = [p_{ij}] \in \mathbb{R}^{k\times l}$ and $W \in \mathbb{R}^{\mu\times\nu}$. The vector $c = [c_1, c_2, \ldots, c_s]^T$ is frequently called the abscissae. The Y_i are called the internal stages and $y_i^{[n]}$ the external stages. The external stages can consist of y-values, hy'-values or arbitrary linear combinations of such quantities. The representation of general linear methods by the coefficient matrices A, U, B and V is in some sense not unique. Two different methods may be equivalent in that their external stage vector, $y^{[n]}$, are related by a linear combination.

The complexity of implementation of general linear methods depend on the structure of the matrix A, as for Runge-Kutta methods. When the abscissaes are in increasing order, a general linear method as stated above, is explicit if the A matrix is strictly lower triangular and implicit otherwise.

Example 1.1 A general linear method with $r = s = 2$ and order 4 was derived by Dekker [39]

$$\left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] = \left[\begin{array}{cc|cc} \frac{2}{3} & 0 & 1 & -\frac{7}{6} \\ \frac{2}{3} & \frac{2}{3} & 1 & \frac{1}{6} \\ \hline \frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{6} \\ 0 & 1 & 0 & 0 \end{array}\right].$$

This method is diagonally implicit and thus can be implemented efficiently. □

The Runge-Kutta and the linear multistep methods are important methods in their own right and these methods have been usually studied separately. However, these methods can be considered as special cases of general linear methods.

1.4.2 Runge-Kutta methods

The simplest and most well known of all numerical methods is perhaps the explicit Euler method

$$y_{n+1} = y_n + hf(x_n, y_n).$$

It is first-order only and so has very low accuracy. Runge-Kutta methods are generalizations of the Euler method in which the one-step format is retained but the linearity with respect to the function evaluations is sacrificed.

Although the derivation of Runge-Kutta methods can be traced back to about 100 years, its development into a major theory is due to Butcher and a comprehensive account of this can be found in the book [18]. An s-stage Runge-Kutta method is

defined by

$$Y_i = y_n + h\sum_{j=1}^{s} a_{ij} f(x_n + hc_j, Y_j), \qquad i = 1, 2, \ldots, s, \tag{1.10}$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i f(x_n + hc_i, Y_i), \tag{1.11}$$

where the Y_i are the internal stages and y_n is the calculated approximation to the solution $y(x_n)$. Alternatively, these equations can be written in a tableau form

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

It is always assumed that the coefficients in (1.10) satisfy the row sum condition, $c_i = \sum_{j=1}^{s} a_{ij}$. A Runge-Kutta method satisfies the general linear format with $U = e$, the vector of ones, $B = b^T$, $V = 1$, $y^{[n+1]} = y_{n+1}$ and $y^{[n]} = y_n$. The structure of matrix A of a Runge-Kutta method plays a very important role in the computational cost of the method, and based on this, all Runge-Kutta methods are classified as

- *explicit* when $a_{ij} = 0$ for all $i \leq j$, which means that matrix A is strictly lower triangular,
- *semi-implicit* when $a_{ij} = 0$ for all $i < j$ and at least one $a_{ii} \neq 0$,
- *implicit* when $a_{ij} \neq 0$ for some $i < j$,

where it is assumed that the abscissaes are arranged in increasing order.

Example 1.2 The classical fourth order method is an example of an explicit

method and can be written in general linear format as

$$\left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 & 0 & 1 \\ 0 & \frac{1}{2} & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ \hline \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & 1 \end{array}\right].$$

□

In the derivation of a Runge-Kutta method of a particular order, the matrix A and vectors b and c are chosen to satisfy the order conditions. For low order methods this can be done easily but for higher orders the derivations get quite complicated. A systematic approach to their derivation is the Butcher theory in which the order conditions are related to a structure called *rooted trees*. For a method of a particular order, using the Butcher theory, one gets a set of nonlinear equations that need to be solved in order to get the method coefficients. Implicit Runge-Kutta methods are investigated in Chapter 2.

Definition 1.6 *If $y(x_n)$ and $y(x_{n+1})$ are the exact solutions of (1.1) at the points x_n and x_{n+1} respectively, then the local truncation error (LTE), denoted by T_{n+1} for an s-stage Runge-Kutta method (1.10-1.11) is defined by*

$$T_{n+1} = y(x_{n+1}) - y(x_n) - h\sum_{i=1}^{s} b_i f(x_n + hc_i, Y_i). \tag{1.12}$$

Definition 1.7 *An s-stage Runge-Kutta method (1.10-1.11) is said to be consistent, if for all initial value problems satisfying the hypothesis of Theorem 1.1, the local truncation error T_{n+1}, defined by (1.12), satisfies*

$$\lim_{h\to 0} \frac{T_{n+1}}{h} = 0.$$

Using Taylor series expansion, we get

$$\begin{aligned} T_{n+1} &= y(x_{n+1}) - y(x_n) - h\sum_{i=1}^{s} b_i f(x_n + hc_i, Y_i) \\ &= y(x_n) + hy'(x_n) + O(h^2) - y(x_n) - h\sum_{i=1}^{s} b_i f(x_n, y(x_n)) \\ &= hy'(x_n) - h\sum_{i=1}^{s} b_i y'(x_n) + O(h^2). \end{aligned}$$

If the coefficients, b_i, satisfy the condition

$$\sum_{i=1}^{s} b_i = 1, \tag{1.13}$$

then we see that $T_{n+1} = O(h^2)$ and the method is consistent, since,

$$\lim_{h\to 0} \frac{T_{n+1}}{h} = \lim_{h\to 0} O(h) = 0.$$

The condition (1.13) is called the consistency condition. We can now state the necessary conditions for convergence.

Theorem 1.8 *A numerical method is convergent if and only if it is zero-stable and consistent.*

Theorem 1.8 is the fundamental theorem in this subject was first proved for the linear multistep methods by Dahlquist [37]. In the next section we shall see that a Runge-Kutta method is always zero-stable and consequently a consistent Runge-Kutta method is always convergent. Runge-Kutta methods have local truncation errors of the form

$$T_{n+1} = C(y(x_n))h^{p+1} + O(h^{p+2}),$$

where p is an integer greater than or equal to 1 and is called the order of accuracy and C is a function that depends on the elementary differentials calculated at $(x_n, y(x_n))$. The term $C(y(x_n))h^{p+1}$ is called the *principal local truncation error* (PLTE) of the method. It is noted that a consistent numerical method has order at least one.

1.4.3 Linear multistep methods

The general form of a linear multistep method with k steps is

$$y_n = \sum_{j=1}^{k} \alpha_j y_{n-j} + h \sum_{j=0}^{k} \beta_j f_{n-j}, \tag{1.14}$$

where $f_n = f(x_n, y_n)$. To compute a solution at x_n using a k step multistep method requires the numerical solution calculated at the previous k points, $x_{n-1}, x_{n-2}, \ldots, x_{n-k}$. The multistep method is explicit if $\beta_0 = 0$ and implicit otherwise.

The linear multistep method (1.14) can be cast in the general linear format with

$$\left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] = \left[\begin{array}{c|cccccccccc} \beta_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{k-1} & \alpha_k & \beta_1 & \beta_2 & \cdots & \beta_{k-1} & \beta_k \\ \hline \beta_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{k-1} & \alpha_k & \beta_1 & \beta_2 & \cdots & \beta_{k-1} & \beta_k \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 1 & 0 \end{array}\right].$$

In this representation

$$\begin{aligned} Y_1^{[n]} &= y_1^{[n]} = y_n, \quad y_2^{[n]} = y_{n-1}, \quad \cdots, \quad y_k^{[n]} = y_{n-k+1}, \\ y_{k+1}^{[n]} &= hf_n, \quad y_{k+2}^{[n]} = hf_{n-1}, \quad \cdots, \quad y_{2k}^{[n]} = hf_{n-k+1}. \end{aligned}$$

Example 1.3 The Adams-Bashforth method,

$$y_n = y_{n-1} + h\left(\frac{3}{2}f_{n-1} - \frac{1}{2}f_{n-1}\right),$$

in the general linear formulation has,

$$\left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] = \left[\begin{array}{c|ccc} 0 & 1 & \frac{3}{2} & -\frac{1}{2} \\ \hline 0 & 1 & \frac{3}{2} & -\frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right],$$

where

$$Y_1^{[n]} = y_1^{[n]} = y_n, \quad y_2^{[n]} = hf_n, \quad y_3^{[n]} = hf_{n-2}.$$

□

The local truncation error of the linear multistep method, (1.14), is defined as

$$T_{n+k} = y(x_n) - \sum_{j=1}^{k} \alpha_j y(x_n - jh) - h \sum_{j=0}^{k} \beta_j y'(x_n - jh), \tag{1.15}$$

since we have $y'(x_n - jh) = f(x_n - jh, y(x_n - jh))$. Assuming that $y(x)$ is differentiable sufficiently many times and expanding (1.15) by using a Taylor series, we obtain

$$T_{n+k} = C_0 y(x_n) + C_1 y'(x_n) + \cdots + C_p h^p y^{(p)}(x_n) + C_{p+1} h^{p+1} y^{(p+1)}(x_n) + \cdots$$

where $C_0, C_1, \ldots$ are constants.

Definition 1.9 *The linear multistep method (1.14) is said to be of order p if*

$$C_0 = C_1 = \cdots = C_p = 0, \quad C_{p+1} \neq 0.$$

Therefore the local truncation error of a linear multistep method of order p is

$$T_{n+k} = C_{p+1} y^{(p+1)}(x_n) + O(h^{p+2}) = O(h^{p+1}),$$

where the term, $C_{p+1} y^{(p+1)}(x_n)$, is the principal local truncation error and the coefficient, C_{p+1}, is called the error constant of the method. An order $p \geq 1$ linear mutistep method is, by definition, consistent. For linear multistep methods the following alternative definition of zero-stability can be given.

Definition 1.10 *The linear multistep method (1.14) is said to be zero-stable if all the roots of the characteristic polynomial,* $\rho(\theta) = 1 - \sum_{j=1}^{k} \alpha_j \theta^j$*, have modulus less than or equal to* 1 *and every root with modulus equal to* 1 *is simple.*

This alternative definition of zero-stability is equivalent to Definition 1.4 and is widely known as the root condition. Runge-Kutta methods are a special case of linear mutistep methods. Therefore, using the above definition of zero-stability we see that a Runge-Kutta method will always be zero-stable since its characteristic polynomial, $\rho(\theta) = 1 - \theta$, has only a simple root whose modulus is 1.

Zero-stability is a severe restriction on linear mutistep methods and it restricts their order for a given value of k. The following theorem has been proved by Dahlquist [38] and is known as Dahlquist's first barrier.

Theorem 1.11 *No zero-stable linear* k*-step method can have order exceeding* $k+1$ *when* k *is odd and* $k+2$ *when* k *is even.*

Within the class of linear mutistep methods there are several sub-classes. One important sub-class is the *Adams* methods. These have the form

$$y_n = y_{n-1} + h \sum_{j=0}^{k} \beta_j f_{n-j},$$

with characteristic polynomial, $\rho(\theta) = \theta^k - \theta^{k-1}$, and are clearly zero-stable. Adams methods which are implicit are called the the *Adams-Moulton methods* while the explicit methods are called *Adams-Bashforth methods*. These methods are often used as *predictor-corrector methods* in which the explicit method is used as the predictor and the implicit method is used as a corrector.

Example 1.4 The following is an example of an Adams Bashforth, Adams Moulton pair used as a predictor-corrector method,

$$\begin{aligned}
P: \quad y_{n+1}^{[0]} &= y_n + \frac{h}{12}\left[23f_n - 16f_{n-1} + 5f_{n-2}\right], \\
E: \quad f_{n+1}^{[l]} &= f(x_{n+1}, y_{n+1}^{[l]}) \quad l = 0, 1, \ldots, M-1, \\
C: \quad y_{n+1}^{[l+1]} &= y_n + \frac{h}{24}\left[9f_{n+1}^{[l]} + 19f_n - 5f_{n-1} + f_{n-2}\right], \quad l = 0, 1, \ldots, M-1, \\
E: \quad f_{n+1}^{[M]} &= f(x_{n+1}, y_{n+1}^{[M-1]}),
\end{aligned}$$

where the upper index M means that the corrector is iterated M times until convergence. Implicit equations as in step C are solved either by fixed point iteration or Newton iteration. The above scheme constitutes a $P(EC)^M E$ predictor-corrector method. Sometimes the final function evaluation is omitted in which case we have the $P(EC)^M$ mode. □

Although the explicit Runge-Kutta methods and the Adams predictor-corrector methods are widely used for solving initial value problems, they are not suitable for stiff problems since they have bounded regions of absolute stability.

1.5 An overview of this thesis

In Chapter 2 we study the phenomena of stiffness and examine numerical methods which are suitable for the solution of stiff initial value problems. We also examine some of the existing parallel numerical methods for stiff problems. In Chapter 3 we analyse DIMSIM methods in general and type 4 methods in which the order is equal to the number of stages. In Chapter 4 we look at a modification of these methods with a view to their implementation in a variable stepsize, variable order code. We develop procedures for estimating the errors, controlling stepsize and order. Some results are presented and some difficulties with the use of these methods are outlined. In Chapter 5 we derive a second order type 4 method in which the diagonal elements in the A matrix are allowed to vary as a result of which the error constant of the method is reduced. We further derive a new set

of type 4 methods which has one more stage than the order of the method and smaller error constants. A–stable methods of orders 1 to 8, and ways of estimating errors in these methods are proposed. Some numerical results for solving some stiff problems are given and some difficulties with the use of these methods are outlined.

Chapter 2

Numerical methods for stiff problems

Since we are interested in parallel numerical methods for stiff initial value problems, we need to understand stiffness. In this chapter we make an attempt to define the phenomenon of stiffness and to analyse the properties of stiff systems. We further consider stability properties necessary for the solution of stiff problems and investigate numerical methods that are suitable for solving stiff problems.

2.1 The phenomenon of stiffness

Broadly, initial value problems are of two types, *stiff* and *nonstiff*, although the distinction between the two groups is not completely clearcut. Stiff problems occur in many areas of mathematical modelling such as chemical kinetics, electric circuits, fluid dynamics and robotics. As we shall see in the next few sections some numerical methods encounter difficulties when they are used to solve stiff problems, and consequently, such methods are not appropriate for solving such problems.

Stiffness is a phenomenon which leads to some practical difficulties when numerically solving some initial value problems. There is no one precise definition of

stiffness. However, many qualitative statements such as the following can be made in order to explain the notion of stiffness.

- Stiffness occurs when some components of the solution decay much more rapidly than the others.
- If a numerical method with a bounded region of absolute stability, when applied to a system with any initial condition, is forced to use an excessively small stepsize in relation to the smoothness of the solution, then the system is stiff. This means that stability rather than accuracy restricts the stepsize.

To illustrate these concepts, we consider some examples.

Example 2.1 Consider the system

$$\begin{aligned} y_1' &= y_1 - 2y_2, \\ y_2' &= 1001y_1 - 1002y_2. \end{aligned} \tag{2.1}$$

This is a constant coefficient linear system with Jacobian eigenvalues $\lambda_1 = -1$, $\lambda_2 = -1000$ and *general solution*,

$$\begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} = k_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{-x} + k_2 \begin{bmatrix} \frac{2}{1001} \\ 1 \end{bmatrix} e^{-1000x}. \tag{2.2}$$

With the initial conditions $y_1(0) = 1$ and $y_2(0) = -1$, the system has a particular solution,

$$\begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} = \frac{1003}{999} \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{-x} - \frac{2002}{999} \begin{bmatrix} \frac{2}{1001} \\ 1 \end{bmatrix} e^{-1000x}.$$

The component of the solution involving the term e^{-1000x} decays very fast and is called the fast transient, whereas the other component involving e^{-x}, decays very slowly, compared to the fast component, and is called the slow transient. It is the presence of these two widely different decay scales which cause problems for some numerical methods when used to solve such a problem. It can be easily seen that the system (2.1) can be decoupled to two independent differential equations,

$$\begin{aligned} z' &= -z, \\ w' &= -1000w. \end{aligned} \tag{2.3}$$

In order to illustrate the difficulty experienced by explicit methods, we solve the second differential equation using the explicit Euler method (1.5) and obtain

$$w_{n+1} = (1 - 1000h)w_n = (1 - 1000h)^{n+1}w(0)$$

The exact solution decays with increasing x, and we require the numerical solution to do the same. This is obviously only possible when $|1 - 1000h| < 1$ or equivalently, $500h < 1$. This means that the stepsize h will need to be extremely small for otherwise the numerical solution will diverge. Since most other explicit methods also have similar bounds on the stepsize, they impose severe restrictions on stepsize. In the solution of stiff problems, it is often required to integrate the problem until all the transients die out. The presence of a slow transient will force the method to integrate for a long time using a very small stepsize. This results in high computational costs and is usually unacceptable.

Stiffness is a property of a system of differential equations and its general solution, and not the particular solution. For example, the system (2.1) with initial conditions, $y_1(0) = 1$ and $y_2(0) = 1$, has the exact solution

$$\begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} e^{-x}.$$

In this situation even though the fast transient is not present in the exact solution, the presence of the fast transient in the general solution (2.2) will still cause any explicit numerical method to encounter stepsize restrictions. Hence, stiffness is a property of the general solution and not a particular solution.

In contrast, the use of an implicit method such as the implicit Euler method,

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}), \tag{2.4}$$

to solve (2.3), leads to

$$\begin{aligned} w_{n+1} &= w_n + 1000hw_{n+1} \\ &= \frac{1}{1 + 1000h} w_n \\ &= \left(\frac{1}{1 + 1000h}\right)^{n+1} w(0) \end{aligned}$$

and since $\frac{1}{1+1000h} < 1 \ \forall \ h > 0$, stability considerations do not restrict the size of h. This is true for many implicit methods. Hence, we see that implicit methods are often more suited to solving stiff problems. □

The last example is a system of ordinary differential equations. However, stiffness can occur for a single differential equation.

Example 2.2 The Prothero-Robinson problem,

$$y'(x) = m(y(x) - g(x)) + g'(x), \qquad \mathrm{Re}(m) \ll 0, \tag{2.5}$$

where $g(x)$ is a smooth function, has exact solution

$$y(x) = g(x) + k\exp(m(x - x_0)), \qquad k = y(x_0) - g(x_0).$$

Since $\mathrm{Re}(m) \ll 0$, the exponential term in the solution will decay rapidly and become insignificant compared to the first term. The degree of stiffness in (2.5) depends on how negative $\mathrm{Re}(m)$ is. This problem was used by Prothero and Robinson to study a phenomenon called *order reduction* which will be discussed later on. □

There are many other examples of stiff problems which are traditionally used for testing new numerical methods. We consider two examples which will be used in a later chapter to test some numerical methods.

Example 2.3 The *Robertson problem* [68] is a well known ODE system which comes from chemical kinetics and contains fast and slow transients. This is a mildly stiff, nonlinear three dimensional problem, described by

$$\begin{cases} y_1'(x) &= -0.04y_1(x) + 10^4 y_2(x)y_3(x), & y_1(0) &= 1, \\ y_2'(x) &= 0.04y_1(x) - 10^4 y_2(x)y_3(x) - 3\times 10^7 y_2^2(x), & y_2(0) &= 0, \\ y_3'(x) &= 3\times 10^7 y_2^2(x), & y_3(0) &= 0. \end{cases} \tag{2.6}$$

Since the problem is nonlinear we cannot decouple the individual equations but we can get an idea of the stiffness of the system by examining its Jacobian matrix

$$\frac{\partial f}{\partial y} = \begin{bmatrix} -0.04 & 10^4 y_3 & 10^4 y_2 \\ 0.04 & -10^4 y_3 - 6\times 10^7 y_2 & -10^4 y_2 \\ 0 & 6\times 10^7 y_2 & 0 \end{bmatrix}. \tag{2.7}$$

At $x = 0$ the eigenvalues of the Jacobian are, -0.04, 0 and 0 and the problem is not stiff. Eventually this system tends to an equilibrium state, as shown in Figure 2.1, with $y_1 = y_2 = 0$ and $y_3 = 1$ as $x \to \infty$ and the Jacobian has eigenvalues 0, 0, $-10^4 - 0.04$. The problem becomes stiff immediately after $x = 0$.

Due to the physical nature of the problem, the three components have non-negative values and since $\sum_{i=1}^{3} y_i' = 0$, the total of the three components is one during the integration. Although physically no solution component can be zero, it is difficult to avoid the numerical method producing negative solutions. It is extremely difficult to deal with y_2 becoming negative as the endpoint of integration, x_{end}, becomes large, for example, 10^{10}. □

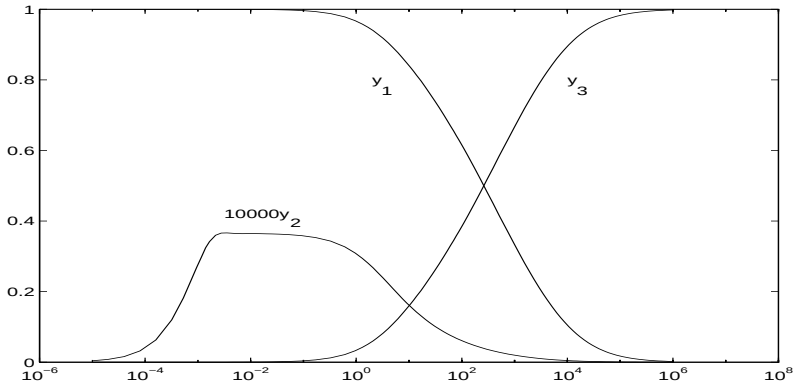

Figure 2.1: Solution profile: Robertson problem.

Example 2.4 Another well known stiff problem which many numerical methods have difficulty solving is the van der Pol problem [49]

$$\begin{cases} y_1'(x) = y_2(x), & y_1(0) = 2, \\ y_2'(x) = \epsilon(1 - y_1^2(x))y_2(x) - y_1(x), & y_2(0) = 0, \end{cases} \tag{2.8}$$

with Jacobian

$$\frac{\partial f}{\partial y} = \begin{bmatrix} 0 & 1 \\ -2\epsilon y_1 y_2 - 1 & \epsilon(1 - y_1^2) \end{bmatrix}. \tag{2.9}$$

An analysis of this problem for different magnitudes of ϵ is given in [70]. Basically, the increase in the magnitude of ϵ increases the stiffness of the system. This system is more interesting when $\epsilon \gg 1$ because the non-linearity becomes important then. Using $\epsilon = 10^6$, it can be seen in Figure 2.2 that the periodic solution in one cycle is

composed of a segment in which y_1 changes slowly, and a short segment in which it changes very quickly, another segment of slow change, and a final segment of rapid change. The second component of the solution, y_2, can be explained as follows. Since ϵ is large, the derivative of y_2' with respect to y_2 is very large and negative when $y_2' \approx 0$ and $|y_1| > 1$. Hence, the solution will rapidly approach $y_2' = 0$, which means that $y_2 = y_1/\epsilon(1 - y_1^2)$. Moreover, since y_1' depends on y_2, $y_1' \to \infty$ as $y_2 \to \infty$ when $y_1 \approx \pm 1$. Since the solution changes rapidly near some values of x the difficulty due to stiffness is compounded by the need to adapt stepsize rapidly to this change. □

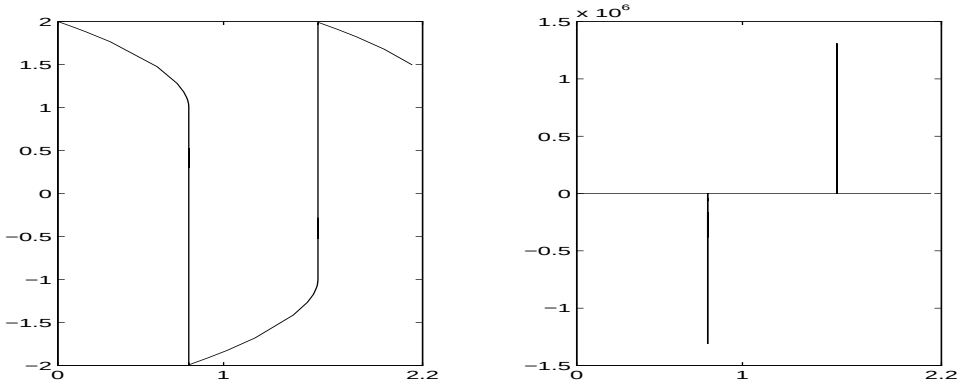

Figure 2.2: Solution profile: van der Pol problem, y_1 on left and y_2 on right.

2.2 Stability and stiffness

From the previous section it is apparent that the explicit methods have bounded regions of absolute stability. The bigger this region is, the less severe the restriction on h. Therefore, using an explicit method always leads to some restriction on the values of h that can be used. For the solution to stiff problems we would ideally like to use numerical methods in which there is no restriction at all on h. This implies that the region of absolute stability of the numerical methods should be the whole of the left half of the complex plane. This leads to the following definition of *A-stability* by Dahlquist [38].

Definition 2.1 *A numerical method is said to be A-stable if its region of absolute stability includes the whole of the left half plane.*

Concerning the linear stability properties of linear mutistep methods Dahlquist [38] proved the following theorem.

Theorem 2.2 *(i) An explicit linear multistep method cannot be A-stable.*
(ii) The order of an A-stable linear multistep method cannot exceed 2.

This theorem is often known as the *second Dahlquist barrier*. The most well known first order A-stable method is the implicit Euler method (2.4). Dahlquist [38] showed that the Trapezoidal rule is the second order A-stable linear multistep method which has the smallest error constant.

In one sense A-stability is too demanding a requirement, particularly for linear multistep methods, since these methods cannot be A-stable for orders greater than two. However, linear multistep methods of higher orders have some computational advantages and in order to be able to use some of these methods for the solution of stiff problems we consider two properties weaker than A-stability, the concepts of *A(α)-stability and stiff-stability.*

For many practical problems with eigenvalues not close to the imaginary axis, a method having a stability region which is only part of the left half plane and includes the negative real axis is sufficient to solve such problems. This motivates the following definition of $A(\alpha)$-stablility proposed by Widlund [73].

Definition 2.3 *A method is said to be $A(\alpha)$-stable, $\alpha \in (0, \pi/2)$ if*

$$R_A \supseteq \{z \in \mathbb{C} : -\alpha < \pi - arg(z) < \alpha\},$$

where R_A is the region of absolute stability of the method.

For many problems the eigenvalues which produce the fastest transients all lie to the left of the line $\mathrm{Re}(z) = -a$, where $a > 0$ and the remaining eigenvalues clustered fairly close to the origin. In this case we are assuming that there are no eigenvalues with small negative real part and large imaginary part. This motivates the concept of stiff-stability which was introduced by Gear [45].

Definition 2.4 *A method is said to be stiffly-stable if* $R_A \supseteq R_1 \cup R_2$, *where* $R_1 = \{z \in \mathbb{C} : Re(z) < -a\}$, $R_2 = \{z \in \mathbb{C} : -a \leq Re(z) < 0, -c \leq Im(z) \leq c\}$, *and* a *and* c *are positive real numbers and* R_A *is the region of absolute stability.*

Although A-stability ensures that there are no restrictions on h, it does not always ensure that $y_n \to 0$ as $x \to \infty$ rapidly, as in the case for the solution of the test equation. For example, consider the Trapezoidal Rule

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right), \tag{2.10}$$

which, when applied to the test equation (1.4) gives

$$y_{n+1} = R(z)y_n = (R(z))^{n+1}\, y(0),$$

where

$$R(z) = \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}},$$

is called the *stability function* of the method and $z = qh$. The method is A-stable. However, when $|\mathrm{Re}(q)|$ is large and h not very small, $R(z)$ will be close to -1. This means that the term $(exp(z))^n$ which tends to zero rapidly as $n \to \infty$ is being approximated by $(R(z))^n$ which tends to zero very slowly, and with alternating sign, as $n \to \infty$. We expect a slowly damped oscillating error when the Trapezoidal rule is applied in such situations. This leads to the following definition of *L-stability* by Ehle [41] and Axelsson [2].

Definition 2.5 *A one-step method is said to be L-stable if it is A-stable and, in addition, when applied to the scalar test equation (1.4), it yields* $y_{n+1} = R(z)y_n$, *and*

$$\lim_{|z| \to \infty} R(z) = 0.$$

Thus, the Trapezoidal rule is not L-stable but the Implicit Euler method, with $R(z) = 1/(1 - z)$, is L-stable. When the general linear method (1.9) is applied to the scalar test equation (1.4), we obtain

$$\begin{aligned} Y &= [I - zA]^{-1} U y^{[n]}, \\ y^{[n+1]} &= zB[I - zA]^{-1} U y^{[n]} + V y^{[n]}, \end{aligned}$$

from which we get

$$y^{[n+1]} = M(z)y^{[n]},$$

where

$$M(z) = V + zB[I - zA]^{-1}U, \tag{2.11}$$

is called the *stability matrix* of the general linear method. The method (1.9) is stable if $M(z)$ is power bounded. For a Runge-Kutta method the stability matrix (2.11) simplifies to the stability function

$$R(z) = 1 + zb^T(I - zA)^{-1}e. \tag{2.12}$$

The following definition is due to Prothero and Robinson [67].

Definition 2.6 *An implicit Runge-Kutta method with nonsingular A is said to be stiffly-accurate if*

$$\lim_{|z|\to\infty} R(z) = 0. \tag{2.13}$$

This definition of stiff accuracy is exactly the same as the definition of L-stability for A-stable methods. Hence, a stiffly-accurate A-stable method is an L-stable method.

Theorem 2.7 *If an s-stage implicit Runge-Kutta method with nonsingular A satisfies the following conditions*

$$(i) \qquad a_{sj} = b_j, \qquad j = 1, 2, \ldots, s,$$

$$(ii) \qquad c_s = 1,$$

then the method is stiffly-accurate.

Proof This theorem can be easily proved by considering (2.12) and taking the limit

$$\lim_{|z|\to\infty} R(z) = 1 - b^T A^{-1} e.$$

From condition (i) we have

$$b = [a_{s1}, a_{s2}, \ldots, a_{ss}]^T = A^T e_s,$$

where $e_s = [0, \ldots, 0, 1]^T$. Hence,

$$\lim_{|z|\to\infty} R(z) = 1 - e_s^T A A^{-1} e = 1 - 1 = 0.$$

Condition (ii) of the above theorem guarantees that the Runge-Kutta method is consistent if condition (i) holds, since we have

$$\sum_{j=1}^{s} a_{sj} = c_s, \qquad \text{row sum condition,}$$

$$\sum_{j=1}^{s} a_{sj} = 1, \qquad \text{consistency condition.}$$

□

In one sense A-stability is too strong a requirement, while in another sense A-stability is not sufficient since it does not ensure the rapid damping of errors, thus the need for L-stability. Linear stability theory plays a very important role in the selection of numerical methods suitable for solving stiff problems. However, as stated in Chapter 1, we cannot always rely on linear stability theory for the stability behaviour of nonlinear problems. Dahlquist proposed the study of of a nonlinear test equation for linear mutistep methods and this lead to the concept of *G-stability*. Butcher introduced an analogous property known as *B-stability* [14] for Runge-Kutta methods.

Definition 2.8 *Let* $\{\ldots, y_{n-1}, y_n, \ldots\}$ *and* $\{\ldots, z_{n-1}, z_n, \ldots\}$ *be two sequences of approximate solutions of* $y' = f(y(x))$ *calculated using a Runge-Kutta method with fixed stepsize* h*. Let* $< \cdot, \cdot >$ *denote an inner product in* $\mathbb{R}^m$ *and let* $\|\cdot\|$ *be the corresponding norm. The Runge-Kutta method is said to be B-stable if for any* f *satisfying*

$$\langle f(u) - f(v), u - v\rangle \leq 0, \qquad \textit{for all} \quad u, v \in R^m, \tag{2.14}$$

it follows that

$$\|y_n - z_n\| \leq \|y_{n-1} - z_{n-1}\|.$$

The relation (2.14) is called the *contractivity condition*. B-stability is a stronger condition than A-stability because of the requirement that the sequence $\|z_n - y_n\|$ be non-increasing rather than only bounded. B-stability is related to a condition called *algebraic stability* which was introduced by Burrage and Butcher [7] and also by Crouzeix [34]. An algebraic condition for B-stability is given by the following definition.

Definition 2.9 *If the coefficients of a Runge-Kutta method satisfy*

$$b_i \geq 0,$$

$$M = BA + A^T B - bb^T \qquad \textit{is non-negative definite},$$

where $B = diag(b_1, b_2, \ldots, b_s)$*, then the method is B-stable.*

Therefore, Runge-Kutta methods which satisfy the two conditions listed in the definition are said to be algebraically stable. Butcher and Burrage [7] extended the concept of B-stability to the non-autonomous problem $y' = f(x, y(x))$ and introduced *BN-stability*. It has been shown that algebraic stability is sufficient for B-stability as well as BN-stability. For *nonconfluent* Runge-Kutta methods the concepts of algebraic stability, BN-stability and B-stability are equivalent.

Definition 2.10 *A Runge-Kutta methods is said to be nonconfluent if all the components of the abscissae vector, c, are distinct.*

Since it is impossible for a linear multistep method to be of order greater than 2 and be A-stable, it seems that they would not be able to solve stiff problems satisfactorily. However, there is a sub-class of linear multistep methods called the *backward differentiation formulae* (BDF), which are considered to have stability regions large enough to be able to solve stiff problems. They are A(α)-stable for α close to 90^o, and are also stiffly stable.

2.3 BDF methods

The BDF methods were introduced by Curtiss and Hirschfelder [36]. The BDF method of k steps and order k can be written in the backward difference form

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = h f_{n+1}, \tag{2.15}$$

where the backward difference operator, ∇^j, is defined by

$$\nabla^j y_n = \nabla^{j-1} y_n - \nabla^{j-1} y_{n-1}, \quad \nabla^0 y_n = y_n.$$

These methods are A-stable for orders 1 and 2 and $A(\alpha)$-stable for orders $k = 3, 4, 5, 6$. For methods with $k \geq 7$ the methods do not satisfy the root condition and are thus not zero-stable [35]. We note that (2.15) is the implicit Euler method for $k = 1$.

Table 2.1: The angles α in $A(\alpha)$-stability of BDF methods.

order k	1	2	3	4	5	6
angle α	90°	90°	88°	73°	52°	18°

From Table 2.1 it can be seen that methods of orders 1 to 5 have stability regions which include most of the left half plane. In the stability regions parts of the left half plane are excluded and so higher order methods will be inefficient only for problems with eigenvalues close to the imaginary axis. In such cases the BDF methods of orders more than 2 are no better than the explicit methods. The method of order 6 has a very small value of α and consequently is not used in most software for solving initial value problems.

Some efficient implementations of the BDF methods, such as in LSODE [50] and VODE [4], have established them as one of the most efficient methods for solving stiff initial value problems on sequential computers. The success of the BDF methods is due to the following reasons.

1. They are computationally cheap as only one function evaluation per step is needed and usually the Newton iteration scheme has very fast convergence.

2. Asymtotically correct error estimates for the local truncation error are derived very easily. Since the local truncation error needs to be estimated and monitored in every step for controlling the step size, this is a very important consideration.

3. The ease of local truncation error estimation makes possible the implementation of the BDF methods in a variable stepsize, variable order mode. This makes them very efficient since it allows the code to choose the appropriate stepsize and method at every step.

4. These methods have been programmed for over 20 years and so there is a lot of computational experience with their implementation and use.

However, the BDF methods have the following disadvantages.

1. Only methods of orders 1 and 2 are A-stable. Hence, in a variable order implementation with say orders 1 to 5, it can be inefficient for solving problems with eigenvalues close to the imaginary axis.

2. Even though it is easy to estimate the local truncation errors, stepsize changing requires a complicated procedure.

3. The error constants of BDF methods are relatively larger than those of the implicit Runge-Kutta methods.

4. When solving problems with frequent discontinuities the BDF methods can be quite inefficient as this involves a restart when a discontinuity is located. Since they need to start with a first order method this can become computationally expensive.

2.4 Implicit Runge-Kutta methods

Since explicit Runge-Kutta methods have bounded regions of absolute stability they are not suited for the efficient solution of stiff problems. Some implicit methods on the other hand can be A-stable or L-stable and are well suited for

the solution of stiff problems as far as stability is concerned. The fully implicit methods are categorized by the class of quadrature formulae they revert to when $y' = f(x)$. From the theory of the order conditions of the Runge-Kutta methods, we have the simplifying assumptions [49]

$$B(p): \qquad \sum_{j=1}^{s} b_j c_j^{q-1} = \frac{1}{q}, \quad q = 1, 2, \ldots, p, \tag{2.16}$$

$$C(\eta): \qquad \sum_{j=1}^{s} a_{ij} c_j^{q-1} = \frac{1}{q} c_i^q, \quad q = 1, 2, \ldots, \eta, \tag{2.17}$$

$$D(\zeta): \qquad \sum_{i=1}^{s} b_i c_i^{q-1} a_{ij} = \frac{b_j}{q}(1 - c_j^q), \quad j = 1, 2, \ldots, s, \quad q = 1, 2, \ldots, \zeta, \tag{2.18}$$

which can be used in the derivation of the method coefficients. In the following sections we examine the *Gauss, Radau* and *Lobatto* families of methods. An important property of implicit Runge-Kutta methods which determines their suitability for solving stiff problems is the *stage order*.

Definition 2.11 *Consider a Runge-Kutta method of order p and q is the largest number such that the $C(q)$ condition*

$$\sum_{j=1}^{s} a_{ij} c_j^{q-1} = \frac{1}{q} c_i^q$$

holds. Then the stage order is defined as the minimum of (p, q) [49].

2.4.1 Gauss methods

These fully implicit methods are based on the *Gauss-Legendre* polynomials. These methods which were introduced by Butcher [11], have the highest possible order for a given number of stages. They revert to the Gaussian quadrature formulae with ordinates $x_n + c_j h$ and weights b_j, when solving $y' = f(x)$. The following theorem is due to Butcher [11].

Theorem 2.12 *The $s-$stage Gauss method has order $2s$. Its stability function is the (s,s)-Padé approximation and the method is A-stable.*

The abscissaes of an s stage Gauss method are the roots of the s degree Legendre polynomial, $P_s(c) = \frac{d^s}{dc^s}(c^s(c-1)^s)$. The A matrix is defined by the $C(s)$ condition and the b vector is defined by the $B(2s)$ condition. The $B(2s)$ and the $C(s)$ conditions give linear systems of equations which can be easily solved to derive the method coefficients. The one-stage Gauss method is the implicit Mid-point Rule.

2.4.2 Radau methods

The Radau methods which were originally proposed by Butcher [11] are not A-stable. Based on the ideas of Butcher, Ehle [41] constructed an A-stable family of methods called Radau IA methods. As s-stage Radau IA method has order $2s-1$. The abscissae of an s-stage Radau IA method are the roots of the degree s Radau left polynomial, $P_s(c) = \frac{d^{s-1}}{dc^{s-1}}(c^s(c-1)^{s-1})$. As a result of this $c_s = 0$ for all Radau IA methods. Matrix A is determined by the $D(s)$ condition and the b vector is determined by the $B(2s-1)$ condition. A second class of Radau methods, called Radau IIA methods, were developed by Axelsson [2]. As s-stage Radau IIA method has order $2s-1$. The abscissae are the roots of the degree s Radau right polynomial, $P_s(c) = \frac{d^{s-1}}{dc^{s-1}}(c^{s-1}(c-1)^s)$, as a result of which these methods have $c_s = 1$. The b vector is determined by the $B(2s-1)$ condition, while matrix A is determined by the $C(s)$ condition. The one-stage Radau IIA method is the implicit Euler method.

Theorem 2.13 *The s-stage Radau IA and IIA methods have order $2s-1$ and their stability function is the $(s-1,s)$ Padé approximation. These methods are A-stable and L-stable [49].*

2.4.3 Lobatto methods

An s-stage Lobatto method has order $2s-2$ and the abscissaes are the zeros of the Lobatto polynomial $P_s(c) = \frac{d^{s-2}}{dc^{s-2}}\left(c^{s-1}(c-1)^{s-1}\right)$. As a consequence $c_1 = 0$ and $c_s = 1$ for all Lobatto methods. There are various possibilities for choosing matrix A and the b vector. Among the most useful Lobatto methods are the Lobatto IIIA and IIIB methods of Ehle [41] and the Lobatto IIIC methods of Chipman [33].

For the three Lobatto methods the b vector is determined by the $B(2s-2)$ condition. The A matrix for the Lobatto IIIA methods is determined by the $C(s)$ condition while that for Lobatto IIIB methods is determined by the $D(s)$ condition. For Lobatto IIIC methods, $a_{i1} = b_1$ for $i = 1, 2, \ldots, s$, and the rest of matrix A is determined by the $C(s-1)$ condition. The 2−stage Lobatto IIIA method is the Trapezoidal Rule.

Theorem 2.14 *The s-stage Lobatto IIIA, IIIB and IIIC methods have order $2s-2$. The stability function for the Lobatto IIIA and IIIB methods is the $(s-1, s-1)$ Padé approximation, while that for the Lobatto IIIC method is the $(s-2, s)$ Padé approximations. All Lobatto III methods are A-stable and Lobatto IIIC methods are L-stable [49].*

In the study of stiffness, Prothero and Robinson [67] used problem (2.5) and showed that the global errors not only depended on the local errors but also on the errors contributed by the computations of the internal stage. If the method has stage order much less than the overall order, the global errors will, for moderate stepsizes, be dominated by the errors produced by the stages. Therefore, the accuracy of the solutions appeared to be related to the stage order and not the overall order of the method. They termed this phenomenon order reduction. They further showed that methods which are stiffly-accurate do not suffer from order reduction due to the cancellation of errors produced in the calculation of the internal stages as $|z| \to \infty$. We have seen earlier that A-stable stiffly-accurate methods are just L-stable methods and that they satisfy, $a_{sj} = b_j$, $j = 1, 2, \ldots, s$. The Gauss methods, which have stage order of only s compared to their overall order of $2s$, and are not L-stable suffer from order reduction. Hence, inspite of the

fact that the Radau and Lobatto methods have lower overall order for the same number of internal stages, compared to the Gauss methods, Theorems 2.13 and 2.14 and the study of Prothero and Robinson show that the Radau and Lobatto methods have better stability and accuracy properties amongst the implicit Runge-Kutta methods. The well known code, RADAU5 [49], is based on the 3−stage fifth order Radau IIA method

$$\begin{array}{c|ccc} \frac{4-6\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+6\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array} \tag{2.19}$$

The variable order RADAU code consists of Radau IIA methods of three, five and seven stages of orders five, nine and thirteen respectively. The code PSIDE [72] is a parallel implementation of the 4-stage Radau IIA method of order seven. RADAU5 and PSIDE are used to solve some stiff problems later in this thesis.

2.4.4 Solution of the implicit stage equations

In this section we look at the solution of the implicit stage equations that arise in the use of the implicit Runge-Kutta methods when solving the initial value problem (1.1). The general form of an s-stage implicit Runge-Kutta method is

$$y_{n+1} \;=\; y_n + h\sum_{i=1}^{n} b_i f(x_n + hc_i, Y_i),$$

where the stages Y_i are determined by solving the set of implicit equations

$$Y_i \;=\; y_n + h\sum_{j=1}^{n} a_{ij} f(x_n + hc_j, Y_j), \qquad i = 1, 2, \ldots, s.$$

The stage equations can be written as

$$Y = e \otimes y_n + h(A \otimes I_m)F(Y), \tag{2.20}$$

where m is the dimension of the problem, and

$$e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \qquad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \qquad F(Y) = \begin{bmatrix} f(x_n + c_1 h, Y_1) \\ f(x_n + c_2 h, Y_2) \\ \vdots \\ f(x_n + c_s, Y_s) \end{bmatrix}.$$

Since we are interested in solving stiff systems, fixed point iteration is not appropriate. So we use a modified Newton Raphson iteration to solve (2.20). This can be defined as

$$\begin{aligned} M\Delta Y^{[k]} &= G(Y^{[k]}), \qquad (2.21) \\ Y^{[k+1]} &= Y^{[k]} + \Delta Y^{[k]} \qquad k = 0, 1, \ldots, \end{aligned}$$

where

$$\begin{aligned} M &= I_s \otimes I_m - h(A \otimes J), \qquad (2.22) \\ G(Y^{[k]}) &= -Y^{[k]} + e \otimes y_n + h(A \otimes I_m)F(Y^{[k]}), \qquad (2.23) \end{aligned}$$

and J is the Jacobian of f evaluated at some recent point. The total cost of computations in this scheme include

1. the evaluation of F and G,
2. the evaluation of J,
3. the evaluation of M,
4. LU factorization of the iteration matrix, M, (or some variant of this),
5. back substitution to get the Newton update vector, $\Delta Y^{[k]}$.

In any implementation of these methods the costs associated with the items 2 to 4 can be reduced by keeping the Jacobian and the iteration matrix constant over several steps. The LU factorization of the the $ms \times ms$ iteration matrix, M, has an operation count of $s^3m^3/3$ and the cost of backsubstitution is s^2m^2. Hence, the total linear algebra cost is about $s^3m^3/3 + s^2m^2$. As the number of stages, s, of the method, or size of the system, m, increases, the computational cost increases considerably. Compared to the BDF cost, $m^3/3+m^2$, this is considered excessively

high. This is is the main reason why the implicit Runge-Kutta methods are not competitive with the BDF methods.

The main advantages of using implicit Runge-Kutta methods for solving stiff initial value problems are

1. They can be A-stable or L-stable for high orders.
2. They are highly accurate since they have small error constants.
3. They are one-step methods so restarting can be done at any order when there is a need, as in problems with discontinuities.
4. Stepsize can be changed easily if an estimate of the local truncation error exists.

Their main disadvantages are

1. They have very high linear algebra operation costs and also have high function evaluation costs.
2. It is difficult to derive schemes for estimating local truncation errors.
3. It is difficult to derive schemes for estimating errors for the next higher order method. Consequently, automatic order control is very difficult.

Fortunately some costs associated with the linear algebra can be reduced by imposing certain structures on the Runge-Kutta matrix. Butcher [15] proposed an ingenious technique for reducing the computational costs in linear algebra and is based on similarity transformations. We investigate the cost reduction techniques in the following sections.

2.4.5 Diagonally implicit Runge-Kutta methods

As we have seen in the last section, fully implicit Runge-Kutta methods are computationally expensive, as at every stage we need to factorise a matrix of size $ms \times ms$. By requiring that matrix A be lower triangular, we have methods which are called *diagonally implicit* Runge-Kutta (DIRK) methods, for which computational costs are lower. DIRK methods which are A-stable or L-stable can be derived for a range of orders. If the diagonal elements of matrix A are all equal

then to emphasize this property the methods are called *singly diagonally implicit* (SDIRK) methods. Consider a DIRK method of form

$$\begin{array}{c|cccc} c_1 & \mu_1 & & & \\ c_2 & a_{21} & \mu_2 & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & \mu_s \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{2.24}$$

The method is defined as

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{i=1}^{n} b_i f(x_n + hc_i, Y_i), \\ Y_i &= y_n + h\sum_{j=1}^{i-1} a_{ij} f(x_n + hc_j, Y_j) + \mu_i f(x_n + hc_i, Y_i), \qquad i = 1, 2, \ldots, s. \end{aligned} \tag{2.25}$$

It can be seen that in contrast to the fully implicit methods, each stage, Y_i, depends on the previous stages and itself only. Since each stage is uncoupled it can be solved individually, avoiding the need to factorise a large matrix. The Newton iteration scheme for solving (2.25) is defined as

$$\begin{aligned} (I_m - h\mu J)\Delta Y_i^{[k]} &= -Y_i^{[k]} + y_n + \sum_{j=1}^{i-1} a_{ij} f(x_n + hc_j, Y_j) + \mu_i f(x_n + hc_i, Y_i^{[k]}) \\ Y_i^{[k+1]} &= Y_i^{[k]} + \Delta Y_i^{[k]}, \quad k = 0, 1, \ldots, \quad i = 1, 2, \ldots, s, \end{aligned}$$

where J is the Jacobian of the system evaluated at some recent point. The matrix $I_m - h\mu_i J$, which is of size $m \times m$ needs to be factorised at every integration step and thus the total linear algebra operation count for an $s-$stage DIRK method is about $sm^3/3 + sm^2$. This is considerably less compared to the fully implicit methods and is quite comparable to the BDF methods. For the case where the diagonal elements of the A matrix are all equal to μ, the matrix factors of a single matrix, $I_m - h\mu J$, can be used for all the stages. Thus, for the SDIRK case, the cost is further reduced to $m^3/3 + sm^2$. The code SIMPLE is an implementation of a three stage, third order SDIRK method by Nørsett and Thomson [65].

However, the DIRK methods suffer from a low stage order. Since the DIRK methods satisfy only the $C(1)$ condition, they have a stage order of only one. It has been observed by Prothero and Robinson [67] that when a Runge-Kutta method is used to solve a very stiff problem the observed order is less than the order of the method and can be as low as the stage order. That is, the DIRK methods suffer from order reduction. Therefore, it seems that it is very important for methods for use in solving stiff problems to have the order and stage order as close as possible, preferably equal. Consequently, the DIRK methods are not suited for the solution of stiff initial value problems.

2.4.6 Use of transformations to reduce costs

In order to reduce the computational cost of the fully implicit Runge-Kutta methods, Butcher [15] introduced similarity transformations. In this approach we use transformation of the iterates, in such a way that the iteration matrix, $M = I_s \otimes I_m - h(A \otimes J)$, transforms to a lower triangular form which can be exploited to reduce the computational costs. Consider the modified Newton iteration scheme (2.21) and the transformations from $Y^{[k]}, G(Y^{[k]})$ to $\widehat{Y}^{[k]}, \widehat{G}(Y^{[k]})$ given by

$$\widehat{Y}^{[k]} = (Q^{-1} \otimes I_m)Y^{[k]}, \tag{2.26}$$

$$\widehat{G}(Y^{[k]}) = (P \otimes I_m)G(Y^{[k]}), \tag{2.27}$$

where P and Q are two nonsingular $s \times s$ matrices such that

$$Q^{-1}A^{-1}Q = \begin{bmatrix} 1/\lambda_1 & 0 & 0 & \cdots & 0 \\ \mu_1 & 1/\lambda_2 & 0 & \cdots & 0 \\ 0 & \mu_2 & 1/\lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1/\lambda_s \end{bmatrix}, \tag{2.28}$$

is the Jordan canonical form of A^{-1}. Furthermore, μ_i, $i = 1, 2, \ldots, s-1$, is zero if $\lambda_i \neq \lambda_{i+1}$ and either zero or an arbitrary non-zero number if $\lambda_i = \lambda_{i+1}$, and $PAQ = D = diag(\lambda_1, \lambda_2, \ldots, \lambda_s)$, where λ_i, $i = 1, 2, \ldots, s$, are the eigenvalues of

matrix A. This means that

$$\begin{aligned} PQ &= (DQ^{-1}A^{-1})Q = D(Q^{-1}A^{-1}Q) \\ &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \epsilon_1 & 1 & 0 & \cdots & 0 \\ 0 & \epsilon_2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \end{aligned}$$

where each of the subdiagonal elements, $\epsilon_1, \epsilon_2, \ldots$, is either 0 or 1. Then (2.21) is transformed to

$$\widehat{M}\Delta\widehat{Y}^{[k]} = \widehat{G}(Y^{[k]}), \tag{2.29}$$

where

$$\begin{aligned} \widehat{M} &= (P \otimes I_m)M(Q \otimes I_m) \\ &= (PQ) \otimes I_m - (PAQ) \otimes J. \end{aligned}$$

The matrix $\widehat{M}$ now consists of diagonal blocks of the form $I_m - h\lambda J$, together with subdiagonals blocks of either the zero or the identity matrix. Hence, we now require the LU factorisations of each of the diagonal blocks and the back substitutions break into s separate blocks with the subdiagonal elements of PQ contributing a further O(n) operations. The total linear algebra cost depends on whether the eigenvalues of the A matrix are real or complex. Where all the eigenvalues are real and distinct, the total cost reduces to $sm^3/3 + sm^2$, a considerable saving compared to the original number of operations. Where the eigenvalues are complex smaller savings are realised. The cost of transformations is a further $2s^2m$ operations. As well as transforming the matrices related to the method, it is also possible to transform a full Jacobian, J, to upper Hessenberg form as proposed by Enright [44] and this can be especially beneficial if the same J is used over as many steps as possible.

2.4.7 Singly implicit methods

From the last section it is evident that for sequential computation, when matrix A has a single eigenvalue, the linear algebra cost is reduced to about $m^3/3 + sm^2$, which is the same as the cost in the SDIRK method and is very close to the BDF cost. The methods where the A matrix has a single eigenvalue are called the *singly implicit Runge-Kutta* (SIRK) methods. We briefly summarise some of the properties of these methods. More details can be found in [16].

Let $\xi_1, \xi_2, \ldots, \xi_s$ be the distinct zeros of the degree s Laguerre polynomial, then the transformation matrix T for the SIRK method is chosen to be

$$T = \begin{bmatrix} L_0(\xi_1) & L_1(\xi_1) & \cdots & L_{s-1}(\xi_1) \\ L_0(\xi_2) & L_1(\xi_2) & \cdots & L_{s-1}(\xi_2) \\ \vdots & \vdots & & \vdots \\ L_0(\xi_s) & L_1(\xi_s) & \cdots & L_{s-1}(\xi_s) \end{bmatrix},$$

and the matrix T has the property

$$T^{-1}AT = \begin{bmatrix} \lambda & 0 & 0 & \cdots & 0 \\ -\lambda & \lambda & 0 & \cdots & 0 \\ 0 & -\lambda & \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda \end{bmatrix},$$

where λ is the single s-fold eigenvalue of the Runge-Kutta matrix A. The matrix T^{-1} can be explicitly defined in terms of the roots of the Laguerre polynomials [16]. For a method with s-stages, the value of λ will be chosen by requiring that the characteristic polynomial of the A matrix be $(z - \lambda)^s$. Having chosen λ, we select $c_1 = \lambda\xi_1, c_2 = \lambda\xi_2, \ldots, c_s = \lambda\xi_s$. Then matrix A is chosen to satisfy the $C(s)$ condition (2.17) and $b_1, b_2, \ldots, b_s$ are chosen to satisfy the $B(s)$ condition (2.16).

Definition 2.15 *The Laguerre polynomial of degree s is defined as*

$$L_n(x) = \sum_{i=0}^{n} \binom{n}{i} \frac{(-x)^i}{i!}, \tag{2.30}$$

The following theorem on the order of SIRKs is due to Burrage [6].

Theorem 2.16 *An s-stage singly implicit method has order s or s + 1 and it is either L-stable or A-stable respectively.*

Since the s-stage method of order s satisfy the stage order condition $C(s)$, they have a stage order equal to s, therefore, order reduction is not expected when these methods are used to solve stiff problems. The following is an example of a second order, L-stable SIRK method.

$$\begin{array}{c|cc} 3-2\sqrt{2} & \frac{5-3\sqrt{2}}{4} & \frac{7-5\sqrt{2}}{4} \\ 1 & \frac{1+\sqrt{2}}{4} & \frac{3-\sqrt{2}}{4} \\ \hline & \frac{1+\sqrt{2}}{4} & \frac{3-\sqrt{2}}{4} \end{array}$$

The SIRK methods have been implemented in a variable stepsize, variable order code by Burrage, Butcher and Chipman [9]. Although the SIRK methods have much lower computational costs compared to the fully implicit Runge-Kutta methods, the cost of transformations remain high for lower dimensional problems. The number of function evaluations also remain quite high. Another disadvantage that results from the choice of λ is that the abscissae values lie far outside the integration interval for methods of orders more than three. To overcome some of the problems of SIRK methods, a recent generalisation is the *diagonally extended singly implicit* (DESI) Runge-Kutta methods of Butcher, Cash and Diamantikis [25], in which the SIRK methods are appended with a few extra diagonally implicit stages. These additional stages provide extra freedom in the choice of method parameters. An implementation of these methods [40] has shown them to be competitive with the BDF methods for the stiff DETEST set of problems. Another generalisation is the *effective order singly implicit* (ESIRK) methods of Butcher and Chartier [28], in which the initial values are perturbed using effective order. For these methods the extra freedom allows a free choice of abscissae. Generalising effective order to DESI methods leads to *diagonally extended singly implicit Runge-Kutta methods with effective order* (DESIRE) methods of Butcher and Diamantakis [29]. Although these methods are promising to be good competitors to the BDF methods the internal stages in these methods need to be calculated sequentially and cannot take advantage of parallel processors.

Since the calculation of the implicit stages constitutes the most computationally

expensive part, an obvious way of reducing computational costs is to use parallel processors to calculate the stages. This means that we need to develop methods in which the stages can be computed in parallel or to develop procedures in which some of the existing methods can be implemented in parallel.

A family of Runge-Kutta methods have been developed to take advantage of the use of parallel processors in the past decade and in the following sections we outline some of these methods.

2.5 Some parallel numerical methods for stiff problems

In this section we examine a selection of the existing parallel numerical methods for the solution of initial value problems. Some of these method are modified versions of the methods as they were used on sequential computers, while others are new methods especially designed for parallel implementation.

