General linear methods with inherent Runge-Kutta stability

William Wright

A thesis submitted for the degree of
Doctor of Philosophy at The University of Auckland, 2002.

Abstract

General linear methods were derived approximately thirty years ago as a unifying approach for the study of consistency, stability and convergence of the Runge-Kutta and the linear multistep methods. Their discovery opened the possibility of obtaining essentially new methods which were neither Runge-Kutta nor linear multistep methods nor slight variations of these methods. It was hoped that general linear methods would exist which are practical and have advantages over the traditional methods. Locating such practical methods has proved difficult though, for several reasons. For example, the complexity of the order conditions becomes very high, making it difficult to even find the required conditions in many cases let alone solve them.

Several simplifying assumptions must be made, which limit this large class to situations where practical methods are likely to exist. The first assumption is that the stage order is equal to the overall order of the general linear method. This results in methods which, among other things, are not affected by the order reduction phenomenon. The second assumption is that a Nordsieck vector is passed from step to step. This enables such methods to vary the stepsize in a convenient and practical way. The final assumption is that the stability regions of the general linear methods should be identical to those of corresponding Runge-Kutta methods.

The first two assumptions are satisfied by the structure of the U and V matrices of the general linear methods. In order to satisfy the last assumption sufficient conditions are developed. These conditions result in a class of general linear methods with a property known as inherent Runge-Kutta stability (IRKS). The IRKS conditions relate the coefficient matrices of the general linear method with a doubly companion matrix X to satisfy

$$BA = XB, \qquad BU \equiv XV - VX, \qquad \sigma(V) = \{1, 0\}.$$

Constructing general linear methods with the IRKS property in the most general way possible is the main aim of this thesis. To derive these methods a transformation is used; this transformation brings all methods of this class into a particular form, which allows the construction using only linear operations.

Several special properties of methods with the IRKS property are introduced. For example, conditions which show that the ESIRK methods are a special case of the IRKS methods are introduced. This then allows the introduction of a new class of ESIRK methods which may have advantages over those already known. Also, methods which have a property known as strong stiff accuracy are developed which make them similar to strictly stable Runge-Kutta methods. Methods with strong stiff accuracy are likely to be considered good, particularly because they are the most suitable amongst the IRKS methods for the solution of differential algebraic equations.

The theoretical properties of the general linear methods with IRKS are investigated using various implementations from fixed stepsize and fixed order to variable stepsize and variable order codes. The IRKS methods are experimentally compared with several traditional methods, which provides evidence that the IRKS methods may in fact be suitable to form the kernel of an efficient solver of ordinary differential equations.

Acknowledgements

Many of the wonderful opportunities that have presented themselves to me are due to the kindness and support of my supervisor Prof John Butcher. He has basically taught me how to do mathematics and his outstanding contributions to the field of numerical analysis have always been of encouragement to me. I would like to thank him for believing in me, and helping me believe in myself.

My co-supervisor Dr Robert Chan has always been extremely encouraging towards me and has made himself available for discussions on any topic whenever I wished. He also checked drafts of my thesis and contributed several very useful suggestions.

I wish to express my sincere thanks to Dr Allison Heard for the many hours she has spent checking my work, not only in this thesis but in research papers I have submitted. Her frequent suggestions have helped me greatly improve the way in which I present my work.

As a means of presenting our work, the Numerical Analysis group at Auckland have weekly meetings. I would like to thank the members of this group for offering support and informative discussions on many aspects of my work. In addition to the above mentioned people I would particularly like to thank my fellow PhD students Nicolette Moir and Shirley Huang, who are now not only colleagues but also close friends.

There are several researchers who I have been fortunate enough to meet during the course of my doctoral research. Some of these people have spent a lot of time with me discussing several aspects of numerical analysis and my thesis research in particular. I would especially like to acknowledge Dr Tina Chan, Dr Philippe Chartier, Dr David Chen, Prof Zdzisław Jackiewicz, Dr Chris Kennedy, Dr Ander Murua and Dr Philip Sharp for these stimulating discussions.

In terms of financial assistance, I have been very fortunate to have received two major scholarships. The Top Achiever Doctoral scholarship funded by the New Zealand government and a Doctoral scholarship funded by The University of Auckland. These scholarships have resulted in me not having to work as many hours teaching or tutoring as many of my fellow students have had to I especially wish to recognise this fact. The graduate research fund sponsored by The University of Auckland enables graduate students to attend overseas conferences. I wish to thank the University for giving me the opportunity to attend conferences overseas on two occasions. These conferences made it possible to meet many of the researchers in the field of numerical analysis of ordinary differential equations, which was both stimulating and rewarding. Prof John Butcher also kindly supported my overseas trips with the help of his Marsden fund.

Finally I wish to thank my family and friends for all the wonderful times we have had, it is without doubt that I would not have lasted through the thesis without your encouragement and support. I would like to dedicate this thesis to my late brother Alban Anderson, there is never a day I do not think about you, Albs.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Ordinary differential equations are often used to describe physical systems. The solution of such equations gives valuable insight into how the system evolves and what the effect of changes in the system are. In general, it is extremely difficult, if not impossible, to obtain an analytic solution. It is for this reason that numerical methods, which find approximate solutions, are an invaluable tool. A very brief description of traditional numerical methods for ordinary differential equations is given along with several more recent modifications, some of which are within the problem and some of which are within the numerical method.

1.1 Framework

Assume that a physical system can be effectively modelled using an ordinary differential equation. Depending on the physical system which is being modelled, the resulting ordinary differential equation will naturally be expressed in one of two forms. The first is known as nonautonomous form where the ordinary differential equation has the structure

$$y'(x) = f\big(x, y(x)\big).$$

The independent variable x is often thought of naturally in physical systems as time. The dependent variable, $y(x)$, is the solution. The second, known as autonomous form, has the structure

$$y'(x) = f\big(y(x)\big).$$

It is an advantage to consider only the autonomous form, as problems in nonautonomous form can be easily represented in autonomous form by adding the extra trivial differential equation $x' = 1$. One necessary requirement of a numerical method is that it can solve this differential equation exactly. It is therefore preferable to consider only differential equations in autonomous form. It is important to realise that $y(x)$ may in fact be a vector valued function, that is to say

$$y(x) : \mathbb{R} \to \mathbb{R}^m, \qquad f\big(y(x)\big) : \mathbb{R}^m \to \mathbb{R}^m,$$

where m is the dimensionality of the system. If the ordinary differential equation is given along with the initial condition $y(x_0) = y_0$, then

$$y'(x) = f\big(y(x)\big), \qquad y(x_0) = y_0,$$

is known as an initial value problem. An initial value problem is quite natural in the context of a physical system. The state of the physical system is known now, denoted by $y(x_0)$ and it is of interest to estimate the state a certain time in the future, denoted by $y(x_N)$. It is essential to consider whether such a solution actually exists and if it does exist, whether it is unique. These questions can be answered for both the exact and numerical solutions using what is known as a Lipschitz condition which is defined as follows.

Definition 1.1 *A function $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ satisfies a Lipschitz condition if there exists a constant L and a norm $\|\cdot\|$ such that for any $Y, Z \in \mathbb{R}^m$, $\|f(Y) - f(Z)\| \leq L\|Y - Z\|$.*

This leads to the following existence and uniqueness result, the proof of which can be found in [33].

Theorem 1.2 *Given an initial value problem, where $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is continuous in the independent variable and satisfies a Lipschitz condition, then there exists a unique solution to this problem.*

From now on it will be assumed that $f\big(y(x)\big)$ satisfies a Lipschitz condition, and therefore a unique solution exists. It could also be asked, if given a unique solution exists, how sensitive this solution is to perturbations in the initial condition. This can also be answered, see [33], using a Lipschitz argument.

1.2 Types of differential equations

Assume that $f\big(y(x)\big)$ satisfies a Lipschitz condition, and therefore a unique solution exists. For a very long time practitioners would solve ordinary differential equations using their favourite numerical method. It would have come as quite a surprise when Curtiss and Hirschfelder [58] in 1952 realised that certain types of problems were in fact best solved using certain types of numerical method. This was the first such tailoring of the method to the problem which has become common practice today. The phenomenon they recognised is known as stiffness. Since then problems have been divided into stiff and non-stiff problems. Many attempts have been made to obtain a definition of stiffness, however it is best understood by what goes wrong with certain numerical methods. Stiffness is the practical difficulty faced when numerically solving some initial value problems. In the first line of the first section of Hairer and Wanner's book [78] they write "Stiff problems are problems in which explicit methods don't work.". The main reason explicit methods don't work for stiff problems is that the bounded stability domains (which will be discussed in Section 2.7) of these methods force the numerical method to take excessively small stepsizes in relation to the smoothness of the solution. That is, stability rather than accuracy restricts the stepsize. Stiffness often occurs when some components of the solution decay much more rapidly than others. This results in problems which have highly stable exact solutions but have highly unstable numerical solutions. It is worth noting that there are in fact problems, oscillatory problems for example, in which explicit methods don't work but are not stiff problems.

The following one dimensional example exhibits the stiffness phenomenon

$$y'(x) = \lambda\big(y(x) - g(x)\big) + g'(x), \qquad y(x_0) = y_0,$$

where λ is negative and has large magnitude and $g(x)$ is a smooth and slowly varying function. The exact solution of this problem is

$$y(x) = \big(y_0 - g(x_0)\big)\exp\big(\lambda(x - x_0)\big) + g(x).$$

Since λ negative and has large magnitude the first terms decays very quickly leaving $y(x) \approx g(x)$. The error of a numerical method will depend on the stepsize used and the derivatives of $g(x)$. On the other hand stability will depend on the stepsize and λ. Since λ is large and negative the stepsize will have to be excessively small to ensure numerical stability.

In Figure 1.1 two plots are given, both containing the phase portrait of the example above. In the left plot the explicit Euler method is used with 38 steps, on the right the implicit Euler method is used with a stepsize several times as large. In this example $g(x) = \cos(x)$ and $\lambda = -50$.

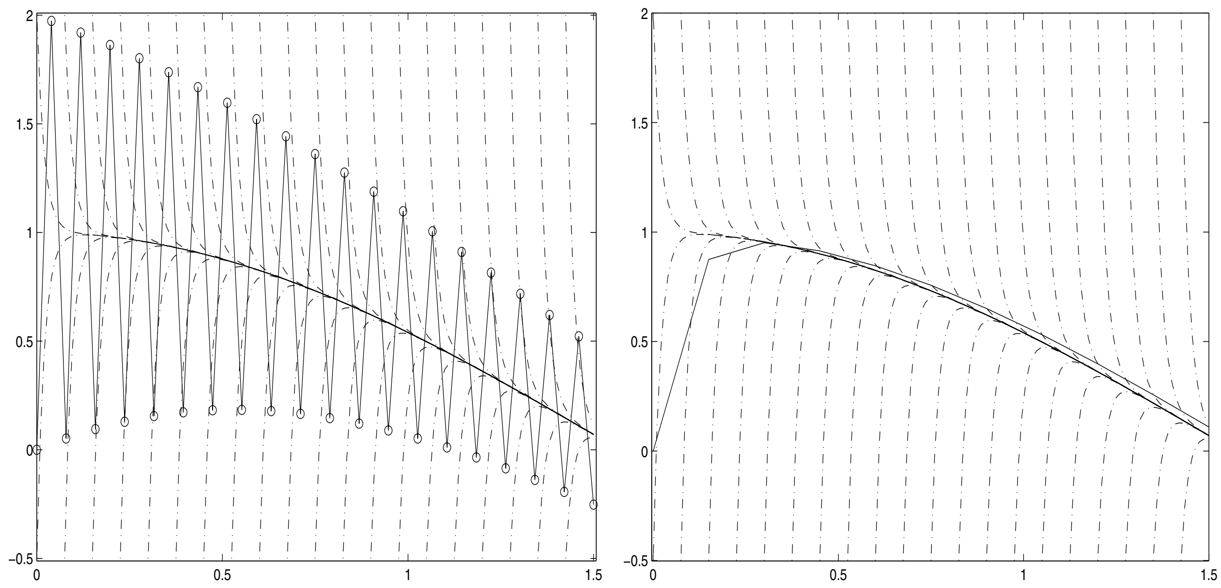


Figure 1.1: Explicit and implicit Euler methods on the problem $y'(x) = \lambda\big(y(x) - g(x)\big) + g'(x)$.

The explicit Euler method oscillates around the solution. The oscillations are slowly damped away; however if the stepsize was much larger the oscillations would diverge, whereas the implicit Euler method converges quickly to the exact solution even with a much larger stepsize.

To integrate stiff equations effectively, numerical methods must be implicit. The reason is that all explicit methods have bounded regions of absolute stability. The bigger the region is the less severe the restriction on the stepsize. It is therefore an ideal situation if there is no restriction on the stepsize at all. To address this issue Dahlquist [60] introduced the concept of A-stability. A numerical method is said to be A-stable if its stability region contains the left half plane. Numerical methods which are A-stable are suitable for the solution of stiff differential equations. These issues and related ones are discussed further in Section 2.7.

In 1982 Gear wrote “Although it is common to talk about ‘stiff differential equations,’ an equation per se is not stiff, a particular initial value problem for that equation may be stiff, in some regions, but the sizes of those regions depends on the initial values and the error tolerance.” This is one of the main reasons that obtaining a robust definition of stiffness has proved so difficult. If the error tolerance was extremely small then to obtain the required accuracy the stepsize would need to be smaller than the requirements on stability. It would therefore be of extreme benefit if an implementation was able to determine whether the computation was being affected by stiffness or not, and if it was, then an appropriate implicit method should be used.

1.3 A brief history

As previously mentioned, in general it is extremely difficult, if not impossible, to obtain an analytic solution to an arbitrary ordinary differential equation. Therefore, numerical methods are invaluable tools for obtaining information about the exact solution. Numerical methods are often called discrete solutions as they attempt to break the integration interval into several small steps.

The first numerical method which forms the basis for all others is known as the Euler method, after its founder. This method is extremely simple; given the solution y_{n-1} at some point, move along the tangent at this point $f(y_{n-1})$ a certain distance $x_n - x_{n-1}$. This is represented as

$$y_n = y_{n-1} + (x_n - x_{n-1})f(y_{n-1}).$$

The integration interval in this case $x_n - x_{n-1}$ is known as the stepsize and is denoted as h.

It is worth looking at a simple example; apply four steps of the explicit Euler method to the problem

$$y'(x) = x^2\cos(x) + \frac{2}{x}y(x), \qquad y(x_0) = y_0,$$

which has an analytic solution

$$y(x) = x^2\sin(x) + x^2\left(\frac{y_0}{x_0^2} - \sin(x_0)\right).$$

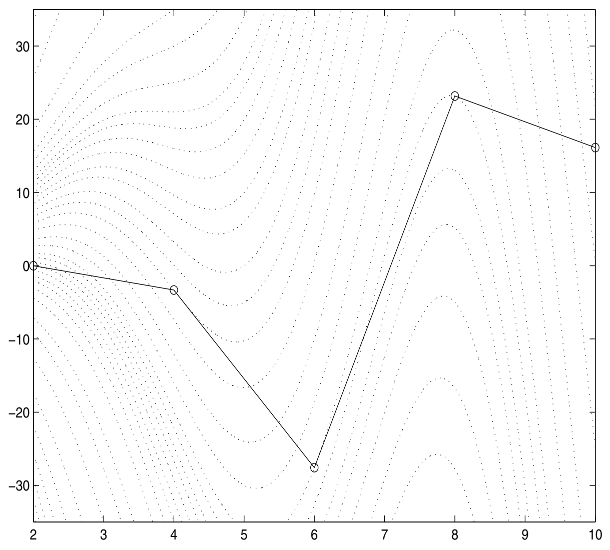


Figure 1.2: Explicit Euler method on the problem $y'(x) = x^2\cos(x) + \frac{2}{x}y(x)$ with $y(2) = 0$.

An important observation is that after each step of the numerical method a new initial value problem is approximated. This initiates the idea of the local error where after each step the incoming data is assumed to be exact. Therefore, accuracy is measured by comparing the numerical solution over one step with the corresponding Taylor series expansion of the exact solution, given by

$$y(x_n) = y(x_{n-1}) + hy'(x_{n-1}) + \frac{h^2}{2!}y''(x_{n-1}) + \cdots .$$

The Euler method agrees with the Taylor series expansion of the exact solution up to the first power of the h. The method is therefore of order one. There are generally two main approaches to increasing the accuracy or order of an integration method. These approaches are in some sense opposites. The first approach increases accuracy by including more information from the past; these methods are known as multivalue methods. The second increases accuracy by approximating the solution at several internal points during the integration interval; these methods are known as multistage methods.

The first methods adopting the multivalue philosophy were derived in the 1883 paper by Bashforth and Adams, [4] and are now known as Adams-Bashforth methods, even though from the title of the paper it is clear that Adams developed the numerical component of the work. The Adams-Bashforth methods are a special case of methods known as linear multistep methods which have the general form

$$y_n = \alpha_1 y_{n-1} + \cdots + \alpha_k y_{n-k} + h\big(\beta_0 f(y_n) + \beta_1 f(y_{n-1}) + \cdots + \beta_k f(y_{n-k})\big).$$

This is known as a k step linear multistep method, as information from the last k steps is required to update the solution. The special case when $\alpha_1 = 1$, $\alpha_2 = \cdots = \alpha_k = 0$, are the Adams-Bashforth methods. In [4], the case when $\beta_0 \neq 0$ was also introduced. This case was developed independently by Moulton [99] in 1926 and are now known as the Adams-Moulton methods. Other linear multistep methods were derived by Nyström [104] in 1925 and Milne [97] in 1926. To obtain the same accuracy as the Adams-Bashforth methods, the Adams-Moulton methods need less information from the past. It is also worth noting that the error constants are smaller for the implicit methods. However, in this case y_n does not depend completely on known information but rather an algebraic equation which must be solved to yield y_n. To overcome this problem Adams used Newton's method to solve for y_n. This is common practice today for the solution of stiff problems. On the other hand Milne [98] in 1949 suggested that methods should be implemented using a predictor, corrector pair. This means that y_n is predicted by the Adams-Bashforth method and then corrected using the Adams-Moulton method. The two main approaches are often expressed as $\mathrm{P(EC)}^N\mathrm{E}$ or $\mathrm{P(EC)}^N$, where the upper index implies the corrector is iterated N times or to convergence. Since the final approximation y_n is found using only known information these methods are explicit in nature. Methods of this type also have a simple type of error estimate known as the Milne's device. In this case the scaled difference of the predicted and corrected solution can be used as an estimate of the error in the corrected solution.

To implement linear multistep methods in an adaptive fashion which includes the possibility of variations in the order as well as the stepsize, several problems must be overcome. Characteristically these are starting values, error estimation, interpolation and change of stepsize and order. The first major adaptive code was developed by Krogh [94] in 1969. Since the linear multistep methods above require the approximations at equidistant points, the integration coefficients need to be computed at every step which is a big disadvantage of this approach. In 1962, Nordsieck [102] remarked that all methods of numerical integration are equivalent to finding an approximating polynomial. To represent this polynomial Nordsieck proposed using the first scaled $k+1$ derivatives, which includes the order zero derivative, thus making changing stepsize much more convenient. The information passed from step to step in this case is known as a Nordsieck vector. The production codes LSODE [82] of Hindmarsh and VODE [9] of Brown, Bryne and Hindmarsh use Nordsieck vectors.

As the order of explicit linear multistep methods increases the relative size of the stability regions decreases. This is primarily due to the fact that there is only one function evaluation performed at each step. For the $\mathrm{P(EC)}^N\mathrm{E}$ or $\mathrm{P(EC)}^N$ approaches, the relative stability domain is larger as there are more function evaluations. For this reason the stepsize of linear multistep methods must be much smaller than that of a comparative Runge-Kutta method. It is impossible to talk about linear multistep methods without mentioning the name of Dahlquist who introduced several fundamental properties to the study of numerical methods. In 1956, Dahlquist [59], introduced the concepts of consistency, stability and convergence. He showed that if a numerical method is consistent and stable then the method is necessarily convergent.

It was also shown how the coefficients of a linear multistep method can be characterized using the polynomials $\rho(z)$ and $\sigma(z)$, where

$$\rho(z) = \alpha_k z^k + \alpha_{k-1} z^{k-1} + \cdots + \alpha_1 z + \alpha_0,$$
$$\sigma(z) = \beta_k z^k + \beta_{k-1} z^{k-1} + \cdots + \beta_1 z + \beta_0.$$

A linear multistep method is consistent if

$$\rho(1) = 0, \qquad \rho'(1) = \sigma(1),$$

and stable if all the zeros of $p(z)$ lie in the unit circle and if any lie on the boundary they must be simple. The order p of a linear multistep method is the largest value satisfying

$$\rho\big(\exp(z)\big) - z\sigma\big(\exp(z)\big) = O(z^{p+1}).$$

Even though it is possible to find linear multistep methods with order $2k$ it was also shown how order of accuracy is related to stability, this is now known as the "Dahlquist barrier", which limits the order of a convergent linear k-step method to $k+1$ if k is odd and $k+2$ if k is even.

When $\beta_1 = \beta_2 = \cdots = \beta_k = 0$ these methods are known as the backward differentiation formulae, first discovered by Curtiss and Hirschfelder [58]. These methods, as the name suggests, are based on numerical differentiation. As was mentioned in the previous section, methods suitable for the numerical integration of stiff problems require large stability regions. However, Dahlquist in [60] showed that linear multistep methods are only A-stable for orders two or less. In spite of this fact these methods have stability regions which are wide enough to be considered for the solution of stiff problems. For many stiff problems, in which the eigenvalues do not have large imaginary components, the stability requirement does not limit stepsize. Even so, most practical implementations of backward differentiation formulae are based on orders five and less. One of the reasons for their success is that the implementation cost for these methods is low. At every integration step, the total costs for the solution of a nonlinear system of m differential equations requires $O(m^3) + O(m^2)$ operations. The most widely used adaptive codes for stiff differential equations are based on the backward differentiation formulae, the first of which was written by Gear [71], in 1980. This subroutine, which used Nordsieck vectors to represent the data, was part of the program named DIFSUB.

To obtain an even deeper understanding of stiff problems Dahlquist [61] in 1975 introduced the concept of G-stability, which now comes under the banner of non-linear stability. Consider two exact solutions which possess the property

$$\|y(x) - z(x)\| \leq \|y(x_0) - z(x_0)\|, \qquad x > x_0.$$

G-stability requires an equivalent discrete condition for two numerical solutions. However, it turned out that this type of condition was much more appropriate in the study of Rung-Kutta methods. The analogous condition known as B-stability was found by Butcher in 1975 on the flight home from the conference in which Dahlquist presented G-stability. Linear and non-linear stability are discussed further in Section 2.7.

The first methods adopting the multistage philosophy are generally attributed to Runge [107] in 1895. Heun, [81] also developed some low order methods in 1900. Kutta, [95] in 1901 completely characterised all methods of order four and proposed some methods of order five. In 1925, Nyström [104] corrected a fifth order method of Kutta by realising a typographical error Kutta had made. Nyström also developed methods which for second order differential equations were cheaper to compute than transforming the problem into a first order system.

These multistage methods are now known as Runge-Kutta methods and the computations performed in one step of the method are as follows

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

The Y_i are known as the internal stages and are approximations to the solution at various points $y(x_{n-1} + c_i h)$, generally within the current integration interval. The coefficients of the method, often expressed in the tableau

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array},$$

are primarily used to increase the order of the method. Gill's important paper [72] pointed the way to a full analysis of the order conditions by considering the Taylor series expansions of the numerical and the exact solutions for autonomous problems. Merson [96] used what are now known as elementary differentials to express the exact solution, but failed to see that the numerical solution could also be represented using elementary differentials. This was shown to be possible in the work of Butcher [16] and a complete theory of order was developed. Below are the order conditions needed to ensure a method has order four

$$\begin{array}{llll} b^T e = 1, & b^T c = \frac{1}{2}, & b^T A c = \frac{1}{6}, & b^T c^2 = \frac{1}{3}, \\ b^T A A c = \frac{1}{24}, & b^T A c^2 = \frac{1}{12}, & b^T C A c = \frac{1}{8}, & b^T c^3 = \frac{1}{4}, \end{array}$$

where e is the vector of ones, c^2 and c^3 are the vectors obtained by raising each component of c to the appropriate power and C is a diagonal matrix with c on the diagonal. The number of order conditions for a particular order is equivalent to the number of rooted trees of that order, and each tree can be associated with an elementary differential; more details are given in Sections 2.4 and 2.5. In [16] the simplifying assumptions were introduced. The simplifying assumptions consist of a set of conditions which, when satisfied, reduce the number of conditions that are needed to obtain a particular order. The following are the C, D, E simplifying assumptions,

$$C(\eta) : \sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{1}{k} c_i^k, \qquad i = 1, 2, \ldots, s, \quad k = 1, 2, \ldots, \eta,$$
$$D(\xi) : \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = \frac{1}{k} b_j (1 - c_j^k), \qquad j = 1, 2, \ldots, s, \quad k = 1, 2, \ldots, \xi,$$
$$E(\xi, \eta) : \sum_{i=1}^{s} \sum_{j=1}^{s} b_i c_i^{k-1} a_{ij} c_j^{l-1} = \frac{1}{l(k+l)}, \qquad k = 1, 2, \ldots, \xi, \quad l = 1, 2, \ldots, \eta.$$

There are certain restrictions on the values η, ξ for explicit methods, see for example [33]. The simplifying assumptions made it possible to derive methods of high order. Two extremely important papers, the first by Butcher [22], in 1972 and the second by Hairer and Wanner [77], developed the algebraic theory of Runge-Kutta methods. For an implementation to be adaptive local error estimates need to be part of the design. This is generally achieved by a technique known as embedding. Where a pair of Runge-Kutta methods of differing orders are derived, the difference is used as an approximation to the local error. The local truncation error estimate derived in this way is usually not asymptotically correct. Efficient Runge-Kutta pairs have been derived by, for example, Dormand and Prince [63] and Verner [119]. These were constructed by minimising the contributions of the higher order elementary differentials.

Several codes based on these pairs are constructed. For example Shampine [7] developed RKSUITE and Hairer and Wanner [78] developed DOPRI5 and DOP853. Constructing methods with inbuilt local error estimates is costly, designing methods which compute a suitable estimate which can be used to vary the order is even more difficult. Thus none of the codes mentioned above has a variable order facility. Recently it has become more important to have approximations continuously throughout the integration interval rather than just at the grid points. Continuous solutions are important because this allows the code to develop a stepsize pattern which is not continually interrupted so as to provide enough information for plotting, event location, Poincaré sections or delay differential equations. Continuous Runge-Kutta methods require additional stages above what is required by the order conditions.

In 1964 Butcher [17] introduced implicit methods based on Gauss, Radau and Lobatto quadrature formulae. The Gauss methods are A-stable and have order $2s$. The Radau and Lobatto methods of order $2s-1$ and $2s-2$ respectively turned out not to be A-stable. This led Ehle [64] in 1969 to construct A-stable Radau IA, Radau IIA, Lobatto IIIA, Lobatto IIIB and low order Lobatto IIIC methods. Lobatto IIIC methods were generalised by Chipman [57] in 1971. All these methods use simplifying assumptions which are given above. However, for an s stage implicit Runge-Kutta method the nonlinear system of equations now has sm equations. This means the Newton method at every integration step requires $O(s^3m^3) + O(s^2m^2)$ operations. This cost increases rapidly as s or m increases. This is enough in itself to consider cheaper alternatives. It is also difficult to derive methods for estimating the local truncation error or estimating the error of the next highest order formula. Consequently developing variable stepsize, variable order implementations is again difficult.

To reduce the computational cost of implicit Runge-Kutta methods, Alexander in 1977 introduced the so called diagonally implicit Runge-Kutta methods [3]. When the matrix A is lower triangular the nonlinear system can be solved stage by stage sequentially. This reduces the total cost to $O(sm^3) + O(sm^2)$, because there is only one Jacobian evaluation and matrix factorisation necessary per integration step if the diagonal elements of A are equal. This is considerably cheaper than the cost of a fully implicit Runge-Kutta method and is comparable to s steps of the backward differentiation formulae. Unfortunately the stage order of these methods is only one. The concept of stiff order was first introduced for linear problems by Prothero and Robinson [105] and extended to nonlinear problems by Frank, Schneid and Überhuber in [67]. The main idea of stiff order is that when a stiff problem is solved the order achieved by the numerical method is not the classical order p but closely related to the stage order q. The stiff order is never greater than the $q-1$. This means the diagonally implicit methods are not really suitable for stiff problems.

The search for high accuracy of both the internal and external stages and good stability, while retaining computational efficiency, remained incomplete. To achieve these aims Nørsett in 1976 developed what are now known as the singly implicit methods [103]. In this case the matrix A has a one point spectrum. This allows the stages and the output approximation to be of the same order. These methods were furnished with error estimates by Burrage [11] in 1978. Butcher in 1979 showed how to transform A to Jordan canonical form [24]. In 1980, Burrage, Butcher and Chipman developed the variable stepsize variable order code STRIDE [14]. However, the required transformation increases the computational cost by $O(s^2m)$ every integration step. This extra cost makes these methods less competitive, especially for small and medium dimensional problems. As a design choice the order p, the stage order q and the number of stages s were equal. This forces some of the abscissae to lay outside of the integration step for orders greater than two. This causes difficulties when solving nonlinear problems or problems with discontinuities.

In a surprising article [21], Butcher introduced the idea of effective order. A Runge-Kutta method a is said to have effective order p if there is a method d such that dad^{-1} has order p. In order to ensure that a step of either dd^{-1} or $d^{-1}d$ leaves the initial condition intact, d^{-1} is defined as follows

$$\begin{array}{c|c} c-e & A-eb^T \\ \hline & -b^T \end{array}.$$

By introducing the starting method d, more freedom is available in the method a. Notice that

$$(dad^{-1})^n = da^n d^{-1},$$

therefore the starting method d and the finishing method d^{-1} only need to be used at the beginning and end of the computation respectively. The weaknesses in diagonally and singly implicit Runge-Kutta methods have motivated several modifications in the existing methods. Relaxing the order requirement of singly implicit Runge-Kutta methods to effective order was proposed by Butcher and Chartier [35]. The extra freedom due to effective order allows the abscissae to be freely chosen. On the other hand, Butcher and Cash [34] developed the diagonally extended singly implicit Runge-Kutta methods. For these methods A consists of a singly implicit block with some extra diagonal stages. Even though the linear algebra cost is about the same as a singly implicit method, the extra diagonal stages improve the performance of the method, because they have considerably smaller error constants. However there are still restrictions on the abscissae. These approaches were combined in the diagonally extended singly implicit Runge-Kutta methods with effective order [41]. The combined effect of effective order and the additional stages removed the restriction on the abscissae, making them more suitable for nonlinear problems and reduced the error constants, allowing the method to take larger stepsizes and therefore, reduced the overall cost. The cost of the transformations still makes these methods considerably more expensive than the diagonally implicit methods.

Even though general linear methods have been known since 1966 when Butcher [19] constructed these methods as a unifying framework of the traditional methods, very few methods which are significantly different from the traditional methods have been developed. The Runge-Kutta and linear multistep methods are at extreme ends of the general linear methods spectrum. Most new general linear methods constructed add some Runge-Kutta flavour to linear multistep methods, such as the hybrid, generalised multistep and modified multistep methods developed by Gear [68], Gragg and Stetter [73] and Butcher [18] respectively. Or alternatively add a linear multistep flavour to the Runge-Kutta methods, such as the pseudo Runge-Kutta methods of Byrne and Lambert [52] and the multistep Runge-Kutta methods of Burrage [12].

Rather than modify the traditional methods, it seems more desirable to decide what properties a method should have and then seek methods within the class of general linear methods with these properties. Admittedly this is very difficult and has been the main reason for the lack of genuinely new methods. From the discussion above it is clear what properties a practical method should have. For example, the A matrix should be lower triangular so as to obtain the low costs of the diagonally implicit Runge-Kutta methods. The method should have high stage order q which is comparable to the overall order p. This is necessary for stiff problems because the stiff order is approximately the stage order. It also enables dense output and asymptotically correct error estimates to be constructed, see Sections 4.4 and 4.10. For stiff problems A-stability is sought, and for non-stiff problems large stability regions are desirable, see Section 2.7. Generalising the idea of effective order further, see Section 2.3, allows methods with high stage order to be constructed without requiring A to be singly implicit, but rather diagonally implicit. An example of such methods are the diagonally implicit multistage integration methods introduced by Butcher [28]. In generalising order in this way the concept of stability becomes much more complicated. Since Runge-Kutta methods have excellent stability, it was required that these methods have “Runge-Kutta stability”. This can be thought of as the stability domains being equivalent. One of the primary problems with diagonally implicit multistage integration methods is that to obtain a Runge-Kutta stability region, very complicated systems of nonlinear equations must be solved. Butcher, Jackiewicz and Mittelmann developed sophisticated numerical searches to find methods of orders five through eight see [43], [44], [46]. However, as a consequence of the choice in defining parameters, there is no flexibility available to reduce the higher order error.

To overcome this problem it would be nice to ask whether certain matrix relations exist such that Runge-Kutta stability is possible while leaving freedom to control the higher order error contributions, along with asymptotically correct error estimates and dense output for both explicit and implicit methods. This makes it possible to implement the methods in a variable stepsize, order and stiffness mode. Such methods exist with a property known as “inherent Runge-Kutta stability”. Constructing such methods is the main aim of this thesis.

1.4 Other problems

As is often the case, ordinary differential equations are not the best means of representing certain physical systems. Particular physical systems have properties which need information not only from the current state but from previous states to be modelled accurately. It is also possible that the physical system has invariants occurring in the solution, such as angular momentum or concentration, which should be conserved. It is possible to furnish the ordinary differential equations with these constraints or construct methods which automatically conserve them. As technology is ever changing it is important to seek new approaches to solve problems using these technologies. One such technology is parallel computers. What follows is a partial listing of some of the modifications to ordinary differential equations and the previously mentioned numerical methods. These are all major research areas for numerical analysts.

Often physical systems are modelled accurately using delay differential equations

$$y'(x) = f\big(y(x), y(x-\tau)\big),$$

or more generally, where the function depends on more than one retarded argument,

$$y'(x) = f\big(y(x), y(x-\tau_1), \ldots, y(x-\tau_n)\big).$$

An example of where a delay differential equation is natural is in the modelling of the spread of infectious diseases. The delay arises as the incubation period. The rate of change of the solution at time x is not only determined by the solution at time x but at various previous times $x-\tau_1, \ldots, x-\tau_n$. That is, the solution depends on its history. The complexity of the delays $\tau_1, \ldots, \tau_n$, determine the type of delay equation. The delays can be constant (constant delay case), functions of x (variable delay case), or functions of both x and y (state dependent case). The first main difference between delay differential equations and ordinary differential equations is that an initial value function $\phi(x)$ is required rather than just an initial value. A consequence of this is that even if $f(y,z), \tau(x,y), \phi(x)$ are C^∞ continuous, the solution $y(x)$ and $\phi(x)$ are unlikely to be more than C^0 continuous. Such discontinuities spread throughout the integration interval. Delay differential equations often result in interesting mathematical phenomena, for example, instability, limit cycles and periodic behaviour. Sometimes a physical system can be better modelled by requiring that the delay not be a fixed value $x-\tau$ but stretch out as the integration proceeds. Problems of this type are known as Integro differential equations and given by

$$y'(x) = f\left(y(x), \int_{x-\tau}^{x} K\big(\eta, y(\eta)\big)\, d\eta\right).$$

Computing the numerical solution for these problems is much more expensive; fortunately it is often the case that integro differential equations can be expressed as ordinary or delay differential equations by using new variables for the integral.

Often models require not only a differential equation but a purely algebraic constraint. This takes into account conservation laws, geometrical or kinematic constraints. These problems are known as differential algebraic equations. The form of the algebraic constraint is crucial in the difficulty of determining the solution. The index of the differential algebraic equation effectively measures the number of differentiations required to return the differential algebraic equation into a differential equation. An index 1 differential algebraic equation is given by

$$\begin{aligned} y'(x) &= f\big(y(x), z(x)\big), \\ 0 &= g\big(y(x), z(x)\big), \end{aligned}$$

provided g_z is non-singular. An index 2 differential algebraic equation is given by

$$\begin{aligned} y'(x) &= f\big(y(x), z(x)\big), \\ 0 &= g\big(y(x)\big), \end{aligned}$$

provided $g_y f_z$ is non-singular. An index 3 differential algebraic equation is given by

$$\begin{aligned} y'(x) &= f\big(y(x), z(x)\big), \\ z'(x) &= k\big(y(x), z(x), u(x)\big), \\ 0 &= g\big(y(x)\big), \end{aligned}$$

provided $g_y f_z k_u$ is non-singular. Differential algebraic equations can also be thought of as differential equations on manifolds, where the manifold is given by the algebraic constraint. As a differential algebraic equation can be converted into an ordinary differential equation it is possible to solve the ordinary differential equation using a numerical method already discussed. This approach is not the most convenient for several reasons. Apart from the difficulties of finding the underlying ordinary differential equation, this process can destroy the original structure of the problem, even transforming a stable problem into an unstable one. Another problem associated with index reduction is that in the new system the numerical solution does not need to satisfy the original algebraic equations and a drift from these constraints can occur. Therefore, differential algebraic equations are solved directly using methods for ordinary differential equations which have been adapted to solve differential algebraic equations.

In developing models used to describe physical systems, stochastic effects have been ignored until recently. The main reason for this is the difficulty in deriving solutions, partly due to the lack of suitable numerical methods and also the non-availability of sufficiently powerful computers. Stochastic differential equations have a natural place in the modelling of investment finance, turbulent flows, population dynamics, biological waste treatment and hydrology. The general form of an autonomous stochastic differential equation is

$$dy(x) = f\big(y(x)\big)dx + g\big(y(x)\big)dW(x),$$

where f is the slowly varying continuous component called the drift coefficient (an m vector valued function), g is the rapidly varying continuous component called the diffusion coefficient (an $m \times d$ matrix valued function) and $dW(x)$ is a d dimensional process having independent Wiener process components. The diffusion coefficient is generally interpreted using either Itô or Stratonovich calculus.

Numerical methods can be developed for stochastic differential equations but they need to be developed in their own right, rather than using simplistic adaptations of deterministic numerical methods. However, numerical methods have been derived using rooted tree theory.

A new and exciting area of numerical analysis known as geometric integration has become popular in the last 10 or so years. The idea is to preserve qualitative and geometric features of the differential equation when it is discretised. Below is a partial listing of different types of qualitative properties that a numerical methods would like to preserve:

1. *Geometric structure*: Properties of the phase space in which the system evolves yields much insight into the global properties of solutions.

2. *Conservation laws*: Many physical systems have underlying invariants. Generally these are conservations of integrals such as mass, energy or momentum. Also quantities can be conserved along the flow of the differential equation such as fluid density or potential vorticity.

3. *Symmetries*: Many systems are invariant under the actions of symmetries. Examples are Lie group, scaling and involution symmetries.

4. *Asymptotic behaviours*: Interested in numerical methods which preserve structurally stable features of the solution, such as invariant sets, and invariant curves.

5. *Solution orderings*: Often in parabolic partial differential equations the solutions satisfy $y(x_0) < z(x_0)$ then $y(x) < z(x)$. That is a preservation of solution orderings.

By preserving qualitative structure it is hoped that some improvements in the computations will result. It often turns out that only a subset of all the qualitative features can be preserved and this results in requiring somewhat of a subjective judgement. Also, it is usually the case that variable stepsize implementations destroy the qualitative properties, thus making computations more costly.

Ordinary differential equations are used to model a wide variety of physical systems. It is often the case that the length of the integration interval, the number of equations in the system and also the sparsity of the problem can vary considerably from one system to another. In problems with these characteristics, it has been shown that the use of parallel computers can provide a solution. Even though most numerical methods are sequential in nature, several methods have been recently developed to take advantage of the speed up that parallel computers can provide. There have been several attempts to exploit parallelism and these can be classified into three main categories:

1. *Parallelism across the problem*: Divide the computational cost of evaluating the function, using different processors for different components.

2. *Parallelism across the steps*: The solution at different points is evaluated simultaneously. This type of parallelism requires a large number of processors.

3. *Parallelism across the method*: Parallelism which is inherently available within a method. For example, this could include the parallel evaluation of the stages of a multistage method. This approach is effective for problems with complex function evaluations.

CHAPTER 2

General linear methods

Traditional methods for solving ordinary differential equations generally fall into two main classes: linear multistep (multivalue) and Runge-Kutta (multistage) methods. Both of these classes have well known advantages and disadvantages. In 1966 Butcher [19] introduced general linear methods as a unifying framework for the traditional methods to study the properties of consistency, stability and convergence and to formulate new methods with clear advantages over the traditional methods.

2.1 Framework

A general linear method used for the numerical solution of an autonomous system of ordinary differential equations

$$y'(x) = f\big(y(x)\big), \qquad y(x) \in \mathbb{R}^m, \qquad f\big(y(x)\big) : \mathbb{R}^m \to \mathbb{R}^m,$$

is both multistage and multivalued. Denote the internal stage values of step number n by

$$Y_1^{[n]}, Y_2^{[n]}, \dots, Y_s^{[n]}$$

and the derivatives evaluated at these steps by

$$f\big(Y_1^{[n]}\big), f\big(Y_2^{[n]}\big), \dots, f\big(Y_s^{[n]}\big).$$

At the start of step number n, r quantities denoted by

$$y_1^{[n-1]}, y_2^{[n-1]}, \dots, y_r^{[n-1]},$$

are available from approximations computed in step $n-1$. Corresponding quantities

$$y_1^{[n]}, y_2^{[n]}, \dots, y_r^{[n]},$$

are evaluated in the step number n. For compactness of notation, introduce vectors $Y^{[n]}$, $f\big(Y^{[n]}\big)$, $y^{[n-1]}$ and $y^{[n]}$ given by

$$Y^{[n]} = \begin{bmatrix} Y_1^{[n]} \\ Y_2^{[n]} \\ \vdots \\ Y_s^{[n]} \end{bmatrix}, \quad f\big(Y^{[n]}\big) = \begin{bmatrix} f\big(Y_1^{[n]}\big) \\ f\big(Y_2^{[n]}\big) \\ \vdots \\ f\big(Y_s^{[n]}\big) \end{bmatrix}, \quad y^{[n-1]} = \begin{bmatrix} y_1^{[n-1]} \\ y_2^{[n-1]} \\ \vdots \\ y_r^{[n-1]} \end{bmatrix}, \quad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_r^{[n]} \end{bmatrix}.$$

If h denotes the stepsize, then the quantities imported into and evaluated in step number n are related by

$$\begin{aligned} Y^{[n]} &= h(A \otimes I_m) f\big(Y^{[n]}\big) + (U \otimes I_m) y^{[n-1]}, \\ y^{[n]} &= h(B \otimes I_m) f\big(Y^{[n]}\big) + (V \otimes I_m) y^{[n-1]}, \end{aligned}$$

where $n = 1, 2, \dots, N$ and m is the dimension of the system.

For ease of representation in the rest of this thesis, the Kronecker product with I_m will be removed. The method is then

$$\begin{aligned} Y^{[n]} &= hAf\big(Y^{[n]}\big) + Uy^{[n-1]}, \\ y^{[n]} &= hBf\big(Y^{[n]}\big) + Vy^{[n-1]}. \end{aligned} \tag{2.1.1}$$

Another abuse of notation which significantly simplifies many results is when a function, say $g(v)$ with a vector argument v, for example $f\big(y(x)\big)$, can be interpreted as having a vector of vectors argument where the function acts on each vector in a component by component sense. An example of this is $f\big(Y^{[n]}\big)$. Another common example, especially useful when considering Taylor series expansions, is $y(ex+\delta)$, where x is a scalar and e and δ are vectors. It is convenient to write the coefficients of the method, that is, the elements of A, B, U and V, as a partitioned $(s+r)\times(s+r)$ matrix

$$M = \left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right]. \tag{2.1.2}$$

This formulation of general linear methods was introduced by Burrage and Butcher [13] in 1980. The structure of the leading coefficient matrix A which is similar to that of the A matrix in Runge-Kutta methods, determines the implementation cost of these methods. General linear methods can borrow popularised names used in Runge-Kutta theory such as diagonally implicit and singly implicit.