2.5.1 Motivation for parallel methods

Traditional methods used for the numerical solution of stiff ordinary differential equations are either linear multistep methods or Runge-Kutta methods. The advantages and disadvantages of of these methods have been outlined in the previous sections.

Among the main drawbacks for linear multistep methods is the their lack of A-stability for orders greater than two. Compared to linear multistep methods, the sequential costs of using implicit Runge-Kutta methods for solving stiff problems is a lot higher. Although transformations have been used to reduce the computational costs, these methods are still not competitive in a sequential computing environment. To reduce the costs associated with solving large problems one can look beyond serial computation, as computers with parallel computing capabilities become widespread.

Very few existing methods can be adapted to take advantage of parallelism in the method. There is almost no prospect of parallelism in the use of the multistep methods. The explicit Runge-Kutta methods require the stages to be calculated successively, while the fully implicit methods require the stages to be calculated by solving a large non-linear system and obviously cannot take advantage of parallelism without modifications or by the development of new methods in which some of the stages can be evaluated in parallel. We examine some of these methods in the following sections.

2.5.2 Parallel block methods

These methods were introduced by Sommeijer et. al. [71] and further generalised by Chartier [32]. These methods are a direct generalisation of the implicit one-step method

$$y_{n+1} = ay_n + hbf(y_n) + hdf(y_{n+1}).$$

Define $c = [c_1, c_2, \ldots, c_k]^T$ with $c_1 = 1$ and assume that c_i are pairwise distinct, and let $y_{n,i}$ denote the numerical approximation to the solution $y(x_n + (c_i - 1)h)$ at step n. Let Y_n denote the vector with components $y_{n,i}$, $i = 1, 2, \ldots, k$. The final approximation to the solution is given by $y_{n+1,1}$, which is the first component of Y_{n+1}. If I denotes the $k \times k$ identity matrix, then a k-dimensional parallel block method is defined by the recursion

$$Y_{n+1} = (A \otimes I)Y_n + h(B \otimes I)F(Y_n) + h(D \otimes I)F(Y_{n+1}), \tag{2.31}$$

where A, B and D are $k \times k$ real matrices and D is assumed to be diagonal. The diagonal structure of D decouples the the implicit system, thus allowing the stages to be solved in parallel. This method can be formulated as a general linear method

$$\begin{bmatrix} Y \\ y^{[n+1]} \end{bmatrix} = \begin{bmatrix} \mathcal{A} \otimes I & \mathcal{U} \otimes I \\ \mathcal{B} \otimes I & \mathcal{V} \otimes I \end{bmatrix} \begin{bmatrix} hF(Y) \\ y^{[n]} \end{bmatrix},$$

where

$$Y = Y_n, \quad \mathcal{A} = D, \quad \mathcal{U} = \left[\begin{array}{cc} A & B \end{array} \right],$$

$$y^{[n]} = \left[\begin{array}{c} Y_n \\ hF(Y_n) \end{array} \right], \quad \mathcal{B} = \left[\begin{array}{c} D \\ I \end{array} \right], \quad \mathcal{V} = \left[\begin{array}{cc} A & B \\ 0 & 0 \end{array} \right].$$

A special choice of method parameters is considered in [32]. Methods with

- $B = 0$, to ensure that $\rho(M(\infty)) = 0$,
- $c = [1, 2, \ldots, k]^T$,
- $A = VU^{-1} - DWU^{-1}$, where

$$U = [e, (c-e), (c-e)^2, \ldots, (c-e)^{k-1}],$$

$$V = [e, c, c^2, \ldots, c^{k-1}],$$

$$W = [0, e, 2c, \ldots, (k-1)c^{k-1}],$$

- $D = (1/r)(I + \mathrm{diag}(c))$, where $\mathrm{diag}(c)$ is the $k \times k$ diagonal matrix with diagonal elements c_i, and $r > 0$,

are called $\mathcal{M}(k, r)$. The details concerning this choice of parameters can be found in [32]. It has been shown that when $r \geq \frac{k-1}{2}$, the method $\mathcal{M}(k, r)$ for some choice of r is of order at least $k - 1$, zero-stable and L-stable. Because of the choices $B = 0$ and $c = [1, 2, \ldots, k]^T$, $\mathcal{M}(k, r)$ can be considered as a generalisation of the BDF methods.

Example 2.5 Method $\mathcal{M}(4, 5)$

$$A = \left[\begin{array}{cccc} \frac{2}{15} & \frac{6}{5} & -\frac{2}{5} & \frac{1}{15} \\ -\frac{1}{10} & \frac{3}{5} & \frac{7}{10} & -\frac{1}{5} \\ \frac{4}{15} & -\frac{6}{5} & \frac{12}{5} & -\frac{7}{15} \\ \frac{5}{6} & -3 & \frac{7}{2} & -\frac{1}{3} \end{array} \right]$$

□

An implementation of $\mathcal{M}(8,r)$ where $r \approx 7.25$, showed that on a selected set of stiff problems of dimensions 2 to 15, it is sometimes more efficient compared to RADAU5 and LSODE, if the parallel CPU times were 1/8 the sequential CPU times. Since only problems of small dimensions were used this may not be justified.

2.5.3 Parallel Runge-Kutta methods

In order to solve stiff problems we need to use implicit methods. Fully implicit Runge-Kutta methods require the solution of a large system of equations in general. In order to be able to solve some stages in parallel, we need to have methods in which some of the stages can be solved independently. We look at some of the methods in which we can have this type of implementation. Later we examine some other types of of implementations which can take advantage of parallelism.

2.5.3.1 Strictly block diagonal implicit Runge-Kutta methods

These methods are among the ones that have been specifically designed to take advantage of parallelism. The A matrix of such methods consist of diagonal blocks. An example of such a method that has been derived by Iserles and Nørsett [58] is

$$\begin{array}{c|cccc}
\frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{5}{12} & \frac{1}{12}-\frac{\sqrt{3}}{6} & 0 & 0 \\
\frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{12}+\frac{\sqrt{3}}{6} & \frac{5}{12} & 0 & 0 \\
\frac{1}{2}-\frac{\sqrt{3}}{6} & 0 & 0 & \frac{1}{2} & -\frac{\sqrt{3}}{6} \\
\frac{1}{2}+\frac{\sqrt{3}}{6} & 0 & 0 & \frac{\sqrt{3}}{6} & \frac{1}{2} \\
\hline
 & \frac{3}{2} & \frac{3}{2} & -1 & -1
\end{array}$$

This method is L-stable but not algebraically stable. In this method the 2×2 blocks can be solved independently and so the blocks can be solved entirely in parallel. This method belongs to a family of methods in which the diagonal consists of 2×2 blocks. It has been shown in [58] that, independent of the number of blocks, the maximal attainable order is 4. Thus, the advantage one can gain from exploiting parallelism is limited in this family of methods.

2.5.3.2 Parallel DIRK methods

By deriving DIRK methods in which the A matrix has a diagonal block structure, the implicit stages in each block can be be solved in parallel. Such methods are called *parallel DIRK* methods. Methods of this form have been derived by Iserles and Nørsett [58]. Such a fourth order, 4 stage method is given by

$$\begin{array}{c|cccc} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{2}{3} & 0 & \frac{2}{3} & 0 & 0 \\ \frac{1}{2} & -\frac{5}{2} & \frac{5}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & -\frac{5}{3} & \frac{4}{3} & 0 & \frac{2}{3} \\ \hline & -1 & \frac{3}{2} & -1 & \frac{3}{2} \end{array}$$

This method is L-stable but not algebraically-stable. Since the first two stages can be solved independently, they can be solved in parallel, and once these two stage values have been calculated, the third and the fourth stages can similarly be solved in parallel on two processors. Hence, this method can be implemented in parallel on two processors. Moreover, since $a_{11} = a_{33}$ and $a_{22} = a_{44}$, only two matrices need to be factorised to solve for all the stages by the Newton iteration method.

In general for parallel implementation, matrix A of the parallel DIRK methods can be written in block form as

$$A = \begin{bmatrix} D_1 & 0 & 0 & \cdots & 0 \\ A_{21} & D_2 & 0 & \cdots & 0 \\ A_{31} & A_{32} & D_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ A_{p1} & A_{p2} & A_{p3} & \cdots & D_p \end{bmatrix}$$

where each D_k is a (possibly different) diagonal matrix. The stages corresponding D_1 can be solved in parallel and the results are used to solve for the stages corresponding to D_2 in parallel, and so on.

As the order of the methods increase, more possibilities arise for the diagonal block structure of the Runge-Kutta matrix. In particular, the location of the

zero and non-zero terms in the lower triangular part of the Runge-Kutta matrix determine the number of processors that can be used to solve the stages and also the relationships between stages. This structure has been analysed to reveal the capacity of parallelism using directed graphs [58].

As discussed earlier the DIRK methods suffer from low stage order and so are not likely to form the basis for parallel IVP software for stiff problems.

2.5.3.3 PIRK methods

In order to solve (2.20) van der Houwen and Sommeijer [52] proposed the scheme

$$Y^{[0]} - h(B \otimes I_m)F(Y^{[0]}) = e \otimes y_n + h(C \otimes I_m)F(Y^{[0]}), \tag{2.32}$$

$$Y^{[j+1]} - h(D \otimes I_m)F(Y^{[j+1]}) = e \otimes y_n + h\left((A - D) \otimes I_m\right)F(Y^{[j]}), \tag{2.33}$$

for $j = 0, 1, \ldots$, where B and D are assumed to be diagonal matrices, while C is an arbitrary full matrix. This form makes the iterative process (2.33) parallel, since now, given $Y^{[j]}$, $Y^{[j+1]}$ can be computed using s processors, each one dealing with one-stage. Explicit and implicit schemes based on the above formulation have been derived by van der Houwen and Sommeijer in [52, 51, 54] and we examine the main results of their work. In the following sections, the term predictor is used to refer to the method used for obtaining values of $Y^{[0]}$, while corrector refers to the method used to calculate $Y^{[i]}$, $i = 1, 2, \ldots$. In this context (2.32) is the predictor and (2.33) the corrector.

By setting $B = 0$ and $D = 0$, the resulting explicit methods are called *parallel iterated Runge-Kutta* (PIRK) methods.

2.5.3.4 PDIRK methods

van der Houwen et. al. [54] considered parallel methods by considering a non-zero choice of the matrix D in (2.33). Then the method (2.33) becomes an implicit method, suitable for the solution of stiff problems. When D is chosen as a diagonal matrix, the resulting methods are called *parallel diagonally iterated Runge-Kutta*

or PDIRK methods. Since matrix D is of diagonal form, the s components of the stage vector, Y, can now be computed in parallel on s processors, using some form of Newton iterations.

Different ways of choosing matrix D and the number of iterations, M, has been considered. Keeping the number of iterations fixed such that the orders of the corrector and PDIRK are equal and by choosing the other iteration parameters such that the stability regions in the left half plane is optimized, leads to a class of DIRK methods. Using this approach and $D = dI$, van der Houwen et. al. [54], have derived several A-stable and L-stable methods of orders up to 10. They have shown that these methods have better performance on a parallel machine than the basically sequential codes, LSODE and SIMPLE, for two stiff problems.

In the second approach the number of iterations, M, is allowed to vary and the diagonal matrix D is chosen to minimize the spectral radius of the stability matrix at infinity, $Z(\infty) = I_m - D^{-1}A$. This ensures strong damping of the stiff components of the iteration error. The performance of the methods derived in this way depended on the type of methods used as the the predictor and corrector. For example, it has been reported that the Gauss correctors when used with explicit predictors did not perform satisfactorily. Some of the methods derived in this way had better performance than the existing DIRK methods provided appropriate correctors were chosen. One problem of the PDIRK methods is the poor convergence and even divergence in the first few iterations which becomes worse as the number of stages of the underlying Runge-Kutta method increases [56].

2.5.3.5 Triangularly implicit iteration methods

These methods have been proposed by van der Houwen and de Swart [56] in order to improve the convergence of the PDIRK methods. The A matrix in (2.21) is replaced by a more 'convenient' matrix B, which is required to be lower triangular, that is $B = L + D$, and L is strictly lower triangular while D is a diagonal matrix

with positive diagonal entries, d_{ii}. This leads to the iteration scheme

$$\begin{aligned}(I - D \otimes hJ)\Delta Y^{[j]} &= (L \otimes hJ)\Delta Y^{[j]} - G(Y^{[j]}), \qquad (2.34)\\ Y^{[j+1]} &= Y^{[j]} + \Delta Y^{[j]}, \qquad j = 0, 1, \ldots,\end{aligned}$$

where $I = I_s \otimes I_m = I_{ms}$. In the case where $L = 0$ and the underlying method is a Runge-Kutta method, the resulting scheme is the PDIRK method discussed earlier. The method (2.34) requires LU decompositions of $m \times m$ matrices, $I_m - hd_{ii}J$, $i = 1, \ldots, s$, and in each iteration the evaluation $G(Y^{[j]})$, s forward-backward substitutions and the matrix-vector multiplication $(L \otimes hJ)\Delta Y^{[j]}$. Since J is an approximate Jacobian, this multiplication can be expressed in terms of F, and this leads to the scheme

$$\begin{aligned}(I - D \otimes hJ)\Delta Y^{[j]} &= h(L \otimes I_m)\left(F(Y^{[j+1]}) - F(Y^{[j]})\right) - G(Y^{[j]}), \qquad (2.35)\\ Y^{[j+1]} &= Y^{[j]} + \Delta Y^{[j]}, \qquad j = 0, 1, \ldots\end{aligned}$$

In the schemes (2.34) and (2.35) the s LU decompositions and the evaluation of $G(Y^{[j]})$ can be done in parallel. These schemes are called parallel, triangularly implicit iterated methods and where the underlying method is a Runge-Kutta method then they are called *parallel triangularly implicit Runge-Kutta* (PTIRK) methods. A further degree of parallelism is introduced by using the Butcher similarity transformation (2.26), which enables the elimination of the $(L \otimes hJ)\Delta Y^{[j]}$ term in (2.34). This leads to the scheme

$$\begin{aligned}(I - D \otimes hJ)\Delta \widehat{Y}^{[j]} &= -(Q^{-1} \otimes I_m)G(Y^{[j]}), \qquad (2.36)\\ Y^{[j+1]} &= Y^{[j]} + (Q \otimes I_m)\Delta \widehat{Y}^{[j]},\end{aligned}$$

where $BQ = QD$. In each iteration of this scheme, the s forward-backward substitution can now be done in parallel, in addition to the other components that can be done in parallel. Apart from the case where $L = 0$, the scheme (2.36), when implemented in parallel on s processors, has the lowest cost among the schemes (2.34)-(2.36) [57].

In search of suitable B matrices Hoffmann and de Swart [57] applied the iteration scheme to the test problem $y' = qy$ and defined the amplification matrix (stability

matrix) as

$$Z(z) = z(I_m - zB)^{-1}(A - B), \tag{2.37}$$

where $z = qh$, and the stiff and non-stiff amplification matrices respectively, as

$$Z_\infty(B) = \lim_{|z|\to\infty} Z(z) = I_m - B^{-1}A, \tag{2.38}$$

$$Z_0(B) = \lim_{|z|\to\infty} \frac{Z(z)}{|z|} = A - B. \tag{2.39}$$

One choice that has been considered in [57] is for PTIRK methods with $B = T$ such that, $A = TU$, is the Crout decomposition of A. This means that U is unit upper triangular. It follows that, the stiff amplification matrix $Z_\infty(T)$ is strictly upper triangular and that $(Z_\infty(T))^j = 0$ for $j > s$. Furthermore, it has been proved in [57] that the diagonal entries of T chosen using this criteria are always positive.

Using the four stage Radau IIA corrector in a constant stepsize setting with a Jacobian evaluation in every step, it is shown in [56], that the PTIRK methods have faster convergence in the first few iterations than the PDIRK methods, on a few representative stiff problems.

2.5.3.6 PILSRK methods

In these methods the modified Newton iteration scheme (2.21) is solved by an inner linear iteration in which the matrices to be factorised are of the form $I - B \otimes hJ$ where B is similar to a diagonal matrix with positive diagonal entries. This inner iteration process is referred to as *parallel iterative linear system solver for Runge-Kutta* (PILSRK) methods and is defined by

$$(I - B \otimes hJ)(Y^{[j+1,\mu+1]} - Y^{[j+1,\mu]}) = -(I - A \otimes hJ)Y^{[j+1,\mu]} + C^{[j]}, \tag{2.40}$$

$$C^{[j]} = (I - A \otimes hJ)Y^{[j]} - G(Y^{[j]},$$

where $Y^{[j+1,0]} = Y^{[j,r]}$ and $Y^{[n,r]}$ is accepted as the solution Y of (2.21). Since $C^{[j]}$ does not depend on μ, the inner iteration requires only one evaluation of C. Using

this in conjunction with the Newton iteration scheme as above, is termed the Newton-PILSRK iteration scheme. The matrix B is assumed to be nondefective and to have positive eigenvalues, and so satisfies $B = QTQ^{-1}$, with Q an arbitrary real, nonsingular matrix and T a lower triangular matrix with positive diagonal entries. Hence, using the transformation $Y^{[j,\mu]} = (Q \otimes I_m)\widehat{Y}^{[j,\mu]}$ we obtain

$$(I - T \otimes hJ)(\widehat{Y}^{[j+1,\mu+1]} - \widehat{Y}^{[j+1,\mu]}) =$$

$$-(I - Q^{-1}AQ \otimes hJ)\widehat{Y}^{[j+1,\mu-1]} + (Q^{-1} \otimes I_m)C^{[j]}, \tag{2.41}$$

for $\mu = 1, 2, \ldots, r$ and $\widehat{Y}^{[j+1,0]} = (Q^{-1} \otimes I_m)Y^{[j]}$. If for a given $j+1$ the transformed inner iterates, $\widehat{Y}^{[j+1,\mu+1]}$, converge to a vector $\widehat{Y}^{[j+1,\infty]}$, the Newton iterate defined by (2.21) can be obtained from $Y^{[j+1]} = (Q \otimes I_m)\widehat{Y}^{[j+1,\infty]}$. Given the matrix A, the PILSIRK method (2.41) is completely defined by the matrix B or the matrix pair (T, Q). The derivation of suitable matrix pairs (T, Q) is discussed in [55]. The code PSIDE [72] is an implementation of the four stage Radau IIA method in which the stages are computed in parallel using the PILSRK iteration scheme.

2.5.4 DIMSEMs

A set of general linear methods called *diagonally implicit single eigenvalue methods* (DIMSEMs) has been derived by Enenkel and Jackson [42, 43]. As the name implies the stability matrix of DIMSEMs has a single non-zero eigenvalue and all other eigenvalues are equal to zero. The method is of the form given by (1.9), where A is a diagonal matrix, to enable a parallel implementation for stiff problems. It has been shown that if $A = \lambda I$, the maximum order attainable is 2. As a consequence the more general diagonally implicit methods in which $A = \text{diag}[\lambda_1, \lambda_2, \ldots, \lambda_r]$, has been considered. Even with this form it has been shown that it is not possible to derive A-stable methods in which $r = s$. Hence, the methods considered satisfy $s = r - 1$ and have order and stage order equal to $r - 1$.

By using the simplifying assumptions for general linear methods, the stability matrix, $M(z) = V + zB(I - zA)^{-1}U$, is transformed to a form in which the U and V matrices have been eliminated. This transformed stability matrix is then used

to satisfy the $r-1$ zero eigenvalue conditions. This transformation left c_i and λ_i as free parameters. However, it seems there were too many parameters which could not be handled in any reasonable way, so the authors decided an unusual approach. They chose $c_i = 0$, $i = 1, 2, \ldots, s$. L-stable DIMSEMs of orders 2 to 6 have been derived using a symbolic manupulator. More details on the derivations of these methods can be found in [42].

A fixed order implementation of DIMSEMs with starting values using LSODE, performed better at more stringent tolerances than LSODE on a test problem of dimension 100.

From this brief review it is evident that it is difficult to adapt the existing numerical methods for parallel computation and that very few new numerical methods which can take advantage of parallel processors have been developed.

Chapter 3

DIMSIMs

Although general linear methods were proposed about 30 years ago, they have never been widely adopted as practical numerical methods. The main difficulty has been identifying practical methods. From this large class, a sublcass which has potential for efficient implementations, has been identified by Butcher [19] as the *diagonally implicit multistage integration* (DIMSIMs). It is hoped that these methods will have many of the advantages possessed by Runge-Kutta methods and linear multistep methods but without their disadvantages. What makes DIMSIMs more attractive is their potential for parallel methods.

In this chapter we analyse DIMSIMs in general and type 4 DIMSIMs in which the order, the stage order, and the number of internal and external stages are all equal.

Since we are investigating parallelism across methods and this applies to scalar problems, for notational convenience we will use the scalar autonomous initial value problem in the rest of this thesis, unless otherwise stated. However, the formulae generalise easily using Kronecker products, to systems of ordinary differential equations and therefore to nonautonomous equations.

3.1 Introduction to DIMSIMs

DIMSIMs are a subclass of general linear methods and are characterised by four integer parameters, p, q, r and s. The values of p and q are the order and the stage order respectively, whereas r gives the number of approximations passed from one step to the next, and s is the number of internal stages. The methods can be represented by the equations

$$Y = AhF(Y) + Uy^{[n]}, \tag{3.1}$$

$$y^{[n+1]} = BhF(Y) + Vy^{[n]}, \tag{3.2}$$

or in a partitioned matrix form

$$\begin{bmatrix} Y \\ y^{[n+1]} \end{bmatrix} = \begin{bmatrix} A & U \\ B & V \end{bmatrix} \begin{bmatrix} hF(Y) \\ y^{[n]} \end{bmatrix}, \tag{3.3}$$

where

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \qquad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_p^{[n]} \end{bmatrix}, \qquad F(Y) = \begin{bmatrix} f(Y_1) \\ f(Y_2) \\ \vdots \\ f(Y_s) \end{bmatrix}.$$

The Y_i are called the internal stages and the $y_i^{[n]}$ the external stages. Since DIMSIMs are general linear methods equations (3.1-3.2) are equivalent to (1.7-1.8).

The structure of matrix A, which corresponds to the coefficient matrix in a Runge-Kutta method, plays a central role in the implementation costs of the method. In order to be competitive for practical computation the A matrix is required to be lower triangular and based on this structure, we have four 'types' of DIMSIMs, each of which is capable of solving stiff or nonstiff problems in a parallel or sequential environment, as in Table 3.1.

In identifying these four types of DIMSIMs, methods in which the A matrix is full has not been considered because such methods would then have very high computational costs, similar to that of the fully implicit Runge-Kutta methods.

Type	Structure of A	Problem	Computer
1	$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ a_{s1} & a_{s2} & a_{s3} & \cdots & 0 \end{bmatrix}$	nonstiff	sequential
2	$\begin{bmatrix} \lambda & 0 & 0 & \cdots & 0 \\ a_{21} & \lambda & 0 & \cdots & 0 \\ a_{31} & a_{32} & \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ a_{s1} & a_{s2} & a_{s3} & \cdots & \lambda \end{bmatrix}$	stiff	sequential
3	$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$	nonstiff	parallel
4	$\begin{bmatrix} \lambda & 0 & 0 & \cdots & 0 \\ 0 & \lambda & 0 & \cdots & 0 \\ 0 & 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \lambda \end{bmatrix}$	stiff	parallel

Table 3.1: Types of DIMSIMs.

Furthermore, the stage order is required to be identical to the overall order or be close to it. This property is necessary for stiff problems and will allow low cost local error estimation and interpolatory output.

DIMSIMs of types 1 and 3 are explicit methods while types 2 and 4 are diagonally implicit. The explicit DIMSIMs have an advantage over explicit Runge-Kutta methods in that the order barriers do not apply to the explicit methods. Hence, there exist explicit DIMSIMs in which $p = s$ for high values of p. The diagonally implicit DIMSIMs have computational complexity similar to the DIRK methods

but do not suffer from low stage orders. Some of the first references on DIMSIMs considered methods in which the $p = q = r = s$, while a later paper of Butcher and Jackiewicz [30] considered adjacent methods in which $\{r = q = s+1, p = q$ or $p = q+1\}$, $\{s = r+1 = q, p = q$ or $p = q+1\}$, and $\{s = r = q, p = q+1\}$. Many issues of implementation were outlined in another paper by Butcher and Jackiewicz [31] and by Butcher [22]. Some techniques on error estimation, stepsize changing and interpolation are outlined in these papers, and some of these ideas will be used in an implementation of type 4 methods later in this thesis.

Jackiewicz, Vermiglio and Zenarro [59] proposed an alternative stepsize changing strategy in which the method coefficients are varied and showed how the incorporation of additional external stages could provide a continuous method.

In a research report by Van Wieren [75], an alternative error estimation technique, based on the use of a method with its companion method, is outlined for a type 2 DIMSIM. In this approach two methods which differ only in the B and V matrices and error constants are used to get two solutions. Then a weighted difference of the solutions provides an estimate of the error. This method is similar to the embedded approach used for many Runge-Kutta methods. According to this report this new estimator worked well for some selected problems.

Type 3 methods have been introduced in a paper by Butcher [21] and analysed further in [20]. These methods can be used to solve nonstiff problems in a parallel environment. Since for type 3 methods $A = 0$, the stages do not depend on the stage derivatives. These methods have bounded stability regions, whose size depend on the choice of the methods parameters.

In the next section we consider the DIMSIMs in a general way to determine the order conditions and later focus on the analysis of the type 4 DIMSIMs.

3.1.1 Order conditions

We seek a method of order p, using starting values of the form

$$y_i^{[0]} = \sum_{k=0}^{p} \alpha_{ik} y^{(k)}(x_0) h^k + O\left(h^{p+1}\right), \tag{3.4}$$

where $y_i^{[n]}$ denotes the approximation number i at the integration point n. α_{ik} must be chosen so that, within the first step, with stepsize h, the stage values, which are approximations to the solution at points $x_0 + hc_i$, $i = 1, 2, \ldots, s$, are given by

$$Y_i = \sum_{k=0}^{p} \frac{c_i^k}{k!} y^{(k)}(x_0) h^k + O\left(h^{p+1}\right), \tag{3.5}$$

and the output values computed at the end of the step are given by

$$y_i^{[1]} = \sum_{k=0}^{p} \alpha_{ik} y^{(k)}(x_1) h^k + O\left(h^{p+1}\right), \tag{3.6}$$

with $x_1 = x_0 + h$. Although the starting values are given in (3.4) by a weighted Taylor series, there is no need in practice to evaluate the various derivatives.

Theorem 3.1 *The numbers $\alpha_{ik}, i = 1, 2, \ldots, r, k = 0, 1, 2, \ldots, p$, in (3.4) and the matrices A, B, U and V satisfy (3.5) and (3.6) if and only if*

$$\exp(cz) = zA\exp(cz) + Uw + O\left(z^{p+1}\right), \tag{3.7}$$

$$\exp(z)w = zB\exp(cz) + Vw + O\left(z^{p+1}\right), \tag{3.8}$$

where $\exp(cz)$ *denotes a vector with components given by* $\exp(c_i z)$, $i = 1, 2, \ldots, s$, *and w denotes the vector with elements given by*

$$w_i = \sum_{k=0}^{p} \alpha_{ik} z^k, \qquad i = 1, 2, \ldots, r. \tag{3.9}$$

This theorem is very important as it simplifies the order conditions which otherwise need to be analysed using the concept of the *B Series*, the simplifying assumptions and the appropriate choice of a function called the correct value function [10, 49].

Proof The proof of these order conditions is given in [19], but because of its special relevance to this work, it is repeated here. Since Y_i, given by (3.5), satisfies

$$Y_i = y(x_0 + hc_i) + O\left(h^{p+1}\right), \tag{3.10}$$

it follows that

$$\begin{aligned} hf(Y_i) &= hy'(x_0 + hc_i) + O\left(h^{p+2}\right) \\ &= \sum_{k=1}^{p+1} \frac{c_i^{k-1}}{(k-1)!} y^{(k)}(x_0) h^k + O\left(h^{p+2}\right) \\ &= \sum_{k=1}^{p} \frac{c_i^{k-1}}{(k-1)!} y^{(k)}(x_0) h^k + O\left(h^{p+1}\right). \end{aligned}$$

Using Taylor expansions, (3.6) can be written as

$$y_i^{[1]} = \sum_{k=0}^{p} \left(\sum_{l=0}^{k} \frac{1}{l!} \alpha_{i,k-1} \right) \alpha_{ik} y^{(k)}(x_0) h^k + O\left(h^{p+1}\right). \tag{3.11}$$

Substituting the Taylor expansions for Y_i, $hf(Y_i)$, and $y_i^{[1]}$ in the equations

$$\begin{aligned} Y_i &= h \sum_{j=1}^{s} a_{ij} f(Y_j) + \sum_{j=1}^{r} u_{ij} y_j^{[0]}, \\ y_i^{[1]} &= h \sum_{j=1}^{s} b_{ij} f(Y_j) + \sum_{j=1}^{r} v_{ij} y_j^{[0]}, \end{aligned}$$

which define the computations performed in the first step, it is found that

$$\sum_{k=0}^{p} \left(c_i^k - \sum_{j=1}^{s} k a_{ij} c_j^{k-1} - \sum_{j=1}^{r} u_{ij} \alpha_{jk} k! \right) \frac{h^k}{k!} y^{(k)}(x_0) = O\left(h^{p+1}\right),$$

$$\sum_{k=0}^{p} \left(\sum_{l=0}^{k} \frac{k!}{l!} \alpha_{i,k-1} - \sum_{j=1}^{s} k b_{ij} c_j^{k-1} - \sum_{j=1}^{r} v_{ij} \alpha_{jk} k! \right) \frac{h^k}{k!} y^{(k)}(x_0) = O\left(h^{p+1}\right).$$

Equating the coefficients of $h^k y^{(k)}(x_0)/k!$, $k = 0, 1, \ldots, p$, to zero and then multiplying these coefficients by $z^k/k!$ and adding from $k = 0$ to $k = p$, we obtain

$$\exp(c_i z) - \sum_{j=1}^{s} z a_{ij} \exp(c_j z) - \sum_{j=1}^{r} u_{ij} w_j = O\left(z^{p+1}\right), \quad i = 1, 2, \ldots, s,$$

$$\exp(z) w_i - \sum_{j=1}^{s} z b_{ij} \exp(c_j z) - \sum_{j=1}^{r} v_{ij} w_j = O\left(z^{p+1}\right), \quad i = 1, 2, \ldots, r,$$

which are equivalent to (3.7) and (3.8) respectively. □

Another key result in [19] is the formula for the elements of matrix B when $p = r = s$ and the components of c are distinct.

Theorem 3.2 *Let $p = r = s$ and $U = I$ then the DIMSIM (3.3) with $Ve = e$ is of order p and of stage order p if and only if*

$$B = B_0 - AB_1 - VB_2 + VA, \tag{3.12}$$

where the (i, j) elements of B_0, B_1 and B_2 are given respectively by

$$\frac{\int_0^{1+c_i} \phi_j(x)dx}{\phi_j(c_j)}, \qquad \frac{\phi_j(1+ci)}{\phi_j(c_j)}, \qquad \frac{\int_0^{c_i} \phi_j(x)}{\phi_j(c_j)}dx,$$

and, for $j = 1, 2, \ldots, s$,

$$\phi_j(x) = \prod_{k \neq j}(x - c_k).$$

Proof A more detailed proof than the one given in [19] is given here.

From (3.7) the vector valued function $w(x)$ satisfies

$$w(x) = (I - zA)\exp(cz) + O\left(z^{p+1}\right),$$

where, here and elsewhere, $u(z) = v(z)+O(z^m)$, for u and v vector valued functions of z, will mean that $\|u\| - \|v\| = O(z^m)$. Substituting this last result for $w(x)$ into (3.8), we get

$$(B - VA)z\exp(cz) = \exp(z)\exp(cz) - Az\exp(z)\exp(cz)$$

$$- V\exp(cz) + O\left(z^{p+1}\right). \tag{3.13}$$

The condition $Ve = e$, is found by substituting $z = 0$. Given this condition, we operate on (3.13) by $\Phi_j(D)$, $j = 1, 2, \ldots, s$, where

$$\Phi_j(x) = \int_0^x \phi_j(t)dt,$$

and D denotes differentiation with respect to z, and set $z = 0$. Since the set of polynomials consisting of 1, together with Φ_j, for $j = 1, 2, \ldots, s$, form a basis for

the set of polynomials of degree not exceeding $p = s$, the result of carrying out these operation and the substituting $z = 0$ is equivalent to equating the coefficients up to degree p in the Taylor series of (3.13) to zero. Since $\Phi_j(D)(O(z^{p+1})) = 0$, we get

$$\begin{aligned}(B - VA)\left(\Phi_j(D) z\exp(cz)\right)|_{z=0} &= \left(\Phi_j(D)\exp(z)\exp(cz)\right)|_{z=0} \\ &- A\left(\Phi_j(D) z\exp(z)\exp(cz)\right)|_{z=0} - V\left(\Phi_j(D)\exp(cz)\right)|_{z=0} \qquad (3.14)\end{aligned}$$

By writing $\phi_j(x)$ as

$$\phi_j(x) = \prod_{k\neq j}(x - c_k) = \sum_{k=1}^{s} d_{k-1}x^{k-1}$$

for some appropriate d_i, $i = 0, 1, \ldots, s-1$, we have

$$\begin{aligned}\Phi_j(D)\left(z\exp(cz)\right)|_{z=0} &= \sum_{k=1}^{s}\frac{d_{k-1}}{k}D^k\left(z\exp(cz)\right)|_{z=0} \\ &= \sum_{k=1}^{s} d_{k-1}D^{k-1}\left(\exp(cz)\right)|_{z=0} \\ &= \phi_j(D)\left(\exp(cz)\right)|_{z=0} \\ &= \prod_{k\neq j}(D - c_k)\exp(cz)|_{z=0} \\ &= \phi_j(c_j)e_j,\end{aligned}$$

where e_j is a vector with a 1 in the j^{th} position and zeros elsewhere. Furthermore, we have

$$\begin{aligned}\Phi_j(D)\left(\exp(cz)\right)|_{z=0} &= \sum_{k=1}^{s}\frac{d_{k-1}}{k}D^k\left(\exp(cz)\right)|_{z=0} \\ &= \sum_{k=1}^{s} d_{k-1}\frac{c^k}{k} \\ &= \int_0^{c_i}\phi_j(x)dx, \quad i = 1, 2, \ldots, s.\end{aligned}$$

Similarly

$$\begin{aligned}\Phi_j(D)\left(\exp(z)\exp(cz)\right)|_{z=0} &= \Phi_j(D)\left(\exp(e+c)z\right)|_{z=0} \\ &= \int_0^{1+c_i} \phi_j(x)dx, \quad i=1,2,\ldots,s.\end{aligned}$$

We also have

$$\begin{aligned}\Phi_j(D)\left(z\exp(z)\exp(cz)\right)|_{z=0} &= \Phi_j(D)\left(z\exp(e+c)z\right)|_{z=0} \\ &= \phi_j(D)\left(\exp(e+c)z\right)|_{z=0} \\ &= \prod_{k\neq j}(1+c_i-c_k) \\ &= \phi_j(1+c_i), \quad i=1,2,\ldots,s.\end{aligned}$$

Using the expressions derived above, (3.14) simplifies to

$$(B - VA - B_0 - AB_1 - VB_2)e_j = 0,$$

thus completing the proof. □

Theorems 3.1 and 3.2 greatly simplify the construction of DIMSIMs. Once the A and V matrices and the abscissae vector are chosen, Theorem 3.2 then enables the calculation of matrix B.

3.1.2 Consistency, convergence and stability

The starting values and the calculated values of DIMSIMs are more general quantities as given by equations (3.4) and (3.6) respectively, while the internal stage values satisfy (3.5). If we let $\upsilon = [\alpha_{10}, \alpha_{20}, \ldots, \alpha_{r0}]$ for notational convenience, then the starting values, the external stage values and the internal stage values satisfy

$$\begin{aligned}y^{[0]} &= \upsilon y(x_0) + O(h), \\ y^{[1]} &= \upsilon y(x_1) + O(h) = \upsilon y(x_0) + O(h), \\ Y &= y(x_0) + O(h).\end{aligned}$$

Using these in the method given by equations (3.1-3.2), we get the relationships

$$Uv = e \quad \text{and} \quad Vv = v,$$

which are called the *preconsistency* conditions and a method which satisfies these conditions is said to be *preconsistent*. Since the choice $U = I$ is used for DIMSIMs, these reduce to the condition $Ve = e$.

Let the starting values, calculated values and the stage values satisfy

$$\begin{aligned} y^{[0]} &= vy(x_0) + \omega h y'(x_0) + O(h^2), \\ y^{[1]} &= vy(x_1) + \omega h y'(x_1) + O(h^2), \\ Y &= y(x_0) + cy'(x_0) + O(h^2), \end{aligned}$$

where we have used $\omega = [\alpha_{11}, \alpha_{21}, \ldots, \alpha_{r1}]$. We also have

$$hF(Y) = hy'(x_0 + ch) = hy'(x_0) + O(h^2).$$

Assuming that the method is preconsistent, then the use of (3.1-3.2) for one step gives

$$Be + V\omega = \omega + e,$$

which is called the *consistency* condition. We also obtain

$$Ae + Uv = c,$$

which represents the relative spacing of the vectors $Y_1, Y_2, \ldots, Y_s$ as approximations to the values of the solution $y(x)$. Hence, the consistency condition ensures that the accuracy of the calculated solution is at least of first order and any useful numerical method must be consistent.

The preconsistency and consistency conditions can also be viewed as the requirement that the methods can solve exactly the differential equations $y' = 0$ and $y' = 1$, respectively.

Furthermore, when solving the differential equation $y' = 0$, we have

$$y^{[n]} = Vy^{[n-1]} = V^n y^{[0]}.$$

To ensure that the solution remains stable, it is necessary that matrix V be power bounded, that is, $\|V^n\| \leq K$ for some constant K and $n = 1, 2, \ldots$. The eigenvalues of matrix V determine the power boundedness or the zero stability of the method, and must all have magnitudes not greater than 1. A typical design choice is to require all eigenvalues to be equal to zero except for the unit eigenvalue associated with eigenvector e, which results in V being of rank 1 and all rows equal.

As in [17], convergence can be defined informally as the condition that if

$$y^{[0]} = vy(x_0) + O(h),$$

then

$$y^{[n]} = vy(x_0 + nh) + O(h),$$

for all n, subject to bounded nh. This condition guarantees that, given accurate enough initial approximations, we can obtain an arbitrarily accurate numerical solution at any fixed point, by taking h sufficiently small.

The relationship concerning the concepts of consistency, stability and convergence is that stability and consistency are necessary and sufficient for convergence and this has been proved in [12] for general linear methods.

3.2 Analysis of type 4 DIMSIMs

Type 4 DIMSIMs are intended for the solution of stiff problems in a parallel computing environment.

Stiffly accurate type 4 DIMSIMs are discussed in a paper by Butcher and Chartier [26]. As for stiffly accurate Runge-Kutta methods, the following definition of stiffly accurate type 4 DIMSIMs has been adopted.

Definition 3.3 *An s-stage r-value type 4 DIMSIM characterised by the matrices A, B, U and V, is said to be stiffly accurate iff*

- *A is non-singular,*

- $M(\infty) = V - BA^{-1}U = 0$,
- *B is of rank r.*

The stiffly accurate methods considered in [26] have matrix A of the form, $A = \text{diag}\{\lambda_i\}$. According to the authors the stronger condition, $M(\infty) = 0$, is desirable for methods which will be used for solving differential algebraic equations. Since the requirement, $M(\infty) = 0$, implies that $B = VA$, the derivation of methods is simplified a great deal. In particular, there is no need to consider separately cases for different ranks for the V matrix. However, it is shown that many of the methods which satisfy $M(\infty) = 0$ are not A-stable. In particular, it is shown that when $p = q = r = s = 3$ the method is not A-stable. Methods which satisfy $r = s = p = q + 1$ and $p = q = r - 1 = s - 1$ are A-stable. The construction of methods with $p = q = r - 1 = s - 1$ is considered. For these methods the λ_i's and the abscissae are free parameters. Methods in which the λ_i's and the abscissae are chosen to be based on the roots of the shifted Chebyschev polynomial of the second kind, are considered, in order to have small coefficients. However, it is shown that even with this choice, where the abscissae lie in $[0, 1]$, the size of the coefficients grow dramatically as the order increases.

The example below is a third order type 4 method which satisfies $M(\infty) = 0$ [26].

Example 3.1 An order 3 method with 4 stages, $c = [0, \frac{1}{4}, \frac{3}{4}, 1]^T$ and $U = I$ has

$$A = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{3}{8} & 0 & 0 \\ 0 & 0 & \frac{5}{8} & 0 \\ 0 & 0 & 0 & \frac{3}{4} \end{bmatrix} \quad \text{and} \quad V = \begin{bmatrix} \frac{1}{4} & -\frac{2}{3} & 2 & -\frac{7}{12} \\ \frac{13}{12} & -\frac{31}{12} & \frac{47}{12} & -\frac{17}{12} \\ \frac{21}{4} & -\frac{137}{12} & \frac{51}{4} & -\frac{67}{12} \\ \frac{109}{12} & -\frac{58}{3} & \frac{62}{3} & -\frac{113}{12} \end{bmatrix},$$

while B is determined uniquely by the requirement, $V - BA^{-1}U = 0$. □

Some low order methods performed well when used to solve selected stiff problems.

Further type 4 methods, in which matrix A has a constant diagonal, have been derived by Butcher [24]. These methods satisfy $\rho(M(\infty)) = 0$ and A-stable methods in which $p = q = r = s$ have been derived for orders up to 10. This means

that the matrices A, B, U and V which define the method are all square and of size $r \times r$. The A matrix takes the form $A = \lambda I$, and since $r = s$ the special choice, $U = I$, is used. In the remaining part of this chapter we analyse these methods. Since $p = q = r = s$, we will use p to represent q, r and s, for uniformity. An implementation of these methods is discussed in Chapter 4.

3.2.1 Stability analysis

If we apply the DIMSIM method (3.3) to the standard test equation (1.4), the stability matrix is given by

$$M(z) = V + zB(I - zA)^{-1}U, \tag{3.15}$$

and using $A = \lambda I$ and $U = I$, it simplifies to

$$M(z) \;=\; V + \frac{z}{1 - \lambda z}B. \tag{3.16}$$

Stability is then determined by the polynomial

$$\begin{aligned} \varphi(w, z) &= \det(wI - M(z)) \\ &= (1 - \lambda z)^{-p} \det\left((1 - \lambda z)wI - (V + z(B - \lambda V))\right) \end{aligned} \tag{3.17}$$

or alternatively, by the polynomial

$$\phi(w, z) = \det\left((1 - \lambda z)wI - (V + z(B - \lambda V))\right), \tag{3.18}$$

and on substituting $B = B_0 - \lambda B_1 - VB_2 + \lambda V$, from (3.12) this reduces to

$$\phi(w, z) = \det\left((1 - \lambda z)wI - (V + z(B_0 - \lambda B_1 - VB_2))\right). \tag{3.19}$$

For the linear test equation considered, the stability region is determined by the requirement that for z in this region, it is not possible that $\phi(w, z) = 0$ if $|w| > 1$. The simplest possibility for the V matrix is rank 1. Making the rank 1 assumption $V = ev^T$, we examine the determinant of the matrix

$$\begin{aligned} P &= (1 - \lambda z)wI - (V + z(B_0 - \lambda B_1 - VB_2)) \\ &= (1 - \lambda z)wI - z(B_0 - \lambda B_1 - VB_2) - V \\ &= (1 - \lambda z)wI - z\widehat{B} - V. \end{aligned}$$

For any non-singular square matrix X, we have

$$\begin{aligned}\det(X+V) &= \det(X+ev^T)\\ &= \det(X(I+X^{-1}ev^T))\\ &= \det(X)\det(I+X^{-1}ev^T).\end{aligned}$$

We also have

$$\text{eig}(X^{-1}ev^T) = \{v^TX^{-1}e, 0, 0, \ldots, 0\},$$

and

$$\text{eig}(I+X^{-1}ev^T) = \{1+v^TX^{-1}e, 1, 1, \ldots, 1\}.$$

Since the determinant of any matrix is the product of its eigenvalues, we have

$$\det(I+X^{-1}ev^T) = 1+v^TX^{-1}e,$$

using which, we get

$$\det(X+ev^T) = \det(X)\left(1+v^TX^{-1}e\right) = \det(X)+v^T\text{adj}(X)e,$$

where we have used $X^{-1} = \text{adj}(X)/\det(X)$, and $\text{adj}(X)$ is the transposed matrix of cofactors of X. Applying this result to $\det(P)$ we get

$$\begin{aligned}\det(P) &= \det\left((1-\lambda z)wI - z\widehat{B} - V\right)\\ &= \det\left((1-\lambda z)wI - z\widehat{B}\right) - v^T\text{adj}\left((1-\lambda z)wI - z\widehat{B}\right)e.\end{aligned}$$

Now $\det\left((1-\lambda z)wI - z\widehat{B}\right)$ is a homogeneous polynomial in $(1-\lambda z)w$ and z of degree p, while $\text{adj}\left((1-\lambda z)wI - z\widehat{B}\right)$ is also a homogeneous polynomial in the same variables but of degree $p-1$. This implies that ϕ takes the form

$$\begin{aligned}\phi_1(w,z) &= (1-\lambda z)^pw^p - (\alpha_0-\beta_1 z)(1-\lambda z)^{p-1}w^{p-1}\\ &\quad -z(\alpha_1-\beta_2 z)(1-\lambda z)^{p-2}w^{p-2} - \cdots - z^{p-1}(\alpha_{p-1}-\beta_p z), \qquad (3.20)\end{aligned}$$

where the notation $\phi_1(w,z)$ represents the stability polynomial for rank 1 methods.

Example 3.2 For a method with $p = q = r = s = 2$, $A = \lambda I$, $U = I$ and

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}, \qquad V = \begin{bmatrix} 1-v & v \\ 1-v & v \end{bmatrix},$$

it can be shown that the stability polynomial (3.18) satisfies

$$\phi_1(w, z) = (1 - \lambda z)^2 w^2 - (\alpha_0 - \beta_1 z)(1 - \lambda z)w - z(\alpha_1 - \beta_2 z),$$

with

$$\begin{aligned} \alpha_0 &= 1, \\ \alpha_1 &= b_{12} - b_{22} + v(-b_{11} - b_{12} + b_{21} + b_{22}), \\ \beta_1 &= b_{11} + b_{22} - \lambda, \\ \beta_2 &= b_{12}b_{21} - b_{11}b_{22} + \lambda(b_{22} - b_{12}) + \lambda v(b_{11} + b_{12} - b_{21} - b_{22}). \end{aligned}$$

□

The main stability requirements for type 4 methods are that the stability region includes all the points in the left half complex plane, with possibly some stronger property at infinity.

Let $t = (1 - \lambda z)w$. Since $\phi_1(\exp(z), z) = O\left(z^{p+1}\right)$, we get from (3.20)

$$t^p - (\alpha_0 - \beta_1 z)t^{p-1} - z(\alpha_1 - \beta_2 z)t^{p-2} - \cdots - z^{p-1}(\alpha_{p-1} - \beta_p z) = O\left(z^{p+1}\right)$$

and upon rearrangement

$$t^p + \beta_1 z t^{p-1} + \beta_2 z^2 t^{p-2} + \cdots + \beta_p z^p = \alpha_0 t^{p-1} + \alpha_1 z t^{p-2} + \cdots + \alpha_{p-1} z^{p-1} + O\left(z^{p+1}\right).$$

On factorising the term t on the left hand side and rearrangement, we get

$$\begin{aligned} t &= \frac{\alpha_0 t^{p-1} + \alpha_1 z t^{p-2} + \cdots + \alpha_{p-1} z^{p-1}}{t^{p-1} + \beta_1 z t^{p-2} + \beta_2 z^2 t^{p-3} + \cdots + \beta_p z^{p-1} t^{-1}} \quad + \quad O\left(z^{p+1}\right) \\ &= \frac{\alpha_0 + \alpha_1\left(\frac{z}{t}\right) + \cdots + \alpha_{p-1}\left(\frac{z}{t}\right)^{p-1}}{1 + \beta_1\left(\frac{z}{t}\right) + \cdots + \beta_p\left(\frac{z}{t}\right)^{p}} \quad + \quad O\left(z^{p+1}\right). \end{aligned}$$

Hence, we can write the relation between w and z given by (3.20) in the form

$$\begin{aligned} w(1-\lambda z) &= \frac{\alpha_0 + \alpha_1\left(\frac{z}{w(1-\lambda z)}\right) + \cdots + \alpha_{p-1}\left(\frac{z}{w(1-\lambda z)}\right)^{p-1}}{1 + \beta_1\left(\frac{z}{w(1-\lambda z)}\right) + \cdots + \beta_p\left(\frac{z}{w(1-\lambda z)}\right)^{p}} \quad + \quad O\left(z^{p+1}\right) \\ &= \widehat{F}\left(\frac{z}{w(1-\lambda z)}\right), \end{aligned}$$

where $\widehat{F}$ is now an approximation, correct to within

$$O\left(\left(\frac{z}{w(1-\lambda z)}\right)^{p+1}\right),$$

to the function F defined by

$$\exp(z)(1-\lambda z) = F\left(\frac{z}{\exp(z)(1-\lambda z)}\right).$$

The rational approximations to F are investigated in [24], and the next theorem gives the first row, corresponding to $\beta_1 = \beta_2 = \cdots = \beta_p = 0$, and the first column, corresponding to $\alpha_1 = \alpha_2 = \cdots = \alpha_{p-1} = 0$, of the Padé table for this function. Here the notation $L_n(x)$ denotes the Laguerre polynomial of degree n, defined by (2.15), and $L'_n(x)$ represents its derivative.

Theorem 3.4 *Let λ denote a positive real number. Define F in the neighbourhood of 0 by the series*

$$F(Z) = 1 + \sum_{n=1}^{\infty} (-1)^{n+1} \frac{\lambda^n}{n+1} L'_{n+1}\left(\frac{n+1}{\lambda}\right) Z^n. \tag{3.21}$$

Then F satisfies the functional equation

$$\exp(z)(1-\lambda z) = F\left(\frac{z}{\exp(z)(1-\lambda z)}\right), \tag{3.22}$$

for sufficiently small $|z|$. Furthermore, the function G defined in a neighbourhood of 0 by the series

$$G(Z) = 1 + (\lambda - 1)Z + \sum_{n=2}^{\infty} (-1)^{n+1} \frac{\lambda^{n-1}}{n} L'_n\left(\frac{n-1}{\lambda}\right) Z^n, \tag{3.23}$$

satisfies the equation

$$F(Z)G(Z) = 1,$$

for sufficiently small $|z|$.

Proof Let $Z = z/(\exp(z)(1-\lambda z))$, so that $F(Z) = z/Z$, and let $\psi(z) = (1-\lambda z)\exp(z)$, so that we have

$$z = Z\psi(z) = \alpha_0 Z + \alpha_1 Z^2 + \alpha_2 Z^3 + \ldots,$$

for the case when $\beta_1 = \beta_2 = \cdots = \beta_p = 0$. We find the coefficients in the above Taylor expansion using the Lagrange series. Consider the the power series [66]

$$w = a_1 z + a_2 z^2 + \cdots + a_n z^n + \ldots$$

where $a_1 \neq 0$ and which converges for $z = 0$. The relationship between z and w can also be represented as

$$z = b_1 w + b_2 w^2 + \cdots + b_n w^n + \ldots$$

where $a_1 b_1 = 1$. To compute the second series from the first, we set

$$\vartheta(z) = \frac{1}{a_1 + a_2 z + \cdots + a_n z^{n-1} + \ldots},$$

then the equation $z = w\vartheta(z)$, where $\vartheta(z)$ is regular in a neighbourhood of $z = 0$ implies that

$$\left.\begin{array}{rcl} z & = & b_1 w + b_2 w^2 + \cdots + b_n w^n + \ldots \\ b_n & = & \frac{1}{n!}\left[\frac{d^{n-1}}{dx^{n-1}}[\vartheta(x)]^n\right]_{x=0} \end{array}\right\} \tag{3.24}$$

More generally, with $f(z)$ regular in the neighbourhood of $z = 0$, we have

$$\left.\begin{array}{rcl} f(z) & = & b_0 + b_1 w + b_2 w^2 + \cdots + b_n w^n + \ldots \\ b_0 & = & f(0) \\ b_n & = & \frac{1}{n!}\left[\frac{d^{n-1}}{dx^{n-1}} f'(x)[\vartheta(x)]^n\right]_{x=0}. \end{array}\right\} \tag{3.25}$$

Applying (3.24) to

$$z = Z\psi(z) = \alpha_0 Z + \alpha_1 Z^2 + \alpha_2 Z^3 + \ldots$$

gives

$$\begin{aligned}
\alpha_n &= \frac{1}{(n+1)!}\left[\frac{d^n}{dz^n}[\psi(z)]^{n+1}\right]_{z=0} \\
&= \frac{1}{(n+1)!}\left[\frac{d^n}{dz^n}(1-\lambda z)^{n+1}\exp((n+1)z)\right]_{z=0} \\
&= \frac{1}{(n+1)!}\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right)\frac{d^{n-k}}{dz^{n-k}}(1-\lambda z)^{n+1}\bigg|_{z=0}\quad\frac{d^k}{dz^k}\left(\exp((n+1)z)\right)\bigg|_{z=0} \\
&= \frac{1}{(n+1)!}\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right)\frac{(n+1)!}{(k+1)!}(-\lambda)^{n-k}(n+1)^k \\
&= (-1)^n\lambda^n\sum_{k=0}^{n}\left(\begin{array}{c} n \\ k \end{array}\right)\frac{(-1)^k}{(k+1)!}\left(\frac{n+1}{\lambda}\right)^k \\
&= (-1)^{n+1}\frac{\lambda^n}{n+1}L'_{n+1}\left(\frac{n+1}{\lambda}\right).
\end{aligned}$$

Consider $G(Z) = Z/z = 1/(\exp(z)(1-\lambda z)) = \chi(z)$, then from the generalisation (3.25) we have

$$\chi(z) = \beta_0 + \beta_1 Z + \beta_2 z^2 + \cdots,$$

where

$$\beta_0 = \chi(0) = 1, \quad \beta_1 = \chi'(z)\psi(z)|_{z=0} = \lambda - 1,$$

and

$$\begin{aligned}
\beta_n &= \frac{1}{n!}\left[\frac{d^{n-k}}{dz^{n-k}}\chi'(z)[\psi(z)]^n\right]_{z=0} \\
&= \frac{1}{n!}\frac{d^{n-k}}{dz^{n-k}}\left[-\exp((n-1)z)(1-\lambda z)^{n-1} - \lambda\exp((n-1)z)(1-\lambda z)^{n-2}\right]_{z=0} \\
&= (-1)^n\frac{\lambda^{n-1}}{n}\left[L_{n-1}\left(\frac{n-1}{\lambda}\right) - L'_{n-1}\left(\frac{n-1}{\lambda}\right)\right].
\end{aligned}$$

Using the identity

$$L'_n(x) = L'_{n-1}(x) - L_{n-1}(x),$$

it follows that

$$\beta_n = (-1)^n \frac{\lambda^{n-1}}{n} L'_n\left(\frac{n-1}{\lambda}\right).$$

Therefore we obtain

$$G(Z) = 1 + (\lambda - 1)Z + \sum_{n=2}^{\infty} (-1)^{n+1} \frac{\lambda^{n-1}}{n} L'_n\left(\frac{n-1}{\lambda}\right) Z^n,$$

as required. □

3.2.2 Choice of λ for A-stability

Using elements from the first row of the Padé table for the function F, the resulting methods have the property that the stability function takes the form

$$\phi_1(w,z) = (1-\lambda z)^p w^p - (1-\lambda z)^{p-1} w^{p-1}$$

$$- \alpha_1 z(1-\lambda z)^{p-2} w^{p-2} - \cdots - \alpha_{p-1} z^{p-1}, \qquad (3.26)$$

since $\alpha_0 = 1$ and $\beta_i = 0$, $i = 1, 2, \ldots, p$. At $z = \infty$ the stability matrix has characteristic polynomial w^p, so that if a method within this class is A-stable, then it will be L-stable. To obtain the stability function (3.26) the series for F has been truncated one term too soon to obtain order p. Hence, if order p is to be retained then it is necessary to make the next term in the series to be zero. Thus, in search of methods with good order and stability, we choose λ to satisfy the condition

$$L'_{p+1}\left(\frac{p+1}{\lambda}\right) = 0, \qquad (3.27)$$

and the values of $\alpha_0, \alpha_1, \ldots, \alpha_{p-1}$, according to Theorem 3.4 are given by

$$\alpha_n = (-1)^{n+1} \frac{\lambda^n}{(n+1)} L'_{n+1}\left(\frac{n+1}{\lambda}\right). \qquad (3.28)$$

To investigate the possible A-stability of methods defined in this way, a numerical investigation as outlined in [24] can be carried out. By the maximum modulus principle, it is necessary only to verify that for all $y > 0$, the zeros of $\phi_1(w, iy) = 0$ lie in the unit disc. Let $iW = (1 - \lambda iy)w$, then (3.26) with $z = iy$ reduces to

$$\phi_1(W, iy) = -iW^p + W^{p-1} + \alpha_1 y W^{p-2} + \alpha_2 y^2 W^{p-3} + \cdots + \alpha_{p-1} y^{p-1}. \qquad (3.29)$$

Hence, it is equivalent to verify that for all $y > 0$, the zeros of $\phi_1(W, iy)$, always satisfy $|W| \leq \sqrt{1+\lambda^2 y^2}$. This can be checked by plotting $W/\sqrt{1+\lambda^2 y^2}$ for $y > 0$, and checking that the solutions always lie in the unit disc.

Example 3.3 Consider a method with $p = q = r = s = 2$, then the stability polynomial becomes

$$\phi_1(w, z) = (1 - \lambda z)^2 w^2 - (1 - \lambda z)w - \alpha_1 z$$

where we require $L_3'(x) = 0$, so that the method is of order 2 and

$$L_3(x) = 1 - 3x + \frac{3}{2}x^2 - \frac{1}{6}x^3.$$

Then we have

$$\begin{aligned} L_3'(x) &= 0 \quad \Longrightarrow x = 3 \pm \sqrt{3}, \\ L_3'\left(\frac{3}{\lambda}\right) &= 0 \quad \Longrightarrow \lambda = \frac{3 \pm \sqrt{3}}{2} \end{aligned}$$

and

$$\alpha_1 = \frac{\lambda}{2} L_2'\left(\frac{\lambda}{2}\right) = -\lambda + 1 = \frac{-1 \mp \sqrt{3}}{2}$$

Since we have two choices for λ, we can investigate which of these lead to A-stable methods by using the numerical investigation outlined above. Thus, we need to check the roots of

$$\phi_1(W, iy) = -iW^2 + W + \alpha_1 y.$$

As stated earlier, we can check that $|W| \leq \sqrt{1+\lambda^2 y^2}$ by plotting $W/\sqrt{1+\lambda^2 y^2}$ for $y > 0$. For both values of λ, we find that $W/\sqrt{1+\lambda^2 y^2}$ lies inside the unit disc as in Figure 3.1. Hence, both values of λ lead to A-stable methods. □

Example 3.4 For the method of order 3, we need to choose λ to satisfy $L_4'(4/\lambda) = 0$. This gives us three values of λ,

$$\begin{aligned} \lambda_1 &= 0.5155456021, \\ \lambda_2 &= 1.210138313, \\ \lambda_3 &= 4.274316085. \end{aligned}$$

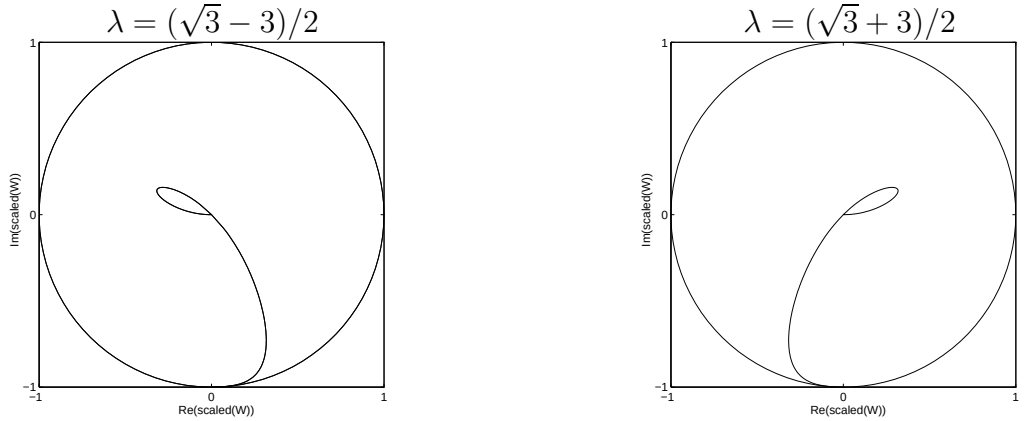

Figure 3.1: Stability plots for $p = 2$, where scaled (W) means $W/\sqrt{1+\lambda^2 y^2}$.

We then need to investigate the roots of

$$\phi_1(W, iy) = -iW^3 + W^2 + \alpha_1 yW + \alpha_2 y^2,$$

where

$$\begin{aligned} \alpha_1 &= 1 - \lambda, \\ \alpha_2 &= \frac{3}{2} - 3\lambda + \lambda^2. \end{aligned}$$

The three plots of $W/\sqrt{1+\lambda^2 y^2}$, corresponding to λ_1, λ_2 and λ_3, are shown in Figure 3.2. Using these three diagrams as well as by examining the maximum numerical values of $W/\sqrt{1+\lambda^2 y^2}$, it is seen that λ_1 does not lead to an A-stable method. λ_3 also does not lead to an A-stable method as one of its 'branches' actually goes slightly outside the unit circle, although this is not entirely obvious from the diagram. The choice of λ_2 leads to an A-stable method. □

Hence, it is seen that there are two A-stable methods of order two and a single method of order three. Stability plots for orders 4 to 8 are shown in [24]. It is found that there is a single method for each of orders 4 to 8 which is A-stable. Although there is no A-stable method of order 9, there is an A-stable method of order 10. Table 3.2 gives a list of values of λ for A-stable methods, where for each positive integer p, $\xi_1^{(p)}$, $\xi_2^{(p)}$,...,$\xi_p^{(p)}$, are the zeros of L'_{p+1}, numbered in increasing order.

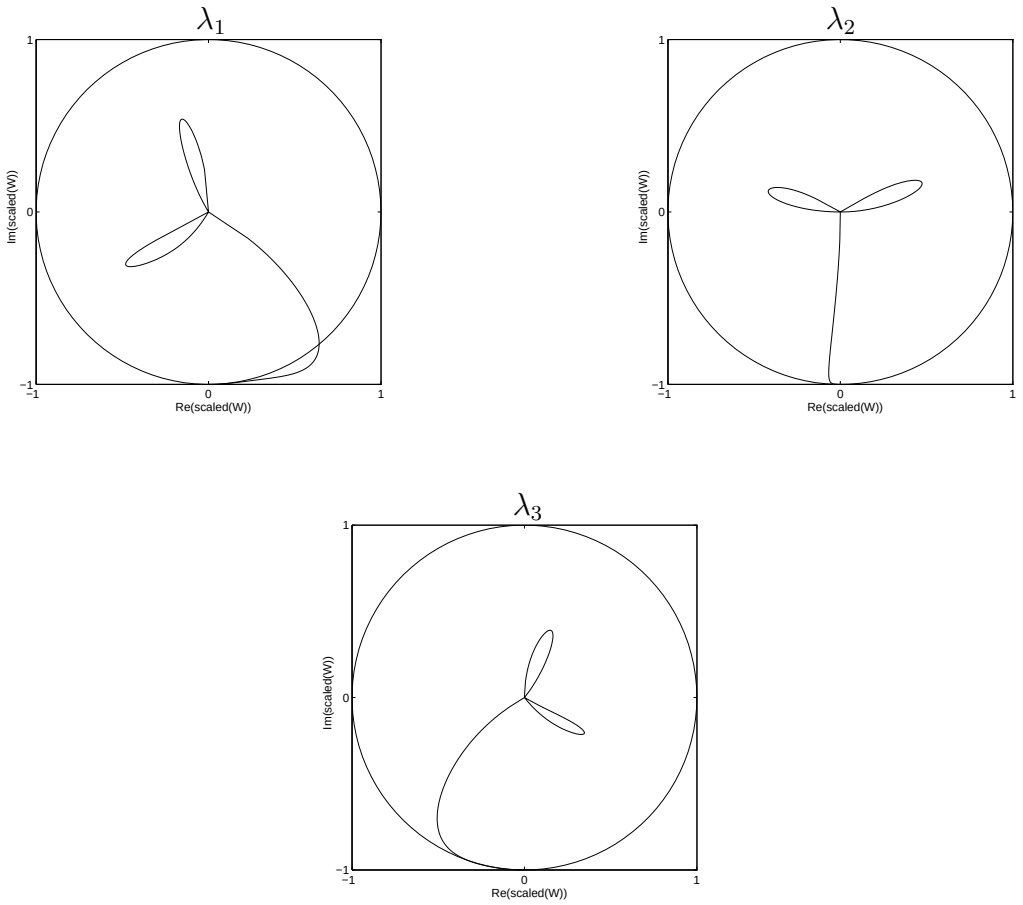

Figure 3.2: Stability plots for $p = 3$, where scaled (W) means $W/\sqrt{1+\lambda^2 y^2}$.

3.2.3 Derivation of A-stable methods

By making the use transformations we can derive formulas for the coefficients of the method. This transformation removes the abscissae from the transformed matrices completely. This convenience of being able to regard the choice of abscissae as a separate question, is possible only for type 3 and 4 methods. The transformations are given by the following theorem [24].

Theorem 3.5 *Consider a DIMSIM method such that* $p = q = r = s$, $A = \lambda I$,

p	λ
1	$\frac{2}{\xi_1^{(1)}} = 1$
2	$\frac{3}{\xi_2^{(2)}} = \frac{3-\sqrt{3}}{2} \approx 0.63$
2	$\frac{3}{\xi_1^{(2)}} = \frac{3+\sqrt{3}}{2} \approx 2.36$
3	$\frac{4}{\xi_2^{(3)}} \approx 1.2101383127$
4	$\frac{5}{\xi_2^{(4)}} \approx 1.9442883555$
5	$\frac{6}{\xi_3^{(5)}} \approx 1.3012832613$
6	$\frac{7}{\xi_3^{(6)}} \approx 1.8056866912$
7	$\frac{8}{\xi_4^{(7)}} \approx 1.3521971029$
8	$\frac{9}{\xi_4^{(8)}} \approx 1.7368002358$
10	$\frac{11}{\xi_5^{(10)}} \approx 1.6956068006$

Table 3.2: λ for A-stability of type 4 methods with $p = s$.

$U = I$, $Ve = e$ and abscissae vector c. Define the matrix T by

$$T = \begin{bmatrix} P^{(p)}(c_1) & P^{(p-1)}(c_1) & P^{(p-2)}(c_1) & \dots & P'(c_1) \\ P^{(p)}(c_2) & P^{(p-1)}(c_2) & P^{(p-2)}(c_2) & \dots & P'(c_2) \\ P^{(p)}(c_3) & P^{(p-1)}(c_3) & P^{(p-2)}(c_3) & \dots & P'(c_3) \\ \vdots & \vdots & \vdots & & \vdots \\ P^{(p)}(c_p) & P^{(p-1)}(c_p) & P^{(p-2)}(c_p) & \dots & P'(c_p) \end{bmatrix}, \tag{3.30}$$

where the polynomial P is defined by

$$P(x) = \frac{1}{p!} \prod_{i=1}^{p} (x - c_i). \tag{3.31}$$

Define

$$\overline{B} = T^{-1} B T, \tag{3.32}$$

$$\overline{V} = T^{-1} V T, \tag{3.33}$$

then

$$\overline{V}e_1 = e_1, \tag{3.34}$$

and

$$\overline{B} = \widehat{E} - \lambda E + \overline{V} \begin{bmatrix} \lambda & 0 & 0 & \cdots & 0 \\ -1 & \lambda & 0 & \cdots & 0 \\ 0 & -1 & \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \ldots & \lambda \end{bmatrix}, \tag{3.35}$$

where

$$\widehat{E} = \begin{bmatrix} 1 & \frac{1}{2!} & \frac{1}{3!} & \cdots & \frac{1}{(p-1)!} & \frac{1}{p!} \\ 1 & 1 & \frac{1}{2!} & \ldots & \frac{1}{(p-2)!} & \frac{1}{(p-1)!} \\ 0 & 1 & 1 & \ldots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & \frac{1}{2!} \\ 0 & 0 & 0 & \ldots & 1 & 1 \end{bmatrix}, \qquad E = \begin{bmatrix} 1 & 1 & \frac{1}{2!} & \cdots & \frac{1}{(p-2)!} & \frac{1}{(p-1)!} \\ 0 & 1 & 1 & \ldots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} \\ 0 & 0 & 1 & \ldots & \frac{1}{(p-4)!} & \frac{1}{(p-3)!} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 1 \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}.$$

Proof Since $Te_1 = e$, (3.34) follows from $Ve = e$. This implies that $\overline{V}$ is of the form

$$\overline{V} = [e_1 | V_0]. \tag{3.36}$$

Furthermore, if $\widehat{V}$ is defined by

$$\widehat{V} = [V_0 | 0], \tag{3.37}$$

then the last term in (3.35) is equal to $\lambda\overline{V} - \widehat{V}$.

Substituting $U = I$ and $A = \lambda I$ into the order condition (3.7) and using the resulting equation to eliminate w in (3.8), we get

$$\exp(z)(1 - \lambda z)\exp(cz) = zB\exp(cz) + V(1 - \lambda z)\exp(cz) + O\left(z^{p+1}\right). \tag{3.38}$$

Define the sequence of numbers $\gamma_1, \gamma_2, \ldots, \gamma_p$ and the function $\phi(z)$ by

$$\begin{aligned} P(x) &= \frac{z^p}{p!} + \gamma_1 \frac{z^{p-1}}{(r-1)!} + \gamma_2 \frac{z^{p-2}}{(r-2)!} + \cdots + \gamma_p, \\ \phi(z) &= 1 + \gamma_1 z + \gamma_2 z^2 + \cdots + \gamma_p z^p. \end{aligned}$$

Define the matrices C and G and the vector Z by

$$C = \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{p!}c_1^p \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{p!}c_2^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_p & \frac{1}{2!}c_p^2 & \cdots & \frac{1}{p!}c_p^p \end{bmatrix}, \quad G = \begin{bmatrix} 1 & \gamma_1 & \gamma_2 & \cdots & \gamma_p \\ 0 & 1 & \gamma_1 & \cdots & \gamma_{p-1} \\ 0 & 0 & 1 & \cdots & \gamma_{p-2} \\ \vdots & \vdots & \vdots & & \gamma_{p-1} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^p \end{bmatrix},$$

then we have

$$GZ = \phi(z)Z + O\left(z^{p+1}\right), \tag{3.39}$$

$$CZ = \exp(cz)Z + O\left(z^{p+1}\right). \tag{3.40}$$

Multiply (3.38) by $\phi(z)$ and substitute for GZ and CZ from (3.39) and (3.40) to obtain the equivalent condition

$$\left(\exp(z)(1-\lambda z) - zB - V(1-\lambda z)\right) CGZ + O\left(z^{p+1}\right). \tag{3.41}$$

We have

$$CG = \left[P^{(p)}(c), P^{(r-1)}(c), P^{(r-2)}(c), \ldots, P'(c), P(c)\right],$$

where the columns consisting of polynomials with argument c are evaluated componentwise. Since the zeros of P are components of c, $P(c) = 0$. Hence, (3.41) can be written in the form

$$\left(\exp(z)(1-\lambda z) - zB - V(1-\lambda z)\right) T\widehat{Z} = O\left(z^{p+1}\right),$$

where $\widehat{Z}$ is a vector which contains the first p components of Z. Premultiplying this simplified order condition by T^{-1} and using (3.32) and (3.33), we get

$$\exp(z)(1-\lambda z)\widehat{Z} = z\overline{B}\widehat{Z} + \overline{V}(1-\lambda z)\widehat{Z} + O\left(z^{p+1}\right),$$

which can be written as

$$\exp(z)\begin{bmatrix} 1-\lambda z \\ z-\lambda z^2 \\ z^2-\lambda z^3 \\ \vdots \\ z^{p-1}-\lambda z^{p-1} \end{bmatrix} = z\overline{B}\begin{bmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{p-1} \end{bmatrix} + \overline{V}\begin{bmatrix} 1-\lambda z \\ z-\lambda z^2 \\ z^2-\lambda z^3 \\ \vdots \\ z^{p-1}-\lambda z^{p-1} \end{bmatrix} + O\left(z^{p+1}\right). \tag{3.42}$$

Since $\overline{V}$ is of the form given by (3.36), we have

$$\overline{V}\widehat{Z} = e_1 e_1^T + z\widehat{V}\widehat{Z}.$$

We also have

$$\exp(z)\widehat{Z} - e_1 e_1^T = z\widehat{E}\widehat{Z}.$$

Hence, subtracting $e_1 e_1^T$ from both sides of (3.42) and dividing by z gives

$$\widehat{E}\widehat{Z} - \lambda E\widehat{Z} = \overline{B}\widehat{Z} + \widehat{V}\widehat{Z} - \lambda\overline{V}\widehat{Z} + O(z^p),$$

from which we obtain

$$\overline{B} = \widehat{E} - \lambda E - \widehat{V} + \lambda\overline{V}. \tag{3.43}$$

This is equivalent to (3.35). □

3.2.4 Methods with rank 1 for V

Here we consider how λ and $\overline{V}$ should be chosen to ensure that the stability matrix of the method, represented by (3.16), has zero spectral radius at infinity. We consider the rank 1 case so that $\overline{V} = e_1[1, v_2, v_3, \ldots, v_p]$. The stability matrix then takes the form $M(z) = \overline{V} + (z/(1-\lambda z))\overline{B}$, and using the known form of $\overline{B}$ and $\overline{V}$, we get

$$\begin{aligned}
\widehat{M} &= -\lambda M(\infty) \\
&= -\lambda\left(\overline{V} - \frac{1}{\lambda}\overline{B}\right) \\
&= \widehat{E} - \lambda E - e_1[v_2, v_3, \ldots, v_p, 0].
\end{aligned}$$

Theorem 3.6 *The matrix $\widehat{M}$ has spectral radius equal to zero if and only if*

$$v_i = \frac{(-1)^{p+1}(p-i+2)}{p+1}\lambda^{i-1}L_{p+1}^{p-i+2}\left(\frac{p+1}{\lambda}\right), \qquad i = 2, 3, \ldots, p,$$

and λ satisfies the equation (3.27) [24].

Proof We compute sequence of vectors given by

$$\begin{aligned}\widehat{u}_1^T &= e_p^T,\\ \widehat{u}_n^T &= \widehat{u}_{n-1}^T\widehat{M}, \qquad n=2,3,\ldots,p+1.\end{aligned}$$

Because of the form of the component matrices, the matrix $\widehat{M}$ has all terms zero below the leading subdiagonal, that is, the Hessenberg form. This means that the members of the sequence $\widehat{u}_1, \widehat{u}_2, \ldots, \widehat{u}_p$ have $p-1, p-2, \ldots, 0$ initial zero components respectively. The leading subdiagonal elements of $\widehat{M}$ are equal to 1 so the first elements in $\widehat{u}_n$, $n = 1, 2, \ldots, p$, after the initial zeros are equal to 1. Denote by u_n the final n-element subvector of $\widehat{u}_n$, $n = 1, 2, \ldots, p$. Since the values of the elements of $\overline{V}$ do not enter into the values of these vectors, we can evaluate them recursively as functions of λ. They are given by the formula

$$u_n^T = \frac{(-1)^n}{n}\left[nL_n^{(n)}\left(\frac{n}{\lambda}\right), (n-1)\lambda L_n^{(n-1)}\left(\frac{n}{\lambda}\right),\right.$$
$$\left.(n-2)\lambda^2 L_n^{(n-2)}\left(\frac{n}{\lambda}\right), \ldots, \lambda^{n-1}L_n^{(1)}\left(\frac{n}{\lambda}\right)\right],$$

where $L_n^{(m)}$ denotes the m^{th} derivative of the Laguerre polynomial of degree n. This formulae can be proved by mathematical induction. The formulae for u_{n+1}^T can be proved using the relation

$$u_{n+1}^T = u_n^T\left(\begin{bmatrix} 1 & 1 & \frac{1}{2!} & \ldots & \frac{1}{(p-2)!} & \frac{1}{(p-1)!}\\ 0 & 1 & 1 & \ldots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!}\\ \vdots & \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & 0 & \ldots & 1 & \frac{1}{2!}\\ 0 & 0 & 0 & \ldots & 1 & 1 \end{bmatrix} - \lambda\begin{bmatrix} 0 & 1 & 1 & \ldots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!}\\ 0 & 0 & 1 & \ldots & \frac{1}{(p-4)!} & \frac{1}{(p-3)!}\\ \vdots & \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & 0 & \ldots & 1 & 1\\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}\right),$$

where we note that the matrices in this equation are the $\widehat{E}$ and the E matrices respectively, with their first row removed. The first row of these matrices will be treated separately since the $\overline{V}$ matrix has non zero elements in the first row.

Performing the vector and matrix multiplication on the right-hand-side we find that the i^{th} component of the resultant vector can be written as

$$\frac{(-1)^n}{n}\sum_{j=1}^{i-1}\frac{\lambda^{i-j-1}}{(j-1)!}L_n^{(n-i+j+1)}\left(\tfrac{n}{\lambda}\right)+\frac{(-1)^n(n-i+1)}{n}\sum_{j=0}^{i-1}\frac{\lambda^{i-j-1}}{j!}L_n^{(n-i+j+1)}\left(\tfrac{n}{\lambda}\right)$$

$$-\frac{(-1)^n\lambda}{n}\sum_{j=1}^{i-2}\frac{\lambda^{i-j-2}}{(j-1)!}L_n^{(n-i+j+2)}\left(\tfrac{n}{\lambda}\right)-\frac{(-1)^n\lambda(n-i+1)}{n}\sum_{j=0}^{i-2}\frac{\lambda^{i-j-2}}{j!}L_n^{(n-i+j+2)}\left(\tfrac{n}{\lambda}\right).$$

We can simplify this further because of the Taylor series embedded in each summation, for example, the first term can be simplified as follows

$$\begin{aligned}\frac{(-1)^n}{n}\sum_{j=1}^{i-1}\frac{\lambda^{i-j-1}}{(j-1)!}L_n^{(n-i+j+1)}\left(\tfrac{n}{\lambda}\right) &= \frac{(-1)^n}{n}\lambda^{i-2}\sum_{j=1}^{i-1}\frac{L_n^{(n-i+j+1)}\left(\frac{n}{\lambda}\right)}{\lambda^{j-1}(j-1)!}\\ &= \frac{(-1)^n}{n}\lambda^{i-2}L_n^{(n-i+2)}\left(\tfrac{n+1}{\lambda}\right).\end{aligned}$$

In a similar way the summations of the other three series can be simplified to give the following equivalent form

$$\frac{(-1)^n}{n}\lambda^{i-2}L_n^{(n-i+2)}\left(\tfrac{n+1}{\lambda}\right)+\frac{(-1)^n(n-i+1)}{n}\lambda^{i-1}L_n^{(n-i+1)}\left(\tfrac{n+1}{\lambda}\right)$$

$$-\frac{(-1)^n}{n}\lambda^{i-2}L_n^{(n-i+3)}\left(\tfrac{n+1}{\lambda}\right)-\frac{(-1)^n(n-i+2)}{n}\lambda^{i-1}L_n^{(n-i+2)}\left(\tfrac{n+1}{\lambda}\right).$$

It is necessary to verify that this is equivalent to

$$\frac{(-1)^{n+1}(n-i+2)}{n+1}\lambda^{i-1}L_{n+1}^{(n-i+2)}\left(\frac{n+1}{\lambda}\right).$$

This follows by differentiating $n-i+1$ times, the differential equation for the Laguerre polynomials

$$xL_n''(x)+(1-x)L_n'(x)+nL_n(x)=0$$

and combining this with the result of differentiating $n-i+1$ times, the identity

$$L_{n+1}'(x)=L_n'(x)-L_n(x).$$

Since $e_p^T \widehat{M}^{p-1}$ is not a zero vector, it is necessary and sufficient for $\rho(M(\infty)) = 0$ that $e_p^T \widehat{M}^p$ is zero. This is equivalent to $\widehat{u}_{p+1}^T = 0$. Hence, we have

$$\begin{aligned} \widehat{u}_{p+1}^T &= \widehat{u}_p^T \widehat{M} = u_p^T \widehat{M} \\ &= u_p^T (\widehat{E} - \lambda E) - u_p^T \overline{V} \\ &= u_{p+1}^T - [v_2, v_3, \ldots, v_p, 0]. \end{aligned}$$

The last result implies that v_i equals the i^{th} component of the vector u_{p+1}^T and is given by

$$\frac{(-1)^{p+1}(p-i+2)}{p+1} \lambda^{i-1} L_{n+1}^{(p-i+2)} \left(\frac{p+1}{\lambda} \right),$$

as required. □

It is possible to find methods for any choice of the abscissae as long as the abscissae are distinct. In [24] two different choices of abscissae are considered. The first choice is motivated by the abscissae for explicit Runge-Kutta methods, where these are uniformly spaced in the current integration step, that is, $c = [0, 1/(p-1), 1/(p-2), \ldots, 1]^T$. The second choice considered is, $c = [-p+2, -p+3, \ldots, 0, 1]^T$, which corresponds to the same spread of abscissae as for the BDF methods. Some rank 1 methods are listed below. It is seen that as the order increases, the magnitudes of the coefficients in the B and V matrices increase. It is further seen that of the two choices of abscissae considered, the case with the equally spaced values inside the integration step, has larger magnitudes in the B and V matrices for a given order. This has also been observed to be the case for the type 3 methods [20] and for the stiffly accurate A-stable type 4 methods [26]. Theoretically, any distinct set of abscissae can be used. However, computational efficiency of the method is likely to depend on the this choice as it affects the magnitude of the method coefficients and will have an influence on round-off errors. Thus very high order methods are not likely to perform well.

3.2.5 Some rank 1 methods

- $p = 2, \quad \lambda = \frac{3-\sqrt{3}}{2}, \quad c = [\ 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} \frac{18-11\sqrt{3}}{4} & \frac{-12+7\sqrt{3}}{4} \\ \frac{22-13\sqrt{3}}{4} & \frac{-12+9\sqrt{3}}{4} \end{bmatrix}, \quad V = \begin{bmatrix} \frac{3-2\sqrt{3}}{2} & \frac{-1+2\sqrt{3}}{4} \\ \frac{3-2\sqrt{3}}{2} & \frac{-1+2\sqrt{3}}{4} \end{bmatrix}.$$

- $p = 3, \quad \lambda = 1.21013831273, \quad c = [\ 0 \quad \frac{1}{2} \quad 1\ ]^T,$

$$B = \begin{bmatrix} -6.4518297302 & 14.0277199958 & -6.4454753274 \\ -7.4536347096 & 16.9914682673 & -7.9074186195 \\ -8.9155780017 & 20.3754931643 & -9.3295002244 \end{bmatrix},$$

$$V = \begin{bmatrix} -4.3541275713 & 10.9690850189 & -5.6149574476 \\ -4.3541275713 & 10.9690850189 & -5.6149574476 \\ -4.3541275713 & 10.9690850189 & -5.6149574476 \end{bmatrix}.$$

- $p = 3, \quad \lambda = 1.21013831273, \quad c = [\ -1 \quad 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} -1.4307831989 & 3.1337349783 & -1.0725368412 \\ -1.5141165323 & 5.0105399577 & -1.8660084873 \\ -2.3075881783 & 7.3076215626 & -2.3696184461 \end{bmatrix},$$

$$V = \begin{bmatrix} -0.8059281583 & 3.2422712547 & -1.4363430965 \\ -0.8059281583 & 3.2422712547 & -1.4363430965 \\ -0.8059281583 & 3.2422712547 & -1.4363430965 \end{bmatrix}.$$

- $p = 4, \quad \lambda = 1.94428836, \quad c = [\ 0 \quad \frac{1}{3} \quad \frac{2}{3} \quad 1\ ]^T,$

$$B = \begin{bmatrix} -324.7473606870 & 967.1475438327 & -925.9431184761 & 287.8200887525 \\ -322.9280723315 & 959.8842792995 & -915.0968327875 & 282.7511125749 \\ -317.8590961538 & 941.4276629444 & -891.9462402547 & 273.3214935529 \\ -308.4294771319 & 908.7781630341 & -853.8251424780 & 258.7536099978 \end{bmatrix},$$

$$V = \begin{bmatrix} -153.6462411729 & 482.0084784307 & -490.2467730766 & 162.8845358188 \\ -153.6462411729 & 482.0084784307 & -490.2467730766 & 162.8845358188 \\ -153.6462411729 & 482.0084784307 & -490.2467730766 & 162.8845358188 \\ -153.6462411729 & 482.0084784307 & -490.2467730766 & 162.8845358188 \end{bmatrix}.$$

- $p = 4, \quad \lambda = 1.94428836, \quad c = [\ -2 \quad -1 \quad 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} -11.8430964150 & 41.2047385552 & -34.4921752375 & 8.4076865194 \\ -11.8847630817 & 43.6906935774 & -35.8947969264 & 8.3660198528 \\ -11.8430964150 & 43.4823602441 & -33.1588419042 & 6.7967314972 \\ -10.2738080595 & 37.2468734887 & -23.9514451044 & 3.2555330974 \end{bmatrix},$$

$$V = \begin{bmatrix} -4.7518271422 & 19.6331861055 & -19.7337373625 & 5.8523783991 \\ -4.7518271422 & 19.6331861055 & -19.7337373625 & 5.8523783991 \\ -4.7518271422 & 19.6331861055 & -19.7337373625 & 5.8523783991 \\ -4.7518271422 & 19.6331861055 & -19.7337373625 & 5.8523783991 \end{bmatrix}.$$

- $p = 5, \quad \lambda = 1.3012832613, \quad c = [\ 0 \quad \frac{1}{4} \quad \frac{1}{2} \quad \frac{3}{4} \quad 1\ ]^T,$

$$B = \begin{bmatrix} 968.5096071975 & -3924.2586814438 & 5885.2161044676 & -3866.8652650014 & 939.4046510865 \\ 967.2954767140 & -3918.1946262485 & 5873.1116051880 & -3854.8156268329 & 934.8595874858 \\ 962.7504131133 & -3896.6834387285 & 5833.7250243764 & -3821.4694901055 & 924.1839076507 \\ 952.0747332782 & -3847.8501031538 & 5748.4794135456 & -3754.0992725664 & 904.1516452028 \\ 932.0424708303 & -3758.3644707492 & 5596.9901246409 & -3639.0222589178 & 871.3605505022 \end{bmatrix},$$

$$V = \begin{bmatrix} 677.0206293684 & -2862.4070325364 & 4498.6271470802 & -3110.0900487990 & 797.8493048868 \\ 677.0206293684 & -2862.4070325364 & 4498.6271470802 & -3110.0900487990 & 797.8493048868 \\ 677.0206293684 & -2862.4070325364 & 4498.6271470802 & -3110.0900487990 & 797.8493048868 \\ 677.0206293684 & -2862.4070325364 & 4498.6271470802 & -3110.0900487990 & 797.8493048868 \\ 677.0206293684 & -2862.4070325364 & 4498.6271470802 & -3110.0900487990 & 797.8493048868 \end{bmatrix}.$$

- $p = 5, \quad \lambda = 1.3012832613, \quad c = [\ -3 \quad -2 \quad -1 \quad 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} 3.2136104048 & -18.3608115765 & 28.4737820086 & -15.5951642413 & 2.7749997108 \\ 3.1872215159 & -16.5789727596 & 27.8058320806 & -15.6979420191 & 2.7902774885 \\ 3.2024992937 & -16.6817505374 & 29.7404486753 & -16.5186697248 & 2.7638885996 \\ 3.1761104048 & -16.5345283152 & 29.3737820086 & -14.3201642413 & 1.8112164495 \\ 2.2234382547 & -11.7975564532 & 19.9942827291 & -5.1601094062 & -0.7536388179 \end{bmatrix},$$

$$V = \begin{bmatrix} 1.6448050733 & -9.6436884308 & 20.2744423025 & -14.6906232993 & 3.4150643543 \\ 1.6448050733 & -9.6436884308 & 20.2744423025 & -14.6906232993 & 3.4150643543 \\ 1.6448050733 & -9.6436884308 & 20.2744423025 & -14.6906232993 & 3.4150643543 \\ 1.6448050733 & -9.6436884308 & 20.2744423025 & -14.6906232993 & 3.4150643543 \\ 1.6448050733 & -9.6436884308 & 20.2744423025 & -14.6906232993 & 3.4150643543 \end{bmatrix}.$$

3.2.6 Methods with rank 2 for V

If $\overline{V}$ is generalised from the rank 1 form to

$$\overline{V} = \begin{bmatrix} 1 & \tilde{v}_2 & \dots & \tilde{v}_{p-1} & \tilde{v}_p \\ 0 & 0 & \dots & 0 & \tilde{v}_0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \tag{3.44}$$

we find that the freedom due to the additional non-zero term, $\tilde{v}_0$, of this matrix will allow the value of λ to vary and still retain the desirable property of perfect damping at infinity and A-stability. It is found that the stability polynomial of an order p method with this generalised $\overline{V}$ matrix changes in a simple way as stated in the following theorem.

Theorem 3.7 *Consider a DIMSIM method with $p = q = r = s$ such that $A = \lambda I$, $U = I$, and $V = T\overline{V}T^{-1}$, with T given by (3.30) and $\overline{V}$ given by (3.44). Then the stability polynomial has the form*

$$\begin{aligned}\phi_2(w,z) &= (1-\lambda z)^p w^p - (\tilde{\alpha}_0 - \tilde{\beta}_1 z)(1-\lambda z)^{p-1} w^{p-1} \\ &\quad - z(\tilde{\alpha}_1 - \tilde{\beta}_2 z)(1-\lambda z)^{p-2} w^{p-2} - \cdots - z^{p-1}(\tilde{\alpha}_{p-1} - \tilde{\beta}_p z) + z^{p-2}\gamma.\end{aligned}$$

Proof To show that the only new term is of the form γz^{p-1}, we consider the characteristic polynomial of the transformed stability matrix, since this is equivalent to the characteristic polynomial of the original stability matrix. This is given by

$$\phi_2(w,z) = \det(Q) = \det\left((1-\lambda z)wI - (z\overline{B} + (1-\lambda z)\overline{V}\right),$$

with $\overline{B} = \widehat{E} - \lambda E - \widehat{V} + \lambda V$ as given by (3.43). The matrix $\widehat{E}$ is upper Hessenberg while the matrix E is upper triangular. Simplifying matrix Q we have

$$Q = (1-\lambda z)wI - z(\widehat{E} - \lambda E - \widehat{V}) - \overline{V},$$

which inherits the upper Hessenberg form, with $-z$ on the leading subdiagonal. The additional term $\tilde{v}_0$ in (3.44) causes the additional terms $z\tilde{v}_0$ and $-\tilde{v}_0$ in the last two elements of the second row of the matrix Q. We note here that with $\tilde{v}_0 = 0$ the transformed stability matrix is equivalent to the stability matrix of the rank 1 methods discussed earlier. We are interested in the form of the additional term in $\det(Q)$ which is not present in the stability polynomial of the rank 1 methods. In order to get this additional term we use the following reasoning. Consider the matrix Q' which is exactly the same as Q except that we take $\tilde{v}_0 = 0$. Since we are interested in the additional term in $\det(Q)$ that is not present in $\det(Q')$, we require $\det(Q) - \det(Q')$. Since determinants are linear functions of the rows in

the underlying matrix, the only elements in Q which contribute a non-zero term to this difference are those shown as non-zero in the matrix

$$\begin{bmatrix} (1-\lambda z)w-1-z+\lambda z+z\tilde{v}_2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & z\tilde{v}_0 & -\tilde{v}_0 \\ 0 & -z & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -z & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -z & (1-\lambda z)w-z+\lambda z \end{bmatrix}$$

and the determinant of this is

$$\left((1-\lambda z)w-1-z+\lambda z+z\tilde{v}_2\right)\left((1-\lambda z)w-1-z+\lambda z\right)z^{p-2}\tilde{v}_0,$$

of which only the term, $\tilde{v}_o z^{p-2}$, is of a different form than the terms already in (3.20). □

Since we wish to retain the property of perfect stability at infinity, that is $\rho(M(\infty)) = 0$, we need to specialise the choice of the matrix $\overline{V}$, on which the method is built, so that each of the $\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_p$ is zero. The next theorem shows that the rank 2 choice of the $\overline{V}$ matrix affects the formulae for $\phi_2(w,z)$ only for the last two of the $\tilde{\alpha}$ coefficients. Furthermore, the formulae for the first row of the $\overline{V}$ matrix is changed in only its last three elements.

Theorem 3.8 *Under the assumptions of Theorem 3.7, the stability function of a method with* $\tilde{\beta}_1 = \tilde{\beta}_2 = \cdots = \tilde{\beta}_p = 0$ *is given by*

$$\begin{aligned} \phi_2(w,z) &= (1-\lambda z)^p w^p - \tilde{\alpha}_0(1-\lambda z)^{p-1}w^{p-1} \\ &\quad - \tilde{\alpha}_1 z(1-\lambda z)^{p-2}w^{p-2} - \cdots - \tilde{\alpha}_{p-1}z^{p-1} + z^{p-2}\gamma, \end{aligned} \tag{3.45}$$

where

$$\tilde{\alpha}_i = \begin{cases} \alpha_i, & i \le p-3, \\ \alpha_{p-2} - \frac{\alpha_p}{\lambda - \frac{1}{2}}, & i = p-2, \\ \alpha_{p-1} - \frac{(\lambda-1)\alpha_p}{\lambda-\frac{1}{2}}, & i = p-1, \end{cases}$$

$$\gamma = -\frac{\alpha_p}{\lambda - \frac{1}{2}},$$

and α_i, $i = 1, 2, \ldots, p$ are the rank 1 coefficients given by (3.28). Furthermore, the values of $\tilde{v}_2, \tilde{v}_3, \ldots, \tilde{v}_p$ and $\tilde{v}_0$ in (3.44) required to produce the method of this type, are given by

$$\begin{aligned}
\tilde{v}_i &= \begin{cases} v_i, & 2 \leq i \leq -2, \\ v_{p-1} + \frac{v_{p+1}}{\lambda - \frac{1}{2}}, & i = p-1, \\ v_3 - \frac{3(\lambda-1)v_4}{\lambda - \frac{1}{2}} + \left(\frac{v_4}{\lambda - 1}\right)^2, & i = p = 3, \\ v_p - \frac{p(\lambda-1)v_{p+1}}{\lambda - \frac{1}{2}}, & i = p > 3, \end{cases} \\
\tilde{v}_0 &= -\frac{v_{p+1}}{\lambda - \frac{1}{2}},
\end{aligned}$$

where v_i, $i = 2, 3, \ldots, p+1$, are the coefficients for the rank 1 case given by Theorem 3.6.

Proof The first part of the theorem involving $\tilde{\alpha}_i$, can be proved by comparing the formulae

$$(1 - \lambda z)\exp(pz) - \sum_{k=0}^{p} \alpha_k z^k (1 - \lambda z)^{p-k-1} \exp\left((p-k-1)z\right) = O\left(z^{p+1}\right),$$

which results from Theorem 3.4, and

$$(1 - \lambda z)\exp(pz) - \sum_{k=0}^{p-1} \tilde{\alpha}_k z^k (1 - \lambda z)^{p-k-1} \exp\left((p-k-1)z\right) + z^{p-1}\gamma = O\left(z^{p+1}\right),$$

which is the equivalent of $\phi_2(\exp(z), z) = O\left(z^{p+1}\right)$. Equating successive coefficients of powers of z as far as $p-3$, leads to the values given for $\tilde{\alpha}_k$ for $k \leq p-3$. The remaining values of $\tilde{\alpha}_{p-2}$, $\tilde{\alpha}_{p-1}$ and γ are obtained by comparing the rest of the powers up to z^p.

The second part of the theorem can be proved by modifying the proof for the rank 1 methods. This is done by considering the effect of the additional term, $\tilde{v}_0$, in the $\overline{V}$ matrix. Since this additional term occurs in the second row, we only need to consider separately $\widehat{u}_p^T$ and $\widehat{u}_{p+1}^T$.

Since $\widehat{u}_{p-1}^T$ has initial terms 0 and 1, we have

$$\begin{aligned}\widehat{u}_p^T &= \widehat{u}_{p-1}^T\widehat{M}\\ &= \widehat{u}_{p-1}^T(\widehat{E}-\lambda E)-\widehat{u}_{p-1}^T\overline{V}\\ &= u_p^T-\tilde{v}_0e_{p-1}^T.\end{aligned}$$

Using this result, we further consider

$$\begin{aligned}\widehat{u}_{p+1}^T &= \widehat{u}_p^T\widehat{M}\\ &= (u_p^T-\tilde{v}_0e_{p-1}^T)(\widehat{E}-\lambda E-\overline{V})\\ &= u_p^T(\widehat{E}-\lambda E)-\tilde{v}_0e_{p-1}^T(\widehat{E}-\lambda E)-u_p^T\overline{V}+\tilde{v}_0e_{p-1}^T\overline{V}. \qquad (3.46)\end{aligned}$$

The individual terms in the last expression can be simplified as follows.

$$u_p^T\overline{V} = [1,t,0,\ldots,0]\overline{V} = [\tilde{v}_2,\tilde{v}_3,\ldots,\tilde{v}_p,0]+[0,0,\ldots,0,t\tilde{v}_0,0],$$

$$\tilde{v}_0u_p^T(\widehat{E}-\lambda E) = [0,0,\ldots,0,t\tilde{v}_0,0](\widehat{E}-\lambda E)=\tilde{v}_0[0,0,\ldots,0,1,1-\lambda,\frac{1}{2}-\lambda].$$

We note that, $\tilde{v}_0e_{p-1}^T\overline{V}=0$, except when $p=3$, which we analyse later. We also note that the vector u_p^T has the first component equal to one and the second component

$$t=\frac{(-1)^p(p-1)}{p}\lambda L_p^{(p-1)}\left(\frac{p}{\lambda}\right)=(p-1)(1-\lambda).$$

We also have

$$u_p^T(\widehat{E}-\lambda E)=u_{p+1}^T=[v_2,v_3,\ldots,v_{p+1}].$$

Substituting these simplified expressions into (3.46), we get

$$\begin{aligned}\widehat{u}_{p+1}^T &= [v_2,v_3,\ldots,v_{p+1}]-[\tilde{v}_2,\tilde{v}_3,\ldots,\tilde{v}_p,0]\\ &\quad-\tilde{v}_0[0,0,\ldots,0,1,1-\lambda,\frac{1}{2}-\lambda]-[0,0,\ldots,0,t\tilde{v}_0,0].\end{aligned}$$

As for rank 1 methods, we require that $\widehat{u}_{p+1}^T=0$, in order to satisfy $\rho(M(\infty))=0$. Thus, equating the final term in the resulting vector to zero, we get

$$\tilde{v}_0=-\frac{v_{p+1}}{\lambda-\frac{1}{2}}.$$

By equating each of the other terms to zero, and substituting the expression for $\tilde{v}_0$, we get the expressions for $\tilde{v}_i$, as in the theorem. For the case when $r = 3$, we have

$$\tilde{v}_0 e_2^T \overline{V} = [0, \tilde{v}_0^2, 0],$$

and

$$\widehat{u}_4^T = u_3^T(\widehat{E} - \lambda E) - \tilde{v}_0 e_2^T(\widehat{E} - \lambda E) - u_3^T \overline{V} + \tilde{v}_0 e_2^T \overline{V} = 0,$$

which simplifies to

$$[v_2, v_3, v_4] - [\tilde{v}_2, \tilde{v}_3, 0] - [0, t\tilde{v}_0, 0] - \tilde{v}_0[1, 1 - \lambda, \frac{1}{2} - \lambda] + [0, \tilde{v}_0^2, 0] = 0,$$

where $t = 2 - 2\lambda$. From the third term we get

$$v_0 = -\frac{v_4}{\lambda - \frac{1}{2}}$$

and using this we get

$$\tilde{v}_3 = v_3 - \frac{3(\lambda - 1)v_4}{\lambda - \frac{1}{2}} + \left(\frac{v_4}{\lambda - 1}\right)^2,$$

as required. □

The generalisation made possible by Theorems 3.8 and 3.7 allows some freedom in the values of λ, for which A-stability is possible. This has been investigated in [24] by using the numerical technique discussed in Section 3.2.1. The only change is that (3.29) is replaced by

$$\phi_2(W, iy) = -iW^p + W^{p-1} + \tilde{\alpha}_1 y W^{p-2} + \tilde{\alpha}_2 y^2 W^{p-3} + \cdots + \tilde{\alpha}_{p-1} y^{p-1} + i\gamma y^{p-2}.$$

It is found that λ now falls in an interval for an A-stable method, as given in Table 3.3.

3.2.7 Some rank 2 methods

Some rank 2 methods for the smallest values of λ are listed below. It is seen that, compared to the rank 1 methods with the same abscissae, the rank 2 methods have smaller method coefficients. Of the two types of abscissae considered, the

p	λ interval (3 dp)
3	$0.810 \leq \lambda \leq 2.137$
4	$1.081 \leq \lambda \leq 3.354$
5	$0.977 \leq \lambda \leq 1.867$
6	$1.280 \leq \lambda \leq 2.529$
7	$1.150 \leq \lambda \leq 1.766$
8	$1.415 \leq \lambda \leq 2.225$

Table 3.3: Intervals of λ for A-stability of rank 2 methods.

methods with the abscissae which lie in $[0, 1]$ have larger coefficients. This was also observed to be the case with rank 1 methods.

- $p = 3, \quad \lambda = 0.81, \quad c = [\ 0 \quad \frac{1}{2} \quad 1\ ]^T,$

$$B = \begin{bmatrix} 0.0718823104 & 2.5821330578 & -1.4073239704 \\ 2.2616014072 & 0.0793862621 & -0.5942962715 \\ 4.3913205040 & -2.8033605336 & 0.6587314275 \end{bmatrix},$$

$$V = \begin{bmatrix} 0.1658238304 & 2.1617351349 & -1.3275589653 \\ 2.7992066261 & -3.1050304565 & 1.3058238304 \\ 5.4325894218 & -8.3717960479 & 3.9392066261 \end{bmatrix}.$$

- $p = 3, \quad \lambda = 0.81, \quad c = [\ -1 \quad 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} -0.4620584115 & 1.5970656623 & -0.3883158530 \\ 1.1794739864 & 0.9406922645 & -0.3734748530 \\ 2.5110063842 & -0.0956811334 & 0.3313661470 \end{bmatrix},$$

$$V = \begin{bmatrix} -0.3052168919 & 2.3571251816 & -1.0519082897 \\ 1.0114745060 & -0.2762576141 & 0.2647831081 \\ 2.3281659038 & -2.9096404098 & 1.5814745060 \end{bmatrix}.$$

- $p = 4, \quad \lambda = 1.081, c = [\ 0 \quad \frac{1}{3} \quad \frac{2}{3} \quad 1\ ]^T,$

$$B = \begin{bmatrix} 3.0648114293 & -13.4012626618 & 19.3782582549 & -8.2178070224 \\ 16.2918355854 & -50.7460534664 & 55.3011007291 & -19.6895495148 \\ 31.0419708527 & -93.2411776044 & 96.6219432034 & -32.9320697850 \\ 47.5628838976 & -141.2963017423 & 143.4171190110 & -47.8597011663 \end{bmatrix},$$

$$V = \begin{bmatrix} 2.3411483016 & -10.9656392470 & 17.3798335892 & -7.7553426438 \\ 12.1763266260 & -40.4711742203 & 46.8853685625 & -17.5905209682 \\ 22.0115049504 & -69.9767091935 & 76.3909035358 & -27.4256992927 \\ 31.8466832748 & -99.4822441668 & 105.8964385090 & -37.2608776171 \end{bmatrix}.$$

- $p = 4, \quad \lambda = 1.081, c = [\ -2 \quad -1 \quad 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} -1.7219214320 & 2.9466155264 & -1.7092258319 & 0.3085317375 \\ -0.0358751063 & 0.4789408077 & 0.7489846286 & -0.3680503299 \\ 1.7335045526 & -3.8197339110 & 5.6191950890 & -1.7089657306 \\ 4.1672175450 & -10.6924086297 & 12.6444055494 & -3.2952144647 \end{bmatrix},$$

$$V = \begin{bmatrix} -1.1215770746 & 3.6612281487 & -2.1337250736 & 0.5940739995 \\ -0.0287794830 & 0.3828353739 & 1.1446677012 & -0.4987235921 \\ 1.0640181086 & -2.8955574009 & 4.4230604760 & -1.5915211837 \\ 2.1568157002 & -6.1739501757 & 7.7014532508 & -2.6843187753 \end{bmatrix}.$$

- $p = 5, \quad \lambda = 0.977, \quad c = [\ 0 \quad \frac{1}{4} \quad \frac{1}{2} \quad \frac{3}{4} \quad 1\ ]^T,$

$$B = \begin{bmatrix} 489.1575241628 & -1913.2374040021 & 2756.3775649325 & -1735.4467745097 & 403.5340894165 \\ 374.1462055737 & -1478.1985056367 & 2140.4559480370 & -1347.8229614589 & 312.0543134848 \\ 256.7768036514 & -1032.2524406046 & 1505.3593311415 & -945.3883150747 & 216.3896208865 \\ 135.2224850624 & -567.7398755725 & 839.3207142460 & -520.2795020239 & 114.6111782881 \\ 7.5544164733 & -76.8434772070 & 130.7110973505 & -64.9751889731 & 4.9381523564 \end{bmatrix},$$

$$V = \begin{bmatrix} 443.0067808965 & -1833.7309477621 & 2813.2687979070 & -1895.8318761140 & 474.2872450725 \\ 339.4482769339 & -1419.4969319116 & 2191.9177741313 & -1481.5978602635 & 370.7287411099 \\ 235.8897729713 & -1005.2629160612 & 1570.5667503557 & -1067.3638444131 & 267.1702371473 \\ 132.3312690087 & -591.0289002107 & 949.2157265800 & -653.1298285626 & 163.6117331846 \\ 28.7727650461 & -176.7948843603 & 327.8647028043 & -238.8958127122 & 60.0532292220 \end{bmatrix}.$$

- $p = 5, \quad \lambda = 0.977, \quad c = [\ -3 \quad -2 \quad -1 \quad 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} 5.5740198561 & -23.0810409509 & 27.0341335850 & -12.5932887418 & 1.9511762515 \\ 3.1576948679 & -13.6818426227 & 17.2051551960 & -7.9906269958 & 1.1946195545 \\ 0.7830365464 & -5.8429776278 & 9.3301768070 & -3.7816319164 & 0.3963961907 \\ -1.6332884417 & 2.2458873670 & -0.5218015820 & 2.7980298296 & -1.0038271730 \\ -4.6516134299 & 13.3030856952 & -16.1437799709 & 13.4206915756 & -3.0433838701 \end{bmatrix},$$

$$V = \begin{bmatrix} 3.6808738610 & -16.9314470716 & 27.8838880491 & -17.8119303271 & 4.1786154887 \\ 2.0627722366 & -10.4590405740 & 18.1752783026 & -11.3395238295 & 2.5605138643 \\ 0.4446706121 & -3.9866340763 & 8.4666685561 & -4.8671173318 & 0.9424122399 \\ -1.1734310123 & 2.4857724214 & -1.2419411904 & 1.6052891659 & -0.6756893845 \\ -2.7915326367 & 8.9581789190 & -10.9505509369 & 8.0776956635 & -2.2937910089 \end{bmatrix}.$$

3.2.8 Error constants

An important property of any numerical method, that has an effect on the efficiency, is the magnitude of the error constant. For type 4 DIMSIMs with

$p = q = r = s$, $U = I$ and $A = \lambda I$, analysed in this chapter, the error constants are independent of the abscissae and depend on the rank of the V matrix. The values of these error constants, C_{p+1}, can be derived by considering the relationship

$$\phi_1(\exp(z), z) = C_{p+1} z^{p+1} + O(z^{p+2}).$$

For the rank 1 methods, using (3.26), and considering the terms of $O\left(z^{p+1}\right)$ from (3.21) of Theorem 3.4, we have on rearrangement,

$$\begin{aligned}
&(1-\lambda z)^p \quad \exp(pz) - (1-\lambda z)^{p-1} \quad \exp((p-1)z) \\
&\quad - \alpha_1 z(1-\lambda z)^{p-2} \quad \exp((p-2)z) - \cdots - \alpha_{p-1} z^{p-1} \\
&\qquad = \frac{\alpha_p z^p}{(1-\lambda z) \quad \exp(z)} + \frac{\alpha_{p+1} z^{p+1}}{(1-\lambda z)^2 \quad \exp(2z)} + O(z^{p+2}) \\
&\qquad = \alpha_p (1 + \lambda z - z) z^p + \alpha_{p+1} z^{p+1} + O(z^{p+2}) \\
&\qquad = \alpha_{p+1} z^{p+1} + O(z^{p+2}).
\end{aligned}$$

For rank 1 methods since λ is chosen to satisfy (3.27), we have $\alpha_p = 0$. Thus, the error constant for a rank 1 method of order p, is given by

$$C_{p+1} = \alpha_{p+1} = (-1)^{p+2} \frac{\lambda^{p+1}}{p+2} L'_{p+2}\left(\frac{p+2}{\lambda}\right). \tag{3.47}$$

The numerical values of these error constants for the A-stable methods are given in Table 3.4. There are two methods of order 2 and one of them has a much smaller error constant. Since the value of λ is fixed for the rank 1 methods, the error constants are also fixed.

p	1	2	2	3	4	5	6	7	8	10
$\lvert C_{p+1}\rvert$	0.5	0.25	4.08	1.03	8.39	2.78	22.98	8.56	71.55	240.06

Table 3.4: Error constants for methods with $s = p$ and V of rank 1.

For methods where the V matrix has rank 2, the derivation of the error constants is slightly more complicated and can be considered by examining the stability

polynomial for the rank 2 methods and considering

$$\phi_2(\exp(z), z) = C_{p+1} z^{p+1} + O\left(z^{p+2}\right),$$

where C_{p+1} is the error constant for the rank 2 method. By considering the relationship between $\tilde{\alpha}_i$ and α_i, we obtain

$$C_{p+1} = \alpha_p \left(\frac{\frac{1}{6} - \frac{\lambda}{2}}{\lambda - \frac{1}{2}} + \lambda - 1 \right) + \alpha_{p+1}.$$

Since the α_i are polynomials in λ, these error constants are also polynomials in λ. For rank 2 methods, since λ can be varied, the size of these error constants can be varied. This relationship can be examined by plotting λ against C_{p+1} for λ in the A-stability interval, as in Figure 3.3.

For methods where the V matrix has rank 2, the numerical values of these constants for the smallest allowed value of λ, are given in Table 3.5. It can be seen from Figure 3.3 that it is possible to reduce the error constants for methods of orders 4 and 6 by choosing slightly larger values of λ. For a method with order 4, if we choose $\lambda = 1.2255$, then we get $|C_5| = 0.0095$ and for a method with order 6, with $\lambda = 1.3125$, we get $|C_7| = 0.0032$. The alternative method of order 4 is given below.

- $p = 4, \quad \lambda = 1.2255$, $c = [\ 0 \quad \frac{1}{3} \quad \frac{2}{3} \quad 1\ ]^T$,

$$B = \begin{bmatrix} 6.7251687979 & -19.0227304118 & 23.6475693020 & -9.9480076881 \\ 26.3689651403 & -74.6008176224 & 76.9311880295 & -26.9640022140 \\ 47.8248725938 & -136.3407381665 & 136.7688067569 & -46.1842745176 \\ 71.4850578251 & -205.0856587105 & 203.6702588177 & -67.6676579323 \end{bmatrix},$$

$$V = \begin{bmatrix} 4.7841007690 & -14.5858166906 & 19.0253310743 & -8.2236151526 \\ 18.1038395520 & -54.5450330396 & 58.9845474233 & -21.5433539357 \\ 31.4235783350 & -94.5042493887 & 98.9437637723 & -34.8630927187 \\ 44.7433171180 & -134.4634657377 & 138.9029801214 & -48.1828315017 \end{bmatrix}.$$

- $p = 4, \quad \lambda = 1.2255$, $c = [\ -2 \quad -1 \quad 0 \quad 1\ ]^T$,

$$B = \begin{bmatrix} -2.1643510397 & 5.3620827796 & -3.4435556307 & 0.6478238909 \\ 0.3476722125 & 0.9481506654 & 0.5737388409 & -0.4675617188 \\ 2.9430287981 & -5.4412814487 & 7.2920333125 & -2.3917806618 \\ 6.3472187170 & -14.9827135628 & 16.8878277840 & -4.8503329382 \end{bmatrix},$$

$$V = \begin{bmatrix} -1.2595048135 & 4.8724202496 & -3.5643260589 & 0.9514106227 \\ 0.2204661624 & 0.4325073220 & 0.8755868688 & -0.5285603532 \\ 1.7004371383 & -4.0074056057 & 5.3154997965 & -2.0085313291 \\ 3.1804081142 & -8.4473185334 & 9.7554127242 & -3.4885023050 \end{bmatrix}.$$

The magnitude of the coefficients of this method is slightly larger than is the case for $\lambda = 1.081$ but they are unlikely to cause any serious difficulties during computations. For methods of orders 3, 5, 7 and 8 it is possible to reduce the error constants but the values of λ required will be towards the top end of the interval, as in Table 3.3. Choosing larger values of λ is not such a good idea as this leads to large values of the method coefficients. In our investigations with these methods it has been observed that the product, λh, has some effect on Newton iterations as the stepsize, h, increases. This is another reason for choosing smaller values of λ.

p	3	4	5	6	7	8
$\lvert C_{p+1}\rvert$	0.24	0.45	0.33	0.53	2.21	5.01

Table 3.5: Error constants for methods with $s = p$, V of rank 2 and the smallest λ in the stability interval.