The vector $y^{[n]}$ can have a very general structure. The traditional interpretations must be able to fit into this structure. That is to say the quantities could approximate the solution and the derivatives at various previous points, backward difference approximations to the derivatives or approximations to a Nordsieck vector, all of which are common choices in linear multistep methods. Likewise for Runge-Kutta methods, $y^{[n]}$ could be an approximation to y_n or a perturbation of y_n using the generalisation to effective order. It is also convenient to define the exact value function $z(x,h)$ which takes values in $\mathbb{R}^{r\times m}$. Each of the r components of $z(x,h)$ represents an m dimensional function which relates in some way to the exact solution $y(x)$. For example, the exact value function has a very simple form for a Runge-Kutta method with the traditional interpretation of order

$$z(x,h) = y(x).$$

Whereas for a linear r step method the exact value function is given by

$$z(x,h) = \begin{bmatrix} y(x) \\ y(x-h) \\ \vdots \\ y(x-(r-2)h) \\ y(x-(r-1)h) \end{bmatrix},$$

or, alternatively, the exact value function for a Nordsieck method is

$$z(x,h) = \begin{bmatrix} y(x) \\ hy'(x) \\ \frac{h^2}{2!}y''(x) \\ \vdots \\ \frac{h^r}{r!}y^{(r)}(x) \end{bmatrix}.$$

The computations performed in a step give a new approximation $y^{[n]}$ to $z(x_n, h)$ from an approximation $y^{[n-1]}$ to $z(x_{n-1}, h)$.

The stage values are approximations to the solution. That is to say $Y_i^{[n]} \approx y(x_{n-1} + c_i h)$, $i = 1, 2, \ldots, s$. These quantities are not used after the current step. Generally it is preferred that the stages approximate the solution within the current integration interval. This is equivalent to requiring that $0 \leq c_i \leq 1$. However, this is not always the case. When any of the abscissae are less than zero, a starting problem occurs; any of the abscissae greater than one makes it difficult to step to a point of discontinuity, if information is required past the discontinuity. The vector

$$c = \begin{bmatrix} c_1 & c_2 & \cdots & c_{s-1} & c_s \end{bmatrix}^T,$$

is called the vector of abscissae. Most multistage methods, in particular Runge-Kutta methods, are derived so that the internal stage approximations become more accurate as more approximations are computed.

2.2 Pre-consistency, consistency, stability and convergence

The pre-consistency condition is concerned with solving the trivial one dimensional differential equation $y'(x) = 0$, with solution $y(x) = 1$, exactly at both the beginning and end of each step. The pre-consistency vector u is determined by ensuring

$$\begin{aligned} y^{[n-1]} &= uy(x_{n-1}) + O(h), \\ y^{[n]} &= uy(x_n) + O(h). \end{aligned}$$

Using a general linear method to solve the problem $y'(x) = 0$, the internal stages and the output approximations are represented by

$$\begin{aligned} Y^{[n]} &= Uy^{[n-1]} \quad &\Longrightarrow& \quad e = Uu, \\ y^{[n]} &= Vy^{[n-1]} \quad &\Longrightarrow& \quad u = Vu. \end{aligned}$$

The pre-consistency conditions are therefore, $Vu = u$ and $Uu = e$.

The consistency condition is concerned with solving the one dimensional differential equation $y'(x) = 1$, given the initial condition $y(x_0) = 0$ which has an exact solution $y(x) = x - x_0$. The numerical solution should be exact at both the beginning and end of each step. The quantities passed from step to step represent approximations to the solution $y(x)$, the scaled derivative $hy'(x)$, or an arbitrary linear combination. The consistency vector v is determined by ensuring

$$\begin{aligned} y^{[n-1]} &= uy(x_{n-1}) + vhy'(x_{n-1}) + O(h^2), \\ y^{[n]} &= uy(x_n) + vhy'(x_n) + O(h^2). \end{aligned}$$

Using a general linear method to solve the problem $y'(x) = 1$, the internal stages and the output approximations are represented by

$$\begin{aligned} Y^{[n]} &= Aeh + Uy^{[n-1]} \quad &\Longrightarrow& \quad c = Ae + Uv, \\ y^{[n]} &= Beh + Vy^{[n-1]} \quad &\Longrightarrow& \quad u + v = Be + Vv. \end{aligned}$$

The consistency conditions are therefore $c = Ae + Uv$ and $Be + Vv = u + v$.

Just as for linear multistep methods a concept of stability is necessary. As a preliminary step, define what it means for a matrix to be stable.

Definition 2.1 *The matrix V is stable if there exists a constant k such that $\|V^n\|_\infty \le k$, for all $n = 1, 2, \dots$.*

Stability of a general linear method M, defined by (2.1.2), is guaranteed if the solution of the trivial differential equation, $y'(x) = 0$, is bounded. This has the effect of ensuring that errors introduced in a step do not dramatically affect later steps. For the trivial differential equation, expressing the output approximations in terms of the input approximations,

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

then shows the method is stable if V is a stable matrix. This leads to the following definition.

Definition 2.2 *A general linear method is said to be strictly stable if all the eigenvalues of V are inside the unit circle except one which is on the boundary.*

A general linear method is convergent if there exists a non-zero vector $u \in \mathbb{R}^r$ such that if $y^{[0]} = uy(x_0) + O(h)$, then $y^{[n]} = uy(x_0 + nh) + O(h)$ for all n, given that nh is bounded. The fundamental result by Dahlquist [59], that stability and consistency are necessary and sufficient conditions for convergence, was generalised for general linear methods by Butcher in [19].

2.3 Order of accuracy

A starting procedure is often required to obtain approximations to the initial vector, $y^{[0]}$, given only the initial condition $y(x_0)$. The starting procedure can be quite general. Define the internal stage values of the starting procedure as $\bar{Y}_1, \bar{Y}_2, \dots, \bar{Y}_{\bar{s}}$ and the derivatives at these internal points as $f(\bar{Y}_1), f(\bar{Y}_2), \dots, f(\bar{Y}_{\bar{s}})$. The starting procedure is given by

$$\begin{aligned}\bar{Y} &= hS_{11}f\big(\bar{Y}\big) + S_{12}y_0,\\ y^{[0]} &= hS_{21}f\big(\bar{Y}\big) + S_{22}y_0,\end{aligned}$$

where the vectors $\bar{Y}$ and $f\big(\bar{Y}\big)$ are given by

$$\bar{Y} = \begin{bmatrix} \bar{Y}_1 \\ \bar{Y}_2 \\ \vdots \\ \bar{Y}_{\bar{s}} \end{bmatrix}, \quad f\big(\bar{Y}\big) = \begin{bmatrix} f\big(\bar{Y}_1\big) \\ f\big(\bar{Y}_2\big) \\ \vdots \\ f\big(\bar{Y}_{\bar{s}}\big) \end{bmatrix}.$$

For pre-consistency $S_{22} = u$ and $S_{12} = e$. The starting procedure can be represented by a partitioned $(\bar{s} + r) \times (\bar{s} + 1)$ matrix

$$S = \left[\begin{array}{c|c} S_{11} & S_{12} \\ \hline S_{21} & S_{22} \end{array}\right],$$

where r denotes the number of approximations which need to be computed and $\bar{s}$ is the number of stages required to do so. The starting procedure could alternatively be thought of as r different generalised Runge-Kutta methods, with tableau

$$\begin{array}{c|c} c & A \\ \hline b_0 & b^T \end{array},$$

where $b_0 = 0$ is possible. The first generalised Runge-Kutta methods used as starting procedures were developed by Gear [71] in 1980. These were used as a means of starting the integration using a sufficiently high order multistep method.

The combined effect of obtaining the starting procedure S and completing a step of method M can be conveniently expressed as $M \circ S$. The exact solution shifted from x_{n-1} to x_n, is represented by the shift operator E. The order of accuracy of the method M can be defined relative to S as the value p for which $M \circ S - S \circ E = O(h^{p+1})$. This is shown in Figure 2.1.

$$\begin{tikzcd}
& y^{[1]} \arrow[d, dash, "O(h^{p+1})"] \\
y^{[0]} \arrow[ur, dash, "M"] \arrow[r, "E"'] & \bullet \\
y(x_0) \arrow[u, "S"] \arrow[r, "E"'] & y(x_1) \arrow[u, "S"]
\end{tikzcd}$$

Figure 2.1: Order of accuracy.

Assuming that a general linear method has order p, it is generally the case that each of the r components of $y^{[0]}$ is accurate to order p. That is $y^{[0]} = z(x_0, h) + O(h^{p+1})$. The methods derived in Chapter 3 also have this property. However, it is not necessary for all r components to be of order p, but at least the component carrying the information about the solution should be of order p. Almost Runge-Kutta methods are an example of this, where a three component Nordsieck vector is passed from step to step with the first two components at least of order p and the last a low order approximation to the scaled second derivative. The method must then be carefully constructed so as to annihilate this low order approximation.

The idea of local truncation error leads to an approximation of the global error, by repeating the approximation from Figure 2.1 as many times as the method takes steps.

$$\begin{tikzcd}
& y^{[1]} \arrow[r, dash, "M"] \arrow[d, dash] & y^{[2]} \arrow[r, dash, dashed] \arrow[d, dash] & y^{[N-2]} \arrow[r, dash, "M"] \arrow[d, dash] & y^{[N-1]} \arrow[r, dash, "M"] \arrow[d, dash] & y^{[N]} \arrow[d, dash] \\
y^{[0]} \arrow[ur, dash, "M"] \arrow[r, "E"'] & \bullet \arrow[r, "E"'] & \bullet \arrow[r, dash, dashed] & \bullet \arrow[r, "E"'] & \bullet \arrow[r, "E"'] & \bullet \\
y(x_0) \arrow[u, "S"] \arrow[r, "E"'] & y(x_1) \arrow[u, "S"] \arrow[r, "E"'] & y(x_2) \arrow[u, "S"] \arrow[r, dash, dashed] & y(x_{N-2}) \arrow[u, "S"] \arrow[r, "E"'] & y(x_{N-1}) \arrow[u, "S"] \arrow[r, "E"'] & y(x_N) \arrow[u, "S"]
\end{tikzcd}$$

Figure 2.2: Global error estimate.

A finishing procedure, F, is constructed to undo the work of the starting procedure S so that the final answer can be found correct to within $O(h^p)$. That is, apply F to $y^{[N]}$ which gives $y(x_N) + O(h^p)$ as long as Nh is bounded. This is shown in the Figure 2.3.

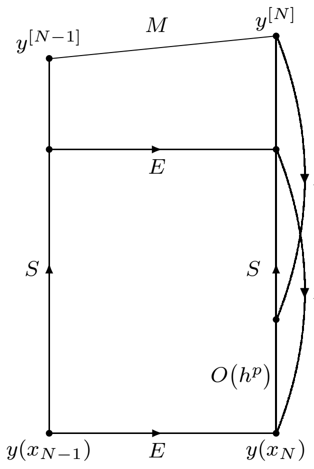


Figure 2.3: Finishing procedure.

Most practical general linear methods have one component of $y^{[n]}$ which directly approximates the solution. The finishing procedure in this case is simply to pick this information out of $y^{[n]}$. Again the methods derived in Chapter 3 will adopt this approach.

As for Runge-Kutta methods, the Taylor series expansions for general linear methods needed to analyse order become very complicated expressions as the order increases. Therefore, a framework is needed for simplifying these conditions. The approach used for general linear methods is the same as that for Runge-Kutta methods, where rooted trees and B-series are used to develop the order conditions.

2.4 Rooted trees

Before showing how the order conditions can be constructed, it is necessary to discuss rooted trees, as these allow a clear and concise means of representing the order conditions. Rooted trees are single rooted connected graphs with no cycles and are used in the analysis of order for general linear methods. Let T denote the set of all rooted trees including the empty set. For $t \in T$, the order of the tree, denoted $r(t)$, is the number of vertices in the tree. All trees subject to $r(t) > 1$ can be represented recursively by deleting the root of t and denoting the distinct subtrees left as $t_1, t_2, \ldots, t_m$. The relationship between t and $t_1, t_2, \ldots, t_m$, is denoted by $t = \left[t_1^{n_1}, t_2^{n_2}, \ldots, t_m^{n_m}\right]$, where $n_1, n_2, \ldots n_m$ denote the number of times the trees $t_1, t_2, \ldots, t_m$ respectively occur. This is known as grafting the trees $t_1, t_2, \ldots, t_m$ to a new root and is formalised in the following definition from [26].

Definition 2.3 *The set of rooted trees T is recursively defined by*

$$t = \begin{cases} \emptyset, & \text{if} \quad r(t) = 0, \\ \tau, & \text{if} \quad r(t) = 1, \\ \left[t_1^{n_1}, t_2^{n_2}, \ldots, t_m^{n_m}\right], & \text{if} \quad r(t) > 1, t_i \in T, i = 1, \ldots, m. \end{cases}$$

Now define some real-valued functions on the set of rooted trees T. The density $\gamma(t)$ is a measure of how "unbushy" the tree in question is. The symmetry $\sigma(t)$ is the order of the automorphism group of t. The following definition [26] shows how these functions are generated recursively.

Definition 2.4 *If* $t = \left[t_1^{n_1}, t_2^{n_2}, \ldots, t_m^{n_m}\right]$, *where* t_1, t_2, $\ldots$, t_m *are distinct trees, then*

$$\begin{aligned} r(t) &= 1 + n_1 r(t_1) + \cdots + n_m r(t_m), \\ \gamma(t) &= r(t)\gamma(t_1)^{n_1} \ldots \gamma(t_m)^{n_m}, \\ \sigma(t) &= n_1! n_2! \ldots n_m! \sigma(t_1)^{n_1} \ldots \sigma(t_m)^{n_m}, \end{aligned}$$

with

$$\begin{aligned} r(\emptyset) = \gamma(\emptyset) = \sigma(\emptyset) = 0, \\ r(\tau) = \gamma(\tau) = \sigma(\tau) = 1. \end{aligned}$$

It is also necessary to discuss sets of rooted trees. It most cases a set of rooted trees is called a forest. Forests generally occur when the root of a rooted tree is removed. That is, when the root is removed from a rooted tree $t = [t_1, t_2, \ldots, t_m]$, the forest $t_1, t_2, \ldots, t_m$ is left. Let F denote the set of all forests including the empty set. For ease of representation let

$$\mathrm{op}(t) = \mathrm{op}\big([t_1, t_2, \ldots, t_m]\big) = t_1, t_2, \ldots, t_m.$$

To generate rooted trees in a recursive way using an application such as Maple or Matlab it must be possible to construct the rooted trees in a compact way. The idea is to represent the trees as binary trees. This is done using the product of two rooted trees [26]. This product is often referred to as the Butcher product.

Definition 2.5 *The product* $t \cdot v \in T$ *of two rooted trees is defined as follows,*

$$[\emptyset] \cdot v = [v], \qquad [t_1, t_2, \ldots, t_m] \cdot v = [t_1, t_2, \ldots, t_m, v].$$

Moreover given $t, v_1, v_2 \in T$, the product satisfies

$$(t \cdot v_1) \cdot v_2 = (t \cdot v_2) \cdot v_1.$$

Given that a tree t cannot, in general, be decomposed as the product of two lower order trees uniquely, Murua [101] introduced the standard decomposition which can be determined if some ordering is decided upon. The order relation for T is recursively defined as follows:

Definition 2.6 *Given* $t_1, t_2 \in T$, *then* $t_1 < t_2$ *if one of the three conditions is fulfilled:*

- $r(t_1) < r(t_2)$,
- $r(t_1) = r(t_2)$ *and* $\mathrm{dec}_1(t_1) < \mathrm{dec}_1(t_2)$,
- $r(t_1) = r(t_2)$, $\mathrm{dec}_1(t_1) = \mathrm{dec}_1(t_2)$ *and* $\mathrm{dec}_2(t_1) < \mathrm{dec}_2(t_2)$.

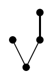
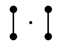
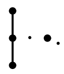
The standard decomposition is given in the following definition.

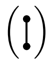
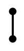
Definition 2.7 *The standard decomposition is defined by* $\mathrm{dec}(t) = \big(\mathrm{dec}_1(t), \mathrm{dec}_2(t)\big)$, *where the arguments satisfy*

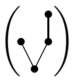
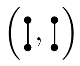
$$t = \mathrm{dec}_1(t) \cdot \mathrm{dec}_2(t),$$

and $\mathrm{dec}_2(t)$ *is maximal.*

That is, search among all pairs (t_1, t_2) such that $t = t_1 \cdot t_2$ to find a pair with maximal t_2. For example

 $=$  $=$ 

Since r  $> r(\bullet)$ it follows that  $> \bullet$ and the standard decomposition is

dec  $=$ 

Using this standard decomposition it is possible to also recursively derive $r(t)$, $\gamma(t)$ and $\sigma(t)$. First let T be represented by the set of integers. Now the order of the tree is simply

$$r(t) = r\big(\mathrm{dec}_1(t)\big) + r\big(\mathrm{dec}_2(t)\big),$$

with $r(1) = 1$, where the argument 1 represents the only tree of order one. The density $\gamma(t)$ is found using a special case of the composition rule described in Section 2.5. It is necessary to first compute the first derivative of $\gamma(t)$ denoted by $\gamma D_1(t)$ as

$$\gamma D_1(t) = \gamma D_1\big(\mathrm{dec}_1(t)\big)\gamma\big(\mathrm{dec}_2(t)\big),$$

see (2.5.9) for an explanation. The density $\gamma(t)$ is now given by

$$\gamma(t) = r(t)\gamma D_1(t).$$

This can be seen by noting that $\gamma D_1(t)$ is the product $\gamma(t_1)\gamma(t_2)\cdots\gamma(t_m)$ of all subtrees of t (see (2.5.8)). Finally the symmetry $\sigma(t)$ satisfies

$$\sigma(t) = \mathcal{K}\big(\mathrm{op}(t), \mathrm{dec}_2(t)\big)\sigma\big(\mathrm{dec}_1(t)\big)\sigma\big(\mathrm{dec}_2(t)\big),$$

where $\mathcal{K}\big(\mathrm{op}(t), \mathrm{dec}_2(t)\big)$ is the number of copies of $\mathrm{dec}_2(t)$ that are in the forest $\mathrm{op}(t)$. To obtain a recursive approach to $\mathcal{K}\big(\mathrm{op}(t), \mathrm{dec}_2(t)\big)$, note that if the second term in the decomposition of $\mathrm{dec}_1(t)$ is equal to $\mathrm{dec}_2(t)$ then $\mathrm{dec}_2(t)$ occurs at least twice. Therefore $\mathcal{K}\big(\mathrm{op}(t), \mathrm{dec}_2(t)\big)$ can be recursively generated as follows

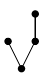
$$\mathcal{K}(t) = \begin{cases} \mathcal{K}\big(\mathrm{dec}_1(t)\big) + 1, & \text{if } \ \mathrm{dec}_2\big(\mathrm{dec}_1(t)\big) = \mathrm{dec}_2(t), \\ 1, & \text{if } \ \mathrm{dec}_2\big(\mathrm{dec}_1(t)\big) \neq \mathrm{dec}_2(t), \end{cases}$$

where $\mathcal{K}(1) = 1$ and $\sigma(1) = 1$.

The last real-valued function discussed is denote by $\epsilon(t)$, which is the number of ways of labelling t, subject to two conditions. The first condition is that each vertex is labelled only once. The second condition is, that if (i, j) is a labelled edge then $i < j$. It then follows that

$$\epsilon(t) = \frac{r(t)!}{\gamma(t)\sigma(t)}.$$

It is clear that labellings which are symmetrically equivalent are counted only once. The first tree for which $\epsilon(t) \neq 1$ is



which has three possible labellings.

The following algorithm is used to find $\mathrm{dec}_1(t)$, $\mathrm{dec}_2(t)$, $r(t)$, $\sigma(t)$, $\gamma(t)$ for all trees up to $p_{\max}$.

Algorithm 2.8

```
$t = 1$
first$(1) = 1$
$\gamma(1) = 1$
$\gamma'(1) = 1$
for $p = 2, \ldots, p_{\max}$
    % Index of 1st tree of order $p$
    first$(p) = t + 1$
    % Construct all the trees $t$ $\big(\mathrm{dec}(t) = (t_1, t_2)\big)$ of order $p$
    for $q = 1, \ldots, p - 1$
        % All the $t_1$ of order $q$
        for $t_1 = \mathrm{first}(q), \ldots, \mathrm{first}(q + 1) - 1$
            $l = \max\big(\mathrm{first}(p - q), \mathrm{dec}_2(t_1)\big)$
            % All the $t_2 \geq \mathrm{dec}_2(t_1)$ of order $p - q$
            for $t_2 = l, \ldots, \mathrm{first}(p - q + 1) - 1$
                $t = t + 1$
                $r(t) = p$
                $\mathrm{dec}_1(t) = t_1$
                $\mathrm{dec}_2(t) = t_2$
                $\gamma'(t) = \gamma'\big(\mathrm{dec}_1(t)\big)\gamma\big(\mathrm{dec}_2(t)\big)$
                $\gamma(t) = r(t)\gamma'(t)$
```

This algorithm is the basis for deriving the underlying one step method discussed in Section 2.6. In this case $\sigma(t)$ is not needed and is therefore not included in the Matlab code given in Appendix I. It is also possible to extend this algorithm to the case when the rooted trees have more than one type of node, see [101].

Table 2.1 gives the functions $r(t)$, $\gamma(t)$, $\sigma(t)$ and the standard decomposition for all trees up to order 5.

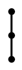
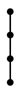
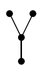
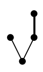
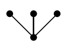
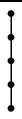
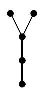
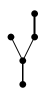
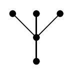
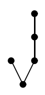
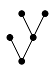
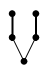
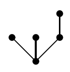
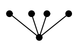
t																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$r(t)$	1	2	3	3	4	4	4	4	5	5	5	5	5	5	5	5	5
$\gamma(t)$	1	2	6	3	24	12	8	4	120	60	40	20	30	15	20	10	5
$\sigma(t)$	1	1	1	2	1	2	1	6	1	2	1	6	1	2	2	2	24
$\mathrm{dec}_1(t)$	-	1	1	2	1	1	2	4	1	1	1	1	2	2	3	4	8
$\mathrm{dec}_2(t)$	-	1	2	1	3	4	2	1	5	6	7	8	3	4	2	2	1

Table 2.1: Standard decomposition for trees of order up to five.

Let a_i denote the number of rooted trees with i vertices. The numbers $a_1, a_2, a_3, \dots$ satisfy the relation

$$a_1 + a_2 x + a_3 x^2 + \cdots = \prod_{n=1}^{\infty} (1 - x^n)^{-a_n}. \tag{2.4.1}$$

The following table gives the number of rooted trees for each order and the cumulative number of rooted trees for that order.

i	1	2	3	4	5	6	7	8	9	10
a_i	1	1	2	4	9	20	48	115	286	719
$\sum_{j=1}^{i} a_j$	1	2	4	8	17	37	85	200	486	1205

Table 2.2: Number of trees for orders one to ten.

2.5 Elementary differentials, B-series and composition rules

To determine the order of a general linear method, compare the Taylor series expansions of the exact and numerical solutions over a step of the method, assuming the incoming information is exact. The Taylor series expansions can be closely related to rooted trees. In this section several definitions will be given and they will be used in obtaining an algebraic analysis of order.

To compare Taylor series expansions of the exact and numerical solutions, $f\big(y(x)\big)$ must be differentiated many times using the chain rule. Therefore, any particular derivative of $f\big(y(x)\big)$ is made up of several known parts. Butcher [16] in 1963 named these parts as elementary differentials; they can be constructed recursively using the following definition.

Definition 2.9 *For any $t \in T$, the elementary differential, $F(t)$, for a function f is defined by*

$$F(t)\big(y(x)\big) = \begin{cases} y(x), & \text{if } \ t = \emptyset, \\ f\big(y(x)\big), & \text{if } \ t = \tau, \\ f^{(m)}\big(F(t_1), F(t_2), \dots, F(t_m)\big)\big(y(x)\big), & \text{if } \ t = [t_1, t_2, \dots, t_m]. \end{cases}$$

Immediately a connection between elementary differentials and rooted trees is evident. The following table shows this connection for all trees up to order four where f represents $f\big(y(x)\big)$, f' represents $f'\big(y(x)\big)$ and so on.

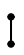
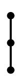
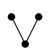
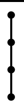
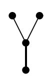
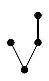
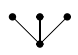
t								
$F(t)$	f	$f'f$	$f'f'f$	$f''(f,f)$	$f'f'f'f$	$f'f''(f,f)$	$f''(f,f'f)$	$f'''(f,f,f)$

Table 2.3: Connection between trees and elementary differentials.

The number of elementary differentials of any particular order is equal to the number of rooted trees with that order. The derivative of $y(x)$ of a particular order can be expressed as a unique linear combination of elementary differentials of that order. This is formalised in the following theorem.

Theorem 2.10 *If $y(x)$ is n times differentiable then*

$$y^{(n)}(x) = \sum_{r(t)=n} \epsilon(t)F(t)\big(y(x)\big).$$

The proof of this theorem can be found in [33]. In order to represent the Taylor series expansions, a formal series involving elementary differentials is needed.

Definition 2.11 *If $\alpha(t) : T \to \mathbb{R}^n$ is a vector valued function, define $B\big(\alpha(t), y(x)\big)$ by the formal series*

$$B\big(\alpha(t), y(x)\big) = \sum_{t\in T} \frac{\alpha(t)}{\sigma(t)} h^{r(t)} F(t)\big(y(x)\big),$$

which is interpreted in a component by component sense. This series is called a 'B-series' for the elementary weight function $\alpha(t)$ at $y(x)$ over a stepsize h.

The B-series was introduced by Hairer and Wanner [77] in 1974, and was named after Butcher for his contribution to the theory of Runge-Kutta methods. The scaling used above is the one used by Butcher.

The following basic rules are essential for manipulating B-series:

- There exists an elementary weight function $\mathbf{1}(t)$ which acts as the identity mapping. That is
$$B\big(\mathbf{1}(t), y_{n-1}\big) = y_{n-1}.$$
- For any matrix $M \in \mathbb{R}^{m\times n}$ and elementary weight function $\alpha(t) \in \mathbb{R}^n$, then
$$MB\big(\alpha(t), y_{n-1}\big) = B\big(M\alpha(t), y_{n-1}\big).$$
- The sum of two B-series with elementary weight functions $\alpha(t)$ and $\beta(t)$ at the same initial value y_{n-1} is a B-series with elementary weight function $(\alpha+\beta)(t)$; that is,
$$B\big(\alpha(t), y_{n-1}\big) + B\big(\beta(t), y_{n-1}\big) = B\big((\alpha+\beta)(t), y_{n-1}\big).$$
- Given an elementary weight function $\alpha(t)$, there exists an inverse elementary weight function $\alpha^{-1}(t)$, such that $\alpha\alpha^{-1}(t) = \mathbf{1}(t)$, or alternatively
$$y_n = B\big(\alpha(t), y_{n-1}\big) \qquad \Longleftrightarrow \qquad y_{n-1} = B\big(\alpha^{-1}(t), y_n\big).$$

Some special elementary weight functions are now defined. The elementary weight function $E(t)$ of the Picard integral, for any $t \in T$, means that $B\big(E(t), y_{n-1}\big)$ is the exact solution, where

$$E(t) = \frac{1}{\gamma(t)} e.$$

It is also convenient to use the reverse exact elementary weight function $E^{-1}(t)$, where for any $t \in T$,

$$E^{-1}(t) = \frac{1}{\gamma(t)} (-e)^{r(t)}.$$

For general linear methods with high stage order the exact solution is required at the abscissae points within the current step. The elementary weight function $C(t)$ for the exact solution of the stages is defined, for any $t \in T$, as

$$C(t) = \frac{1}{\gamma(t)} c^{r(t)}.$$

If the function $y(x)$ is sufficiently differentiable in a neighbourhood of x, then

$$\begin{aligned} B\big(C(t), y(x)\big) &= \sum_{t \in T} \frac{C(t)}{\sigma(t)} h^{r(t)} F(t)\big(y(x)\big) \\ &= \sum_{i=0}^{\infty} \frac{c^i h^i}{i!} \sum_{r(t)=i} \frac{i!}{\sigma(t)\gamma(t)} F(t)\big(y(x)\big) \\ &= \sum_{i=0}^{\infty} \frac{c^i h^i}{i!} y^{(i)}(x) \\ &= y(ex + ch), \end{aligned}$$

interpreted in a component by component sense. Thus $E(t)$ is a special case of $C(t)$. Therefore, the exact solution $y(x+h)$ is a B-series. The derivative i operator, $D_i(t)$, is important in deriving recursively the order conditions for general linear methods and also for higher order differential systems. For any $t \in T$, let

$$D_i(t) = \begin{cases} \dfrac{r(t)!}{\gamma(t)}, & \text{if } \ r(t) = i, \\ 0, & \text{if } \ r(t) \neq i. \end{cases}$$

If the function $y(x)$ is sufficiently differentiable in a neighbourhood of x, then

$$\begin{aligned} B\big(D_i(t), y(x)\big) &= \sum_{r(t)=i} \frac{D_i(t)}{\sigma(t)} h^i F(t)\big(y(x)\big) \\ &= \sum_{r(t)=i} \frac{r(t)!}{\sigma(t)\gamma(t)} h^i F(t)\big(y(x)\big) \\ &= \sum_{r(t)=i} \epsilon(t) h^i F(t)\big(y(x)\big) \\ &= h^i y^{(i)}(x). \end{aligned} \tag{2.5.1}$$

Let the elementary weight function for the stages $Y^{[n]}$ be $\xi(t)$. This means that $Y^{[n]}$ can be written as the B-series for elementary weight function $\xi(t)$ at y_{n-1} with stepsize h as follows

$$Y^{[n]} = B\big(\xi(t), y_{n-1}\big). \tag{2.5.2}$$

The starting procedure is also a B-series with elementary weight function $S(t)$ which takes the initial condition $y(x_0)$ to $y^{[0]}$, that is

$$y^{[0]} = B\big(S(t), y(x_0)\big). \tag{2.5.3}$$

It is now important to consider the composition of two B-series. This is used when the output of one B-series is used as the input to another B-series. The following definition was given in [77].

Definition 2.12 *Let $\alpha(t) : T \to \mathbb{R}^N$ and $\beta(t) : T \to \mathbb{R}^N$ be two elementary weight functions such that $\alpha(\emptyset) = e$. Then the composition of the two corresponding B-series is a B-series*

$$B\big(\beta(t), B\big(\alpha(t), y(x)\big)\big) = B\big(\alpha\beta(t), y(x)\big). \tag{2.5.4}$$

The product $\alpha\beta(t) : T \to \mathbb{R}^N$ is given by the mapping

$$\alpha\beta(t) = \beta(\emptyset)\alpha(t) + \beta(t) + \sum_{u \prec t} \beta(u)\alpha(t\backslash u), \qquad \forall t \in T. \tag{2.5.5}$$

Here $u \prec t$ denotes any proper subtree u, sharing the same root with t, and $t\backslash u$ denotes the remainder of the tree t after deleting the subtree u from it. Represent the product as

$$\alpha\beta(t) = \beta(\emptyset)\alpha(t) + \beta(t) + R(t), \tag{2.5.6}$$

where $R(t)$ denotes the remainder which is made up of trees of order less than t. The formula (2.5.5) was first published by Butcher in [22]. For a complete description of the $\alpha\beta(t)$ mapping see, for example, [26].

In the following table the product $\alpha\beta(t)$ for all trees of order less than or equal to four is given.

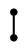
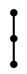
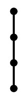
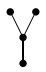
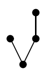
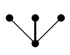
t		$\alpha\beta(t)$
$\emptyset$	t_0	$\beta(t_0)$
	t_1	$\alpha(t_1)\beta(t_0) + \beta(t_1)$
	t_2	$\alpha(t_2)\beta(t_0) + \alpha(t_1)\beta(t_1) + \beta(t_2)$
	t_3	$\alpha(t_3)\beta(t_0) + \alpha(t_2)\beta(t_1) + \alpha(t_1)\beta(t_2) + \beta(t_3)$
	t_4	$\alpha(t_4)\beta(t_0) + \alpha(t_1)^2\beta(t_1) + 2\alpha(t_1)\beta(t_2) + \beta(t_4)$
	t_5	$\alpha(t_5)\beta(t_0) + \alpha(t_3)\beta(t_1) + \alpha(t_2)\beta(t_2) + \alpha(t_1)\beta(t_3) + \beta(t_5)$
	t_6	$\alpha(t_6)\beta(t_0) + \alpha(t_4)\beta(t_1) + \alpha(t_1)^2\beta(t_2) + 2\alpha(t_1)\beta(t_3) + \beta(t_6)$
	t_7	$\alpha(t_7)\beta(t_0) + \alpha(t_1)\alpha(t_2)\beta(t_1) + \big(\alpha(t_1)^2 + \alpha(t_2)\big)\beta(t_2) + \alpha(t_1)\big(\beta(t_3) + \beta(t_4)\big) + \beta(t_7)$
	t_8	$\alpha(t_8)\beta(t_0) + \alpha(t_1)^3\beta(t_1) + 3\alpha(t_1)^2\beta(t_2) + 3\alpha(t_1)\beta(t_4) + \beta(t_8)$

Table 2.4: Composition of elementary weight functions.

As the order increases the complexity of $\alpha\beta(t)$ dramatically increases and to evaluate $\alpha\beta(t)$ in an environment such as Matlab or Maple it is essential to provide a recursive way to compute the product $\alpha\beta(t)$. Before doing so it is useful to introduce some notation. Let

$$G = \big\{\alpha \,|\, \alpha : T \to \mathbb{R},\ \alpha \text{ is a linear functional}\big\}.$$

A special case of G used for composition of methods, is defined as

$$G_1 = \big\{\alpha \,|\, \alpha \in G, \text{ with } \alpha(\emptyset) = 1\big\}.$$

The composition of two B-series is such that $G_1 \times G \to G$. To obtain the product recursively it is necessary to define the dual set of G, known as $\widehat{G}$, as follows

$$\widehat{G} = \big\{ \widehat{t} : G \to \mathbb{R} \,|\, \widehat{t}(\zeta) = \zeta(t), \text{ for all } \zeta \in G \big\}.$$

Denote the set of dual rooted trees $\widehat{t}$ by $\widehat{T}$. The product notation can also be extended to the dual tree case, that is, $\widehat{T} \times \widehat{T} \to \widehat{T}$ by

$$\widehat{t} \cdot \widehat{u} = \widehat{tu}.$$

Using the dual set of G the following theorem (see [26]) makes it possible to generate $\alpha\beta(t)$ in a recursive way.

Theorem 2.13 *If $\alpha(t) \in G_1$ and $t \in T$ then there exists a member $\lambda(\alpha, t)$ of G such that, for $\beta(t) \in G$,*

$$\alpha\beta(t) = \lambda(\alpha, t)(\beta) + \alpha(t)\beta(\emptyset).$$

Furthermore, $\lambda : G_1 \times T \to G$ satisfies the following conditions for any $\alpha(t) \in G_1$:

$$\lambda(\alpha, t) = \begin{cases} 0, & \text{if } \ t = \emptyset, \\ \widehat{\tau}, & \text{if } \ t = \tau \\ \lambda(\alpha, u)\lambda(\alpha, v) + \alpha(v)\lambda(\alpha, u), & \text{if } \ t = uv. \end{cases}$$

Note that Chan [55], has constructed a recursive definition of $\lambda(\alpha, t)$ based on the alternative representation of the rooted trees.

Consider now an application of the composition formulae. The composition formulae is crucial in determining the B-series of the scaled stage derivatives. First use the B-series for the scaled derivatives (2.5.1) with $i = 1$, then use the B-series for the internal stages (2.5.2), and finally use the composition formulae (2.5.4) to put it together as follows

$$\begin{aligned} hf\big(Y^{[n]}\big) &= B\big(D_1(t), Y^{[n]}\big) \\ &= B\big(D_1(t), B\big(\xi(t), y_{n-1}\big)\big) \\ &= B\big(\xi D_1(t), y_{n-1}\big). \end{aligned} \tag{2.5.7}$$

For special types of general linear methods the composite function

$$\alpha D_i(t) = D_i(t) + \sum_{u \prec t} D_i(u)\alpha(t \backslash u),$$

is often used. This can be decomposed into three parts.

- If $r(t) < i$, then $D_i(t) = 0$, for all trees $u \prec t$, $D_i(u) = 0$.
- If $r(t) = i$, then $D_i(t) = r(t)!/\gamma(t)$, but for all trees $u \prec t$, $D_i(u) = 0$.
- If $r(t) > i$, then $D_i(t) = 0$, but some non-empty subtrees u exist when $r(u) = i$. Note that

$$D_i(u)\alpha(t \backslash u) = \frac{i!}{\gamma(u)}\alpha(t \backslash u).$$

The composition $\alpha D_i(t)$ is now given by

$$\alpha D_i(t) = \begin{cases} 0, & \text{if } \; r(t) < i, \\ \dfrac{r(t)!}{\gamma(t)}, & \text{if } \; r(t) = i, \\ \displaystyle\sum_{\substack{r(u)=i \\ u \prec t}} \frac{i!}{\gamma(u)} \alpha(t \backslash u), & \text{if } \; r(t) > i. \end{cases}$$

The most common situation is when the first derivative operator is used. In this case, the only subtree u which is nonzero is τ, which leads to

$$\begin{aligned} \alpha D_1(t) &= \alpha D_1\big([t_1\, t_2\, \dots\, t_m]\big) \\ &= \alpha(t_1)\alpha(t_2)\dots\alpha(t_m). \end{aligned} \tag{2.5.8}$$

Alternatively, using the product and the standard decomposition, then

$$\begin{aligned} \alpha D_1(t) &= \alpha D_1\big(\mathrm{dec}_1(t), \mathrm{dec}_2(t)\big) \\ &= \alpha D_1\big(\mathrm{dec}_1(t)\big)\alpha\big(\mathrm{dec}_2(t)\big). \end{aligned} \tag{2.5.9}$$

These can be seen as equivalent by choosing the maximal t_i, $i = 1, 2, \dots, m$, say $t_m = \mathrm{dec}_2(t)$, and grafting the rest to the root, $[\alpha(t_1)\,\alpha(t_2)\,\dots\,\alpha(t_{m-1})] = \mathrm{dec}_1(t)$.

The elementary weight function $E(t)$ for the Picard integral satisfies the following

$$\begin{aligned} E(t) &= \frac{1}{r(t)\gamma(t_1)\cdots\gamma(t_m)} \\ &= \frac{1}{r(t)} E(t_1)E(t_2)\cdots E(t_m) \\ &= \frac{1}{r(t)} ED_1\big([t_1, t_2, \dots, t_m]\big), \end{aligned}$$

and hence the first derivative formula is

$$ED_1(t) = r(t)E(t). \tag{2.5.10}$$

The natural question is whether one can obtain a formula for higher derivatives in terms of only $E(t)$. This leads to the following theorem.

Theorem 2.14 *For each $i \in \mathbb{N}$, the derivative number i of the elementary weight function $E(t)$, for any $t \in T$, is given by*

$$ED_i(t) = \frac{r(t)!}{(r(t)-i)!} E(t). \tag{2.5.11}$$

Proof: The exact solution at θh through the step is

$$y(x + \theta h) = y(x) + \sum_{t \in T} \frac{E(t)\theta^{r(t)}}{\sigma(t)} F(t)\big(y(x)\big)h^{r(t)}.$$

Differentiating both sides with respect to θ, gives

$$hy'(x + \theta h) = \sum_{t \in T} \frac{E(t)r(t)\theta^{r(t)-1}}{\sigma(t)} F(t)\big(y(x)\big)h^{r(t)}.$$

Comparing the elementary weight function when $\theta = 1$ with the elementary weight function of (2.5.1), yields

$$ED_1(t) = r(t)E(t).$$

Suppose now that

$$h^k y^{(k)}(x+\theta h) = \sum_{t\in T} \frac{E(t)r(t)\big(r(t)-1\big)\cdots\big(r(t)-k+1\big)\theta^{r(t)-k}}{\sigma(t)} F(t)\big(y(x)\big)h^{r(t)}. \tag{2.5.12}$$

Again comparing the elementary weight function when $\theta = 1$ with the elementary weight function of (2.5.1), yields

$$ED_k(t) = \frac{r(t)!}{\big(r(t)-k\big)!}E(t).$$

Assume this has been proved for $i = 2, 3, \dots, k$ so that, to complete the proof by induction, it remains to prove the result for $i = k+1$. To do so, first differentiate (2.5.12) with respect to θ, which gives

$$h^{k+1} y^{(k+1)}(x+\theta h) = \sum_{t\in T} \frac{E(t)r(t)\big(r(t)-1\big)\cdots\big(r(t)-k\big)\theta^{r(t)-(k+1)}}{\sigma(t)} F(t)\big(y(x)\big)h^{r(t)}.$$

The result follows by comparing the elementary weight function with $\theta = 1$ with the elementary weight function of (2.5.1). □

Return attention to B-series and, in particular to representing the method (2.1.1) in terms of B-series. To represent the computations of the internal stages, that is, the first equation of (2.1.1) as a B-series, use the expressions (2.5.2), (2.5.7) and (2.5.3), and simplify using the basic rules in the following way

$$\begin{aligned} B\big(\xi(t), y_{n-1}\big) &= AB\big(\xi D_1(t), y_{n-1}\big) + UB\big(S(t), y_{n-1}\big),\\ &= B\big(A\xi D_1(t) + US(t), y_{n-1}\big). \end{aligned}$$

The output approximations, that is, the second equation of (2.1.1) can also be represented as a B-series. Again use the expressions (2.5.7) and (2.5.3), and simplify using the basic rules, as follows

$$\begin{aligned} B\big(\alpha(t), y_{n-1}\big) &= BB\big(\xi D_1(t), y_{n-1}\big) + VB\big(S(t), y_{n-1}\big)\\ &= B\big(B\xi D_1(t) + VS(t), y_{n-1}\big). \end{aligned}$$

Translating in terms of trees, gives the following generating functions for a general linear method

$$\xi(t) = A\xi D_1(t) + US(t), \tag{2.5.13}$$

$$\alpha(t) = B\xi D_1(t) + VS(t). \tag{2.5.14}$$

Since (2.5.14) has one component, if the method being constructed is a Runge-Kutta method, to obtain order p, (2.5.14) must be satisfied for all trees up to order p. Alternatively, if an r-step linear multistep method is being constructed of order p, then all r components of (2.5.14) must be satisfied for all trees up to order p. However, it is possible to construct general linear methods of order p where not all r components of (2.5.14) are satisfied for all trees up to order p. An example is the Almost Runge-Kutta methods. In this case it is necessary that the component of (2.5.14) which carries the information about the solution has order p. The methods which are mainly discussed in this thesis require all components of both (2.5.13) and (2.5.14) to agree to order p.

The generating function for the outgoing approximations (2.5.14) can be equivalently expressed as

$$ES(t) = B\xi D_1(t) + VS(t),$$

for all trees of order less than or equal to p.

When $t = \emptyset$, then (2.5.13) and (2.5.14) become

$$\begin{aligned} \xi(\emptyset) &= US(\emptyset), \\ S(\emptyset) &= VS(\emptyset). \end{aligned}$$

These are just the pre-consistency conditions of Section 2.2, that is, $\xi(\emptyset) = e$ and $S(\emptyset) = u$.

2.6 Underlying one step method

In 1986 Kirchgraber [91], stated that every strictly stable linear multistep method is essentially equivalent to a one step method. This result was generalised by Stoffer [112] in 1993 to general linear methods.

First assume that the general linear method is strictly stable from Definition 2.2. This means that V has 1 as a simple eigenvalue and the remaining $r-1$ eigenvalues have moduli less than one. Thus there is a transformation matrix T such that

$$T^{-1}MT = \left[\begin{array}{c|c} A & UT \\ \hline T^{-1}B & T^{-1}VT \end{array}\right],$$

where it can be assumed, without loss of generality, that

$$T^{-1}VT = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \bar{V} \end{array}\right],$$

with spectral radius satisfying $\rho\big(\bar{V}\big) < 1$. Note that most methods of practical interest satisfy this condition. In this case the data passed from step to step has be transformed to

$$z^{[n]} = T^{-1}y^{[n]},$$

where the first component of $z^{[n]}$ directly approximates the solution $y(x_n)$. The main result in [112] is the following theorem.

Theorem 2.15 *Let M be a strictly stable general linear method of order p. Then there exists a starting method S^* and a one step method ϕ, such that*

- *M is of order p relative to S^*.*
- *The one step method is of order p.*
- *The one step method ϕ is conjugate to M.*

The first item implies that an alternative starting procedure S^* can be found so that M is of order p. Note that if $z(x_0, h) = y^{[0]} + O(h^{p+1})$ is satisfied then S and S^* are equivalent to order p. This means equality between the elementary weight functions,

$$S(t) = S^*(t),$$

for all trees of order less than or equal to p.