The magnitude of the error constant for the Radau IIA method (2.19) is $\frac{1}{7200}$ while those for the BDF methods of orders 1 to 5 are $\frac{1}{2}$, $\frac{2}{9}$, $\frac{3}{22}$, $\frac{12}{125}$, $\frac{10}{137}$, respectively [62]. Thus, the magitudes of the error constants for the type 4 DIMSIMs discussed in this chapter are quite large. It is likely that these large error constants will have an effect on the efficiency of the methods.

In Chapter 4 we develop a variable stepsize, variable order implementation of the type 4 methods discussed in this chapter, as solvers of stiff ordinary differential equations.

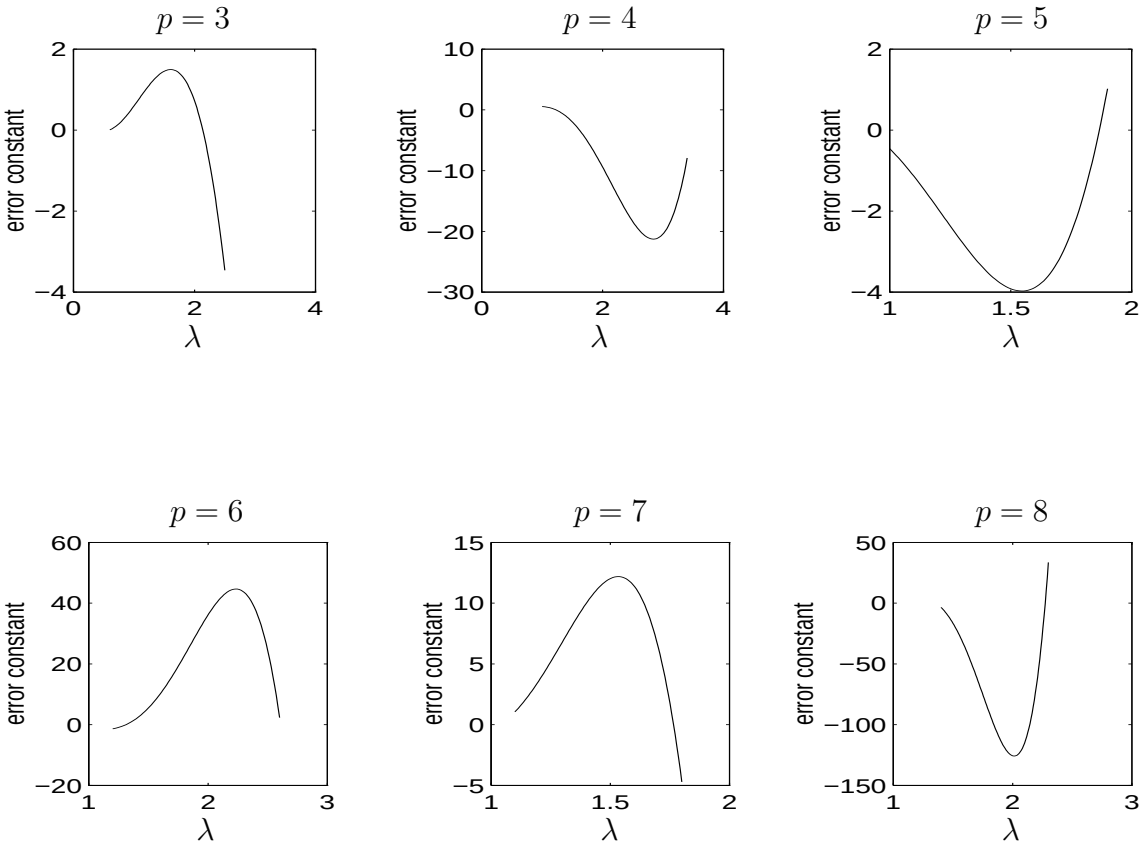

Figure 3.3: Error constants as functions of λ for rank 2 methods with λ in the A–stability interval.

Chapter 4

An implementation of type 4 DIMSIMs with $s = p$

In this chapter we consider an implementation of type 4 DIMSIMs in which the number of stages is the same as the order of the method. In this implementation, the methods which have been analysed in the last chapter are modified to give the Nordsieck vector as the output vector. This modification gives a convenient procedure for changing stepsize. In order to implement these methods in a variable stepsize, variable order code, we derive procedures for the estimation of errors and for controlling stepsize and order. We further discuss the prediction of starting values, convergence control and the stopping criteria. Finally, we discuss the performance of these methods in solving some well known stiff problems.

Since for these methods $p = q = r = s$, we will use p to denote all these quantities, for uniformity.

4.1 Fixed stepsize implementation

In order to understand some of the complications in using DIMSIMs to solve ODEs, we initially used fixed stepsize; this enabled us to carry out preliminary numerical testing in order to verify the accuracy and order of the methods. The

parameters and procedures followed in carrying these out are outlined below.

4.1.1 Starting procedure

Similar to a linear multistep method, a DIMSIM method of order p requires a starting procedure to obtain the starting values $y^{[0]}$. Consider the method given by (3.1-3.2), then from (3.1), the starting values satisfy

$$\begin{aligned} y_i^{[0]} &= Y_i - \lambda h f(Y_i) + O\left(h^{p+1}\right) \\ &= y(x_0 + c_i h) - \lambda h y'(x_0 + c_i h) + O\left(h^{p+1}\right) \\ &= y(x_0) + (c_i - \lambda) h y'(x_0) + \left(\frac{c_i^2}{2!} - \lambda c_i\right) h^2 y''(x_0) \\ &\quad + \cdots + \left(\frac{c_i^p}{p!} - \lambda \frac{c_i^{p-1}}{(p-1)!}\right) h^p y^{(p)}(x_0) + O\left(h^{p+1}\right), \end{aligned}$$

where the c_i are the abscissae for the DIMSIM method. These last equations hold for $i = 1, \ldots, s$, and together we get

$$y^{[0]} = y(x_0)e + (C - \lambda D)\bar{y}(x_0), \tag{4.1}$$

where

$$C = \begin{bmatrix} c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{p!}c_1^p \\ c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{p!}c_2^p \\ \vdots & \vdots & & \vdots \\ c_p & \frac{1}{2!}c_p^2 & \cdots & \frac{1}{p!}c_p^p \end{bmatrix}, \tag{4.2}$$

$$D = \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{(p-1)!}c_1^{p-1} \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{(p-1)!}c_2^{p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_p & \frac{1}{2!}c_p^2 & \cdots & \frac{1}{(p-1)!}c_p^{p-1} \end{bmatrix}, \tag{4.3}$$

and $\bar{y}(x_0) = [hy'(x_0), \ldots, h^p y^{(p)}(x_0)]^T$. The unknown vector, $y^{[0]}$, can be calculated using a suitable method such as a Runge-Kutta method of order and stage

order equal to p, from which we can obtain the vector $\bar{y}(x_0)$. We used the SIRK methods of the same order. Consider a SIRK method of order p with abscissae, $g = [g_1, g_2, \ldots, g_p]^T$, then we have

$$\begin{aligned} hf(Y_i) &= hy'(x_0 + g_i h) + O\left(h^{p+1}\right) \\ &= hy'(x_0) + h^2 g_i y''(x_0) + h^3 \frac{g_i^2}{2!} y'''(x_0) \\ &\quad + \cdots + h^p \frac{g_i^{p-1}}{(p-1)!} y^{(p)}(x_0) + O\left(h^{p+1}\right), \end{aligned}$$

for $i = 1, \ldots, s$, and if we let $F(Y) = [f(Y_1), f(Y_2), \ldots, f(Y_p)]^T$ then we have

$$hF(Y) = G\bar{y}(x_0),$$

where matrix G has the same structure as matrix D but c_i is replaced by g_i. Hence, we have

$$\bar{y}(x_0) = G^{-1} hF(Y),$$

and using this and (4.1) we obtain

$$y^{[0]} = y(x_0)e + (C - \lambda D)G^{-1} hF(Y). \tag{4.4}$$

This involves the use of the appropriate SIRK method as a first step, in order to obtain an estimate of $hF(Y)$, which is the vector of weighted stage values of the SIRK method. This is the starting procedure we used in our experiments involving fixed stepsizes.

4.1.2 The stage solver

From (3.1) we have

$$Y_i = \lambda h f(Y_i) + y_i^{[n]}, \qquad i = 1, 2, \ldots, p.$$

Let

$$G(Y_i) = -Y_i + \lambda h f(Y_i) + y_i^{[n]}, \qquad i = 1, 2, \ldots, p,$$

then, to solve $G(Y_i) = 0$, we use the Newton iteration scheme which can be stated as follows

$$\begin{aligned}[I - \lambda h J(Y_i^{[K]})]\Delta Y_i^{[k]} &= G(Y_i^{[k]}),\\ Y_i^{[k+1]} &= Y_i^{[k]} + \Delta Y_i^{[k]}, \qquad k = 0, 1, \ldots,\end{aligned}$$

where $Y_i^{[k]}$ refers to the k^{th} iterate of Y_i. Since the stage values, Y_i, of these DIMSIM methods satisfy

$$Y_i = y(x_n + hc_i) + O\left(h^{p+1}\right) = y(x_n) + O(h),$$

we have used $Y_i^{[0]} = y(x_n)$, $i = 1, 2, \ldots, s$. Since the aim of the fixed stepsize implementation is to verify the orders attained, we used the full Newton iteration scheme.

4.1.3 Verification of order

Using the fixed stepsize implementation, we solved the Prothero and Robinson problem

$$y'(x) = g'(x) + m(y(x) - g(x)), \qquad y(0) = g(0), \tag{4.5}$$

with the exact solution $y(x) = g(x)$, $g(x) = \sin(x)$ and $m = -10^6$, which makes the problem stiff but non-dissipative. This problem was used to investigate the phenomenon of order reduction which occurs when some Runge-Kutta methods are used to solve such problems. Using the exact solution, the global errors were calculated at the final point. The problem was integrated several times, halving the stepsize each time. A log-log plot of global errors versus the stepsize was obtained and we examined the slopes of the these plots. For methods of orders 2 to 5 the plots were linear and slopes obtained were very close to the expected order. This confirms the belief that there is no order reduction when these methods are used to solve stiff problems because these are multivalue methods and have stage order equal to the overall order of the method. For methods of orders more than five the plots obtained were not linear. For these methods the global errors

were seen to be sometimes larger for smaller stepsizes. This is possibly related to the rounding errors introduced by the large coefficients of the method. We used double precision data representation for these calculations but the results were still disappointing.

4.2 Modification for variable stepsize and order

An efficient implementation of any numerical method for for solving ODEs requires variable stepsize and possibly variable order. This enables the solver to choose the most appropriate stepsize/order to solve a given problem at a particular point. One way of implementing type 4 DIMSIMs in a variable stepsize mode is to modify the external stages vector $y^{[n]}$, which has p components, to the Nordsieck vector, $\tilde{y}^{[n]} = [y(x_n), hy'(x_n), h^2y''(x_n), \ldots, h^py^{(p)}(x_n)]$, with $p+1$ components. This is achieved by modifying the DIMSIM characterised by the matrices A, U, B and V, which are all $p \times p$ matrices, to one characterised by A, $\widetilde{U}$, $\widetilde{B}$ and $\widetilde{V}$, which are of sizes $p \times p$, $p \times (p+1)$, $(p+1) \times p$ and $(p+1) \times (p+1)$, respectively. Then changing stepsize from say h to rh, involves rescaling the the Nordsieck vector to $\tilde{y}^{[n]} = [y(x_n), rhy'(x_n), r^2h^2y''(x_n), \ldots, r^ph^py^{(p)}(x_n)]$.

It is possible to derive these methods characterised by the the matrices A, $\widetilde{U}$, $\widetilde{B}$ and $\widetilde{V}$ directly using the order conditions. However, this will not lead to any practical routine for deriving these coefficients accurately. The coefficients of the methods characterised by the A, U, B and V matrices can be easily calculated using the routines based on their derivation as outlined in the last chapter. Therefore, assuming that U, B and V coefficients are already available, we will consider deriving the matrices $\widetilde{U}$, $\widetilde{B}$ and $\widetilde{V}$.

4.2.1 Defining $\widetilde{U}$

We consider the DIMSIMs represented by (3.1-3.2) with $U = I$ and $A = \lambda I$ for the type 4 methods discussed in Chapter 3. From the first equation of (3.1), after

rearrangement, we have

$$\begin{aligned} y_i^{[n]} &= Y_i - \lambda h f(Y_i) + O\left(h^{p+1}\right) \\ &= y(x_n + c_i h) - \lambda h y'(x_n + c_i h) + O\left(h^{p+1}\right) \\ &= y(x_n) + (c_i - \lambda) h y'(x_n) + \left(\frac{c_i^2}{2!} - \lambda c_i\right) h^2 y''(x_n) \\ &\quad + \cdots + \left(\frac{c_i^p}{p!} - \lambda \frac{c_i^{p-1}}{(p-1)!}\right) h^p y^{(p)}(x_n) + O\left(h^{p+1}\right) \end{aligned}$$

for $i = 1, 2, \ldots, s$. Putting these equations together gives

$$y^{[n]} = \widetilde{U} \tilde{y}^{[n]}, \tag{4.6}$$

where $\tilde{y}^{[n]} = [y(x_n), hy'(x_n), \ldots, h^p y^{(p)}(x_n)]^T$ is the Nordsieck vector. The elements of $\widetilde{U}$ can be obtained from

$$\widetilde{U} = D - \lambda E, \tag{4.7}$$

where matrix D is redefined as

$$D = \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{p!}c_1^p \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{p!}c_2^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_p & \frac{1}{2!}c_p^2 & \cdots & \frac{1}{p!}c_p^p \end{bmatrix}, \tag{4.8}$$

and matrix E is defined as

$$E = \begin{bmatrix} 0 & 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{(p-1)!}c_1^{p-1} \\ 0 & 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{(p-1)!}c_2^{p-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & c_p & \frac{1}{2!}c_p^2 & \cdots & \frac{1}{(p-1)!}c_p^{p-1} \end{bmatrix}. \tag{4.9}$$

The method (3.3) is then modified to

$$\begin{bmatrix} Y \\ \tilde{y}^{[n+1]} \end{bmatrix} = \begin{bmatrix} A & \widetilde{U} \\ \widetilde{B} & \widetilde{V} \end{bmatrix} \begin{bmatrix} hF(Y) \\ \tilde{y}^{[n]} \end{bmatrix}, \tag{4.10}$$

or

$$Y = AhF(Y) + \widetilde{U}\tilde{y}^{[n]}, \tag{4.11}$$

$$\tilde{y}^{[n+1]} = \widetilde{B}hF(Y) + \widetilde{V}\tilde{y}^{[n]}. \tag{4.12}$$

Theorem 4.1 *When the original DIMSIM method given by (3.3) is modified to the form (4.10), where the output vector is the Nordsieck vector,* $\tilde{y}^{[n]}$*, then*

$$\widetilde{U}\widetilde{B} = B, \tag{4.13}$$

$$\widetilde{U}\widetilde{V} = V\widetilde{U}. \tag{4.14}$$

Proof Premultiply (4.12) by $\widetilde{U}$ to get

$$\widetilde{U}\tilde{y}^{[n+1]} = \widetilde{U}\widetilde{B}hF(Y) + \widetilde{U}\widetilde{V}\tilde{y}^{[n]} \tag{4.15}$$

and using (4.6) in (3.2) we get

$$\widetilde{U}\tilde{y}^{[n+1]} = BhF(Y) + V\widetilde{U}\tilde{y}^{[n]}. \tag{4.16}$$

By comparing equations (4.15) and (4.16) we obtain the result. □

We therefore need to determine the matrices $\widetilde{B}$ and $\widetilde{V}$ to satisfy equations (4.13) and (4.14). However, the choice of $\widetilde{B}$ and $\widetilde{V}$ is not unique. Since the calculation of matrices $\widetilde{B}$ and $\widetilde{V}$ depend on the rank of the V matrix, we look at these separately in the following sections.

4.2.2 Modification to rank 1 methods

4.2.2.1 Determination of $\widetilde{V}$

Since V is a rank 1 matrix, we let $V = e[v_1, v_2, \ldots, v_p]$. $\widetilde{B}$ and $\widetilde{V}$ are interrelated as they need to satisfy (4.12). With the choice of $\widetilde{B}$ given in the next section, the

$(p+1) \times (p+1)$ matrix $\widetilde{V}$ takes the form

$$\widetilde{V} = \begin{bmatrix} \tilde{v}_1 & \tilde{v}_2 & \dots & \tilde{v}_{p+1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

We now have

$$\widetilde{U}\widetilde{V} = e[\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_{p+1}],$$

and

$$V\widetilde{U} = e\left[\sum_{i=1}^{p} v_i, \sum_{i=1}^{p} v_i\tilde{u}_{i,2}, \sum_{i=1}^{p} v_i\tilde{u}_{i,3}, \cdots, \sum_{i=1}^{p} v_i\tilde{u}_{i,p+1}\right].$$

Since we have $\widetilde{U}\widetilde{V} = V\widetilde{U}$, by equating terms, we get

$$\begin{aligned} \tilde{v}_1 &= \sum_{i=1}^{p} v_i = 1, \\ \tilde{v}_j &= \sum_{i=1}^{p} v_i\tilde{u}_{i,j}, \qquad j = 2, 3, \dots, p+1. \end{aligned}$$

Hence, the matrix $\widetilde{V}$ can be easily calculated since $\widetilde{U}$ is known explicitly.

4.2.2.2 Determination of $\widetilde{B}$

The structure of matrix $\widetilde{B}$ depends on the abscissae and the rank of the V matrix. Here we consider rank 1 choice of V and two different choices of the abscissae. For the first case we have $c = [0, \dots, 1]^T$, that is $c_1 = 0$ and $c_p = 1$. Consider equation (3.2) for steps n and $n-1$. This can be written as

$$y_i^{[n]} = \sum_{j=1}^{p} b_{i,j} hf(Y_j^{[n]}) + \sum_{j=1}^{p} v_{i,j} y_j^{[n-1]}, \quad i = 1, 2, \dots, p,$$

from which, with $i = 1$, we have

$$y_1^{[n]} = \sum_{j=1}^{p} b_{1,j} hf(Y_j^{[n]}) + \sum_{j=1}^{p} v_{1,j} y_j^{[n-1]}. \tag{4.17}$$

Here the notation $Y_j^{[n]}$ refers to the stage value, Y_j, at the n^{th} step. Similarly, we can write (3.1) in the form

$$Y_i^{[n+1]} = y_i^{[n]} + \lambda h f(Y_i^{[n+1]}), \quad i = 1, 2, \ldots, p,$$

from which, with $i = 1$, we have

$$Y_1^{[n+1]} = y_1^{[n]} + \lambda h f(Y_1^{[n+1]}). \tag{4.18}$$

Furthermore, we have $Y_1^{[n+1]} \approx y(x_n + c_1 h) = y(x_n)$ since $c_1 = 0$. Using this and (4.17) and (4.18), we get

$$\begin{aligned} y(x_n) &= y_1^{[n]} + \lambda h f(Y_1^{[n+1]}) \\ &= \sum_{j=1}^{p} b_{1,j} h f(Y_j^{[n]}) + \sum_{j=1}^{p} v_{1,j} y_j^{[n-1]} + \lambda h f(Y_1^{[n+1]}). \end{aligned}$$

Since $c_1 = 0$ and $c_p = 1$ we have $Y_1^{[n+1]} = Y_p^{[n]}$ and the last equation simplifies to

$$y(x_n) = \sum_{j=1}^{p-1} b_{1,j} h f(Y_j^{[n]}) + (b_{1,p} + \lambda) h f(Y_p^{[n]}) + \sum_{j=1}^{p} v_{1,j} y_j^{[n-1]}.$$

Consequently, the first row of $\widetilde{B}$ is $[b_{11}, b_{12}, \ldots, b_{1p}+\lambda]$ and since $hy'(x_n + h) = hf(Y_p^{[n]})$, the second row is $[0, 0, \ldots, 0, 1]$. Therefore, let $\widetilde{B}$ have the form

$$\widetilde{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1,p-1} & b_{1,p} + \lambda \\ 0 & 0 & \cdots & 0 & 1 \\ \tilde{b}_{31} & \tilde{b}_{32} & \cdots & \tilde{b}_{3,p-1} & \tilde{b}_{3,p} \\ \vdots & \vdots & & \vdots & \vdots \\ \tilde{b}_{p+1,1} & \tilde{b}_{p+1,2} & \cdots & \tilde{b}_{p+1,p-1} & \tilde{b}_{p+1,p} \end{bmatrix}, \tag{4.19}$$

then, using (4.12), we require that

$$\sum_{i=1}^{p} \tilde{b}_{k,i} h y'(x_{n-1} + h c_i) = h^{k-1} y^{(k-1)}(x_n) + O\left(h^{p+1}\right), \tag{4.20}$$

for $k = 3, 4, \ldots, p+1$. By writing $x_{n-1} + c_i h = x_n + (c_i - 1)h$ and expanding the left-hand side of each of the above equations using a Taylor expansion about x_n,

and equating the coefficients, we get a system of linear equations which can be written as,

$$\sum_{i=1}^{p} \frac{\tilde{b}_{k,i}(c_i - 1)^q}{q!} = \begin{cases} 0, & q = 0, 1, \ldots, p-1, \quad q \neq k-2, \\ 1, & q = k-2, \end{cases}$$

for $k = 3, 4, \ldots, p+1$. After simplifications, this system of equations can be written as a matrix equation,

$$\begin{bmatrix} \tilde{b}_{31} & \tilde{b}_{32} & \cdots & \tilde{b}_{3,p} \\ \tilde{b}_{41} & \tilde{b}_{42} & \cdots & \tilde{b}_{4,p} \\ \vdots & \vdots & & \vdots \\ \tilde{b}_{p+1,1} & \tilde{b}_{p+1,2} & \cdots & \tilde{b}_{p+1,p} \end{bmatrix} \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{(p-1)!}c_1^{p-1} \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{(p-1)!}c_2^{p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_p & \frac{1}{2!}c_p^2 & \cdots & \frac{1}{(p-1)!}c_p^{p-1} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 1 & \frac{1}{2!} & \cdots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} \\ 0 & 0 & 1 & 1 & \cdots & \frac{1}{(p-4)!} & \frac{1}{(p-3)!} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{4.21}$$

This is a linear system which can be solved easily to get the matrix of unknowns on the left. Thus, $\widetilde{B}$ can be determined quite easily.

For the second case the abscissae is $c = [-p+2, \ldots, 0, 1]^T$. In this case $c_{p-1} = 0$ and only the first row of the $\widetilde{B}$ matrix is different. Using an argument as for the case when $c_1 = 0$, we get the first row of $\widetilde{B}$ as $[b_{p-1,1}, b_{p-1,2}, \cdots, b_{p-1,p-1}, b_{p-1,p} + \lambda]$ and the remaining elements of $\widetilde{B}$ are determined in exactly the same way as for the case $c_1 = 0$ using (4.21). Using the procedure outlined above we give the modified matrices for some methods below.

4.2.2.3 Some modified rank 1 methods

- The implicit Euler method can be considered as a type 4 DIMSIM with $p = 1, \quad \lambda = 1, \quad c = 1,$

$$B = [1], \quad V = [1],$$

$$\widetilde{B} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \widetilde{U} = [1 \quad 0], \quad \widetilde{V} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

- $p = 2, \quad \lambda = \frac{3-\sqrt{3}}{2}, \quad c = [\ 0 \quad 1\ ]^T,$

$$B = \begin{bmatrix} \frac{18-11\sqrt{3}}{4} & \frac{-12+7\sqrt{3}}{4} \\ \frac{22-13\sqrt{3}}{4} & \frac{-12+9\sqrt{3}}{4} \end{bmatrix}, \quad V = \begin{bmatrix} \frac{3-2\sqrt{3}}{2} & \frac{-1+2\sqrt{3}}{4} \\ \frac{3-2\sqrt{3}}{2} & \frac{-1+2\sqrt{3}}{4} \end{bmatrix},$$

$$\widetilde{B} = \begin{bmatrix} \frac{18-11\sqrt{3}}{4} & \frac{-6+5\sqrt{3}}{4} \\ 0 & 1 \\ -1 & 1 \end{bmatrix}, \quad \widetilde{U} = \begin{bmatrix} 1 & \frac{-3+\sqrt{3}}{2} & 0 \\ 1 & \frac{-1+\sqrt{3}}{2} & \frac{-2+\sqrt{3}}{2} \end{bmatrix},$$

$$\widetilde{V} = \begin{bmatrix} 1 & \frac{-4+3\sqrt{3}}{2} & \frac{8-5\sqrt{3}}{4} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

- $p = 3, \quad \lambda = 1.21013831273, \quad c = [\ 0 \quad \frac{1}{2} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} -6.4518297302 & 14.0277199958 & -5.2353370147 \\ 0 & 0 & 1 \\ 1 & -4 & 3 \\ 4 & -8 & 4 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -1.2101383127 & 0 & 0 \\ 1 & -0.7101383127 & -0.4800691564 & -0.1304339558 \\ 1 & -0.2101383127 & -0.7101383127 & -0.4384024897 \end{bmatrix},$$

$$\tilde{v} = \begin{bmatrix} 1 & -1.3405532509 & -1.2785229832 & 1.0308701745 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

4.2.2.4 Error estimation for stepsize control

In this section we look for procedures for estimating the local truncation error, $C_{p+1}h^{p+1}y^{(p+1)}(x_n)$, for methods of order p with the V matrix of rank 1. This error estimate is used for controlling the stepsize in a variable stepsize implementation. Details on how the error estimate is used to control stepsize will be explained in a later section.

It is possible to take various linear combination of stage derivatives of two consecutive steps, in order to get an estimate of $h^{p+1}y^{(p+1)}(x_n)$. We can either take all the past and present stage values or ignore some of these in getting these estimates. Since the components of the Nordsieck vector are calculated by a linear combination of the stage derivatives in that step, it is also possible to use these components. This is equivalent to using all the stage derivatives in each step.

Theorem 4.2 *If we use a type* 4 *DIMSIM method of rank* 1 *and order* p*, and the stepsize changes from* h/r *to* h*, then the local truncation error satisfies*

$$h^{p+1}y^{(p+1)}(x_n) = \left(\frac{rp}{p + (r-1)\sum_{j=1}^{p} c_j}\right)\left(\tilde{y}_{p+1}^{[n]} - r^p\tilde{y}_{p+1}^{[n-1]}\right) + O\left(h^{p+2}\right). \quad (4.22)$$

Proof

$$\begin{aligned}
\tilde{y}_{p+1}^{[n]} - r^p\tilde{y}_{p+1}^{[n-1]} &= \sum_{i=1}^{p}\tilde{b}_{p+1,i}hf(Y_i^{[n]}) - r^p\sum_{i=1}^{p}\tilde{b}_{p+1,i}\frac{h}{r}f(Y_i^{[n-1]}) \\
&\approx \sum_{i=1}^{p}\tilde{b}_{p+1,i}hy'(x_{n-1} + c_ih) - r^p\sum_{i=1}^{p}\tilde{b}_{p+1,i}\frac{h}{r}y'(x_{n-2} + c_i\frac{h}{r}) \\
&= h^py^{(p)}(x_n + \theta h) - r^p(\frac{h}{r})^py^{(p)}(x_{n-1} + \theta\frac{h}{r}) + O\left(h^{p+2}\right) \\
&= h^p\left[y^{(p)}(x_n + \theta h) - y^{(p)}(x_n + h(-1 + \frac{\theta}{r}))\right] + O\left(h^{p+2}\right) \\
&= \left(\theta\left(1 - \frac{1}{r}\right) + 1\right)h^{(p+1)}y^{(p+1)}(x_n) + O\left(h^{p+2}\right).
\end{aligned}$$

Considering equation (4.20) when $k = p + 1$, it is seen that θ is the coefficient of the next term in the Taylor series, that is,

$$\begin{aligned} \theta &= \sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i}(c_i - 1)^p}{p!} \\ &= \sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i}c_i^p}{p!} - 1, \end{aligned} \tag{4.23}$$

where the following relationships are satisfied

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i}c_i^q = 0, \qquad q = 0, 1, \ldots, p-2, \tag{4.24}$$

$$\sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i}c_i^{p-1}}{(p-1)!} = 1. \tag{4.25}$$

We can determine an alternate expression for θ, which does not involve $\tilde{b}_{p+1,i}$, by considering

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i} \prod_{j=1}^{p} (c_i - c_j) = 0.$$

By expanding the left-hand side and using (4.24) we get

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i}c_i^p - \sum_{j=1}^{p} c_j \sum_{i=1}^{p} \tilde{b}_{p+1,i}c_i^{p-1} = 0.$$

Therefore, we have

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i}c_i^p = \sum_{j=1}^{p} c_j \sum_{i=1}^{p} \tilde{b}_{p+1,i}c_i^{p-1},$$

and using (4.25), we get

$$\sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i}c_i^p}{p!} = \sum_{j=1}^{p} c_j \sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i}c_i^{p-1}}{p!} = \frac{\sum_{j=1}^{p} c_j}{p}.$$

Substituting this into (4.23), we get

$$\theta = \frac{\sum_{j=1}^{p} c_j}{s} - 1, \tag{4.26}$$

and

$$\theta\left(1-\frac{1}{r}\right)+1 \;=\; \frac{1}{r}+(1-\frac{1}{r})\frac{\sum_{j=1}^{p} c_j}{p}. \tag{4.27}$$

Hence, we obtain

$$\tilde{y}_{p+1}^{[n]} - r^p \tilde{y}_{p+1}^{[n-1]} = \left(\frac{1}{r}+(1-\frac{1}{r})\frac{\sum_{j=1}^{p} c_j}{p}\right) h^{p+1} y^{(p+1)}(x_n) + O\left(h^{p+2}\right).$$

from which the result of the theorem follows. □

Other linear combinations of the stage derivatives of two consecutive steps can also be used to obtain estimates of $h^{p+1}y^{(p+1)}(x_n)$.

Example 4.1 Considering the third order method with $c = [0, \frac{1}{2}, 1]^T$, we have

$$h^{p+1}y^{(p+1)}(x_n) + O(h^4) = b_1 hf(Y_1^{[n-1]}) + b_2 hf(Y_2^{[n-1]}) + b_3 hf(Y_3^{[n-1]})$$
$$+b_4 hf(Y_1^{[n]}) + b_5 hf(Y_2^{[n]}) + b_6 hf(Y_3^{[n]}), \tag{4.28}$$

where b_1 to b_6 are functions of the stepsize ratio, r, and can be chosen in various ways as in Table 4.1. □

In our numerical experiments it was found that amongst all the estimators based on the weightings from Table 4.1, the first one, which uses all the stage derivatives in two consecutive steps, gave the smoothest error estimates. Using the first set of values from Table 4.1 makes (4.28) equivalent to (4.22) with $s = 3$ and $c = [0, \frac{1}{2}, 1]^T$. Since the weightings of this estimator can be generated easily for methods of any order using (4.22), this is the estimator that is used in our experimental code.

4.2.3 Modification to rank 2 methods

4.2.3.1 Determination of $\widetilde{V}$

For rank 2 methods the form of the $\widetilde{B}$ matrix is the same as the case for rank 1 methods as given by (4.19). However, the contents of the $\widetilde{B}$ matrix are chosen in

b_1	b_2	b_3	b_4	b_5	b_6
$\frac{-8r^4}{1+r}$	$\frac{16r^4}{1+r}$	$\frac{-8r^4}{1+r}$	$\frac{8r}{1+r}$	$\frac{-16r}{1+r}$	$\frac{8r}{1+r}$
$\frac{-12r^4}{1+r}$	$\frac{48r^4}{1+2r}$	$-12r^3$	0	0	$\frac{12r^2}{(1+r)(1+2r)}$
$\frac{-12r^4}{(1+r)(2+r)}$	0	0	$12r$	$\frac{-48}{2+r}$	$\frac{12r}{1+r}$
$\frac{-24r^4}{(1+r)(2+r)}$	$\frac{48r^4}{(1+r)(1+2r)}$	0	0	$\frac{-48r^2}{2+3r+r^2}$	$\frac{24r^2}{1+3r+2r^2}$
$\frac{-24r^4}{2+r}$	$\frac{48r^4}{1+r}$	$-24r^3$	0	$\frac{48r^2}{2+3r+r^2}$	0
$\frac{-12r^4}{(1+r)(2+r)}$	0	$12r^2$	0	$\frac{-48r}{2+r}$	$\frac{12r}{1+r}$

Table 4.1: Weightings for the order 3 error estimate.

a more complicated way as detailed in the next section. With this form of matrix $\widetilde{B}$, if V has rank 2 then $\widetilde{V}$ has form

$$\widetilde{V} = \begin{bmatrix} \tilde{v}_{1,1} & \tilde{v}_{1,2} & \cdots & \tilde{v}_{1,p-1} & \tilde{v}_{1,p} & \tilde{v}_{1,p+1} \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & \tilde{v}_{3,p} & \tilde{v}_{3,p+1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \tilde{v}_{p+1,p} & \tilde{v}_{p+1,p+1} \end{bmatrix}.$$

Using equation (4.14), we need to satisfy $\widetilde{U}\widetilde{V} = V\widetilde{U}$. By multiplying the relevant matrices and equating terms we obtain the elements in the first row of the $\widetilde{V}$ matrix as

$$\tilde{v}_{1,1} = \sum_{i=1}^{p} v_{1,i} = 1,$$

$$\tilde{v}_{1,k} = \sum_{i=1}^{p} v_{1,i}\tilde{u}_{i,k}, \quad k = 2, 3, \ldots, p-1.$$

The remaining elements of $\widetilde{V}$ can be determined by solving the equation $\widehat{U}\widehat{V} = V\overline{U}$, where $\widehat{U}$, $\widehat{V}$ and $\overline{U}$ are submatrices defined as

$$\begin{aligned}
\widehat{U} &= \widetilde{U} \quad \text{with column 2 removed,} \\
\widehat{V} &= \widetilde{V} \quad \text{with columns 1,2 and row 2 removed,} \\
\overline{U} &= \widetilde{U} \quad \text{with columns 1 and 2 removed.}
\end{aligned}$$

4.2.3.2 Determination of $\widetilde{B}$

For the cases $c = [0, \ldots, 1]^T$ and $c = [-p+2, \ldots, 0, 1]^T$, the first two rows of the $\widetilde{B}$ matrix are defined in exactly the same way as that of the rank 1 case. The other elements are determined in a similar way. Since rows $3, 4, \ldots, p+1$, of $\widetilde{V}$ contain some non-zero elements, using (4.12) we need to satisfy

$$\begin{aligned}
&\sum_{i=1}^{p} \tilde{b}_{k,i} h y'(x_n + h(c_i - 1)) + \sum_{i=p}^{p+1} \tilde{v}_{k,i} h^{i-1} y^{(i-1)}(x_n - h) \\
&\qquad = h^{k-1} y^{(k-1)}(x_n) + O\left(h^{p+1}\right), \quad k = 3, 4, \ldots, p+1. \qquad (4.29)
\end{aligned}$$

By equating the coefficients in the Taylor expansion for each of the equations, we get systems of linear equations. For order p the systems of linear equations resulting from (4.29) can be written for $k = 3, 4, \ldots, p+1$, as

$$\begin{aligned}
\sum_{i=1}^{p} \tilde{b}_{k,i} \frac{(c_i - 1)^q}{q!} &= \begin{cases} 0, & q = 0, 1, \ldots, p-3, \quad q \neq k-2, \\ 1, & q = k-2, \end{cases} \\
\sum_{i=1}^{p} \frac{\tilde{b}_{k,i}(c_i - 1)^{p-2}}{(p-2)!} + \tilde{v}_{k,p} &= \begin{cases} 0, & k \neq p, \\ 1, & k = p, \end{cases} \qquad (4.30) \\
\sum_{i=1}^{p} \frac{\tilde{b}_{k,i}(c_i - 1)^{p-1}}{(p-1)!} - \tilde{v}_{k,p} + \tilde{v}_{k,p+1} &= \begin{cases} 0, & k \neq p+1, \\ 1, & k = p+1. \end{cases}
\end{aligned}$$

When these equations are simplified and put together, they satisfy the matrix equation

$$\begin{bmatrix} \tilde{b}_{31} & \tilde{b}_{32} & \cdots & \tilde{b}_{3,p} \\ \tilde{b}_{41} & \tilde{b}_{42} & \cdots & \tilde{b}_{4,p} \\ \vdots & \vdots & & \vdots \\ \tilde{b}_{p+1,1} & \tilde{b}_{p+1,2} & \cdots & \tilde{b}_{p+1,p} \end{bmatrix} \begin{bmatrix} 1 & c_1 & \frac{1}{2!}c_1^2 & \cdots & \frac{1}{(p-1)!}c_1^{p-1} \\ 1 & c_2 & \frac{1}{2!}c_2^2 & \cdots & \frac{1}{(p-1)!}c_2^{p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_p & \frac{1}{2!}c_p^2 & \cdots & \frac{1}{(p-1)!}c_p^{p-1} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 1 & \frac{1}{2!} & \cdots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} \\ 0 & 0 & 1 & 1 & \cdots & \frac{1}{(p-4)!} & \frac{1}{(p-3)!} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & \cdots & 0 & v_{3,p} & v_{3,p+1} \\ 0 & \cdots & 0 & v_{4,p} & v_{4,p+1} \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & v_{p,p} & v_{p,p+1} \\ 0 & \cdots & 0 & v_{p+1,p} & v_{p+1,p+1} \end{bmatrix}. \quad (4.31)$$

Comparing this with the corresponding matrix equation for the rank 1 case, (4.21), we see that the matrix being subtracted here is the $\widetilde{V}$ matrix for the rank 2 case, with its first two rows and the first column removed. Since the right hand side is known, this matrix equation can be solved by a linear equation solver to get the unknown elements in $\widetilde{B}$.

4.2.3.3 Some modified rank 2 methods

- $p = 3, \quad \lambda = 0.81, \quad c = [\ 0 \quad \frac{1}{2} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} 0.0718823104 & 2.5821330578 & -0.5973239704 \\ 0 & 0 & 1 \\ -4.1749003488 & 3.9088099671 & 0.2660903817 \\ -12.6932269318 & 17.5122902165 & -4.8190632847 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -0.81 & 0 & 0 \\ 1 & -0.31 & -0.28 & -0.0804166667 \\ 1 & 0.19 & -0.31 & -0.2383333333 \end{bmatrix},$$

$$\widetilde{V} = \begin{bmatrix} 1 & -1.0566913978 & -0.1937425585 & 0.1425620196 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1.2204953653 & 0.3783535632 \end{bmatrix}.$$

- $p = 4, \quad \lambda = 1.081, c = [\ 0 \quad \frac{1}{3} \quad \frac{2}{3} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} 3.0648114293 & -13.4012626618 & 19.3782582549 & -7.1368070224 \\ 0 & 0 & 0 & 1 \\ -35.4590194154 & 98.6708013894 & -93.9645445325 & 30.7527625585 \\ -36.4288776113 & 110.9585864510 & -112.6305400681 & 38.1008312284 \\ -74.2097721365 & 210.0164999157 & -197.4036834218 & 61.5969556427 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -1.0810000000 & 0 & 0 & 0 \\ 1 & -0.7476666667 & -0.3047777778 & -0.0538827160 & -0.0061584362 \\ 1 & -0.4143333333 & -0.4984444444 & -0.1908395062 & -0.0451522634 \\ 1 & -0.0810000000 & -0.5810000000 & -0.3738333333 & -0.1385000000 \end{bmatrix},$$

$$\widetilde{V} = \begin{bmatrix} 1 & -0.9050000000 & -0.8149442602 & 0.1733051580 & 0.3569073224 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0229174285 & -0.5943150260 \\ 0 & 0 & 0 & 0.8142273759 & -0.4730661054 \end{bmatrix}.$$

4.2.3.4 Error estimation for stepsize control

There are a few different ways in which error estimators can be chosen for methods where the matrix V has rank 2. The following method is equivalent to (4.22) for rank 1 methods.

Theorem 4.3 *If we use a type* 4 *DIMSIM method of rank* 2 *and order* p*, and the stepsize changes from* h/r *to* h*, then the local truncation error satisfies:*

$$h^{p+1} y^{(p+1)}(x_n) = \frac{1}{\xi}(\tilde{y}_{p+1}^{[n]} - r^p \tilde{y}_{p+1}^{[n-1]}) + O\left(h^{p+2}\right), \tag{4.32}$$

where

$$\xi = \left(1 - \frac{1}{r}\right)\left(\left(\sum_{j=1}^{p} c_j\right)\frac{(1 - \tilde{v}_{p+1,p+1})}{p} - \left(\sum_{i=1}^{p-1}\prod_{j=i+1}^{p} c_i c_j\right)\frac{\tilde{v}_{p+1,p}}{p(p-1)}\right) + \frac{1}{r}.$$

Proof As for rank 1 methods, we have

$$\tilde{y}_{p+1}^{[n]} - r^p \tilde{y}_{p+1}^{[n-1]} = \left(\theta\left(1 - \frac{1}{r}\right) + 1\right) h^{p+1} y^{(p+1)}(x_n) + O\left(h^{p+2}\right), \tag{4.33}$$

where the expression for θ can be obtained by considering (4.29) when $k = p+1$. The coefficient of the next term in the Taylor expansion about x_n, of the right hand side of (4.29), is

$$\theta = \sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i}(c_i - 1)^p}{p!} + \frac{\tilde{v}_{p+1,p}}{2} - \tilde{v}_{p+1,p+1}, \tag{4.34}$$

where, the coefficients satisfy the conditions

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i} c_i^q = 0, \qquad q = 0, 1, \ldots, p-3, \tag{4.35}$$

$$\sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i} c_i^{p-2}}{(p-2)!} + \tilde{v}_{p+1,p} = 0, \tag{4.36}$$

$$\sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i} c_i^{p-1}}{(p-1)!} + \tilde{v}_{p+1,p+1} = 1. \tag{4.37}$$

By expanding the first term of (4.34), and simplifying it using (4.35)-(4.37), it can be shown that

$$\theta = \sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i} c_i^p}{p!} - 1. \tag{4.38}$$

As for rank 1 methods, we can look for an equivalent expression for θ, by considering

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i} \prod_{j=1}^{p} (c_i - c_j) = 0.$$

Expand the left-hand side and use (4.35) to get

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i} c_i^p - \left(\sum_{j=1}^{p} c_j \right) \sum_{i=1}^{p} \tilde{b}_{p+1,i} c_i^{p-1} + \left(\sum_{i=1}^{p-1} \prod_{j=i+1}^{p} c_i c_j \right) \sum_{i=1}^{p} \tilde{b}_{p+1,i} c_i^{p-2} = 0.$$

On rearrangement of this last equation and using (4.36-4.37), we have

$$\sum_{i=1}^{p} \tilde{b}_{p+1,i} c_i^p = \left(\sum_{j=1}^{p} c_j \right) (1 - \tilde{v}_{p+1,p+1})(p-1)! - \left(\sum_{i=1}^{p-1} \prod_{j=i+1}^{p} c_i c_j \right) \tilde{v}_{p+1,p}(p-2)!$$

or

$$\sum_{i=1}^{p} \frac{\tilde{b}_{p+1,i} c_i^p}{p!} = \left(\sum_{j=1}^{p} c_j \right) \frac{(1 - \tilde{v}_{p+1,p+1})}{p} - \left(\sum_{i=1}^{p-1} \prod_{j=i+1}^{p} c_i c_j \right) \frac{\tilde{v}_{p+1,p}}{p(p-1)}.$$

Substituting this expression into (4.38), we obtain

$$\theta = \left(\sum_{j=1}^{p} c_j \right) \frac{(1 - \tilde{v}_{p+1,p+1})}{p} - \left(\sum_{i=1}^{p-1} \prod_{j=i+1}^{p} c_i c_j \right) \frac{\tilde{v}_{p+1,p}}{p(p-1)} - 1 \tag{4.39}$$

and it follows that

$$\begin{aligned} \xi &= \theta \left(1 - \frac{1}{r} \right) + 1 \\ &= \left(1 - \frac{1}{r} \right) \left(\left(\sum_{j=1}^{p} c_j \right) \frac{(1 - \tilde{v}_{p+1,p+1})}{p} - \left(\sum_{i=1}^{p-1} \prod_{j=i+1}^{p} c_i c_j \right) \frac{\tilde{v}_{p+1,p}}{p(p-1)} \right) + \frac{1}{r}, \end{aligned}$$

completing the proof. □

Other error estimators, which take linear combinations of the stage derivatives of two consecutive steps, can also be used. It can be shown that one such estimator is

$$h^{p+1} y^{(p+1)}(x_n) = \eta \left(\sum_{i=1}^{p} \hat{b}_i h f(Y_i^{[n]}) - r^p \sum_{i=1}^{p} \hat{b}_i \frac{h}{r} f(Y_i^{[n-1]}) \right) + O\left(h^{p+2}\right), \tag{4.40}$$

where

$$\eta = \left(\frac{rp}{p + (r-1) \sum_{j=1}^{p} c_j} \right).$$

$hf(Y_i^{(n)})$ and $hf(Y_i^{(n-1)})$, $i = 1, 2, \ldots, p$, are the internal stages derivatives at step n and $n-1$ respectively, from the rank 2 method and $\hat{b}_i$, $i = 1, 2, \ldots, p$, are the weightings which are numerically equal to the last row of the $\widetilde{B}$ matrix of the corresponding rank 1 method, that is,

$$\hat{b}_i = \tilde{b}_{p+1,i} \quad \text{of the corresponding rank 1 method.}$$

The error estimators (4.32) and (4.40) were used in the implementation of the rank 2 methods. It was found that (4.40) gave a smoother behaviour and this is the one that is used in our experimentation of the rank 2 methods.

4.2.4 Nordsieck vector update for change of order

When order is changed we need to modify the external stage vector. Decreasing order is very straightforward since we simply need to remove the last component of the Nordsieck vector, $h^{p+1}y^{(p+1)}(x_n)$.

When using a method of order p, we have

$$\tilde{y}^{[n]} = [y(x_n), hy'(x_n), \ldots, h^p y^{(p)}(x_n)]^T.$$

In order to use a method of order $p+1$, we need

$$\tilde{y}^{[n]} = [y(x_n), hy'(x_n), \ldots, h^p y^{(p)}(x_n), h^{p+1}y^{(p+1)}(x_n)]^T.$$

The additional term, $h^{p+1}y^{(p+1)}(x_n)$, is readily available from the local error estimate in the last step of the method of order p. The other terms can be directly used but these terms are accurate to order p only. The accuracy of these terms can be improved by considering the next term in the Taylor series expansion. Expanding (4.20) and (4.29) using a Taylor series about x_n and retaining the terms containing h^{p+1}, we get

$$\tilde{y}_k = h^k y^k(x_n) + \phi_k h^{p+1} y^{(p+1)} + O\left(h^{p+2}\right),$$

where

$$\phi_k = \sum_{i=1}^{p} \frac{\tilde{b}_{k+1,i}(c_i - 1)^p}{p!}, \quad \text{for rank 1 methods,}$$

$$\phi_k = \sum_{i=1}^{p} \frac{\tilde{b}_{k+1,i}(c_i - 1)^p}{p!} + \frac{\tilde{v}_{k+1,s}}{2} - \tilde{v}_{k+1,p+1}, \quad \text{for rank 2 methods,}$$

for $k = 2, 3, \ldots, p$. It should be noted here that, $\phi_p = \theta$, where θ is defined by (4.26) for rank 1 methods and by (4.39) for rank 2 methods.

The additional terms, which can be easily calculated using the local error estimate and the method coefficients, can be subtracted from the respective components to increase the accuracy of these terms in the Nordsieck vector to $p+1$. We investigated the effect of these corrections on the performance of the methods.

These additional terms were very small and did not make any difference to the overall performance of the method in terms of the number of accepted/rejected steps and the global error in the final computed solution of the problems tested.

4.2.5 Error estimates for variable order

When using a method of order p, we need an estimate of the error, $h^{p+2}y^{(p+2)}(x_n)$, in order to estimate the stepsize h, that will be used by the method of order $p+1$. This error estimate can be calculated using a difference of error estimates which are used for controlling stepsize. In order to do this, we need two such estimates, $h^{p+1}y^{(p+1)}(x_n)$ and $h^{p+1}y^{(p+1)}(x_{n-1})$, which can be obtained by keeping the stepsize constant for three steps. Thus, we have

$$\begin{aligned} h^{p+1}y^{(p+1)}(x_n) &\approx \nabla h^p y^{(p)}(x_n), \\ h^{p+1}y^{(p+1)}(x_{n-1}) &\approx \nabla h^p y^{(p)}(x_{n-1}) \end{aligned}$$

and it follows that

$$h^{p+2}y^{(p+2)}(x_n) \approx \nabla h^{p+1}y^{(p+1)}(x_n) - \nabla h^{p+1}y^{(p+1)}(x_{n-1}) = \nabla^2 h^{p+1}y^{(p+1)}(x_n).$$

4.2.6 Interpolation

Interpolation can be used to calculate the solution at a value of x in between two points x_{n-1} and x_n where the solutions $y(x_{n-1})$ and $y(x_n)$ have already been calulated using stepsizes h_{n-1} and h_n, respectively. Let

$$\sigma = \frac{x - x_{n-1}}{h_n}$$

and, if x is closer to x_{n-1} then it is to x_n, we can use

$$\begin{aligned} y(x) &= y(x_{n-1} + \sigma h_n) \\ &= y(x_{n-1}) + \sigma h_n y'(x_{n-1}) + \frac{\sigma^2}{2} h_n^2 y''(x_{n-1}) + \cdots \\ &\quad + \frac{\sigma^p}{p!} h_n^p y^{(p)}(x_{n-1}) + O(h^{p+1}), \end{aligned}$$

where the scaled derivatives $h_n y'(x_{n-1}), h_n^2 y''(x_{n-1}), \ldots, h_n^p y^{(p)}(x_{n-1})$ are available from the Nordsieck vector which has been calculated in preparation for the computation of the solution at x_n. If x is closer to x_n, then, in order to reduce any errors due to interpolation, it is better to use

$$\begin{aligned} y(x) &= y(x_n + (\sigma - 1)h_n) \\ &= y(x_n) + (\sigma - 1)h_n y'(x_n) + \frac{(\sigma-1)^2}{2} h_n^2 y''(x_n) + \cdots \\ &\quad + \frac{(\sigma-1)^p}{p!} h_n^p y^{(p)}(x_n) + O(h^{p+1}), \end{aligned}$$

where the scaled derivatives are available from the Nordsieck vector $\tilde{y}^{[n]}$. Therefore, interpolation can be used for dense output as well as for the calculation of the solution at the endpoint of the integration interval, since the use of variable stepsize usually takes the final computation past the endpoint.

4.3 Variable stepsize/variable order implementation

Using the modified methods as outlined in the previous sections, the type 4 DIMSIMS in which $s = p$ can be implemented in a variable stepsize/variable order code. Apart from the error estimation techniques that have been outlined, there are a few more details which need to be specified. Among these are procedures for controlling stepsize and order, criteria for convergence control of iterations, prediction of starting values for iterations and the stopping criteria for iterations. In developing these procedures, there are many parameters whose values need to be determined. There are very few guidelines available in the literature. Many of the parameters that are mentioned in literature seem to be ad hoc choices. We have experimented with many of these parameters and these will be mentioned in the appropriate sections.

4.3.1 Variable stepsize procedure

Suppose that we are using a method of order p and that an estimate of the LTE, $E = [e_1, e_2, \ldots, e_m]$, has been calculated using an appropriate procedure. Then we set

$$t_i = Atol + Rtol|y_{n,i}|, \tag{4.41}$$

$$w_i = e_i/t_i, \tag{4.42}$$

where $y_{n,i}$ refers to the i^{th} component of the computed solution, y_n, and $Atol$ and $Rtol$ are the relative and absolute tolerances respectively, that are set by the user. We require that the following test be satisfied

$$\|w\| < 1. \tag{4.43}$$

This error test controls an estimate of the local truncation error. If $Rtol = 0$, then we have absolute error control in which case (4.43) reduces to

$$\|E\| < Atol,$$

which means that the norm of the calculated error should be less than the tolerance allowed. If $Rtol \neq 0$ and $Atol = 0$, then we have pure relative error control but we need to be careful that no solution component goes through zero when using such a control.

If the error control (4.43) is satisfied then the integrator accepts the calculated solution and calculates the new stepsize for the next step. The new stepsize, if chosen too large, will result in an excessive error which will not satisfy (4.43) at the next step. If h is the current stepsize and $\bar{h}$ the optimum stepsize for the same step, then we have

$$e_i = Ch^{p+1}y_i^{(p+1)}(x_n),$$

$$t_i = C\bar{h}^{p+1}y_i^{(p+1)}(x_n),$$

where $y_i^{(p+1)}(x_n)$ is the i^{th} component of the $(p+1)^{th}$ derivative of the exact solution $y(x)$ at x_n. We can only hope to satisfy these using any norm, in which case we

have

$$\left(\frac{h}{\bar{h}}\right)^{p+1} = \|w\|,$$

from which we get

$$\bar{h} = h\frac{1}{\|w\|^{1/p+1}}.$$

In case of absolute error control, this reduces to the usual stepsize selection rule

$$\bar{h} = h\left(\frac{Atol}{\|E\|}\right)^{1/p+1}.$$

In order to make the stepsize changing procedure safe, we include a safety factor f, $0 < f < 1$, upper and lower bounds, r_{max} and r_{min} respectively, to get

$$\bar{h} = h * \min\left(r_{min}, \max\left(r_{max}, \frac{1}{\|w\|^{1/p+1}}f\right)\right). \tag{4.44}$$

The selection of the parameters f, r_{min} and r_{max} is based on experimentation. The choice of values affect the efficiency of the code as it determines the stepsizes that are used by the code. Based on our experience with these methods we used $f = 0.8$ after an accepted step, $f = 0.7$ after a rejected step and $r_{min} = 0.5$. Larger values of f such as $f = 0.8$ after a rejected step and $f = 0.9$ after an accepted step often resulted in the next step being rejected, even for smooth problems. It is important to guard against large stepsize increases, so we choose $r_{max} = 2.0$. Using r_{max} larger than 2 increases the chance of rejected steps due to either non-convergence of iterations at the next calculation or large errors.

The error estimates for stepsize control are calculated using the information from two steps. Hence, the first two steps of methods of any order need to be taken with constant stepsize. As outlined in the next section, integration always starts with a first order method. Therefore, the integration needs to start with a very small stepsize. However, using the stepsize controller outlined above, the integrator quickly adapts to an appropriate stepsize.

4.3.2 Variable order procedure

The estimation of error for variable stepsize is done in every successful step. However, the estimation of error for variable order only needs to be done when we think it will be advantageous to change the order. When this needs to be done is not so obvious. It was observed that when using variable stepsize and constant order, the stepsize eventually settles to an almost constant value for problems with smooth solutions. When this stage of calculation is reached, the stepsize ratio becomes close to 1. At this point we could test for order change. In our code we required the stepsize ratio to be between 0.9 and 1.1 before we attempt this. When this was satisfied we forced the calculation for two more steps using the last value of h. This gives us three steps of calculated values using the same stepsize, h. Using these calculated solutions we obtain an estimate of $h^{p+2}y^{(p+2)}(x_n)$ as outlined in Section 4.2.5.

In addition, we required that at least 10 steps of calculation must be carried out using the present method before an attempt is made to change the order. This was done to ensure that the error estimates remained reliable. Furthermore, order change is never contemplated after a rejected step.

Assuming that we are using a method of order p with a stepsize h, the principal local truncation errors for methods of order $p-1$, p and $p+1$ are

$$\begin{aligned} E_{p-1} &= C_{p-1}h^p y^{(p)}(x_n), \\ E_p &= C_p h^{p+1} y^{(p+1)}(x_n), \\ E_{p+1} &= C_{p+1}h^{p+2} y^{(p+2)}(x_n), \end{aligned}$$

where C_{p-1}, C_p and C_{p+1} denote the error constants of methods of order $p-1$, p and $p+1$ respectively. An estimate for $h^p y^{(p)}(x_n)$ is available from the last component of the Nordsieck vector.

The optimal stepsizes h_{p-1}, h_p and h_{p+1}, for orders $p-1$, p and $p+1$ are calculated

in a way similar to that in Section 4.3.1, as

$$h_{p-1} = h\frac{1}{\|w_{p-1}\|^{1/p}},$$

$$h_p = h\frac{1}{\|w_p\|^{1/p+1}},$$

$$h_{p+1} = h\frac{1}{\|w_{p+1}\|^{1/p+2}},$$

where

$$\begin{aligned} w_{p-1,i} &= E_{p-1,i}/t_i, \\ w_{p,i} &= E_{p,i}/t_i, \\ w_{p+1,i} &= E_{p+1,i}/t_i, \\ t_i &= Atol + Rtol\|y_{n,i}\|, \end{aligned}$$

and $E_{p,i} = C_p h^{p+1} y_i^{(p+1)}(x_n)$, is the i^{th} component of E_p and $w_{p,i}$ refers to the i^{th} component of the vector w_p, etc. In the calculation of these stepsizes the safety factors $0.9, 0.8, 0.7$ were used in the calculation of h_{p-1}, h_p and h_{p+1} respectively. A few different values of maximum stepsize ratio in the range $[1.0, 2.0]$ was experimented with, in order to gauge its effect on the acceptance/rejection of the first two steps of the calculated solution for the higher order method. It was found that values near 2.0 resulted in the rejection of the first two steps of the higher order method. There is a need to be conservative in increasing the stepsize when order is increased, since the second difference used to determine this stepsize is not very reliable. But it is better to increase the stepsize by a small amount then not to change it at all. Hence, we used 1.2 as the maximum stepsize ratio after a change of order. This was observed to be acceptable for the problems we tested and for most of the order changes.

For parallel methods, such as the type 4 DIMSIMs that we are considering, the stages can be computed in parallel. This means that no matter how many stages

the method has, the cost of the methods will be exactly the same in parallel. Therefore, the optimum order selected should be the method which corresponds to

$$\hat{h} = \max\{h_{p-1}, h_p, h_{p+1}\}.$$

However, because of the overheads for changing orders, it is only beneficial to change orders if the new stepsize is a reasonable increase. Hence, in our code order is changed only if the stepsize ratio satisfies $\hat{h}/h_p > 1.1$.

Since a DIMSIM method of order more than 1 requires a starting procedure, the integration always needs to start with a method of order 1, which requires only the initial value, and proceed through methods of consecutive higher orders. In this way we avoid having to use a starting procedure. The first order method used is the implicit Euler method. One drawback of using a first order method to start the integration of stiff problems is that we need a very small initial stepsize.

4.3.3 Convergence control of the implicit equation solver

From the stage equations (4.11), we have

$$Y_i = \lambda h f(Y_i) + \psi_i, \qquad i = 1, 2, \ldots, p. \tag{4.45}$$

where $\psi_j = \sum_{j=1}^{p+1} \widetilde{u}_{ij} \tilde{y}_j^{[n]}$ contains the calculations from the past step. Let

$$G(Y_i) = -Y_i + \lambda h f(Y_i) + \psi_i, \qquad i = 1, 2, \ldots, p. \tag{4.46}$$

then, in order to solve $G(Y_i) = 0$, we use a modified Newton iteration scheme which can be stated as

$$\begin{aligned} M\Delta Y_i^{[k]} &= G(Y_i^{[k]}), \\ Y_i^{[k+1]} &= Y_i^{[k]} + \Delta Y_i^{[k]}, \qquad k = 0, 1, \ldots, \end{aligned} \tag{4.47}$$

where, $M = I - \lambda h J$, is the iteration matrix and J is a recent approximation to the Jacobian of f. The notation, $Y_i^{[k]}$, refers to the k^{th} iterate of Y_i. The most important aspect of solving an IVP by the DIMSIM method involves the solution

to the implicit equation (4.47). This is also computationally the most expensive part and its supervision and control which determine the performance of the code, constitute the convergence control.

Using (4.47), we need to solve for $\Delta Y_i^{[k]}$ in order to approximate the stages Y_i, $i = 1, 2, \ldots, s$. The iteration matrix, $M = I - \lambda h J$, is of size $m \times m$ and J is an approximation to the Jacobian. The right hand side of the system, $G(Y_i^{[k]})$, depends on the stage values at the k^{th} iteration and the external stages. This system is solved by factorising the iteration matrix M into LU or PLU factors and then performing a back substitution to get $\Delta Y_i^{[k]}$. To increase efficiency, J and M are not always updated at the beginning of each step but are kept fixed for as many steps as possible as long as a satisfactory convergence rate is being achieved. This strategy is adopted by many codes to reduce computational costs and is based on the fact that the Jacobian evaluations and matrix factorisations are computationally expensive, especially when the size of the system is large and the Jacobian matrix is dense. By doing Jacobian evaluation/matrix factorisation as infrequently as possible we can reduce the overall computational costs. However, if this is too infrequent it can lead to slow convergence which can increase the overall computational costs. To understand why the approximate Jacobian, J, which is supposed to be an approximation to the solution, Y^*, of the algebraic equation (4.46), has an important effect on the convergence, we consider the following analysis. Let

$$J \approx \frac{\partial F}{\partial y}(Y^*),$$

then, using (4.47), the iterates satisfy

$$Y_i^{[k+1]} = \psi + h\lambda f(Y_i^{[k]}) + h\lambda J Y_i^{[k+1]} - h\lambda J Y_i^{[k]}.$$

The desired solution satisfies

$$Y_i^* = \psi + h\lambda f(Y_i^*) + h\lambda J Y_i^* - h\lambda J Y_i^*.$$

Therefore, the iteration error satisfies

$$Y_i^{[k+1]} - Y_i^* = h\lambda \mathbb{J}(Y_i^{[k]} - Y_i^*) + h\lambda J(Y_i^{[k+1]} - Y_i^*) - h\lambda J(Y_i^{[k]} - Y_i^*),$$

where, a mean value has been used for f and the matrix $\mathbb{J}$ consists of the Jacobian of f with its entries evaluated at different points between $Y_i^{[k]}$ and Y_i^*. It can be further shown that

$$Y_i^{[k+1]} - Y_i^* = (I - h\lambda J)^{-1} h\lambda (\mathbb{J} - J)(Y_i^{[k]} - Y_i^*).$$

If there is a number $\rho < 1$ such that

$$\|(I - h\lambda J)^{-1} h\lambda (\mathbb{J} - J)\| \leq \rho,$$

then the error is decreased by a factor of ρ at each iteration. This expression for the effect of an iteration shows what must be done to improve the rate of convergence. The two possibilities are to reduce h or to make J a better approximation to the local Jacobian. Both of these possibilities increase the computational cost which we are trying to reduce for better efficiency. In order to improve the efficiency we need to achieve some sort of a balance in trying to use the old Jacobian and the convergence rate that results. Hence, we need to address the following questions in order to have an effective convergence criteria:

1. When and how often should we evaluate a new Jacobian?
2. When and how often should we form and factorise the iteration matrix?
3. When should we stop the iterations?
4. What should the maximum number of iterations be?
5. How to deal with convergence failures?

Hence, there is a need for a strategy to determine how often one should do a Jacobian evaluation/matrix factorisation in order to obtain the greatest efficiency. These issues are are discussed in publications of Shampine [69] and Diamantakis [40]. The following strategy concerning Jacobian evaluation/matrix factorisation has been adopted:

- In some problems the Jacobian varies very slowly in time and remains almost constant. In such cases, by using the old Jacobian the computational cost can be reduced. So, at each step a previously computed Jacobian is used unless:

1. It is N steps since the last Jacobian evaluation/matrix factorisation. This can be used as a safeguard specially when the convergence rate is unavailable because the all stages in a step converge within one iteration, although this means that for a linear problem there will be unnecessary Jacobian evaluations. In cases where the Jacobian is a constant or varies very slowly we can set N to be a large number in order to avoid unnecessary evaluations.
2. The convergence rate, which is defined in a later section, at the last step is less than a preset convergence rate.
3. There has been a convergence failure at the last step.

- If a new Jacobian is evaluated then the iteration matrix is updated and factorised.

- When the calculated stepsize change ratio satisfies $ratio_{min} < h_{new}/h_{old} < ratio_{max}$, the stepsize changes are small and the old stepsize is used. This allows us to use the old Jacobian and matrix factors. This will reduce the linear algebra costs. This however, has some effect on the overall costs, as it has been noticed that the DIMSIM methods perform better, in terms of less rejected steps, if the integrator allows gradual stepsize changes. As a compromise we used $ratio_{min} = 0.95$ and $ratio_{max} = 1.05$. Using smaller and larger values of $ratio_{min}$ and $ratio_{max}$ respectively, tends to increase the number of rejected steps because this makes the stepsize changes less smooth.

- If the stepsize changes, then a new iteration matrix M is formed and factorised.

In our implementation the *LAPACK* routines *DGETRF* and *DGETRS* [76] were used for the LU or the PLU factorisation and the solution to the linear system respectively.

In using DIMSIM methods it has been noted that the performance of the code deteriorates if the above criterion are modified in favour of evaluating Jacobians

and matrix factorisations less often.

4.3.4 Evaluation of stage derivatives

Suppose that the iterations converge at the k^{th} iteration, then the calculated stage value, $Y_i^{[k+1]}$, is accepted as an accurate enough approximation to Y_i. The derivative values are not available at the k^{th} iteration. An obvious way of obtaining the derivative value is to form the function evaluation, $f(x_n + hc_i, Y_i^{[k+1]})$, $i = 1, 2, \ldots, p$. This corresponds to the $P(EC)^M E$ mode for the predictor-corrector methods.

Since these derivatives are used for error estimation, they are likely to amplify the errors in stiff problems if they are calculated in this way, as explained by Shampine [69]. So instead of a direct function evaluation the stage derivatives are obtained using the stage equations $G(Y_i) = 0$. Therefore we have

$$f(Y_i) = \frac{Y_i - \psi_i}{\lambda h},$$

where $Y_i = Y_i^{[k+1]}$. This corresponds to the $P(EC)^M$ of the predictor-corrector methods. According to the observations of Nørsett and Thomsen [65] this is more efficient than performing function evaluations.

4.3.5 Prediction of starting stage values

The convergence of the Newton iterations (4.47) is sensitive to the choice of starting values of the stages, Y. Starting values which are outside the basin of attraction of Newton's method will lead to divergence while starting values which are not close enough may lead to very slow convergence. For best efficiency good staring values play a very important role. We denote the starting value of the i^{th} stage as

$Y_i^{[0]}$. For type 4 DIMSIMs, the stage values satisfy

$$\begin{aligned} Y_i &= y(x_n + c_i h) + O\left(h^{p+1}\right) \\ &= y(x_n) + c_i h y'(x_n) + \frac{c_i^2}{2} h^2 y''(x_n) \\ &\quad + \cdots + \frac{c_i^p}{p!} h^p y^{(p)}(x_n) + O\left(h^{p+1}\right). \end{aligned}$$

Therefore, the best stage predictor without any additional computational cost is

$$Y_i^{[0]} = y(x_n) + c_i h y'(x_n) + \frac{c_i^2}{2} h^2 y''(x_n) + \cdots + \frac{c_i^p}{p!} h^p y^{(p)}(x_n),$$

where, the approximations to $y(x_n), y''(x_n), \ldots, y^{(p)}(x_n)$, are available from the external stage vector, $\tilde{y}^{[n]}$. In practice this predictor worked well in keeping the number of iterations low and resulted in very few Newton failures.

A better estimator of starting values will reduce the number of iterations that are required to obtain the stage values to the desired accuracies. It has been noticed that the later stages usually require more iterations to converge than the first stage. This has some effect on the load balance on the different processors when these stages are calculated in parallel. Hence, a better estimator for predicting the starting values is needed if we want all the stages to converge with approximately the same number of iterations which we hope to keep as low as possible, preferably one or two. However, with the stage predictor mentioned above the difference in the number of iterations required by the different stages is usually only about one and rarely two or more, for most of the problems we tested.

The use of previous stages in the calculation of starting values of latter stages is a well accepted idea for serial computations but this is not feasible in a parallel implementation.

4.3.6 Stopping criteria

We would like to use some criteria to stop the iterations when we think that the iterates we have obtained satisfy

$$\|Y_i^{[k+1]} - Y_i\| < \tau, \tag{4.48}$$

where τ is the error tolerance of the stage solver and depends on the tolerance used in the stepsize controller. Since Y_i is unknown, the above test is impractical. However, at the end of the k^{th} iteration we have estimates $Y_i^{[k+1]}$, $Y_i^{[k]}$, $Y_i^{[k-1]}$ which can be used. The convergence rate for the i^{th} stage at the k^{th} iteration is defined as

$$r_i^{[k]} = \frac{\|Y_i^{[k+1]} - Y_i^{[k]}\|}{\|Y_i^{[k]} - Y_i^{[k-1]}\|}. \tag{4.49}$$

If we assume that, $r_i < 1$, is an upper bound of the convergence rates, then after a very large number of iterations N, $Y_i^{[k+N+l]}$, $l = 1, 2, \ldots$ is equal to Y_i, then we have

$$\begin{aligned}
\|Y_i^{[k+1]} - Y_i\| &= \|(Y_i^{[k+1]} - Y_i^{[k+2]}) + (Y_i^{[k+2]} - Y_i^{[k+3]}) + \cdots + (Y_i^{[k+N]} - Y_i)\| \\
&\leq \|(Y_i^{[k+1]} - Y_i^{[k+2]})\| + \|Y_i^{[k+2]} - Y_i^{[k+3]}\| + \cdots + \|Y_i - Y_i^{[k+N]}\| \\
&\approx r_i\|W_i^{[k]}\| + r_i^2\|W_i^{[k]}\| + \cdots + r_i^N\|W_i^{[k]}\| \\
&= r_i(1 + r_i + \cdots + r_i^{N-1})\|W_i^{[k]}\| \\
&= r_i\frac{1 - r_i^N}{1 - r_i}\|W_i^{[k]}\| \\
&\approx \frac{r_i}{1 - r_i}\|W_i^{[k]}\|,
\end{aligned}$$

where $W_i^{[k]} = Y_i^{[k+1]} - Y_i^{[k]}$. Hence, if

$$\frac{r_i}{1 - r_i}\|W_i^{[k]}\| < \tau, \tag{4.50}$$

then

$$\|Y_i^{[k+1]} - Y_i\| < \tau,$$

is expected to hold. The upperbound, r_i, is not known but as the number of iterations increases we expect $r_i^{[k]}$ to get closer to r_i. Therefore, we can use

$$r_i = r_i^{[k]}, \tag{4.51}$$

where $k =$ the number of the last iteration. Hence, we accept convergence at the k^{th} iterate if the test

$$\max_{1\leq i\leq p}\left(\frac{r_i^{[k]}}{1-r_i^{[k]}}\|W_i^{[k]}\|\right) < \tau, \qquad k > 0, \tag{4.52}$$

is satisfied. Obviously, this stopping criteria cannot be used at the first iteration since we have no convergence rate available. In this case we can use

$$\|Y_i^{[1]} - Y_i^{[0]}\| < \tau. \tag{4.53}$$

As mentioned in a previous section, we need a measure of the convergence rate in order to have an effective convergence control. In cases where there are at least two iterations than the convergence rate can be calculated and we take the convergence rate of the last stage as the overall convergence rate of that step. If the last stage converges within one iteration but one of the previous stages takes more than one iteration than we can use this convergence rate as the overall convergence rate. This situation is rare, as it has been noticed that the last stage usually takes more iterations to converge than the previous ones and usually at least two. However, the convergence rate cannot be defined by (4.49) when convergence is established using (4.53) within the first iteration. When all the stages converge within the first iteration the overall convergence rate of that step cannot be defined. So the problem faced here is that of controlling convergence without a measure of the convergence rate. There is no simple solution to this problem. If we insist on having a measure of the convergence rate at every step then we need to have at least two iterations at each step but this increases the overall cost. The other alternative is to exclude the convergence rate in the convergence control whenever the overall convergence rate of the last step is not defined. In this case we can depend on the other criteria as mentioned in the last section to control the evaluation of Jacobians and matrix factorisations.

Another important factor for the stopping criteria is the choice of the tolerance for iterations, τ, which depends on the tolerance of the stepsize controller, $T = \|t\|$, where t is defined by equation (4.41). Since the error estimates are calculated using the calculated stage derivatives we need to carry out the iterations to a higher degree of accuracy. Hence, τ needs to be chosen to be smaller than T. However, choosing τ very much smaller than T increases the computational cost but does not make the solution any more accurate. We experimented with $\tau = T/10$, $\tau = T/20$, $\tau = T/50$ and $\tau = T/100$, and found that we obtained the best performance for $\tau = T/50$. Therefore, we have used $\tau = T/50$ in our experimental code.

4.4 Numerical experiments

Theoretically, type 4 DIMSIMs can be constructed for any choice of the c vector with distinct values. For the type 4 methods discussed earlier in this thesis we have considered only two such possibilities, $c = [0, 1/(p-1), 2/(p-2), \ldots, 1]^T$ and $c = [-p+2, -p+3, \ldots, 0, 1]^T$. The first and second order methods we use have $c = 1$ and $c = [0, 1]^T$ respectively. Hence, the different choices only affect methods of order more than two. In order to compare the performance of the methods based on different choices of the abscissae and rank 1 or 2 for V, we have solved the problems listed below for various tolerances. We used the variable stepsize/variable order implementation of the four sets of methods. The maximum order of the methods was set to 5 as it has been observed that methods of order 6 and higher resulted in unreliable error estimates which resulted in unstable behaviour of the solvers. In the tabulated results and discussions we refer to the sets of methods as follows:

- DIMSIM A: V of rank 1 and $c = [0, 1/(p-1), 2/(p-2), \ldots, 1]^T$,
- DIMSIM B: V of rank 1 and $c = [-p+2, -p+3, \ldots, 0, 1]^T$,
- DIMSIM C: V of rank 2 and $c = [0, 1/(p-1), 2/(p-2), \ldots, 1]^T$,
- DIMSIM D: V of rank 2 and $c = [-p+2, -p+3, \ldots, 0, 1]^T$.

In order to compare numerical methods for the solution of initial value problems, we need to get some measure of the cost of computation. The cost of computations is directly related to the following:

- the total number of steps required to reach the endpoint of integration,
- the number of function evaluations in parallel,
- the number of Jacobian evaluations,
- the number of LU decompositions,

For parallel numerical methods, a good measure is the the total number of function evaluations done in parallel. Since the stages are evaluated in parallel, the stages with the maximum number of function evaluations in every step determine the overall time taken. So instead of the total number of function evaluations which is the usual criteria for serial methods, we can look at the total of the maximum number of function evaluations.

When the analytic solution is known, the accuracy of the calculated solution can be measured by the maximum global error produced in the integration interval. If the analytic solution is not known, then the endpoint global error can usually be calculated by having an accurate numerical solution. For the problems tested these solutions were readily available.

4.4.1 Problems tested

The Robertson problem (2.6) was solved with $Atol = tol$ and $Rtol = 10^{-4}tol$, where tol is the user set tolerance, $Atol$ is the absolute tolerance and $Rtol$ is the relative tolerance, for $x \in [0, 10^6]$. The Van der Pol problem (2.8) was solved with $Atol = Rtol = tol$, for $x \in [0, 2]$. The following additional problems were solved.

Kaps problem

Kaps problem [61] is a 2 dimensional stiff system described by the equations

$$\begin{cases} y_1'(x) &= -1002y_1(x) + 1000y_2^2(x), & y_1(0) &= 1, \\ y_2'(x) &= y_1(x) - y_2(x)(1 + y_2(x)), & y_2(0) &= 1, \end{cases} \tag{4.54}$$

with the Jacobian matrix

$$\begin{bmatrix} -1002 & 2000y_2 \\ 1 & -1-2y_2 \end{bmatrix}.$$

The exact solution for the Kaps problem is

$$y_1(x) = e^{-2x}, \quad \text{and} \quad y_2(x) = e^{-x}.$$

We consider solution to this problem for $x \in [0, 10]$ using $Atol = Rtol = tol$ and since we have an analytic solution for Kaps problem, we can report the maximum global error for each of the two components. This problem was used to study the error estimates and the order control behaviour of the solver.

Oregonator problem

This is a famous model with a periodic solution describing the Belusov-Zhabotinskii reaction [49] and is given by

$$\begin{cases} y_1'(x) = 77.27(y_2(x) + y_1(x)(1 - 8.375 \times 10^{-6} y_1(x) - y_2(x))), & y_1(0) = 1, \\ y_2'(x) = \frac{1}{77.27}(y_3(x) - y_2(x)(1 + y_1(x))), & y_2(0) = 2, \\ y_3'(x) = 0.161(y_1(x) - y_3(x)), & y_3(0) = 3, \end{cases} \tag{4.55}$$

with the Jacobian matrix

$$\begin{bmatrix} 77.27 - 1.294272 \times 10^{-3} y_1 - 77.27 y_2 & 77.27 - 77.27 y_1 & 0 \\ -\frac{y_2}{77.27} & \frac{-1-y_1}{77.27} & \frac{1}{77.27} \\ 0.161 & 0 & -0.161 \end{bmatrix}.$$

We have integrated this problem for $x \in [0, 30]$ using $Atol = 10^{-6} tol$, and $Rtol = tol$.

Ring Modulator problem

This is an ODE system which models a circuit and is from the CWI testset [63].

It is a very stiff system of 15 differential equations which can be written as,

$$\begin{cases} Cy_1' &= y_8 - 0.5y_{10} + 0.5y_{11} + y_{14} - y_1/R, \\ Cy_2' &= y_9 - 0.5y_{12} + 0.5y_{13} + y_{15} - y_2/R, \\ C_s y_3' &= y_{10} - g(z_1) + g(z_4), \\ C_s y_4' &= -y_{11} + g(z_2) - g(z_3), \\ C_s y_5' &= y_{12} + g(z_1) - g(z_3), \\ C_s y_6' &= -y_{13} - g(z_2) + g(z_4), \\ C_p y_7' &= -y_7/R_i + g(z_1) + g(z_2) - g(z_3) - g(z_4), \\ L_h y_8' &= -y_1, \\ L_h y_9' &= -y_2, \\ L_{s2} y_{10}' &= 0.5y_1 - y_3 - R_{g2} y_{10}, \\ L_{s3} y_{11}' &= -0.5y_1 + y_4 - R_{g3} y_{11}, \\ L_{s2} y_{12}' &= 0.5y_2 - y_5 - R_{g2} y_{12}, \\ L_{s3} y_{13}' &= -0.5y_1 + y_6 - R_{g3} y_{13}, \\ L_{s1} y_{14}' &= -y_1 + e_1(x) - (R_0 + R_{g1}) y_{14}, \\ L_{s1} y_{15}' &= -y_2 - (R_a + R_{g1}) y_{15}, \end{cases} \tag{4.56}$$

where

$$\begin{aligned} z_1 &= y_3 - y_5 - y_7 - e_2(x), \\ z_2 &= -y_4 + y_6 - y_7 - e_2(x), \\ z_3 &= y_4 + y_5 + y_7 + e_2(x), \\ z_4 &= -y_3 - y_6 + y_7 + e_2(x), \end{aligned}$$

and the function g which modelizes the functioning of the diode, is given by

$$g(z) = 40.67286402 \times 10^{-9}[\exp(17.7493332z) - 1].$$

The technical parameters take the following values

$$C = 16 \times 10^{-9}, \quad C_p = 10^{-8}, \quad L_h = 4.45, \quad L_{s1} = 0.002, \quad L_{s2} = L_{s3} = 0.0005,$$

$$R_{g1} = 36.3, \quad R_{g2} = R_{g3} = 17.3, \quad R_0 = R_i = 50, \quad R_a = 600, \quad R = 2500,$$

and the signals are given by

$$e_1(x) = 0.5sin(2\pi 10^3 x), \quad e_2(x) = 2sin(2\pi 10^4 x).$$

The initial conditions are given by

$$y_i(0) = 0, \quad i = 1, 2, \ldots, 15.$$

The system behaves differently for different choices of C_s. Here we have integrated the system for $C_s = 10^{-9}$. This value is technically meaningful and makes the system solvable in a reasonable amount of time, for $x \in [0, 10^{-3}]$. We have used $Atol = Rtol = tol$.

4.4.2 Effect of initial stepsize

The choice of the initial stepsize, h_0, has some effect on the number of total number of steps required for the integration to complete. The optimum initial stepsize depends on the tolerance used to solve the problem. Since these methods start with order one, the initial stepsize needs to be quite small. We investigated the effect of h_0 using the Kaps problem and DIMISIM A. Values of h_0 between 10^{-2} and 10^{-8} were used to solve the problem for a few different tolerances, and a set of results is presented in Table 4.2. For simplicity only the number of accepted and rejected steps are shown. It is seen that when the tolerance is less stringent the initial stepsize can be bigger without many rejected steps. In such cases, if the initial stepsize is made very small then the total number of steps required to complete the integration increases. However, for more stringent tolerances the initial stepsize needs to be appropriately smaller. Larger initial stepsizes result in many rejected steps at the start of the integration while the integrator tries to find a suitable stepsize. The suitability of the initial stepsize is also problem dependent, for example, it has been noticed that the van der Pol problem required much smaller initial stepsize compared to the Kaps problem. However, the integrator is able to quickly adapt the stepsize to suit the conditions of the problem. Hence, we set $h_0 = 10^{-6}$ as the initial stepsize for all problems.

Table 4.2: Effect of h_0, Kaps problem and DIMSIM A.

TOL	h_0	TOT ST	REJ ST
10^{-2}	10^{-2}	28	0
	10^{-4}	31	0
	10^{-6}	38	0
	10^{-8}	45	0
10^{-6}	10^{-2}	216	10
	10^{-4}	208	2
	10^{-6}	219	2
	10^{-8}	220	0
10^{-10}	10^{-2}	578	20
	10^{-4}	567	8
	10^{-6}	555	0
	10^{-8}	563	0

4.4.3 Performance of error estimators

Kaps problem was used to study the error estimates, since we can calculate the true errors using the exact solution. The error estimates for stepsize control were calculated using (4.22) for DIMSIM A and B and using (4.40) for DIMSIM C and D. In order to illustrate how well these estimates represent the errors they are trying to approximate, we obtained the values of these errors for two tolerances, 10^{-4} and 10^{-8}. These errors are plotted in Figures 4.3-4.10. for a selected set of steps. On these diagrams error(1) and error(2) refer to the errors in components 1 and 2 respectively.

Since DIMSIMs A-D have the same methods of orders 1 and 2, we have plotted the errors for orders 1 and 2 separately in Figure 4.3. Since we start the integration with a small stepsize the error estimates at the beginning are very good for both the components. The estimates for both the components behave in a similar manner. When, $tol = 10^{-4}$, the smoothness of the estimates is interrupted at about $x = 0.1$ where the order changes to 2, and we see that this is also true for the estimates of the second component. At this order change we see that the error estimates are not very close to the true errors for component 1, however, they improve after only a few steps. The error estimates for component 2 are always

smaller in magnitude (as the solution has smaller magnitude) and these estimates seem to be always close to the exact the errors even after an order change. Since we use the maximum norm, the error estimates of component 1 control the stepsize. The behaviour of the error estimates when $tol = 10^{-8}$, is similar except that the order change occurs much earlier at $x \approx 0.001$.

Figures 4.4-4.7 show the errors for orders 3 and higher. Since these methods are different for DIMSIM A-D we have plotted them separately. When $tol = 10^{-4}$, we see from Figure 4.1 that the methods use order 3 for the rest of the integration. Hence, Figures 4.4 and 4.5 show the errors for the order 3 methods, which have different abscissae and ranks of the V matrix. Again we see that at the order change, the estimates and the exact errors are not close but as the integration progresses the estimates get closer to the true errors. However, we also see that towards the end of integration the estimates get further away from the true errors and this is due to the larger stepsizes. The estimates for the second component behave in a similar manner but it is seen that the error estimates for the first component are usually over-estimates, while those for the second component are under-estimates.

When $tol = 10^{-8}$, methods of orders 2 to 5 are used, as can be seen from Figure 4.1. From Figure 4.6 we again see that, when order changes the error estimates are affected but they improve as the integration progresses. The error estimates for the remaining part of the integration interval for DIMSIMs A and B behave in a similar way. From Figure 4.7 we see that the estimates in DIMSIMs C and D seem to behave in a chaotic manner for the order 5 method which is used after $x \approx 0.5$. The estimates in DIMSIM D for the remainder of the interval improves but the estimates from DIMSIM C which is given in Figure 4.8 does not improve very much. Hence, we expect more rejected steps for DIMSIM C.

The higher order error estimates for order control are plotted in Figures 4.9 and 4.10. These figures show all the estimates that are calculated, irrespective of whether order was changed or not, for $tol = 10^{-8}$. It is fairly obvious that the estimated errors are not close to the errors they are trying to estimate. However, these errors are only used to determine whether order change will be beneficial

and by using other conservative criteria as discussed earlier, these estimates can be used to control order.

4.4.4 Performance of order control

As stated earlier, the variable stepsize, variable order DIMSIM methods have been implemented to begin with a method of order 1. For smooth problems, such as the Kaps problem, the variable order strategy causes the orders of the methods to increase progressively. While solving such problems with lower tolerances, the lower orders are usually selected, as seen in Figure 4.1. It is also seen that for more stringent tolerances the higher order methods are used and this is what one would expect. Since Kaps problem has a smooth solution the orders do not fluctuate. The Oregonator problem has a solution which is not so smooth, and the way in which orders are varied to solve this problem are different from that of the Kaps problem, as seen in Figures 4.1 and 4.2. For this problem it is seen that the orders sometimes decrease and sometimes increase and again for the lower tolerance, 10^{-2}, only orders 1 to 2 are used but for a more stringent tolerance, 10^{-6}, orders 1 to 5 are used. At $x \approx 23$, there is a sudden change in the solution of all three components, and this results in a change in the magnitudes of the higher order derivatives which cause a reduction in stepsize. After $x = 25$, the solution again becomes smooth and so the orders increase. The order change behaviour is as one would expect for these types of problems. These observations are similar to that of Shampine [70], where the behaviour of the variable order Adams and BDF codes are illustrated.

Variable order codes solve problems more efficiently by selecting the order most appropriate for the conditions during the integration. As mentioned earlier low orders are usually selected at lax tolerances and high orders for more stringent tolerances. To illustrate this we solved Kaps problem by restricting the maximum order available for a few stringent tolerances. The results given in Table 4.3 are for DIMSIM A. It is seen that the availability of the fourth and fifth order methods have drastically reduced the number of steps that are taken when the maximum

order is three. The other statistics, such as, the number of Jacobian evaluations, the number of LU factorisations, and the number of function evaluations in parallel are also markedly reduced and at the same time the global error has been slightly reduced. Hence, we expect a variable order code to perform more efficiently than a fixed order code.

Table 4.3: Effect of variable order, Kaps problem and DIMSIM A.

TOL	MAX ORD	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	MAX GL ER
10^{-8}	3	627	3	72	142	1247	1.8×10^{-7}
	4	527	2	59	120	915	5.7×10^{-8}
	5	321	3	44	117	579	1.3×10^{-7}
10^{-10}	3	1868	0	176	242	3730	6.0×10^{-9}
	4	1252	0	121	191	2191	1.2×10^{-9}
	5	555	0	59	134	983	5.5×10^{-10}

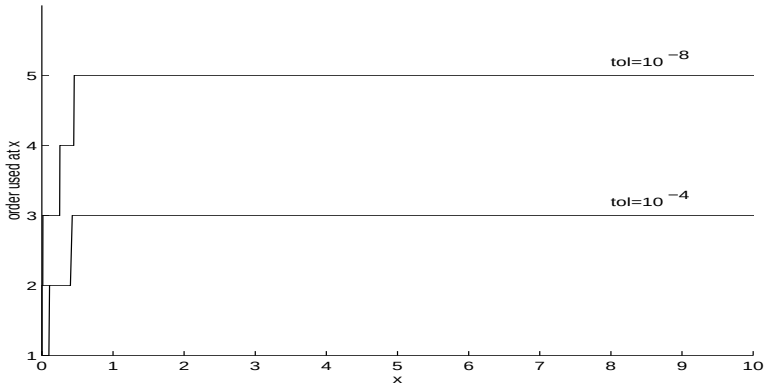

Figure 4.1: Order control for Kaps problem.

4.4.5 Performance of type 4 DIMSIMs

In order to compare the results obtained by these methods, the stiff problems were also solved using two of the best available solvers, RADAU5 [49] and VODE [4]. RADAU5 is based on the Runge-Kutta Radau IIA method of 3-stages and order 5 (2.19), while VODE for stiff problems is based on the BDF methods of orders 1 to 5. These results are given in Appendix B. The importance of comparing

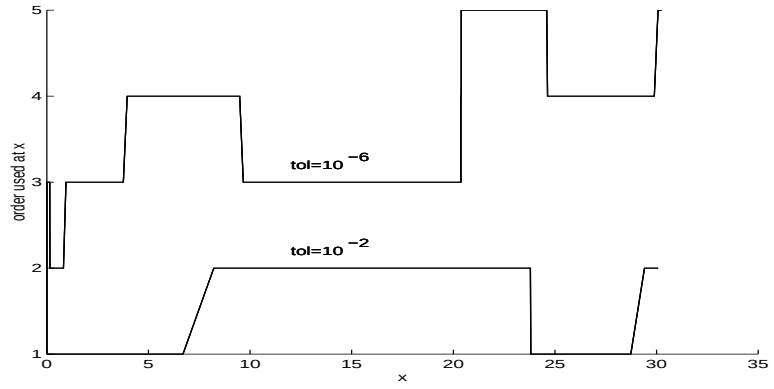

Figure 4.2: Order control for Oregonator problem.

parallel numerical methods with the state of the art sequential solvers is discussed by Burrage [5].

Comparing the performance of numerical methods is not an easy task. There are many factors which affect the performance of the methods and different methods handle some of the important tasks such as error estimation and order control differently. However, some of the statistics such as the number of function evaluations needed to calculate the solution to a measured accuracy usually gives a good measure of the performance of the method. In the following discussions we use the functions counts to determine if the choice of abscissae or the rank of the V matrix affects the performance of type 4 DIMSIMs, and to compare the performance of these methods with RADAU5 and VODE.

For Kaps problem there was no significant difference in performance between the four sets of methods, DIMSIMs A-D, as seen from the results in Table 4.4 and Figure 4.11. However, it is seen that method A, which has V of rank 1 and the abscissae is equally spaced within the integration interval, gives the a slightly better performance. There are very few rejected steps due to excessive local truncation errors. The larger number of rejected steps for DIMSIM C with smaller tolerances is due to the behaviour of the error estimates as illustrated in Section 4.4.3. There were no rejected steps due to the non convergence of Newton iterations.

The Robertson problem is much more difficult to solve and the problem certainly

imposes a severe test on the stepsize and order selection algorithms. These methods are able to solve the problem using stepsizes in the range $[10^{-6}, 10^{5}]$. These methods had difficulty solving this problem for some tolerances when $x > 10^{6}$. In such instances, it has been noticed that, the rounding errors make the second component of the solution slightly negative and this leads to overflow in the solution. To handle such situations, we need to make special efforts to reduce rounding off errors which we have not done in this implementation. The number of rejected steps are very low and these are always due to excessive local truncation errors. DIMSIM D which has rank 2 for the V matrix performs a little bit better than the other methods for small tolerances. This is most likely due to the fact that these methods have a smaller error constants.

The van der Pol problem is a much more difficult problem to solve numerically, not only because of stiffness but also because of the sudden change in the solution at two points in the integration range, as in Figure 2.2. At these points the stepsize needs to reduce very quickly to cope with the sudden change in the solution. All the four sets of methods solve this problem adequately, although this results in a number of rejected steps which are always due to excessive local truncation errors. DIMSIM B has the least number of rejected steps.

The first component of the solution of the Oregonator problem has a sudden change in solution at two places in the integration interval, as seen in Figure 4.16. At these points the numerical methods need to be able to reduce stepsizes very quickly and the four sets of methods cope with this well. The last plot in Figure 4.16 shows the variation of stepsizes used by DIMSIM A to solve this problem with $tol = 10^{-2}$. Most of the rejected steps are due to excessive local truncation errors. DIMSIM C has a lot more rejected steps compared to the other methods. Although there is no clear-cut performance difference between the four sets of methods, method D performs a little bit better than the other methods for small tolerances, as seen from Figure 4.14 .

The solution components of the ring modulator problem are highly oscillatory as shown in Figure 4.17. As a consequence, there are many rejected steps, almost all of which are due to excessive local truncation errors. The performance of method

D is slightly better than the others at small tolerances.

Overall there does not seem to be much difference in the performance of DIMSIMs A-D when they are used to solve these problems. In particular, the choice of the abscissae does not seem to be critical to the performance of the methods when only orders 1 to 5 are used. The rank of the V matrix determines the size of the error constants. As seen earlier, methods with rank 1 for the V matrix have larger error constants than methods where the V matrix has rank 2. This is the most likely reason for the slightly better performance of DIMSIM D in most of the problems. However, this difference is not very marked. Rank 1 methods seem to have better error estimators.

The performance of a differential equation solver depends very much on the error estimation algorithm and the stepsize and order controllers. Since the stepsize selection and order control depend on the error estimates, the single most important factor determining the performance of the methods is the error estimation. The error estimates for DIMSIMs A, B and D seem to be satisfactory while the error estimates for DIMSIM C are not satisfactory as these result in many more rejected steps.

The starting values used for the Newton iterations seem to be working quite well, as there were very few step rejections due to non convergence of the iterations, in any of the problems solved. The maximum number of iterations required to solve the stages is usually less than 2 for most of the problems, although it is sometimes more than 3 for some problems such as the van der Pol oscillator. This is closely related to the stepsizes that are used during the integration. It is noted here that the DIMSIMs A-D usually take many more steps compared to either RADAU5 or VODE. The use of smaller stepsizes results in higher computational costs in terms of higher number of function evaluations, Jacobian evaluations and LU decompositions. Even with these larger number of steps the DIMSIMs sometimes give lower accuracy than that requested compared to RADAU5 or VODE. For the Robertson and the Oregonator problems the number of function evaluations are quite comparable to that of VODE but VODE has much lower Jacobian/LU factorisation costs. Thus, it is apparent that in order to make the

DIMSIMs competitive we need to reduce the computational costs. This can only be achieved by reducing the number of steps that are required to complete an integration. However, this does not seem to be possible with these methods.

As noted in chapter 4 the error constants of the type 4 DIMSIMs tested in this chapter are quite large. This is one of the reasons for the large number of steps. Suppose that we have integrated at x_n and now want to choose a stepsize, h_n, for the next step. From the stepsize selection algorithm the new stepsize is approximately given by

$$h_n = \left(\frac{tol}{|C_{p+1}| \|y^{(p+1)}(x_n)\|} \right)^{\frac{1}{p+1}},$$

from which we see that the new stepsize is proportional to $(\frac{1}{|C_{p+1}|})^{\frac{1}{p+1}}$. Larger error constants will result in smaller stepsizes, so the methods will take many smaller steps to complete the integration. On the other hand, smaller error constants will result in larger stepsizes, which in turn will result in fewer steps to complete the integration. This we hope will lead to lower computational costs. This motivates the construction of type 4 methods which have smaller error constants and these will be discussed in Chapter 5.

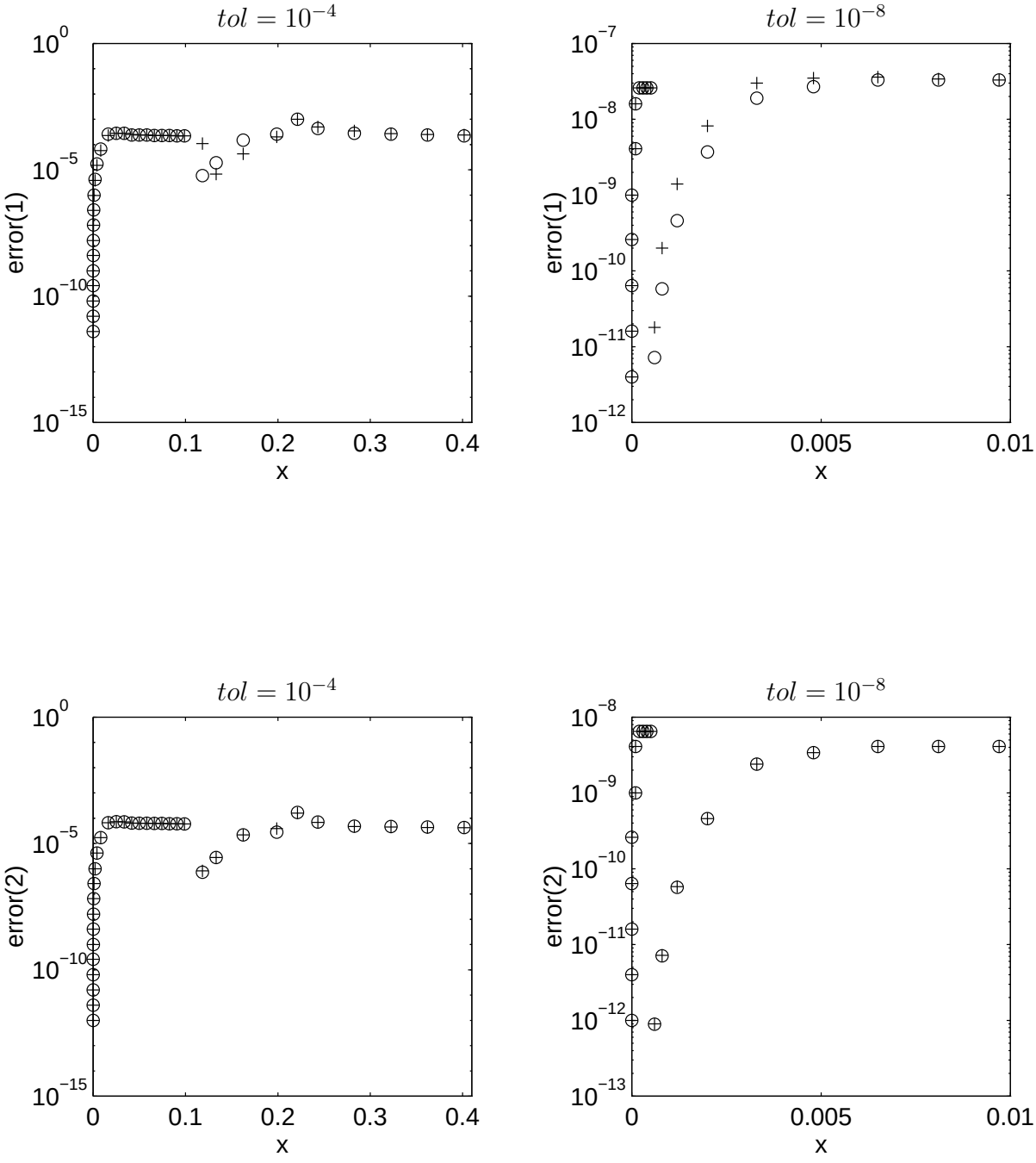

Figure 4.3: Errors (+ – est, ∘ – exact), Kaps problem, orders 1 and 2.

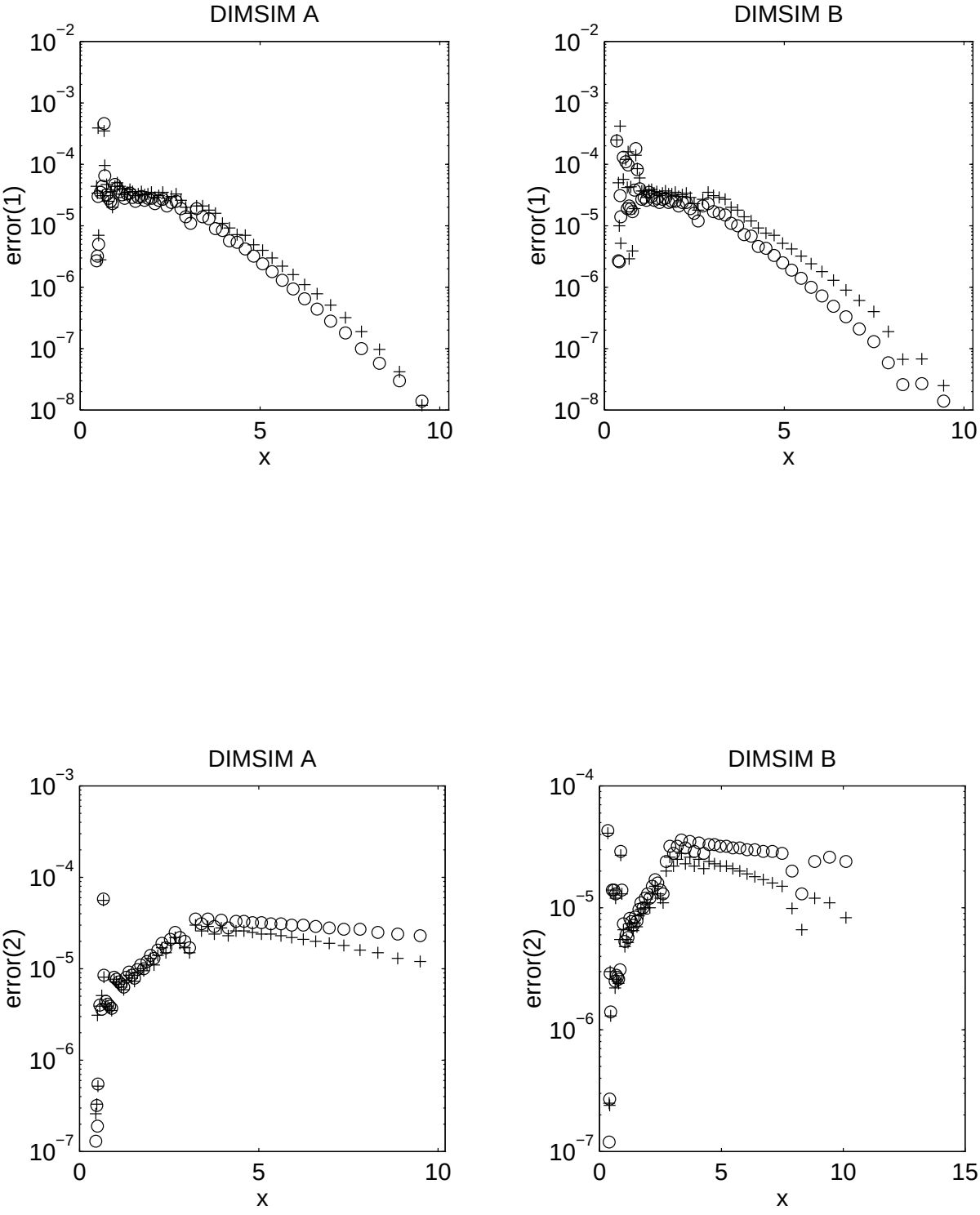

Figure 4.4: Errors (+ – est, ∘ – exact), Kaps problem, $tol = 10^{-4}$, using DIMSIMs A and B.

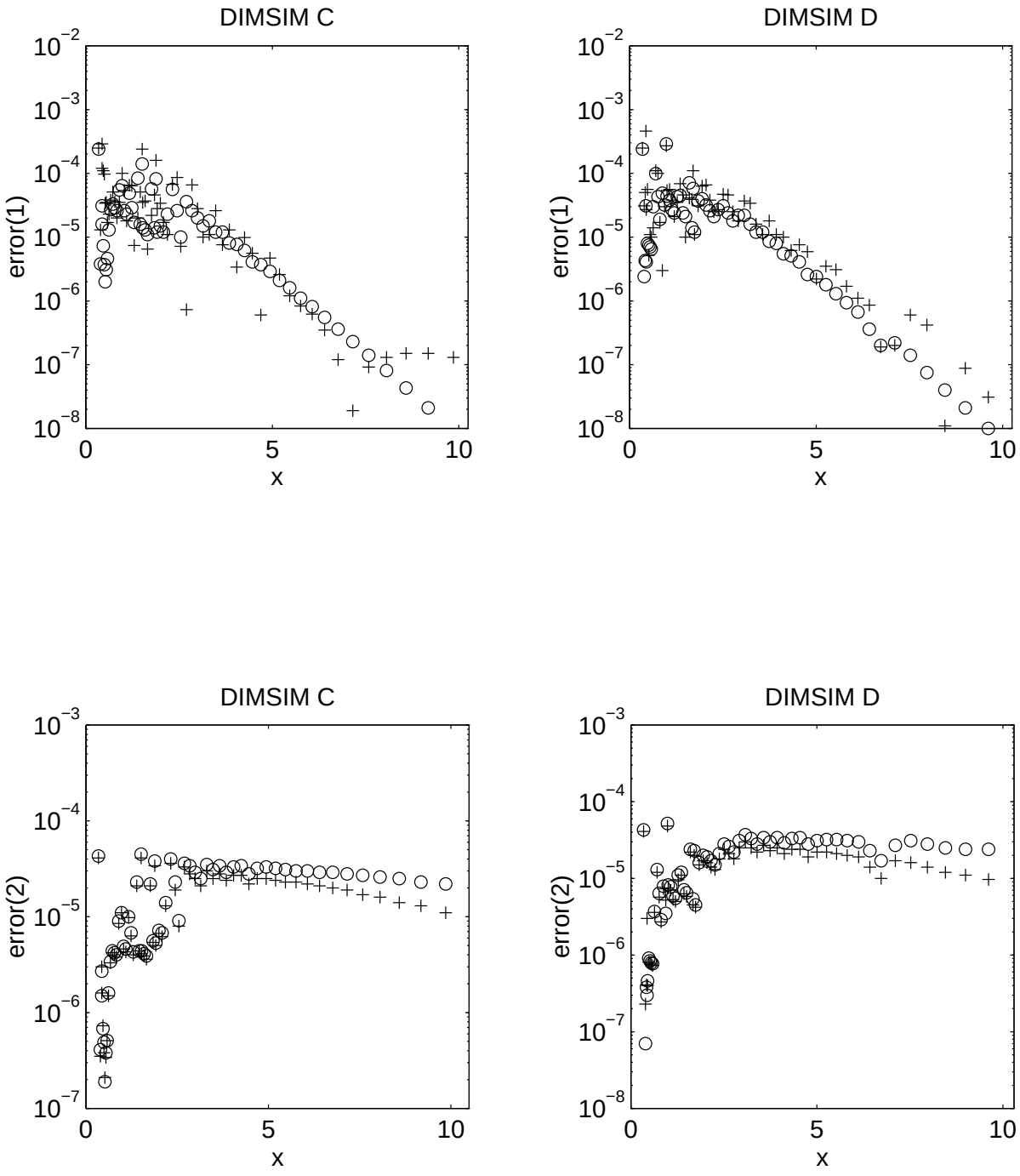

Figure 4.5: Errors (+ – est, ∘ – exact), Kaps problem, $tol = 10^{-4}$, using DIMSIMs C and D.

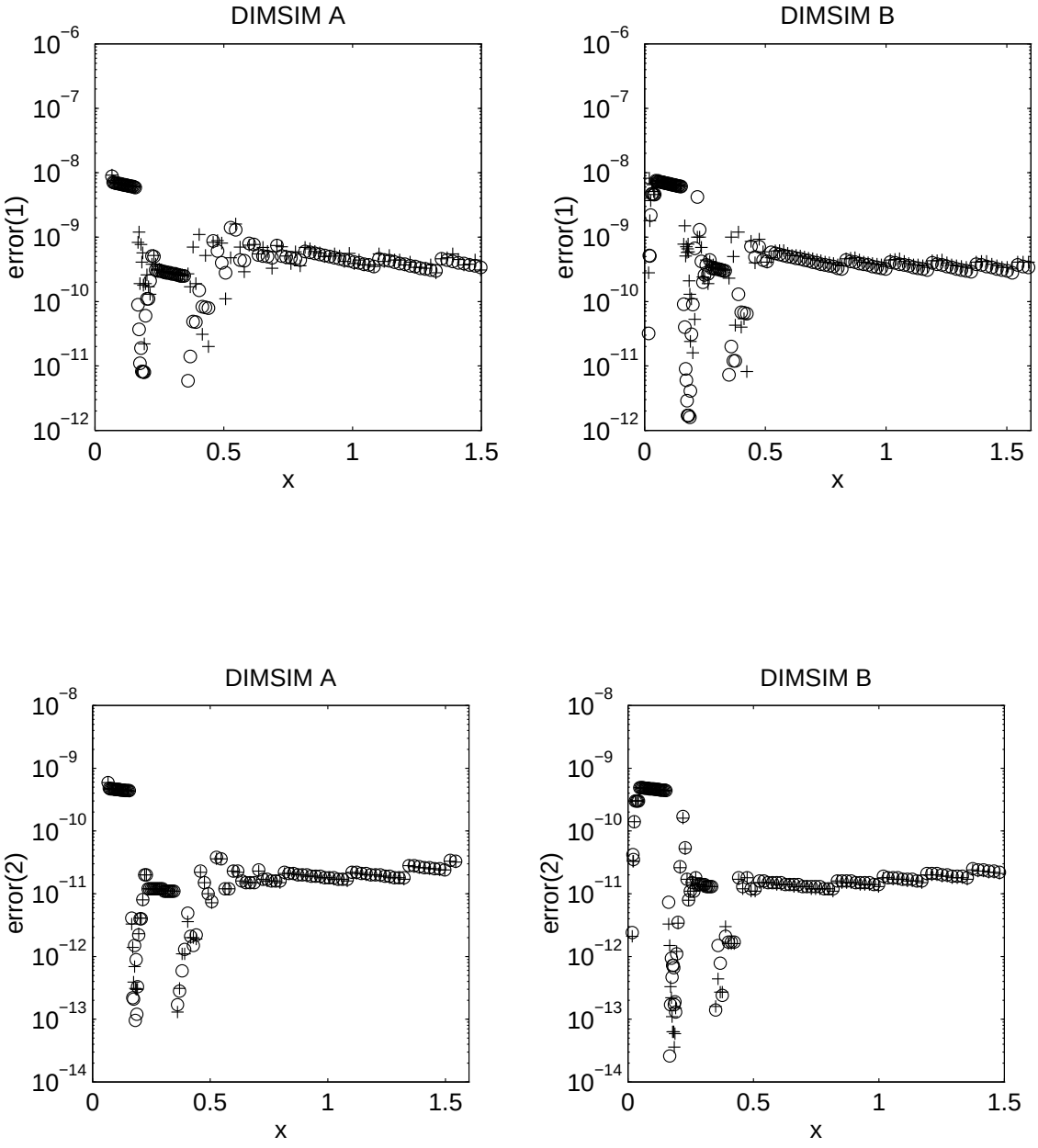

Figure 4.6: Errors (+ – est, ∘ – exact), Kaps problem, $tol = 10^{-8}$ using DIMSIMs A and B.

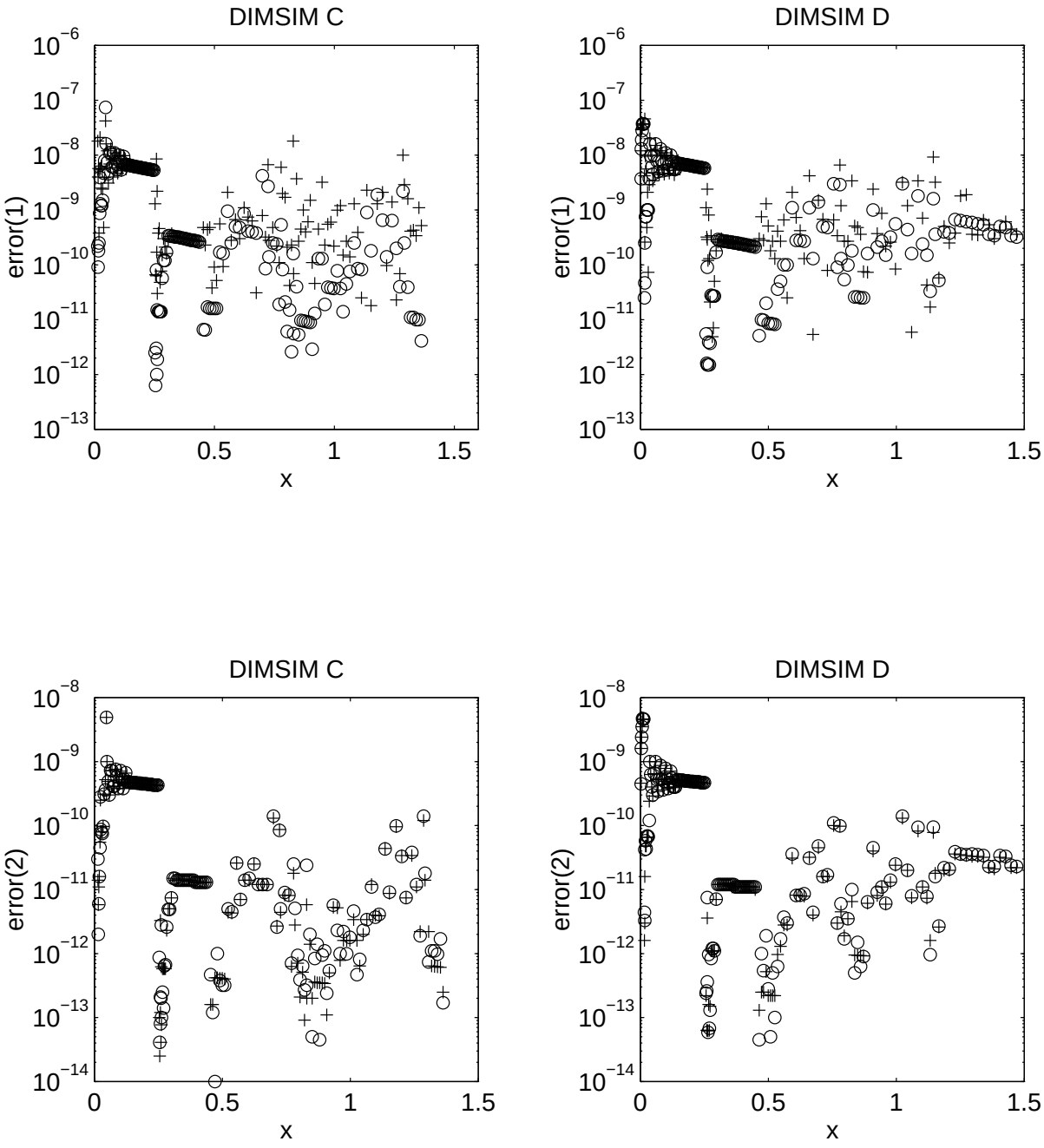

Figure 4.7: Errors (+ – est, ∘ – exact), Kaps problem, $tol = 10^{-8}$, using DIMSIMs C and D.