This is generally the case for several reasons: It makes deriving the method M easier as the particular starting procedure is not an issue, except that it exists and commutes with $y^{[0]}$ to within $O(h^{p+1})$. As the cost of the starting procedure is a one-off, the cost is small relative to the whole computation. In fact, $z(x_0, h) = y^{[0]} + O(h^{p+1})$ can be considerably relaxed by assuming that the first component of $y^{[0]}$ is accurate to within $O(h^{p+1})$.

The second item guarantees the one step method has the same order as the general linear method only when the stepsizes used are fixed. It is also the case for variable stepsizes, if all the output approximations have the same order. It is not the case in general if all the output approximations do not have the same order. The Almost Runge-Kutta methods are an example of this. The order can be restored when variable stepsizes are used only if special care is taken in the implementation of the method, see [47]. Moreover, the behaviour of the general linear method is exponentially attracted to the dynamics of the flow, and therefore the approximations in $\mathbb{R}^{m\times r}$ are exponentially attracted to $\mathbb{R}^m$. This is because the general linear method is strictly stable.

The last item implies that the diagram shown in Figure 2.4 commutes:

$$\begin{tikzcd}
\mathbb{R}^{m\times r} \arrow[r, "M"] & \mathbb{R}^{m\times r} \\
\mathbb{R}^{m} \arrow[u, "S^*"] \arrow[r, "\phi"'] & \mathbb{R}^{m} \arrow[u, "S^*"']
\end{tikzcd}$$

Figure 2.4: The general linear method and the underlying one step method commute.

The one step method ϕ is known as the underlying one step method. The existence of this method is interesting. In general it is not possible to construct the actual method but it is possible to determine the values of the elementary differentials for all orders. To understand more about the underlying one step method and the starting procedure S^*, the approach used by Chartier [56] is followed. The commutative diagram represented in Figure 2.4 is expressed in terms of B-series, that is

$$\begin{aligned}
B\big(\xi(t), y\big) &= AB\big(\xi D_1(t), y\big) + UB\big(S^*(t), y\big), \\
B\big(S^*(t), B\big(\phi(t), y\big)\big) &= BB\big(\xi D_1(t), y\big) + VB\big(S^*(t), y\big).
\end{aligned}$$

Translated in terms of trees, this gives the generating functions

$$\xi(t) = A\xi D_1(t) + US^*(t), \tag{2.6.1}$$

$$\phi S^*(t) = B\xi D_1(t) + VS^*(t). \tag{2.6.2}$$

When $t = \emptyset$, then (2.6.1) and (2.6.2) become

$$\begin{aligned}
\xi(\emptyset) &= US^*(\emptyset) = e, \\
S^*(\emptyset) &= VS^*(\emptyset).
\end{aligned}$$

These are again just the pre-consistency conditions; that is $S^*(\emptyset) = u$. The second equation implies that V has an eigenvalue of one with corresponding eigenvector $S^*(\emptyset)$. Now, using the strict stability assumption, V can be decomposed as follows

$$V = S^*(\emptyset)v^T + \widetilde{V},$$

where v^T is the left eigenvector and $S^*(\emptyset)$ is the right eigenvector of V. Therefore

$$\begin{aligned} v^T V &= v^T S^*(\emptyset) v^T + v^T \widetilde{V} \\ &= v^T, \end{aligned}$$

which implies that $v^T S^*(\emptyset) = 1$ and $v^T \widetilde{V} = 0^T$, and

$$\begin{aligned} V S^*(\emptyset) &= S^*(\emptyset) v^T S^*(\emptyset) + \widetilde{V} S^*(\emptyset) \\ &= S^*(\emptyset), \end{aligned}$$

which implies that $v^T S^*(\emptyset) = 1$ and $\widetilde{V} S^*(\emptyset) = 0$. Note that strict stability implies that $\rho\big(\widetilde{V}\big) < 1$. It is now possible to find recursive expressions for $\phi(t)$ and $S^*(t)$. First, recall from (2.5.6) that

$$\phi S^*(t) = S^*(t) + \phi(t) S^*(\emptyset) + R(t), \tag{2.6.3}$$

where $R(t)$ is an expression involving only trees of order less than the order of t. Substituting (2.6.3) into (2.6.2) gives

$$\begin{aligned} S^*(t) + \phi(t) S^*(\emptyset) - V S^*(t) &= B\xi D_1(t) - R(t), \\ \big(I - \widetilde{V}\big) S^*(t) + \big(\phi(t) - v^T S^*(t)\big) S^*(\emptyset) &= B\xi D_1(t) - R(t). \end{aligned} \tag{2.6.4}$$

Now pre-multiplying (2.6.4) by v^T gives

$$v^T \big(I - \widetilde{V}\big) S^*(t) + v^T \big(\phi(t) - v^T S^*(t)\big) S^*(\emptyset) = v^T \big(B\xi D_1(t) - R(t)\big).$$

Since $v^T \widetilde{V} = 0^T$ and $v^T S^*(\emptyset) = 1$, it follows that

$$\phi(t) = v^T \big(B\xi D_1(t) - R(t)\big). \tag{2.6.5}$$

Rearranging (2.6.4) yields

$$\begin{aligned} S^*(t) &= \big(I - \widetilde{V}\big)^{-1} \big(B\xi D_1(t) - R(t)\big) - \big(\phi(t) - v^T S^*(t)\big) \big(I - \widetilde{V}\big)^{-1} S^*(\emptyset) \\ &= \big(I - \widetilde{V}\big)^{-1} \big(B\xi D_1(t) - R(t)\big) - \big(\phi(t) - v^T S^*(t)\big) S^*(\emptyset). \end{aligned} \tag{2.6.6}$$

Apart from when $t = \emptyset$ and $v^T S^*(\emptyset) = 1$, $v^T S^*(t)$ can be chosen while constructing the starting procedure. This means that the starting method S^* is not uniquely defined. Therefore, $\phi(t)$ which depends on $\xi(t)$, which in turn depends on $S^*(t)$, is not unique. In [112], the following questions were asked. Given that the starting method S^* is not unique, is there an optimal choice of S^*?; is there a simple way to find S^* such that the associated one step method ϕ has minimal local error coefficients in some reasonable sense?

It is often the case that one of the components of $y^{[n]}$, say $y_1^{[n]}$, directly approximates the solution. The linear multistep, predictor-corrector and Nordsieck methods are common examples. In this case it seems natural to choose $y_1^{[0]} = y(x_0)$. Here the starting method S^* and thus the underlying one step method is uniquely defined. Without loss of generality, the method can be transformed so that the pre-consistency vector $S^*(\emptyset) = e_1$. When this is the case the last term of the right hand side of (2.6.6) only affects the first component in the starting procedure. If it is required that $y_1^{[0]} = y(x_0)$ then the first component of the starting procedure is therefore defined as

$$S_1^*(t) = \mathbf{1}(t), \tag{2.6.7}$$

as this assumes the initial condition is exact. The last $r - 1$ components of

$$S^*(t) = \big(I - \widetilde{V}\big)^{-1} \big(B\xi D_1(t) - R(t)\big), \tag{2.6.8}$$

gives the rest of the starting procedure.

In [112], methods with this structure are called "essentially one step methods". The reason for this is

$$y_1^{*[n]} = \phi\big(h, y_1^{*[n-1]}\big),$$

for all $n \geq 1$. The methods developed in this thesis also have the first component of $y^{[n]}$ directly approximating the solution. The starting method S^* is always constructed so that (2.6.7) is satisfied. In Appendix I, Maple and Matlab programs are given which compute $\phi(t)$, $S^*(t)$ and $\xi(t)$ from (2.6.5), (2.6.8) and (2.6.1) respectively. These programs are based on Algorithm 2.8 which also calculates the local error of the underlying one step method.

2.7 Linear and non-linear stability

Several types of stability are considered using successively more general test problems. The discussion is split into two. First the stability for stiff problems and secondly for non-stiff problems are discussed.

The basic linear stability behaviour of general linear methods is considered using the standard linear test problem of Dahlquist [60],

$$y'(x) = qy(x),$$

where q is possibly complex. If the method (2.1.1) is applied to the test problem then the internal stages are given by

$$\begin{aligned} Y^{[n]} &= zAY^{[n]} + Uy^{[n-1]} \\ &= (I - zA)^{-1}Uy^{[n-1]}, \end{aligned}$$

where $(I - zA)$ is nonsingular and $z = hq$. The outgoing approximations are given by

$$\begin{aligned} y^{[n]} &= zBY^{[n]} + Vy^{[n-1]} \\ &= \big(V + zB(I - zA)^{-1}U\big)y^{[n-1]}. \end{aligned}$$

Let $M(z)$ be defined by

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

and called the stability matrix. When a general linear method is used with a sufficiently large stepsize, the numerical solution may become unstable, that is become unbounded, even though the exact solution is bounded or decays to a finite value. For certain methods applied to stiff problems, the stepsize necessary for stability may be excessively small in relation to the smoothness of the exact solution. This means stability rather than accuracy is restricting the stepsize. To ensure there is no restriction on the stepsize due to stability the following condition is desired.

Definition 2.16 *A general linear method is A-stable if for all $z \in \mathbb{C}^-$, $I - zA$ is non-singular and $M(z)$ is a stable matrix.*

For methods with this property the stepsize is never restricted by stability on linear constant coefficient problems, regardless of the stiffness. In some situations, it may be desirable to damp the very stiff components of the numerical solution. This leads to the concept of L-stability.

Definition 2.17 *A general linear method is L-stable if it is A-stable and*

$$\rho\big(M(\infty)\big) = 0,$$

or the stronger condition

$$M(\infty) = 0.$$

Most L-stable general linear methods satisfy the $\rho\big(M(\infty)\big) = 0$ condition as the stronger condition is more restrictive and often leaves the methods without much freedom.

The first generalisation of the standard test problem is given by the non-autonomous problem

$$y'(x) = q(x)y(x),$$

where $q(x)$ is possibly complex. If the method (2.1.1) is applied to this problem then the internal stages are given by

$$Y^{[n]} = (I - AZ)^{-1}Uy^{[n-1]},$$

where $(I - AZ)$ is nonsingular and $Z = \operatorname{diag}(z_1, z_2, \ldots, z_s)$ with $z_i = hq(x_{n-1} + c_i h)$. The outgoing approximations are given by

$$y^{[n]} = \big(V + BZ(I - AZ)^{-1}U\big)y^{[n-1]}.$$

The stability matrix, $M(Z)$, is given by

$$M(Z) = V + BZ(I - AZ)^{-1}U.$$

Since the test problem has been generalised, the types of stability need also to be generalised. The first is as follows.

Definition 2.18 *A general linear method is weakly AN-stable if for all $z_1, z_2, \ldots, z_s \in \mathbb{C}^-$, $I - AZ$ is non-singular and $M(Z)$ is a stable matrix.*

Weak AN-stability is not enough to guarantee stable behaviour for the second test problem, because different values of Z may occur at each step. Instead, the product of matrices like $M(Z)$ should be bounded and this leads to the following definition.

Definition 2.19 *A general linear method is strongly AN-stable if there exists a norm $\|\cdot\|$ such that $\|M(Z)\| \le 1$ for all $z_1, z_2, \ldots, z_s \in \mathbb{C}^-$, and $I - AZ$ is non-singular.*

Strong AN-stability is further strengthened by requiring the norm $\|\cdot\|$ to be an inner product norm, this property being known as 'Euclidean AN-stability'.

Consider, finally, the nonlinear problem

$$y'(x) = f\big(y(x)\big),$$

where the inner product

$$\big\langle f(u), u\big\rangle \le 0.$$

This implies that $\|y(x)\|$ is non-increasing because

$$\frac{d}{dx}\|y(x)\|^2 = 2\big\langle y'(x), y(x)\big\rangle = 2\big\langle f(y(x)), y(x)\big\rangle \le 0.$$

Stable behaviour is guaranteed for this problem if there exists a positive definite $r \times r$ matrix G and a positive diagonal $s \times s$ matrix D, such that

$$H = \begin{bmatrix} G - V^T G V & U^T D - V^T G B \\ DU - B^T G V & DA + A^T D - B^T G B \end{bmatrix}$$

is non-negative definite. This property was introduced by Burrage and Butcher [13], and is known as algebraic stability. The matrix H plays a very important role in the search for methods suitable for Hamiltonian problems. In the case of Runge-Kutta methods, $H = 0$ is required if the method is to be symplectic [108], and therefore mimic the dynamics of the flow. It has also been conjectured that if a general linear method is to be symplectic then $H = 0$, see [65].

There are several connections between the various types of stability defined above. These are as follows:

$$\begin{tikzcd}
\text{algebraic stability} \arrow[d, Rightarrow, shift right] \\
\text{Euclidean } AN\text{-stability} \arrow[u, Rightarrow, shift right] \arrow[d, Rightarrow] \\
\text{strong } AN\text{-stability} \arrow[d, Rightarrow, shift right] \\
\text{weak } AN\text{-stability} \arrow[u, Rightarrow, shift right, "/" marking] \arrow[d, Rightarrow, shift right] \\
A\text{-stability} \arrow[u, Rightarrow, shift right, "/" marking]
\end{tikzcd}$$

It is not known whether or not strong AN-stability implies Euclidean AN-stability. The proofs and counterexamples leading to these results can be found in [13] or [25].

Consider now the stability of explicit methods. It is well known that an explicit method cannot be A-stable, and therefore cannot possess any of the other types of stability properties mentioned above. Stability issues are therefore restricted to the first test problem. An important indication of the suitability of a method is the size of the stability region,

$$R = \{ z \in \mathbb{C}, \exists C : \|M(z)\|^n \le C, \forall n \ge 1 \}.$$

An equivalent statement is as follows: all eigenvalues w of $M(z)$ with multiplicity one satisfy $|w| \le 1$, and all eigenvalues w of $M(z)$ with multiplicity greater than one, satisfy $|w| < 1$. The characteristic polynomial is defined as

$$\begin{aligned} p(w,z) &= \det\big(wI - M(z)\big) \\ &= p_0(z)w^r + p_1(z)w^{r-1} + \cdots + p_r(z), \end{aligned} \tag{2.7.1}$$

where the complex functions $p_0(z)$, $p_1(z)$, …, $p_r(z)$ are of degree at most s. Therefore, a point z of the complex plane belongs to R if and only if $p(w,z)$ has all its roots within the unit circle except for the roots on the unit circle which are distinct.

To find the stability region of an arbitrary general linear method first find $p(w,z) = 0$, then solve this equation as a function of w, therefore, $z = f(w)$. Consider values of w on the unit circle as this is equivalent to a root of $p(w,z)$ lying on the unit circle. For $w = \exp(i\theta)$ with $\theta \in [0, 2\pi]$, the resulting z values give the boundary of the stability region. This procedure is known as the 'boundary locus method'.

If two methods being compared are both explicit and have the same number of function evaluations per step, then their stability domains cannot be included one in the other [90]. The interesting interpretation of this result is that for any two methods, there exists a differential equation $y'(x) = qy(x)$ such that one of them performs better than the other. No "miracle" method is possible [78]. Even though it is not possible to obtain a miracle method, it is however possible to find methods with relatively large stability regions, but identifying where such methods exist is extremely complicated in this general setting. In comparing the relative sizes of the stability regions of Runge-Kutta and linear multistep methods, it soon becomes clear that the main contributing factor to a large stability region is the number of stages of the method. In an attempt to accurately compare the stability regions of linear multistep methods to those of Runge-Kutta methods, it is first necessary to find the composition of s linear multistep methods, where s is the number of stages of the Runge-Kutta method. This composition yields a linear multistep method with the same number of stages as the Runge-Kutta method.

In Figure 2.5, the stability regions of Runge-Kutta methods for orders $p = 1, 2, 3, 4$, where $p = s$, are plotted alongside the stability regions of the composite Adams-Bashforth methods where $p = s$.

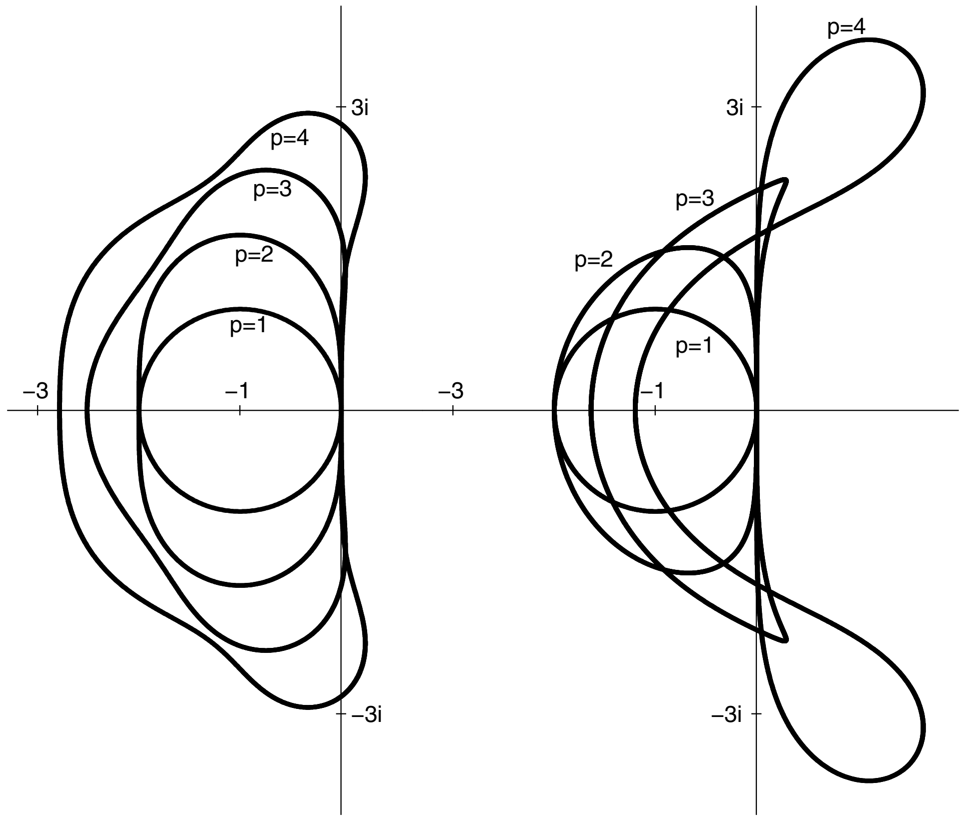


Figure 2.5: Stability regions of the Runge-Kutta (left) and composite Adams-Bashforth (right) methods.

It is clear from Figure 2.5 that the stability domains of the Runge-Kutta methods contain more of the left half plane than the comparative composite Adams-Bashforth method. Therefore, Runge-Kutta methods have more favourable stability properties than the composite Adams-Bashforth methods. Since the complexity of $p(w, z)$ dramatically increases as the order is increased, obtaining suitable conditions on the method which will result in a large stability region becomes extremely complicated. Since Runge-Kutta methods have such good stability properties it would be desirable to obtain general linear methods with the same stability regions as the equivalent Runge-Kutta methods. This leads to the following definition.

Definition 2.20 *If the characteristic polynomial of $M(z)$, known as the stability function, has the special form*

$$p(w,z) = \det\big(wI - M(z)\big) = w^{r-1}\big(w - R(z)\big),$$

then the resulting general linear method is said to possess Runge-Kutta stability.

In this case, the rational function $R(z)$ has the same role as the stability function of a Runge-Kutta method. An implication of requiring a general linear method to have Runge-Kutta stability is that the number of stages s must be at least equal to p for explicit methods, since $R(z) = \exp(z) + O(z^{p+1})$.

2.8 Reducibility

It is possible for two numerical methods to appear somewhat different but when they are used on an initial value problem, they give the same answer. This leads to the following definition.

Definition 2.21 *Two general linear methods M and $\bar{M}$ are equivalent if they generate the same numerical solution for all initial value problems, for a small enough stepsize h.*

The relation "having the same numerical solution as" is an equivalence relation. An equivalence relation on a set makes it possible to partition the set into equivalence classes. Within each equivalence class general linear methods which give the same numerical solution to an initial value problem exist. Therefore, it is common to refer to a general linear method as an equivalence class. To identify whether two general linear methods are equivalent it is necessary to discuss the concept of reducibility.

If it is possible to partition a general linear method with $s = s_1 + s_2$ and $r = r_1 + r_2 + r_3$ with $s_2 + r_2 + r_3 > 0$ so that M has a sparsity pattern

$$M = \left[\begin{array}{cc|ccc} A_{11} & 0 & U_{11} & 0 & U_{13} \\ A_{21} & A_{22} & U_{21} & U_{22} & 0 \\ \hline B_{11} & 0 & V_{11} & 0 & V_{13} \\ B_{21} & B_{22} & V_{21} & V_{22} & V_{23} \\ 0 & 0 & 0 & 0 & V_{33} \end{array}\right], \tag{2.8.1}$$

then the method is equivalent to the partitioned $(s_1 + r_1) \times (s_1 + r_1)$ matrix

$$M = \left[\begin{array}{c|c} A_{11} & U_{11} \\ \hline B_{11} & V_{11} \end{array}\right].$$

This is because the information at the start of step number n is partitioned into $y_1^{n-1}, y_2^{n-1}, y_3^{n-1}$ with $y_3^{n-1} = 0$, then at the end of the step, y_1^n depends only on y_1^{n-1} and $y_3^n = 0$. The last s_2 stages are not used to compute y_1^n and are therefore not needed.

To consider the most general situation allow the stages of A to be permuted to the appropriate form and allow the incoming and outgoing data to be reorganised so as to achieve the appropriate sparsity pattern.

Definition 2.22 *A general linear method M written as an $(s+r) \times (s+r)$ matrix is irreducible if there does not exist an $s \times s$ permutation matrix P and a non-singular matrix R such that*

$$\left[\begin{array}{c|c} P^TAP & P^TUR \\ \hline R^{-1}BP & R^{-1}VR \end{array}\right],$$

has a sparsity pattern given by (2.8.1), where at least one of s_2, r_2 and r_3 are positive.

In the case of Runge-Kutta methods it is possible to show that the equivalence class of methods forms a group (see [26]). Returning now to equivalence classes, it is only necessary to consider the irreducible method of each equivalence class.

Certain methods are irreducible in the sense of Definition 2.22 but should really be represented in an alternative form. The Lobatto IIIA methods are a perfect example. Consider, for example, the 3 stage order 4 Lobatto IIIA method given by

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array},$$

which can be alternatively represented in general linear form as follows

$$\left[\begin{array}{cc|cc} \frac{1}{3} & -\frac{1}{24} & 1 & \frac{5}{24} \\ \frac{2}{3} & \frac{1}{6} & 1 & \frac{1}{6} \\ \hline \frac{2}{3} & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 1 & 0 & 0 \end{array}\right].$$

The question is really what the correct representation of the method is. The answer is the general linear method. The first reason is that it is cheaper in the sense of function evaluations, which is generally the measure used. The second reason is that for stiff problems, using the final stage derivative in the previous step is the only appropriate way to compute the derivative of the first stage of the method. As a consequence, some starting step is needed before the first step of the method is carried out. In fact all Runge-Kutta methods with the first same as last (FSAL) property can be represented as general linear methods with one less stage and one extra output approximation. It will also be assumed, along with the fact that the general linear method is irreducible, that the method is represented in such a way that the minimum number of stages is required. Since such a large proportion of the optimal Runge-Kutta methods constructed have the FSAL property this suggests that fewer stages and more information passed from step to step may lead to better methods.

2.9 Parameter choices

When deriving any type of numerical method certain design choices need to be made so that the method is the most suitable for its desired purpose. This section attempts to explain the design choices used to derive the methods discussed in this thesis. There are four main parameter choices available for general linear methods. They are:

- p, the overall order of the method.
- q, the stage order of the method.
- r, the number of approximations passed from step to step.
- s, the number of stages.

The aim of finding practical general linear methods is very much tied up with suitable relations among the defining parameters. The fact that the defining parameters are interlinked makes it difficult to discuss the most suitable choice in a straightforward manner. The first choice to make is that the stage order q is at least equal to $p-1$. The reason for this is that when a numerical method is used to integrate a stiff problem the order exhibited is not the order p but closely related to the stage order q. This phenomenon, called stiff order, was first introduced by Prothero and Robinson [105] for linear problems and extended to nonlinear problems by Frank, Schneid and Überhuber in [67]. It seems that the stiff order is never greater than $q-1$. The only real benefit that a method such as one from the Gauss, Radau or Lobatto families may have over a method where the order and stage order are very close is when the problem becomes non-stiff and the method exhibits its true order. However, in this case, it would be suitable to use a non-stiff method anyway. General linear methods with $q = p-1$ and $q = p$ are quite different and in general need to be considered separately. Due to the fact that several additional advantages occur when $q = p$, methods are restricted to this case. One of the main advantages is that it is easier to obtain a local error estimate, which is used when the method is implemented in an adaptive mode. This makes it easier to decide whether to increase the order. It also makes it more convenient for deriving specific methods. For these reasons any method, whether it be suitable for stiff or non-stiff problems, will be derived with the assumption that $q = p$.

For a general linear method to be competitive with standard methods such as Runge-Kutta and linear multistep methods, it must be possible to vary the stepsize. This means that the quantities passed from step to step must be sufficiently simple and that the change in stepsize can be performed straight forwardly. For linear multistep methods stepsize can be varied by recomputing the coefficients of the method each time the stepsize is changed. Several codes based on this approach have been developed, the first major one by Krogh [94] in 1969. Even though it is possible to vary stepsize in this way for linear multistep methods and probably for most general linear methods, it will depend on the actual choice of the incoming and outgoing approximations and will be quite specific to the method. An example of such an approach was developed by Jackiewicz, Vermiglio and Zennaro [89] for diagonally-implicit multistage integration methods. A generic and arguably easier approach is to use Nordsieck vectors, introduced by Nordsieck [102] in 1962. The Nordsieck vector has the form

$$y^{[0]} \approx \begin{bmatrix} y(x_0) \\ hy'(x_0) \\ h^2y''(x_0) \\ \vdots \\ h^k y^{(k)}(x_0) \end{bmatrix}.$$

The value k depends on the accuracy required by each component and the stage order q. Since the stage order q is equal to p it follows that $k = p+1$. In the case of Almost Runge-Kutta methods, $q = 2$ and $k = 3$. Almost all general linear methods constructed so far reinterpret the method in Nordsieck form so as to allow for stepsize variation. Examples are the effective order singly implicit Runge-Kutta methods [39], and the two-step Runge-Kutta methods [113].

Any method with $p+1$ independent incoming and outgoing approximations can be represented in Nordsieck form using a simple change of basis. If the method is constructed with incoming and outgoing approximations which do not approximate a Nordsieck vector and have less than $p+1$ independent incoming and outgoing approximations then extra independent components are artificially added at the end. The method can then be reinterpreted in Nordsieck form. The diagonally-implicit multistage integration methods are examples in which extra components are added so as to obtain the $p+1$ independent incoming and outgoing approximations, see [37]. As high stage order is required, that is $p = q$, it is natural to derive methods directly in Nordsieck form with $r = p+1$.

This leaves only the need to discuss suitable choices of the number of stages s relative to the order p. In Section 2.7 it was argued, in order to obtain a suitable stability region, that Runge-Kutta stability should be required. In this case s must be at least equal to p, since $R(z) = \exp(z) + O(z^{p+1})$. When $s = p$ and the other conditions discussed above hold, to obtain Runge-Kutta stability very complicated systems of non-linear equations must be solved. Butcher, Jackiewicz and Mittelmann developed sophisticated numerical searches to find methods of orders five through eight, see [43], [44], [46]. However, as a consequence of $s = p$ there is no flexibility available to reduce the higher order error. When $s > p$, it turns out that it is possible to control the higher order error coefficients along with the surprising fact that the methods can be developed using only linear operations (see Section 3.8). Thus it becomes much easier to search through available methods for desirable properties. One such approach which has proved to produce reasonably good methods is to search for methods where the underlying one step method has minimised local error coefficients. At this stage methods will be restricted to the class where $s = p+1$. However, it is believed that good methods with $s = p+2$ will also exist but this case is left for future work.

The cost of computing the stage approximations for a general linear method is closely related to the structure of the A matrix. It is well known from the theory of Runge-Kutta methods that the cheapest cost occurs when the method is diagonally-implicit. When $r = p+1$ it is possible to have both high stage order and a diagonally-implicit matrix A. Even though the methods which are most likely to be competitive are diagonally implicit the methods will be derived in such a way that the matrix A can have a much more general structure. The main reason for this is to show that some well known methods are special cases.

2.10 Order conditions using trees

For most general linear methods all the output approximations have at least order p. The linear multistep methods are the most common example. This, however, need not be the case. For example, the Almost Runge-Kutta methods have one of the output approximations of lower order than the rest, see, for example, [30]. Given that some of the output approximations are of lower order it is now possible to write down the order conditions for these methods using mappings from trees to real numbers. To take account of the errors introduced by lower order approximations it is necessary to assume that the incoming and outgoing information have the following form

$$S(t) = \begin{bmatrix} E^{-1}\alpha_1(t) \\ E^{-1}\alpha_2(t) \\ E^{-1}\alpha_3(t) \\ \vdots \\ E^{-1}\alpha_r(t) \end{bmatrix}, \qquad \alpha(t) = \begin{bmatrix} \alpha_1(t) \\ \alpha_2(t) \\ \alpha_3(t) \\ \vdots \\ \alpha_r(t) \end{bmatrix}. \tag{2.10.1}$$

With these elementary weight functions it is more convenient to represent the generating functions (2.5.13), (2.5.14) as

$$\begin{aligned}
\xi(t) &= U_1E^{-1}\alpha_1(t) + U_2E^{-1}\alpha_2(t) + \cdots + U_rE^{-1}\alpha_r(t) + A\xi D_1(t),\\
\alpha(t) &= V_1E^{-1}\alpha_1(t) + V_2E^{-1}\alpha_2(t) + \cdots + V_rE^{-1}\alpha_r(t) + B\xi D_1(t).
\end{aligned}$$

These generating functions have mainly been used in the construction of Almost Runge-Kutta methods, see [47]. In this case $r = 3$ and $\alpha(t)$ is a three component Nordsieck vector where only the last approximation is of lower order, in this case order two. To ensure the lower order error is removed from the solution component certain conditions known as annihilating conditions can be found from the generating functions. The reason for including this extra information is to increase the stage order q. If $r > 3$ then the stage order can be increased further. It turns out that the lowest order approximation passed from step to step can always be at least of order q. In Section 2.9 it was argued that the stage order q should be equal to the order p, the incoming and outgoing information should be expressed in the form of a Nordsieck vector, that is $r = p + 1$, and each component of the Nordsieck vector is accurate to order p.

The elementary weight function used to generate a Nordsieck vector at the start and end of the step can be written as

$$S(t) = \begin{bmatrix} \mathbf{1}(t)\\ D_1(t)\\ D_2(t)\\ \vdots\\ D_p(t)\end{bmatrix}, \qquad \alpha(t) = \begin{bmatrix} E(t)\\ ED_1(t)\\ ED_2(t)\\ \vdots\\ ED_p(t)\end{bmatrix}. \tag{2.10.2}$$

With the above assumptions it follows that (2.10.1) and (2.10.2) are equivalent expressions. Therefore the generating functions (2.5.13), (2.5.14) are conveniently represented as

$$\begin{aligned}
\xi(t) &= U_1\mathbf{1}(t) + U_2D_1(t) + \cdots + U_{p+1}D_p(t) + A\xi D_1(t),\\
\alpha(t) &= V_1\mathbf{1}(t) + V_2D_1(t) + \cdots + V_{p+1}D_p(t) + B\xi D_1(t).
\end{aligned}$$

The requirement that the stage order q be equal to the overall order p means the internal stages can be expanded using a Taylor series. Using the generating functions, compare the elementary weight function $\xi(t)$ with the elementary weight function $C(t)$ for trees of order three or less as follows

$$\begin{aligned}
\xi(\emptyset) &= U_1\mathbf{1}(\emptyset) = e \implies U_1 = e,\\
\xi(\bullet) &= U_2D_1(\bullet) + A\xi(\emptyset) = U_2 + Ae = c \implies U_2 = c - Ae,\\
\xi\left(\begin{smallmatrix}\bullet\\|\\\bullet\end{smallmatrix}\right) &= U_3D_2\left(\begin{smallmatrix}\bullet\\|\\\bullet\end{smallmatrix}\right) + A\xi(\bullet) = U_3 + Ac = \frac{c^2}{2} \implies U_3 = \frac{c^2}{2!} - Ac,\\
\xi\left(\begin{smallmatrix}\bullet\\|\\\bullet\\|\\\bullet\end{smallmatrix}\right) &= U_4D_3\left(\begin{smallmatrix}\bullet\\|\\\bullet\\|\\\bullet\end{smallmatrix}\right) + A\xi\left(\begin{smallmatrix}\bullet\\|\\\bullet\end{smallmatrix}\right) = U_4 + A\frac{c^2}{2} = \frac{c^3}{6} \implies U_4 = \frac{c^3}{3!} - A\frac{c^2}{2!},\\
\xi\left(\overset{\bullet\,\bullet}{\underset{\bullet}{\vee}}\right) &= U_4D_3\left(\overset{\bullet\,\bullet}{\underset{\bullet}{\vee}}\right) + A\xi^2(\bullet) = 2U_4 + Ac^2 = \frac{c^3}{3} \implies U_4 = \frac{c^3}{3!} - A\frac{c^2}{2!}.
\end{aligned}$$

The nature of high stage order ensures that the internal stage approximations are identical for all trees of a specific order. So to obtain stage order p the coefficient matrix, U, is determined in terms of the vector c and the matrix A that is

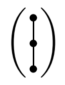
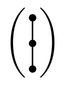
$$U_i = \begin{cases} e, & \text{if } \ i = 1,\\ \dfrac{c^{i-1}}{(i-1)!} - A\dfrac{c^{i-2}}{(i-2)!}, & \text{if } \ i = 2,\dots,p+1. \end{cases} \tag{2.10.3}$$

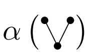
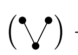
Using the generating functions, compare the elementary weight function $\alpha(t)$ with the elementary weight function $ED_i(t)$ for trees of order three and less as follows

$$\alpha(\emptyset) = V_1\mathbf{1}(\emptyset) = E_1 \implies V_1 = E_1,$$

$$\alpha(\bullet) = V_2 D_1(\bullet) + B\xi(\emptyset) = V_2 + Be = E_2 \implies V_2 = E_2 - Be,$$

α $= V_3 D_2$ $+ B\xi(\bullet) = V_3 + Bc = E_3 \implies V_3 = E_3 - Bc,$

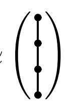
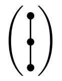
α  $= V_4 D_3$  $+ B\xi$ $= V_4 + B\dfrac{c^2}{2} = E_4 \implies V_4 = E_4 - B\dfrac{c^2}{2!},$

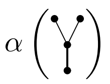
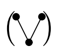
α  $= V_4 D_3$  $+ B\xi^2(\bullet) = 2V_4 + Bc^2 = 2E_4 \implies V_4 = E_4 - B\dfrac{c^2}{2!}.$

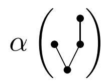
The nature of high stage order ensures that the incoming and outgoing approximations are identical for all trees of a specific order. So to obtain order p the coefficient matrix, V, is determined in terms of the vector c and the matrix B that is

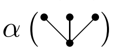
$$V_i = \begin{cases} E_1, & \text{if} \quad i = 1, \\ E_i - B\dfrac{c^{i-2}}{(i-2)!}, & \text{if} \quad i = 2, \dots, p+1. \end{cases} \tag{2.10.4}$$

High stage order ensures that many of the elementary differentials of order higher than p are dependent. As an example, consider a method with stage order and order three. First compute the elementary weight functions of the outgoing approximations for all trees of order four as follows

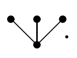
α  $= B\xi$  $= B\dfrac{c^3}{6} = E_5,$

α  $= B\xi$  $= B\dfrac{c^3}{3} = 2E_5,$

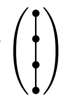
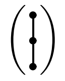
α  $= B\xi$ $\xi(\bullet) = B\dfrac{c^3}{2} = 3E_5,$

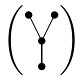
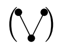
α  $= B\xi^3(\bullet) = Bc^3 = 6E_5.$

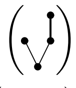
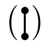
These conditions are all dependent. For order four there is only one independent condition, and without loss of generality assume this is the condition for the bushy tree

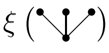


The elementary weight functions of the internal stages for all trees of order four, are

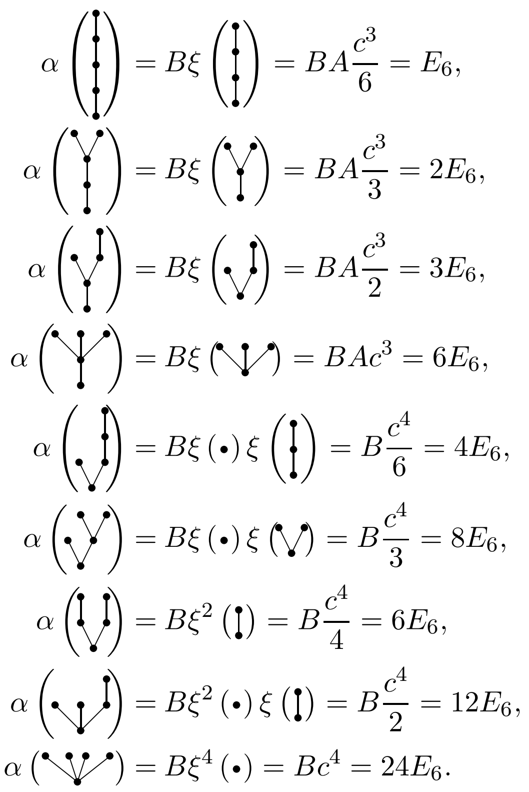
ξ  $= A\xi$  $= A\dfrac{c^3}{6},$

ξ  $= A\xi$  $= A\dfrac{c^3}{3},$

ξ  $= A\xi$  $\xi(\bullet) = A\dfrac{c^3}{2},$

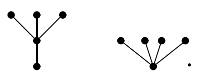
ξ  $= A\xi^3(\bullet) = Ac^3.$

Consider now the elementary weight functions of the outgoing approximations for all trees of order five

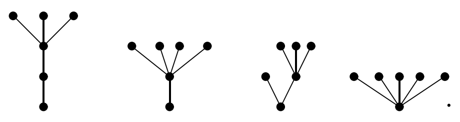


$$\alpha\left(\;\right) = B\xi\left(\;\right) = BA\frac{c^3}{6} = E_6,$$
$$\alpha\left(\;\right) = B\xi\left(\;\right) = BA\frac{c^3}{3} = 2E_6,$$
$$\alpha\left(\;\right) = B\xi\left(\;\right) = BA\frac{c^3}{2} = 3E_6,$$
$$\alpha\left(\;\right) = B\xi\left(\;\right) = BAc^3 = 6E_6,$$
$$\alpha\left(\;\right) = B\xi\left(\bullet\right)\xi\left(\;\right) = B\frac{c^4}{6} = 4E_6,$$
$$\alpha\left(\;\right) = B\xi\left(\bullet\right)\xi\left(\;\right) = B\frac{c^4}{3} = 8E_6,$$
$$\alpha\left(\;\right) = B\xi^2\left(\;\right) = B\frac{c^4}{4} = 6E_6,$$
$$\alpha\left(\;\right) = B\xi^2\left(\bullet\right)\xi\left(\;\right) = B\frac{c^4}{2} = 12E_6,$$
$$\alpha\left(\;\right) = B\xi^4\left(\bullet\right) = Bc^4 = 24E_6.$$

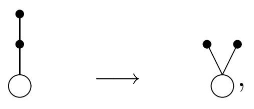
So for order five there are two independent conditions, and without loss of generality, assume they are those corresponding to the trees



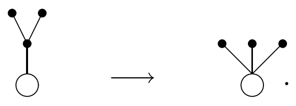
Repeating this process for order six trees yields four independent conditions related to the trees



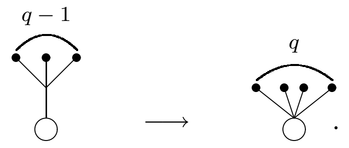
Having high stage order for general linear methods is very similar to the C condition for Runge-Kutta methods. Using the above example it can be seen that stage order two means all trees of the form on the left of the following diagram get reduced to the trees on the right.



where ○ denotes some common tree. Stage order three means all trees of the form on the left of the following diagram get reduced to the trees on the right.

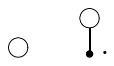


Stage order q means all trees of the form on the left of the following diagram get reduced to the trees on the right.

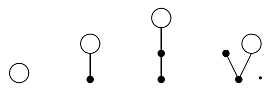


In order to count the exact number of independent trees it is necessary to first note that if a method has stage order q then the independent trees of order greater than q and less than $2q+3$ are bushy trees of order at least $q+1$ placed on top of a tall tree with only twigs. Trees of this form are best shown diagrammatically. For order p to equal $q+1$ there is one independent condition, the bushy tree of order $q+1$, represented as follows by

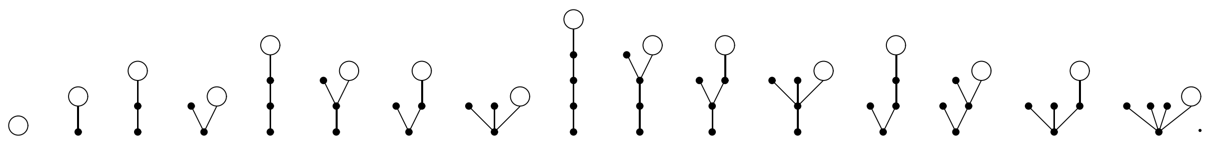
For order p to be equal to $q+2$ there are two independent conditions, represented as



For order p to be equal to $q+3$ there are four independent conditions,

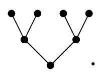


For order p to be equal to $q+5$ there are 16 independent conditions,

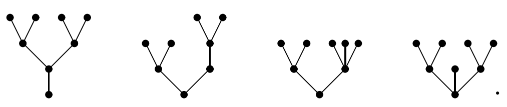


It follows that if order p is equal to $q+i$ then there are 2^{i-1} independent order conditions if $q+i<2q+3$.

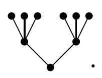
When this inequality is broken certain trees not described above are introduced. For a method with stage order two the first tree, of order seven, which is not one of the above trees, is given by



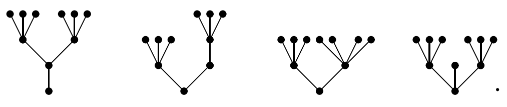
The four trees of order eight which are not one of the trees given above they are



For a method with stage order three the first tree which is of order nine which is not one of the above trees is given by



There are again four trees of order ten which are not one of the trees given above they are



The number of independent trees eventually grows at the same rate as the traditional Runge-Kutta case just delayed due to the increase in stage order. Below is a table with the number of independent order conditions for various values of p and q. Note that it is always assumed that p is at least equal to q.

		q									
		1	2	3	4	5	6	7	8	9	10
p	1	1									
	2	1	1								
	3	2	1	1							
	4	4	2	1	1						
	5	9	4	2	1	1					
	6	20	8	4	2	1	1				
	7	48	17	8	4	2	1	1			
	8	115	36	16	8	4	2	1	1		
	9	286	76	33	16	8	4	2	1	1	
	10	719	161	68	32	16	8	4	2	1	1

Table 2.5: Number of independent trees for various values of p versus q.

A similar recursion to (2.4.1) for the number of independent trees can also be obtained. Knowing the number of independent rooted trees is especially useful when searching for good methods. That is to say when searching for general linear methods where the underlying one step method is minimised, yields the same number of independent conditions.