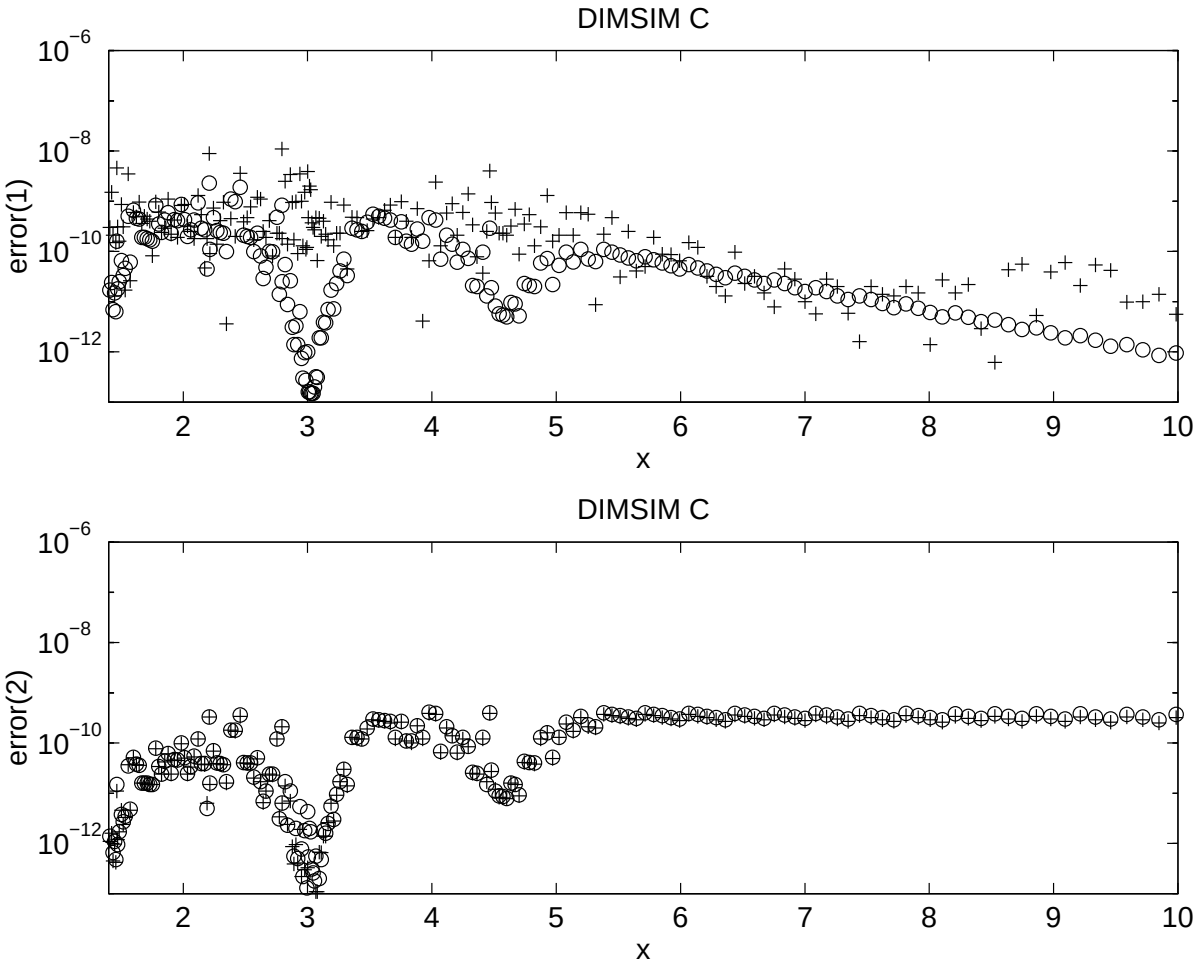

Figure 4.8: Errors (+ – est, ∘ – exact), Kaps problem, $tol = 10^{-8}$, using DIMSIM C.

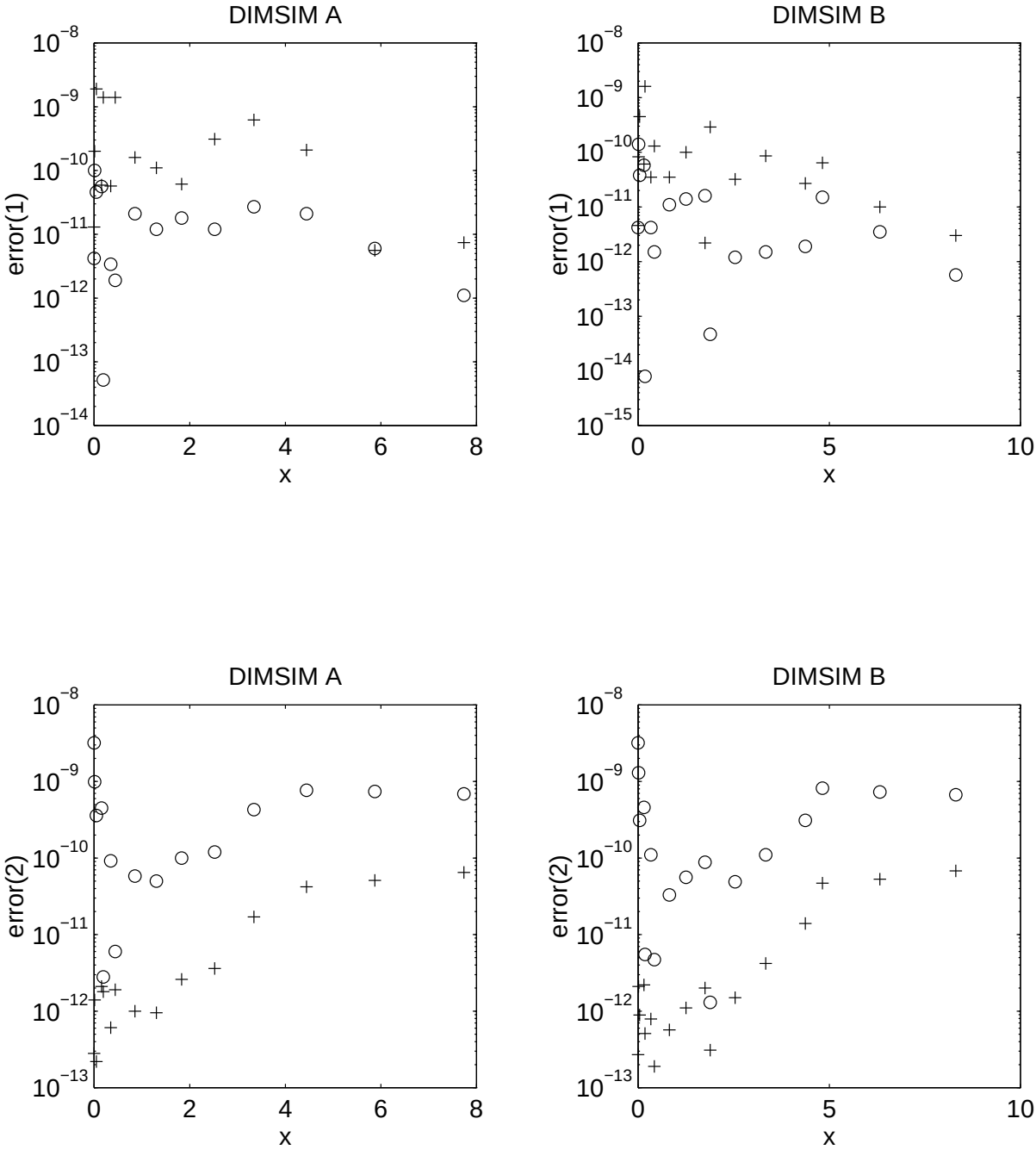

Figure 4.9: Errors (+ – est, ∘ – exact) for order control, Kaps problem, $tol = 10^{-8}$, using DIMSIMs A and B.

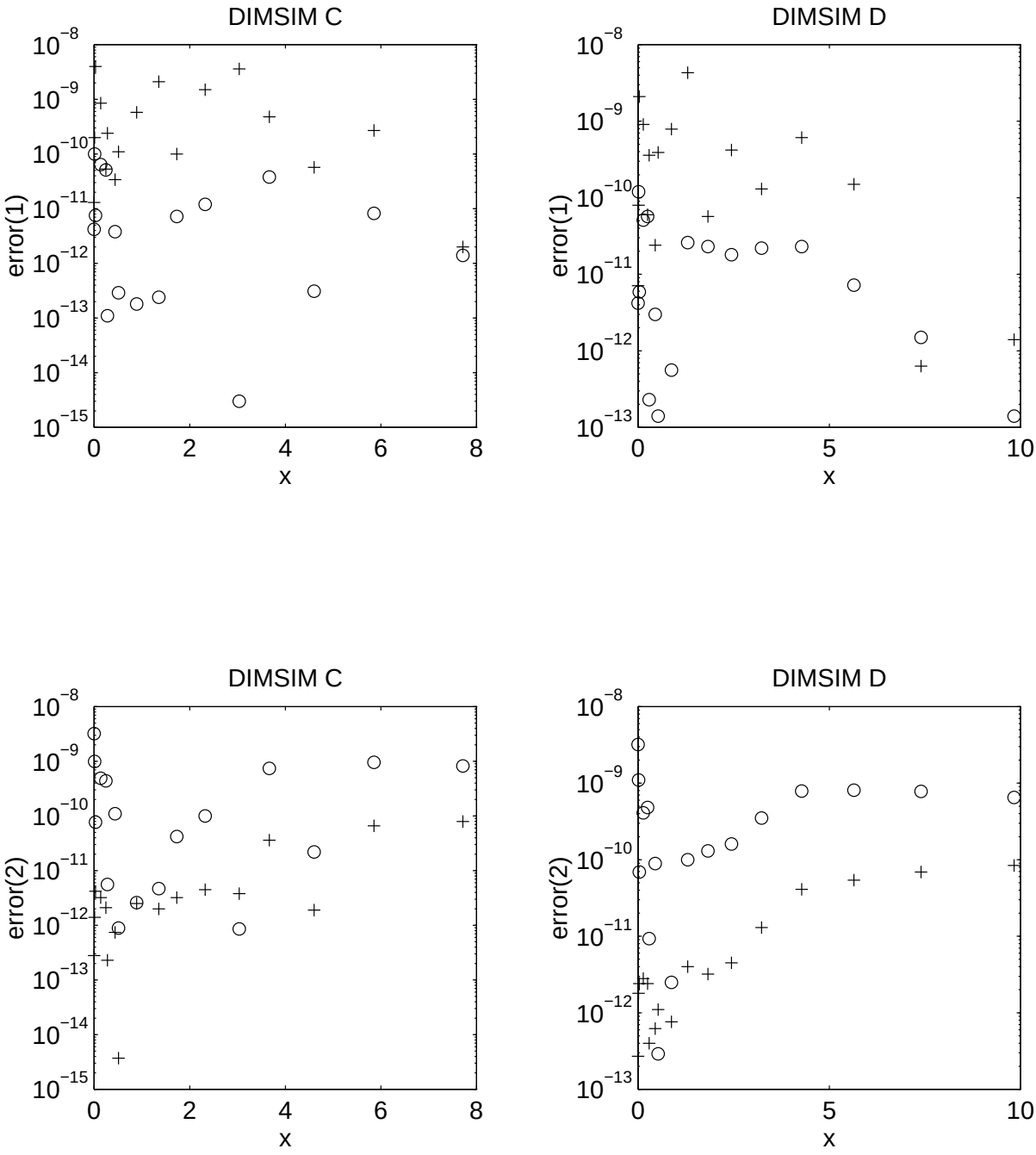

Figure 4.10: Errors (+ – est, ∘ – exact) for order control, Kaps problem, $tol = 10^{-8}$, using DIMSIMs C and D.

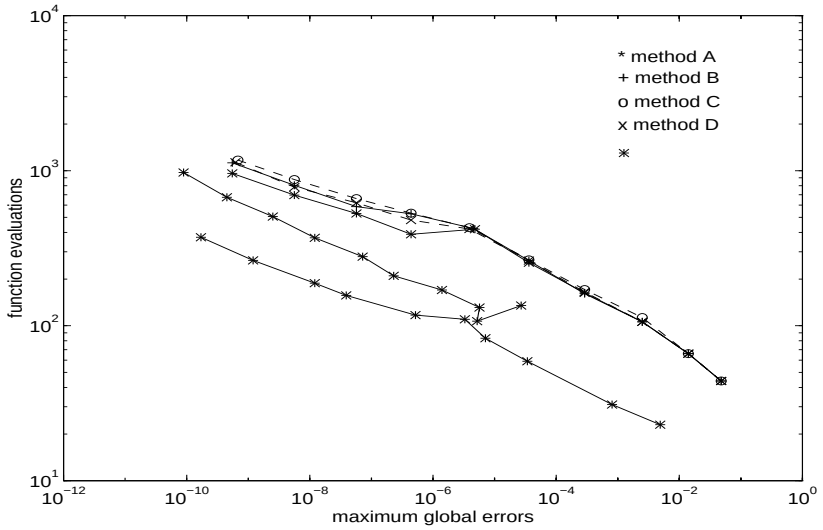

Figure 4.11: Function evaluations/final global errors: Kaps problem.

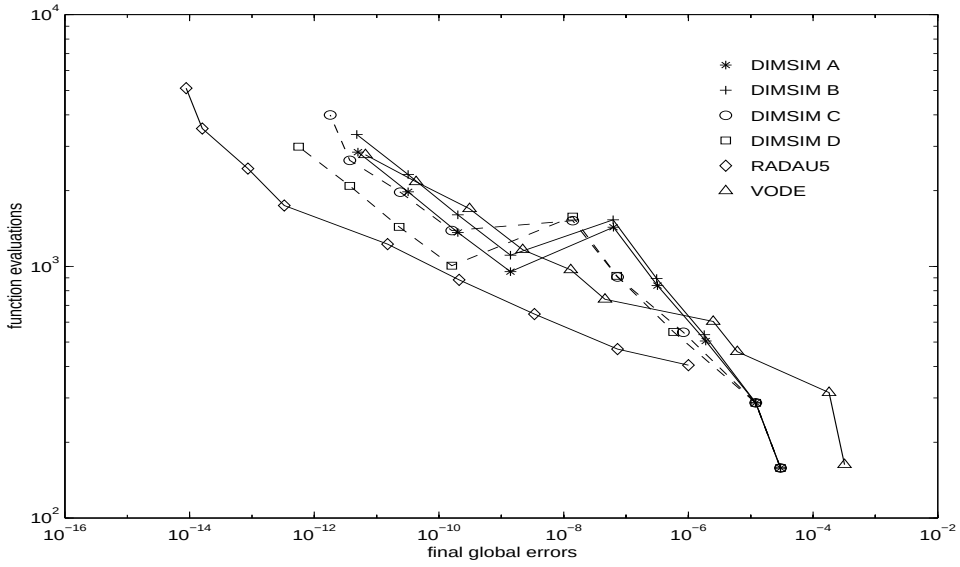

Figure 4.12: Function evaluations/final global errors: Robertson problem.

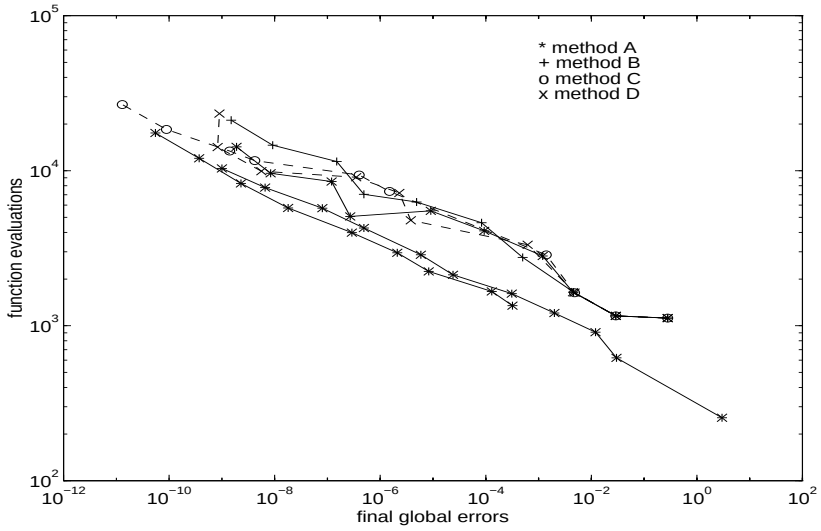

Figure 4.13: Function evaluations/final global errors: van der Pol problem.

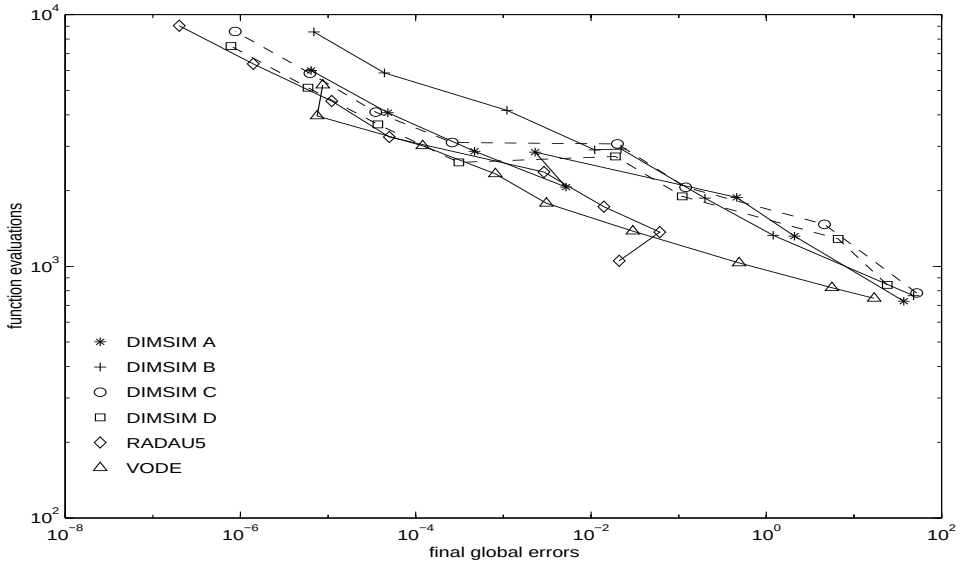

Figure 4.14: Function evaluations/final global errors: Oregonator problem.

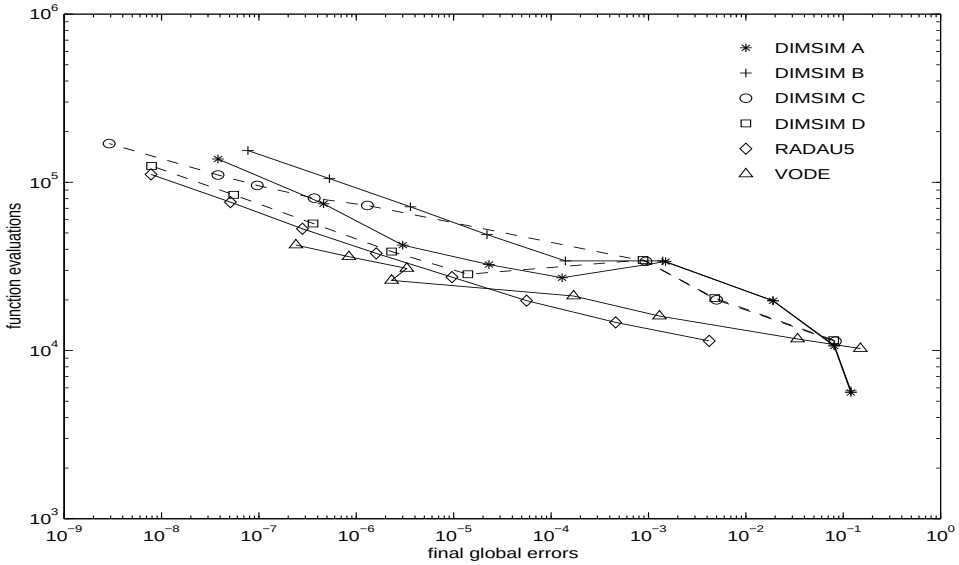

Figure 4.15: Function evaluations/final global errors: Ring Modulator problem.

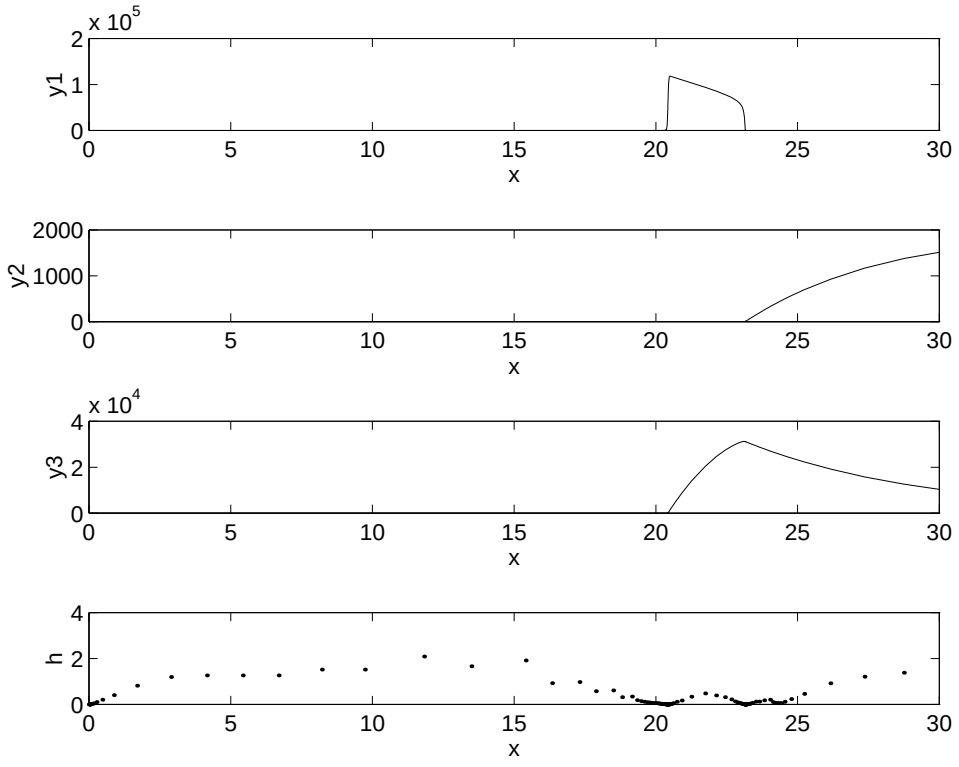

Figure 4.16: Solution profile: Oregonator problem.

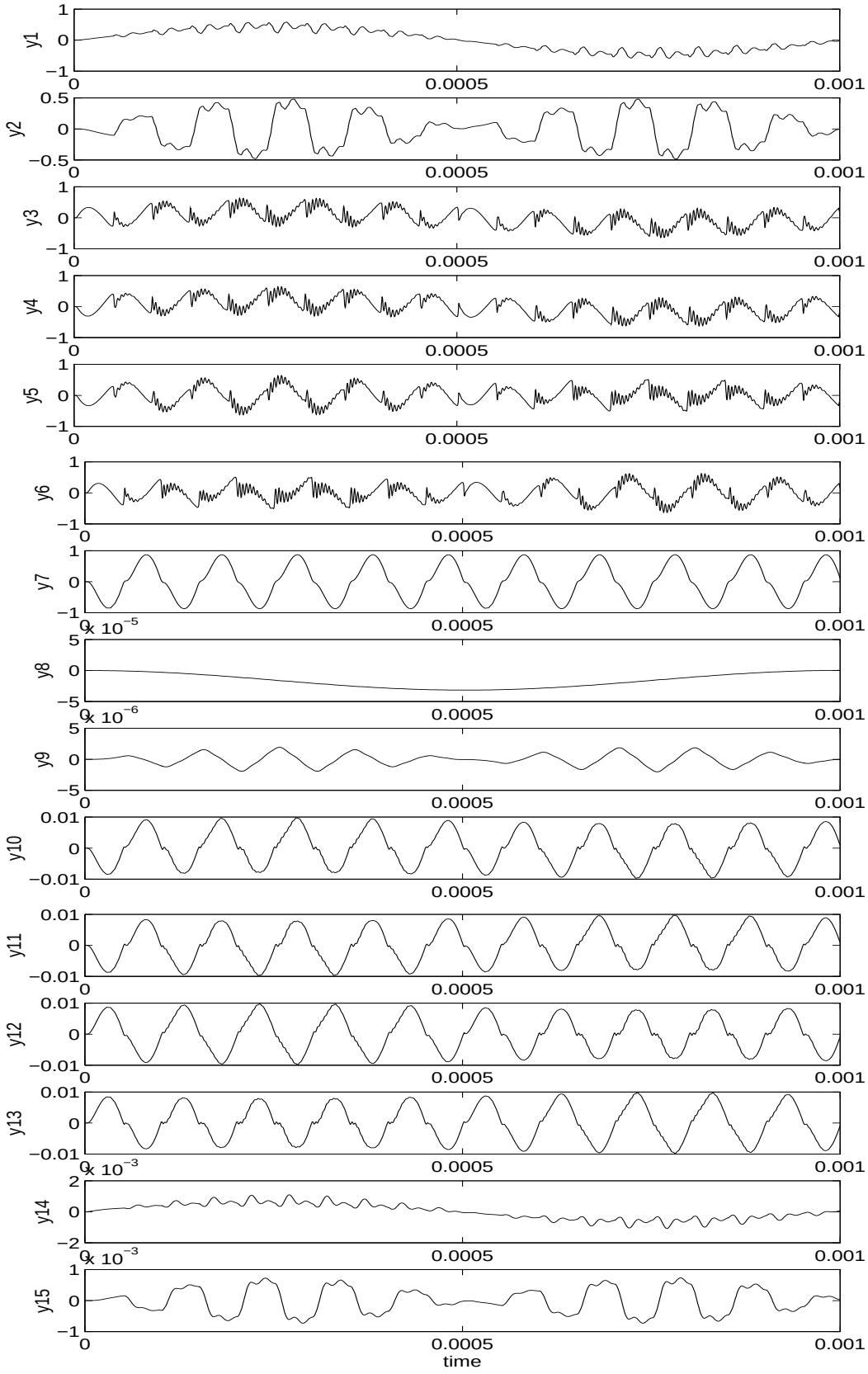

Figure 4.17: Solution profile: Ring Modulator problem.

Table 4.4: Results for Kaps problem using DIMSIMs with $s = p$.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	MAX GL ER
DIMSIM A	10^{-1}	28	0	16	28	44	4.8×10^{-2}
	10^{-2}	38	0	16	35	71	1.4×10^{-2}
	10^{-3}	60	2	23	55	108	2.5×10^{-3}
	10^{-4}	89	4	21	71	167	2.9×10^{-4}
	10^{-5}	134	2	24	76	259	3.6×10^{-5}
	10^{-6}	219	2	33	94	431	7.3×10^{-6}
	10^{-7}	320	3	42	110	599	4.4×10^{-7}
	10^{-8}	321	3	44	117	579	1.3×10^{-7}
	10^{-9}	414	0	48	129	747	5.6×10^{-9}
	10^{-10}	555	0	59	134	983	5.5×10^{-10}
DIMSIM B	10^{-1}	28	0	16	28	44	4.8×10^{-2}
	10^{-2}	38	0	16	35	71	1.4×10^{-2}
	10^{-3}	61	2	21	55	110	2.5×10^{-3}
	10^{-4}	91	4	22	68	171	2.9×10^{-4}
	10^{-5}	141	2	27	81	273	7.1×10^{-5}
	10^{-6}	221	2	34	92	435	7.3×10^{-6}
	10^{-7}	267	2	38	111	525	4.4×10^{-7}
	10^{-8}	323	2	45	114	637	1.6×10^{-7}
	10^{-9}	431	1	54	114	855	2.0×10^{-8}
	10^{-10}	593	1	68	136	1177	2.0×10^{-9}
DIMSIM C	10^{-1}	28	0	16	28	44	4.8×10^{-2}
	10^{-2}	38	0	16	35	71	1.4×10^{-2}
	10^{-3}	60	1	19	52	108	2.5×10^{-3}
	10^{-4}	113	5	26	84	162	2.9×10^{-4}
	10^{-5}	159	10	26	109	308	3.6×10^{-5}
	10^{-6}	225	9	31	133	443	3.9×10^{-6}
	10^{-7}	367	3	47	126	728	9.7×10^{-7}
	10^{-8}	395	8	53	247	728	1.5×10^{-7}
	10^{-9}	502	21	57	336	952	5.6×10^{-9}
	10^{-10}	673	22	72	428	1284	5.5×10^{-10}
DIMSIM D	10^{-1}	28	0	16	28	44	4.8×10^{-2}
	10^{-2}	38	0	16	35	71	1.4×10^{-2}
	10^{-3}	65	4	21	56	119	2.5×10^{-3}
	10^{-4}	91	4	21	70	171	2.9×10^{-4}
	10^{-5}	145	9	26	94	281	4.8×10^{-5}
	10^{-6}	216	2	31	87	425	3.9×10^{-6}
	10^{-7}	363	5	43	111	720	4.4×10^{-7}
	10^{-8}	348	7	44	182	648	5.7×10^{-8}
	10^{-9}	425	7	51	178	809	5.6×10^{-9}
	10^{-10}	582	4	64	160	1138	6.2×10^{-10}

Table 4.5: Results for Oregonator problem using DIMSIMs with $s = p$.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
DIMSIM A	10^{-3}	433	26	200	404	727	3.7×10^{1}
	10^{-4}	823	24	276	724	1318	2.1×10^{0}
	10^{-5}	1289	22	322	1079	1878	4.6×10^{-1}
	10^{-6}	1833	32	409	1497	2842	2.3×10^{-3}
	10^{-7}	1869	6	393	1511	2065	5.2×10^{-3}
	10^{-8}	2645	4	502	2095	2864	4.7×10^{-4}
	10^{-9}	3807	11	684	2999	4078	4.8×10^{-5}
	10^{-10}	5567	24	966	4364	6004	6.4×10^{-6}
DIMSIM B	10^{-3}	432	41	206	416	764	4.8×10^{1}
	10^{-4}	837	18	277	727	1329	1.2×10^{0}
	10^{-5}	1158	8	337	973	1867	2.0×10^{-1}
	10^{-6}	1758	5	417	1423	2928	2.2×10^{-2}
	10^{-7}	1833	2	434	1483	2901	1.1×10^{-2}
	10^{-8}	2654	1	531	2112	4169	1.1×10^{-3}
	10^{-9}	3752	2	691	2954	5866	4.4×10^{-5}
	10^{-10}	5482	0	949	4262	8541	6.9×10^{-6}
DIMSIM C	10^{-3}	463	35	209	434	784	5.2×10^{1}
	10^{-4}	975	23	293	829	1468	4.6×10^{0}
	10^{-5}	1412	43	332	1215	2061	1.2×10^{-1}
	10^{-6}	2042	45	419	1710	3065	2.0×10^{-2}
	10^{-7}	2147	57	418	1807	3105	2.6×10^{-4}
	10^{-8}	2886	59	534	2378	4101	3.5×10^{-5}
	10^{-9}	4054	76	724	3292	5850	6.2×10^{-6}
	10^{-10}	5872	65	1006	4715	8579	8.7×10^{-7}
DIMSIM D	10^{-3}	512	46	224	496	844	2.4×10^{1}
	10^{-4}	790	16	267	679	1285	6.6×10^{0}
	10^{-5}	1249	23	317	1055	1898	1.1×10^{-1}
	10^{-6}	1730	5	381	1398	2733	1.9×10^{-2}
	10^{-7}	1896	23	415	1554	2587	3.1×10^{-4}
	10^{-8}	2718	33	516	2189	3663	3.7×10^{-5}
	10^{-9}	3791	10	682	2995	5126	5.9×10^{-6}
	10^{-10}	5523	9	951	4310	7500	7.8×10^{-7}

Table 4.6: Results for Robertson problem using DIMSIMs with $s = p$.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
DIMSIM A	10^{-2}	105	0	65	101	158	3.0×10^{-5}
	10^{-3}	181	1	88	165	287	1.2×10^{-5}
	10^{-4}	347	3	132	259	505	1.9×10^{-6}
	10^{-5}	589	3	142	287	839	3.2×10^{-7}
	10^{-6}	998	3	174	381	1431	6.3×10^{-8}
	10^{-7}	951	1	166	414	953	1.4×10^{-9}
	10^{-8}	1361	1	235	475	1363	2.0×10^{-10}
	10^{-9}	1975	2	336	597	1978	3.2×10^{-11}
	10^{-10}	2837	3	482	761	2842	5.0×10^{-12}
DIMSIM B	10^{-2}	105	0	65	101	158	3.0×10^{-5}
	10^{-3}	181	1	88	165	287	1.2×10^{-5}
	10^{-4}	352	3	127	253	535	1.8×10^{-6}
	10^{-5}	592	3	129	298	894	3.1×10^{-7}
	10^{-6}	1007	3	176	396	1531	6.2×10^{-8}
	10^{-7}	975	2	200	414	1107	1.4×10^{-9}
	10^{-8}	1391	2	256	534	1602	2.0×10^{-10}
	10^{-9}	2004	3	342	598	2317	3.2×10^{-11}
	10^{-10}	2867	5	487	827	3341	4.8×10^{-12}
DIMSIM C	10^{-2}	105	0	65	101	158	3.0×10^{-5}
	10^{-3}	181	1	88	165	287	1.2×10^{-5}
	10^{-4}	362	10	127	263	547	8.3×10^{-6}
	10^{-5}	603	9	128	307	907	7.3×10^{-7}
	10^{-6}	997	3	174	379	1519	1.4×10^{-8}
	10^{-7}	1325	33	238	733	1388	1.6×10^{-10}
	10^{-8}	1874	51	332	1014	1970	2.4×10^{-11}
	10^{-9}	2500	59	435	1172	2636	3.7×10^{-12}
	10^{-10}	3747	83	649	1734	3993	1.8×10^{-12}
DIMSIM D	10^{-2}	105	0	65	101	158	3.0×10^{-5}
	10^{-3}	181	1	88	165	287	1.2×10^{-5}
	10^{-4}	356	3	113	252	549	5.8×10^{-7}
	10^{-5}	592	3	113	304	917	7.1×10^{-8}
	10^{-6}	1008	6	177	402	1577	1.4×10^{-9}
	10^{-7}	997	3	174	445	1006	1.6×10^{-10}
	10^{-8}	1400	2	241	540	1436	2.3×10^{-11}
	10^{-9}	2024	5	345	621	2088	3.7×10^{-12}
	10^{-10}	2872	6	488	781	2990	5.6×10^{-13}

Table 4.7: Results for van der Pol problem using DIMSIMs with $s = p$.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
DIMSIM A	10^{-1}	359	114	292	347	1121	2.8×10^{-1}
	10^{-2}	480	106	362	450	1157	2.9×10^{-2}
	10^{-3}	808	59	590	732	1644	4.5×10^{-3}
	10^{-4}	1474	11	1048	1354	2816	1.2×10^{-3}
	10^{-5}	2349	9	1297	2049	4092	9.3×10^{-5}
	10^{-6}	3355	22	1622	2919	5509	9.1×10^{-6}
	10^{-7}	3775	46	1630	3229	5070	2.7×10^{-7}
	10^{-8}	6081	48	3038	5274	8511	1.2×10^{-7}
	10^{-9}	7472	63	3173	6530	9632	8.4×10^{-9}
	10^{-10}	10887	98	4591	9420	14245	1.9×10^{-9}
DIMSIM B	10^{-1}	359	114	292	347	1121	2.8×10^{-1}
	10^{-2}	480	106	362	450	1157	2.9×10^{-2}
	10^{-3}	809	59	589	735	1649	4.6×10^{-3}
	10^{-4}	1399	4	1028	1310	2757	5.0×10^{-4}
	10^{-5}	2324	7	1390	2071	4616	8.3×10^{-5}
	10^{-6}	3154	11	1704	2744	6285	4.9×10^{-6}
	10^{-7}	3539	13	1809	3068	7037	4.9×10^{-7}
	10^{-8}	5757	17	3053	4941	11471	1.5×10^{-7}
	10^{-9}	7326	20	3243	6020	14605	9.2×10^{-9}
	10^{-10}	10616	22	4531	8796	21163	1.5×10^{-9}
DIMSIM C	10^{-1}	359	114	292	347	1121	2.8×10^{-1}
	10^{-2}	480	106	362	450	1157	2.9×10^{-2}
	10^{-3}	798	54	570	723	1632	4.8×10^{-3}
	10^{-4}	1434	8	996	1264	2862	1.4×10^{-3}
	10^{-5}	3819	89	2268	3469	7353	1.5×10^{-6}
	10^{-6}	4826	95	2664	4239	9375	4.0×10^{-7}
	10^{-7}	5968	150	3185	5336	11627	4.2×10^{-9}
	10^{-8}	6862	146	3477	6043	13393	1.4×10^{-9}
	10^{-9}	9412	172	4171	8110	18451	8.9×10^{-11}
	10^{-10}	13495	260	5895	11667	26715	1.3×10^{-11}
DIMSIM D	10^{-1}	359	114	292	347	1121	2.8×10^{-1}
	10^{-2}	480	106	362	450	1157	2.9×10^{-2}
	10^{-3}	803	59	576	731	1634	4.8×10^{-3}
	10^{-4}	1744	44	1162	1601	3329	6.2×10^{-4}
	10^{-5}	2510	56	1307	2245	4787	3.8×10^{-6}
	10^{-6}	3811	14	1837	3140	7188	2.3×10^{-6}
	10^{-7}	4827	24	2230	3926	9080	3.5×10^{-7}
	10^{-8}	5344	36	2282	4423	9931	5.6×10^{-9}
	10^{-9}	7620	57	3218	5972	14199	8.3×10^{-10}
	10^{-10}	12445	337	5313	10253	23370	9.0×10^{-10}

Table 4.8: Results for Ring Modulator using DIMSIMs with $s = p$.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
DIMSIM A	10^{-2}	1797	349	963	1776	5629	1.2×10^{-1}
	10^{-3}	4523	658	1788	3877	10631	8.1×10^{-2}
	10^{-4}	9160	703	3033	6167	19771	1.9×10^{-2}
	10^{-5}	16467	716	4250	8809	33761	1.5×10^{-3}
	10^{-6}	17955	236	5096	9616	27090	1.3×10^{-4}
	10^{-7}	24903	147	5471	10730	32409	2.3×10^{-5}
	10^{-8}	35950	105	6708	12734	42354	3.0×10^{-6}
	10^{-9}	53227	63	24672	46159	74502	4.6×10^{-7}
	10^{-10}	81109	48	35338	69835	137565	3.8×10^{-8}
DIMSIM B	10^{-2}	1760	355	978	1740	5755	1.2×10^{-1}
	10^{-3}	4533	696	1813	3975	10761	8.1×10^{-2}
	10^{-4}	9091	762	2953	6244	19765	1.9×10^{-2}
	10^{-5}	16358	810	4209	8885	34117	1.4×10^{-3}
	10^{-6}	17153	151	4867	14289	34141	1.4×10^{-4}
	10^{-7}	24485	109	5716	19882	48884	2.2×10^{-5}
	10^{-8}	35819	102	7145	28685	71607	3.6×10^{-6}
	10^{-9}	52662	90	9503	41704	105268	5.3×10^{-7}
	10^{-10}	77345	92	18113	62060	154656	7.7×10^{-8}
DIMSIM C	10^{-2}	1985	365	1020	1911	5862	5.9×10^{-2}
	10^{-3}	4907	675	1943	4177	11349	8.3×10^{-2}
	10^{-4}	9267	719	2860	6216	20010	5.0×10^{-3}
	10^{-5}	16453	666	3927	8546	33883	9.4×10^{-4}
	10^{-6}	37316	907	8943	31911	72820	1.3×10^{-6}
	10^{-7}	41494	743	9194	34983	80470	3.7×10^{-7}
	10^{-8}	49170	507	9596	40576	95828	9.6×10^{-8}
	10^{-9}	55927	92	27205	48949	110539	3.8×10^{-8}
	10^{-10}	85929	137	31014	72776	169919	2.9×10^{-9}
DIMSIM D	10^{-2}	1909	354	958	1859	5692	4.9×10^{-2}
	10^{-3}	4915	742	1930	4266	11525	8.0×10^{-2}
	10^{-4}	9435	793	3015	6262	20471	4.8×10^{-3}
	10^{-5}	16436	851	3909	8767	34581	8.8×10^{-4}
	10^{-6}	17157	152	3653	7909	28413	1.4×10^{-5}
	10^{-7}	24482	93	4884	19707	38758	2.3×10^{-6}
	10^{-8}	35807	100	6566	28501	56948	3.6×10^{-7}
	10^{-9}	52642	91	9952	41829	84298	5.5×10^{-8}
	10^{-10}	77333	85	13862	60899	125252	7.9×10^{-9}

Chapter 5

New type 4 DIMSIMs

In this chapter we investigate type 4 DIMSIMs in which the diagonal elements of the A matrix are allowed to be different, and methods in which $A = \lambda I$ but the number of stages is more than the overall order.

5.1 Motivation

Theoretically, type 4 DIMSIMs in which $s = p$, seem very promising, since these methods are A-stable, the abscissae can be chosen inside the integration interval, the overall order of the method is equal to the stage order and the internal stages in these methods can be implemented in parallel. As seen in the last chapter these methods have been successfully implemented in a variable stepsize, variable order code and can be used to solve stiff problems. However, these methods are not efficient when compared to the well known codes RADAU5 and VODE. These methods take many steps to complete the integration of problems tested in the last chapter and this results in high computational costs even if the stages are considered to be done in parallel. This is most likely due to the fact that these methods have large error constants, as seen in Tables 3.4 and 3.5. It is seen that as orders increase, the size of the error constants increase considerably. Since we need to have computations from two steps with constant stepsize in order to

obtain the error estimates, their implementation is quite complicated. This gives us strong motivation for developing type 4 methods, where the error constants are much smaller and where error estimators for controlling the stepsize are available within a single step.

Here we consider two approaches. The first is to allow the diagonal elements of matrix A to vary, and the second approach is to keep the diagonal elements equal but to allow the methods to have more stages than the order. Using these approaches the size of error constants can be reduced.

5.2 Methods with different λ

Type 4 methods are for parallel implementation. However, the diagonal elements of matrix A do not need to be equal in order to take advantage of parallelism. Making the assumption that they are all equal simplifies the derivation a great deal, leading to a simple form of the stability polynomial and transformations which are used to derive the method coefficients. We consider type 4 methods with $s = p$ and in which the matrix A has the generalised form

$$A = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_s \end{bmatrix}.$$

The implementation of methods of this type will be slightly more complicated than the methods where the $A = \lambda I$. However, once the method coefficients have been derived, the details of the Nordsieck implementation are very similar to the case where $A = \lambda I$, and will be given in some detail later. The parallel computational cost for a method based on this choice of matrix A is not likely to be any higher than the case where the λ are chosen to be equal. Although each of the stages requires a different LU decomposition, these can be computed in parallel along with the stage iterations. Thus the overall parallel cost remains the same. The derivation and analysis of methods where matrix A takes this more general form is much more difficult since the transformations considered earlier do not apply

anymore. By using the order conditions directly in the next section, we see that it is possible to derive methods with smaller error constants.

5.2.1 A second order method

We use the order conditions directly to derive a method with $p = q = r = s = 2$, abscissae $c = [0, 1]^T$ and

$$A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad V = \begin{bmatrix} 1-v & v \\ 1-v & v \end{bmatrix}.$$

We substitute the second order approximations

$$\begin{aligned} e^z &= 1 + z + z^2/2 + O(z^3), \\ e^{cz} &= \begin{bmatrix} 1 \\ 1 + z + z^2/2 \end{bmatrix} + O(z^3), \end{aligned}$$

in the order condition (3.7). Dropping the terms involving $O(z^3)$ we obtain

$$w = e^{cz} - zAe^{cz} = \begin{bmatrix} 1 - z\lambda_1 \\ 1 + z(1-\lambda_2) + z^2(\frac{1}{2} - \lambda_2) \end{bmatrix}.$$

Using this expression of w in the second order condition (3.8), we get

$$\begin{aligned} & e^z w - zBe^{cz} - Vw \\ &= \begin{bmatrix} (1 - v - \lambda_1 v + \lambda_2 v - b_{11} - b_{12})z + (\frac{1}{2} - \lambda_1 - \frac{v}{2} + \lambda_2 v - b_{12})z^2 \\ (2 + \lambda_1 - \lambda_2 - v - \lambda_1 v + \lambda_2 v - b_{21} - b_{22})z + (2 - 2\lambda_2 - \frac{v}{2} + \lambda_2 v - b_{22})z^2 \end{bmatrix}. \end{aligned}$$

Since this is a second order method, the coefficients of z and z^2 should be zero. This gives the expressions for the elements of matrix B in terms of λ_1, λ_2 and v. For a second order method, the stability polynomial satisfies

$$\phi(q, e^z) = \det(wI - M(z)) = O(z^3),$$

where $M(z)$ is given by (3.15). Using this in a Mathematica program it is found that λ_1 satisfies

$$\lambda_1 = \frac{-3 + 8\lambda_2 - 5\lambda_2^2}{-2 + 3\lambda_2}. \tag{5.1}$$

If we put $\lambda_1 = \lambda_2 = \lambda$ and solve this last equation, we obtain $\lambda = \frac{3\pm\sqrt{3}}{2}$, which are precisely the values obtained for the second order method with $A = \lambda I$. Since we require positive values of λ_1 and λ_2, the last equation requires that

$$0.6 < \lambda_2 < \frac{2}{3} \quad \text{or} \quad \lambda_2 > 1.$$

The A-stability of a method with stability polynomial, $\phi(w, z)$, is equivalent to the statement that there do not exist complex numbers w and z such that

(i) $\phi(w, z) = 0$,

(ii) $\mathrm{Re}(z) \leq 0$,

(iii) $|w| > 1$.

By the maximum modulus principle, we can replace (ii) by $z = iy$. In order to investigate the possible values of λ_2 for A-stability, we can use a recursive argument based on the *Schur criterion* for a polynomial having all its zeros in the closed unit disc. We recall that a polynomial, all of whose zeros lie within the open unit disc in the complex plane, is called a Schur polynomial [64]. By a well-known property of Schur polynomials we can reduce these polynomials to lower order polynomials, which are also Schur polynomials [64]. Thus, the value of λ_2 needs to be chosen to ensure that the reduced stability polynomial is always positive. In this case the reduced polynomial is a quadratic whose coefficients a, b and c are polynomials in λ_2. For A-stability we need to ensure that this quadratic is always positive for the chosen value of λ_2. The simplest way of doing this is by choosing λ_2 such that $a > 0$, $b > 0$ and $c > 0$. This gives the following intervals of λ_2 for an A-stable method

$$0.5740 < \lambda_2 < \frac{2}{3} \quad \text{or} \quad 0.6712 < \lambda_2 < 3.03815.$$

Consequently, the values of λ_2 which lead to A-stable methods fall in the interval

$$0.6 < \lambda_2 < \frac{2}{3} \quad \text{or} \quad 1.0 < \lambda_2 < 3.03815.$$

The error constant of the method is given by

$$C = \frac{23}{12} - 5\lambda_2 + \frac{5}{2}\lambda_2^2.$$

Values of λ_2 near the zeros of this quadratic will give methods with small error constants. The zero which falls in the A-stability interval is about 1.48. Choosing $\lambda_2 = 1.5$, we get the A-stable method with the following coefficient matrices.

$$A = \begin{bmatrix} \frac{9}{10} & 0 \\ 0 & \frac{3}{2} \end{bmatrix}, \qquad B = \begin{bmatrix} \frac{63}{40} & -\frac{21}{40} \\ \frac{103}{40} & -\frac{9}{8} \end{bmatrix}, \quad V = \begin{bmatrix} \frac{9}{8} & -\frac{1}{8} \\ \frac{9}{8} & -\frac{1}{8} \end{bmatrix}.$$

This method has an error constant of $\frac{1}{24} = 0.04167$, which is much smaller than the error constants, 0.2484, 4.0817, of the two second order methods in which $A = \lambda I$.

5.2.2 Nordsieck representation

The Nordsieck representation of type 4 DIMSIMs outlined in Chapter 4 applies to the methods with this generalised matrix A, with a few changes. The matrix $\widetilde{U}$ is now given by

$$\widetilde{U} = D - AE$$

where the matrices D and E are defined by (4.8) and (4.9) respectively. For the case where $c_1 = 0$ the first row of the matrix $\widetilde{B}$ becomes
$[b_{11}, b_{12}, \ldots, b_{1,s-1}, b_{1,s} + \lambda_1]$ while the other details concerning the $\widetilde{B}$ matrix remain exactly the same. The error estimators also remain unchanged. The other important change involves the stage equations and it is seen that (4.45) now becomes

$$Y_i = \lambda_i h f(Y_i) + \psi_i, \qquad i = 1, 2, \ldots, s, \tag{5.2}$$

and (4.46) changes to

$$G(Y_i) = -Y_i + \lambda_i h f(Y_i) + \psi_i, \qquad i = 1, 2, \ldots, s. \tag{5.3}$$

The modified Newton iteration scheme for the solution of $G(Y_i) = 0$, can now be stated as

$$\begin{aligned} M_i \Delta Y_i^{[k]} &= G(Y_i^{[k]}), \qquad (5.4) \\ Y_i^{[k+1]} &= Y_i^{[k]} + \Delta Y_i^{[k]}, \qquad k = 0, 1, \ldots, \end{aligned}$$

where, $M_i = I - \lambda_i hJ$, is the iteration matrix for stage i. It is now apparent that each stage requires its own LU decomposition. However, these can be done in parallel, so the overall cost will not increase.

As the second order method derived above illustrates, it is possible to derive type 4 methods with much smaller error constants, if we allow the diagonal elements of matrix A to vary. However, the derivation of higher order methods of this type, using the procedure used above, is much more difficult, especially the part involving the verification of A-stability, and unless we can find a systematic procedure for this derivation it is not practical. Hence, we do not consider higher order methods based on this approach in this thesis.

5.3 Type 4 DIMSIMs with $s = p + 1$

We consider the derivation of A-stable type 4 methods in which $A = \lambda I$, but λ does not satisfy (3.27). By not requiring this condition to hold the methods proposed have order one less than the number of stages. The stage order of such methods can still be equal to the order. We hope to be able to control the magnitude of the error constants by the appropriate choice of λ.

5.3.1 A first-order method with 2 stages

Consider a method with coefficient matrices, $A = \lambda I$, $\widetilde{U}$, $\widetilde{B}$, $\widetilde{V}$, and $c = [0, 1]^T$. Using the Nordsieck vector as the external stage vector, we have $w = [1, z]^T$. Using the order conditions given by (3.7) and (3.8) we have

$$\begin{aligned} e^{cz} &= z\lambda e^{cz} + \widetilde{U}w + O(z^2), \\ e^{z}w &= z\widetilde{B}e^{cz} + \widetilde{V}w + O(z^2). \end{aligned}$$

Using these we get

$$\widetilde{U} = \begin{bmatrix} 1 & -\lambda \\ 1 & 1-\lambda \end{bmatrix}, \qquad \widetilde{B} = \begin{bmatrix} b_1 & b_2 \\ 1-q & q \end{bmatrix}, \qquad \widetilde{V} = \begin{bmatrix} 1 & v \\ 0 & 0 \end{bmatrix},$$

where

$$b_1 + b_2 + v = 1. \tag{5.5}$$

Using the stability matrix for the method, $M(z) = \widetilde{V} + \frac{z}{1-\lambda z}\widetilde{B}\widetilde{U}$, we obtain

$$M(\infty) = \widetilde{V} - \frac{\widetilde{B}\widetilde{U}}{\lambda}.$$

For perfect damping at infinity, we require that the characteristic polynomial

$$\begin{aligned} p(w) &= \det(wI - M(\infty)) \\ &= \left(1 - \frac{b_2}{\lambda} + \frac{b_1 q}{\lambda^2} + \frac{b_2 q}{\lambda^2} - \frac{q}{\lambda} + \frac{v}{\lambda}\right) \\ &\quad + \left(-2 - \frac{b_1}{\lambda} + \frac{b_2}{\lambda} + \frac{q}{\lambda}\right) w + w^2 \end{aligned}$$

satisfies $p(w) \to 0$ as $w \to 0$. Hence, we set the constant term and the coefficient of w equal to zero. Solving these two equations for b_1 and b_2 gives

$$\begin{aligned} b_1 &= 2\lambda - \lambda^2 - q - \lambda q + q^2 - \lambda v, \\ b_2 &= \lambda^2 + \lambda q - q^2 + \lambda v, \end{aligned}$$

and adding them together gives

$$b_1 + b_2 = 2\lambda - q.$$

Using this equation with (5.5) and solving for q gives

$$q = -1 + 2\lambda + v.$$

Putting these together we have

$$\widetilde{B} = \begin{bmatrix} 2 - 3\lambda + \lambda^2 - 3v + 2\lambda v + v^2 & -1 + 3\lambda - \lambda^2 + 2v - 2\lambda v - v^2 \\ 2 - 2\lambda - v & -1 + 2\lambda + v \end{bmatrix}.$$

Using the Schur criterion we obtain the region of A-stability as

$$\frac{3 - \sqrt{3}}{2} \leq \lambda \leq \frac{3 + \sqrt{3}}{2}. \tag{5.6}$$

When $\lambda = \frac{3\pm\sqrt{3}}{2}$ the method has an error constant equal to zero and reduces to the second order method which has previously been derived by Butcher [24] and discussed in Chapter 3. The error constant of this new method is given by

$$C = \frac{3}{2} - 3\lambda + \lambda^2.$$

Since this is a positive quadratic with zeros at $\frac{3\pm\sqrt{3}}{2}$, it is clear that the error constant can be kept small by selecting a value of λ near these zeros but inside the interval (5.6). The free parameter, v, can be chosen in any way, as it does not affect the size of the error constant. For example, by choosing $\lambda = \frac{3}{4}$, and $v = 0$, we have $A = \lambda I$ and the following method

$$\widetilde{B} = \begin{bmatrix} \frac{5}{16} & \frac{11}{16} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \qquad \widetilde{U} = \begin{bmatrix} 1 & -\frac{3}{4} \\ 1 & \frac{1}{4} \end{bmatrix}, \qquad \widetilde{V} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$

which has an error constant, $C = -\frac{3}{16}$. With the same value of λ but $v = 1 - \lambda$, the method has the same error constant but the $\widetilde{B}$ and $\widetilde{V}$ become

$$\widetilde{B} = \begin{bmatrix} 0 & \frac{3}{4} \\ \frac{1}{4} & \frac{3}{4} \end{bmatrix}, \qquad \widetilde{V} = \begin{bmatrix} 1 & \frac{1}{4} \\ 0 & 0 \end{bmatrix}.$$

In this last method the final internal stage is calculated in exactly the same way as the first external stage, in much the same way as stiffly accurate Runge-Kutta methods in which the final stage is calculated using exactly the same coefficients as the output solution.

Apart from arbitrary choices of v, as in the above examples, one can choose v to minimise some norm of the B matrix. This was investigated using the Matlab function $fmins$ after having chosen λ. Using $\lambda = \frac{7}{10}$ it was found that the value of v near 0 which minimises the infinity norm of matrix B is close to $-\frac{1}{20}$. Using these the we get

$$\widetilde{B} = \begin{bmatrix} \frac{189}{400} & \frac{231}{400} \\ \frac{13}{20} & \frac{7}{20} \end{bmatrix}, \qquad \widetilde{V} = \begin{bmatrix} 1 & \frac{-1}{20} \\ 0 & 0 \end{bmatrix}.$$

This method has an error constant equal to 0.11, a value much smaller than the error constant of the implicit Euler method. In practice it is expected to perform better than the implicit Euler method, if the two stages are computed in parallel. Of course, in serial computation this new method is more costly because of the extra stage but we do not see this as a disadvantage in a parallel implementation.

5.3.2 Higher order methods

Using the stability polynomial and the numerical investigation outlined in Sections 3.2.1-3.2.2, the intervals of λ required for A-Stability of the new methods have been determined. These are shown in Table 5.1. For $p = 8$ and for $p > 10$ there are no A-stable methods.

p	Interval for λ	λ for $s = p$
1	$[\frac{3-\sqrt{3}}{2}, \frac{3+\sqrt{3}}{2}]$	$\frac{3\pm\sqrt{3}}{2}$
2	$[0.576, 3.833]$	1.2101383127
3	$[0.875, 1.9449]$	1.9442883555
4	$[1.053, 2.665]$	1.3012832613
5	$[1.052, 1.8059]$	1.8056866912
6	$[1.265, 2.290]$	1.3521971029
7	$[1.211, 1.739]$	1.7368002358
8	-	-
9	$[1.548, 1.703]$	1.6956068006

Table 5.1: Intervals of λ for A-stability.