2.11 Composition of methods

When comparing numerical methods it is important to express the methods in such a way that the comparison is fair. It is well known that a linear multistep method of order p will generally require more steps to complete the integration than a Runge-Kutta method of order p. This is partly because the stability region of a linear multistep method of order p is much smaller than that of a corresponding order p Runge-Kutta method. The main reason that the stability regions are smaller, as was discussed in Section 2.7, is that a linear multistep method has one function evaluation per step, whereas a Runge-Kutta method has s. As there seems to be no hard and fast rule for measuring the performance of numerical methods, a sensible choice would be to compare the number of function evaluations, at least in the non-stiff case. In the stiff case this becomes much more complicated due to the need to perform iterations to compute the stages $Y^{[n]}$. It now becomes important to represent the methods so that the number of function evaluations are equivalent in each of the methods. Therefore, given two numerical methods with s_1 and s_2 stages respectively, it seems natural to compare s_2 steps of the first method with s_1 steps of the second method. However, several problems now arise. For example finding the stability regions of the composite method or the higher order error coefficients. To overcome these problems it is natural to perform say m steps of the method, with stepsize h/m. The main complication with this approach is that the incoming and outgoing approximations can be complicated expressions involving the stepsize h.

In Section 2.9 it was argued that the information passed from step to step should in fact be a Nordsieck vector. If this is the case then it is simple to rescale the incoming and outgoing information by pre-multiplying the Nordsieck vector by the diagonal matrix D given by

$$D = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{m^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{m^{p-2}} & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{m^{p-1}} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{m^p} \end{bmatrix}.$$

In a lot of cases where the information passed from step to step is not a Nordsieck vector it is also possible to construct the matrix D which re-interprets the information with the new stepsize.

Consider the general linear method (2.1.2) with m consecutive steps of size $1/m$. That is

$$\begin{aligned} Y^{[1]} &= \frac{1}{m}hAf\big(Y^{[1]}\big) + UDy^{[0]}, \\ Dy^{[1]} &= \frac{1}{m}hBf\big(Y^{[1]}\big) + VDy^{[0]}, \\ Y^{[2]} &= \frac{1}{m}hUBf\big(Y^{[1]}\big) + \frac{1}{m}hAf\big(Y^{[2]}\big) + UVy^{[0]}, \\ Dy^{[2]} &= \frac{1}{m}hVBf\big(Y^{[1]}\big) + \frac{1}{m}hBf\big(Y^{[2]}\big) + V^2Dy^{[0]}, \\ &\vdots \\ Y^{[m]} &= \frac{1}{m}hUV^{m-2}Bf\big(Y^{[1]}\big) + \frac{1}{m}hUV^{m-3}Bf\big(Y^{[2]}\big) + \cdots + \frac{1}{m}hAf\big(Y^{[m]}\big) + UV^{m-1}y^{[0]}, \\ Dy^{[m]} &= \frac{1}{m}hV^{m-1}Bf\big(Y^{[1]}\big) + \frac{1}{m}hV^{m-2}Bf\big(Y^{[2]}\big) + \cdots + \frac{1}{m}hBf\big(Y^{[m]}\big) + V^mDy^{[0]}. \end{aligned}$$

This is expressed as a single general linear methods as follows

$$M = \left[\begin{array}{ccccc|c} \frac{1}{m}A & 0 & \cdots & 0 & 0 & UD \\ \frac{1}{m}UB & \frac{1}{m}A & \cdots & 0 & 0 & UVD \\ \frac{1}{m}UVB & \frac{1}{m}UB & \cdots & 0 & 0 & UV^2D \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ \frac{1}{m}UV^{m-4}B & \frac{1}{m}UV^{m-5}B & \cdots & 0 & 0 & UV^{m-2}D \\ \frac{1}{m}UV^{m-3}B & \frac{1}{m}UV^{m-4}B & \cdots & \frac{1}{m}A & 0 & UV^{m-2}D \\ \frac{1}{m}UV^{m-2}B & \frac{1}{m}UV^{m-3}B & \cdots & \frac{1}{m}UB & \frac{1}{m}A & UV^{m-1}D \\ \hline \frac{1}{m}D^{-1}V^{m-1}B & \frac{1}{m}D^{-1}V^{m-2}B & \cdots & \frac{1}{m}D^{-1}VB & \frac{1}{m}D^{-1}B & D^{-1}V^mD \end{array}\right]. \tag{2.11.1}$$

When comparing explicit methods it can be assumed that they have been represented so that the number of stages is the same for both methods. The stability regions of the Runge-Kutta and linear multistep methods plotted in Figure 2.5 have the same number of stages.

2.12 Examples of general linear methods

In this section several well known examples of general linear methods will be reviewed. The methods will be represented in general linear form and the main advantages and disadvantages of the methods will be discussed. The methods already described namely the linear multistep and Runge-Kutta will only be discussed very briefly.

2.12.1 Linear multistep methods

The general form of a k-step linear multistep method is

$$\sum_{i=0}^{k} \alpha_i y_{n-i} = h \sum_{i=0}^{k} \beta_i f(y_{n-i}),$$

where the coefficients α and β are chosen to maximise the order of the method. There are two main types of linear multistep methods, those suitable for non-stiff problems and those suitable for stiff problems. These are respectively known as the Adams methods and the backward differentiation formulae methods. Even though the Adams methods have an implicit option this is generally only used in the predict evaluate correct (evaluate) situation as the implicit methods have very small stability regions.

Adams methods

The first type of linear multistep methods derived are now known as the Adams methods. These are generally broken down into two types explicit ($\beta_0 = 0$), commonly known as Adams-Bashforth methods, and implicit ($\beta_0 \neq 0$), commonly known as Adams-Moulton methods. They are generally represented as

$$y_n = y_{n-1} + h \sum_{i=0}^{k} \beta_i f(y_{n-i}),$$

or in general linear form as follows

$$\left[\begin{array}{c} Y_1 \\ \hline y_n \\ hf(Y_1) \\ hf(y_{n-1}) \\ hf(y_{n-2}) \\ \vdots \\ hf(y_{n-k-1}) \end{array}\right] = \left[\begin{array}{c|cccccc} \beta_0 & 1 & \beta_1 & \beta_2 & \cdots & \beta_{k-1} & \beta_k \\ \hline \beta_0 & 1 & \beta_1 & \beta_2 & \cdots & \beta_{k-1} & \beta_k \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{array}\right] \left[\begin{array}{c} hf(Y_1) \\ \hline y_{n-1} \\ hf(y_{n-1}) \\ hf(y_{n-2}) \\ hf(y_{n-3}) \\ \vdots \\ hf(y_{n-k}) \end{array}\right].$$

Most variable step, variable order codes using explicit methods are based on the Adams methods. However generally these methods are implemented in a predict evaluate correct (PEC) or predict evaluate correct evaluate (PECE) scheme. Both of these methods are based on an Adams-Bashforth predictor and an Adams-Moulton corrector,

$$y_n^* = y_{n-1} + h \sum_{i=1}^{k} \beta_i^* f(y_{n-i}),$$

$$y_n = y_{n-1} + h\beta_0 f(y_n^*) + h \sum_{i=1}^{k} \beta_i f(y_{n-i}),$$

where $*$ represents the Adams-Bashforth predictor.

The PEC method is represented in general linear form as follows

$$\left[\begin{array}{c} Y_1 \\ \hline y_n \\ hf(y_n) \\ hf(y_{n-1}) \\ hf(y_{n-2}) \\ \vdots \\ hf(y_{n-k-1}) \end{array}\right] = \left[\begin{array}{c|cccccc} 0 & 1 & \beta_1^* & \beta_2^* & \cdots & \beta_{k-1}^* & \beta_k^* \\ \hline \beta_0 & 1 & \beta_1 & \beta_2 & \cdots & \beta_{k-1} & \beta_k \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{array}\right] \left[\begin{array}{c} hf(Y_1) \\ \hline y_{n-1} \\ hf(y_{n-1}) \\ hf(y_{n-2}) \\ hf(y_{n-3}) \\ \vdots \\ hf(y_{n-k}) \end{array}\right].$$

Alternatively the PECE method in general linear form is represented as

$$\left[\begin{array}{c} Y_1 \\ Y_2 \\ \hline y_n \\ hf(y_n) \\ hf(y_{n-1}) \\ hf(y_{n-2}) \\ \vdots \\ hf(y_{n-k-1}) \end{array}\right] = \left[\begin{array}{cc|cccccc} 0 & 0 & 1 & \beta_1^* & \beta_2^* & \cdots & \beta_{k-1}^* & \beta_k^* \\ \beta_0 & 0 & 1 & \beta_1 & \beta_2 & \cdots & \beta_{k-1} & \beta_k \\ \hline \beta_0 & 0 & 1 & \beta_1 & \beta_2 & \cdots & \beta_{k-1} & \beta_k \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{array}\right] \left[\begin{array}{c} hf(Y_1) \\ hf(Y_2) \\ \hline y_{n-1} \\ hf(y_{n-1}) \\ hf(y_{n-2}) \\ hf(y_{n-3}) \\ \vdots \\ hf(y_{n-k}) \end{array}\right].$$

Generally the methods are implemented in Nordsieck form as this makes it considerably more convenient to change the stepsize and order. One of the main disadvantages of these methods is that the stability regions are relatively small.

Backward Differentiation Formulae methods

Backward Differentiation Formulae methods, or commonly known as BDF methods were originally developed by Curtiss and Hursfielder [58]. These methods were introduced to overcome the difficulty in solving stiff problems with Adams methods due to their lack of stability. The updated approximation to the solution is computed as follows

$$y_n = \sum_{i=1}^{k} \alpha_i y_{n-i} + h\beta_0 f(y_n),$$

which is represented in general linear form, as

$$\left[\begin{array}{c} Y_1 \\ \hline y_n \\ y_{n-1} \\ y_{n-2} \\ y_{n-3} \\ \vdots \\ y_{n-k-1} \end{array}\right] = \left[\begin{array}{c|cccccc} \beta_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{k-1} & \alpha_k \\ \hline \beta_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{k-1} & \alpha_k \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{array}\right] \left[\begin{array}{c} hf(Y_1) \\ \hline y_{n-1} \\ y_{n-2} \\ y_{n-3} \\ y_{n-4} \\ \vdots \\ y_{n-k} \end{array}\right].$$

When implemented these methods are generally reinterpreted in Nordsieck form to overcome the difficulty of variable stepsize. The BDF methods are only A-stable for orders one and two but the stability regions are considered by many as satisfactory for orders up to five. Therefore, all variable step, variable order implementations of BDF methods use only order one through to five. These implementations are satisfactory if the eigenvalues of the problem do not have large imaginary components.

2.12.2 Runge-Kutta methods

Runge-Kutta methods are one of the major types of general linear methods. An s stage Runge-Kutta method has the form

$$\begin{aligned} Y_i &= y_{n-1} + \sum_{j=0}^{s} a_{ij} f(Y_j), \qquad i = 1, 2, \ldots, s, \\ y_n &= y_{n-1} + \sum_{i=0}^{s} b_i f(Y_i). \end{aligned}$$

This is expressed in general linear form as

$$\left[\begin{array}{c} Y_1 \\ Y_2 \\ \vdots \\ Y_{s-1} \\ Y_s \\ \hline y_n \end{array}\right] = \left[\begin{array}{ccccc|c} a_{11} & a_{12} & \cdots & a_{1s-1} & a_{1s} & 1 \\ a_{21} & a_{22} & \cdots & a_{2s-1} & a_{2s} & 1 \\ \vdots & \vdots & & \vdots & \vdots & \\ a_{s-11} & a_{s-12} & \cdots & a_{s-1s-1} & a_{s-1s} & 1 \\ a_{s1} & a_{s2} & \cdots & a_{ss-1} & a_{ss} & 1 \\ \hline b_1 & b_2 & \cdots & b_{s-1} & b_s & 1 \end{array}\right] \left[\begin{array}{c} hf(Y_1) \\ hf(Y_2) \\ \vdots \\ hf(Y_{s-1}) \\ hf(Y_s) \\ \hline y_{n-1} \end{array}\right].$$

Runge-Kutta methods have excellent stability properties for both explicit and implicit methods. However this comes at the cost of a large number of function evaluations. This is further increased if continuous solutions are required. Changing stepsize and order is convenient and practical since only one quantity is passed from step to step. However estimating the high order error used to change order is difficult.

Much of the theoretical basis of general linear methods is based on that of Runge-Kutta methods. For example, using the mappings from trees to real numbers to compute the order conditions for a method. The generating functions for the Runge-Kutta methods are given by

$$\begin{aligned} \xi(t) &= e\mathbf{1}(t) + A\xi(t), \\ \alpha(t) &= \mathbf{1}(t) + b^T \xi(t). \end{aligned}$$

So for a method to be of order p, $\alpha(t) = E(t)$ for all trees of order less than or equal to p.

2.12.3 Generalised multistep, hybrid and modified multistep methods

Generalised multistep, hybrid and modified multistep methods are the names given by Gragg and Stetter [73], Gear [68] and Butcher [18] to describe methods which include an additional off-step point to a predictor-corrector pair. These methods were discovered simultaneously and independently by the three parties. The off-step point allows the famous Dahlquist barrier to be broken, allowing an order of $2k + 1$. Using off-step points is of course a familiar feature of the Runge-Kutta methods and thus the reason for the name hybrid methods.

Modifying the formula for a multistep method to include information about the derivative part way through the step, gives

$$y_n = \alpha_1 y_{n-1} + \alpha_1 y_{n-1} + \cdots + \alpha_1 y_{n-1} + h(\beta f_{n-\theta} + \beta_0 f_n + \beta_1 f_{n-1} + \beta_2 f_{n-2} + \cdots + \beta_k f_{n-k}),$$

where θ, which is not an integer, is typically chosen so that $0 < \theta < 1$. In a similar way to linear multistep methods, it is first necessary to estimate $y_{n-\theta}$ and y_n. Note that y_n could be computed using an iterative technique, however the general approach adopts the use of a predictor, making the overall method explicit. The off-step approximation is predicted first. This information is then used to predict y_n. The predictor formulae take the following form

$$\begin{aligned}
\widehat{y}_{n-\theta} &= a_1 y_{n-1} + a_2 y_{n-2} + \cdots + a_k y_{n-k} + h(b_1 f_{n-1} + b_2 f_{n-2} + \cdots + b_k f_{n-k}),\\
\widehat{y}_n &= A_1 y_{n-1} + A_2 y_{n-2} + \cdots + A_k y_{n-k} + h(B f_{n-\theta} + B_1 f_{n-1} + B_2 f_{n-2} + \cdots + B_k f_{n-k}).
\end{aligned}$$

That is, the predicted values y_n and $f_{n-\theta}$ are denoted by $\widehat{y}_n$ and $\widehat{f}_{n-\theta}$. The method coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k, \beta, \beta_0, \ldots, \beta_k$ can be chosen so that y_n can have order $2k+1$. In the same way the coefficients $a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_k$ can be chosen so that $\widehat{y}_{n-\theta}$ has order $2k-1$ and $A_1, A_2, \ldots, A_k, B_1, B_2, \ldots, B_k$ can be chosen so that $\widehat{y}_n$ has order $2k$. However it is generally regarded as more appropriate to choose the coefficients so that the errors of the predictors are in a ratio so that the leading term of the error in $\beta \widehat{f}_{n-\theta} + \beta_0 \widehat{f}_n$ vanishes.

The hybrid methods can be represented in general linear form as follows

$$\left[\begin{array}{c}
\widehat{y}_{n-\theta}\\ \widehat{y}_n\\ y_n\\ \hline y_n\\ y_{n-1}\\ \vdots\\ y_{n-k+1}\\ f_n\\ f_{n-1}\\ \vdots\\ f_{n-k+1}
\end{array}\right]
=
\left[\begin{array}{ccc|cccccccc}
0 & 0 & 0 & a_1 & a_2 & \cdots & a_k & b_1 & b_2 & \cdots & b_k\\
B & 0 & 0 & A_1 & A_2 & \cdots & A_k & B_1 & B_2 & \cdots & B_k\\
\beta & \beta_0 & 0 & \alpha_1 & \alpha_2 & \cdots & \alpha_k & \beta_1 & \beta_2 & \cdots & \beta_k\\ \hline
\beta & \beta_0 & 0 & \alpha_1 & \alpha_2 & \cdots & \alpha_k & \beta_1 & \beta_2 & \cdots & \beta_k\\
0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots\\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0\\
0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0\\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots\\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0
\end{array}\right]
\left[\begin{array}{c}
h\widehat{f}_{n-\theta}\\ h\widehat{f}_n\\ hf_n\\ \hline y_{n-1}\\ y_{n-2}\\ \vdots\\ y_{n-k}\\ f_{n-1}\\ f_{n-2}\\ \vdots\\ f_{n-k}
\end{array}\right].$$

Since the original three papers on hybrid methods at least five related papers have appeared. Butcher in [20] investigated methods with two and three off-step points. It was shown that these methods could obtain order $2k+2$ and $2k+3$ respectively. The methods were constructed with a predictor for each off-step point and at y_n, along with a corrector at y_n. These methods were elegantly derived using techniques from complex analysis. Kohfeld and Thomson [92] developed accurate predictors so that the highly accurate correctors of Gragg and Stetter [73] are not lost. Along with Brush [10] they developed generalised multistep methods with two off-step points. In [93] Kohfeld and Thomson followed Nordsieck [37] and Gragg and Stetter [73] to construct generalised multistep methods in Nordsieck form. Beaudet [5] considered methods where all derivative evaluations were performed at locations off the grid points of final solution values. The most recent paper by Butcher and O'Sullivan [48] derived methods of order five directly in Nordsieck form using using techniques from complex analysis. These methods were comparable to the methods of Dormand and Prince [63]. The main disadvantage of these methods is that it is difficult to control the size of the stability region.

2.12.4 Mono-implicit Runge-Kutta methods

The mono-implicit Runge-Kutta methods first discovered by Cash [53] are a special class of Runge-Kutta methods. For further details of these methods see for example the papers by Cash and Singal [54], van Bockhoven [118] and Gupta [75] and references within.

Given a Runge-Kutta method

$$\begin{array}{c|c} c & \widehat{A} \\ \hline & b^T \end{array},$$

if there exists a vector w such that $A = \widehat{A} - wb^T$ is strictly lower triangular then the method can be written as

$$\begin{aligned} Y^{[n]} &= ey_{n-1} + h(A + wb^T)f\big(Y^{[n]}\big) \\ &= ey_{n-1} + whb^T f\big(Y^{[n]}\big) + hAf\big(Y^{[n]}\big) \\ &= ey_{n-1} + w(y_n - y_{n-1}) + hAf\big(Y^{[n]}\big). \end{aligned}$$

So a mono-implicit Runge-Kutta method is given by

$$\begin{aligned} Y^{[n]} &= (e - w)y_{n-1} + wy_n + hAf\big(Y^{[n]}\big), \\ y_n &= y_{n-1} + hb^T f\big(Y^{[n]}\big), \end{aligned}$$

which is conveniently represented in the following tableau

$$\begin{array}{c|c|c} c & w & A \\ \hline & & b^T \end{array}.$$

Compare the Taylor series expansions of the stages $Y^{[n]}$ with $y(ex_{n-1} + ch)$ and it follows that a method has stage order q if

$$\frac{c^i}{i!} = \frac{w}{i!} + A\frac{c^{i-1}}{(i-1)!},$$

for $i = 1, 2, \ldots, q$. It can easily be shown that the maximum stage order of a mono-implicit Runge-Kutta method is three. The generating functions for these methods are given by

$$\begin{aligned} \xi(t) &= (e - w)\mathbf{1}(t) + wE\mathbf{1}(t) + A\xi D_1(t), \\ \alpha(t) &= \mathbf{1}(t) + b^T\xi D_1(t). \end{aligned}$$

To obtain order p, $\alpha(t) = E(t)$ for all trees less than or equal to order p. To overcome the condition on the stage order without breaking the one implicit unknown feature of these methods, a more general form is required. This is given by

$$\begin{aligned} Y^{[n]} &= (e - w)y_{n-1} + wy_n + uhy'_{n-1} + vhy'_n + hAf\big(Y^{[n]}\big), \\ y_n &= y_{n-1} + hb^T f\big(Y^{[n]}\big). \end{aligned}$$

To obtain stage order q it is necessary to satisfy

$$\begin{aligned} c &= w + u + v + Ae, \\ \frac{c^i}{i!} &= \frac{w}{i!} + \frac{v^{i-1}}{(i-1)!} + A\frac{c^{i-1}}{(i-1)!}, \end{aligned}$$

for $i = 2, 3, \ldots, q$.

The generating functions of these methods are

$$\begin{aligned}\xi(t) &= (e-w)\mathbf{1}(t) + wE\mathbf{1}(t) + uD_1(t) + vED_1(t) + A\xi D_1(t),\\ \alpha(t) &= \mathbf{1}(t) + b^T\xi D_1(t).\end{aligned}$$

Again to obtain order p, $\alpha(t) = E(t)$ for all trees less than or equal to order p. The mono-implicit Runge-Kutta methods are especially popular for the solution of two-point boundary value problems, based on deferred correction. Important contributions to mono-implicit Runge-Kutta methods for boundary value problems have been made by Enright and Muir [100], where they examined how these methods fit into the general class of implicit Runge-Kutta methods.

2.12.5 Almost Runge-Kutta methods

Almost Runge-Kutta methods are a special class of general linear methods which are very similar to the Runge-Kutta methods. The almost Runge-Kutta methods developed by Butcher in [30] and [31] only pass three quantities from step to step. These are approximations to $y(x_n), hy'(x_n)$ and $h^2y''(x_n)$. The stability regions are identical to those of the Runge-Kutta methods. This means that Runge-Kutta stability defined in Definition 2.20 holds. The main advantage these methods have is that the order of the stages is increased from one for the Runge-Kutta methods to two for the Almost Runge-Kutta methods. This is achieved by increasing the number of quantities passed from step to step. The approximation to $h^2y''(x_n)$ is only of order two and the error caused by this approximation must be eliminated. This results in three types of conditions:

- Order conditions.
- Annihilation conditions.
- Runge-Kutta stability conditions.

The main advantage of the increase in stage order is that error estimates and continuous solutions can be more easily achieved. An almost Runge-Kutta method takes the form

$$\left[\begin{array}{c} Y^{[n]} \\ \hline y_n \\ hy'_n \\ y''_n \end{array}\right] = \left[\begin{array}{c|ccc} A & e & c-Ae & \frac{1}{2}c^2 - Ac \\ \hline b^T & 1 & b_0 & 0 \\ e_s^T & 0 & 0 & 0 \\ \beta^T & 0 & \beta_0 & 0 \end{array}\right] \left[\begin{array}{c} hf(Y^{[n]}) \\ \hline y_{n-1} \\ hy'_{n-1} \\ y''_{n-1} \end{array}\right],$$

where A is a strictly lower triangular $s \times s$ matrix, such that $e_s^T A = b^T$. This means the last row of A is the same as b^T which implies that $c_s - b^T e = b_0$ and $\frac{1}{2}c_s^2 - b^T c = 0$. The form of the U matrix follows directly by comparing the Taylor series expansions of $y(ex_{n-1} + ch)$ with the internal stages of the method. Note that the last column of V has all zero elements, this is to ensure that the low order approximation to $h^2y''(x_n)$ does not affect the approximate solution.

The generating functions for the almost Runge-Kutta methods are given by

$$\begin{aligned}\xi(t) &= \mathbf{1}(t) + (c-Ae)D_1(t) + \Big(\frac{1}{2}c^2 - Ac\Big)\eta(t) + A\xi D_1(t),\\ \alpha(t) &= \mathbf{1}(t) + b_0 D_1(t) + b^T\xi D_1(t),\\ E\eta(t) &= \beta_0 D_1(t) + \beta^T\xi D_1(t).\end{aligned}$$

For these methods it is assumed that the $C(2)$ and $D(1)$ conditions hold. To obtain a method of say order five, $\alpha(t) = E(t)$, for all trees up to and including order five and $E\eta(t) = D_2(t)$ for all trees up to and including order two. These result in the order and the annihilating conditions. Since there are only three approximations passed from step to step $M(z)$ is a 3×3 matrix. The stability conditions are found by forcing the $\operatorname{tr}(M(z)) = \exp(z) + O(z^{p+1})$.

A special fourth order method with zero error coefficients was derived by Butcher and Moir in [47]. This method is very interesting in that it exhibits fifth order behaviour in a constant stepsize implementation but fourth order in a variable stepsize implementation. The order can be enhanced to fifth order in the variable stepsize case if the method is implemented carefully. Extending Almost Runge-Kutta methods so that they are suitable for the solution of stiff problems is difficult. This is because unless A is fully implicit the method will have only low stage order and thus suffer from the order reduction phenomenon. Almost Runge-Kutta methods with more than three quantities passed from step to step have been derived by Chan [55]. These were derived so as to increase the stage order. The difficulty with these methods is that it becomes more and more difficult to force the method to have Runge-Kutta stability.

2.12.6 Two step Runge-Kutta methods

Another extension of Runge-Kutta methods which has become popular recently is the two-step Runge-Kutta methods. These methods were introduced by Renaut [106], Jackiewicz, Renaut and Feldstein [85] and Jackiewicz, Renaut and Zennaro [86]. These methods were generalised by Jackiewicz and Tracogna [87] and are given by

$$\begin{aligned} Y^{[n]} &= h\big(Af\big(Y^{[n-1]}\big) + Bf\big(Y^{[n]}\big)\big) + uy_{n-2} + (e-u)y_{n-1}, \\ y_n &= h\big(v^T f\big(Y^{[n]}\big) + w^T f\big(Y^{[n]}\big)\big) + \theta y_{n-2} + (1-\theta)y_{n-1}. \end{aligned}$$

The method is conveniently represented as follows

$$\begin{array}{c|c|c} u & A & B \\ \hline \theta & v^T & w^T \end{array}.$$

The matrix B determines the implementation costs of the method. Following the idea used for diagonally implicit multistage integration methods [28] there are four subclasses. They are sequential for non-stiff problems (B is strictly lower triangular), sequential for stiff problems (B is lower triangular), parallel for non-stiff problems ($B = 0$) and parallel for stiff problems (B is diagonal). The advantage of these methods over Runge-Kutta methods is that the extra stages make it possible to increase the stage order so that it is close to the overall order.

The two-step Runge-Kutta methods are represented as general linear methods as follows

$$\left[\begin{array}{c} Y^{[n]} \\ \hline y_n \\ y_{n-1} \\ hf\big(Y^{[n]}\big) \end{array}\right] = \left[\begin{array}{c|ccc} B & e-u & u & A \\ \hline w^T & 1-\theta & \theta & v^T \\ 0 & 1 & 0 & 0 \\ I & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} hf\big(Y^{[n]}\big) \\ \hline y_{n-1} \\ y_{n-2} \\ hf\big(Y^{[n-1]}\big) \end{array}\right].$$

Due to the intrinsic nature of these methods it is difficult to change the stepsize. Jackiewicz and Tracogna [88] developed methods on non-uniform meshes, however this requires the coefficients to be computed every time the stepsize is changed. Tracogna [113] represented the two-step Runge-Kutta methods in Nordsieck form and provided them with local error estimates and interpolants. These features require no extra stages to be evaluated because of the high stage order of these methods.

The order conditions for these methods were originally derived by Jackiewicz and Tracogna [87] adopting the approach of Albrecht [2] to these new methods. Hairer and Wanner [79] and Butcher and Tracogna [49] constructed the order conditions using B-series. The resulting generating functions are given by

$$\begin{aligned}\xi(t) &= uE^{-1}(t) + (e-u)\mathbf{1}(t) + AE^{-1}\xi D_1(t) + B\xi D_1(t),\\ \alpha(t) &= \theta E^{-1}(t) + (1-\theta)\mathbf{1}(t) + v^T E^{-1}\xi D_1(t) + w^T \xi D_1(t).\end{aligned}$$

So to obtain order p, $\alpha(t) = E(t)$, for all trees of order less than or equal to p. Some of the disadvantages of two-step Runge-Kutta methods are they become difficult to derive for high orders. Also these methods do not have Runge-Kutta stability or a two-step equivalent which makes finding methods with large stability regions difficult.

2.12.7 Diagonally implicit multistage integration methods

Diagonally Implicit Multistage Integration Methods (DIMSIMs) were introduced by Butcher [28], in 1993. The leading coefficient matrix, A, in these methods can have four different structures known as “types”, depending on whether the intended application is non-stiff or stiff and on whether the computer architecture is sequential or parallel. The four types are shown in Figure 2.6.

Type	Form of A	Problem	Computer
1	$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \cdots & 0 \end{bmatrix}$	Non-stiff	Sequential
2	$\begin{bmatrix} \lambda & 0 & \cdots & 0 \\ a_{21} & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \cdots & \lambda \end{bmatrix}$	Stiff	Sequential
3	$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$	Non-stiff	Parallel
4	$\begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix}$	Stiff	Parallel

Figure 2.6: Structure of the A matrix for DIMSIMs.

The DIMSIMs can be described by the following conditions:

- The A matrix should be lower triangular, with constant diagonals.
- The quantities approximated by the incoming and outgoing data should be related to the exact solution by a weighted Taylor series.
- The V matrix is of rank one, which guarantees zero stability.
- The order of the stages should be close, if not identical, to the overall order of the method.

Including the requirement, that the stage values have a specific order, allows a considerable simplification of the order conditions as was seen in Section 2.10. The key idea in representing these methods is to express the incoming quantities as the weighted Taylor series

$$y_i^{[n-1]} = \alpha_{i0}y(x_{n-1}) + \alpha_{i1}hy'(x_{n-1}) + \cdots + \alpha_{ip}h^p y^{(p)}(x_{n-1}) + O(h^{p+1}), \qquad i = 1, 2, \ldots, r,$$

with the corresponding outgoing approximations given by

$$y_i^{[n]} = \alpha_{i0}y(x_n) + \alpha_{i1}hy'(x_n) + \cdots + \alpha_{ip}h^p y^{(p)}(x_n) + O(h^{p+1}), \qquad i = 1, 2, \ldots, r.$$

It is generally believed that methods where the number of stages and output approximations are equivalent to the order and stage order are the most likely of the DIMSIM family to be of practical use. Most effort in this area has been concentrated in developing these methods, see Butcher [29], Wright [116], Butcher and Jackiewicz [44] and Butcher, Jackiewicz and Mittelmann [46]. DIMSIMs of the form $s+1 = r = q$, $p = q$ or $q+1$, $s = r+1 = q$, $p = q$ or $q+1$ and $s = r = q = p-1$, have also been studied by Butcher and Jackiewicz in [42] and [43]. The large motivation for this first choice was because with the use of the following theorem, there were exactly the same number of free parameters as there were equations required to ensure that the stability region of the method was identical to that of a Runge-Kutta method.

Theorem 2.23 *A DIMSIM with $s = r$ and where $Ve = e$, has order and stage order $p = q = r$, if and only if*

$$B = B_0 - AB_1 - VB_2 + VA,$$

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

$$B_0 = \frac{\int_0^{1+c_i} \phi_j(x)dx}{\phi_j(c_j)}, \qquad B_1 = \frac{\phi_j(1+c_i)}{\phi_j(c_j)}, \qquad B_2 = \frac{\int_0^{c_i} \phi_j(x)dx}{\phi_j(c_j)},$$

and for $j = 1, 2, \ldots, p$, ϕ_j is given by

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

The resulting methods given in [29], [116], [44] and [46] were based on the premise that the abscissae should be equally spaced in the interval $[0, 1]$, including the boundary points. It was shown by Butcher, Chartier and Jackiewicz in [37] how to transform DIMSIMs into a Nordsieck formulation suitable for variable stepsize. Using the methods described above variable stepsize, variable order codes were developed in [38] and [115] which showed that these methods perform well against well known traditional solvers.

CHAPTER 3

Practical general linear methods

There are several reasons why general linear methods are not more widespread. Constructing families of methods rather than isolated examples has proved very difficult. Also once methods are constructed problems in the implementation often occur. To overcome these problems it is necessary to impose certain simplifying assumptions on the methods.

3.1 Framework

The order conditions of general linear methods are very complicated. To have any chance of finding practical methods certain simplifying assumptions must be made. There are two main simplifying assumptions which will be required. These follow from discussions in the previous chapter.

The first simplifying assumption is that the stage values are required to have a specific order q. There are two main choices for q, they are p and $p-1$ where p is the order of the method. Other choices are not suitable for stiff problems as order reduction occurs. While stage order $q = p-1$ allows more freedom in the design of practical methods, there are several reasons why $q = p$ is preferred. For example simple error estimators and interpolators become readily available. It also makes finding good estimates, used to decide whether to increase the order, easier. These will be further discussed in Chapter 4. Therefore, the stage values satisfy

$$Y_i^{[n]} = y(x_{n-1} + c_i h) + O(h^{p+1}), \qquad i = 1, 2, \ldots, s.$$

The second assumption is based upon requiring the quantities passed from step to step to have a simple form. One of the problems which can occur in the implementation of general linear methods is changing the stepsize. To overcome this difficulty in linear multistep methods Nordsieck vectors were proposed by Nordsieck [102] for the use with Adams methods. Later their use was promoted and used with advantage in the code DIFSUB by Gear [70]. Nordsieck vectors have a similar role for general linear methods. If there are $p+1$ independent components of $y^{[n]}$ then this choice does not lose any generality since a simple change of basis could be found to bring the method into this form. The quantities passed from step to step have the special form

$$y_i^{[n]} = h^{i-1} y^{(i-1)}(x_n) + O(h^{p+1}), \qquad i = 1, 2, \ldots, p+1.$$

Usually when methods are represented in Nordsieck form a scale factor $1/(i-1)!$ is included. This factor has been removed because it makes some of the expressions slightly simpler. These can be restored before implementation of the methods if this is preferred.

3.2 Order conditions

From now on it will be assumed that $q = p$ and $r = p+1$, these choices allow much simplification in the order conditions. However, before finding the order conditions it is necessary to introduce some notation. Define the shifting matrices

$$J = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}, \qquad K = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix},$$

which can some times be more conveniently expressed as

$$J = [\ e_2 \quad e_3 \quad e_4 \quad \cdots \quad e_{p+1} \quad 0\ ], \qquad K = [\ 0 \quad e_1 \quad e_2 \quad \cdots \quad e_{p-1} \quad e_p\ ].$$

These matrices are used to shift rows and columns of a particular matrix. Given a matrix R, RJ consists of the last p columns of R with an augmented column of zeros and JR consists of a row of zeros followed by the first p rows of R. Also, RK consists of a column of zeros augmented with the first p columns of R and KR consists of the last p rows of R followed by a row of zeros.

For convenience, write "$\equiv$" to denote the equivalence relation between matrices such that two matrices are equivalent if and only if they "are the identical except for their first rows". If $Fe_1 = \lambda e_1$ then $D \equiv E$ implies $FD \equiv FE$. Moreover, for any G, $D \equiv E$ implies $DG \equiv EG$. Two useful properties of the shifting matrices are

$$Z \equiv zJZ, \tag{3.2.1}$$
$$KZ = zZ + O(z^{p+1}), \tag{3.2.2}$$

where z is a complex parameter and Z is a basis vector given by

$$Z = [\ 1 \quad z \quad z^2 \quad \cdots \quad z^p\ ]^T. \tag{3.2.3}$$

It is clear what $\exp(S)$ means when S is a scalar or a matrix. In the case when S is a vector it is convenient to mean that the exponential function is applied componentwise, that is to say

$$\exp(S) = \begin{bmatrix} \exp(S_1) \\ \exp(S_2) \\ \exp(S_3) \\ \vdots \\ \exp(S_n) \end{bmatrix}.$$

Satisfying the required order for the stages and output approximations, with respect to the incoming approximations, the order conditions can be represented by considering a step from x_{n-1} to x_n. The following result is equivalent to the one first given by Burrage using mappings from trees to real numbers in [15] and by Butcher [28] who used a complex variable.

Theorem 3.1 *A general linear method in Nordsieck form has order and stage order p if and only if*

$$\exp(cz) = zA\exp(cz) + UZ + O(z^{p+1}), \tag{3.2.4}$$
$$\exp(z)Z = zB\exp(cz) + VZ + O(z^{p+1}), \tag{3.2.5}$$

where the exp *function is applied componentwise to a vector.*

Proof: The stage values $Y^{[n]}$, the output approximations $y^{[n]}$ and the scaled first derivatives $hf\big(Y^{[n]}\big)$ can be represented in terms of $y^{[n-1]}$ using Taylor series expansions about x_{n-1}. Therefore,

$$\begin{aligned} Y^{[n]} &= Cy^{[n-1]} + O(h^{p+1}), \\ y^{[n]} &= Ey^{[n-1]} + O(h^{p+1}), \\ hf\big(Y^{[n]}\big) &= CKy^{[n-1]} + O(h^{p+1}), \end{aligned}$$

where the Vandermonde matrix C is given by

$$C = \begin{bmatrix} e & c & \frac{c^2}{2!} & \cdots & \frac{c^{p-1}}{(p-1)!} & \frac{c^p}{p!} \end{bmatrix},$$

and the Toeplitz matrix E is given by

$$E = \begin{bmatrix} 1 & \frac{1}{1!} & \frac{1}{2!} & \cdots & \frac{1}{(p-2)!} & \frac{1}{(p-1)!} & \frac{1}{p!} \\ 0 & 1 & \frac{1}{1!} & \cdots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} & \frac{1}{(p-1)!} \\ 0 & 0 & 1 & \cdots & \frac{1}{(p-4)!} & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \frac{1}{1!} & \frac{1}{2!} \\ 0 & 0 & 0 & \cdots & 0 & 1 & \frac{1}{1!} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} = \exp(K). \tag{3.2.6}$$

Substituting the above expressions into the method (2.1.1) it follows that

$$\begin{aligned} Cy^{[n-1]} &= ACKy^{[n-1]} + Uy^{[n-1]} + O(h^{p+1}), \\ Ey^{[n-1]} &= BCKy^{[n-1]} + Vy^{[n-1]} + O(h^{p+1}). \end{aligned}$$

By making the substitution $h^k y^{(k)}(x_{n-1})$ equals z^k, the stage order and order conditions are now

$$CZ = ACKZ + UZ + O(z^{p+1}), \tag{3.2.7}$$
$$EZ = BCKZ + VZ + O(z^{p+1}). \tag{3.2.8}$$

This substitution is made only as a means of simplification, that is it is more convenient to consider z^k than $h^k y^{(k)}(x_{n-1})$. Using the relation (3.2.2), the stage order and order conditions now become

$$\begin{aligned} CZ &= zACZ + UZ + O(z^{p+1}), \\ EZ &= zBCZ + VZ + O(z^{p+1}). \end{aligned}$$

The result now follows by substituting

$$\begin{aligned} EZ &= \exp(z)Z + O(z^{p+1}), \\ CZ &= \exp(cz) + O(z^{p+1}), \end{aligned}$$

into the above stage order and order conditions. □

A direct consequence of this proof is that the matrices U and V are completely defined by A and B respectively, and the abscissae vector c. To see this, equate coefficients of $z^0, z^1, \ldots, z^p$, in (3.2.7) and (3.2.8) which leads to

$$U = C - ACK, \tag{3.2.9}$$

$$V = E - BCK. \tag{3.2.10}$$

Therefore, if a method has stage order and order p then the stage order and order conditions are satisfied by requiring the U and V matrices to have the above form. This means the A and B matrices are then available for satisfying any special properties which the method is required to have.

The conditions which guarantee a method has stage order and order p using the Taylor series expansions are (3.2.9) and (3.2.10) respectively. The conditions which guarantee a method has stage order and order p using mappings from trees to real numbers are (2.10.3) and (2.10.4) respectively. Comparing these alternative approaches it can be seen that they yield identical conditions. From now on it is always possible to consider the stage order and order conditions given in Theorem 3.1.

An interesting consequence of the method having stage order equal to the order, is the principal eigenvalue and the corresponding eigenvector of the stability matrix $M(z)$ have special forms.

Lemma 3.2 *Given a general linear method in Nordsieck form with $p = q$, then the stability matrix $M(z)$ has an eigenvalue* $\exp(z) + O(z^{p+1})$ *and corresponding eigenvector* $Z + O(z^{p+1})$.

Proof: Rearrange (3.2.4) as follows

$$\exp(cz) = (I - zA)^{-1}UZ + O(z^{p+1}).$$

Now substituting this into (3.2.5) gives

$$\exp(z)Z = \big(V + zB(I - zA)^{-1}U\big)Z + O(z^{p+1}),$$

therefore,

$$\exp(z)Z = M(z)Z + z^{p+1}\phi(z),$$

for some vector valued polynomial $\phi(z)$. Let the "principal" eigenvalue of $M(z)$ (that is, the one which tends to 1 as $z \to 0$) be $\lambda(z)$ and let the corresponding eigenvector (scaled so that the first component is 1) be $v(z)$.

Suppose that $\exp(z) = \lambda(z) + \theta(z)z^m$, where $\theta(0) \neq 0$. Suppose also that $Z = v(z) + \psi(z)z^n$, where $\psi(0)$ is a non-zero vector. Note that the first component of $\psi(0)$ is zero. It will be shown that $m, n \geq p + 1$ by proving that (i) if $m < p + 1$ then $n \leq m$ and (ii) it is impossible that $n \leq m$ and that $n < p + 1$.

To prove (i),

$$\begin{aligned} z^n(M(z) - \exp(z)I)\psi(z) - z^m\theta(z)v(z) \\ &= (M(z) - \exp(z)I)(Z - v(z)) + (\lambda(z) - \exp(z))v(z) \\ &= (M(z) - \exp(z)I)Z - (M(z) - \lambda(z)I)v(z) \\ &= -z^{p+1}\phi(z). \end{aligned} \tag{3.2.11}$$

Compare the coefficients of z^m in the first components and the result follows.

To prove (ii), divide both sides of (3.2.11) by z^n and set $z = 0$. Now

$$(V - I)\psi(0) = 0, \qquad (V - I)\psi(0) = \theta(0)v(0),$$

depending on whether $m > n$ or $m = n$. Pick out the last p components of this result and note that, because $v(0) = e_1$, the conclusion in each case is that

$$(\dot{V} - I)\dot{\psi}(0) = 0,$$

where $\dot{V}$ denotes the $p \times p$ matrix formed by deleting the first row and column of V; also $\dot{\psi}(0)$ denotes the last p elements of $\psi(0)$. Because $\dot{V} - I$ is non-singular, this implies that $\dot{\psi}(0) = 0$. However, this is impossible because $\psi(0) \neq 0$ and the first component of ψ is zero. □

Before attempting to find practical general linear methods, a further two assumptions are required. The simplifying assumptions described at the beginning of this chapter are almost essential requirements for locating practical general linear methods. The following requirements are by no means essential, however it is believed that finding practical methods is made easier with these requirements.

It is required that the general linear methods have Runge-Kutta stability, see Definition 2.20. It could be argued that this requirement is very restrictive. Since there is a tremendous amount of freedom available in the methods with stage order and order equal to p, that is all of the A and B matrices, attempting to obtain Runge-Kutta stability is not an unrealistic aim. Also locating explicit methods with relatively large stability regions is an extremely complicated task. Finding implicit methods is not as difficult as A-stability or L-stability is as much as could be hoped for. That is, it is much easier to obtain conditions which guarantee a method is L-stable than it is to obtain conditions which guarantee the explicit method has a large region of absolute stability. At this stage it may be possible to find low order explicit methods with minimised error coefficients and relatively large stability regions but as the order increases this becomes only possible with sophisticated numerical searches. Only isolated examples could be found in this way, rather than a class of methods.

A result of requiring explicit general linear methods to have Runge-Kutta stability is the number of stages must be at least p. This is a direct consequence of the form of the stability function (2.7.1), since for explicit methods $p_0(z) = 1$ and $p_1(z) = R(z) + O(z^{p+1})$. The obvious choice to consider first is when $s = p$. The first general linear methods derived with Runge-Kutta stability and $s = p$ were the DIMSIMs. These methods were introduced by Butcher in [28]. They were represented in such a way that there was the same number of free parameters as there was equations. This enabled explicit and implicit methods of order two and three to be constructed analytically using Maple or Mathematica. The abscissae vector c which can be freely chosen was evenly spaced within the interval $[0, 1]$. It is possible for Maple or Mathematica to represent the equations which need to be solved in the order four case, however these cannot be solved analytically. In [116] these equations were solved numerically by using sufficiently accurate initial approximations to the unknowns. For orders greater than four it is impossible for even Maple or Mathematica to express these equations let alone solve them. This led Butcher and Jackiewicz [44] and then Butcher, Jackiewicz and Mittelmann [46] for ways of finding explicit and implicit methods for orders up to eight using very sophisticated numerical searches. In [37] Butcher, Chartier and Jackiewicz showed a more satisfactory approach to varying the stepsize than the previous approach used in [43]. The idea was to recast the methods so that the incoming and outgoing data approximated Nordsieck vectors at the appropriate grid point. This approach requires an extra approximation to be artificially created so that there are $p + 1$ incoming and outgoing approximations. The methods in Nordsieck form were then used in a variable stepsize, variable order implementation in [38] and [115]. The results were shown to be comparable to some well known implementations.