The intervals for λ have been obtained experimentally and there is some uncertainty about the cut-off values. However, from the perspective of getting smaller error constants, we need to choose values of λ that are close to the values of λ for the methods where $s = p$, which are given in the final column in Table 5.1. As the orders increase, the choice of λ becomes very sensitive to the size of the error constants, since the error constants are polynomials in λ of increasing orders. Table 5.2 shows a set of choices of λ and the corresponding error constants. Here λ values have been chosen so that the error constants decrease in magnitude as order increases.

p	λ	C_{p+1}
1	0.7	1.1×10^{-1}
2	1.2	2.1×10^{-2}
3	1.944	2.2×10^{-3}
4	1.3012	5.5×10^{-4}
5	1.80568	2.1×10^{-4}
6	1.352193	9.9×10^{-5}
7	1.7368	3.1×10^{-5}
9	1.6956068	7.3×10^{-6}

Table 5.2: Error constants for methods in which $s = p + 1$.

5.3.3 Choice of abscissae and free parameters

For these new methods, the abscissae remains a free parameter just as they do for the methods in which $s = p$. Therefore we can choose the abscissae as we like, as long as they are distinct. We have investigated three choices in which the abscissae are equally spaced in $[0, 1]$, $[-1, 1]$ and in $[-s + 2, 1]$. The choice of the abscissae determines the magnitude of the coefficients. When the abscissae is chosen to be in each of the three intervals mentioned, the coefficients have the largest magnitude in the first case for any particular order. The smallest magnitude of coefficients result in the last case. When these coefficients become very large their use in the solution of an initial value problem will result in rounding errors. So although it is preferable to keep the abscissae inside the integration interval, the resulting methods have very large coefficients as the order increases. These high order methods will be of no practical use due to the influence of rounding errors.

It is possible to choose abscissae to be unequally spaced in $[0, 1]$, for example, they can be based on the roots of the shifted Chebyschev polynomials of the second kind, as in [26]. However, we have no reason to think that this choice will be in any way superior to the equally spaced choice. Hence, we have not considered this in this thesis.

The other remaining free parameters are $\tilde{v}_2, \tilde{v}_3, \ldots, \tilde{v}_s$, which come from the first

row of matrix $\widetilde{V}$. We chose these $\tilde{v}_i$, $i = 2, \ldots, s$, in such a way that we minimise the maximum norm of the B matrix. We used the *Matlab* function $fmins$ to do this minimisation.

After having chosen the abscissae, c, we use the values of λ listed in Table 5.2 and the values of $\tilde{v}$ which minimises the coefficients of the B matrix, to derive the methods which are listed in Appendix A. A *Mathematica* program was used for this purpose.

In an implementation of these methods we need to have these coefficients calculated to the highest possible precision. We investigate the effect of the choice of the abscissae on computations later in this chapter.

5.3.4 Error estimation for stepsize control

For these new methods, one can find an estimate of $h^{p+1}y^{(p+1)}(x_n) + O(h^{p+2})$ within a single step by taking a linear combination of the stage derivatives in much the same way as the calculation of the final component of the external stage vector for methods in which $s = p$. For example, for the first order methods listed in the previous section, we have

$$h^2 y^{(2)}(x_n) + O(h^3) = -hf(Y_1) + hf(Y_2),$$

and for a second order method with three stages and $c = [0, \frac{1}{2}, 1]^T$, the error estimator is

$$h^3 y^{(3)}(x_n) + O(h^4) = 4hf(Y_1) - 8hf(Y_2) + 4hf(Y_3).$$

For $p = 3$ and $c = [-1, 0, 1]^T$ the error estimator becomes

$$h^3 y^{(3)}(x_n) + O(h^4) = hf(Y_1) - 2hf(Y_2) + hf(Y_3).$$

For a method of order p, with $s = p + 1$ stages, the error estimator is given by

$$h^{p+1} y^{(p+1)}(x_n) + O(h^{p+2}) = \sum_{i=1}^{s} b_i hf(Y_i), \tag{5.7}$$

where the weightings, b_i, are numerically equal to the $\tilde{b}_{s+1,i}$ of the methods in which $s = p$ and the same c, and satisfy (4.21). The weightings for error estimation are given with the method coefficients in Appendix A. This procedure for obtaining error estimates is much simpler to implement, as this estimate is available within a single step, unlike the methods with $s = p$ in which one needs to calculate at least two estimates of the solution before any error estimate can be obtained. Using these error estimates these methods can be implemented in a variable stepsize code. The other details concerning the variable stepsize implementation are exactly the same as for methods in which $p = s$ as discussed in the last chapter.

5.3.5 Error estimates for order control

In order to implement these methods in a variable order code we require an estimate of the higher order error, $h^{p+2}y^{(p+2)}(x_n)$. To obtain this estimate we consider the following modification to the methods which we assume has been implemented using variable stepsize. Let us examine the output stage vector $\tilde{y}^{[n]} = [y(x_n), hy'(x_n), \ldots, h^p y^{(p)}(x_n)]^T$ for its error terms. Since there are s stages it is more convenient to use s instead of p. Each of the output stages, except the first, can be expressed as:

$$\tilde{y}_k^{[n]} = h^{k-1}y^{(k-1)}(x_n) + \phi_k h^s y^{(s)}(x_n) + O(h^{s+1}), \quad k = 2, 3, \ldots, s, \quad (5.8)$$

where $\phi_2, \phi_3, \ldots, \phi_s$ depend on the abscissae and can be calculated by considering (4.12) and leaving out the first equation resulting from this. Since the $\widetilde{V}$ matrix has all rows of zeros except the first, we have

$$\sum_{i=1}^{s} \tilde{b}_{ki} h f(Y_i) = h^{k-1}y^{(k-1)}(x_n) + \phi_k h^s y^{(s)}(x_n) + O(h^{s+1}), \quad k = 2, 3, \ldots, s. \quad (5.9)$$

Substituting $f(Y_i) = y'(x_{n-1} + c_i h)$ in the last equation and using a Taylor series expansion we obtain

$$\phi_k = \sum_{i=1}^{s} \tilde{b}_{k,i} \frac{(c_i - 1)^{s-1}}{(s-1)!}, \quad k = 2, 3, \ldots, s,$$

where the $\tilde{b}_{k,i}$ satisfy the following conditions for $k = 2, 3, \ldots, s$:

$$\sum_{i=1}^{s} \frac{\tilde{b}_{k,i}(c_i - 1)^q}{q!} = \begin{cases} 0, & q = 0, 1, \ldots, s-2, \quad q \neq k-2, \\ 1, & q = k-2. \end{cases}$$

Then, if we change stepsize from h to rh, we want the components of the external stages, (5.8), to be rescaled correctly not only for the first term, which is the usual case, but also for the second term, $\phi_k h^s y^{(s)}(x_n)$, $k = 2, 3, \ldots, s$. This can be done using an estimate for $\epsilon^{[n]} = h^s y^{(s)}(x_n) + O(h^{s+1})$ and modifying the appropriate components of the external stage vector to $\hat{y}^{[n]}$, where we want to obtain

$$\hat{y}_k^{[n]} = r^{k-1} h^{k-1} y^{(k-1)}(x_n) + \phi_k r^s h^s y^{(s)}(x_n) + O(h^{s+1}), \quad k = 2, 3, \ldots, s, \quad (5.10)$$

using

$$\hat{y}_k^{[n]} = r^{k-1} \tilde{y}_k^{[n]} + \theta_k \epsilon^{[n]} + O(h^{s+1}), \quad k = 2, 3, \ldots, s. \quad (5.11)$$

Substituting (5.8) into this last equation and comparing it with (5.10), we obtain

$$\theta_k = \phi_k (r^s - r^{k-1}), \quad k = 2, 3, \ldots, s.$$

Substituting this expression for θ_k into (5.10), we obtain

$$\hat{y}_k^{[n]} = r^{k-1} \tilde{y}_k^{[n]} + \phi_k (r^s - r^{k-1}) \epsilon^{[n]} + O(h^{s+1}), \quad k = 2, 3, \ldots, s.$$

In an implementation of these methods, the external stage vector can either be directly modified as stated by this equation or the $\widetilde{B}$ matrix can be modified as outlined below. Using the last equation and

$$\tilde{y}_k^{[n]} = \sum_{i=1}^{s} \tilde{b}_{k,i} h f(Y_i) + O(h^{s+1}), \quad k = 2, 3, \ldots, s,$$

$$\epsilon^{[n]} = h^s y^{(s)}(x_n) + O(h^{s+1}) = \sum_{i=1}^{s} b_i h f(Y_i),$$

where b_i, $i = 1, 2 \ldots, s$ are the weights for error estimation, we obtain

$$\hat{y}_k^{[n]} = \sum_{i=1}^{s} \left[r^{k-1} \tilde{b}_{k,i} + \phi_k (r^s - r^{k-1}) b_i \right] h f(Y_i) + O(h^{s+1}).$$

Hence, if we let the modified method be defined using $\widehat{B}$, then its elements are defined as

$$\begin{aligned}
\hat{b}_{1,i} &= \tilde{b}_{1,i}, \quad i = 1, 2, \ldots, s, \\
\hat{b}_{k,i} &= r^{k-1}\tilde{b}_{k,i} + \phi_k(r^s - r^{k-1})b_i, \quad i = 1, 2, \ldots, s, \quad k = 2, 3, \ldots, s.
\end{aligned}$$

Example 5.1 $p = 1, \quad \lambda = \frac{7}{10}, \quad c = [\ 0 \quad 1\ ]^T,$

$$\begin{aligned}
\widetilde{B} &= \begin{bmatrix} \frac{189}{400} & \frac{231}{400} \\ \frac{13}{20} & \frac{7}{20} \end{bmatrix}, & \widetilde{U} &= \begin{bmatrix} 1 & -\frac{7}{10} \\ 1 & \frac{3}{10} \end{bmatrix}, \\
\widetilde{V} &= \begin{bmatrix} 1 & -\frac{1}{20} \\ 0 & 0 \end{bmatrix}, & b &= [\ -1 \quad 1\ ], \\
\widehat{B} &= \begin{bmatrix} \frac{189}{400} & \frac{231}{400} \\ \frac{13}{20}r^2 & r - \frac{13}{20}r^2 \end{bmatrix}.
\end{aligned}$$

□

Example 5.2 $p = 2, \quad \lambda = \frac{6}{5}, \quad c = [\ 0 \quad \frac{1}{2} \quad 1\ ]^T,$

$$\begin{aligned}
\widetilde{B} &= \begin{bmatrix} \frac{89}{250} & -\frac{314}{125} & \frac{739}{250} \\ -\frac{8}{25} & \frac{16}{25} & \frac{17}{25} \\ \frac{21}{5} & -\frac{52}{5} & \frac{31}{5} \end{bmatrix}, & \widetilde{U} &= \begin{bmatrix} 1 & -\frac{6}{5} & 0 \\ 1 & -\frac{7}{10} & -\frac{19}{40} \\ 1 & -\frac{1}{5} & -\frac{7}{10} \end{bmatrix}, \\
\widetilde{V} &= \begin{bmatrix} 1 & \frac{1}{5} & -\frac{6}{5} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & b &= [\ 4 \quad -8 \quad 4\ ], \\
\widehat{B} &= \begin{bmatrix} \frac{89}{250} & -\frac{314}{125} & \frac{739}{250} \\ -\frac{8r^3}{25} & \frac{16r^3}{25} & r(1 - \frac{8r^2}{25}) \\ r^2\left(1 + \frac{16r}{5}\right) & r^2\left(-4 - \frac{32r}{5}\right) & r^2\left(3 + \frac{16r}{5}\right) \end{bmatrix}.
\end{aligned}$$

□

With this modification, we can obtain an estimate of $h^{p+2}y^{(p+2)}(x_n)$ by taking the difference of $h^{p+1}y^{(p+1)}(x_n)$ and $h^{p+1}y^{(p+1)}(x_{n-1})$ which are obtained from the error

estimates of two consecutive steps with constant stepsize, h. It has been observed that this modification has a mild effect on the error estimates for stepsize control.

5.3.6 Numerical experiments

In order to compare the performance of these methods, some of the problems outlined earlier in this thesis have been solved using an experimental solver based on the DIMSIMs proposed in this chapter. A parallel solver is only useful on sufficiently expensive problems as it is likely to suffer if too little time is spent in the parallel section of the solver. Thus, we desired a larger test problem in order to determine the speedup that can be achieved when the DIMSIMs are used in a parallel. The Medical Akzo Nobel problem from the CWI Testset was used for this purpose. This problem has a dimension of 400 and comes from the semi-discretisation of two partial differential equations. This problem was originally formulated by the Akzo Nobel Research Laboratory in the study of the penetration of radio-labelled antibodies into tumourous tissue. The Jacobian of this problem is a sparse matrix with a banded quin-diagonal structure. More details about this problem can be found in [63]. We have solved this problem using $Atol = Rtol = tol$ and $x \in [0, 20]$.

Some of the procedures such as stepsize control, order control and convergence control are exactly the same as for the methods in which $s = p$. The main difference is the way in which the solution and the error for stepsize and order control are calculated. In the following sections we refer to the three sets of methods based on different abscissae as:

- DIMSIM E: $c = [0, 1/(s-1), 2/(s-2), \ldots, 1]^T$, $s > 2$
- DIMSIM F: $c = [-1, -1 + 2/(s-1), \ldots, 1]^T$, $s > 2$
- DIMSIM G: $c = [-s + 2, -s + 3, \ldots, 0, 1]^T$, $s > 2$.

For DIMSIM E the abscissae are equally spaced in $[0, 1]$, for DIMSIM F they are equally spaced in $[-1, 1]$ except for the method of order 1, and for DIMSIM G

they are equally spaced in $[-s+2, 1]$. They all share the same method of order one with two stages which has the abscissae $c = [0, 1]$. Apart from the abscissae all the other details are same for these three sets of methods.

The aims of the numerical experimentation are to

1. determine the best choice of the maximum order,
2. determine the best choice of abscissae,
3. determine if these methods perform better than the methods in which $s = p$ that have been investigated in Chapters 3 and 4, and
4. compare the performance of these methods with RADAU5, and VODE.

Recently a parallel FORTRAN 77 code, PSIDE [72], based on the four stage Radau IIA method has been released. The four stages in this implementation are computed in parallel using the PILSRK iteration scheme that has been discussed in Chapter 2. We have used PSIDE to solve the Ring Modulator and the Medical Akzo Nobel problems.

5.3.6.1 Some programming details

In order to be able to solve problems in which the Jacobian is banded, we have incorporated the LU factorisations and the solutions of linear systems in which the coefficient matrices are stored in banded form. The LU factorisations are obtained using the LAPACK subroutines *DGBTRF* (banded matrices) and *DGETRF* (full matrices). The linear systems are solved using the subroutines *DGBTRS* (banded matrices) and *DGETRS* (full matrices).

Numerical testing was carried out on a a high performance Silicon Graphics Power Challenge GR computer which has 16, R10000 processors running IRIX 6.2. All CPU times given in the results are obtained using this computer. The following compilers were used:

MIPSpro FORTRAN 77 for RADAU5 and VODE,
MIPSpro Power FORTRAN 77 for PSIDE,
MIPSpro FORTRAN 90 and MIPSpro Power FORTRAN 90 for DIMSIMs.

Ideally we should have used the same compiler for all the codes. However, it was not possible to use the FORTRAN 90 compiler for all the codes. In particular, VODE and PSIDE did not work under FORTRAN 90. The comparison of the CPU times for RADAU5 using both the compilers showed almost no difference. Therefore, the use of different compilers should not be an issue.

Since the level of optimisation that is used by the compilers affects the CPU times, we have compiled all the codes with -O2 level of optimization.

In order to parallise the code we used the SGI parallel directive

```
!*$* ASSERT CONCURRENT CALL
```

This directive is only obeyed when the code is compiled with the MIPSpro Power Fortran 90 flag -pfa, otherwise this line is treated as a comment and the stages are computed in serial.

Although the primary source of parallelism for these methods is in the simultaneous Newton iterations of the stages, the smaller matrix-vector operations such as the calculation of the external stage can also be effectively parallised.

The CPU time in seconds has been calculated by the subroutine DTIME. CPU times have only been reported for the Ring Modulator and the Medical Akzo Nobel problems. For the smaller problems the CPU times were too small to be of any value in comparisons. On a timesharing system it has not been possible to ensure exclusive access to the processors during performance testing. Thus the programs have been run a number of times when the system load was low giving sufficiently consistent results.

For RADAU5 and VODE we give the sequential times and for PSIDE we give the parallel times. For DIMSIMs CPU(1) refers to the CPU time when the stages are calculated using a single processor while CPU(s) are the times when the stages are calculated in parallel by s processors which varies from 1 to 6 depending on the order of method being used.

5.3.6.2 Best choice of maximum order

As stated earlier, one of the problems with DIMSIMs is the magnitude of the method coefficients as the order of the methods increase. The magnitude become larger as the size of the interval in which the abscissae lie becomes smaller. Of the three methods considered for $p = 7$ and $s = 8$, it can be seen from Appendix A that DIMSIM E has some elements in the B matrix of the order of 10^8. The largest element in the B matrix of DIMSIM F is of the order of 10^7 while that of DIMSIM G is of the order of 10^3.

In a variable order code we need to decide on the maximum order of methods that will be available. In order to select the most efficient set of methods in a variable order code, we solved some of the problems for a range of stringent tolerances and by setting the maximum order available as 5, 6 or 7. All the other parameters were kept identical. The plots of the number of function evaluation in parallel versus the endpoint global error is given in Figure 5.1. It is observed that for Kaps problem the best performance is obtained by setting the maximum order to 6. The method of order 7 in DIMSIM E seems to have problems due to round off error. The methods of order 7 in DIMSIMs F and G also produce results with lower accuracies. For the other problems tested even the method of order 6 caused difficulties such as unreliable error estimates. In most cases it has been observed that when the integration was successful using the method of order 6, the accuracy obtained was much lower than if only the order 5 method was used. Thus, for the rest of the problems the maximum order available has been set to 5. Tables 5.6-5.10 show the statistics for the rest of the problems.

5.3.6.3 Best choice of abscissae

Comparing the results for solving the Kaps problem, the Robertson problem, the Oregonator problem, the Ring Modulator problem and the Medical Akzo Nobel problem we can draw some general conclusions about the performance of the type 4 DIMSIMs that we have implemented in this chapter. From the plots 5.2-5.7 it can be generally concluded that there is no significant difference in performance

between the methods. We recall that the major difference between the methods is the abscissae. For DIMSIM E the abscissae is equally spaced in $[0, 1]$, while that for DIMSIM F is equally spaced in $[-1, 1]$, and that for DIMSIM G is equally spaced in $[-s + 2, 1]$, where s is the number of stages. For every iteration these methods require one function evaluation. To investigate effect of the abscissae on the iterations required by the different stages, we have looked at the total number of iterations for every stage of the method of order 5 when $Atol = Rtol = 10^{-10}$ for the Medical Akzo Nobel problem. Table 5.3 shows the average number of iterations for the three methods for each of the stages 1 to 6.

METHOD	ABSCISSAE	S1	S2	S3	S4	S5	S6
E	$[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1]$	2.8	2.8	2.9	3.0	3.0	3.0
F	$[-1, -\frac{3}{5}, -\frac{1}{5}, \frac{1}{5}, \frac{3}{5}, 1]$	2.5	2.5	2.6	2.7	2.7	2.9
G	$[-4, -3, -2, -1, 0, 1]$	3.1	2.8	3.0	3.1	3.4	3.7

Table 5.3: Average number of iterations for the order 5 method for the Medical Akzo Nobel problem.

Some general observations can be made about the iterations for the different abscissae. The average number of iterations generally increase as the stage number increases for any abscissae. The final stage generally takes the most number of iterations to converge. This is to be expected since the abscissae represent the points where the internal stages represent the solutions and the last stage represents the solution at the furthest point. As the abscissae gets more spread out the difference between the average number of iterations amongst the stages increases. From the point of view of trying to balance out the number of iterations in order to have a good load balance amongst the processors, when the stages are solved in parallel, the abscissae in the interval $[0, 1]$ is probably a better choice. These observations are dependent on the starting values that have been used for the iterations. The starting values that were used is probably better suited to DIMSIM E.

5.3.7 Further discussion of results

The order control strategy used in this implementation is based on the estimates of $h^{p+2}y^{(p+2)}(x_n) + O(h^{p+3})$ which do not seem very reliable. Although the implementation seems to work reasonably well for most of the problems we tested, it is not robust. For the van der Pol and the Oregonator problems when there is a sharp change in the solution the order change strategy does not seem to work very well in that the order of the methods do not decrease quickly enough and as a result there are many rejected steps at higher orders. For the Ring Modulator problem the oscillatory nature of the solution caused difficulties for DIMSIMs at smaller tolerances. Although the step control seems to be working well, there are rejected steps and reducing the number of rejected steps will improve the performance of these methods. It has been noticed that the accuracy of the results obtained by using DIMSIMs E, F and G is usually lower than that requested. This did not seem to improve even when iterations were done more accurately. The accuracy obtained is related to the scaled derivatives in the external stage vector and the accuracy of these scaled derivatives could not be improved to get lower endpoint global errors.

Another factor which affects the computational cost is the starting values for the stage iterations. For sequential methods the converged values for one stage can be used for the starting values of the next values. However, when the stages are calculated in parallel, this is not possible. Thus, at the beginning of the step, there is a need for the starting values of all the stages in that step. The present implementation uses the Nordsieck vector of the past step for this purpose. The performance of these methods is likely to improve with better stage predictors than the current one.

The comparison of results obtained by using DIMSIMs A, B, C and D with those obtained by using DIMSIMs E, F and G show that the performance of DIMSIMs E, F and G is only slightly better. For a given tolerance DIMSIMs E, F and G generally take fewer steps, however, the accuracy of the results obtained is sometimes lower. Thus, the reduction in the magnitude of the error constants has not given a big improvement in performance. There are other factors which

affect the performance as discussed below. We notice that VODE takes many shorter steps to integrate a problem compared to RADAU5 and the computational experience with DIMSIMS suggests that DIMSIMs have inherited these smaller steps. However, smaller steps for a multistage method means higher computational cost. Thus the challenge will be an implementation in which the DIMSIMs take larger steps to reduce the computational cost.

The implementation of DIMSIMs discussed in this chapter do not compare favourably with the results for RADAU5 and VODE which are considered to be two of the most efficient stiff solvers. The number of function evaluations in serial computation for RADAU5 and VODE are less than the number of parallel function evaluations in DIMSIMs for all the problems we tested. For the Ring Modulator and the Medical Akzo Nobel problems we have tabulated the serial CPU times for RADAU5 and VODE and the parallel CPU times for DIMSIMs and PSIDE. For DIMSIMs we have tabulated the serial and parallel CPU times for the Ring Modulator and the Medical Akzo Nobel problem in Tables 5.9 and 5.10 respectively. To isolate the effect of the quality of the programming and the utilisation of the parallel machine, the code can be compared to itself using the speedup defined as

$$S = \frac{\text{the CPU time on one processor}}{\text{the CPU time on } n \text{ processors}}.$$

The maximum such speedup that can be attained is n. However, for a variable order code such as DIMSIMs with $s = p + 1$ and maximum order 5, the number of processors varies from two to six depending on the requested tolerance. These results show speedups in the range $1.8 - 2.5$. The higher values are obtained for more stringent tolerances. These speedups can be compared to the bounds provided by Amdahl's law which states that

$$S \leq A = \frac{P + Q}{P/n + Q}$$

where P is the time spent in the parallel section of the code when it runs on a single processor, Q is the time spent in the sequential section and n is the number of processors that can be applied in the parallel section. Table 5.4 shows

these statistics for a set of tolerances for DIMSIM E. The bounds, A, have been calculated using Amdahl's law assuming that all the steps have been taken by the method of maximum order attained for the given tolerance. This is justified as the lower order startup methods take only a small percentage of the total number of steps and take a much smaller percentage of the CPU times. The values of S obtained are good estimates to the expected bounds.

TOL	MAX ORD USED	% STEPS - MAX ORD	P	Q	A	S
10^{-4}	3	84	1.47	0.49	2.3	1.8
10^{-6}	4	89	2.46	0.89	2.4	2.0
10^{-8}	4	95	5.20	1.79	2.5	2.1
10^{-10}	5	96	11.23	3.39	2.8	2.4

Table 5.4: Speedups for DIMSIM E and the Medical Akzo Nobel problem.

One of the factors contributing to the poor performance of the DIMSIMs is the choice of problems. The standard stiff ODE test problems are not the best ones for testing parallel methods. A better selection of problems may yet show that the advantage of these parallel methods. Among the other contributing factors for the not so good parallel performance is the relatively high communication costs between processors on modern parallel computers with a small number of processors. Latency associates with starting and ending a communication becomes too difficult to absorb. The variable order strategy further complicates parallelism. A reduction of order incurs a loss of parallelism. Any change of order would incur a massive movement of data between processors. The order change strategy would need to be more sophisticated so as to take this into account. To overcome some of these problems would require a more sophisticated parallel coding compared to the compiler detection of parallelism that has been used here. In the present implementation speedups of only up to 2.5 have been observed but these are likely to improve and may give DIMSIMS better performance relative to serial methods as communication on parallel computers improve relative to computation.

5.3.8 Concluding remarks and future work

The computational experience gained with these methods suggests that the serial aspects of the methods are working well. For DIMSIMS that have been investigated in this thesis, the important quantities that are used are the error estimates. Some of the error estimation procedures are effective and the methods generally converge at the anticipated rates. Although we now have an effective the order control strategy, there is a need for a more sophisticated strategy which takes into account the communication costs between processors when order is varied.

We have investigated methods in which the V matrix has rank 1 or 2 only. Since we have observed a slight performance gain for rank 2 methods, there is a need to investigate methods with higher ranks. Although the standard assumption has been that the number of stages s is equal to the order p, we have also investigated the case $s = p + 1$. There is no reason why s should not even be greater. The generalisation that the diagonal elements of the A matrix are equal may not necessarily be the best choice. As we have been able to derive a second order method of two stages with different diagonal elements, we further need to investigate higher order methods of this type. Thus, there is a long way to go before we can be really sure which of these large class of methods give the best performance.

The development of efficient software for the solution of ODEs requires many years of experience. Our methods are new and the computational experience gained with them so far is very limited. It is hoped that further experience and understanding of these methods, along with a deeper understanding of the issues relating to parallel computation, will enable a more efficient implementation.

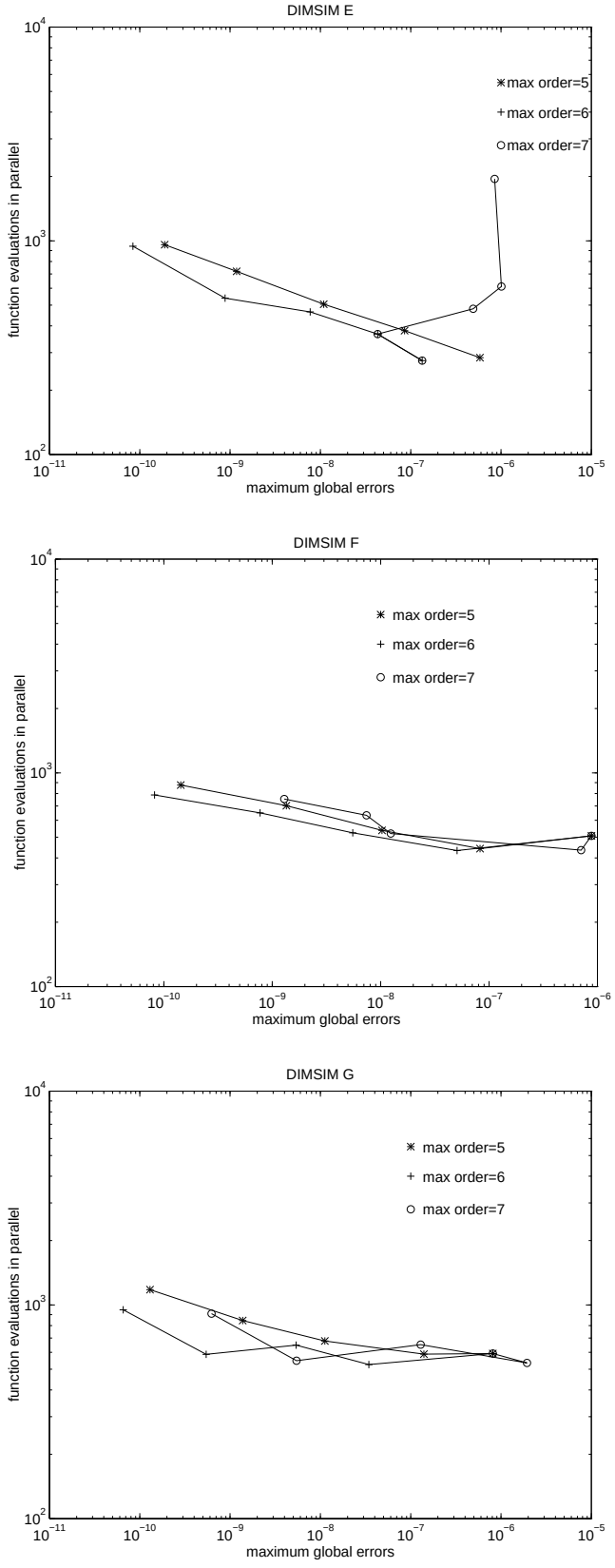

Figure 5.1: Results for Kaps problem with different maximum orders.

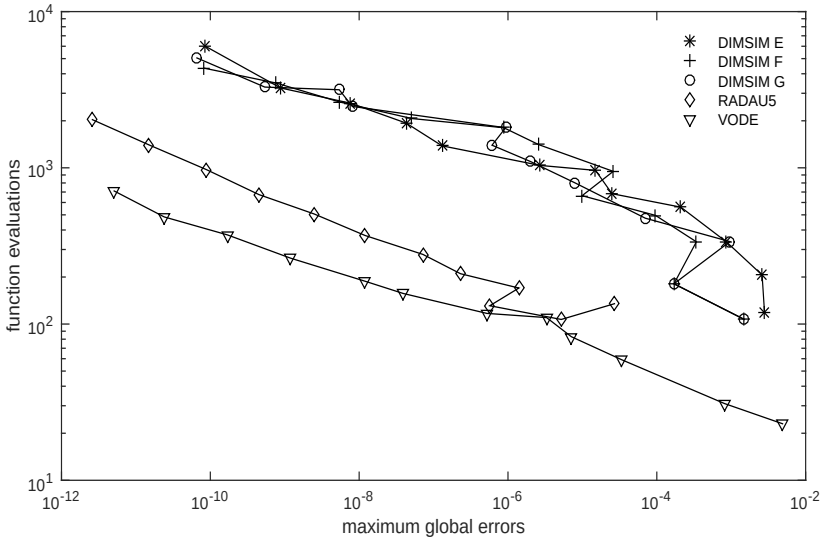

Figure 5.2: Results for Kaps problem using DIMSIMs with $s = p + 1$.

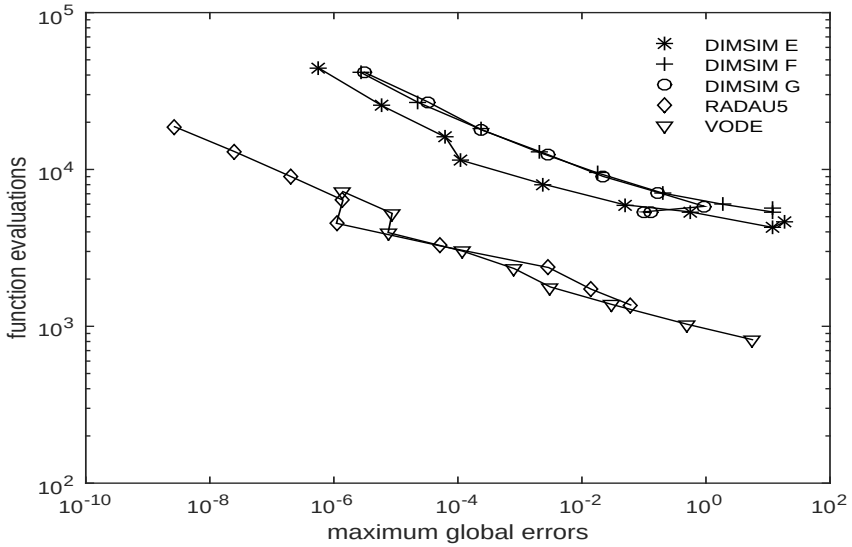

Figure 5.3: Results for the Oregonator problem using DIMSIMs with $s = p + 1$.

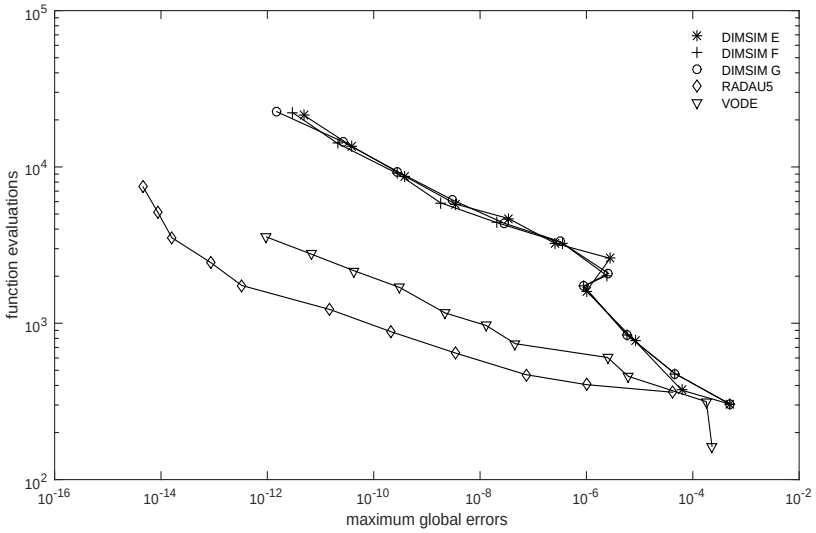

Figure 5.4: Results for the Robertson problem using DIMSIMs with $s = p + 1$.

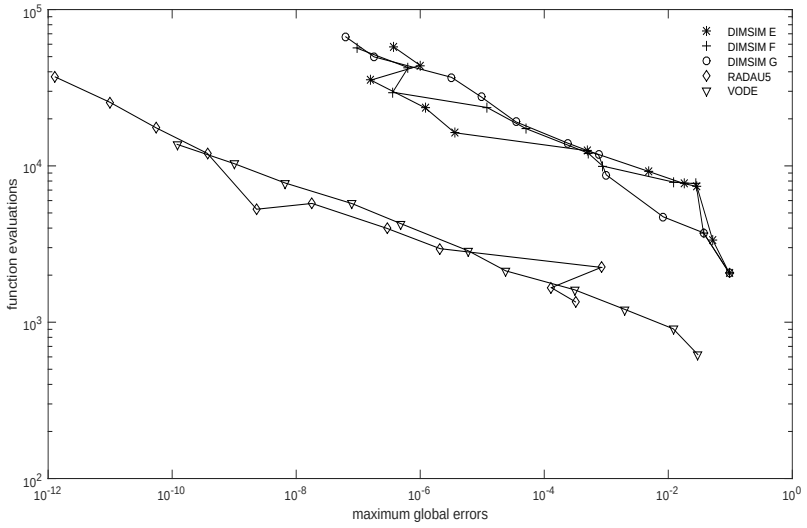

Figure 5.5: Results for the van der Pol problem using DIMSIMs with $s = p + 1$.

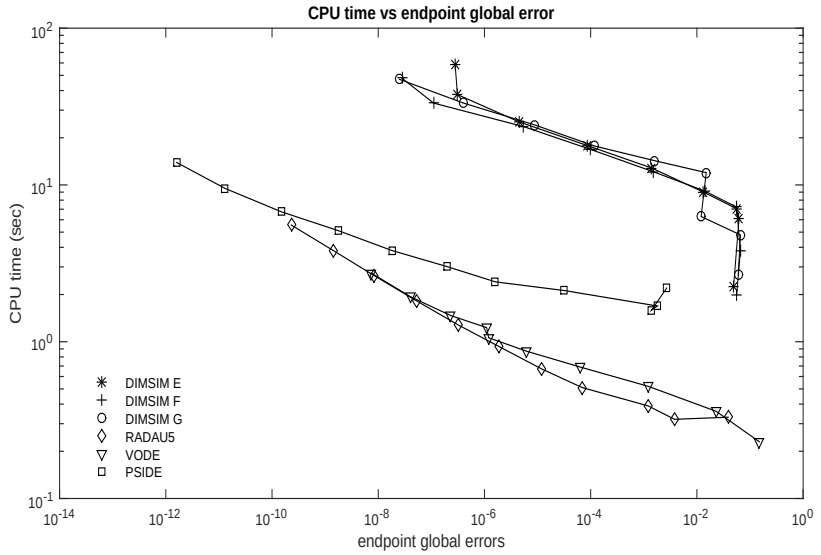

Figure 5.6: Results for the Ring Modulator problem using DIMSIMs with $s = p+1$.

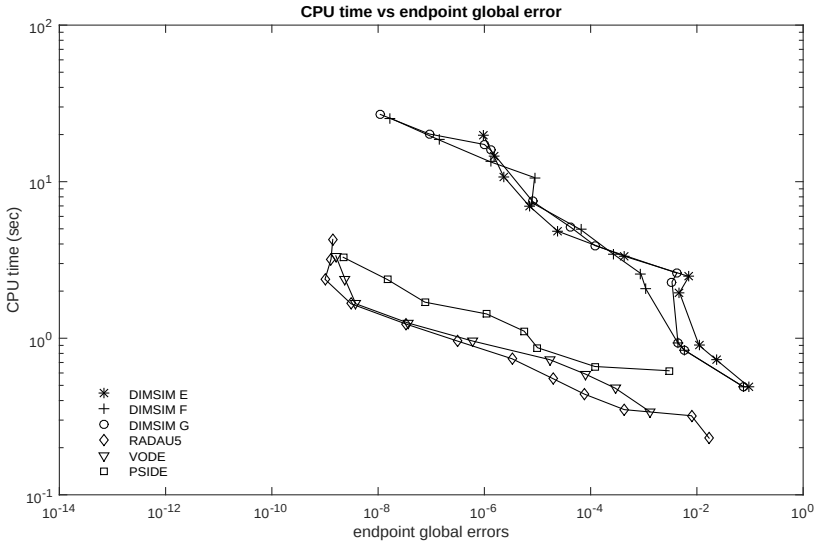

Figure 5.7: Results for the Medical Akzo Nobel problem using DIMSIMs with $s = p + 1$.

Table 5.5: Results for Kaps problem using DIMSIMs with $s = p+1$ and maximum order 6.

METHOD	TOL	TOT ST	REJ	JAC	LU	F-PAR	F-TOT	GL ER
DIMSIM E	10^{-1}	22	0	4	12	52	119	2.8×10^{-3}
	10^{-2}	31	2	7	19	78	206	2.6×10^{-3}
	10^{-3}	47	3	8	28	113	336	8.6×10^{-4}
	10^{-4}	60	3	8	35	154	560	2.1×10^{-4}
	10^{-5}	76	4	10	42	201	683	2.5×10^{-5}
	10^{-6}	99	6	13	55	258	963	1.5×10^{-5}
	10^{-7}	92	1	13	46	220	1036	2.7×10^{-6}
	10^{-8}	114	3	15	59	275	1383	1.3×10^{-7}
	10^{-9}	150	7	17	76	366	1932	4.3×10^{-8}
	10^{-10}	191	7	20	97	465	2589	7.7×10^{-9}
	10^{-11}	224	8	23	95	539	3244	8.8×10^{-10}
	10^{-12}	388	11	35	149	943	6034	8.4×10^{-11}
DIMSIM F	10^{-1}	20	0	5	14	48	107	1.5×10^{-3}
	10^{-2}	28	1	6	18	73	181	1.7×10^{-4}
	10^{-3}	46	3	7	27	122	336	3.4×10^{-4}
	10^{-4}	58	4	8	36	143	496	9.6×10^{-5}
	10^{-5}	72	3	10	40	189	663	9.9×10^{-6}
	10^{-6}	92	5	12	54	240	949	2.6×10^{-5}
	10^{-7}	134	7	16	67	355	1406	2.6×10^{-6}
	10^{-8}	153	8	19	75	407	1805	8.8×10^{-7}
	10^{-9}	172	6	19	85	433	2068	5.1×10^{-8}
	10^{-10}	209	7	22	92	524	2637	5.5×10^{-9}
	10^{-11}	267	9	26	112	650	3486	7.7×10^{-10}
	10^{-12}	329	8	30	134	788	4347	8.2×10^{-11}
DIMSIM G	10^{-1}	20	0	5	14	48	107	1.5×10^{-3}
	10^{-2}	28	1	6	18	73	181	1.7×10^{-4}
	10^{-3}	47	3	8	28	121	337	9.6×10^{-4}
	10^{-4}	58	4	11	36	149	476	7.0×10^{-5}
	10^{-5}	85	4	17	51	204	800	8.0×10^{-6}
	10^{-6}	115	7	20	73	305	1098	2.0×10^{-6}
	10^{-7}	136	4	24	76	342	1396	6.1×10^{-7}
	10^{-8}	176	4	29	106	427	1805	9.3×10^{-7}
	10^{-9}	227	5	38	132	566	2472	8.1×10^{-9}
	10^{-10}	250	9	24	126	648	3160	5.4×10^{-9}
	10^{-11}	248	8	24	111	587	3287	5.4×10^{-10}
	10^{-12}	385	14	34	185	949	5051	6.6×10^{-11}

Table 5.6: Results for Oregonator problem using DIMSIMs with $s = p + 1$.

METHOD	TOL	TOT ST	REJ	JAC	LU	F-PAR	F-TOT	GL ER
DIMSIM E	10^{-4}	526	79	274	443	1215	4664	1.9×10^{1}
	10^{-5}	476	67	249	390	1103	4268	1.2×10^{1}
	10^{-6}	600	68	300	476	1254	5361	5.6×10^{-1}
	10^{-7}	648	35	291	467	1349	5951	4.9×10^{-2}
	10^{-8}	884	46	320	565	1774	8057	2.4×10^{-3}
	10^{-9}	1281	43	401	681	2434	11478	1.1×10^{-4}
	10^{-10}	1767	23	436	735	3302	16077	6.3×10^{-5}
	10^{-11}	2794	26	542	931	5184	25457	6.0×10^{-6}
	10^{-12}	4860	105	848	1601	9234	44289	5.7×10^{-7}
DIMSIM F	10^{-4}	628	51	425	515	1584	5632	1.3×10^{1}
	10^{-5}	584	29	409	488	1457	5356	1.2×10^{1}
	10^{-6}	651	19	415	509	1539	5989	1.9×10^{0}
	10^{-7}	758	17	455	576	1747	7068	2.0×10^{-1}
	10^{-8}	991	18	475	658	2254	9492	1.8×10^{-2}
	10^{-9}	1317	17	501	728	2937	12885	2.0×10^{-3}
	10^{-10}	1844	14	574	852	3980	18084	2.3×10^{-4}
	10^{-11}	2728	15	645	982	5723	26709	2.3×10^{-5}
	10^{-12}	4239	41	847	1246	8705	41293	2.8×10^{-6}
DIMSIM G	10^{-4}	474	56	361	421	1606	5307	1.3×10^{-1}
	10^{-5}	482	40	362	427	1525	5361	9.8×10^{-2}
	10^{-6}	522	29	364	433	1568	5819	9.4×10^{-1}
	10^{-7}	647	24	380	494	1821	7137	1.7×10^{-1}
	10^{-8}	810	15	363	516	2216	9026	2.2×10^{-2}
	10^{-9}	1159	18	414	621	2924	12455	2.8×10^{-3}
	10^{-10}	1717	18	497	760	4043	17871	2.4×10^{-4}
	10^{-11}	2648	25	624	961	5836	26928	3.3×10^{-5}
	10^{-12}	4160	28	861	1248	8816	41638	3.1×10^{-6}

Table 5.7: Results for Robertson problem using DIMSIMs with $s = p + 1$.

METHOD	TOL	TOT ST	REJ	JAC	LU	F-PAR	F-TOT	GL ER
DIMSIM E	10^{-2}	71	4	47	64	143	304	5.0×10^{-4}
	10^{-3}	83	1	52	69	139	373	6.2×10^{-5}
	10^{-4}	156	1	65	101	269	770	8.2×10^{-6}
	10^{-5}	311	4	72	150	558	1606	1.0×10^{-6}
	10^{-6}	348	2	99	176	699	2603	2.8×10^{-6}
	10^{-7}	421	3	120	226	860	3242	2.6×10^{-7}
	10^{-8}	562	3	149	239	1072	4682	3.4×10^{-8}
	10^{-9}	641	8	174	265	1196	5781	3.5×10^{-9}
	10^{-10}	959	8	175	277	1794	8745	3.8×10^{-10}
	10^{-11}	1473	7	239	346	2744	13535	3.9×10^{-11}
	10^{-12}	2284	4	357	472	4362	21388	4.9×10^{-12}
DIMSIM F	10^{-2}	71	4	47	64	143	304	5.0×10^{-4}
	10^{-3}	92	1	66	85	173	477	4.6×10^{-5}
	10^{-4}	161	1	74	105	287	846	5.7×10^{-6}
	10^{-5}	296	4	83	169	582	1726	8.9×10^{-7}
	10^{-6}	344	4	96	178	719	2021	2.4×10^{-6}
	10^{-7}	426	4	122	224	880	3264	3.5×10^{-7}
	10^{-8}	532	4	150	234	1052	4396	2.1×10^{-8}
	10^{-9}	623	2	153	260	1252	5898	1.8×10^{-9}
	10^{-10}	948	4	164	274	1880	9067	2.8×10^{-10}
	10^{-11}	1465	4	233	330	2896	14138	2.1×10^{-11}
	10^{-12}	2283	2	358	467	4531	22250	3.0×10^{-12}
DIMSIM G	10^{-2}	71	4	47	64	143	304	5.0×10^{-4}
	10^{-3}	92	1	66	85	173	477	4.6×10^{-5}
	10^{-4}	161	1	94	145	287	846	5.7×10^{-6}
	10^{-5}	316	4	83	149	582	1726	8.9×10^{-7}
	10^{-6}	352	3	132	177	715	2084	2.7×10^{-6}
	10^{-7}	424	3	162	233	909	3333	3.2×10^{-7}
	10^{-8}	437	4	157	225	1036	4370	2.1×10^{-8}
	10^{-9}	629	2	151	241	1386	6162	3.0×10^{-9}
	10^{-10}	960	3	176	269	1985	9355	2.8×10^{-10}
	10^{-11}	1474	5	233	331	2952	14413	2.7×10^{-11}
	10^{-12}	2290	1	357	464	4549	22579	1.5×10^{-12}

Table 5.8: Results for van der Pol problem using DIMSIMs with $s = p + 1$.

METHOD	TOL	TOT ST	REJ	JAC	LU	F-PAR	F-TOT	GL ER
DIMSIM E	10^{-1}	286	71	196	254	836	1360	1.5×10^{-1}
	10^{-2}	383	74	290	354	1061	2071	9.6×10^{-2}
	10^{-3}	452	72	313	386	1438	3359	5.2×10^{-2}
	10^{-4}	586	74	363	487	2240	7383	2.9×10^{-2}
	10^{-5}	678	41	405	519	2380	7701	1.8×10^{-2}
	10^{-6}	811	24	451	630	2353	9222	4.8×10^{-3}
	10^{-7}	1133	17	540	793	3060	12499	5.0×10^{-4}
	10^{-8}	1513	20	589	909	3597	16296	3.6×10^{-6}
	10^{-9}	2280	17	732	1143	5080	23533	1.2×10^{-6}
	10^{-10}	2953	23	907	1347	7260	35368	1.6×10^{-7}
	10^{-11}	3673	29	960	1492	8856	43917	1.0×10^{-6}
	10^{-12}	4062	11	1161	1631	10498	57817	3.7×10^{-7}
DIMSIM F	10^{-1}	286	71	196	254	836	1360	1.5×10^{-1}
	10^{-2}	388	70	291	354	1072	2077	9.6×10^{-2}
	10^{-3}	462	84	310	390	1579	3731	3.7×10^{-2}
	10^{-4}	606	85	388	515	2369	7691	2.8×10^{-2}
	10^{-5}	648	43	402	505	2339	7885	1.2×10^{-2}
	10^{-6}	759	42	429	564	2380	9970	8.8×10^{-4}
	10^{-7}	983	5	444	632	2804	12074	5.1×10^{-4}
	10^{-8}	1445	5	537	823	3870	17300	5.1×10^{-5}
	10^{-9}	2306	38	621	920	6780	23627	1.2×10^{-5}
	10^{-10}	2974	3	724	1033	8327	29602	3.6×10^{-7}
	10^{-11}	3793	50	959	1355	9217	42526	6.4×10^{-7}
	10^{-12}	4066	63	1213	1689	10283	56672	9.5×10^{-8}
DIMSIM G	10^{-1}	286	71	196	254	836	1360	1.5×10^{-1}
	10^{-2}	380	64	289	350	1055	2067	9.6×10^{-2}
	10^{-3}	462	84	310	390	1579	3731	3.7×10^{-2}
	10^{-4}	604	77	437	526	1498	4722	8.3×10^{-3}
	10^{-5}	694	52	443	554	2858	8763	1.0×10^{-3}
	10^{-6}	800	42	461	597	3090	11778	7.5×10^{-4}
	10^{-7}	1022	32	487	704	3488	13978	2.4×10^{-4}
	10^{-8}	1485	33	584	864	4671	19123	3.5×10^{-5}
	10^{-9}	2237	41	785	1135	6487	27567	9.7×10^{-6}
	10^{-10}	2844	46	917	1286	8550	36848	3.2×10^{-6}
	10^{-11}	3019	63	1016	1460	9805	49831	1.8×10^{-7}
	10^{-12}	4305	83	1165	1801	13088	66938	6.3×10^{-8}

Table 5.9: Results for the Ring Modulator problem using DIMSIMs with $s = p+1$.

METHOD	TOL	TOT ST	REJ	JAC	LU	F-PAR	F-TOT	GL ER	CPU(1)	CPU(s)
DIMSIM E	10^{-5}	2245	263	1367	1943	11099	37867	4.9×10^{-2}	2.24	1.30
	10^{-6}	4484	269	1617	3201	20919	104395	6.2×10^{-2}	6.05	2.82
	10^{-7}	5272	238	1604	3342	22480	115836	5.5×10^{-2}	7.08	3.14
	10^{-8}	7880	313	1614	4351	26119	135024	1.3×10^{-2}	9.04	4.37
	10^{-9}	10781	302	1766	5175	35601	193283	1.4×10^{-3}	12.83	5.99
	10^{-10}	15111	370	1939	6302	46303	259917	8.7×10^{-5}	17.82	8.06
	10^{-11}	21926	410	2236	7579	63543	362003	4.5×10^{-6}	25.24	11.14
	10^{-12}	32532	388	2761	9981	95894	549595	3.1×10^{-7}	37.68	17.64
	10^{-13}	50694	397	3921	16138	147253	848098	2.8×10^{-7}	58.83	27.46
DIMSIM F	10^{-5}	2256	273	1446	1999	12041	34781	5.6×10^{-2}	1.98	1.29
	10^{-6}	3930	340	1374	2771	14529	54664	6.6×10^{-2}	3.78	1.94
	10^{-7}	5475	283	1627	3362	24415	117541	5.6×10^{-2}	7.22	3.42
	10^{-8}	7499	284	1724	4001	28186	139883	1.4×10^{-2}	9.15	3.80
	10^{-9}	10406	235	1787	4593	34767	178122	1.5×10^{-3}	12.15	5.11
	10^{-10}	14574	272	1949	5467	44667	237681	1.0×10^{-4}	16.84	7.30
	10^{-11}	21161	315	2210	6624	58823	320215	5.5×10^{-6}	23.53	10.59
	10^{-12}	30355	239	2682	7717	78973	440172	1.1×10^{-7}	33.29	15.04
	10^{-13}	44505	218	3535	9253	111314	623058	2.8×10^{-8}	47.96	20.53
DIMSIM G	10^{-5}	3067	499	1763	2866	17259	47664	6.0×10^{-2}	2.67	1.54
	10^{-6}	4863	529	1536	3587	20641	74423	6.6×10^{-2}	4.80	2.46
	10^{-7}	6883	456	1489	4259	22958	89888	1.2×10^{-2}	6.29	3.29
	10^{-8}	9573	436	1802	4856	42283	192333	1.5×10^{-2}	11.98	5.38
	10^{-9}	11919	351	1802	5179	46094	216646	1.6×10^{-3}	14.23	6.43
	10^{-10}	15301	287	1864	5495	51200	254073	1.2×10^{-4}	17.69	7.84
	10^{-11}	21643	352	2092	6483	62305	329440	8.9×10^{-6}	24.12	10.52
	10^{-12}	30600	267	2586	7673	79489	437845	4.0×10^{-7}	33.30	14.86
	10^{-13}	44390	288	3528	9411	106983	604864	2.5×10^{-8}	47.17	20.02

Table 5.10: Results for the Medical Akzo Nobel problem using DIMSIMs with $s = p + 1$.

METHOD	TOL	TOT ST	REJ	JAC	LU	F-PAR	F-TOT	GL ER	CPU(1)	CPU(s)
DIMSIM E	10^{-1}	117	15	54	105	325	537	7.5×10^{-2}	0.49	0.53
	10^{-2}	134	14	64	123	415	916	2.3×10^{-2}	0.73	0.66
	10^{-3}	178	9	68	149	432	1116	1.1×10^{-2}	0.90	0.81
	10^{-4}	233	13	78	191	884	2832	4.7×10^{-3}	1.96	1.07
	10^{-5}	300	12	93	189	1033	3526	7.1×10^{-3}	2.48	1.23
	10^{-6}	384	16	82	212	1066	4594	4.2×10^{-4}	3.35	1.67
	10^{-7}	560	15	81	221	1437	6520	2.4×10^{-5}	4.83	2.35
	10^{-8}	852	17	85	233	2059	9705	7.0×10^{-6}	6.99	3.26
	10^{-9}	851	19	108	287	2744	15361	2.3×10^{-6}	10.79	4.29
	10^{-10}	1211	21	121	327	3709	20925	1.5×10^{-6}	14.62	6.05
	10^{-11}	1747	22	141	351	4750	27363	9.7×10^{-7}	19.71	8.01
	10^{-12}	*	*	*	*	*	*	*	*	*
DIMSIM F	10^{-1}	117	15	54	105	325	537	7.5×10^{-2}	0.49	0.53
	10^{-2}	152	19	70	134	475	1071	5.8×10^{-3}	0.84	0.73
	10^{-3}	176	9	68	143	470	1202	4.4×10^{-3}	0.93	0.78
	10^{-4}	240	11	73	189	907	2874	1.1×10^{-3}	2.06	1.21
	10^{-5}	300	10	97	185	1049	3619	8.8×10^{-4}	2.59	1.45
	10^{-6}	370	14	84	200	1150	4763	2.7×10^{-4}	3.44	1.80
	10^{-7}	542	15	81	214	1583	6964	6.5×10^{-5}	4.94	2.51
	10^{-8}	823	15	78	220	2271	10381	7.8×10^{-6}	7.35	3.59
	10^{-9}	868	21	114	281	2850	14966	8.8×10^{-6}	10.62	4.46
	10^{-10}	1187	19	116	290	3435	18433	1.3×10^{-6}	13.39	5.73
	10^{-11}	1723	23	143	329	4478	25008	1.4×10^{-7}	18.49	7.83
	10^{-12}	2504	25	193	390	6012	33859	1.7×10^{-8}	25.44	10.98
DIMSIM G	10^{-1}	117	15	54	105	325	537	7.5×10^{-2}	0.49	0.53
	10^{-2}	152	19	70	134	475	1071	5.8×10^{-3}	0.84	0.73
	10^{-3}	176	9	68	143	470	1202	4.4×10^{-3}	0.94	0.77
	10^{-4}	253	14	81	191	1108	3324	3.4×10^{-3}	2.29	1.28
	10^{-5}	306	10	99	190	1220	3750	4.2×10^{-3}	2.61	1.45
	10^{-6}	410	18	94	220	1428	5460	1.2×10^{-4}	3.91	2.05
	10^{-7}	558	18	86	230	1740	7145	4.2×10^{-5}	5.12	2.60
	10^{-8}	849	22	89	250	2415	10460	8.1×10^{-6}	7.48	3.71
	10^{-9}	1188	27	112	310	4796	23502	1.3×10^{-6}	15.97	6.42
	10^{-10}	1372	28	126	335	4910	24765	1.0×10^{-6}	17.28	7.03
	10^{-11}	1798	32	147	368	5250	27993	9.2×10^{-8}	20.16	8.55
	10^{-12}	2568	32	203	423	6712	36622	1.1×10^{-8}	26.89	11.29

Appendix A

Method coefficients when $s = p + 1$

$\tilde{v}$ = the first row of $\widetilde{V}$, since all the other elements are zeros and
$\hat{b}$ = weighting for the error estimate.