When $s > p$ and certain assumptions about the method are made the nonlinear equations can be solved using only linear operations. This makes it possible to find methods with: minimised higher order error coefficients; coefficients with small magnitudes; certain properties for certain problems. In this thesis methods will be restricted to the case when $s = p + 1$. Methods with $s = p + 1$ provide sufficient freedom to obtain good methods with many advantages over the traditional methods. The relative advantages of methods with $s > p + 1$ will investigated in future work.

3.3 Inherent Runge-Kutta stability

The above assumptions on the method restrict the interest to general linear methods with Runge-Kutta stability. It is a complicated task to determine the conditions on the method in order to ensure Runge-Kutta stability in its most general sense. However, it is possible to find interrelations between the matrices which ensure the method has Runge-Kutta stability. This leads to a condition known as inherent Runge-Kutta stability (IRKS) which is sufficient but not necessary to ensure the general linear method has Runge-Kutta stability. The IRKS methods derived in this section builds on the work from the papers [32], [50] and [117] and are similar in nature but in more generality to the methods derived in [51]. The following definition will formalise this.

Definition 3.3 *A general linear method satisfying $Ve_1 = e_1$ has inherent Runge-Kutta stability if*

$$BA \equiv XB, \tag{3.3.1}$$
$$BU \equiv XV - VX, \tag{3.3.2}$$

for some matrix X and

$$\det(wI - V) = w^p(w - 1).$$

The principal consequence of a method having IRKS is given in the following theorem.

Theorem 3.4 *If a method has IRKS then its stability matrix has the form*

$$\sigma\left(V + zB(I - zA)^{-1}U\right) = \{R(z), 0\}.$$

Proof: The stability relation (3.3.1) can be written as

$$\begin{aligned} B(I - zA) &\equiv (I - zX)B, \\ B &\equiv (I - zX)B(I - zA)^{-1}. \end{aligned} \tag{3.3.3}$$

Rather than considering the characteristic equation of $M(z)$ directly, consider a matrix related to $M(z)$ by similarity. Using (3.3.2) and (3.3.3) it follows that

$$\begin{aligned} (I - zX)M(z)(I - zX)^{-1} &= (V - zXV + z(I - zX)B(I - zA)^{-1}U)(I - zX)^{-1} \\ &\equiv (V - zXV + zBU)(I - zX)^{-1} \\ &\equiv (V - z(BU + VX) + zBU)(I - zX)^{-1} \\ &= V. \end{aligned} \tag{3.3.4}$$

Since $(I - zX)M(z)(I - zX)^{-1}$ is identical to V except for the first row, $M(z)$ inherits from V the property of having a single non-zero eigenvalue. Zero-stability means that $M(0) = V$. Since for a Runge-Kutta method $R(0) = 1$ this implies that $\det(wI - V) = w^p(w - 1)$. □

The previous proof makes no reference to the form of the matrix X. The following theorem shows that X does have an important structure.

Theorem 3.5 *Given a general linear method in Nordsieck form with $q = p$, then the most general matrix X satisfying*

$$\begin{aligned} BA &\equiv XB, \\ BU &\equiv XV - VX, \end{aligned}$$

is of doubly companion form

$$X = \begin{bmatrix} -\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_{p-1} & -\alpha_p & -\alpha_{p+1} - \beta_{p+1} \\ 1 & 0 & 0 & \cdots & 0 & 0 & -\beta_p \\ 0 & 1 & 0 & \cdots & 0 & 0 & -\beta_{p-1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & -\beta_3 \\ 0 & 0 & 0 & \cdots & 1 & 0 & -\beta_2 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -\beta_1 \end{bmatrix}. \tag{3.3.5}$$

Proof: Multiply (3.2.4) by zB and (3.2.5) by $I - zX$ and add, then

$$\begin{aligned} \exp(z)(I - zX)Z &= z^2(BA - XB)\exp(cz) + z(BU - XV)Z + VZ + O(z^{p+1}) \\ &= z^2(BA - XB)\exp(cz) + z(BU - XV + VX)Z + V(I - zX)Z + O(z^{p+1}) \\ &\equiv V(I - zX)Z + O(z^{p+1}). \end{aligned}$$

It follows that

$$(\exp(z)I - V)(I - zX)Z \equiv O(z^{p+1}).$$

The matrix $\exp(z)I - V$ is invertible except when $z = 0$. When $z = 0$ the matrix formed by removing the first row and column of $I - V$ is non singular. So the above equation can be reduced to

$$(I - zX)Z \equiv O(z^{p+1}).$$

From the relation (3.2.1) it follows that

$$(J - X)Z \equiv O(z^p),$$

therefore $J - X$ is zero except in the first row and last column. □

To initially realize this note that if X defined by (3.3.5) and Z defined by (3.2.3) then the doubly companion matrix is the most general matrix satisfying

$$zXZ \equiv Z + O(z^{p+1}). \tag{3.3.6}$$

When a step of the method is performed the stage derivatives are calculated and then used to compute an update of the Nordsieck vector. Similarly substituting (3.2.4) into (3.2.5) and then making use of (3.3.6), it follows that

$$\begin{aligned} \exp(z)Z &= z^2BA\exp(cz) + (zBU + V)Z + O(z^{p+1}) \\ &\equiv z^2BA\exp(cz) + z(BU + VX)Z + O(z^{p+1}), \end{aligned} \tag{3.3.7}$$

where $Ve_1 = e_1$. Now multiplying (3.2.5) on the left by zX, and using equation (3.3.6) then

$$\begin{aligned} z^2XB\exp(cz) + zXVZ &\equiv z\exp(z)XZ + O(z^{p+1}), \\ &\equiv \exp(z)Z + O(z^{p+1}). \end{aligned} \tag{3.3.8}$$

Comparing (3.3.7) and (3.3.8) yields

$$\begin{aligned} BA &\equiv XB, \\ BU &\equiv XV - VX. \end{aligned}$$

Doubly companion matrices were first introduced in the study of Effective order Singly Implicit Runge-Kutta (ESIRK) methods [36]. In the following section several useful properties of doubly companion matrices will be given.

3.4 Doubly companion matrices

In this section a review of the properties of the doubly companion matrices X given by

$$X = \begin{bmatrix} -\alpha_1 & -\alpha_2 & -\alpha_3 & \cdots & -\alpha_{p-1} & -\alpha_p & -\alpha_{p+1} - \beta_{p+1} \\ 1 & 0 & 0 & \cdots & 0 & 0 & -\beta_p \\ 0 & 1 & 0 & \cdots & 0 & 0 & -\beta_{p-1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & -\beta_3 \\ 0 & 0 & 0 & \cdots & 1 & 0 & -\beta_2 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -\beta_1 \end{bmatrix},$$

is included along with several extensions useful for a canonical representation of methods with IRKS.

Consider the set Π of polynomials of degree $p+1$ which map 0 to 1. For example

$$\alpha(w) = 1 + \alpha_1 w + \alpha_2 w^2 + \cdots + \alpha_{p+1} w^{p+1}, \tag{3.4.1}$$
$$\beta(w) = 1 + \beta_1 w + \beta_2 w^2 + \cdots + \beta_{p+1} w^{p+1}. \tag{3.4.2}$$

The set Π becomes a group if

- 1 is defined to be the identity in Π.
- $\gamma = \alpha\beta$ is defined to be the composition in Π so that $\gamma(w) = \alpha(w)\beta(w) + O(w^{p+2})$.
- α^{-1} is defined to be the inverse in Π so that $\alpha^{-1}(w)\alpha(w) = 1 + O(w^{p+2})$.

This group becomes especially useful when w is replaced by the nilpotent matrix K, and the polynomials $\alpha(w)$ and $\beta(w)$ become Toeplitz matrices $\alpha(K)$ and $\beta(K)$.

Throughout this thesis $X(\alpha, \beta)$ will denote the $(p+1) \times (p+1)$ matrix given by (3.3.5), where the elements in the first row and last column are the coefficients of $\alpha, \beta \in \Pi$ given by (3.4.1) and (3.4.2). When there is no possibility of confusion X will be written in place of $X(\alpha, \beta)$.

First consider the characteristic polynomial of X given by (3.3.5). In the special cases $\beta_1 = \beta_2 = \cdots = \beta_{p+1} = 0$ or $\alpha_1 = \alpha_2 = \cdots = \alpha_{p+1} = 0$, the characteristic polynomial is given by

$$\begin{aligned} \det\big(wI - X(\alpha, 1)\big) &= w^{p+1} + \alpha_1 w^p + \alpha_2 w^{p-1} + \cdots + \alpha_{p+1}, \\ \det\big(wI - X(1, \beta)\big) &= w^{p+1} + \beta_1 w^p + \beta_2 w^{p-1} + \cdots + \beta_{p+1}, \end{aligned}$$

respectively.

To find the characteristic polynomial in the more general case, let x be an eigenvalue of X and represent the corresponding column eigenvector $\phi(x)$ and row eigenvector $\chi(x)^T$ as

$$\begin{aligned} \phi(x) &= [\ B_p(x) \quad B_{p-1}(x) \quad \cdots \quad B_1(x) \quad 1\ ]^T, \\ \chi(x)^T &= [\ 1 \quad A_1(x) \quad \cdots \quad A_{p-1}(x) \quad A_p(x)\ ]. \end{aligned}$$

The components of $\phi(x)$ and $\chi(x)^T$ can be found by the recurrence relations

$$\begin{aligned} B_k(x) &= xB_{k-1}(x) + \beta_k, \qquad & k &= 1, 2, \ldots, p, \\ A_k(x) &= xA_{k-1}(x) + \alpha_k, \qquad & k &= 1, 2, \ldots, p, \end{aligned}$$

where $B_0(x) = A_0(x) = 1$. Expanding these recurrence relations gives

$$\begin{aligned} B_k(x) &= x^k + \sum_{j=1}^{k} \beta_j x^{k-j}, \qquad & k &= 0, 1, 2, \ldots, p, \\ A_k(x) &= x^k + \sum_{j=1}^{k} \alpha_j x^{k-j}, \qquad & k &= 0, 1, 2, \ldots, p. \end{aligned}$$

Define $B_{p+1}(x)$ by extending the equations to hold for $p+1$, that is

$$\begin{aligned} B_{p+1}(x) &= x^{p+1} + \sum_{j=1}^{p+1} \beta_j x^{p+1-j}, \\ A_{p+1}(x) &= x^{p+1} + \sum_{j=1}^{p+1} \alpha_j x^{p+1-j}. \end{aligned}$$

When $p = 5$, for example, $\phi(x)$ is given by

$$\phi(x) = \begin{bmatrix} x^5 + \beta_1 x^4 + \beta_2 x^3 + \beta_3 x^2 + \beta_4 x + \beta_5 \\ x^4 + \beta_1 x^3 + \beta_2 x^2 + \beta_3 x + \beta_4 \\ x^3 + \beta_1 x^2 + \beta_2 x + \beta_3 \\ x^2 + \beta_1 x + \beta_2 \\ x + \beta_1 \\ 1 \end{bmatrix}.$$

The first component of $X\phi(x) = x\phi(x)$ is

$$-\sum_{k=1}^{p} \alpha_k B_{p+1-k}(x) - \alpha_{p+1} - \beta_{p+1} = xB_p(x),$$

since

$$B_{p+1}(x) = xB_p(x) + \beta_{p+1}$$

it follows that

$$B_{p+1}(x) + \sum_{k=1}^{p+1} \alpha_k B_{p+1-k}(x) = 0. \tag{3.4.3}$$

Since x is an eigenvalue of X and (3.4.3) is a polynomial of degree $p+1$ which equals zero, then (3.4.3) is the characteristic polynomial of X.

As a brief aside, consider equations with both positive and negative degree

$$Q_{n,m}(x) = q_{-n}x^{-n} + q_{-n+1}x^{-n+1} + \cdots + q_{-1}x^{-1} + q_0x^0 + q_1x + \cdots + q_{m-1}x^{m-1} + q_mx^m.$$

Then define $Q_{n,m}(x) + O(x^{-1})$ to be the polynomial

$$Q_{n,m}(x) + O(x^{-1}) = q_0x^0 + q_1x + \cdots + q_{m-1}x^{m-1} + q_mx^m.$$

This is used in the following lemma.

Lemma 3.6 *Let $P(w) = \det(wI - X)$ denote the characteristic polynomial of X. Then $P(w)$ consists of those terms with non-negative degree in the expansion of the product*

$$w^{-(p+1)}\left(\sum_{k=0}^{p+1}\alpha_k w^{p+1-k}\right)\left(\sum_{j=0}^{p+1}\beta_j w^{p+1-j}\right). \tag{3.4.4}$$

Proof: The characteristic polynomial (3.4.4) is equivalent to

$$\begin{aligned}
w^{-(p+1)}\left(\sum_{k=0}^{p+1}\alpha_k w^{p+1-k}\right)\left(\sum_{j=0}^{p+1}\beta_j w^{p+1-j}\right) &= \left(1 + \sum_{k=1}^{p+1}\alpha_k w^{-k}\right)\left(\sum_{j=0}^{p+1}\beta_j w^{p+1-j}\right)\\
&= B_{p+1}(w) + \sum_{k=1}^{p+1}\alpha_k w^{-k}\sum_{j=0}^{p+1}\beta_j w^{p+1-j}\\
&= B_{p+1}(w) + \sum_{k=1}^{p+1}\alpha_k \sum_{j=0}^{p+1-k}\beta_j w^{p+1-j-k} + O(w^{-1})\\
&= B_{p+1}(w) + \sum_{k=1}^{p+1}\alpha_k B_{p+1-k}(w) + O(w^{-1}),
\end{aligned}$$

which is the same polynomial given in (3.4.3). □

The characteristic polynomial (3.4.4) can alternatively be written as

$$P(w) = \det(I - wX) = \alpha(w)\beta(w) + O(w^{p+2}) = (\alpha\beta)(w).$$

In applications of doubly companion matrices which are of interest here the values of α_k, $k = 1, 2, \ldots, p+1$, are determined so that the eigenvalues of X have specified values; this means

$$\alpha(w) = P(w)\beta(w)^{-1} + O(w^{p+2}) = (P\beta^{-1})(w). \tag{3.4.5}$$

The eigenvalues of X and the β_k, $k = 1, 2, \ldots, p+1$, are free parameters chosen to obtain specific properties of the method.

In deriving methods with IRKS it is necessary to decompose X. This decomposition is used to transform methods into a suitable form so that the IRKS conditions can be satisfied using only linear operations. It is possible to decompose X so that any of the eigenvalues can have multiplicity greater than one. In order to obtain a consistent decomposition the Jordan canonical form is not used in this general case. In this thesis however only inherent Runge-Kutta stable methods where the matrix A has a one point spectrum will be derived. It will be shown that this is equivalent to the doubly companion matrix having a one point spectrum. In this case the decomposition reduces to the Jordan canonical form. The more general case when the eigenvalues of A can be freely chosen will be considered in future work.

So now consider the situation where all zeros of the characteristic polynomial of X are identical. Denote the resulting eigenvalue with multiplicity $p+1$ as λ. The reason this case is of interest is that it will directly relate to finding methods where $\sigma(A)=\{\lambda\}$. It is well known that methods of this type are computationally more efficient than the general case when the eigenvalues are not identical. When the eigenvalues are all identical, generalised eigenvectors are required with the properties given in the following lemma see also [36].

Lemma 3.7 *Given the column eigenvector $\phi(\lambda)$ and row eigenvector $\chi(\lambda)^T$, the doubly companion matrix X has characteristic polynomial*

$$P(w)=(w-\lambda)^{p+1},$$

if X can be decomposed as follows

$$X=\Psi(J+\lambda I)\Psi^{-1}, \tag{3.4.6}$$

where Ψ and Ψ^{-1} are the unit upper triangular matrices

$$\Psi=\left[\begin{array}{cccccc} \frac{1}{p!}\phi^{(p)}(\lambda) & \frac{1}{(p-1)!}\phi^{(p-1)}(\lambda) & \cdots & \phi'(\lambda) & \phi(\lambda) \end{array}\right],$$

$$\Psi^{-1}=\begin{bmatrix} \chi(\lambda)^T \\ \chi'(\lambda)^T \\ \vdots \\ \frac{1}{(p-1)!}\chi^{(p-1)}(\lambda)^T \\ \frac{1}{p!}\chi^{(p)}(\lambda)^T \end{bmatrix}.$$

Proof: Given that λ is an eigenvalue of multiplicity $p+1$ then $X\phi(\lambda)=\lambda\phi(\lambda)$ can be differentiated up to p times. Differentiating k times leads to

$$X\phi^{(k)}(\lambda)=\lambda\phi^{(k)}(\lambda)+k\phi^{(k-1)}(\lambda), \qquad k=1,2,\ldots,p.$$

Since column $p+1-k$ of Ψ satisfies

$$k!\Psi_k=\phi^{(k)}(\lambda), \qquad k=1,2,\ldots,p,$$

it follows that

$$X\Psi_k=\lambda\Psi_k+\Psi_{k-1}, \qquad k=1,2,\ldots,p.$$

Note that the result for Ψ^{-1} can be proved in a similar way. □

There are two slight variants of the nilpotent matrix K that will be used in the representation of the matrix X given by (3.3.5), when X has a one-point spectrum $\sigma(X)=\{\lambda\}$. These are

$$K^+=\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 2 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & p-1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & p \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}, \qquad K^-=\begin{bmatrix} 0 & p & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & p-1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}.$$

The matrices Ψ and Ψ^{-1} can be decomposed into the product of two matrices as shown in the following corollary.

Corollary 3.8 *The matrices* Ψ *and* Ψ^{-1} *can be factorised in the form*

$$\Psi = \beta(K)\exp(\lambda K^-), \qquad \Psi^{-1} = \exp(\lambda K^+)\alpha(K). \tag{3.4.7}$$

Proof: Consider first the formula for Ψ. Let $\Psi(x)$ denote the matrix

$$\Psi(\lambda) = \left[\ \tfrac{1}{p!}\phi^{(p)}(\lambda) \quad \tfrac{1}{(p-1)!}\phi^{(p-1)}(\lambda) \quad \cdots \quad \phi'(\lambda) \quad \phi(\lambda)\ \right]. \tag{3.4.8}$$

It is shown that $\Psi(\lambda) = \beta(K)\exp(xK^-)$ by noting that $\Psi(0) = \beta(K)$ and that $\Phi(\lambda)$ satisfies the differential equation $\Psi'(\lambda) = \Psi(\lambda)K^-$, because

$$\begin{aligned}
\Psi'(\lambda) &= \left[\ \tfrac{1}{p!}\phi^{(p+1)}(\lambda) \quad \tfrac{1}{(p-1)!}\phi^{(p)}(\lambda) \quad \cdots \quad \phi''(\lambda) \quad \phi'(\lambda)\ \right],\\
&= \left[\ 0 \quad \tfrac{1}{(p-1)!}\phi^{(p)}(\lambda) \quad \cdots \quad \phi''(\lambda) \quad \phi'(\lambda)\ \right],\\
&= \left[\ \tfrac{1}{p!}\phi^{(p)}(\lambda) \quad \tfrac{1}{(p-1)!}\phi^{(p-1)}(\lambda) \quad \cdots \quad \phi'(\lambda) \quad \phi(\lambda)\ \right]K^-,\\
&= \Phi(\lambda)K^-.
\end{aligned}$$

It then follows that $\Psi = \Psi(\lambda) = \beta(K)\exp(\lambda K^-)$. The result for Ψ^{-1} can be proved in a similar way. □

It is sometimes more appropriate to consider Ψ^{-1} as

$$\Psi^{-1} = \exp(\lambda K^-)^{-1}\beta(K)^{-1}.$$

This avoids the need for the the matrix $\alpha(K)$ leaving everything in terms of only $\beta(K)$. The matrix $\exp(\lambda K^-)^{-1}$ has the special structure

$$\exp(\lambda K^-)^{-1} = \exp(-\lambda K^-).$$

This can be seen by representing the matrices using Taylor series, that is

$$\begin{aligned}
\exp(\lambda K^-)\exp(-\lambda K^-) &= \sum_{i=0}^{p}\frac{(\lambda K^-)^i}{i!}\sum_{j=0}^{p}\frac{(-\lambda K^-)^j}{j!}\\
&= \sum_{i=0}^{p}\sum_{j=0}^{p}\frac{(-1)^j\lambda^{i+j}(K^-)^{i+j}}{i!j!}\\
&= \sum_{l=0}^{p}\sum_{j=0}^{l}\frac{(-1)^j}{j!(l-j)!}\lambda^l(K^-)^l.
\end{aligned}$$

For $l > 0$,

$$\sum_{j=0}^{l}\frac{(-1)^j}{j!(l-j)!} = 0,$$

and the result follows. Therefore, Ψ^{-1} can be alternatively expressed as

$$\Psi^{-1} = \exp(-\lambda K^-)\beta(K)^{-1}. \tag{3.4.9}$$

To construct the required doubly companion matrix first choose the eigenvalue of X and the coefficients β_k, $k = 1, 2, \ldots, p$. Now the α_k, $k = 1, 2, \ldots, p+1$ are found from (3.4.5). The matrices Ψ and Ψ^{-1} are computed using (3.4.7) and (3.4.9). The doubly companion matrix is then found from (3.4.6). Note that β_{p+1} has no effect because α_{p+1} is always chosen to ensure X has the required eigenvalue. It is without loss of generality that $\beta_{p+1} = 0$ is assumed.

The following three results deduce a certain relationship between Toeplitz matrices and doubly companion matrices. The first result is used to consider the special case of (3.4.6) when $\lambda = 0$. This result then leads to an alternative decomposition of the doubly companion matrix $X(\alpha,\beta)$ when $\lambda > 0$, where the doubly companion matrix is represented by a certain similarity transformation of a companion matrix.

Theorem 3.9 *Given two polynomials $\alpha(w)$ and $\beta(w)$, where $\alpha(w)\beta(w) = 1$ and $\lambda = 0$, then*

$$\beta(K)J\alpha(K) = X(\alpha,\beta).$$

Proof: This is a special case of Lemma 3.7, but an alternative proof is included. First compute K^iJK^j, which is

$$K^iJK^j = \begin{cases} J, & i = j = 0, \\ K^{j-1} - e_1e_j^T, & i = 0, j > 0, \\ K^{i-1} - e_{p+2-i}e_{p+1}^T, & i > 0, j = 0, \\ K^{i+j-1}, & i, j > 0. \end{cases}$$

Given that $\alpha(w)$ and $\beta(w)$ are defined by (3.4.1) and (3.4.2) respectively, expanding $\beta(K)J\alpha(K)$ in powers of K, yields

$$\begin{aligned}
&\left(I + \sum_{i=1}^{p} \beta_i K^i\right) J \left(I + \sum_{j=1}^{p} \alpha_j K^j\right) \\
&\quad = J + \sum_{i=1}^{p} \beta_i K^i J + \sum_{j=1}^{p} \alpha_j J K^j + \sum_{i=1}^{p}\sum_{j=1}^{p} \beta_i\alpha_j K^iJK^j \\
&\quad = J + \sum_{i=1}^{p} \beta_i \big(K^{i-1} - e_{p+2-i}e_{p+1}^T\big) + \sum_{j=1}^{p} \alpha_j \big(K^{j-1} - e_1e_j^T\big) + \sum_{i=1}^{p}\sum_{j=1}^{p} \beta_i\alpha_j K^{i+j-1} \\
&\quad = J - \sum_{i=1}^{p+1} \beta_i e_{p+2-i}e_{p+1}^T - \sum_{j=1}^{p+1} \alpha_j e_1 e_j^T + \sum_{i=1}^{p+1} \beta_i K^{i-1} + \sum_{j=1}^{p+1} \alpha_j K^{j-1} + \sum_{i=1}^{p}\sum_{j=1}^{p} \beta_i\alpha_j K^{i+j-1} \\
&\quad = X(\alpha,\beta) + \sum_{k=1}^{p+1} \gamma_i K^{k-1},
\end{aligned}$$

where

$$\sum_{k=0}^{p+1} \gamma_k w^k = \alpha(w)\beta(w) + O(w^{p+2}) = 1 + O(w^{p+2})$$

so that $\gamma_1 = \gamma_2 = \cdots = \gamma_p = \gamma_{p+1} = 0$. □

It is useful to extend this result so that the polynomials $\alpha(w)$ or $\beta(w)$ could be multiplied by a further polynomial $\gamma(w)$.

Corollary 3.10 *If $\alpha(w)$, $\beta(w)$ and $\gamma(w)$, are such that $\alpha\beta\gamma = 1$, then*

$$\begin{aligned} X(\alpha\gamma,\beta) &= X(\alpha,\beta)\gamma(K), \\ X(\alpha,\gamma\beta) &= \gamma(K)X(\alpha,\beta). \end{aligned}$$

Proof: This following directly from the proof of Theorem 3.9 since $\alpha\gamma\beta = \alpha\beta\gamma = 1$. □

This result from Theorem 3.9 can be extended so that it is applicable for $\lambda > 0$ by modifying the role of $\alpha(w)$. Note that 1 represents the polynomial with constant value 1. Hence, $X(\alpha, 1)$ or $X(1, \beta)$ are companion matrices. In applications of the following lemma, $P(w)$ would be chosen as $(1 - \lambda w)^{p+1}$, so that $X(P\beta^{-1}, \beta)$ has a one point spectrum $\sigma(X) = \{\lambda\}$.

Lemma 3.11 *Given two polynomials $P(w)$ and $\beta(w)$, then*

$$\beta(K)X(P, 1)\beta(K)^{-1} = X(P\beta^{-1}, \beta).$$

Proof: For convenience of notation let $\beta^{-1} = \bar{\beta}$, where $\bar{\beta}$ has coefficients $\bar{\beta}_1, \bar{\beta}_2, \dots, \bar{\beta}_{p+1}$. Using the relations

$$K^i e_1 = 0, \qquad e_j^T K^k = e_{j+k}^T,$$

it follows by expanding $\beta(K)X(P, 1)\beta(K)^{-1}$ in powers of K, that

$$\begin{aligned}
&\left(I + \sum_{i=1}^{p} \beta_i K^i\right)\left(J - \sum_{j=1}^{p+1} P_j e_1 e_j^T\right)\left(I + \sum_{k=1}^{p} \bar{\beta}_k K^k\right)\\
&= J + \sum_{k=1}^{p} \bar{\beta}_k J K^k - \sum_{j=1}^{p+1} P_j e_1 e_j^T - \sum_{j=1}^{p+1}\sum_{k=1}^{p} P_j \bar{\beta}_k e_1 e_{j+k}^T + \sum_{i=1}^{p} \beta_i K^i J + \sum_{i=1}^{p}\sum_{k=1}^{p} \beta_i \bar{\beta}_k K^i J K^k\\
&= J - \sum_{k=1}^{p+1} \bar{\beta}_k e_1 e_k^T - \sum_{j=1}^{p+1} P_j e_1 e_j^T - \sum_{j=1}^{p+1}\sum_{k=1}^{p} P_j \bar{\beta}_k e_1 e_{j+k}^T - \sum_{i=1}^{p+1} \beta_i e_{p+2-i} e_{p+1}^T\\
&= J - \left(\sum_{j=0}^{p+1} P_j e_1 e_j^T\right)\left(\sum_{k=0}^{p+1} \bar{\beta}_k e_1 e_k^T\right) - \sum_{i=1}^{p+1} \beta_i e_{p+2-i} e_{p+1}^T\\
&= X(P\beta^{-1}, \beta).
\end{aligned}$$

□

This result again shows that the coefficient β_{p+1} is redundant, in the sense that however it is chosen α_{p+1} will be chosen to obtain the required eigenvalues of X. Therefore, only the coefficients $\beta_1, \beta_2, \dots, \beta_p$, are free parameters.

3.5 Transformations between method arrays

In order to find methods with the IRKS property it is required to transform the method into an appropriate form which enables us to be sure all methods with this property can be found. In this section the transformed method will be derived. Before considering the required transformation, two results which are crucial for the derivation of methods are given. The first considers a special connection between lower triangular matrices.

Lemma 3.12 *Let L denote a strictly lower triangular $n \times n$ matrix, with $n = p + 1$, satisfying*

$$ML \equiv JM,$$

then M is lower triangular.

Proof: For 1×1 matrices there is nothing to prove. Suppose the result has been proved for $n \times n$ matrices for $n = 2, 3, \dots, p$ so that to complete the proof by induction it is left to prove the result for $n = p + 1$.

Because $Le_{p+1} = 0$, it follows that

$$JMe_{p+1} \equiv MLe_{p+1} = 0,$$

and therefore, the first p components of Me_{p+1} are equal to zero. Write J, L and M as $(p+1) \times (p+1)$ partitioned matrices as follows

$$M = \begin{bmatrix} \dot{M} & 0 \\ m^T & \widehat{m} \end{bmatrix}, \qquad J = \begin{bmatrix} \dot{J} & 0 \\ e_p^T & 0 \end{bmatrix}, \qquad L = \begin{bmatrix} \dot{L} & 0 \\ l^T & 0 \end{bmatrix},$$

so that $ML \equiv JM$ implies that $\dot{M}\dot{L} \equiv \dot{J}\dot{M}$, which, by the induction hypothesis, implies that $\dot{M}$, and therefore M, is lower triangular. □

Let A be a full matrix with a one point spectrum. Therefore, it is possible to find a nonsingular transformation matrix W such that

$$\bar{A} = W^{-1}AW, \tag{3.5.1}$$

is diagonally implicit with the eigenvalues on the main diagonal. The most competitive case is likely to be when $W = I$, for which A is diagonally implicit. By allowing A to have this general structure rather than just the diagonally implicit case makes it possible to represent some well known methods as special cases. The second result shows the first IRKS condition can in fact be strengthened without any loss of generality.

Lemma 3.13 *Given the coefficients $\beta_1, \beta_2, \ldots, \beta_p$, the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_p, \alpha_{p+1}$, are chosen so that the characteristic polynomial of X is of the form $\det(wI - X) = (w - \lambda)^{p+1}$. Then $BA \equiv XB$ implies*

$$BA = XB. \tag{3.5.2}$$

Proof: Substitute $W\bar{A}W^{-1} = A$ into the first IRKS condition (3.3.1), which gives

$$BW\bar{A} \equiv XBW.$$

Use the similarity of X to Jordan canonical form given by

$$X = \Psi(J + \lambda I)\Psi^{-1},$$

so that $BW\bar{A} \equiv \Psi(J + \lambda I)\Psi^{-1}BW$. Rearrange this in the form

$$\Psi^{-1}BW(\bar{A} - \lambda I) \equiv J\Psi^{-1}BW. \tag{3.5.3}$$

Since $\bar{A} - \lambda I$ is strictly lower triangular it follows from Lemma 3.12 that $\Psi^{-1}BW$ is lower triangular and the two sides of (3.5.3) are therefore equal making the equation equivalent to (3.5.2). □

This result is used to find A given B is known. If B is invertible then $A = B^{-1}XB$. In the somewhat special case when B is not invertible but only one of the eigenvalues of B is zero, it will be shown that A is unique. In the very special case when more than one of the eigenvalues of B is zero it will be shown that there is some freedom in the choice of the matrix A. Given that if A and B are known then U and V can be found from the stage order and order conditions, it is crucial to have a convenient way of finding B. The transformation is used to represent B in such a way that the remaining conditions for IRKS are satisfied. The required transformation needs to represent any original method in a certain form suitable for solving the remaining conditions. In order to do so the following three results are needed to understand the effect of the transformation.

First note that

$$\sum_{m=0}^{k} \binom{m+l-1}{m} = \binom{k+l}{k},$$

which can be shown by induction. The proofs of the following two lemmas were suggested by Welfert [114]. The following lemma gives an expression for the negative binomial expansion.

Lemma 3.14 *For $l \geq 1$ and $|w| < 1$, then*

$$\sum_{k=0}^{\infty} \binom{k+l-1}{k} w^k = \left(\sum_{k=0}^{\infty} w^k\right)^l = \frac{1}{(1-w)^l}.$$

Proof: The result is clearly true for $l = 1$. For $l > 1$ then

$$\begin{aligned}
\left(\sum_{k=0}^{\infty} w^k\right)^l &= \left(\sum_{m=0}^{\infty} w^m\right)^{l-1} \sum_{n=0}^{\infty} w^n \\
&= \sum_{m=0}^{\infty} \binom{m-1+l-1}{m} w^m \sum_{n=0}^{\infty} w^n \\
&= \sum_{k=0}^{\infty} \left(\sum_{m=0}^{k} \binom{m+l-2}{m}\right) w^k \\
&= \sum_{k=0}^{\infty} \binom{k+l-1}{k} w^k.
\end{aligned}$$

□

The following lemma develops a special Toeplitz matrix.

Lemma 3.15 *Let λ denote a real number, then for any matrix S satisfying $\rho(\lambda S) < 1$,*

$$\exp\left(S(I+\lambda S)^{-1}\right) = I + \sum_{i=1}^{\infty} N_i(\lambda) S^i, \tag{3.5.4}$$

where

$$N_i(\lambda) = \sum_{k=0}^{i-1} \binom{i-1}{k} \frac{(-\lambda)^k}{(i-k)!}.$$

Proof: Expanding (3.5.4) into Taylor series gives

$$\exp\left(S(I+\lambda S)^{-1}\right) = I + \sum_{l=1}^{\infty} \frac{1}{l!} S^l (I+\lambda S)^{-l}. \tag{3.5.5}$$

The negative binomial expansion from Lemma 3.14 is

$$(I+\lambda S)^{-l} = \sum_{k=0}^{\infty} \binom{k+l-1}{k} (-\lambda S)^k.$$

Substituting the negative binomial expansion into (3.5.5) gives

$$\begin{aligned}\exp\left(S(I+\lambda S)^{-1}\right) &= I + \sum_{l=1}^{\infty}\frac{1}{l!}S^l\left(\sum_{k=0}^{\infty}\binom{k+l-1}{k}(-\lambda S)^k\right)\\ &= I + \sum_{l=1}^{\infty}\sum_{k=0}^{\infty}\binom{k+l-1}{k}\frac{(-\lambda)^k}{l!}S^{l+k}.\end{aligned}$$

Making the substitution $l+k=i$ it follows that

$$\exp\left(S(I+\lambda S)^{-1}\right) = I + \sum_{i=1}^{\infty}\left(\sum_{k=0}^{i-1}\binom{i-1}{k}\frac{(-\lambda)^k}{(i-k)!}\right)S^i.$$

□

Of special interest is the case where S is equal to the nilpotent matrix K. Here the matrix $\exp\left(K(I+\lambda K)^{-1}\right)$ is an upper triangular Toeplitz matrix.

This result can be extended slightly by introducing a scalar x such that

$$\exp\left(xS(I+\lambda S)^{-1}\right) = I + \sum_{i=1}^{\infty}\left(\sum_{k=0}^{i-1}\binom{i-1}{k}\frac{(-\lambda)^k x^{i-k}}{(i-k)!}\right)S^i. \tag{3.5.6}$$

This is used only to show that the previous derivations of methods with the IRKS property are a special case of the derivations discussed in this thesis, see Section 3.12. The previous derivations mentioned are the implicit methods derived in [32], the explicit methods derived in [117] and the methods based on a special transformation which made it possible to construct explicit methods using the approach derived for implicit methods and vice versa, see [50]. To show this slightly more general result, write (3.5.6) in the form

$$\sum_{j=0}^{\infty}A_j\frac{x^j}{j!} = \sum_{j=0}^{\infty}B_j\frac{x^j}{j!}, \tag{3.5.7}$$

so that it is only necessary to prove that $A_j = B_j$, for $j = 0,1,2,\dots,p$. This will hold if

$$\sum_{j=0}^{\infty}A_j t^j = \sum_{j=0}^{\infty}B_j t^j,$$

for sufficiently small $|t|$. However, (3.5.7) holds because

$$\begin{aligned}\sum_{j=0}^{\infty}\left(tS(I+\lambda S)^{-1}\right)^j &= \left(I - tS(I+\lambda S)^{-1}\right)^{-1}\\ &= (I+\lambda S)(I+(\lambda-t)S)^{-1}\\ &= (I-(t-\lambda)S+tS)(I+(\lambda-t)S)^{-1}\\ &= I + tS(I-(t-\lambda)S)^{-1}\\ &= I + \sum_{j=1}^{\infty}\left(t(t-\lambda)^{j-1}\right)S^j\\ &= I + \sum_{j=1}^{\infty}\left(\sum_{k=0}^{j-1}\binom{j-1}{k}(-\lambda)^k t^{j-k}\right)S^j.\end{aligned}$$

Reinterpreting in terms of x gives the required result.

The following table gives $N_n(\lambda)$ for $n = 1, 2, \ldots, 6$.

n	$N_n(\lambda)$
1	1
2	$\frac{1}{2} - \lambda$
3	$\frac{1}{6} - \lambda + \lambda^2$
4	$\frac{1}{24} - \frac{1}{2}\lambda + \frac{3}{2}\lambda^2 - \lambda^3$
5	$\frac{1}{120} - \frac{1}{6}\lambda + \lambda^2 - 2\lambda^3 + \lambda^4$
6	$\frac{1}{720} - \frac{1}{24}\lambda + \frac{5}{12}\lambda^2 - \frac{5}{3}\lambda^3 + \frac{5}{2}\lambda^4 - \lambda^5$

Table 3.1: $N_n(\lambda)$ for orders up to six.

The second result considers a special similarity transformation of the Toeplitz matrix E, which results in the matrices described in the previous lemma. The similarity transformation is described as follows.

Lemma 3.16 *Let λ denote a real number and since K is nilpotent, it follows that*

$$\exp(-\lambda K^-)E\exp(\lambda K^-) = \exp\left(K(I+\lambda K)^{-1}\right).$$

Proof: First note that, for f a polynomial of degree p,

$$\frac{1}{x}\left(\frac{d}{dx}\right)^i f(x) = \left(\frac{d}{dx}\right)^i\left(\frac{1}{x}f(x)\right) + \frac{i}{x}\left(\frac{d}{dx}\right)^{i-1}\left(\frac{1}{x}f(x)\right). \tag{3.5.8}$$

This is a consequence of Liebnitz' rule for the n-fold derivative of the product of x and $x^{-1}f(x)$. It is easy to see that (3.5.8) is also true if $x^{-1}f(x)$ is interpreted as the polynomial consisting only of the terms involving non-negative powers of x.

Define the vectors

$$\xi = \begin{bmatrix} x^p & x^{p-1} & \cdots & x & 1 \end{bmatrix}^T,$$
$$a^T = \begin{bmatrix} 1 & a_1 & \cdots & a_{p-1} & a_p \end{bmatrix},$$

and now let $f(x) = a^T\xi$. Subject to the interpretation of the terms in (3.5.8) as having negative powers of x deleted, then

$$\frac{1}{x}\left(\frac{d}{dx}\right)^i f(x) = a^T(K^-)^iK\xi,$$
$$\left(\frac{d}{dx}\right)^i\left(\frac{1}{x}f(x)\right) = a^TK(K^-)^i\xi,$$
$$\frac{i}{x}\left(\frac{d}{dx}\right)^{i-1}\left(\frac{1}{x}f(x)\right) = ia^TK(K^-)^{i-1}K\xi.$$

From (3.5.8) it follows that

$$a^T\left((K^-)^iK - K(K^-)^i - iK(K^-)^{i-1}K\right)\xi = 0.$$

Since this holds for any value of x and any choice of the components of a, it follows that

$$(K^-)^i K = K(K^-)^i + iK(K^-)^{i-1}K.$$

Multiply by $\lambda^i/i!$ and sum for $i = 0, 1, \ldots$ and it follows that

$$\exp(\lambda K^-)K = K\exp(\lambda K^-)(I + \lambda K),$$

or, what is equivalent,

$$\exp(-\lambda K^-)K\exp(\lambda K^-) = K(I + \lambda K)^{-1}.$$

It follows that for $j = 0, 1, \ldots,$

$$\exp(-\lambda K^-)K^j\exp(\lambda K^-) = (K(I + \lambda K)^{-1})^j,$$

so that multiplying by $1/j!$ and summing over j the required result is obtained.

$$\exp(-\lambda K^-)E\exp(\lambda K^-) = \exp(K(I + \lambda K)^{-1}).$$

□

For ease of representation define the upper triangular Toeplitz matrix F as follows

$$F = \exp\left(K(I + \lambda K)^{-1}\right).$$

Now it is possible to discuss the required transformation. Denote by M the original method and by $\widetilde{M}$ the transformed method. In order to construct M, it is possible to construct $\widetilde{M}$ and then back transform to find M. Use the results from Lemma 3.7 to transform the method and the IRKS conditions. Substituting (3.4.6) and (3.5.1) into (3.5.2) yields

$$\begin{aligned} BW\bar{A}W^{-1} &= \Psi(J + \lambda I)\Psi^{-1}B, \\ \widetilde{B}\widetilde{A} &= J\widetilde{B}, \end{aligned} \tag{3.5.9}$$

where the transformed matrices $\widetilde{A}$ and $\widetilde{B}$ are

$$\begin{aligned} \widetilde{A} &= W^{-1}AW - \lambda I, \\ \widetilde{B} &= \Psi^{-1}BW, \end{aligned} \tag{3.5.10}$$

Therefore, $\widetilde{A}$ is strictly lower triangular and $\widetilde{B}$ is lower triangular. Substituting (3.4.6) into (3.3.2) and simplifying gives

$$\begin{aligned} BU &\equiv \Psi J\Psi^{-1}V - V\Psi J\Psi^{-1}, \\ \widetilde{B}\widetilde{U} &\equiv J\widetilde{V} - \widetilde{V}J, \end{aligned} \tag{3.5.11}$$

where the transformed matrices $\widetilde{U}$ and $\widetilde{V}$ are

$$\begin{aligned} \widetilde{U} &= W^{-1}U\Psi, \\ \widetilde{V} &= \Psi^{-1}V\Psi. \end{aligned}$$

Therefore pre-multiplying (3.2.9) by W^{-1} and post-multiplying by Ψ yields

$$\begin{aligned} \widetilde{U} &= W^{-1}C\Psi - W^{-1}ACK\Psi \\ &= W^{-1}C(I - \lambda K)\Psi - \widetilde{A}W^{-1}CK\Psi. \end{aligned} \tag{3.5.12}$$

Likewise, pre-multiplying (3.2.10) by Ψ^{-1}, post-multiply by Ψ, and noting that upper triangular Toeplitz matrices commute it follows that

$$\begin{aligned}\widetilde{V} &= \Psi^{-1}E\Psi - \Psi^{-1}BCK\Psi \\ &= \exp(-\lambda K^{-})E\exp(\lambda K^{-}) - \widetilde{B}W^{-1}CK\Psi \\ &= F - \widetilde{B}W^{-1}CK\Psi.\end{aligned} \tag{3.5.13}$$

Irrespective of whether A is a full matrix or a strictly lower triangular matrix, $\widetilde{A}$ is strictly lower triangular and $\widetilde{B}$ is lower triangular. What this means is that given W and Ψ, (which can be freely chosen when deriving the method) B is known once $\widetilde{B}$ is known. There are three conditions required to find $\widetilde{B}$. These are:

- The transformed IRKS condition (3.5.11) and thus (3.3.2).
- The spectrum property of $\widetilde{V}$ and thus V.
- The property which requires a certain error constant or L-stability.

The first condition will be further investigated in this section. The last condition will be analysed in the next section. The process of finding $\widetilde{B}$ will be constructed in Section 3.7.

To understand the first of these conditions substitute (3.5.12), (3.5.13) into (3.5.11). Noting that upper triangular Toeplitz matrices commute it turns out that

$$\begin{aligned}\widetilde{B}W^{-1}C(I-KX)\Psi &\equiv J\Psi^{-1}E\Psi - \Psi^{-1}E\Psi J \\ &= J\exp(-\lambda K^{-})E\exp(\lambda K^{-}) - \exp(-\lambda K^{-})E\exp(\lambda K^{-})J \\ &= JF - FJ.\end{aligned} \tag{3.5.14}$$

In order to appreciate (3.5.14) it is best to break the equation up into parts. First however the following lemma is useful for understanding the structure of the right hand side.

Lemma 3.17 *Given the polynomial $f(x) = 1 + a_1x + \cdots + a_{p-1}x^{p-1} + a_px^p$, it follows that $f(K)$ is an upper triangular Toeplitz matrix which satisfies*

$$Jf(K) - f(K)J = \begin{bmatrix} -a_1 & -a_2 & -a_3 & \cdots & -a_{p-1} & -a_p & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & a_p \\ 0 & 0 & 0 & \cdots & 0 & 0 & a_{p-1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & a_3 \\ 0 & 0 & 0 & \cdots & 0 & 0 & a_2 \\ 0 & 0 & 0 & \cdots & 0 & 0 & a_1 \end{bmatrix}.$$

Proof: Given that

$$JK^n - K^nJ = e_{p-n+2}e_{p+1}^T - e_1e_n^T,$$

multiply by a_n and sum for $n = 1, 2, \ldots, p$ and the result follows. $\square$

Since F is an upper triangular Toeplitz matrix from Lemma 3.17 the only non-zero entries on the right hand side are the first row and last column. The non-zero entries are given in Lemma 3.15, therefore,

$$JF - FJ = \begin{bmatrix} -N_1(\lambda) & -N_2(\lambda) & -N_3(\lambda) & \cdots & -N_{p-1}(\lambda) & -N_p(\lambda) & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & N_p(\lambda) \\ 0 & 0 & 0 & \cdots & 0 & 0 & N_{p-1}(\lambda) \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & N_3(\lambda) \\ 0 & 0 & 0 & \cdots & 0 & 0 & N_2(\lambda) \\ 0 & 0 & 0 & \cdots & 0 & 0 & N_1(\lambda) \end{bmatrix}.$$

The last part of the left hand side of (3.5.14) is given by

$$(I - KX)\Psi = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & \beta_p \\ 0 & 0 & 0 & \cdots & 0 & 0 & \beta_{p-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & \beta_{p-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \beta_2 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \beta_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}.$$

Now given that (3.5.9) will be satisfied, (3.5.11) will automatically be satisfied except for the last column. Therefore, (3.5.14) results in surprisingly only p extra conditions.

3.6 Transformations between stability functions

Turn attention now to the relationship between the stability matrices of the method and its transform, in order to determine the connection between their stability functions. The stability matrix of the original method assuming A is given by (3.5.1) and $\sigma(A) = \{\lambda\}$, is

$$\begin{aligned} M(z) &= V + zB\big(I - zW\bar{A}W^{-1}\big)^{-1}U \\ &= V + zBW\big(I - z\bar{A}\big)^{-1}W^{-1}U \\ &= V + \frac{z}{1-\lambda z}BW\left(\frac{1}{1-\lambda z}I - \frac{z}{1-\lambda z}\bar{A}\right)^{-1}W^{-1}U \\ &= V + \frac{z}{1-\lambda z}BW\left(I - \frac{z}{1-\lambda z}(\bar{A} - \lambda I)\right)^{-1}W^{-1}U. \end{aligned}$$

Given that $\widetilde{z}$ and $\widetilde{A}$ are defined as

$$\widetilde{z} = \frac{z}{1-\lambda z}, \qquad \widetilde{A} = \bar{A} - \lambda I,$$

the stability matrix is

$$M\left(\frac{\widetilde{z}}{1+\lambda\widetilde{z}}\right) = V + \widetilde{z}BW(I - \widetilde{z}\widetilde{A})^{-1}W^{-1}U.$$

This is related to $\widetilde{M}(\widetilde{z}\,)$ by a similarity.

$$\begin{aligned}\Psi^{-1}M\left(\frac{\widetilde{z}}{1+\lambda\widetilde{z}}\right)\Psi &= \Psi^{-1}V\Psi + \widetilde{z}\Psi^{-1}BW(I-\widetilde{z}\widetilde{A})^{-1}W^{-1}U\Psi\\ &= \widetilde{V} + \widetilde{z}\widetilde{B}(I-\widetilde{z}\widetilde{A})^{-1}\widetilde{U}\\ &= \widetilde{M}(\widetilde{z}\,).\end{aligned}$$

So there is a direct connection between the stability matrices of the original and transformed methods. In order to obtain a connection between the resulting stability functions it is first convenient to consider equivalent conditions to Definition 3.3 and Theorem 3.4 for inherent Runge-Kutta stability of the transformed method. These are as follows.

Definition 3.18 *A transformed general linear method satisfying $\widetilde{V}e_1 = e_1$ has IRKS if*

$$\begin{aligned}\widetilde{B}\widetilde{A} &= J\widetilde{B},\\ \widetilde{B}\widetilde{U} &\equiv J\widetilde{V} - \widetilde{V}J,\end{aligned}$$

where J is the shifting matrix and

$$\det\big(wI - \widetilde{V}\big) = w^p(w-1).$$

The main consequence of a transformed method having IRKS is given in the following theorem.

Theorem 3.19 *If a transformed method has IRKS then its stability matrix has the form*

$$\sigma\big(\widetilde{V} + \widetilde{z}\widetilde{B}(I-\widetilde{z}\widetilde{A})^{-1}\widetilde{U}\big) = \big\{\widetilde{R}(\widetilde{z}\,), 0\big\}.$$

Proof: This is proved in an identical way to Theorem 3.4. □

The above definition and theorem are mainly included to emphasise the connection between the original and transformed methods. The main aim of satisfying the conditions of Theorem 3.4 is satisfied by transforming the method as described above and satisfying the conditions of Theorem 3.19 and then transforming back to the original method.

An equivalent statement to (3.3.4), for the transformed method, is

$$(I-\widetilde{z}J)\widetilde{M}(\widetilde{z}\,)(I-\widetilde{z}J)^{-1} \equiv \widetilde{V}.$$

This means that the non-zero eigenvalue of $\widetilde{M}(\widetilde{z}\,)$, that is the transformed stability function $\widetilde{R}(\widetilde{z}\,)$, is given by

$$\begin{aligned}\widetilde{R}(\widetilde{z}\,) &= e_1^T(I-\widetilde{z}J)\widetilde{M}(\widetilde{z}\,)(I-\widetilde{z}J)^{-1}e_1\\ &= e_1^T\big(\widetilde{V} + \widetilde{z}\widetilde{B}(I-\widetilde{z}\widetilde{A})^{-1}\widetilde{U}\big)\widetilde{Z},\end{aligned}$$

where $\widetilde{Z}$ is the vector

$$\widetilde{Z} = [\ 1 \quad \widetilde{z} \quad \widetilde{z}^2 \quad \cdots \quad \widetilde{z}^p\ ]^T.$$

Because $\widetilde{A}$ is strictly lower triangular and $\widetilde{B}$ is related to $\widetilde{A}$ by the equation $\widetilde{B}\widetilde{A} = J\widetilde{B}$ it follows from Lemma 3.12 that $\widetilde{B}$ is lower triangular. The transformed stability function is then

$$\widetilde{R}(\widetilde{z}\,) = e_1^T\big(\widetilde{V} + \widetilde{z}\widetilde{B}\widetilde{U}\big)\widetilde{Z}.$$

If the original method is explicit, the freedom in the stability function is used to control the error constant and possibly the stability region. If the original method is implicit, the freedom in the numerator of the stability function is used to guarantee L-stability. These conditions on the original method must be interpreted as conditions on the transformed method. Consider the stability function of the original method, since the matrix A is diagonally implicit the stability function is the same as a diagonally implicit Runge-Kutta method, that is a rational approximation of the form

$$R(z) = \frac{N(z)}{(1-\lambda z)^{p+1}} = \exp(z) - \sigma z^{p+1} + O(z^{p+2}).$$

The stability function of the transformed method is therefore given by

$$\widetilde{R}(\widetilde{z}) = \widetilde{N}(\widetilde{z}) = \exp\left(\frac{\widetilde{z}}{1+\lambda\widetilde{z}}\right) - \sigma\widetilde{z}^{p+1} + O(\widetilde{z}^{p+2}).$$

It is worth considering the exact form of $N(z)$, the numerator of the stability function, and the error constant when the method is implicit. The numerator of the stability function, $N(z)$ is

$$N(z) = \big(\exp(z) - \sigma z^{p+1}\big)(1-\lambda z)^{p+1} + O(z^{p+2}). \tag{3.6.1}$$

Expanding the denominator of the stability function, gives

$$(1-\lambda z)^{p+1} = 1 + q_1 z + q_2 z^2 + \cdots + q_p z^p + q_{p+1} z^{p+1},$$

where the coefficients q_i for $i = 1, 2, \ldots, p+1$ are

$$q_i = (-\lambda)^i \binom{p+1}{i}.$$

Now expanding the right hand side of (3.6.1)

$$\begin{aligned}\big(\exp(z) - \sigma z^{p+1}\big)(1-\lambda z)^{p+1} &= 1 + \left(\frac{1}{1!} + \frac{q_1}{0!}\right) z + \left(\frac{1}{2!} + \frac{q_1}{1!} + \frac{q_2}{0!}\right) z^2 \\ &\quad + \cdots + \left(\frac{1}{(p+1)!} + \frac{q_1}{p!} + \cdots + \frac{q_{p+1}}{0!} - \sigma\right) z^{p+1}.\end{aligned}$$

Given scalars n and m, such that $n \le m$ define the polynomial $M_{n,m}$ as

$$M_{n,m}(\lambda) = \sum_{i=0}^{n} \binom{m}{i} \frac{(-\lambda)^i}{(n-i)!}. \tag{3.6.2}$$

If $m \ge n$ and $\lambda \ne 0$, $M_{n,m}(\lambda)$ is given in terms of Generalised Laguerre polynomials (see for example [1]) as

$$M_{n,m}(\lambda) = (-\lambda)^n L_n^{(m-n)}\left(\frac{1}{\lambda}\right),$$

and if $\lambda = 0$, then

$$M_{n,m}(0) = \frac{1}{n!},$$

and it is convention that $M_{-1,-1} = 0$. So the numerator of the stability function is

$$N(z) = \sum_{n=0}^{p} M_{n,p+1}(\lambda) z^n + \epsilon z^{p+1}.$$

It also follows that the error constant is given by

$$\sigma = M_{p+1,p+1}(\lambda) - \epsilon. \tag{3.6.3}$$

The following table gives expressions for the error constants for orders up to six.

p	σ
1	$\frac{1}{2} - 2\lambda + \lambda^2 - \epsilon$
2	$\frac{1}{6} - \frac{3}{2}\lambda + 3\lambda^2 - \lambda^3 - \epsilon$
3	$\frac{1}{24} - \frac{2}{3}\lambda + 3\lambda^2 - 4\lambda^3 + \lambda^4 - \epsilon$
4	$\frac{1}{120} - \frac{5}{24}\lambda + \frac{5}{3}\lambda^2 - 5\lambda^3 + 5\lambda^4 - \lambda^5 - \epsilon$
5	$\frac{1}{720} - \frac{1}{20}\lambda + \frac{5}{8}\lambda^2 - \frac{10}{3}\lambda^3 + \frac{15}{2}\lambda^4 - 6\lambda^5 + \lambda^6 - \epsilon$
6	$\frac{1}{5040} - \frac{7}{120}\lambda + \frac{7}{40}\lambda^2 - \frac{35}{24}\lambda^3 + \frac{35}{6}\lambda^4 - \frac{21}{2}\lambda^5 + 7\lambda^6 - \lambda^7 - \epsilon$

Table 3.2: Error constants for orders up to six.

Consider the coefficients in certain polynomial or Taylor series expansions. Write

$$\langle z^n, \phi(z) \rangle ,$$

for the z^n coefficient in the Taylor series expansion of $\phi(z)$ about $z = 0$. Note that $R(z)$ and $\widetilde{R}(\widetilde{z}\,)$ are related by the transformations

$$z = \frac{\widetilde{z}}{(1+\lambda\widetilde{z}\,)} \qquad \Longleftrightarrow \qquad \widetilde{z} = \frac{z}{(1-\lambda z)}.$$

In these rational functions, $N(z)$ and $\widetilde{N}(\widetilde{z}\,)$ are polynomials of degree $p+1$. The coefficients of order $p+1$ are given by

$$\begin{aligned} \epsilon &= \left\langle z^{p+1}, N(z) \right\rangle \\ &= -\sigma + \left\langle z^{p+1}, (1-\lambda z)^{p+1} \exp(z) \right\rangle , \\ \widetilde{\epsilon} &= \left\langle \widetilde{z}^{p+1}, \widetilde{N}(\widetilde{z}\,) \right\rangle \\ &= -\sigma + \left\langle \widetilde{z}^{p+1}, \exp\left(\frac{\widetilde{z}}{1+\lambda\widetilde{z}} \right) \right\rangle . \end{aligned}$$

Since, the two adjustments will be identical, it follows that

$$\widetilde{\epsilon} = \epsilon + \left\langle z^{p+1}, \exp\left(\frac{z}{1+\lambda z} \right) - (1-\lambda z)^{p+1} \exp(z) \right\rangle . \tag{3.6.4}$$

The following lemma is essential for understanding what the expression on the right hand side of (3.6.4) is equivalent to.

Lemma 3.20 *Given scalars n and m and the polynomial $M_{n,m}$, the following relations hold*

$$\langle z^n, (1-\lambda z)^m \exp(z) \rangle = M_{n,m}(\lambda), \tag{3.6.5}$$

$$\left\langle \widetilde{z}^n, \exp\left(\frac{\widetilde{z}}{1+\lambda\widetilde{z}} \right) \right\rangle = M_{n,n}(\lambda) + \lambda M_{n-1,n-1}(\lambda). \tag{3.6.6}$$

Proof: To prove (3.6.5) expand $(1-\lambda z)^m \exp(z)$ in powers of z and pick out the coefficients of z^n. To prove (3.6.6) make the substitution $\widetilde{z} = z/(1-\lambda z)$, so that $z = \widetilde{z}/(1+\lambda\widetilde{z}\,)$. If

$$\exp\left(\frac{\widetilde{z}}{1+\lambda\widetilde{z}} \right) = 1 + N_1(\lambda)\widetilde{z} + N_2(\lambda)\widetilde{z}^2 + N_3(\lambda)\widetilde{z}^3 + \cdots ,$$

then

$$\exp(z) = 1 + N_1(\lambda)\frac{z}{1-\lambda z} + N_2(\lambda)\left(\frac{z}{1-\lambda z}\right)^2 + N_3(\lambda)\left(\frac{z}{1-\lambda z}\right)^3 + \cdots .$$

Multiplying both sides by $(1-\lambda z)^{n-1}$ and pick out the coefficients of z^n, where it is noted that the terms

$$\left(\frac{z}{1-\lambda z}\right)^i (1-\lambda z)^{n-1},$$

are polynomials of degree less than n if $i < n$. Then

$$\begin{aligned} N_n(\lambda) &= \left\langle z^n, (1-\lambda z)^{n-1}\exp(z)\right\rangle \\ &= \left\langle z^n, \left((1-\lambda z)^n + \lambda z(1-\lambda z)^{n-1}\right)\exp(z)\right\rangle \\ &= \langle z^n, (1-\lambda z)^n \exp(z)\rangle + \lambda\left\langle z^n, z(1-\lambda z)^{n-1}\exp(z)\right\rangle \\ &= \langle z^n, (1-\lambda z)^n \exp(z)\rangle + \lambda\left\langle z^{n-1}, (1-\lambda z)^{n-1}\exp(z)\right\rangle \\ &= M_{n,n}(\lambda) + \lambda M_{n-1,n-1}(\lambda). \end{aligned} \tag{3.6.7}$$

□

Note that $N_n(\lambda)$ can be further simplified by substituting the required values of (3.6.2) into (3.6.7) and using the relation

$$\binom{n}{i} - \binom{n-1}{i-1} = \binom{n-1}{i},$$

it then follows that

$$N_n(\lambda) = \sum_{i=0}^{n-1}\binom{n-1}{i}\frac{(-\lambda)^i}{(n-i)!}.$$

Substituting (3.6.5) and (3.6.6) into (3.6.4) it follows that

$$\widetilde{\epsilon} = \epsilon + \lambda M_{p,p}(\lambda).$$

It is necessary to obtain a final condition which relates the (1,1) element of $\widetilde{B}$ to the transformed error constant $\widetilde{\epsilon}$. First define the vector Λ, given by

$$\Lambda = [\ \lambda^p \quad \lambda^{p-1} \quad \cdots \quad \lambda \quad 1\ ]^T.$$

The coefficient of the $\widetilde{z}^{p+1}$ term in the transformed stability function can be found noting that $\widetilde{B}$ is lower triangular using (3.2.9), noting that $\widetilde{A}$ is strictly lower triangular and given (3.4.7) then

$$\begin{aligned} \widetilde{\epsilon} = \left\langle \widetilde{z}^{p+1}, \widetilde{R}(\widetilde{z})\right\rangle &= \left\langle \widetilde{z}^{p+1}, e_1^T\left(\widetilde{V} + \widetilde{z}\widetilde{B}\widetilde{U}\right)\widetilde{Z}\right\rangle \\ &= e_1^T\widetilde{B}\widetilde{U}e_{p+1} \\ &= \widetilde{B}_{11}e_1^T W^{-1}C(I-\lambda K)\Psi e_{p+1} \\ &= \widetilde{B}_{11}e_1^T W^{-1}C(I-\lambda K)\beta(K)\exp(\lambda K^-)e_{p+1} \\ &= \widetilde{B}_{11}e_1^T W^{-1}C\beta(K)(I-\lambda K)\Lambda \\ &= \widetilde{B}_{11}e_1^T W^{-1}C\beta(K)e_{p+1}. \end{aligned}$$

Note that $\beta(K)e_{p+1}$ has the simple form

$$\beta(K)e_{p+1} = [\ \beta_p \quad \beta_{p-1} \quad \cdots \quad \beta_1 \quad 1\ ]^T,$$

so that the required condition is

$$\widetilde{B}_{11} e_1^T W^{-1} C \beta(K) e_{p+1} = \epsilon + \lambda M_{p,p}(\lambda). \tag{3.6.8}$$

It is now possible to collect all of the equations to be solved in such a way that all methods with the IRKS property can be found.

Before introducing the canonical forms of methods it is worth considering the choice of λ which guarantees the method is A-stable. An IRKS method, like a Runge-Kutta method, is A-stable if

$$\left| R(iy) \right| \leq 1, \tag{3.6.9}$$

for real y and $R(z)$ is analytic for $\mathrm{Re}(z) < 0$. This follows from the maximum modulus principle. The condition (3.6.9) is equivalent to

$$\begin{aligned} E(y) &= \left|(1-\lambda iy)^{p+1}\right|^2 - \left|N(iy)\right|^2 \\ &= (1-\lambda iy)^{p+1}(1+\lambda iy)^{p+1} - N(iy)N(-iy) \\ &\geq 0. \end{aligned}$$

The values of λ for which the method is A-stable are given in the following table.

p	A-stable choices of λ
1	$[0.29289321, 1.70710678]$
2	$[0.18042530, 2.18560009]$
3	$[0.22364780, 0.57281606]$
4	$[0.24799464, 0.67604239]$
5	$[0.18391465, 0.33414236]$
6	$[0.20408345, 0.37886489]$
7	$[0.15665860, 0.23437316]$

Table 3.3: A-stability regions for values of λ for orders up to seven.

It will always be assumed that when deriving an implicit method λ will always be chosen to lie in the correct interval.

3.7 Canonical forms of methods

Before introducing the approach used to generate methods it is worth collecting all the assumptions that have been made into one place, making it clear what is meant when the term a general linear method with IRKS is used. It will then be possible to obtain canonical forms of methods.

Let (A, U, B, V) denote a general linear method which is characterized by the following properties.

- The stage order and order are each equal to p.
- The number of input vectors and the number of stages are given by $r = s = p + 1$.
- The vector of abscissae is given by the vector c which are free parameters.

- The vector valued function representing the input approximations is equal to Z, which implies that

$$\begin{aligned} U &= C - ACK, \\ V &= E - BCK, \end{aligned}$$

  to ensure stage order and order equal p.

- A can be a full matrix but all the eigenvalues of A are equal to λ. For any matrix A there exists a nonsingular matrix W such that

$$\bar{A} = W^{-1}AW,$$

  is diagonally implicit.

- The method has Runge-Kutta stability with stability function

$$R(z) = \frac{N(z)}{(1-\lambda z)^{p+1}},$$

  with $N(z) = \exp(z)(1-\lambda z)^{p+1} + O(z^{p+1})$ and with the coefficient of z^{p+1} in $N(z)$ equal to ϵ.

- The method has IRKS, which means the method satisfies

$$\begin{aligned} BA &= XB, \\ BU &\equiv XV - VX, \\ \sigma(V) &= \{1, 0\}, \end{aligned} \tag{3.7.1}$$

  where X is a doubly companion matrix, with eigenvalues the same as A. The first row of X (3.3.5), is determined once λ and the last p elements of the last column of X have been chosen. That is, the coefficients of the polynomial $\beta(w)$ are last p elements of last column of X. These conditions automatically ensure the method has Runge-Kutta stability. Note that (3.7.1) implies that V has the form

$$V = \begin{bmatrix} 1 & v^T \\ 0 & \dot{V} \end{bmatrix},$$

  where $\rho(\dot{V}) = 0$.

In the result stated later in this section, a "method" refers to a general linear method satisfying the above requirements. Before proceeding to this result some preparatory work is required.

Recall the factorisation

$$B = \Psi \widetilde{B} W^{-1},$$

where $\widetilde{B}$ is lower triangular. $\Psi = \beta(K)\exp(\lambda K^{-})$ is defined once λ and the coefficients of $\beta(w)$ are chosen. The nonsingular matrix W^{-1} is chosen so that the required form of A results. It is clear that once $\widetilde{B}$ is known, the method is completely determined. Now concentrate on determining the choices of $\widetilde{B}$ which satisfy the requirements of IRKS.

The following $p \times (p+1)$ and $(p+1) \times p$ matrices are used to remove the first row and column of a matrix or to insert an additional first row or column of zeros to a given matrix. They are respectively

$$[\mathrm{O\,I}] = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} \mathrm{O} \\ \mathrm{I} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}.$$

From the requirement $BU \equiv XV - VX$, it was shown in (3.5.14) that if $BA = XB$ is to be satisfied then only p conditions result. These p conditions are the last p elements of the last column of (3.5.14). These conditions can be written in the form

$$[\mathrm{O\,I}]\widetilde{B}W^{-1}C\beta(K)e_{p+1} = [\mathrm{O\,I}](JF - FJ)e_{p+1}.$$

Using this condition, together with the relation on the (1,1) element of $\widetilde{B}$,

$$\widetilde{B}_{11}e_1^T W^{-1}C\beta(K)e_{p+1} = \epsilon + \lambda M_{p,p}(\lambda),$$

then a linear relation of the form

$$\widetilde{B}W^{-1}C\beta(K)e_{p+1} = \delta, \tag{3.7.2}$$

is deduced, where δ is given by

$$\delta = \begin{bmatrix} \epsilon + \lambda M_{p,p}(\lambda) & N_p(\lambda) & \cdots & N_2(\lambda) & N_1(\lambda) \end{bmatrix}^T. \tag{3.7.3}$$

In order to show how a square matrix has zero spectral radius in the most general way consider the following operations on square matrices. These include extracting the upper or lower triangular part of a matrix. Let $\mathcal{L}_n$ denote the set of $n \times n$ lower triangular matrices and $\mathcal{U}_n$ the corresponding set of upper triangular matrices. The notation $\Delta(M)$ will denote the lower triangular part of a square matrix M. Throughout this thesis, $\mathbf{L}(M)$ and $\mathbf{U}(M)$, where again M is a square matrix, will denote a unit lower triangular matrix and an upper triangular matrix respectively such that $M = \mathbf{L}(M)\mathbf{U}(M)$, assuming this LU factorisation exists.

The following lemma shows the connection between two upper triangular matrices and the lower triangular part of a matrix.

Lemma 3.21 *Given two upper triangular matrices G_1 and G_2 then*

$$\Delta(G_1\Delta(H)G_2) = \Delta(G_1 H G_2).$$

Proof: This follows by induction. □

The following lemma is essential in the derivation of methods.

Lemma 3.22 *Define* $f : \mathcal{L}_n \to \mathcal{L}_n$ *by*

$$f(H) = \Delta(G_1 H G_2)$$

then the inverse mapping, if it exists, is defined by

$$f^{-1}(F) = \mathbf{L}(G_1^{-1})\Delta\left(\mathbf{U}(G_1^{-1})F\mathbf{U}(G_2)^{-1}\right)\mathbf{L}(G_2)^{-1}.$$

Proof: Given the LU factors of G_1^{-1} and G_2 are $\mathbf{L}(G_1^{-1})\mathbf{U}(G_1^{-1})$ and $\mathbf{L}(G_2)\mathbf{U}(G_2)$ then

$$\begin{aligned} f(H) &= \Delta(G_1 H G_2) \\ &= \Delta\big(\mathbf{U}(G_1^{-1})^{-1}\mathbf{L}(G_1^{-1})^{-1}H\mathbf{L}(G_2)\mathbf{U}(G_2)\big) \end{aligned}$$

Multiplying on the left by $\mathbf{U}(G_1^{-1})$ and on the right by $\mathbf{U}(G_2)^{-1}$ then from Lemma 3.21, it follows that

$$\Delta\big(\mathbf{U}(G_1^{-1})f(H)\mathbf{U}(G_2)^{-1}\big) = \Delta\big(\mathbf{L}(G_1^{-1})^{-1}H\mathbf{L}(G_2)\big)$$

Since the product of the matrices on the right hand side is already lower triangular. Multiply on the left by $\mathbf{L}(G_1^{-1})$ and on the right by $\mathbf{L}(G_2)^{-1}$ then

$$H = \mathbf{L}(G_1^{-1})\Delta\left(\mathbf{U}(G_1^{-1})f(H)\mathbf{U}(G_2)^{-1}\right)\mathbf{L}(G_2)^{-1}.$$

Substituting F for $f(H)$ yields the required result. □

In the following lemma, the construction of a matrix with zero spectral radius in the most general way is discussed.

Lemma 3.23 *An* $n \times n$ *matrix* M *has spectral radius zero if and only if there exists a permutation* P *on* $(1, 2, \ldots, n)$ *and an lower triangular matrix* L *such that* $\Delta(L^{-1}P^T MPL)$ *is the zero matrix.*

Proof: The 'if' part is obvious and only the 'only' part is proved. Let T denote the transformation to upper Jordan canonical form $T^{-1}MT = K$ and let the $T = PLR$ be an LU decomposition with partial pivoting. Since T is non-singular, this factorisation always exists. Hence, $L^{-1}P^T MPL$ is strictly upper triangular and the result follows. □

To ensure that $\rho\big(\dot{V}\big) = 0$ it is sufficient to ensure that $\rho\big(\dot{\widetilde{V}}\big) = 0$, where $\widetilde{V}$ is

$$\widetilde{V} = \begin{bmatrix} 1 & \widetilde{v}^T \\ 0 & \dot{\widetilde{V}} \end{bmatrix}.$$

The above equation takes the form

$$\rho\left([\mathrm{O}\,\mathrm{I}](F - \widetilde{B}W^{-1}CK\Psi)\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}\right) = 0. \qquad (3.7.4)$$

Using Lemma 3.23, this condition can alternatively be expressed as

$$\Delta\left(T^{-1}[\mathrm{O}\,\mathrm{I}](F - \widetilde{B}W^{-1}CK\Psi)\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}T\right) = 0, \qquad (3.7.5)$$

where T is the product of a permutation matrix and a lower triangular matrix. The remaining two conditions to satisfy are (3.7.2) and (3.7.5). There are two main approaches to satisfying these conditions they are given in the following two subsections.

3.7.1 Approach one

This approach aims to solves the two remaining conditions in a one step process. In order to do this notice that the condition (3.7.2) will define the first column of $\widetilde{B}$, and to ensure that $\dot{V}$ has zero spectral radius the first row and column of (3.5.13) are removed, resulting in (3.7.4). So it is possible to substitute the condition (3.7.2) into the first column of (3.5.13). The rest of the first row can be ignored since $\widetilde{B}$ is lower triangular and (3.7.2) satisfies the (1,1) element. Removing the unnecessary information from (3.5.13) while retaining the $(p+1)\times(p+1)$ size is satisfied as follows

$$\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}](\widetilde{B}W^{-1}CK\Psi)\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}] = \begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}](F-\widetilde{V})\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}].$$

The linear relation (3.7.2) is made into a $(p+1)\times(p+1)$ matrix by adding p columns of zeros, that is

$$\widetilde{B}W^{-1}C\beta(K)e_{p+1}e_1^T = \delta e_1^T.$$

Now it is possible to add the two equations together which gives

$$\widetilde{B}W^{-1}C\big(\beta(K)e_{p+1}e_1^T + K\Psi\big) = \begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}](F-\widetilde{V})\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}] + \delta e_1^T.$$

since

$$K\Psi\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}] = K\Psi.$$

To simplify notation let Ω and Γ be defined as

$$\Omega = W^{-1}C\big(\beta(K)e_{p+1}e_1^T + K\Psi\big), \tag{3.7.6}$$

$$\Gamma = \begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}]F\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}] + \delta e_1^T. \tag{3.7.7}$$

Now introduce a matrix $\mathcal{T}$ which is given by

$$\mathcal{T} = \begin{bmatrix}1 & 0\\ 0 & T\end{bmatrix},$$

where T is the product of a permutation matrix and a lower triangular matrix. It then follows that

$$\Delta\left(\mathcal{T}^{-1}\widetilde{B}\Omega\mathcal{T}\right) = \Delta\left(\mathcal{T}^{-1}\Gamma\mathcal{T}\right).$$

Let $\mathcal{T} = \mathbf{L}(\mathcal{T})\,\mathbf{U}(\mathcal{T})$ and $\Omega\mathcal{T} = \mathbf{L}(\Omega\mathcal{T})\,\mathbf{U}(\Omega\mathcal{T})$ be the LU factors. It follows from the use of Lemma 3.22, that the matrix $\widetilde{B}$ is given by

$$\widetilde{B} = \mathbf{L}(\mathcal{T})\,\Delta\left(\mathbf{U}(\mathcal{T})\,\Delta\left(\mathcal{T}^{-1}\Gamma\mathcal{T}\right)\mathbf{U}(\Omega\mathcal{T})^{-1}\right)\mathbf{L}(\Omega\mathcal{T})^{-1}.$$

Using Lemma 3.21 the above equation reduces to

$$\widetilde{B} = \mathbf{L}(\mathcal{T})\Delta\left(\mathbf{U}(\mathcal{T})\mathcal{T}^{-1}\Gamma\mathcal{T}\mathbf{U}(\Omega\mathcal{T})^{-1}\right)\mathbf{L}(\Omega\mathcal{T})^{-1}. \tag{3.7.8}$$

The main theorem can now be stated.

Theorem 3.24 *Let (A, U, B, V) be a method. It is assumed that the method is non-confluent in the sense that the components of c are distinct. Suppose also that the quantities δ, Ω and Γ are as defined in this section. Then there exists a $(p+1) \times (p+1)$ matrix $\mathcal{T}$ where the last p rows and columns are equal to the product of a permutation matrix and a unit lower triangular matrix such that (3.7.8) holds.*

Proof: Define T so that (3.7.5) holds. This implies (3.7.8) holds. □

Once $\widetilde{B}$ is known it is easy to construct the rest of the method.

3.7.2 Approach two

This approach aims to solves the two remaining conditions in a two step process. To achieve this, let $\widehat{B}$ denote a lower triangular $p \times p$ matrix related to $\widetilde{B}$ by

$$\widetilde{B} = \begin{bmatrix} \mathrm{O} \\ \mathrm{I} \end{bmatrix} \widehat{B} [\mathrm{O}\ \mathrm{I}](I - W^{-1} C \beta(K) e_{p+1} e_1^T) + \delta e_1^T, \tag{3.7.9}$$

so that, once $\widehat{B}$ is known then the values of $\widetilde{B}$ follows. Furthermore this matrix is guaranteed to satisfy (3.7.2). Substituting the expression (3.7.9) into (3.7.5) and collecting all terms involving $\widehat{B}$ yields

$$\begin{aligned} &\Delta\left(T^{-1} \widehat{B} [\mathrm{O}\ \mathrm{I}](I - W^{-1} C \beta(K) e_{p+1} e_1^T) W^{-1} C K \Psi \begin{bmatrix} \mathrm{O} \\ \mathrm{I} \end{bmatrix} T\right) \\ &\quad = \Delta\left(T^{-1} [\mathrm{O}\ \mathrm{I}](F - \delta e_1^T W^{-1} C K \Psi) \begin{bmatrix} \mathrm{O} \\ \mathrm{I} \end{bmatrix} T\right). \end{aligned}$$

Before showing how to find $\widehat{B}$ simplify the discussion by defining

$$\Theta = [\mathrm{O}\ \mathrm{I}]\big(I - W^{-1} C \beta(K) e_{p+1} e_1^T\big) W^{-1} C K \Psi \begin{bmatrix} \mathrm{O} \\ \mathrm{I} \end{bmatrix},$$

$$\Upsilon = [\mathrm{O}\ \mathrm{I}](F - \delta e_1^T W^{-1} C K \Psi) \begin{bmatrix} \mathrm{O} \\ \mathrm{I} \end{bmatrix},$$

Therefore

$$\Delta\left(T^{-1} \widehat{B} \Theta T\right) = \Delta\left(T^{-1} \Upsilon T\right).$$

Let $T = \mathbf{L}(T)\mathbf{U}(T)$ and $\Theta T = \mathbf{L}(\Theta T)\,\mathbf{U}(\Theta T)$ be the LU factors. It follows from the use of Lemma 3.22, that the matrix $\widehat{B}$ is given by

$$\widehat{B} = \mathbf{L}(T) \Delta\left(\mathbf{U}(T) \Delta\left(T^{-1} \Upsilon T\right) \mathbf{U}(\Theta T)^{-1}\right) \mathbf{L}(\Theta T)^{-1}.$$

Using Lemma 3.21 it is possible to simplify the above equation as

$$\widehat{B} = \mathbf{L}(T) \Delta\left(\mathbf{U}(T) T^{-1} \Upsilon T \mathbf{U}(\Theta T)^{-1}\right) \mathbf{L}(\Theta T)^{-1}. \tag{3.7.10}$$

The main theorem can now be stated.

Theorem 3.25 *Let (A, U, B, V) be a method. It is assumed that the method is non-confluent in the sense that the components of c are distinct. Suppose also that the quantities δ, Θ and Υ are as defined in this section. Then there exists a $p \times p$ matrix T equal to the product of a permutation matrix and a unit lower triangular matrix such that (3.7.10) holds.*

Proof: Define T so that (3.7.5) holds; this implies (3.7.10) holds. □

Once $\widehat{B}$ is known it is easy to construct $\widetilde{B}$ and thus the rest of the method.

3.8 Finding methods

To actually derive a method first choose whether approach one or approach two is preferred. The following procedure is based on approach one, but it is a simple matter to change this to approach two, if this is preferred. Choose the free parameters of the method:

- p: The order of the method.
- λ: The eigenvalue of the matrix A, usually chosen so that the method has A-stability.
- ϵ: The z^{p+1} coefficient in the numerator of the stability function generally chosen to ensure L-stability for implicit methods or small error constant for explicit methods.
- c: The non-confluent abscissae vector generally chosen so that each abscissae is at least equal to zero but no greater than one.
- β: The last p elements of the last column of the doubly companion matrix X.
- T: The $p \times p$ matrix which is the product of a permutation matrix and a unit lower triangular matrix.
- W: The $(p+1)\times(p+1)$ transformation matrix generally chosen so that $W = I$ which ensures the method is diagonally implicit.

Given these free parameters or otherwise construct the following matrices

$$J, \quad K, \quad K^-, \quad E, \quad \beta(K), \quad \exp(\lambda K^-), \quad \exp(-\lambda K^-).$$

Using these matrices or otherwise construct

$$\begin{aligned}
\Psi &= \beta(K)\exp(\lambda K^-),\\
X &= \Psi(J+\lambda I)\Psi^{-1},\\
F &= \exp(-\lambda K^-)E\exp(\lambda K^-),\\
\Omega &= W^{-1}C\big(\beta(K)e_{p+1}e_1^T + K\Psi\big),\\
\Gamma &= \begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\ \mathrm{I}]F\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\ \mathrm{I}] + \delta e_1^T.
\end{aligned}$$

Note that the matrix Γ can alternatively be found in a two step process which is useful for the implementation of this process. The first step is

$$\Gamma = F + (JF - FJ)P,$$

where $P = K + e_{p+1}e_1^T$. The second step is to replace the (1,1) element of Γ by $\epsilon + M_{p,p}(\lambda)$. This avoids needing to cut rows and columns and then replace them with rows and columns of zeros. This two step process creates a Γ which is identical to (3.7.7).

Now find the LU factors of $\mathcal{T}$ and $\Omega\mathcal{T}$, which are required for developing methods using approach one of Section 3.7. It then follows that

$$\widetilde{B} = \mathbf{L}(\mathcal{T})\Delta\left(\mathbf{U}(\mathcal{T})\mathcal{T}^{-1}\Gamma\mathcal{T}\mathbf{U}\left(\Omega\mathcal{T}\right)^{-1}\right)\mathbf{L}\left(\Omega\mathcal{T}\right)^{-1}.$$

The original method is therefore given by

$$\begin{aligned} B &= \Psi \widetilde{B} W^{-1}, \\ A &= B^{-1} X B, \\ U &= C - ACK, \\ V &= E - BCK. \end{aligned}$$

In Appendix I a Maple code is included to derive methods using approach one of Section 3.7 summarised above.

Not all choices of the free parameters guarantee that a method exists. Constructing a method can fail, for example, when $\Omega\mathcal{T}$ does not have LU factors. Another possibility is when $\widetilde{B}$ is singular and the resulting method is reducible in the number of stages. This is not always the case though; in fact methods exist where the free parameters result in the matrix $\widetilde{B}$ being singular but the method is not reducible. There are two main cases to consider. The first is when $\widetilde{B}_{11} = 0$ and the second is when $\widetilde{B}_{ii} = 0$, for $i > 1$. Note that $\widetilde{B}$ is lower triangular and $\widetilde{A}$ is strictly lower triangular. In the first case the method, if it exists, is still unique. This can be seen by splitting the equation $\widetilde{B}\widetilde{A} = J\widetilde{B}$ up

$$\left[\begin{array}{c|c} 0 & 0 \\ \hline \dot{b} & \dot{B} \end{array}\right] \left[\begin{array}{c|c} 0 & 0 \\ \hline \dot{A} & 0 \end{array}\right] = \left[\begin{array}{c|c} 0 & 0 \\ \hline e_1 & \dot{J} \end{array}\right] \left[\begin{array}{c|c} 0 & 0 \\ \hline \bar{B} & \bar{b} \end{array}\right],$$

where $\dot{A}, \dot{B}, \dot{J}$ and $\bar{B}$ are $p \times p$ matrices and $\dot{b}$ and $\bar{b}$ are vectors of length p. Without loss of generality the system can then be reduced to

$$\dot{B}\dot{A} = \dot{J}\bar{B}. \tag{3.8.1}$$

The matrix A is then computed by

$$A = W \left(\begin{bmatrix} \mathrm{O} \\ \mathrm{I} \end{bmatrix} \dot{B}^{-1} \dot{J} \bar{B} [\mathrm{O}\; \mathrm{I}] + \lambda I \right) W^{-1}.$$

It follows from (3.8.1) that $\dot{A}_{11} = 0$ which means that if $W = I$ and $\lambda = 0$ the method is necessarily reducible. It also follows from (3.6.8) that when $\widetilde{B}_{11} = 0$, λ and ϵ are related by the equation

$$\epsilon = -\lambda M_{p,p}(\lambda). \tag{3.8.2}$$

In the second case, if the method exists, it is generally (but not always see Section 3.9) not unique. To show that if $\widetilde{B}_{ii} = 0$, for $i > 1$ then $\widetilde{B}_{jj} = 0$ for $j < i$ an induction argument is used. To avoid the tedious nature of this proof, an example when $p = 2$ is included which conveys the main points. Equation (3.5.9) when $p = 2$ and $\widetilde{B}_{22} = 0$ is

$$\begin{bmatrix} \widetilde{B}_{11} & 0 & 0 \\ \widetilde{B}_{21} & 0 & 0 \\ \widetilde{B}_{31} & \widetilde{B}_{32} & \widetilde{B}_{33} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \widetilde{A}_{21} & 0 & 0 \\ \widetilde{A}_{31} & \widetilde{A}_{32} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \widetilde{B}_{11} & 0 & 0 \\ \widetilde{B}_{21} & 0 & 0 \\ \widetilde{B}_{31} & \widetilde{B}_{32} & \widetilde{B}_{33} \end{bmatrix}.$$

Expanding both sides it follows that

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \widetilde{B}_{32}\widetilde{A}_{21} + \widetilde{B}_{33}\widetilde{A}_{31} & \widetilde{B}_{33}\widetilde{A}_{32} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \widetilde{B}_{11} & 0 & 0 \\ \widetilde{B}_{21} & 0 & 0 \end{bmatrix},$$

which directly implies that $\widetilde{B}_{11} = 0$, $\widetilde{A}_{32} = 0$ and $\widetilde{A}_{21}$ and $\widetilde{A}_{31}$ are governed by a single equation, which leads to a degree of freedom in the method.

As an example choose the free parameters of the method as

$$p = 2, \quad \lambda = \tfrac{1}{4}, \quad \epsilon = -\tfrac{1}{64}, \quad c = [\ \tfrac{1}{3} \ \ \tfrac{2}{3} \ \ 1\ ]^T, \quad \beta = [\ \tfrac{1}{18} \ \ 0 \ \ 1\ ]^T, \quad T = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix}, \quad W = I,$$

leads to the method

$$M = \left[\begin{array}{ccc|ccc} \frac{1}{4} & 0 & 0 & 1 & \frac{1}{12} & -\frac{1}{36} \\ \frac{9}{16} - \frac{a}{2} & \frac{1}{4} & 0 & 1 & -\frac{7}{48} + \frac{a}{2} & -\frac{19}{144} + \frac{a}{6} \\ a & 0 & \frac{1}{4} & 1 & \frac{3}{4} - a & \frac{1}{4} - \frac{a}{3} \\ \hline -\frac{25}{144} & \frac{17}{36} & \frac{17}{72} & 1 & \frac{67}{144} & \frac{1}{144} \\ -\frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 \\ -11 & 4 & 2 & 0 & 5 & 0 \end{array}\right],$$

where a is a free parameter, which is a special case of the method explained above. The methods in which $\widetilde{B}$ is not invertible can be thought of as very special cases, since ϵ must at least be chosen to satisfy (3.8.2). Since ϵ is a free parameter these special cases can be avoided by choosing ϵ so that (3.8.2) does not hold.

3.9 Special cases

There is a large amount of freedom available in the IRKS methods. The free parameters are λ the eigenvalue of A, the abscissae vector c, the coefficients $\beta_1, \beta_2, \ldots, \beta_p$ of the doubly companion matrix X, the z^{p+1} coefficient of the numerator of the stability function, ϵ, and the coefficients of the matrices T and W. In this section special choices of these free parameters are made to find methods with some special properties. It will be shown that ESIRK and DESIRE methods with $s = p + 1$ can be thought of as special cases of the IRKS methods. To date only ESIRK methods with $s = p$ have been derived, therefore the methods derived in this section extend the class of ESIRK methods known. It may also be the case that the methods with $s = p + 1$ may have some advantages over their $s = p$ counterparts. It will also be shown in this section how the free parameters can be chosen so that IRKS methods which are stiffly accurate can be derived.

3.9.1 ESIRK and DESIRE methods

The Singly Implicit Runge-Kutta (SIRK) methods were introduced by Norsett [103] and named by Burrage [11] who furnished the methods with error estimates. The introduction of the SIRK methods comes from a compromise between the fully implicit Runge-Kutta methods which have high implementation costs and the diagonally implicit methods which have low stage order. The SIRK methods are based on two assumptions. The first is that the order of the stage values and the order of the output values is the same. The second assumption ensures that the coefficient matrix A has a one point spectrum, that is $\sigma(A) = \{\lambda\}$. Butcher [24] showed how to transform A into Jordan canonical form which significantly reduced the cost of performing one step of the method. Due to the high stage order of these methods when $s = p$, the abscissae of the method must satisfy $c_i = \lambda\xi_i$, where ξ_i is the zero of the s degree Laguerre polynomial $L_s(x)$, for $i = 1, 2, \ldots, s$. It turns out that some of the abscissae are greater than 1 for methods where $s > 2$. This can cause problems when the problem is nonlinear. This led Butcher and Chartier [35], [36], to a generalisation of SIRK methods in which the traditional order requirements were weakened to effective order.