- $p = 2, \quad \lambda = \dfrac{6}{5}, \quad c = [\ 0 \quad \frac{1}{2} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} \frac{89}{250} & -\frac{314}{125} & \frac{739}{250} \\ -\frac{8}{25} & \frac{16}{25} & \frac{17}{25} \\ \frac{21}{5} & -\frac{52}{5} & \frac{31}{5} \end{bmatrix}, \quad \widetilde{U} = \begin{bmatrix} 1 & -\frac{6}{5} & 0 \\ 1 & -\frac{7}{10} & -\frac{19}{40} \\ 1 & -\frac{1}{5} & -\frac{7}{10} \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad \tfrac{1}{5} \quad -\tfrac{6}{5}\ ], \quad \hat{b} = [\ 4 \quad -8 \quad 4\ ].$$

- $p = 2, \quad \lambda = \dfrac{6}{5}, \quad c = [\ -1 \quad 0 \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} -\frac{939}{31250} & -\frac{311}{15625} & \frac{30311}{31250} \\ \frac{907}{1250} & -\frac{907}{625} & \frac{2157}{1250} \\ \frac{59}{50} & -\frac{84}{25} & \frac{109}{50} \end{bmatrix}, \quad \widetilde{U} = \begin{bmatrix} 1 & -\frac{11}{5} & \frac{17}{10} \\ 1 & -\frac{6}{5} & 0 \\ 1 & -\frac{1}{5} & -\frac{7}{10} \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad \tfrac{2}{25} \quad -\tfrac{1}{2}\ ], \quad \hat{b} = [\ 1 \quad -2 \quad 1\ ].$$

- $p = 3, \quad \lambda = 1.944, \quad c = [\ 0 \quad \frac{1}{3} \quad \frac{2}{3} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} \frac{77346972523}{976562500} & -\frac{120810497847}{488281250} & \frac{250790917569}{976562500} & -\frac{42711572199}{488281250} \\ \frac{144031464}{1953125} & -\frac{432094392}{1953125} & \frac{432094392}{1953125} & -\frac{142078339}{1953125} \\ \frac{1096081}{31250} & -\frac{1620684}{15625} & \frac{3100743}{31250} & -\frac{477728}{15625} \\ -\frac{13464}{125} & \frac{41517}{125} & -\frac{42642}{125} & \frac{14589}{125} \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -\frac{243}{125} & 0 & 0 \\ 1 & -\frac{604}{375} & -\frac{1333}{2250} & -\frac{1031}{10125} \\ 1 & -\frac{479}{375} & -\frac{1208}{1125} & -\frac{3874}{10125} \\ 1 & -\frac{118}{125} & -\frac{361}{250} & -\frac{302}{375} \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad -\tfrac{3}{25} \quad -\tfrac{19}{25} \quad \tfrac{29}{50}\ ], \quad \hat{b} = [\ -27 \quad 81 \quad -81 \quad 27\ ].$$

- $p = 3, \quad \lambda = 1.944, \quad c = [\ -1 \quad -\frac{1}{3} \quad \frac{1}{3} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} \frac{14404886177}{1953125000} & -\frac{45851377281}{1953125000} & \frac{49074033531}{1953125000} & -\frac{15283792427}{1953125000} \\ \frac{211012119}{15625000} & -\frac{633036357}{15625000} & \frac{633036357}{15625000} & -\frac{195387119}{15625000} \\ \frac{137419}{62500} & -\frac{182691}{31250} & \frac{224757}{62500} & \frac{1603}{31250} \\ -\frac{14319}{1000} & \frac{45207}{1000} & -\frac{47457}{1000} & \frac{16569}{1000} \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -\frac{368}{125} & \frac{611}{250} & -\frac{427}{375} \\ 1 & -\frac{854}{375} & \frac{1583}{2250} & -\frac{1156}{10125} \\ 1 & -\frac{604}{375} & -\frac{1333}{2250} & -\frac{1031}{10125} \\ 1 & -\frac{118}{125} & -\frac{361}{250} & -\frac{302}{375} \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad -\tfrac{1}{5} \quad -\tfrac{1}{2} \quad \tfrac{3}{10}\ ], \quad \hat{b} = [\ -\tfrac{27}{8} \quad \tfrac{81}{8} \quad -\tfrac{81}{8} \quad \tfrac{27}{8}\ ].$$

- $p = 3, \quad \lambda = 1.944, \quad c = [\ -2 \quad -1 \quad 0 \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} \frac{7612441963}{11718750000} & -\frac{25571700889}{11718750000} & \frac{33227950889}{11718750000} & -\frac{3315566963}{11718750000} \\ \frac{256625893}{46875000} & -\frac{256625893}{15625000} & \frac{256625893}{15625000} & -\frac{209750893}{46875000} \\ \frac{132443}{187500} & -\frac{101193}{62500} & \frac{7443}{62500} & \frac{148807}{187500} \\ -\frac{1189}{250} & \frac{3817}{250} & -\frac{4067}{250} & \frac{1439}{250} \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -\frac{493}{125} & \frac{736}{125} & -\frac{1958}{375} \\ 1 & -\frac{368}{125} & \frac{611}{250} & -\frac{427}{375} \\ 1 & -\frac{243}{125} & 0 & 0 \\ 1 & -\frac{118}{125} & -\frac{361}{250} & -\frac{302}{375} \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad -\tfrac{1}{50} \quad -\tfrac{1}{10} \quad \tfrac{1}{10}\ ], \quad \hat{b} = [\ -1 \quad 3 \quad -3 \quad 1\ ].$$

- $p = 4, \quad \lambda = 1.3012, \quad c = [\ 0 \quad \tfrac{1}{4} \quad \tfrac{1}{2} \quad \tfrac{3}{4} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} 22.7954048124 & -27.4882859163 & -49.9809044589 & 90.1650474170 & -34.2912618543 \\ 36.9863136352 & -147.9452545409 & 221.9178818114 & -147.9452545409 & 37.9863136352 \\ -319.0490246963 & 1274.8627654519 & -1908.2941481779 & 1264.1960987853 & -311.7156913630 \\ 421.3957802667 & -1701.5831210667 & 2592.3746816000 & -1765.5831210667 & 453.3957802667 \\ 430.3360000000 & -1785.3440000000 & 2774.0160000000 & -1913.3440000000 & 494.3360000000 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -1.3012000000 & 0 & 0 & 0 \\ 1 & -1.0512000000 & -0.2940500000 & -0.0380583333 & -0.0032257812 \\ 1 & -0.8012000000 & -0.5256000000 & -0.1418166667 & -0.0245041667 \\ 1 & -0.5512000000 & -0.6946500000 & -0.2956500000 & -0.0783070312 \\ 1 & -0.3012000000 & -0.8012000000 & -0.4839333333 & -0.1752000000 \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad -\tfrac{1}{5} \quad -\tfrac{97}{100} \quad -\tfrac{94}{100} \quad \tfrac{53}{100}\ ], \quad \hat{b} = [\ 256 \quad -1024 \quad 1536 \quad -1024 \quad 256\ ].$$

- $p = 4, \quad \lambda = 1.3012, \quad c = \begin{bmatrix} -1 & -\frac{1}{2} & 0 & \frac{1}{2} & 1 \end{bmatrix}^T,$

$$\widetilde{B} = \begin{bmatrix} -15.4604309760 & 63.7617239042 & -100.2659191896 & 70.8283905709 & -17.8937643094 \\ -25.8236987215 & 103.2947948862 & -154.9421923293 & 103.2947948862 & -24.8236987215 \\ -15.5434206515 & 61.5070159394 & -90.2605239091 & 56.1736826061 & -11.8767539849 \\ 30.0907562667 & -124.3630250667 & 196.5445376000 & -140.3630250667 & 38.0907562667 \\ 36.5760000000 & -154.3040000000 & 243.4560000000 & -170.3040000000 & 44.5760000000 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -2.3012000000 & 1.8012000000 & -0.8172666667 & 0.2585333333 \\ 1 & -1.8012000000 & 0.7756000000 & -0.1834833333 & 0.0297125000 \\ 1 & -1.3012000000 & 0 & 0 & 0 \\ 1 & -0.8012000000 & -0.5256000000 & -0.1418166667 & -0.0245041667 \\ 1 & -0.3012000000 & -0.8012000000 & -0.4839333333 & -0.1752000000 \end{bmatrix},$$

$$\tilde{v} = \begin{bmatrix} 1 & 0.03 & -0.6 & 0.02 & 0.3 \end{bmatrix}, \quad \hat{b} = \begin{bmatrix} 16 & -64 & 96 & -64 & 16 \end{bmatrix}.$$

- $p = 4, \quad \lambda = 1.3012, \quad c = \begin{bmatrix} -3 & -2 & -1 & 0 & 1 \end{bmatrix}^T,$

$$\widetilde{B} = \begin{bmatrix} -0.8988303845 & 3.7669882045 & -6.4413156401 & 5.2869882045 & -0.8038303845 \\ -2.6314081284 & 10.5256325137 & -15.7884487705 & 10.5256325137 & -1.6314081284 \\ -1.0328237267 & 3.7979615735 & -4.6969423603 & 1.1312949069 & 0.8005096066 \\ 2.7706122667 & -12.0824490667 & 20.6236736000 & -16.0824490667 & 4.7706122667 \\ 3.0960000000 & -13.3840000000 & 21.5760000000 & -15.3840000000 & 4.0960000000 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -4.3012000000 & 8.4036000000 & -10.3554000000 & 9.2304000000 \\ 1 & -3.3012000000 & 4.6024000000 & -3.9357333333 & 2.4016000000 \\ 1 & -2.3012000000 & 1.8012000000 & -0.8172666667 & 0.2585333333 \\ 1 & -1.3012000000 & 0.0000000000 & 0.0000000000 & 0.0000000000 \\ 1 & -0.3012000000 & -0.8012000000 & -0.4839333333 & -0.1752000000 \end{bmatrix},$$

$$\tilde{v} = \begin{bmatrix} 1 & 0.09 & -0.3 & 0.3 & 0.08 \end{bmatrix}, \quad \hat{b} = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \end{bmatrix}.$$

- $p = 5, \quad \lambda = 1.80568, \quad c = [\ -1 \quad -\frac{3}{5} \quad -\frac{1}{5} \quad \frac{1}{5} \quad \frac{3}{5} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} 4.52777700 & -59.41449263 & 215.76040887 & -342.71787415 & 250.42360721 & -67.27942630 \\ -102.02936466 & 510.14682329 & -1020.29364657 & 1020.29364657 & -510.14682329 & 103.02936466 \\ -100.16151919 & 501.43259597 & -1004.94852527 & 1009.11519193 & -510.80759597 & 105.36985253 \\ 449.12669448 & -2239.90430576 & 4462.10027818 & -4431.89194485 & 2191.46680576 & -430.89752782 \\ 338.68944687 & -1670.00973438 & 3277.51946875 & -3199.39446875 & 1552.82223438 & -299.62694687 \\ -520.90625000 & 2643.59375000 & -5365.31250000 & 5443.43750000 & -2760.78125000 & 559.96875000 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -2.80568000 & 2.30568000 & -1.06950667 & 0.34261333 & -0.08357000 \\ 1 & -2.40568000 & 1.26340800 & -0.36102240 & 0.07040448 & -0.01039867 \\ 1 & -2.00568000 & 0.38113600 & -0.03744693 & 0.00247424 & -0.00012305 \\ 1 & -1.60568000 & -0.34113600 & -0.03478027 & -0.00234091 & -0.00011771 \\ 1 & -1.20568000 & -0.90340800 & -0.28902240 & -0.05960448 & -0.00910267 \\ 1 & -0.80568000 & -1.30568000 & -0.73617333 & -0.25928000 & -0.06690333 \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad -0.3 \quad -1.9 \quad -0.3 \quad 1.6 \quad 1.6\ ],$$

$$\hat{b} = [\ -97.65625 \quad 488.28125 \quad -976.5625 \quad 976.5625 \quad -488.28125 \quad 97.65625\ ].$$

- $p = 5, \quad \lambda = 1.80568, \quad c = [\ -4 \quad -3 \quad -2 \quad -1 \quad 0 \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} -5.04487343 & 24.95131161 & -49.25484543 & 47.87206766 & -22.26047827 & 4.83681788 \\ -14.95109726 & 74.75548629 & -149.51097257 & 149.51097257 & -74.75548629 & 15.95109726 \\ -2.95084183 & 15.00420915 & -30.84175163 & 32.50841830 & -18.75420915 & 5.03417516 \\ 9.49263096 & -46.54648815 & 90.25964298 & -85.42630964 & 38.79648815 & -6.57596430 \\ 1.28499594 & -4.92497968 & 5.84995936 & -0.84995936 & -2.57502032 & 1.21500406 \\ -6.73408000 & 34.67040000 & -71.34080000 & 73.34080000 & -37.67040000 & 7.73408000 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -5.80568000 & 15.22272000 & -25.11210667 & 29.92725333 & -27.79392000 \\ 1 & -4.80568000 & 9.91704000 & -12.62556000 & 11.50056000 & -8.11917000 \\ 1 & -3.80568000 & 5.61136000 & -4.94469333 & 3.07424000 & -1.47045333 \\ 1 & -2.80568000 & 2.30568000 & -1.06950667 & 0.34261333 & -0.08357000 \\ 1 & -1.80568000 & 0 & 0 & 0 & 0 \\ 1 & -0.80568000 & -1.30568000 & -0.73617333 & -0.25928000 & -0.06690333 \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad -0.1 \quad -0.3 \quad 0.4 \quad 0.01 \quad 0.25\ ], \quad \hat{b} = [\ -1 \quad 5 \quad -10 \quad 10 \quad -5 \quad 1\ ].$$

- $p = 5, \quad \lambda = 1.80568, \quad c = [\ 0 \quad \frac{1}{5} \quad \frac{2}{5} \quad \frac{3}{5} \quad \frac{4}{5} \quad 1\ ]^T,$

$$\widetilde{B} = \begin{bmatrix} 8888.93173782 & -44815.13507806 & 90783.87293384 & -92416.38404507 & 47286.93091140 & -9726.04646004 \\ 5394.48356410 & -26972.41782048 & 53944.83564097 & -53944.83564095 & 26972.41782049 & -5393.48356411 \\ 548.74439419 & -2742.47197098 & 5480.77727529 & -5472.44394195 & 2723.72197098 & -538.32772753 \\ 4552.93853285 & -22741.77599757 & 45412.71866182 & -45291.88532848 & 22548.02599757 & -4480.02186618 \\ 7941.96980000 & -39522.34900000 & 78544.69800000 & -77919.69800000 & 38584.84900000 & -7629.46980000 \\ -12700.25000000 & 64126.25000000 & -129502.50000000 & 130752.50000000 & -66001.25000000 & 13325.25000000 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -1.80568 & 0 & 0 & 0 & 0 \\ 1 & -1.60568 & -0.341136 & -0.0347802667 & -0.0023409067 & -0.0001177120 \\ 1 & -1.40568 & -0.642272 & -0.1337877333 & -0.0181939200 & -0.0018407253 \\ 1 & -1.20568 & -0.903408 & -0.2890224000 & -0.0596044800 & -0.0091026720 \\ 1 & -1.00568 & -1.124544 & -0.4924842667 & -0.1370180267 & -0.0280862720 \\ 1 & -0.80568 & -1.30568 & -0.7361733333 & -0.25928 & -0.0669033333 \end{bmatrix},$$

$\tilde{v} = [\ 1 \quad -1.17 \quad -3.69 \quad -0.086 \quad 4.28 \quad 3.43\ ], \quad \hat{b} = [\ -3125 \quad 15625 \quad -31250 \quad 31250 \quad -15625 \quad 3125\ ].$

- $p = 6, \quad \lambda = 1.352193, \quad c = \begin{bmatrix} 0 & \frac{1}{6} & \frac{1}{3} & \frac{1}{2} & \frac{2}{3} & \frac{5}{6} & 1 \end{bmatrix}^T$

$$\widetilde{B} = \begin{bmatrix} 97583.76083606 & -589443.31901637 & 1482857.19254093 & -1988693.35672124 & 1499594.85254093 & -602828.69501637 & 100930.16483606 \\ 55419.25306466 & -332515.51838794 & 831288.79596984 & -1108385.06129312 & 831288.79596984 & -332515.51838794 & 55420.25306466 \\ 193935.74925578 & -1163615.69553467 & 2909043.73883668 & -3878734.98511558 & 2909066.23883668 & -1163644.49553467 & 193949.44925578 \\ -144642.32612947 & 867823.95677684 & -2169451.89194210 & 2892378.52258946 & -2168992.89194209 & 867391.95677684 & -144507.32612947 \\ -212621.61023631 & 1275351.66141784 & -3187110.15354460 & 4247140.20472614 & -3182952.15354460 & 1271895.66141784 & -211703.61023631 \\ 28967.92702125 & -176399.56212751 & 448774.90531878 & -610462.54042504 & 468214.90531878 & -191951.56212751 & 32855.92702125 \\ 153125.81625600 & -926530.89753600 & 2335767.24384000 & -3140276.32512000 & 2374647.24384000 & -957634.89753600 & 160901.81625600 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -1.35219300 & 0.00000000 & 0.00000000 & 0.00000000 & 0.00000000 & 0.00000000 \\ 1 & -1.18552633 & -0.21147661 & -0.01800885 & -0.00101121 & -0.00004240 & -0.00000142 \\ 1 & -1.01885967 & -0.39517544 & -0.06894899 & -0.00783247 & -0.00066128 & -0.00004447 \\ 1 & -0.85219300 & -0.55109650 & -0.14819079 & -0.02556652 & -0.00326092 & -0.00033043 \\ 1 & -0.68552633 & -0.67923978 & -0.25110462 & -0.05854451 & -0.01003177 & -0.00136196 \\ 1 & -0.51885967 & -0.77960528 & -0.37306084 & -0.11032597 & -0.02382182 & -0.00406333 \\ 1 & -0.35219300 & -0.85219300 & -0.50942983 & -0.18369883 & -0.04800804 & -0.00987939 \end{bmatrix},$$

$$\tilde{v} = \begin{bmatrix} 1.0 & 0.4 & -0.82 & -0.27 & -0.03 & 0.46 & -0.2 \end{bmatrix},$$

$$\hat{b} = \begin{bmatrix} 46656 & -279936 & 699840 & -933120 & 699840 & -279936 & 46656 \end{bmatrix}.$$

- $p = 6, \quad \lambda = 1.352193, \quad c = \left[\begin{array}{ccccccc} -1 & -\frac{2}{3} & -\frac{1}{3} & 0 & \frac{1}{3} & \frac{2}{3} & 1 \end{array} \right],$

$$\tilde{B} = \left[\begin{array}{rrrrrrr} 1921.11485220 & -11581.24536321 & 29068.63403302 & -38880.28454403 & 29219.89028302 & -11694.82536321 & 1947.61610220 \\ 2422.72670236 & -14536.36021416 & 36340.90053540 & -48454.53404720 & 36340.90053540 & -14536.36021416 & 2423.72670236 \\ 1944.02361618 & -11664.74169708 & 29164.10424269 & -38890.47232359 & 29175.35424269 & -11679.14169708 & 1950.87361618 \\ -3111.05778356 & 18658.84670134 & -46620.11675335 & 62104.15567114 & -46505.36675335 & 18550.84670134 & -3077.30778356 \\ -3432.57225888 & 20548.18355328 & -51211.83388320 & 67989.94517760 & -50692.08388320 & 20116.18355328 & -3317.82225888 \\ 1360.75112341 & -8326.50674044 & 21302.26685111 & -29159.02246814 & 22517.26685111 & -9298.50674044 & 1603.75112341 \\ 2477.64087900 & -15108.84527400 & 38379.61318500 & -51982.81758000 & 39594.61318500 & -16080.84527400 & 2720.64087900 \end{array} \right],$$

$$\tilde{U} = \left[\begin{array}{rrrrrrr} 1 & -2.35219300 & 1.85219300 & -0.84276317 & 0.26703217 & -0.06467471 & 0.01265716 \\ 1 & -2.01885967 & 1.12368422 & -0.34987005 & 0.07500542 & -0.01222655 & 0.00160582 \\ 1 & -1.68552633 & 0.50628656 & -0.08129467 & 0.00886127 & -0.00072987 & 0.00004828 \\ 1 & -1.35219300 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1.01885967 & -0.39517544 & -0.06894899 & -0.00783247 & -0.00066128 & -0.00004447 \\ 1 & -0.68552633 & -0.67923978 & -0.25110462 & -0.05854451 & -0.01003177 & -0.00136196 \\ 1 & -0.35219300 & -0.85219300 & -0.50942983 & -0.18369883 & -0.04800804 & -0.00987939 \end{array} \right],$$

$$\tilde{v} = \left[\begin{array}{ccccccc} 1 & 0.1 & -0.7 & 0.01 & 0.3 & 0.4 & -0.1 \end{array} \right],$$

$$\hat{b} = \left[\begin{array}{ccccccc} 729 & -4374 & 10935 & -14580 & 10935 & -4374 & 729 \end{array} \right].$$

- $p = 6, \quad \lambda = 1.352193, \quad c = \begin{bmatrix} -5 & -4 & -3 & -2 & -1 & 0 & 1 \end{bmatrix},$

$$\widetilde{B} = \begin{bmatrix} -0.94797652 & 6.77327577 & -20.30645332 & 32.77758591 & -30.44048110 & 16.59549800 & -2.45144874 \\ 2.50804533 & -15.04827197 & 37.62067993 & -50.16090657 & 37.62067993 & -15.04827197 & 3.50804533 \\ -1.51416649 & 8.88499892 & -21.46249730 & 26.94999640 & -17.71249730 & 4.08499892 & 0.76916685 \\ -5.68622382 & 33.28400956 & -80.21002391 & 100.72447632 & -67.46002391 & 21.28400956 & -1.93622382 \\ -0.24726918 & -0.26638492 & 6.54096230 & -19.55461641 & 25.79096230 & -16.26638492 & 4.00273082 \\ 6.30700424 & -39.84202543 & 105.60506359 & -150.14008478 & 120.60506359 & -51.84202543 & 9.30700424 \\ 3.96535100 & -24.79210600 & 64.48026500 & -89.30702000 & 69.48026500 & -28.79210600 & 4.96535100 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -6.35219300 & 19.26096500 & -37.73574583 & 54.21235417 & -61.25502604 & 56.91474826 \\ 1 & -5.35219300 & 13.40877200 & -21.48421067 & 25.09005867 & -22.95672533 & 17.22760249 \\ 1 & -4.35219300 & 8.55657900 & -10.58486850 & 9.45986850 & -6.58865137 & 3.75069082 \\ 1 & -3.35219300 & 4.70438600 & -4.03771933 & 2.46959067 & -1.16812867 & 0.44947369 \\ 1 & -2.35219300 & 1.85219300 & -0.84276317 & 0.26703217 & -0.06467471 & 0.01265716 \\ 1 & -1.35219300 & 0 & 0 & 0 & 0 & 0 \\ 1 & -0.35219300 & -0.85219300 & -0.50942983 & -0.18369883 & -0.04800804 & -0.00987939 \end{bmatrix},$$

$$\tilde{v} = \begin{bmatrix} 1 & -1 & -0.5 & 0.1 & 0.2 & 0.5 & 0.5 \end{bmatrix}, \quad \hat{b} = \begin{bmatrix} 1 & -6 & 15 & -20 & 15 & -6 & 1 \end{bmatrix}.$$

- $p = 7, \quad \lambda = 1.7368, \quad c = \left[\begin{array}{cccccccc} 0 & \frac{1}{7} & \frac{2}{7} & \frac{3}{7} & \frac{4}{7} & \frac{5}{7} & \frac{6}{7} & 1 \end{array}\right]^T$

$$\widetilde{B} = \left[\begin{array}{rrrrrrrr}
10973701.30072700 & -76756469.26956426 & 230102231.83833459 & -383243964.28843307 & 383003285.84066832 & -229669136.79473719 & 76516011.43223470 & -10925659.46171152 \\
8244588.27810253 & -57712117.94673688 & 173136353.84015310 & -288560589.73368442 & 288560589.73358852 & -173136353.84021059 & 57712117.94671771 & -8244587.27807922 \\
16559807.54158936 & -115918651.62454960 & 347755949.97353667 & -579593237.70598793 & 579593217.28935242 & -347755905.87359267 & 115918610.79117890 & -16559790.39158935 \\
-19820318.79498392 & 138742268.85930389 & -416226959.29468119 & 693711966.32437515 & -693712540.71322238 & 416228127.94462508 & -138743084.16489840 & 19820539.83942325 \\
-9452593.62934343 & 66168798.53040781 & -198508925.21620700 & 330853939.65200651 & -330862043.02701181 & 198524231.59122339 & -66178102.40540235 & 9454694.50434446 \\
3520414.50470776 & -24636098.69962065 & 73883085.59886084 & -123086054.16476510 & 123020826.99810439 & -73764236.09886198 & 24568470.53295268 & -3506408.67137435 \\
8544756.62671829 & -59771278.88702812 & 179170977.16108391 & -298352184.43514049 & 298058061.93514049 & -178641556.66108370 & 59477156.38702811 & -8485932.12671829 \\
-5536656.05920000 & 38874241.41440000 & -116975671.24320000 & 195547697.07200000 & -196135942.07200000 & 118034512.24320000 & -39462486.41440000 & 5654305.05920000
\end{array}\right],$$

$$\widetilde{U} = \left[\begin{array}{rrrrrrrr}
1 & -1.73680000 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1.59394286 & -0.23791020 & -0.01723654 & -0.00082657 & -0.00002964 & -0.00000085 & -0.00000002 \\
1 & -1.45108571 & -0.45541224 & -0.06700253 & -0.00647375 & -0.00046638 & -0.00002680 & -0.00000128 \\
1 & -1.30822857 & -0.65250612 & -0.14638251 & -0.02138034 & -0.00232087 & -0.00020065 & -0.00001442 \\
1 & -1.16537143 & -0.82919184 & -0.25246103 & -0.04956868 & -0.00720817 & -0.00083346 & -0.00008004 \\
1 & -1.02251429 & -0.98546939 & -0.38232264 & -0.09464459 & -0.01728818 & -0.00250663 & -0.00030155 \\
1 & -0.87965714 & -1.12133878 & -0.53305190 & -0.15979742 & -0.03520619 & -0.00614550 & -0.00088917 \\
1 & -0.73680000 & -1.23680000 & -0.70173333 & -0.24780000 & -0.06403333 & -0.01308444 & -0.00221381
\end{array}\right],$$

$$\tilde{v} = \left[\begin{array}{cccccccc} 1 & 0.4 & -1.4 & -0.15 & 0.13 & 0.16 & 0.75 & -0.05 \end{array}\right],$$

$$\hat{b} = \left[\begin{array}{rrrrrrrr} -823543 & 5764801 & -17294403 & 28824005 & -28824005 & 17294403 & -5764801 & 823543 \end{array}\right].$$

- $p = 7, \quad \lambda = 1.7368, \quad c = \begin{bmatrix} -1 & -\frac{5}{7} & -\frac{3}{7} & -\frac{1}{7} & \frac{1}{7} & \frac{3}{7} & \frac{5}{7} & 1 \end{bmatrix}^T,$

$$\widetilde{B} = \begin{bmatrix} 83732.45718710 & -585804.65646480 & 1756330.12971732 & -2925244.89345827 & 2923148.07003235 & -1752599.78881107 & 583782.51125068 & -83342.84945331 \\ 95947.93737149 & -671635.56160044 & 2014906.68480132 & -3358177.80800220 & 3358177.80800220 & -2014906.68480132 & 671635.56160044 & -95946.93737149 \\ 51596.97530687 & -361178.24381475 & 1083532.28144425 & -1805881.01074042 & 1805870.80240709 & -1083510.23144425 & 361157.82714808 & -51588.40030687 \\ -141328.58841360 & 989309.44250629 & -2967966.50668554 & 4946702.71947590 & -4946846.31669812 & 2968258.66918554 & -989513.26889518 & 141383.84952471 \\ -93503.70902614 & 654606.35380795 & -1964135.26454884 & 3274275.14403973 & -3275288.06591473 & 1966048.56142384 & -655769.33818295 & 93766.31840114 \\ 62828.62363609 & -439375.18836932 & 1316549.90885795 & -2190973.48351325 & 2186896.78559658 & -1309121.81510795 & 435148.42795265 & -61953.25905276 \\ 48389.78247986 & -337415.43048403 & 1007781.93207710 & -1671320.58992016 & 1662129.26179516 & -991237.54145210 & 328224.10235903 & -46551.51685486 \\ -43567.63061875 & 306811.67995625 & -925949.83674375 & 1552441.05603125 & -1561632.38415625 & 942494.22736875 & -316003.00808125 & 45405.89624375 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -2.73680000 & 2.23680000 & -1.03506667 & 0.33113333 & -0.08070000 & 0.01586222 & -0.00261063 \\ 1 & -2.45108571 & 1.49567347 & -0.50379981 & 0.11633694 & -0.02038709 & 0.00287555 & -0.00033919 \\ 1 & -2.16537143 & 0.83617959 & -0.17262157 & 0.02419167 & -0.00256184 & 0.00021787 & -0.00001547 \\ 1 & -1.87965714 & 0.25831837 & -0.01820836 & 0.00086128 & -0.00003064 & 0.00000087 & -0.00000002 \\ 1 & -1.59394286 & -0.23791020 & -0.01723654 & -0.00082657 & -0.00002964 & -0.00000085 & -0.00000002 \\ 1 & -1.30822857 & -0.65250612 & -0.14638251 & -0.02138034 & -0.00232087 & -0.00020065 & -0.00001442 \\ 1 & -1.02251429 & -0.98546939 & -0.38232264 & -0.09464459 & -0.01728818 & -0.00250663 & -0.00030155 \\ 1 & -0.73680000 & -1.23680000 & -0.70173333 & -0.24780000 & -0.06403333 & -0.01308444 & -0.00221381 \end{bmatrix},$$

$$\tilde{v} = \begin{bmatrix} 1 & 0.02 & -1.2 & 0.03 & 0.6 & 0.5 & -0.05 & -0.2 \end{bmatrix},$$

$$\hat{b} = \begin{bmatrix} -6433.92968750 & 45037.50781250 & -135112.52343750 & 225187.53906250 & -225187.53906249 & 135112.52343749 & -45037.507812499 & 6433.9296874998 \end{bmatrix}.$$

- $p = 7, \quad \lambda = 1.7368, \quad c = [\ -6 \quad -5 \quad -4 \quad -3 \quad -2 \quad -1 \quad 0 \quad 1\ ]^T$

$$\widetilde{B} = \begin{bmatrix} 10.98583000 & -77.11077362 & 231.90512841 & -387.24447658 & 387.79637804 & -233.39771770 & 78.62239730 & -10.65676585 \\ 40.62765210 & -284.39356471 & 853.18069413 & -1421.96782355 & 1421.96782355 & -853.18069413 & 284.39356471 & -39.62765210 \\ 18.28738923 & -127.84505792 & 382.83517376 & -636.30862293 & 633.39195626 & -376.53517376 & 122.01172459 & -15.83738923 \\ -30.17143724 & 211.96117180 & -639.00018208 & 1072.50030347 & -1084.22252569 & 662.85018208 & -228.60006069 & 34.68254835 \\ -23.08882044 & 163.49674307 & -497.86522921 & 846.48371535 & -870.10871535 & 542.49022921 & -190.62174307 & 29.21382044 \\ 10.37934621 & -69.82209011 & 198.96627033 & -309.77711722 & 282.61045055 & -149.46627033 & 41.65542344 & -4.54601287 \\ 3.37565205 & -21.12956437 & 54.88869312 & -75.64782187 & 58.14782187 & -23.38869312 & 3.62956437 & 0.12434795 \\ -8.99440000 & 63.96080000 & -194.88240000 & 329.80400000 & -334.80400000 & 203.88240000 & -68.96080000 & 9.99440000 \end{bmatrix},$$

$$\widetilde{U} = \begin{bmatrix} 1 & -7.73680000 & 28.42080000 & -67.26240000 & 116.52480000 & -158.58720000 & 177.34464000 & -168.08749714 \\ 1 & -6.73680000 & 21.18400000 & -42.54333333 & 62.22500000 & -71.27083333 & 66.93055556 & -53.19196429 \\ 1 & -5.73680000 & 14.94720000 & -24.56106667 & 29.19253333 & -27.05920000 & 20.50958222 & -13.13125587 \\ 1 & -4.73680000 & 9.71040000 & -12.31560000 & 11.19060000 & -7.88670000 & 4.52952000 & -2.19243857 \\ 1 & -3.73680000 & 5.47360000 & -4.80693333 & 2.98240000 & -1.42453333 & 0.55203556 & -0.17977905 \\ 1 & -2.73680000 & 2.23680000 & -1.03506667 & 0.33113333 & -0.08070000 & 0.01586222 & -0.00261063 \\ 1 & -1.73680000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -0.73680000 & -1.23680000 & -0.70173333 & -0.24780000 & -0.06403333 & -0.01308444 & -0.00221381 \end{bmatrix},$$

$$\tilde{v} = [\ 1 \quad 0.1 \quad -0.4 \quad 0.1 \quad 0.05 \quad -0.2 \quad 0.1 \quad 0.2\ ],$$

$$\hat{b} = [\ -1 \quad 7 \quad -21 \quad 35 \quad -35 \quad 21 \quad -7 \quad 1\ ].$$

Appendix B

RADAU5, VODE and PSIDE Results

Table B.1: Results for Kaps problem using RADAU5 and VODE.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
RADAU5	10^{-1}	27	0	16	27	135	2.7×10^{-5}
	10^{-2}	17	0	15	17	107	5.2×10^{-6}
	10^{-3}	20	0	17	20	131	5.7×10^{-6}
	10^{-4}	24	0	21	24	170	1.4×10^{-6}
	10^{-5}	28	0	24	27	210	2.3×10^{-7}
	10^{-6}	35	0	29	33	279	7.2×10^{-8}
	10^{-7}	45	0	34	40	369	1.2×10^{-8}
	10^{-8}	59	0	35	44	506	2.5×10^{-9}
	10^{-9}	77	0	43	48	674	4.5×10^{-10}
	10^{-10}	110	0	46	53	973	8.9×10^{-11}
	10^{-11}	155	0	47	55	1402	1.5×10^{-11}
	10^{-12}	224	0	51	61	2034	2.6×10^{-12}
VODE	10^{-1}	16	0	1	5	23	4.9×10^{-3}
	10^{-2}	22	0	1	6	31	8.1×10^{-4}
	10^{-3}	42	1	1	9	59	3.4×10^{-5}
	10^{-4}	64	2	2	12	83	7.1×10^{-6}
	10^{-5}	79	4	2	17	110	3.3×10^{-6}
	10^{-6}	95	1	2	14	117	5.2×10^{-7}
	10^{-7}	123	7	3	26	157	3.9×10^{-8}
	10^{-8}	163	3	3	24	188	1.2×10^{-8}
	10^{-9}	236	4	5	30	264	1.2×10^{-9}
	10^{-10}	345	4	6	36	372	1.7×10^{-10}
	10^{-11}	453	5	8	42	485	2.4×10^{-11}
	10^{-12}	675	4	11	52	711	5.1×10^{-12}

Table B.2: Results for Oregonator problem using RADAU5 and VODE.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
RADAU5	10^{-2}	98	1	67	98	903	1.9×10^{-1}
	10^{-3}	116	4	86	116	1052	2.1×10^{-2}
	10^{-4}	151	6	118	149	1368	6.1×10^{-2}
	10^{-5}	196	4	162	192	1726	1.4×10^{-2}
	10^{-6}	270	5	229	262	2369	2.9×10^{-3}
	10^{-7}	378	4	304	356	3266	5.0×10^{-5}
	10^{-8}	537	3	381	491	4537	1.1×10^{-6}
	10^{-9}	773	3	456	676	6375	1.4×10^{-6}
	10^{-10}	1124	2	551	935	9035	2.0×10^{-7}
	10^{-11}	1644	2	660	1321	12992	2.5×10^{-8}
	10^{-12}	2411	2	764	1857	18732	2.7×10^{-9}
VODE	10^{-3}	464	39	18	92	746	1.7×10^{1}
	10^{-4}	532	45	18	108	822	5.6×10^{0}
	10^{-5}	693	50	20	120	1032	4.9×10^{-1}
	10^{-6}	993	68	23	152	1380	3.0×10^{-2}
	10^{-7}	1331	78	28	171	1781	3.1×10^{-3}
	10^{-8}	1794	100	33	221	2332	8.1×10^{-4}
	10^{-9}	2382	121	41	273	3013	1.2×10^{-4}
	10^{-10}	3230	122	56	317	3960	7.5×10^{-6}
	10^{-11}	4503	149	77	402	5250	8.7×10^{-6}
	10^{-12}	6350	168	105	529	7224	1.4×10^{-6}

Table B.3: Results for Robertson problem using RADAU5 and VODE.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
RADAU5	10^{-1}	52	1	39	52	362	4.2×10^{-5}
	10^{-2}	58	1	48	57	405	1.0×10^{-6}
	10^{-3}	70	0	61	70	469	7.3×10^{-8}
	10^{-4}	89	1	78	89	646	3.4×10^{-9}
	10^{-5}	117	1	104	117	884	2.1×10^{-10}
	10^{-6}	158	1	143	157	1225	1.5×10^{-11}
	10^{-7}	222	1	194	212	1745	3.3×10^{-13}
	10^{-8}	308	1	235	256	2446	8.6×10^{-14}
	10^{-9}	443	2	283	317	3528	1.6×10^{-14}
	10^{-10}	640	2	359	406	5105	8.8×10^{-15}
	10^{-11}	934	1	473	532	7452	4.6×10^{-15}
	10^{-12}	1364	2	634	730	10893	1.8×10^{-15}
VODE	10^{-1}	121	5	6	45	163	3.2×10^{-4}
	10^{-2}	222	9	9	56	315	1.8×10^{-4}
	10^{-3}	272	19	9	71	458	6.1×10^{-6}
	10^{-4}	350	13	9	64	604	2.5×10^{-6}
	10^{-5}	496	21	9	81	740	4.6×10^{-8}
	10^{-6}	700	36	12	106	968	1.3×10^{-8}
	10^{-7}	889	40	15	123	1164	2.2×10^{-9}
	10^{-8}	1309	87	22	202	1693	3.1×10^{-10}
	10^{-9}	1725	130	29	270	2166	4.3×10^{-11}
	10^{-10}	2272	153	39	321	2775	6.6×10^{-12}
	10^{-11}	2997	176	51	391	3567	9.6×10^{-13}
	10^{-12}	3803	122	65	365	4265	1.4×10^{-13}

Table B.4: Results for van der Pol problem using RADAU5 and VODE.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER
RADAU5	10^{-2}	178	6	95	161	1348	3.2×10^{-4}
	10^{-3}	221	7	122	205	1663	1.3×10^{-4}
	10^{-4}	281	7	163	252	2242	8.3×10^{-6}
	10^{-5}	363	8	221	310	2954	2.1×10^{-6}
	10^{-6}	499	8	306	411	3985	2.9×10^{-7}
	10^{-7}	722	7	435	587	5751	1.8×10^{-8}
	10^{-8}	1055	6	483	842	8281	2.3×10^{-9}
	10^{-9}	1548	4	530	1194	12019	3.7×10^{-10}
	10^{-10}	2274	2	619	1713	17526	5.5×10^{-11}
	10^{-11}	3341	2	753	2507	25478	1.0×10^{-11}
	10^{-12}	4923	2	960	3672	37069	1.3×10^{-12}
VODE	10^{-1}	128	17	28	64	255	3.0×10^{-1}
	10^{-2}	374	46	22	114	621	3.0×10^{-2}
	10^{-3}	588	66	19	146	908	1.2×10^{-2}
	10^{-4}	810	83	21	177	1207	2.0×10^{-3}
	10^{-5}	1154	99	24	210	1611	3.1×10^{-4}
	10^{-6}	1538	133	30	273	2130	2.4×10^{-5}
	10^{-7}	2187	160	38	319	2867	5.9×10^{-6}
	10^{-8}	3341	241	55	475	4267	4.9×10^{-7}
	10^{-9}	4647	287	76	597	5734	8.0×10^{-8}
	10^{-10}	6458	347	106	755	7777	6.6×10^{-9}
	10^{-11}	8702	431	144	984	10339	1.0×10^{-9}
	10^{-12}	11842	462	197	1159	13734	1.2×10^{-10}

Table B.5: Results for Ring Modulator using RADAU5, VODE and PSIDE.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER	CPU
RADAU5	10^{-2}	1212	67	669	1138	9493	3.9×10^{-2}	0.33
	10^{-3}	1624	247	795	1389	11074	3.8×10^{-3}	0.32
	10^{-4}	2183	350	989	1740	14496	1.2×10^{-3}	0.39
	10^{-5}	2915	423	1088	2088	19528	6.9×10^{-5}	0.51
	10^{-6}	3974	453	1293	2679	26333	1.2×10^{-5}	0.67
	10^{-7}	5568	493	1577	3588	37067	1.9×10^{-6}	0.93
	10^{-8}	7848	482	1943	4709	52541	3.2×10^{-7}	1.28
	10^{-9}	11152	456	2392	6536	75577	5.3×10^{-8}	1.82
	10^{-10}	16111	445	2945	9224	111116	8.4×10^{-9}	2.64
	10^{-11}	23398	430	3421	12999	163641	1.4×10^{-9}	3.83
	10^{-12}	34112	409	3543	18427	240223	2.3×10^{-10}	5.55
VODE	10^{-3}	6122	372	438	1227	9597	1.5×10^{-1}	0.32
	10^{-4}	7448	474	260	1163	11001	2.3×10^{-2}	0.36
	10^{-5}	10812	729	264	1609	15971	1.2×10^{-3}	0.52
	10^{-6}	15052	909	289	1881	21297	6.3×10^{-5}	0.69
	10^{-7}	19535	1157	338	2311	26524	6.2×10^{-6}	0.87
	10^{-8}	24436	1241	409	2560	31210	1.2×10^{-6}	1.06
	10^{-9}	29611	1150	493	2643	35734	1.1×10^{-6}	1.23
	10^{-10}	36922	951	608	2792	42158	2.3×10^{-7}	1.47
	10^{-11}	49769	953	828	3522	55214	4.0×10^{-8}	1.94
	10^{-12}	71298	930	1175	4542	77978	7.4×10^{-9}	2.74
PSIDE	10^{-2}	1994	701	882	7748	27993	2.7×10^{-3}	2.20
	10^{-3}	1450	509	621	5484	22785	1.4×10^{-3}	1.59
	10^{-4}	1360	349	682	5212	27865	1.8×10^{-3}	1.69
	10^{-5}	1712	475	802	6072	39178	3.2×10^{-5}	2.12
	10^{-6}	2258	539	708	6460	46412	1.6×10^{-6}	2.41
	10^{-7}	3213	629	629	7400	59669	2.0×10^{-7}	3.01
	10^{-8}	4677	710	637	8676	76875	1.8×10^{-8}	3.82
	10^{-9}	6973	770	581	10184	104870	1.8×10^{-9}	5.10
	10^{-10}	10304	621	486	10120	146451	1.5×10^{-10}	6.74
	10^{-11}	15816	573	414	10460	215618	1.3×10^{-11}	9.51
	10^{-12}	24612	556	365	10876	325994	1.6×10^{-12}	13.91

Table B.6: Results for the Medical Akzo Nobel problem using RADAU5, VODE and PSIDE.

METHOD	TOL	TOT ST	REJ ST	J-EVAL	LU	F-EVAL	END GL ER	CPU
RADAU5	10^{-1}	59	0	35	59	524	1.7×10^{-2}	0.23
	10^{-2}	89	10	41	87	694	7.9×10^{-3}	0.32
	10^{-3}	96	17	45	92	763	4.3×10^{-4}	0.35
	10^{-4}	125	20	65	119	933	7.6×10^{-5}	0.44
	10^{-5}	160	19	92	153	1156	2.0×10^{-5}	0.55
	10^{-6}	215	19	131	196	1561	3.3×10^{-6}	0.74
	10^{-7}	284	17	181	242	2056	3.1×10^{-7}	0.96
	10^{-8}	399	23	251	327	2560	3.4×10^{-8}	1.23
	10^{-9}	543	18	339	418	3524	3.1×10^{-9}	1.66
	10^{-10}	787	20	464	577	5138	1.0×10^{-9}	2.38
	10^{-11}	1152	28	462	578	7552	1.3×10^{-9}	3.20
	10^{-12}	1658	27	361	503	10990	1.4×10^{-9}	4.28
VODE	10^{-4}	433	19	10	77	615	1.3×10^{-3}	0.34
	10^{-5}	653	26	12	97	849	2.9×10^{-4}	0.48
	10^{-6}	829	26	14	102	1040	8.0×10^{-5}	0.59
	10^{-7}	1057	32	18	128	1262	1.7×10^{-5}	0.73
	10^{-8}	1416	43	24	164	1622	6.0×10^{-7}	0.96
	10^{-9}	1941	42	33	195	2101	3.7×10^{-8}	1.25
	10^{-10}	2647	34	44	227	2766	3.8×10^{-9}	1.66
	10^{-11}	3837	43	63	302	3975	2.4×10^{-9}	2.37
	10^{-12}	5510	40	91	384	5643	1.6×10^{-9}	3.36
PSIDE	10^{-2}	83	19	34	328	912	3.0×10^{-3}	0.62
	10^{-3}	90	22	30	356	944	1.2×10^{-4}	0.66
	10^{-4}	118	35	34	456	1263	1.0×10^{-5}	0.87
	10^{-5}	142	46	42	548	1668	5.6×10^{-6}	1.11
	10^{-6}	158	45	64	628	2258	1.1×10^{-6}	1.43
	10^{-7}	159	14	109	624	2834	7.6×10^{-8}	1.70
	10^{-8}	238	14	47	596	4253	1.5×10^{-8}	2.37
	10^{-9}	361	14	23	592	6156	2.2×10^{-9}	3.29
	10^{-10}	551	14	21	620	8915	7.4×10^{-15}	4.88
	10^{-11}	849	14	14	640	13236	1.1×10^{-10}	7.46
	10^{-12}	1316	14	12	664	19279	1.5×10^{-10}	11.56

Bibliography

[1] E. Anderson et al, *Lapack users' guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[2] O. Axelsson, *A class of A-stable methods*, BIT, 9:185-199, 1969.

[3] A. Bellen, R. Vermiglio, M. Zenarro, *Parallel ODE-solvers with stepsize control*, Journal of Comput. Appl. Math., 31:277-293, 1990.

[4] P. N. Brown, A. C. Hindmarsh and G. D. Byrne, *VODE: A variable coefficient ODE solver*, August 1992. Available at http://www.netlib.org/ode/vode.f.

[5] K. Burrage, *Parallel and sequential methods for ordinary differential equations*, Oxford University Press, Clarendon Press, Oxford, 1995.

[6] K. Burrage, *A special family of Runge-Kutta methods for solving stiff differential equations*, BIT, 18:22-41, 1978.

[7] K. Burrage, J. C. Butcher, *Stability criteria for implicit Runge-Kutta methods*, SIAM J. Numer. Anal., 16:46, 57, 1979.

[8] K. Burrage, J. C. Butcher, *Non-linear stability of a general class of differential equation methods*, BIT 20:185-203, 1980.

[9] K. Burrage, J. C. Butcher, F. H. Chipman , *An implementation of singly implicit Runge-Kutta methods*, BIT 20:326-340, 1980.

[10] K. Burrage and P. Moss, *Simplifying assumptions for the order of partitioned mutivalue methods*, BIT 20: 452-465, 1980.

[11] J.C. Butcher, *Implicit Runge-Kutta processes*, Math. Comput. 18:233-244, 1964.

[12] J.C. Butcher, *On the convergence of numerical solutions to ordinary differential equations*, Math. Comp., 20: 1-10, 1966.

[13] J.C. Butcher, *An algebraic theory of integration methods*, Math.Comp., 26:79-106, 1972.

[14] J.C. Butcher, *A stability property of implicit Runge-Kutta methods*, BIT, 15:358-361, 1975.

[15] J.C. Butcher, *On the implementation of implicit Runge-Kutta methods*, BIT, 16:237-240, 1976.

[16] J.C. Butcher, *A transformed implicit Runge-Kutta method*, J. Assoc. Comput. Mach., 26:731-738, 1979.

[17] J.C. Butcher, *General linear method: A survey*, Applied Mathematics, 1 : 273-284, 1985.

[18] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, Wiley, Chichester, 1987.

[19] J. C. Butcher, *Diagonally-implicit multi-stage integration methods*, Applied Numerical Mathematics, 11:347-363, 1993.

[20] J.C. Butcher, *General linear methods for the parallel solution of ordinary differential equations*, WSSIAA, 2:99-111, 1993.

[21] J.C. Butcher, *The parallel solution of ordinary differential equations and some special functions*, Approximations and Computation, ISNM 119, Birkhaeuser Verlag, Basel-Bostem-Berlin, 67-74, 1994.

[22] J.C. Butcher, *An introduction to DIMSIMs* Comp. Appl. Math., 14:59-72, 1995.

[23] J.C. Butcher, *General linear methods*, Computers Math. Applic. 31, No. 4/5:105-112, 1996.

[24] J.C. Butcher, *Order and Stability of parallel methods for stiff problems*, Advances in Computational Mathematics, 7: 79-96, 1997.

[25] J.C. Butcher, J.R. Cash, M.T. Diamantikis, *DESI methods for stiff initial value problems*, ACM Trans. Math. Software 22:401-422, 1996.

[26] J.C. Butcher and P. Chartier, *The construction of DIMSIMs for stiff ODEs and DAEs*, Report Series No. 308, The University of Auckland, New Zealand, 1994.

[27] J.C. Butcher and P. Chartier, *Parallel general linear methods for stiff ordinary differential and differential algebraic equations*, Applied Numerical Mathematics, 17:213-222, 1995.

[28] J.C. Butcher , P. Chartier, *A generalization of singly implicit Runge-Kutta methods*, Applied Numerical Mathematics 24:343-350, 1997.

[29] J.C. Butcher , M.T. Diamantikis, *DESIRE: diagonally extended singly implicit Runge-Kutta effective order methods*, Numer. Algorithms, 17:121-145, 1998.

[30] J. C. Butcher and Z. Jackiewicz *Diagonally implicit general linear methods for ordinary differential equations*, BIT, 33:452-472, 1993.

[31] J. C. Butcher and Z. Jackiewicz *Implementation of iagonally implicit multistage integration methods for ordinary differential equations*, SIAM J. Num. Anal., 34:2119-2141, 1997.

[32] P. Chartier, *Parallelism in the numerical solution of initial value problems for ODEs and DAEs*, Thesis, Université de Rennes I, France, 1993.

[33] F.H. Chipman, *A-stable Runge-Kutta processes*, BIT, 11: 384-388, 1971.

[34] M.Crouzeix *Sur la B-stabilité des méthodes de Runge-Kutta* , Numer. Math. 32:75-82, 1979.

[35] C.W. Cryer, *On the instability of high order backward-difference multistep methods*, BIT, 12:17-25, 1972.

[36] C.F. Curtiss and J.O. Hirschfelder, *Integration of stiff equations*, Proc. Nat. Acad. Sci., 38:235-243, 1952.

[37] G. Dahlquist, *Convergence and Stability in the numerical integration of ordinary differential equations*, Math. Scand., 4:33-53, 1956.

[38] G. Dahlquist, *A special stability problem for linear multistep methods*, BIT, 3: 27-43, 1963.

[39] K. Dekker, *Algebraic stability of general linear methods*, University of Auckland, Computer Science Repert No. 25, 1981.

[40] M.T. Diamantakis, *Diagonally extended singly implicit Runge-Kutta methods for stiff initial-value problems*, PhD thesis, Imperial College, University of London, UK, (1995).

[41] B.L. Ehle, *On Padé approximations to the exponential functions and A-stable methods for the numerical solution of initial value problems*, Research Rep. CSRR 2010, Dept. AACS, University of Waterloo, 1969.

[42] R.F. Enenkel, *DIMSEMs-Diagonally implicit single eigenvalue methods for the numerical solution of stiff ODEs on parallel computers*, PhD thesis, (1996), University of Toronto, Canada.

[43] R.F. Enenkel and K.R. Jackson, *DIMSEMs-Diagonally implicit single eigenvalue methods for the numerical solution of stiff ODEs on parallel computers*, Computational Mathematics, 7: 97-133, 1997.

[44] W.H. Enright, *Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations*, ACM trans. on Math. Software, 4:127-136, 1978.

[45] C.W.Gear, *The automatic integration of stiff ordinary differential equations*: in Information Processing 68, Proc. IFFP Congress, Edinnburgh; 187-193, 1969.

[46] C. W. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, Englewood Cliffs, N.J., 1971.

[47] C. W. Gear, *Parallel methods for ordinary differential equations*, Calcolo 25:1-20, 1988.

[48] E. Hairer, S. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I, Nonstiff Problems*, Springer-Verlag, Second Revised Edition, 1993.

[49] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff Problems*, Springer-Verlag, Second Revised Edition, 1996.

[50] A.C. Hindmarsh, *LSODE and LSODI, two new initial value ordinary differential equation solvers*, ACM/SIGNUM newsletter, 15(4): 10-11, 1980.

[51] P.J. van der Houwen and B. Sommeijer, *Parallel iteration of high order Runge-Kutta methods with stepsize control*, Journal of Computational and Applied Mathematics, 29:111-127, 1990.

[52] P.J. van der Houwen and B. Sommeijer, *Iterated Runge-Kutta methods on parallel computers*, SIAM Journal of Scientific and Statistical Computing, 12:1000-1028, 1991.

[53] P.J. van der Houwen and B. Sommeijer, *Block Runge-Kutta methods on parallel computers*, Z. Angnew Math. Mech., 72(1):3-18, 1992.

[54] P.J. van der Houwen, B. Sommeijer and W. Cousy, *Embedded diagonally implicit Runge-Kutta algorithms on parallel computers*, Mathematics of Computations, 58:135-159, 1992.

[55] P.J. van der Houwen and J.J.B. de Swart, *Parallel linear system solvers for Runge-Kutta methods*, Advances in Computational Mathematics, 7:157-181, 1997.

[56] P.J. van der Houwen and J.J.B. de Swart, *Triangularly implicit iteration methods for ODE-IPV solvers*, Applied Numerical Mathematics, 18:387-396, 1995.

[57] W. Hoffmann and J.J.B. de Swart, *Approximating Runge-Kutta matrices by triangular matrices*, BIT Numerical Mathematics, 37(2):346-354, 1997.

[58] A. Iserles and S.P. Nørsett, *On the theory of parallel Runge-Kutta methods*, IMA J. of Numer. Anal., 4:463-488, 1990.

[59] Z. Jackiewicz, R. Vermiglio, M. Zennaro, *Variable stepsize diagonally implicit multistage integration methods for ordinary differential equations*, Applied Numerical Mathematics, 16:343-367, 1995.

[60] K.R. Jackson, *A survey of parallel numerical methods for initial value problems for ordinary differential equations*, IEEE Transactions on Magnetics, 27:3792-3797, 1991.

[61] P. Kaps, *The Rosenbrock-type methods*, Nuemrical methods for stiff initial value problems, (Proceeding, OBerwolfach), G. Dahlquist and R. Jeltsch, Bericht nr. 9, Inst. f r Geometrie und Praktische Mathematik der RWTH Aachen, 1981.

[62] J.D. Lambert, *Numerical methods for ordinary differential systems, the initial value problem*, Wiley, Chichester, 1991.

[63] W.M. Lioen, J.J.B. de Swart and W.A. van der Veen, *Test set for IVP solvers*, http://www.cwi.nl/cwi/projects/IVPtestset.shtml, Test set for IVP solvers, 1996.

[64] J.J.H. Miller, *On the location of zeros of certain classes of polynomials with applications to numerical analysis*, J. Inst. Maths Applics, 8:397-406, 1971.

[65] S.P. Nørsett and P.G. Thomson, *Embedded SDIRK-methods of basic order three*, BIT, 24:634-646, 1984.

[66] G. Pólya aand G. Szegö, *Problems and theorems in Analysis*, Vol 1, Springer Verlag, 1972.

[67] A. Prothero and A. Robinson, *On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations*, Math. Comp., 28:145-162, 1974.

[68] H.H. Robertson, *The solution of a set of reaction rate equations*, In:J. Walshed.: Numer. Anal., an Introduction, Academic Press, 178-182, 1966.

[69] L.F. Shampine, *Implementation of Implicit formulas for the solution of ODEs*, SIAM Journal of Scientific and Statistical Computing, 1:103-118, 1980.

[70] L.F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, New York, 1993.

[71] B. Sommeijer and W. Cousy, P.J. van der Houwen, *A-stable parallel block methods for ordinary and integro-differential equations*, PhD thesis, Universiteit van Amsterdam, CWI, Amsterdam, 1992.

[72] J.J.B. de Swart, *Parallel software for implicit differential equations*, PhD thesis, CWI, Amsterdam, 1997.

[73] O.B. Widlund, *A note on unconditionally stable linear multistep methods*, BIT, 7:65-70, 1967.

[74] J. Van Wieren, *Using diagonally implicit multistage integration methods for solving ordinary differential equations*, Naval Air Warfare Center Weapons Division, China Lake, California, Report, Part 1: Introduction and explicit methods, 1997.

[75] J. Van Wieren, *Using diagonally implicit multistage integration methods for solving ordinary differential equations*, Naval Air Warfare Center Weapons Division, China Lake, California, Report, Part 2: Implicit methods, 1997.

[76] LAPACK routines available at http://www.netlib.org/lapack .