These methods are known as Effective order Singly Implicit Runge-Kutta (ESIRK) methods. In this case where the ESIRK method is designed to have a high stage order, it is sufficient to assume that the incoming quantity is given by the perturbation of the solution

$$y_{n-1} = y(x_{n-1}) + \epsilon_1 h y'(x_{n-1}) + \epsilon_2 h^2 y''(x_{n-1}) + \ldots + \epsilon_p h^p y^p(x_{n-1}) + O(h^{p+1}).$$

The corresponding quantity computed at the end of the step is given by

$$y_n = y(x_n) + \epsilon_1 h y'(x_n) + \epsilon_2 h^2 y''(x_n) + \ldots + \epsilon_p h^p y^p(x_n) + O(h^{p+1}).$$

The effect of relaxing the order requirement means that there is no requirement on the abscissae vector c. The conditions for high stage order are given by

$$\sum_{j=1}^{s} a_{ij}\frac{c_j^{k-1}}{(k-1)!} + \epsilon_k = \frac{c_i^k}{k!}, \qquad i = 1, 2, \ldots, s, \quad k = 1, 2, \ldots, p.$$

The requirements are now on $\epsilon_1, \epsilon_2, \ldots, \epsilon_p$ rather than the abscissae vector c. When implemented in a variable stepsize environment these methods can be conveniently represented as general linear methods in Nordsieck form, see [39], where $U = e\epsilon^T$ and $V = e_1\epsilon^T$ where

$$\epsilon = [\ 1 \quad \epsilon_1 \quad \epsilon_2 \quad \cdots \quad \epsilon_p\ ]^T.$$

Hence the perturbations of the solution used to calculate the stages and the update solution are the same. All ESIRK methods constructed have $s = p$. It is quite likely that advantages may exist for methods with $s = p + 1$. To show that ESIRK methods with $s = p + 1$ can be viewed as special cases of IRKS methods, the extra conditions needed to ensure $U = e\epsilon^T$ and $V = e_1\epsilon^T$ must be derived. The aim is to find relationships for the free parameters λ, β, c, T, W so that $U = e\epsilon^T$ and $V = e_1\epsilon^T$.

Since it is assumed that these methods have IRKS then

$$\begin{aligned} BA &= XB, \\ BU &\equiv XV - VX, \\ \sigma(V) &= \{1, 0\}, \end{aligned}$$

must be satisfied.

It is interesting that a further doubly companion matrix denoted as Y,

$$Y(\epsilon,\delta) = \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 & \cdots & -\epsilon_{p-1} & -\epsilon_p & -\epsilon_{p+1} - \delta_{p+1} \\ 1 & 0 & 0 & \cdots & 0 & 0 & -\delta_p \\ 0 & 1 & 0 & \cdots & 0 & 0 & -\delta_{p-1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & -\delta_3 \\ 0 & 0 & 0 & \cdots & 1 & 0 & -\delta_2 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -\delta_1 \end{bmatrix},$$

is required to analyse this special case. Note that $\epsilon(w)$ and $\delta(w)$ are polynomials given by

$$\epsilon(w) = 1 + \epsilon_1 w + \epsilon_2 w^2 + \cdots + \epsilon_{p+1} w^{p+1}, \tag{3.9.1}$$

$$\delta(w) = 1 + \delta_1 w + \delta_2 w^2 + \cdots + \delta_{p+1} w^{p+1}. \tag{3.9.2}$$

For $U = e\epsilon^T$ it is sufficient for the relation

$$AC = CY, \tag{3.9.3}$$

to hold. Using this relation in (3.2.9) gives

$$\begin{aligned} U &= C - ACK \\ &= C(I - YK) \\ &= Ce_1\epsilon^T \\ &= e\epsilon^T. \end{aligned}$$

For $V = e_1\epsilon^T$ it is sufficient for the relation

$$BC = EY, \tag{3.9.4}$$

to hold. Using this relation in (3.2.10) gives

$$\begin{aligned} V &= E - BCK \\ &= E(I - YK) \\ &= Ee_1\epsilon^T \\ &= e_1\epsilon^T. \end{aligned}$$

A direct consequence of V being rank 1 is that the transformation T must have no effect and therefore, $T = I$. Turn attention now to the first IRKS condition. Substituting the first sufficient condition (3.9.3) into (3.5.2) gives

$$BCY = XBC.$$

Now substituting the second sufficient condition (3.9.4) above gives

$$EY = XE. \tag{3.9.5}$$

This means the doubly companion matrices X and Y are similar. Now consider how W must be chosen so that A is singly implicit and $U = e\epsilon^T$. The transformation matrix W can be chosen in one of two ways. The first choice is to make the transformed matrix $\bar{A}$ into a single Jordan block, that is

$$\bar{A} = W^{-1}AW = J + \lambda I. \tag{3.9.6}$$

To see how this is done, note from Lemma 3.7 that

$$Y = \Upsilon(J + \lambda I)\Upsilon^{-1}, \tag{3.9.7}$$

where the matrices Υ and Υ^{-1} can be factorised in the form

$$\Upsilon = \delta(K)\exp(\lambda K^-), \qquad \Upsilon^{-1} = \exp(\lambda K^+)\epsilon(K). \tag{3.9.8}$$

Therefore, the two doubly companion matrices are also related by the expression

$$\Upsilon^{-1}Y\Upsilon = \Psi^{-1}X\Psi. \tag{3.9.9}$$

Comparing (3.9.5) and (3.9.9) it follows that

$$\Psi\Upsilon^{-1} = E. \tag{3.9.10}$$

Since $C^{-1}AC = Y$, equating with (3.9.7) gives

$$C^{-1}AC = \Upsilon(J+\lambda I)\Upsilon^{-1}.$$

Since $W^{-1}AW = J+\lambda I$, one deduces that

$$W = C\Upsilon.$$

So using (3.9.10) the transformation matrix W is computed using only known information. That is

$$\begin{aligned} W &= CE^{-1}\Psi \\ &= CE^{-1}\beta(K)\exp(\lambda K^{-}). \end{aligned} \tag{3.9.11}$$

What is also evident from (3.9.10) is that for the $s = p+1$ case, the vector ϵ is indirectly a free parameter along with the abscissae vector c. Substituting (3.9.8) into (3.9.10) yields

$$\beta(K)\exp(\lambda K^{-})\exp(\lambda K^{+})\epsilon(K) = E.$$

Choosing the matrix $\epsilon(K)$ and solving for $\beta(K)$ it follows that

$$\beta(K) = E\epsilon(K)^{-1}\exp(-\lambda K^{+})\exp(-\lambda K^{-}). \tag{3.9.12}$$

The alternative choice of the transformation is to factorise $W = C\Upsilon$ into RL, where R is a unit upper triangular and L is a lower triangular matrix. Hence

$$(RL)^{-1}ARL = L^{-1}(R^{-1}AR)L = J+\lambda I,$$

and rearranging gives

$$R^{-1}AR = L(J+\lambda I)L^{-1} = \bar{A},$$

where $\bar{A}$ is a lower triangular matrix with a constant diagonal entry λ. It was suggested by Butcher in [27] that such a choice would lower transformation costs by about 50% compared to the choice (3.9.11). Note that the method is identical regardless of the choice of transformation; the only effect is apparent in the implementation.

Below four methods are given, all of which are from the ESIRK family. The first method is implicit and is L-stable. Note that the U and V matrices have the required structure. Choosing the free parameters of the method as follows

$$p=2, \quad \lambda=\tfrac{1}{2}, \quad \epsilon=0, \quad c=[\,0 \;\; \tfrac{1}{2} \;\; 1\,]^T, \quad \beta=[\,\tfrac{1}{8} \;\; \tfrac{1}{2} \;\; 1\,]^T, \quad T=I,$$

$$W = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{8} \\ 1 & 1 & \frac{1}{4} \\ 1 & \frac{3}{2} & \frac{5}{8} \end{bmatrix},$$

leads to the ESIRK method

$$M = \left[\begin{array}{ccc|ccc} \frac{9}{8} & 0 & -\frac{1}{8} & 1 & -1 & \frac{1}{8} \\ \frac{5}{4} & \frac{1}{2} & -\frac{1}{4} & 1 & -1 & \frac{1}{8} \\ \frac{9}{8} & 1 & -\frac{1}{8} & 1 & -1 & \frac{1}{8} \\ \hline \frac{9}{8} & 1 & -\frac{1}{8} & 1 & -1 & \frac{1}{8} \\ -\frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \end{array}\right].$$

In equation (3.9.12) a relationship between $\beta(K)$ and the inverse of $\epsilon(K)$ was given. Therefore, to derive the SIRK methods with $s = p+1$, simply choose $\beta(K)$ to satisfy (3.9.12) with $\epsilon(K) = I$. The second method is implicit and is L-stable. Note that the β_1, β_2 have been chosen so that $U = ee_1^T$ and $V = e_1e_1^T$. Choosing the free parameters of the method as follows

$$p = 2, \quad \lambda = \tfrac{1}{2}, \quad \epsilon = 0, \quad c = [\ 0 \ \ \tfrac{1}{2} \ \ 1\ ]^T, \quad \beta = [\ -\tfrac{1}{4} \ \ -\tfrac{1}{2} \ \ 1\ ]^T, \quad T = I,$$

$$W = \begin{bmatrix} 1 & -\frac{1}{2} & \frac{1}{4} \\ 1 & 0 & -\frac{1}{8} \\ 1 & \frac{1}{2} & -\frac{1}{4} \end{bmatrix},$$

leads to the SIRK method

$$M = \left[\begin{array}{ccc|ccc} \frac{1}{2} & -1 & \frac{1}{2} & 1 & 0 & 0 \\ -\frac{1}{8} & 1 & -\frac{3}{8} & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & -2 & 2 & 0 & 0 & 0 \\ 3 & -8 & 5 & 0 & 0 & 0 \end{array}\right].$$

The ESIRK methods can in fact be derived very easily. To see how this is achieved, substitute (3.9.5) into (3.9.4) and expand as follows

$$\begin{aligned} B &= XEC^{-1} \\ &= \Psi(J + \lambda I)\Psi^{-1}EC^{-1} \\ &= \Psi(J + \lambda I)W^{-1}. \end{aligned}$$

Comparing with (3.5.10) means that $\widetilde{B}$ has the surprisingly simple form

$$\widetilde{B} = J + \lambda I.$$

When $\lambda = 0$, $\widetilde{B}$ is singular. In this case this does not imply that there is freedom in the matrix A as is the case when this special form is not required. That is

$$\begin{aligned} A &= W(J + \lambda I)W^{-1} \\ &= CE^{-1}\Psi(J + \lambda I)\Psi^{-1}EC^{-1} \\ &= CE^{-1}XEC^{-1}. \end{aligned}$$

This can also be seen by substituting (3.9.5) into (3.9.3). Substituting the expressions for A and B then $U = e\epsilon^T$ and $V = e_1\epsilon^T$ as is required. A direct consequence of the structure of the matrices is that

$$\epsilon = \langle z^{p+1}, N(z) \rangle = 0.$$

This means, when $\lambda \neq 0$ the method has L-stability, or alternatively when $\lambda = 0$ the method has a fixed error constant $1/(p+1)!$. Notice that the first row of $\exp(-\lambda K^-)$ is given by

$$e_1^T \exp(-\lambda K^-) = \left[\ 1 \quad \binom{p}{1}(-\lambda) \quad \binom{p}{2}(-\lambda)^2 \quad \cdots \quad \binom{p}{p-1}(-\lambda)^{p-1} \quad \binom{p}{p}(-\lambda)^p\ \right].$$

It is convenient to define the vector $E_{p\_1}^T$ as

$$E_{p\_1}^T = [\ \tfrac{1}{p!} \quad \tfrac{1}{(p-1)!} \quad \cdots \quad \tfrac{1}{2!} \quad \tfrac{1}{1!}\ ].$$

It is also convenient to define the reverse vector $E^T_{1\_p}$ as

$$E^T_{1\_p} = [\ \tfrac{1}{1!}\ \ \tfrac{1}{2!}\ \ \cdots\ \ \tfrac{1}{(p-1)!}\ \ \tfrac{1}{p!}\ ].$$

Substituting (3.9.11) into (3.6.8) it follows that

$$\begin{aligned}\epsilon + \lambda M_{p,p}(\lambda) &= \widetilde{B}_{11} e_1^T \Psi^{-1} E \beta(K) e_{p+1}\\ &= \widetilde{B}_{11} e_1^T \exp(-\lambda K^-) \beta(K)^{-1} E \beta(K) e_{p+1}\\ &= \widetilde{B}_{11} e_1^T \exp(-\lambda K^-) E e_{p+1}\\ &= \widetilde{B}_{11} e_1^T \exp(-\lambda K^-) E_{p\_1}\\ &= \widetilde{B}_{11} M_{p,p}(\lambda).\end{aligned}$$

Since $\widetilde{B} = J + \lambda I$ it follows that $\epsilon = 0$.

The third method is explicit, but the matrix A is a full matrix, which means that the stages can not be computed sequentially. Since these methods are not intended for the solution of stiff problems the stages can be computed using a fixed point iteration scheme instead of the usual variant of the Newton iteration scheme. Due to the high stage order it is possible to derive very accurate predictors, which may then result in only a few iterations being required at each step. This may make these methods comparable to explicit Runge-Kutta methods, which require more stages as the order increases. These methods are known as ESIRK methods with zero spectral radius. Choosing the free parameters of the method as follows

$$p = 2, \quad \lambda = 0, \quad \epsilon = 0, \quad c = [\ 0\ \ \tfrac{1}{2}\ \ 1\ ]^T, \quad \beta = [\ -\tfrac{1}{4}\ \ \tfrac{3}{4}\ \ 1\ ]^T, \quad T = I,$$

$$W = \begin{bmatrix} 1 & -\frac{1}{4} & -\frac{1}{2} \\ 1 & \frac{1}{4} & -\frac{1}{2} \\ 1 & \frac{3}{4} & -\frac{1}{4} \end{bmatrix},$$

leads to the ESIRK method with zero spectral radius

$$M = \left[\begin{array}{ccc|ccc} \frac{3}{8} & -\frac{1}{8} & -\frac{1}{2} & 1 & \frac{1}{4} & \frac{9}{16} \\ \frac{13}{8} & -\frac{15}{8} & \frac{1}{2} & 1 & \frac{1}{4} & \frac{9}{16} \\ \frac{19}{8} & -\frac{25}{8} & \frac{3}{2} & 1 & \frac{1}{4} & \frac{9}{16} \\ \hline \frac{19}{8} & -\frac{25}{8} & \frac{3}{2} & 1 & \frac{1}{4} & \frac{9}{16} \\ 1 & -2 & 2 & 0 & 0 & 0 \\ -2 & 2 & 0 & 0 & 0 & 0 \end{array}\right].$$

Note that error constant for this method is fixed to 1/3!.

In the above explanation no distinction is made between the ESIRK and the DESIRE methods. The difference is that for the DESIRE methods extra conditions are required. These conditions must ensure that the p elements of column $p+1$, in the matrix A are zero. This can be achieved by choosing $\beta(K)$ in a special way. Before introducing this result it is necessary to consider the inverse of a scaled Vandermonde matrix as in the following lemma.

Lemma 3.26 *The scaled Vandermonde matrix C is defined as*

$$C = \left[\ e\ \ c\ \ \tfrac{c^2}{2!}\ \ \cdots\ \ \tfrac{c^{p-1}}{(p-1)!}\ \ \tfrac{c^p}{p!}\ \right],$$

for some non-confluent vector c. The Lagrange basis elements are

$$\ell_j(x) = \prod_{l \neq j} \frac{x - c_l}{c_j - c_l} = \ell_{1,j} + \ell_{2,j}x + \cdots + \ell_{p+1,j}x^p. \tag{3.9.13}$$

Then the (i,j) element of the inverse of the scaled Vandermonde matrix is

$$C_{ij}^{-1} = (i-1)!\ell_{ij}. \tag{3.9.14}$$

Proof: The first thing to notice is that the scaled Vandermonde matrix can be decomposed as the product $C = VD$ where V is a Vandermonde matrix

$$V = \begin{bmatrix} e & c & c^2 & \cdots & c^{p-1} & c^p \end{bmatrix},$$

and D is the diagonal scaling matrix

$$D = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{1!} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2!} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{(p-2)!} & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{(p-1)!} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{p!} \end{bmatrix}. \tag{3.9.15}$$

Notice that (3.9.14) can alternatively be expressed in matrix form as the product $C^{-1} = D^{-1}\ell$. Therefore it is sufficient to show that $V^{-1} = \ell$. This follows from the fact that $\ell_j(x) = 1$ when $x = c_j$ and $\ell_j(x) = 0$ when $x \neq c_j$. □

To ensure that a method is a DESIRE method the following theorem must be satisfied.

Theorem 3.27 *For an ESIRK method to be strengthened to a DESIRE method*

$$Ae_{p+1} = \lambda e_{p+1}. \tag{3.9.16}$$

To satisfy this relation the free parameters $\beta_1, \beta_2, \ldots, \beta_p$ must be chosen so that

$$\beta(K) = E(I - \lambda K)\zeta(K), \tag{3.9.17}$$

where the polynomial $\zeta(w)$ is given by

$$\zeta(w) = \frac{1}{p!}\left(q_0 p! + q_1(-w)(p-1)! + q_2(-w)^2(p-2)! + \cdots + q_p(-w)^p 0!\right),$$

and the coefficients $q_0, q_1, \ldots, q_p$ satisfy the expression

$$q_0x^p + q_1x^{p-1} + q_2x^{p-2} + \ldots + q_p = (x - c_1)(x - c_2)\cdots(x - c_p).$$

Proof: Substituting (3.9.6) into (3.9.16) yields

$$W(J + \lambda I)W^{-1}e_{p+1} = \lambda e_{p+1}.$$

which simplifies to

$$JW^{-1}e_{p+1} = 0.$$

A direct consequence of the above result is that

$$JWe_{p+1} = 0,$$

also holds. Substituting (3.9.11) into the above expression yields

$$JCE^{-1}\beta(K)\exp(\lambda K^{-})e_{p+1} = 0.$$

Define the vector Λ, given by

$$\Lambda = [\ \lambda^p \quad \lambda^{p-1} \quad \cdots \quad \lambda \quad 1\ ]^T.$$

Now substituting (3.9.17) into $\beta(K)$ gives

$$\begin{aligned} 0 &= JCE^{-1}E(I-\lambda K)\zeta(K)\exp(\lambda K^{-})e_{p+1} \\ &= JC\zeta(K)(I-\lambda K)\Lambda \\ &= JC\zeta(K)e_{p+1}, \end{aligned}$$

since upper triangular Toeplitz matrices commute. It is now sufficient to show that the last column of $\zeta(K)$ is the last column of C^{-1} up to a scale factor, since $Je_{p+1} = 0$. The last column of $\zeta(K)$ is

$$\zeta(K)e_{p+1} = \frac{1}{p!}\begin{bmatrix} (-1)^p q_p 0! \\ (-1)^{p-1} q_{p-1} 1! \\ \vdots \\ -q_1(p-1)! \\ q_0 p! \end{bmatrix},$$

which is $1/p!$ times the numerator of each component of the last row of C^{-1}, compare (3.9.13). $\square$

The fourth method is implicit and is L-stable. The free parameters β_1, β_2 have been chosen so that $A_{13} = A_{23} = 0$. Therefore the method is a DESIRE method. Choosing the free parameters of the method as follows

$$p = 2, \quad \lambda = \tfrac{1}{2}, \quad \epsilon = 0, \quad c = [\ \tfrac{1}{3} \quad \tfrac{2}{3} \quad 1\ ]^T, \quad \beta = [\ 0 \quad -\tfrac{5}{36} \quad 1\ ]^T, \quad T = I,$$

$$W = \begin{bmatrix} 1 & \frac{1}{3} & 0 \\ 1 & \frac{2}{3} & 0 \\ 1 & 1 & \frac{1}{9} \end{bmatrix},$$

leads to the DESIRE method

$$M = \left[\begin{array}{ccc|ccc} \frac{7}{6} & -\frac{1}{3} & 0 & 1 & -\frac{1}{2} & -\frac{1}{9} \\ \frac{4}{3} & -\frac{1}{6} & 0 & 1 & -\frac{1}{2} & -\frac{1}{9} \\ \frac{5}{3} & -\frac{2}{3} & \frac{1}{2} & 1 & -\frac{1}{2} & -\frac{1}{9} \\ \hline \frac{5}{3} & -\frac{2}{3} & \frac{1}{2} & 1 & -\frac{1}{2} & -\frac{1}{9} \\ \frac{5}{4} & -\frac{5}{2} & \frac{9}{4} & 0 & 0 & 0 \\ \frac{3}{2} & -6 & \frac{9}{2} & 0 & 0 & 0 \end{array}\right].$$

It is necessary to make it clear that not all ESIRK or DESIRE methods can be derived in this way. When representing these methods in a Nordsieck formulation, p free parameters are available. These free parameters only affect the B and thus V matrices. The underlying ESIRK method, that is the method not represented in Nordsieck form, is the same though. However for general linear methods with IRKS every non-singular matrix B has a unique A. Therefore, the last p free parameters have been chosen to ensure IRKS is guaranteed.

3.9.2 Stiffly accurate diagonally implicit methods

Runge-Kutta methods are called stiffly accurate if

$$A_s^T = b^T. \tag{3.9.18}$$

A-stable stiffly accurate Runge-Kutta methods are necessarily L-stable. To ensure L-stability $R(\infty) = 0$, given that (3.9.18) can be equivalently written as $A^T e_s = b$ which is $b^T A^{-1} = e_s^T$. Therefore,

$$\begin{aligned} R(\infty) &= 1 - b^T A^{-1} e \\ &= 1 - e_s^T e \\ &= 0. \end{aligned}$$

The stiffly accurate methods are particularly useful for the solution of singularly perturbed problems and also for differential algebraic equations. This is because the stages of a Runge-Kutta method applied to one of the above problems are solved so that the algebraic constraint is satisfied. For stiffly accurate methods the solution component is also the same as the last stage and therefore, also satisfies the algebraic constraint. Therefore, it is not necessary to project the solution onto the algebraic constraint as is necessary for other methods.

Stiffly accurate general linear methods also exist. Assuming $s = p + 1$ they require

$$A_{p+1}^T = B_1^T, \qquad U_{p+1}^T = V_1^T. \tag{3.9.19}$$

For methods in Nordsieck form it is possible to strengthen the requirement of stiff accuracy so that the scaled first derivative is equal to the scaled derivative of the last stage. This requires the extra conditions

$$B_2^T = e_{p+1}^T, \qquad V_2^T = 0. \tag{3.9.20}$$

In this subsection the conditions which are needed to guarantee an IRKS method has strict stiff accuracy are derived. It is of course a necessary requirement that the last abscissae $c_{p+1} = 1$. The last row of U and the first row of V for an IRKS method, given that $c_{p+1} = 1$, are

$$\begin{aligned} U_{p+1}^T &= E_{1\_p}^T - A_{p+1}^T C K, \\ V_1^T &= E_{1\_p}^T - B_1^T C K. \end{aligned}$$

So to satisfy (3.9.19) it is necessary to ensure that $A_{p+1}^T = B_1^T$ since the second condition automatically holds. The second row of V is given by

$$V_2^T = E_{0\_p-1}^T - E_{1\_p}^T K = 0.$$

Again to satisfy (3.9.20) it is necessary only to ensure that $B_2^T = e_{p+1}^T$. The second row of the IRKS condition $BA = XB$ is

$$B_2^T A = B_1^T - \beta_p B_{p+1}^T.$$

Thus if $\beta_p = 0$ then $B_2^T = e_{p+1}^T$ implies $B_1^T = A_{p+1}^T$. This leads to the following definition.

Definition 3.28 *An IRKS method has strict stiff accuracy if*

$$B_2^T = e_{p+1}^T, \qquad \beta_p = 0, \qquad c_{p+1} = 1.$$

Given that β_p and c_{p+1} are free parameters it is necessary to show only how to ensure that $B_2^T = e_{p+1}^T$ with the remaining free parameters. First the following lemma is required.

Lemma 3.29 *Given the shifting matrices J and K then*

$$K\exp(\lambda K^-)J\exp(-\lambda K^-) = I - \lambda K - e_{p+1}e_{p+1}^T.$$

Proof: Using Lemma 3.7 and (3.4.7), the doubly companion matrix can be written as

$$X(P\beta^{-1}, \beta) = \beta(K)\exp(\lambda K^-)(J + \lambda I)\exp(-\lambda K^-)\beta(K)^{-1}.$$

Multiply on the left by $\beta(K)^{-1}$ and the right by $\beta(K)$ and subtract λI from both sides gives

$$\beta(K)^{-1}X(P\beta^{-1}, \beta)\beta(K) - \lambda I = \exp(\lambda K^-)J\exp(-\lambda K^-).$$

It follows from Lemma 3.11 that

$$X(P, 1) - \lambda I = \exp(\lambda K^-)J\exp(-\lambda K^-).$$

Premultiply by K and the result follows. □

The following theorem shows how the free parameters are chosen so as to ensure the IRKS method has strict stiff stability.

Theorem 3.30 *An IRKS method has strict stiff accuracy if the matrix $\mathcal{T}^{-1}$ is unit lower triangular and given by*

$$\mathcal{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & T_{2,1}^{-1} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & T_{p-1,1}^{-1} & T_{p-1,2}^{-1} & \cdots & 1 & 0 \\ 0 & T_{p,1}^{-1} & T_{p,2}^{-1} & \cdots & T_{p,p-1}^{-1} & 1 \end{bmatrix},$$

where $T_{i,j}^{-1}$, $i = 2, 3, \ldots, p-1$, $j = 1, 2, \ldots, p-2$, are free parameters and $T_{p,j}^{-1}$ are given by

$$T_{p,j}^{-1} = \frac{\displaystyle\sum_{l=0}^{j-2} \binom{p-1-l}{j-2-l} \lambda^{j-l-2}\beta_l}{\displaystyle\sum_{l=0}^{p-1} \lambda^{p-1-l}\beta_l}, \qquad j = 2, 3, \ldots, p-1.$$

The last column of $\mathcal{T}^{-1}$ can alternatively be represented as

$$\mathcal{T}_{p+1}^{-1} = \frac{1}{e_2^T\beta(K)\Lambda} e_2^T\beta(K)\exp(\lambda K^-). \tag{3.9.21}$$

Proof: The matrix B is given by $B = \Psi\widetilde{B}W^{-1}$, since only diagonally implicit methods are considered $W^{-1} = I$ and because $\mathcal{T}^{-1}$ is unit lower triangular, it follows from (3.7.9) that

$$B = \Psi\mathcal{T}\Delta\left(\mathcal{T}^{-1}\Gamma\mathcal{T}\mathbf{U}\,(\Omega\mathcal{T})^{-1}\right)\mathbf{L}\,(\Omega\mathcal{T})^{-1},$$

where only the second row is of interest. It is necessary to first show that

$$e_2^T\Psi\mathcal{T} = \big(e_2^T\beta(K)\Lambda\big)e_{p+1}^T. \tag{3.9.22}$$

Substitute the decomposition of (3.4.7) and multiply on the right by $\mathcal{T}^{-1}$ is equivalent to

$$e_2^T\beta(K)\exp(\lambda K^-) = \big(e_2^T\beta(K)\Lambda\big)e_{p+1}^T\mathcal{T}^{-1}.$$

This follows immediately by substituting (3.9.21) for last row of $\mathcal{T}^{-1}$. A direct consequence of (3.9.22) is that only the last row of

$$\Delta\left(\mathcal{T}^{-1}\Gamma\mathcal{T}\mathbf{U}\,(\Omega\mathcal{T})^{-1}\right)\mathbf{L}\,(\Omega\mathcal{T})^{-1},$$

needs to be considered. This is

$$\begin{aligned}
e_{p+1}^T\Delta\left(\mathcal{T}^{-1}\Gamma\mathcal{T}\mathbf{U}\,(\Omega\mathcal{T})^{-1}\right)\mathbf{L}\,(\Omega\mathcal{T})^{-1} &= e_{p+1}^T\mathcal{T}^{-1}\Gamma\mathcal{T}\mathbf{U}\,(\Omega\mathcal{T})^{-1}\mathbf{L}\,(\Omega\mathcal{T})^{-1}\\
&= e_{p+1}^T\mathcal{T}^{-1}\Gamma\mathcal{T}(\Omega\mathcal{T})^{-1}\\
&= e_{p+1}^T\mathcal{T}^{-1}\Gamma\Omega^{-1}.
\end{aligned}$$

To ensure that $B_2^T = e_{p+1}^T$, it is required that

$$e_{p+1}^T\mathcal{T}^{-1}\Gamma\Omega^{-1} = \frac{1}{e_2^T\beta(K)\Lambda}e_{p+1}^T,$$

which simplifies to

$$e_2^T\beta(K)\exp(\lambda K^-)\Gamma = e_{p+1}^T\Omega. \tag{3.9.23}$$

Now consider each side of the above equation separately. Substituting (3.4.7) and (3.7.6) into the right hand side gives

$$\begin{aligned}
e_{p+1}^T\Omega &= e_{p+1}^TC\big(\beta(K)e_{p+1}e_1^T + K\Psi\big)\\
&= e_1^TE\big(\beta(K)e_{p+1}e_1^T + K\beta(K)\exp(\lambda K^-)\big)\\
&= e_1^TE\beta(K)e_{p+1}e_1^T + e_1^TE\beta(K)K\exp(\lambda K^-).
\end{aligned} \tag{3.9.24}$$

Given that $\beta(K)$ and $\exp(\lambda K^-)$ are upper triangular matrices it follows that the first element of the vector $e_2^T\beta(K)\exp(\lambda K^-)$ will be zero and therefore the first row of Γ given by (3.7.7) will play no part in (3.9.23). Given the vector δ (3.7.3), except for the first element has a special form, Γ except for the first row can be represented using as

$$\begin{aligned}
\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}]F\begin{bmatrix}\mathrm{O}\\ \mathrm{I}\end{bmatrix}[\mathrm{O}\,\mathrm{I}] + \delta e_1^T &\equiv \exp(-\lambda K^-)E\exp(\lambda K^-) + J\exp(-\lambda K^-)E\exp(\lambda K^-)e_{p+1}e_1^T\\
&= \exp(-\lambda K^-)E\exp(\lambda K^-) + J\exp(-\lambda K^-)E\Lambda e_1^T.
\end{aligned}$$

Substituting into the left hand side of (3.9.23) gives

$$\begin{aligned}
e_2^T\beta(K)\exp(\lambda K^-)\Gamma &= e_2^T\beta(K)E\exp(\lambda K^-) + e_2^T\beta(K)\exp(\lambda K^-)J\exp(-\lambda K^-)E\Lambda e_1^T\\
&= e_1^TE\beta(K)K\exp(\lambda K^-) + e_1^T\beta(K)K\exp(\lambda K^-)J\exp(-\lambda K^-)E\Lambda e_1^T\\
&= e_1^TE\beta(K)K\exp(\lambda K^-) + e_1^T\beta(K)(I - \lambda K - e_{p+1}e_{p+1}^T)E\Lambda e_1^T\\
&= e_1^TE\beta(K)K\exp(\lambda K^-) + e_1^T\beta(K)Ee_{p+1}e_1^T - e_1^T\beta(K)e_{p+1}E\Lambda e_1^T\\
&= e_1^TE\beta(K)K\exp(\lambda K^-) + e_1^TE\beta(K)e_{p+1}e_1^T,
\end{aligned} \tag{3.9.25}$$

since $\beta_p = 0$. Comparing (3.9.24) and (3.9.25) the result follows. □

A consequence of this proof is that only the first element of the first row of Γ plays any part in constructing methods. It must also be mentioned that in the Runge-Kutta case stiff accuracy ensures the method has L-stability. This is not the case for IRKS methods. In order to ensure the method has L-stability the free parameter ϵ must be chosen so that $\epsilon = 0$.

The following two methods are stiffly accurate diagonally implicit methods. The first method is explicit with error constant equal to zero. This means the method could be interpreted as order three. Note also that the last column of V is zero which makes the method have a similar property to the Almost Runge-Kutta methods. Choosing the free parameters of the method as follows

$$p = 2, \quad \lambda = 0, \quad \epsilon = \tfrac{1}{6}, \quad c = [\ \tfrac{1}{3}\ \ \tfrac{2}{3}\ \ 1\ ]^T, \quad \beta = [\ 0\ \ \tfrac{1}{2}\ \ 1\ ]^T, \quad T = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}, \quad W = I,$$

leads to the following method

$$M = \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & \frac{1}{3} & \frac{1}{18} \\ \frac{1}{2} & 0 & 0 & 1 & \frac{1}{6} & \frac{1}{18} \\ 0 & \frac{3}{4} & 0 & 1 & \frac{1}{4} & 0 \\ \hline 0 & \frac{3}{4} & 0 & 1 & \frac{1}{4} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & -3 & 2 & 0 & -2 & 0 \end{array}\right].$$

The second method is implicit and is L-stable. Note that V is rank one which is a desirable property but can only be achieved in a few special cases. Choosing the free parameters of the method as follows

$$p = 2, \quad \lambda = \tfrac{1}{4}, \quad \epsilon = 0, \quad c = [\ \tfrac{1}{4}\ \ \tfrac{1}{2}\ \ 1\ ]^T, \quad \beta = [\ 0\ \ \tfrac{1}{4}\ \ 1\ ]^T, \quad T = I, \quad W = I,$$

leads to the following method

$$M = \left[\begin{array}{ccc|ccc} \frac{1}{4} & 0 & 0 & 1 & 0 & -\frac{1}{32} \\ \frac{1}{6} & \frac{1}{4} & 0 & 1 & \frac{1}{12} & -\frac{1}{24} \\ \frac{1}{6} & \frac{1}{2} & \frac{1}{4} & 1 & \frac{1}{12} & -\frac{1}{24} \\ \hline \frac{1}{6} & \frac{1}{2} & \frac{1}{4} & 1 & \frac{1}{12} & -\frac{1}{24} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 & 0 & 0 \end{array}\right].$$

The example methods listed in Appendix III have the strong stiff accuracy property and are used for comparisons against some well known Runge-Kutta and Adams methods in Chapter 5. These methods perform extremely well and since they have the added advantage of being useful for differential algebraic equations it is likely that this class of methods will be used in a code based on the IRKS methods.

3.10 Some example methods

Even though several methods have already been given three further examples are included. The first method is explicit. Note that the transformation matrix T has been chosen so that $T = I$. This guarantees the $\dot{V}$ matrix found by removing the first row and column of V is strictly upper triangular. An advantage of this choice is that the method will be zero-stable for any choice of variable mesh, see Section 4.6 for more details.

Choosing the free parameters as follows

$$p=2, \quad \lambda=0, \quad \epsilon=\tfrac{1}{6}, \quad c=[\ 0 \ \ \tfrac{1}{2} \ \ 1\ ]^T, \quad \beta=[\ \tfrac{1}{4} \ \ \tfrac{1}{2} \ \ 1\ ]^T, \quad T=I, \quad W=I,$$

leads to the method

$$M=\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & -\frac{1}{2} & \frac{1}{8} \\ \frac{1}{3} & \frac{2}{3} & 0 & 1 & 0 & \frac{1}{6} \\ \hline \frac{7}{12} & \frac{1}{3} & \frac{1}{4} & 1 & -\frac{1}{6} & \frac{1}{12} \\ -\frac{1}{6} & \frac{2}{3} & \frac{1}{2} & 0 & 0 & \frac{1}{6} \\ -1 & 0 & 1 & 0 & 0 & 0 \end{array}\right].$$

The second method is diagonally implicit and is L-stable. The transformation matrix T was chosen so that $T=I$, which again ensures that $\dot{V}$ is strictly upper triangular. Choosing the free parameters as follows

$$p=2, \quad \lambda=\tfrac{1}{4}, \quad \epsilon=0, \quad c=[\ 0 \ \ \tfrac{1}{2} \ \ 1\ ]^T, \quad \beta=[\ \tfrac{1}{8} \ \ \tfrac{1}{4} \ \ 1\ ]^T, \quad T=I, \quad W=I,$$

leads to the method

$$M=\left[\begin{array}{ccc|ccc} \frac{1}{4} & 0 & 0 & 1 & -\frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 1 & 0 & \frac{1}{8} \\ \hline \frac{1}{2} & -\frac{1}{8} & \frac{1}{2} & 1 & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & \frac{1}{4} \\ 0 & -2 & 2 & 0 & 0 & 0 \end{array}\right].$$

The third method is implicit. There are three things to notice about the following method. The first thing to notice is that the last columns of the U and V matrices are zero. This means the quantity $h^2y''_n$ is not needed in the computation and the method is therefore reducible in the number of output approximations. The second thing to notice is that the elements in the second column of U and V_{12} are identical which means the method has effective order, since the stage order equals the order. The third thing to notice is that the diagonal and sub-diagonals of A are identical for a method of order two see Section 2.11. This suggests the method is the composition of a simpler method. Choosing the free parameters as follows

$$p=2, \quad \lambda=\tfrac{1}{6}, \quad \epsilon=\tfrac{1}{216}, \quad c=[\ \tfrac{1}{3} \ \ \tfrac{2}{3} \ \ 1\ ]^T, \quad \beta=[\ 0 \ \ \tfrac{1}{6} \ \ 1\ ]^T, \quad T=I, \quad W=I,$$

leads to the method

$$M=\left[\begin{array}{ccc|ccc} \frac{1}{6} & 0 & 0 & 1 & \frac{1}{6} & 0 \\ \frac{1}{3} & \frac{1}{6} & 0 & 1 & \frac{1}{6} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & 1 & \frac{1}{6} & 0 \\ \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & 1 & \frac{1}{6} & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -3 & 3 & 0 & 0 & 0 \end{array}\right].$$

In fact this method is the composition of the trapezoidal rule over three steps with stepsize $h/3$, see Section 2.11 for the general formulae of the composition method.

3.11 Searching for good methods

It is very difficult to define what it means to be a “good” or “optimal” method. In this section certain characteristics a good or optimal method may posses are discussed. It would for example be desirable for a method to have coefficients with small magnitudes. This reduces the loss of significance that can result when methods have large coefficients. For almost all general linear methods only isolated examples can be constructed. This is generally because methods are constructed by finding numerical solutions to complicated systems of nonlinear equations which describe the order and often stability conditions. If there were several solutions available it would only then be possible to choose the method with the smallest coefficients.

One of the advantages of IRKS methods which cannot be overstated is that the methods can be derived using only linear operations. This makes it possible to search for certain choices of the free parameters which lead to the coefficients of the method having small magnitudes. In fact methods where all coefficients have magnitude less than or equal to one can be found for methods of high order.

In Runge-Kutta theory, methods are regarded as optimal if the error coefficients of the elementary differentials are very small. A lot of work has been done in constructing optimal Runge-Kutta methods for example by Dormand and Prince [63] and Verner [119]. For general linear methods the error coefficients of the high order elementary differentials are much more complicated to analyse. This is because the error coefficients depend on the choice of starting procedure as well as the method. In Section 2.6 the underlying one step method was discussed. The underlying one step method provides one possible interpretation of the error of strictly stable general linear methods. The basic idea (see Section 2.6 for more details) is to find a starting procedure S^* and the underlying one step method ϕ such that

$$S^*M = \phi S^*.$$

There is freedom available in the choice of S^* and ϕ. This freedom is eliminated if the initial condition is assumed to be exact. The commutative expression above can be reinterpreted in terms of B-series and thus generating functions. The generating functions for $\phi(t)$ and $S^*(t)$ are given by

$$\begin{aligned}\phi(t) &= v^T\big(B\xi D_1(t) - R(t)\big),\\ S^*(t) &= \big(I-\widetilde{V}\big)^{-1}\big(B\xi D_1(t) - R(t)\big).\end{aligned}$$

Since the IRKS methods have high stage order only some of the trees are independent. It has been observed that the trees which are independent in $\phi(t)$ are the same as those discussed in Section 2.10. However, this is yet to be proved. Nevertheless this limits the number of conditions that need to be considered when attempting to minimise the error coefficients of the underlying one step method ϕ.

The basic approach used to find an IRKS method with an underlying one step method with minimised error coefficients is as follows:

- Choose initial choice of free parameters.
- Compute the IRKS method.
- Compute the independent coefficients of the underlying one step method.
- Weight the independent coefficients.
- Repeat process until the weighting of the independent coefficients is minimised.

In Appendix I a Matlab code is included which minimises the error coefficients of the underlying one step method. It would be likely that better methods could be found if more sophisticated searching strategies were used. In order to increase the speed of the search the function which generates $R(t)$ has explicit expressions for each tree up to the required order, rather than using the recursive approach discussed in Section 2.5. In Appendix III methods of orders two through to five are given. These have been constructed so that underlying one step method has minimal error coefficients.

In both of the approaches described above the resulting coefficients of the method are generally fractions with very large numerators and denominators. This makes the method appear cumbersome and difficult to present. It would be much more elegant if methods could be constructed with small numerators and denominators. However, it is not yet known how this can be accomplished.

3.12 Other approaches

In Section 3.7 a canonical class of methods with IRKS was developed. Since this class of methods is canonical it is necessary that the implicit IRKS methods derived in [32], the explicit IRKS methods derived in [117] and the transformed IRKS methods derived in [50] are special cases of the methods found in this chapter. It was shown in [50] that the methods constructed in [32] and [117] were special cases, therefore it is only necessary to show that the methods from [50] are a special case of the methods derived in this chapter.

This section should be read in conjunction with [50]. To make this section self contained would require an unnecessary amount of work to be repeated in a slightly less general form. From Section 3 of [50], the transformed order conditions take the form

$$\begin{aligned}
\exp\left(\frac{c\widehat{z}}{1+\theta\widehat{z}}\right) &= \widehat{z}\widehat{A}\exp\left(\frac{c\widehat{z}}{1+\theta\widehat{z}}\right)\widehat{\psi}(\widehat{z}) + \widehat{U}\widehat{\psi}(\widehat{z})^{-1}\widehat{Z} + O(\widehat{z}^{p+1}),\\
\exp\left(\frac{\widehat{z}}{1+\theta\widehat{z}}\right)\widehat{\psi}(\widehat{z})^{-1}\widehat{Z} &= \widehat{z}\widehat{B}\exp\left(\frac{c\widehat{z}}{1+\theta\widehat{z}}\right) + \widehat{V}\widehat{\psi}(\widehat{z})^{-1}\widehat{Z} + O(\widehat{z}^{p+1}).
\end{aligned}$$

To return to the original method make the substitution $\widehat{z} = z/(1-\theta z)$. First note that the relationship between $\widehat{Z}$ and Z can be written in the form

$$\widehat{Z} = (1-\theta\widehat{z})^{-p}\exp(-\theta K^{-})Z.$$

The order conditions are therefore,

$$\begin{aligned}
\exp(cz) &= z(\widehat{A}+\theta I)\exp(cz) + \widehat{U}\exp(-\theta K^{-})\psi(z)^{-1}Z + O(z^{p+1}),\\
\exp(z)\psi(z)^{-1}Z &= \exp(\theta K^{-})\widehat{B}\exp(cz) + \exp(\theta K^{-})\widehat{V}\exp(-\theta K^{-})\psi(z)^{-1}Z + O(z^{p+1}),
\end{aligned}$$

where

$$\psi(z) = \widehat{\psi}\left(\frac{z}{1-\theta z}\right)^{-1}(1-\theta z)^{1-p}.$$

Since the coefficients of $\widehat{\psi}(\widehat{z})$ are free parameters of the method, it is equivalent to choose the coefficients of $\psi(z)$ instead. It follows from

$$\psi(z)Z = \psi(K)Z + O(z^{p+1}),$$

that the coefficients of the original method are

$$\begin{aligned}
A &= \widehat{A} + \theta I,\\
B &= \psi(K)\exp(\theta K^{-})\widehat{B},\\
U &= \widehat{U}\exp(-\theta K^{-})\psi(K)^{-1},\\
V &= \psi(K)\exp(\theta K^{-})\widehat{V}\exp(-\theta K^{-})\psi(K)^{-1}.
\end{aligned}$$

The IRKS conditions lead to the formulae

$$\begin{aligned}
BA &\equiv \psi(K)\exp(\theta K^{-})(J+\theta I)\exp(-\theta K^{-})\psi(K)^{-1}B,\\
BU &\equiv \psi(K)\exp(\theta K^{-})J\exp(-\theta K^{-})\psi(K)^{-1}V - V\psi(K)\exp(\theta K^{-})J\exp(-\theta K^{-})\psi(K)^{-1},\\
&\equiv \psi(K)\exp(\theta K^{-})(J+\theta I)\exp(-\theta K^{-})\psi(K)^{-1}V - \\
&\quad V\psi(K)\exp(\theta K^{-})(J+\theta I)\exp(-\theta K^{-})\psi(K)^{-1}.
\end{aligned}$$

Now it is sufficient to show that

$$X(\theta\psi^{-1},\psi) = \psi(K)\exp(\theta K^{-})(J+\theta I)\exp(-\theta K^{-})\psi(K)^{-1}, \tag{3.12.1}$$

where $\theta(w) = (1-\theta w)^{p+1}$. This follows directly from Lemmas 3.7 and 3.11. Therefore $X(\theta\psi^{-1},\psi)$ has a one point spectrum

$$\sigma\big(X(\theta\psi^{-1},\psi)\big) = \{\theta\}.$$

The first IRKS condition can be strengthened to an equality using Lemma 3.13. Also comparing (3.12.1) with Lemma 3.11 implies that

$$X(\theta,1) = \exp(\theta K^{-})(J+\theta I)\exp(-\theta K^{-}),$$

which can be regarded as a special case of Lemma 3.7. All methods which were derived in the early papers [32], [50] and [117] can be derived using the techniques discussed in Section 3.7. What is especially interesting is that the use of the permutation matrix included in Lemma 3.23 is never required, even for methods in [32] where the matrix $\dot{V}$ was strictly lower triangular. Whether the use of the permutation is ever required is still unclear.

3.13 Composition of methods

Consider the composition of two methods with IRKS. Let M_1 and M_2 denote the two IRKS methods with s_1 and s_2 stages respectively. First apply M_1 to $y^{[n-1]}$ to generate $y^{[n]}$, then apply M_2 to $y^{[n]}$ to generate $y^{[n+1]}$. The method which generates $y^{[n+1]}$ from $y^{[n-1]}$ in one step of length $2h$ is given by

$$M_1 \circ M_2 = \left[\begin{array}{cc|c} A_1 & 0 & U_1\\ U_2B_1 & A_2 & U_2V_1\\ \hline V_2B_1 & B_2 & V_2V_1 \end{array}\right].$$

Reinterpreting the composition method as a single step of a method of size h (compare (2.11.1)) yields

$$M_1 \circ M_2 = \left[\begin{array}{cc|c} \frac{1}{2}A_1 & 0 & U_1H\\ \frac{1}{2}U_2B_1 & \frac{1}{2}A_2 & U_2V_1H\\ \hline \frac{1}{2}H^{-1}V_2B_1 & \frac{1}{2}H^{-1}B_2 & H^{-1}V_2V_1H \end{array}\right],$$

where H is the diagonal scaling matrix defined as

$$H = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2^2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{2^{p-2}} & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{2^{p-1}} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{2^p} \end{bmatrix}.$$

Does the composition method have IRKS? To help answer this question the following two lemmas are useful.

Lemma 3.31 *Given a diagonal rescaling matrix H, then*

$$H^{-1}\exp(\lambda K^-)J\exp(-\lambda K^-)H = 2\exp\left(\frac{1}{2}\lambda K^-\right)J\exp\left(-\frac{1}{2}\lambda K^-\right).$$

Proof: Expressing as a Taylor series, gives

$$H^{-1}\left(I+\sum_{i=1}^{p}\frac{(\lambda K^-)^i}{i!}\right)J\left(I+\sum_{j=1}^{p}\frac{(-\lambda K^-)^j}{j!}\right)H = 2\left(I+\sum_{i=1}^{p}\frac{(\lambda K^-)^i}{2^i i!}\right)J\left(I+\sum_{j=1}^{p}\frac{(-\lambda K^-)^j}{2^j j!}\right).$$

When expanded out

$$\begin{aligned} H^{-1}&\left(J+\sum_{i=1}^{p}\frac{\lambda^i}{i!}(K^-)^iJ+\sum_{j=1}^{p}\frac{(-\lambda)^j}{j!}J(K^-)^j+\sum_{i=1}^{p}\sum_{j=1}^{p}(-1)^j\frac{\lambda^{i+j}}{i!j!}(K^-)^iJ(K^-)^j\right)H \\ &= 2\left(J+\sum_{i=1}^{p}\frac{\lambda^i}{2^i i!}(K^-)^iJ+\sum_{j=1}^{p}\frac{(-\lambda)^j}{2^j j!}J(K^-)^j+\sum_{i=1}^{p}\sum_{j=1}^{p}(-1)^j\frac{\lambda^{i+j}}{2^{i+j}i!j!}(K^-)^iJ(K^-)^j\right). \end{aligned}$$

Since $H^{-1}JH = 2J$ it remains to verify that

$$H^{-1}(K^-)^iJ(K^-)^jH = \frac{1}{2^{i+j-1}}(K^-)^iJ(K^-)^j. \qquad (3.13.1)$$

The following relations are useful

$$\begin{aligned} (K^-)^jH &= K^jHJ^j(K^-)^j, \\ H^{-1}(K^-)^i &= (K^-)^iJ^iH^{-1}K^i. \end{aligned}$$

Substituting into the left hand side of (3.13.1) and using the proof of Theorem 3.9 gives

$$\begin{aligned} H^{-1}(K^-)^iJ(K^-)^jH &= (K^-)^iJ^iH^{-1}K^iJK^jHJ^j(K^-)^j \\ &= (K^-)^iJ^iH^{-1}K^{i+j-1}HJ^j(K^-)^j \\ &= \frac{1}{2^{i+j-1}}(K^-)^iJ^iK^{i+j-1}J^j(K^-)^j \\ &= \frac{1}{2^{i+j-1}}(K^-)^iJ(K^-)^j, \end{aligned}$$

since $(K^-)^i, (K^-)^j$ and J are all nilpotent. □

The next lemma shows an interesting connection between two doubly companion matrices.

Lemma 3.32 *Let $X(P\beta^{-1},\beta)$ be a doubly companion matrix with a $(p+1)$-fold eigenvalue λ, with characteristic polynomial*

$$P(w)=(w-\lambda)^{p+1}.$$

Then there exists another doubly companion matrix $X(\bar{P}\bar{\beta}^{-1},\bar{\beta})$ given by

$$X(\bar{P}\bar{\beta}^{-1},\bar{\beta})=\frac{1}{2}H^{-1}X(P\beta^{-1},\beta)H, \qquad (3.13.2)$$

with a $(p+1)$-fold eigenvalue $\frac{\lambda}{2}$, therefore the characteristic polynomial is

$$\bar{P}(w)=\left(w-\frac{1}{2}\lambda\right)^{p+1},$$

and where

$$H^{-1}\beta(K)=\bar{\beta}(K)H^{-1}.$$

Proof: Substitute the decomposed form of the doubly companion matrix (3.4.6), noting that Ψ can also be decomposed as in (3.4.7), into (3.13.2), then

$$\begin{aligned}
X(\bar{P}\bar{\beta}^{-1},\bar{\beta}) &= \frac{1}{2}H^{-1}\beta(K)\exp(\lambda K^-)(J+\lambda I)\exp(-\lambda K^-)\beta(K)^{-1}H\\
&= \frac{1}{2}\bar{\beta}(K)H^{-1}\exp(\lambda K^-)(J+\lambda I)\exp(-\lambda K^-)H\bar{\beta}(K)^{-1}\\
&= \frac{1}{2}\bar{\beta}(K)H^{-1}\exp(\lambda K^-)J\exp(-\lambda K^-)H\bar{\beta}(K)^{-1}+\frac{1}{2}\lambda I\\
&= \bar{\beta}(K)\exp\left(\frac{1}{2}\lambda K^-\right)J\exp\left(-\frac{1}{2}\lambda K^-\right)\bar{\beta}(K)^{-1}+\frac{1}{2}\lambda I\\
&= \bar{\beta}(K)\exp\left(\frac{1}{2}\lambda K^-\right)\left(J+\frac{1}{2}\lambda I\right)\exp\left(-\frac{1}{2}\lambda K^-\right)\bar{\beta}(K)^{-1},
\end{aligned}$$

which is the required result. □

A similar condition to the first IRKS condition (3.3.1) for the composition method is

$$\begin{aligned}
&\begin{bmatrix} \frac{1}{2}H^{-1}V_2B_1 & \frac{1}{2}H^{-1}B_2 \end{bmatrix}\begin{bmatrix} \frac{1}{2}A_1 & 0 \\ \frac{1}{2}U_2B_1 & \frac{1}{2}A_2 \end{bmatrix}\\
&\qquad = [\ \tfrac{1}{4}H^{-1}V_2B_1A_1+\tfrac{1}{4}H^{-1}B_2U_2B_1 \quad \tfrac{1}{4}H^{-1}B_2A_2\ ]\\
&\qquad \equiv [\ \tfrac{1}{4}H^{-1}V_2X_1B_1+\tfrac{1}{4}H^{-1}(X_2V_2-V_2X_2)B_1 \quad \tfrac{1}{4}H^{-1}X_2B_2\ ]\\
&\qquad = [\ \tfrac{1}{4}H^{-1}XV_2B_1 \quad \tfrac{1}{4}H^{-1}XB_2\ ]\\
&\qquad = Y[\ \tfrac{1}{2}H^{-1}V_2B_1 \quad \tfrac{1}{2}H^{-1}B_2\ ],
\end{aligned}$$

assuming that $X_1=X_2=X$ and Y is a doubly companion matrix such that $Y=\frac{1}{2}H^{-1}XH$ as in Lemma 3.32. A similar condition to the second IRKS condition (3.3.2) for the composition method is

$$\begin{aligned}
[\ \tfrac{1}{2}H^{-1}V_2B_1 \quad \tfrac{1}{2}H^{-1}B_2\ ]\begin{bmatrix} U_1H \\ U_2V_1H \end{bmatrix} &= \frac{1}{2}H^{-1}V_2B_1U_1H+\frac{1}{2}H^{-1}B_2U_2V_1H\\
&\equiv \frac{1}{2}H^{-1}V_2(X_1V_1-V_1X_1)H+\frac{1}{2}H^{-1}(X_2V_2-V_2X_2)V_1H\\
&\equiv \frac{1}{2}H^{-1}XV_2V_1H-\frac{1}{2}H^{-1}V_2V_1XH\\
&\equiv YH^{-1}V_2V_1H-H^{-1}V_2V_1HY,
\end{aligned}$$

again assuming that $X_1 = X_2 = X$ and Lemma 3.32 holds. The final IRKS condition for the composition method is

$$\det(wI - H^{-1}V_2V_1H) = w^p(w-1).$$

This will not always hold for two IRKS methods, however when $T = I$ for both methods it will always hold. This is because the matrix left after the first row and column of V_1 and V_2 has been removed will be strictly upper triangular. This implies the matrix formed after removing the first row and column of $H^{-1}V_2V_1H$ will also be strictly upper triangular. To determine whether the last IRKS condition for the composition method is satisfied in the general case see Section 4.6.

It will be interesting to consider in the general case when the doubly companion matrices of two methods are not the same what the resulting stability region of the composition method is like. It may well be the case that methods with certain properties are more likely to be suitable than others when composed together. Note similar results can be derived for the composition of more than two methods.

3.14 The non-existence of parallel methods

There are four types of DIMSIMS; these include the combination of stiff and non-stiff in a sequential or parallel environment see Subsection 2.12.7. So far in this thesis only methods for the application of non-stiff and stiff problems using a sequential computer architecture has been discussed. For applications using parallel computers, the matrix $A = 0$ for non-stiff problems and $A = \lambda I$ for the stiff problems. Do parallel methods exist with the IRKS property?

Theorem 3.33 *Parallel general linear methods with the IRKS property do not exist for non-stiff problems and exist only for orders one and two if $A = \lambda I$.*

Proof: The stability matrix when $A = \lambda I$ is given by

$$M(z) = V + \frac{z}{1-\lambda z}BU.$$

Since IRKS is required, $\operatorname{tr}(M(z)) = \exp(z) + O(z^{p+1})$. However, since $\operatorname{tr}(BU) = 1$ it follows that

$$\operatorname{tr}(M(z)) = 1 + \frac{z}{1-\lambda z}.$$

Expanding using the Binomial theorem it follows that the maximum obtainable order is 2 when $\lambda = \frac{1}{2}$. In the case when $\lambda = 0$ the method is reducible. □

Even though a set of parallel methods with IRKS does not exist, a more general class for the application of stiff problems may exist when the matrix A is not singly implicit but can have $p+1$ different eigenvalues. Methods with this property will be considered in future work.

CHAPTER 4

Implementation issues

When any new numerical method is constructed it is important to compare it with existing numerical solvers. This enables the advantages and disadvantages the method is claimed to have to be verified. Before doing so, it is essential to discuss certain choices regarding how the method is implemented.

4.1 Framework

Very little literature exists on the implementation of general linear methods which are neither Runge-Kutta nor linear multistep methods. It is for this reason that very standard approaches to the implementation of the IRKS methods will be used. That is to say, at this early stage in the development of the IRKS methods, to best ensure that the methods compare well with the traditional methods, certain choices available in the implementation should be kept as similar to those of the traditional methods as possible. For example, the stepsize control procedure could be identical. When this is the case, there is more evidence that the results can be interpreted as results of the IRKS methods rather than the implementation choices. With this in mind it should also be remarked that there may well be features of the IRKS methods that could be taken advantage of when implemented. Some of these will be alluded to, but this will be largely left for future work. The five main issues with regards to implementation of the IRKS methods are:

- *Starting procedures*: Constructing the initial Nordsieck vector.
- *Error estimation*: Ensuring the local error estimate is asymptotically correct and requires no extra stages.
- *Variable stepsize*: Deciding how the method varies stepsize and the type of controller used to estimate the new stepsize.
- *Variable order*: Estimating the local error of the method with order one greater than the current order.
- *Computing the stages*: Taking advantage of the fact that the matrix A has a one point spectrum and the fact that the method has high stage order.

Each of these issues will be addressed; however the main emphasis will be on the implementation of explicit methods. The thesis of Huang [83] deals in much more detail with the implementation issues which are particular to implicit methods. Also features which are particular to the implicit IRKS methods are taken advantage of in the implementation.

4.2 Starting procedures

General purpose multivalue codes generally include a means of varying the order as the integration proceeds. This initially allows the order to start at say one or two which requires only known information. This is the strategy that the general purpose codes based on IRKS methods will adopt. However, it is also necessary to compare particular methods of fixed order with currently available methods, and this requires a starting procedure. The first generalised Runge-Kutta methods used as starting procedures were developed by Gear [71] in 1980. These were used as a means of starting the integration using a sufficiently high order multistep method. They are also useful for problems with frequent discontinuities or sudden large increases in the derivatives as this does not require building the order up slowly. For the IRKS methods it is necessary to approximate the initial Nordsieck vector

$$y^{[0]} = \begin{bmatrix} y(x_0) \\ hy'(x_0) \\ h^2y''(x_0) \\ \vdots \\ h^py^{(p)}(x_0) \end{bmatrix} + O(h^{p+1}).$$

It is often the case for other general linear methods that the starting procedure can be used to construct the initial information a certain distance from the initial condition. In other words, the starting procedure also performs the first step of the integration. This approach will not be adopted; the starting procedure will approximate the Nordsieck vector at the initial point.

To construct an approximation to the initial Nordsieck vector, $p+1$ different Runge-Kutta type methods need to be found. The first two components of the Nordsieck vector are trivial. The first is the initial condition, the second is the initial condition substituted into the differential equation and scaled by the stepsize. When the problem is non-stiff, for low orders it is easy to find explicit methods to do the job. As the order increases this becomes more and more difficult. Implicit methods can be easily constructed for this purpose. For non-stiff problems the resulting system of non-linear equations can be solved using fixed point iteration, instead of the modified Newton's method used in the stiff case. In both the non-stiff and stiff case it is possible to construct singly implicit Runge-Kutta type methods with stage order equal to order. Methods of this type are special cases of the order conditions

$$\begin{aligned} \exp(cz) &= zA\exp(cz) + UZ + O(z^{p+1}), \\ Z &= zB\exp(cz) + VZ + O(z^{p+1}), \end{aligned}$$

where the vector of abscissae c are free parameters. These order conditions follow directly from those derived in Theorem 3.1 except the Nordsieck vector computed at the end of the step is just the Nordsieck vector at the beginning of the step. The U and V matrices must be non-zero in the first column only as this allows only the initial condition to be used to compute the Nordsieck vector. From the above order conditions it follows that the matrices U and V, are given by

$$U = C - ACK, \tag{4.2.1}$$

$$V = I - BCK. \tag{4.2.2}$$

The U and V matrices need to be chosen so that

$$U = ee_1^T, \qquad V = e_1e_1^T.$$

For $U = ee_1^T$ it is sufficient for the relation

$$AC = CJ,$$

to hold. Substituting into (4.2.1) gives

$$\begin{aligned} U &= C - ACK, \\ &= C(I - JK), \\ &= ee_1^T. \end{aligned}$$

For $V = e_1 e_1^T$ it is sufficient for the relation

$$BC = J,$$

to hold. Substituting into (4.2.2) gives

$$\begin{aligned} V &= I - BCK, \\ &= I - JK, \\ &= e_1 e_1^T. \end{aligned}$$

In Appendix I a Maple code is included to derive the starting methods. The procedure requires the order p and a vector of abscissae c of length $p+1$. The following starting methods of orders two, three and four have abscissae evenly spaced in $[0, 1]$ and are given by

$$\left[\begin{array}{ccc|c} 0 & 0 & 0 & 1 \\ \frac{1}{8} & \frac{1}{2} & -\frac{1}{8} & 1 \\ -\frac{1}{2} & 2 & -\frac{1}{2} & 1 \\ \hline 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ -3 & 4 & -1 & 0 \end{array}\right],$$

$$\left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ \frac{5}{36} & \frac{2}{9} & -\frac{1}{36} & 0 & 1 \\ \frac{1}{3} & -\frac{2}{9} & \frac{7}{9} & -\frac{2}{9} & 1 \\ \frac{5}{4} & -3 & \frac{15}{4} & -1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1 & 0 \\ 18 & -45 & 36 & -9 & 0 \end{array}\right],$$

$$\left[\begin{array}{ccccc|c} 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{49}{576} & \frac{67}{288} & -\frac{5}{48} & \frac{13}{288} & -\frac{5}{576} & 1 \\ \frac{1}{72} & \frac{11}{18} & -\frac{1}{3} & \frac{5}{18} & -\frac{5}{72} & 1 \\ -\frac{27}{64} & \frac{75}{32} & -\frac{45}{16} & \frac{69}{32} & -\frac{33}{64} & 1 \\ -\frac{37}{18} & \frac{80}{9} & -\frac{38}{3} & \frac{80}{9} & -\frac{37}{18} & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{25}{3} & 16 & -12 & \frac{16}{3} & -1 & 0 \\ \frac{140}{3} & -\frac{416}{3} & 152 & -\frac{224}{3} & \frac{44}{3} & 0 \\ -160 & 576 & -768 & 448 & -96 & 0 \end{array}\right].$$

4.3 Error propagation

In order to understand how best to vary stepsize it is necessary first to consider how the errors in general linear methods, with stage order equal to order, propagate. It is possible to understand and control how the error propagates up to order $O(h^{p+2})$. This can then be used to control not only the stepsize but also the order of the method. As this is ongoing research in collaboration with J. C. Butcher and Z. Jackiewicz, in this thesis only error propagation up to $O(h^{p+1})$ will be considered. Many of the results in the next few sections are similar to those derived in [45]. These results are included so that particular properties of the IRKS methods which follow from these results can be included. For the purpose of error propagation it is often more appropriate to represent the method (2.1.2) as follows

$$\left[\begin{array}{c|c} A & U \\ \hline B & V \end{array}\right] = \left[\begin{array}{c|cc} A & e & \mathsf{U} \\ \hline \mathsf{b}^T & 1 & \mathsf{v}^T \\ \mathsf{B} & 0 & \mathsf{V} \end{array}\right]. \tag{4.3.1}$$

This means the computations performed in one step are

$$\begin{aligned} Y^{[n]} &= hAf\big(Y^{[n]}\big) + ey_{n-1} + \mathsf{U}\mathsf{y}^{[n-1]}, \\ y_n &= h\mathsf{b}^T f\big(Y^{[n]}\big) + y_{n-1} + \mathsf{v}^T\mathsf{y}^{[n-1]}, \\ \mathsf{y}^{[n]} &= h\mathsf{B}f\big(Y^{[n]}\big) + \mathsf{V}\mathsf{y}^{[n-1]}. \end{aligned} \tag{4.3.2}$$

Note that the original incoming and outgoing information satisfies

$$y^{[n-1]} = \begin{bmatrix} y_{n-1} \\ \mathsf{y}^{[n-1]} \end{bmatrix}, \qquad y^{[n]} = \begin{bmatrix} y_n \\ \mathsf{y}^{[n]} \end{bmatrix}.$$

It is also convenient to represent the exact value function as

$$z(x,h) = \begin{bmatrix} y(x) \\ \mathsf{z}(x,h) \end{bmatrix},$$

where $\mathsf{z}(x,h)$ is given by

$$\mathsf{z}(x,h) = \begin{bmatrix} hy'(x) & h^2y''(x) & \cdots & h^py^{(p)}(x) \end{bmatrix}^T.$$

Partitioning the method in this way makes it considerably more convenient to understand error propagation as the solution component acts quite differently from the remaining components of the Nordsieck vector. As very little confusion can arise, the term Nordsieck vector will refer to either $z(x,h)$ or $\mathsf{z}(x,h)$. From the context it will be clear which one is being used. To satisfy the stage order and order conditions of this alternative representation first define the scaled Vandermonde matrices C and D as

$$\mathsf{C} = [\; e \quad c \quad \cdots \quad \tfrac{c^{p-2}}{(p-2)!} \quad \tfrac{c^{p-1}}{(p-1)!} \;], \qquad \mathsf{D} = [\; c \quad \tfrac{c^2}{2!} \quad \cdots \quad \tfrac{c^{p-1}}{(p-1)!} \quad \tfrac{c^p}{p!} \;],$$

and the $p \times p$ Toeplitz matrix E by

$$\mathsf{E} = \begin{bmatrix} 1 & \frac{1}{1!} & \frac{1}{2!} & \cdots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} & \frac{1}{(p-1)!} \\ 0 & 1 & \frac{1}{1!} & \cdots & \frac{1}{(p-4)!} & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} \\ 0 & 0 & 1 & \cdots & \frac{1}{(p-5)!} & \frac{1}{(p-4)!} & \frac{1}{(p-3)!} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \frac{1}{1!} & \frac{1}{2!} \\ 0 & 0 & 0 & \cdots & 0 & 1 & \frac{1}{1!} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix},$$

which is just the last p rows and last p columns of E defined in (3.2.6). Now it follows from (3.2.9) and (3.2.10) that

$$\mathsf{U} = \mathsf{D} - A\mathsf{C}, \tag{4.3.3}$$

$$\mathsf{V} = \mathsf{E} - \mathsf{BC}, \tag{4.3.4}$$

$$\mathsf{v}^T = E_{1\_p}^T - \mathsf{b}^T\mathsf{C}, \tag{4.3.5}$$

where the vector $E_{1\_p}^T$ is defined as

$$E_{1\_p}^T = [\begin{array}{ccccc} \frac{1}{1!} & \frac{1}{2!} & \cdots & \frac{1}{(p-1)!} & \frac{1}{p!} \end{array}].$$

It is also convenient to define the reverse vector $E_{p\_1}^T$ as

$$E_{p\_1}^T = [\begin{array}{ccccc} \frac{1}{p!} & \frac{1}{(p-1)!} & \cdots & \frac{1}{2!} & \frac{1}{1!} \end{array}].$$

To investigate the error propagation of the method (4.3.1) assume that the input quantities y_{n-1} and $y^{[n-1]}$ to step number n satisfy

$$\begin{array}{rcl} y_{n-1} & = & y(x_{n-1}) - \alpha_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) + O(h^{p+2}), \\ \mathsf{y}^{[n-1]} & = & \mathsf{z}(x_{n-1},h) - \beta_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) + O(h^{p+2}), \end{array} \tag{4.3.6}$$

where α_{n-1} is a scalar and β_{n-1} is a vector of length p. Once the step is completed, the output approximations y_n and $\mathsf{y}^{[n]}$, which are also the input quantities to step number $n+1$, satisfy

$$\begin{array}{rcl} y_n & = & y(x_n) - \alpha_n h^{p+1}y^{(p+1)}(x_n) + O(h^{p+2}), \\ \mathsf{y}^{[n]} & = & \mathsf{z}(x_n,h) - \beta_n h^{p+1}y^{(p+1)}(x_n) + O(h^{p+2}). \end{array} \tag{4.3.7}$$

Expressions for α_n and β_n need to be found in terms of α_{n-1} and β_{n-1}, so as to obtain an expression for how errors propagate. In order to do so, it is first necessary to consider the errors propagated in the stage values. Since the methods have stage order equal to order, it follows that

$$Y^{[n]} = y(ex_{n-1} + ch) - \xi_n h^{p+1}y^{(p+1)}(x_{n-1}) + O(h^{p+2}). \tag{4.3.8}$$

The stage derivatives then satisfy

$$f(Y^{[n]}) = y'(ex_{n-1} + ch) - \xi_n h^{p+1}\frac{\partial f}{\partial y}y^{(p+1)}(x_{n-1}) + O(h^{p+2}),$$

and the scaled stage derivatives are then

$$hf(Y^{[n]}) = hy'(ex_{n-1} + ch) + O(h^{p+2}). \tag{4.3.9}$$

The following theorem shows how the errors are propagated in the stages, the approximate solution and the Nordsieck vector.

Theorem 4.1 *The vector ξ_n of stage errors, the constant α_n, the error in the solution and the the vector β_n of Nordsieck errors, are given by*

$$\xi_n = \frac{c^{p+1}}{(p+1)!} - \mathsf{A}\frac{c^p}{p!} + \mathsf{U}\beta_{n-1} + e\alpha_{n-1},$$

$$\alpha_n = \frac{1}{(p+1)!} - \mathsf{b}^T\frac{c^p}{p!} + \mathsf{v}^T\beta_{n-1} + \alpha_{n-1},$$

$$\beta_n = E_{p\_1} - \mathsf{B}\frac{c^p}{p!} + \mathsf{V}\beta_{n-1}. \tag{4.3.10}$$

Proof: Substituting (4.3.6), (4.3.7), (4.3.8), (4.3.9) into (4.3.2), it follows that

$$\begin{aligned}
y(ex_{n-1}+ch) &- \xi_n h^{p+1} y^{(p+1)}(x_{n-1}) \\
&= hAy'(ex_{n-1}+ch) + e\big(y(x_{n-1}) - \alpha_{n-1}h^{p+1}y^{(p+1)}(x_{n-1})\big) \\
&+ \mathsf{U}\big(\mathsf{z}(x_{n-1},h) - \beta_{n-1}h^{p+1}y^{(p+1)}(x_{n-1})\big) + O(h^{p+2}),
\end{aligned}$$

$$\begin{aligned}
y(x_n) &- \alpha_n h^{p+1} y^{(p+1)}(x_n) \\
&= h\mathsf{b}^T y'(ex_{n-1}+ch) + y(x_{n-1}) - \alpha_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) \\
&+ \mathsf{v}^T\big(\mathsf{z}(x_{n-1},h) - \beta_{n-1}h^{p+1}y^{(p+1)}(x_{n-1})\big) + O(h^{p+2}),
\end{aligned}$$

$$\begin{aligned}
\mathsf{z}(x_n,h) &- \beta_n h^{p+1} y^{(p+1)}(x_n) \\
&= h\mathsf{B}y'(ex_{n-1}+ch) \\
&+ \mathsf{V}\big(\mathsf{z}(x_{n-1},h) - \beta_{n-1}h^{p+1}y^{(p+1)}(x_{n-1})\big) + O(h^{p+2}).
\end{aligned}$$

Expanding $y(ex_{n-1}+ch)$, $y'(ex_{n-1}+ch)$ and $y(x_n)$ into Taylor series around x_{n-1}, the first expression is given by

$$\begin{aligned}
\mathsf{D}\mathsf{z}(x_{n-1},h) &+ \frac{c^{p+1}}{(p+1)!}h^{p+1}y^{(p+1)}(x_{n-1}) - \xi_n h^{p+1}y^{(p+1)}(x_{n-1}) \\
&= A\mathsf{C}\mathsf{z}(x_{n-1},h) + A\frac{c^p}{p!}h^{p+1}y^{(p+1)}(x_{n-1}) - e\alpha_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) \\
&+ \mathsf{U}\mathsf{z}(x_{n-1},h) - \mathsf{U}\beta_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) + O(h^{p+2}).
\end{aligned}$$

The terms up to $O(h^p)$ are the stage order conditions given by (4.3.3). The term of $O(h^{p+1})$ is

$$\frac{c^{p+1}}{(p+1)!} - \xi_n = A\frac{c^p}{p!} - e\alpha_{n-1} - \mathsf{U}\beta_{n-1}.$$

Again using Taylor series expansions about x_{n-1}, the second expression is given by

$$\begin{aligned}
E_{1\_p}^T\mathsf{z}(x_{n-1},h) &+ \frac{1}{(p+1)!}h^{p+1}y^{(p+1)}(x_{n-1}) - \alpha_n h^{p+1}y^{(p+1)}(x_{n-1}) \\
&= \mathsf{b}^T\mathsf{C}\mathsf{z}(x_{n-1},h) + \mathsf{b}^T\frac{c^p}{p!}h^{p+1}y^{(p+1)}(x_{n-1}) - \alpha_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) \\
&+ \mathsf{v}^T\mathsf{z}(x_{n-1},h) - \mathsf{v}^T\beta_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) + O(h^{p+2}).
\end{aligned}$$

The terms up to $O(h^p)$ are the order conditions for the solution component, given by (4.3.5). The term of $O(h^{p+1})$ is

$$\frac{1}{(p+1)!} - \alpha_n = \mathsf{b}^T\frac{c^p}{p!} - \alpha_{n-1} - \mathsf{v}^T\beta_{n-1}.$$

Using Taylor series expansions about x_{n-1}, it follows that

$$\mathsf{z}(x_n,h) = E\mathsf{z}(x_{n-1},h) + E_{p\_1}h^{p+1}y^{(p+1)}(x_{n-1}) + O(h^{p+2}).$$

The last expression now becomes

$$\begin{aligned}
\mathsf{E}\mathsf{z}(x_{n-1},h) &+ E_{p\_1}h^{p+1}y^{(p+1)}(x_{n-1}) - \beta_n h^{p+1}y^{(p+1)}(x_{n-1}) \\
&= \mathsf{B}\mathsf{C}\mathsf{z}(x_{n-1},h) + \mathsf{B}\frac{c^p}{p!}h^{p+1}y^{(p+1)}(x_{n-1}) \\
&+ \mathsf{V}\mathsf{z}(x_{n-1},h) - \mathsf{V}\beta_{n-1}h^{p+1}y^{(p+1)}(x_{n-1}) + O(h^{p+2}).
\end{aligned}$$

The terms up to $O(h^p)$ are the order conditions for the Nordsieck vector, given by (4.3.4). The term of $O(h^{p+1})$ is

$$E_{p\_1} - \beta_n = \mathsf{B}\frac{c^p}{p!} - \mathsf{V}\beta_{n-1}.$$

Rearranging the terms of $O(h^{p+1})$ gives the required result. $\square$

Note that the expression for β_n only depends on β_{n-1} and constant vectors and matrices. When the stepsize is constant the value of β_n tends to a fixed point β, that is

$$\mathsf{y}^{[n]} = \mathsf{z}(x_n, h) - \beta h^{p+1} y^{(p+1)}(x_n) + O(h^{p+2}).$$

The value of the fixed point β is found letting $\beta_n = \beta_{n-1}$ in (4.3.10), and rearranging gives

$$\beta = (I - \mathsf{V})^{-1}\left(E_{p\_1} - \mathsf{B}\frac{c^p}{p!}\right).$$

The vector β has a surprising form for the IRKS methods, as is shown in the following theorem.

Theorem 4.2 *The vector β which satisfies*

$$\mathsf{y}^{[n]} = \mathsf{z}(x_n, h) - \beta h^{p+1} y^{(p+1)}(x_n) + O(h^{p+2}).$$

is given by

$$\beta = [\ \beta_p \quad \beta_{p-1} \quad \cdots \quad \beta_2 \quad \beta_1\ ]^T,$$

where the β_i are the free parameters from the doubly companion matrix X.

Proof: It is most convenient to use the original form of the method in the first part of this proof. Substituting (3.2.9) into U and (3.2.10) into the first V of (3.3.2), then

$$\begin{aligned} B(C - ACK) &\equiv X(E - BCK) - VX, \\ BC &\equiv XE - VX. \end{aligned} \tag{4.3.11}$$

It was shown in Section 3.5 that if $BA = XB$ is satisfied then $BU \equiv XV - VX$ is zero except in the last column. To consider the last column of (4.3.11), first notice that

$$XEe_{p+1} \equiv E_{p+1\_1} - \bar{\beta},$$

where the vector $\bar{\beta}$ is

$$\bar{\beta} = \begin{bmatrix} \beta_{p+1} & \beta_p & \cdots & \beta_2 & \beta_1 \end{bmatrix}^T.$$

The last column of (4.3.11) is now

$$B\frac{c^p}{p!} \equiv E_{p+1\_1} - \bar{\beta} + V\bar{\beta}.$$

To remove the first row and therefore obtain equality, it follows that

$$\mathsf{B}\frac{c^p}{p!} = E_{p\_1} - \beta + \mathsf{V}\beta.$$

Now rearranging in terms of β gives the required result. $\square$

Since the coefficients $\beta_1, \beta_2, \dots, \beta_p$ are free parameters of the method, the vector β can be chosen as part of the design of the method. If α_{n-1} is assumed to be zero then (4.3.6) is similar to the localising assumptions of Runge-Kutta and linear multistep methods. When $\alpha_{n-1} = 0$, then α_n is equivalent to the error constant, σ, given by

$$\sigma = \frac{1}{(p+1)!} - \mathsf{b}^T \frac{c^p}{p!} + \mathsf{v}^T \beta, \tag{4.3.12}$$

which is also a free parameter of the IRKS methods. Comparing the above equation with (3.6.3), then

$$\epsilon = M_{p+1,p+1}(\lambda) - \frac{1}{(p+1)!} + \mathsf{b}^T \frac{c^p}{p!} - \mathsf{v}^T \beta.$$

The free parameters in the IRKS methods have a large scope for controlling the errors which are propagated by both the solution and the Nordsieck vector.

4.4 Error estimation

For any general linear method to be implemented in an adaptive fashion it must be possible to estimate the local truncation error, as this allows a measure of how accurate the approximations are, and how much the stepsize should be varied. Assuming that the localising assumptions hold, that is $\alpha_{n-1} = 0$, the local truncation error is given by

$$t(x_n) = \sigma h^{p+1} y^{(p+1)}(x_n) + O(h^{p+2}),$$

where σ is the error constant given by (3.6.3) or (4.3.12). In most cases only the principal term of the local truncation error is calculated. This assumes that h is sufficiently small and that $O(h^{p+2})$ is significantly smaller than $h^{p+1}y^{(p+1)}(x_n)$. The principal term of the local truncation error is calculated using the following theorem.

Theorem 4.3 *If the function $y(x)$ is sufficiently differentiable in a neighbourhood of x, then*

$$h^{p+1} y^{(p+1)}(x_n) = \varphi^T h f\big(Y^{[n]}\big) + \psi^T \mathsf{y}^{[n-1]} + O(h^{p+2}), \tag{4.4.1}$$

where the vectors φ^T and ψ^T satisfy the linear system

$$\begin{aligned} \varphi^T \mathsf{C} + \psi^T &= 0, \\ \varphi^T \frac{c^p}{p!} - \psi^T \beta &= 1. \end{aligned} \tag{4.4.2}$$

Proof: Substituting (4.3.6) and (4.3.9) into (4.4.1), it follows that

$$h^{p+1} y^{(p+1)}(x_n) = \varphi^T h y'(e x_{n-1} + ch) + \psi^T \big(\mathsf{z}(x_{n-1}, h) - \beta h^{p+1} y^{(p+1)}(x_{n-1})\big) + O(h^{p+2}).$$

Now expanding $y'(ex_{n-1} + ch)$ and $h^{p+1}y^{(p+1)}(x_n)$ into Taylor series around x_{n-1}, then

$$\begin{aligned} h^{p+1} y^{(p+1)}(x_{n-1}) = \varphi^T &\left(\mathsf{C}\mathsf{z}(x_{n-1}, h) + \frac{c^p}{p!} h^{p+1} y^{(p+1)}(x_{n-1}) \right) \\ &+ \psi^T \big(\mathsf{z}(x_{n-1}, h) - \beta h^{p+1} y^{(p+1)}(x_{n-1})\big) + O(h^{p+2}). \end{aligned}$$

Collecting the terms up to order $p+1$ gives the required result. □

Since the terms of $O(h^{p+2})$ are deemed to be sufficiently small so as not to affect the local truncation error significantly, it is convenient to define the local error estimate $e(x_n)$ as

$$e(x_n) = \varphi^T h f\big(Y^{[n]}\big) + \psi^T \mathsf{y}^{[n-1]}.$$

Therefore, the local truncation error and the local error estimate are related by

$$t(x_n) = \sigma e(x_n) + O(h^{p+2}).$$

It is also possible to compute the local error estimate $e(x_n)$ with a linear combination of the stage derivatives of the previous and current step. This approach was used for DIMSIMs (see [37]). However, this approach is not desirable because it results in conditions involving the stepsize ratio between the two steps. The coefficients in the local error estimate must be computed each time the stepsize is changed resulting in unnecessary amount of extra computation.

To estimate the local truncation error, Runge-Kutta methods are usually constructed in pairs, a technique known as embedding. The local error estimate is then taken to be the difference of these two approximations. Generally the methods differ in order by one. Many researchers regard it as more efficient to propagate the higher order method. In this case the local error estimate is not asymptotically correct. However, many implementations of Runge-Kutta methods do in fact propagate the lower order method and therefore obtain asymptotically correct local error estimates. Both approaches result in the estimate of local error as a linear combination of the stage derivatives of the current step. It is possible to obtain an equivalent approach to embedding for the IRKS methods if the vector ψ^T is zero, that is, obtain local error estimates using only the stage derivatives from the current step. This can only be the case if the number of stages s is greater than the order p. The IRKS methods are chosen so that $s = p + 1$, which satisfies the previous requirement. In this case the linear system (4.4.2) reduces to

$$\varphi^T C = e_{p+1}, \tag{4.4.3}$$

as long as the method is non-confluent. This means that asymptotically correct local error estimates can be computed using only a linear combination of the stage derivatives. Note that for Runge-Kutta methods it is almost always the case that extra stages need to be computed in order to obtain the two different approximations. In all the implementations of the IRKS methods the local truncation error estimates will all use only a linear combination of the stage derivatives.

4.5 Nordsieck representation

From early on only methods represented in Nordsieck form have been discussed. To obtain an effective implementation of any numerical method, stepsize change is a necessary requirement. The multiple output approximations in these methods make stepsize change more difficult than for Runge-Kutta methods. The use of Nordsieck vectors makes it easy to vary stepsize, which is in stark contrast to most other methods not implemented in Nordsieck form. Not all methods can be represented in Nordsieck form, for example, the explicit Runge-Kutta methods with effective order. However any method with $p + 1$ independent approximations passed from step to step can be. The Nordsieck vector consists of an approximation to the solution and the first p scaled derivative approximations. Consider a non-uniform grid $x_0 < x_1 < x_2 < \cdots < x_{N-2} < x_{N-1} < x_N$ with stepsizes $h_1, h_2, h_3, \ldots, h_{N-2}, h_{N-1}, h_N$ given in the following figure.

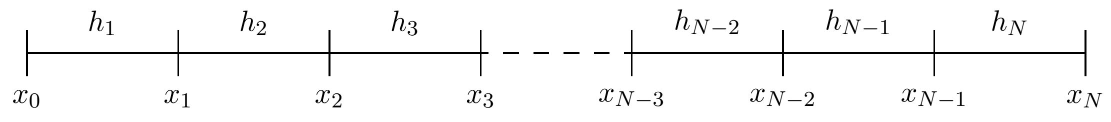


Figure 4.1: Stepsize pattern.

The computations performed in one step of a method on a non-uniform grid are

$$\begin{aligned} Y^{[n]} &= h_n A f\big(Y^{[n]}\big) + U y^{[n-1]}, \\ \widetilde{y}^{[n]} &= h_n B f\big(Y^{[n]}\big) + V y^{[n-1]}, \end{aligned}$$

where the incoming and outgoing Nordsieck vectors are

$$y^{[n-1]} = \begin{bmatrix} y_{n-1} \\ h_n y'_{n-1} \\ \vdots \\ h_n^{p-1} y_{n-1}^{(p-1)} \\ h_n^p y_{n-1}^{(p)} \end{bmatrix} \qquad \widetilde{y}^{[n]} = \begin{bmatrix} y_n \\ h_n y'_n \\ \vdots \\ h_n^{p-1} y_n^{(p-1)} \\ h_n^p y_n^{(p)} \end{bmatrix}.$$

Define the diagonal $(p+1) \times (p+1)$ rescaling matrix $D(\delta_{n+1})$ as

$$D(\delta_{n+1}) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \delta_{n+1} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \delta_{n+1}^2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \delta_{n+1}^{p-2} & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \delta_{n+1}^{p-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \delta_{n+1}^{p} \end{bmatrix},$$

where $\delta_{n+1} = h_{n+1}/h_n$. It then follows that

$$y^{[n]} = D(\delta_{n+1})\widetilde{y}^{[n]}.$$

This means the variable stepsize method can be rewritten as

$$\begin{aligned} Y^{[n]} &= h_n A f\big(Y^{[n]}\big) + U y^{[n-1]}, \\ y^{[n]} &= h_n D(\delta_{n+1}) B f\big(Y^{[n]}\big) + D(\delta_{n+1}) V y^{[n-1]}. \end{aligned}$$

Even though the structure looks somewhat different from the approach derived by Butcher, Chartier and Jackiewicz in [37] it is equivalent. Note that this rescaling incurs an extra error avoided by Runge-Kutta methods. The effect of this rescaling can become significant, especially for high order methods and large changes in the stepsize.

The Nordsieck formulation originally used by Nordsieck [102] and popularised by Gear [69], requires the incoming and outgoing Nordsieck vector to be given by

$$y^{[n-1]} = \begin{bmatrix} y_{n-1} \\ h_n y'_{n-1} \\ \vdots \\ \frac{h_n^{p-1}}{(p-1)!} y_{n-1}^{(p-1)} \\ \frac{h_n^p}{p!} y_{n-1}^{(p)} \end{bmatrix}, \qquad y^{[n]} = \begin{bmatrix} y_n \\ h_{n+1} y'_n \\ \vdots \\ \frac{h_{n+1}^{p-1}}{(p-1)!} y_n^{(p-1)} \\ \frac{h_{n+1}^p}{p!} y_n^{(p)} \end{bmatrix}.$$

To represent the method in this form, a simple change of basis is required. Define the scaling matrix F as

$$F = \operatorname{diag}\Big(\begin{matrix} 0! & 1! & 2! & \cdots & p! \end{matrix} \Big).$$

The matrices in the original Nordsieck form are therefore

$$\bar{U} = UF, \qquad \bar{B} = F^{-1}B, \qquad \bar{V} = F^{-1}VF.$$

It will always be assumed that the method is in Nordsieck form but not the original Nordsieck form. If the original Nordsieck form is preferred, it is easy to rescale the matrices as above. It should be noted that in the original Nordsieck form the coefficients of the IRKS methods are much more balanced. The coefficients in the the matrix B become quite large especially in the last rows in the Nordsieck form, whereas the coefficients in the last columns of U become very small. When searching for methods with minimised error coefficients in the underlying one step method, the magnitude of the largest coefficient of the method is controlled. Thus the original formulation is used. It is also the case that to avoid as much loss of significance as possible it is best to use the original Nordsieck form in the implementations, since the coefficients of the method are more balanced.

Now consider the error propagation of the methods in Nordsieck form. To do so partition the methods as follows

$$\begin{aligned} Y^{[n]} &= h_n A f\big(Y^{[n]}\big) + e y_{n-1} + \mathsf{U}\mathsf{y}^{[n-1]}, \\ y_n &= h_n \mathsf{b}^T f\big(Y^{[n]}\big) + y_{n-1} + \mathsf{v}^T\mathsf{y}^{[n-1]}, \\ \mathsf{y}^{[n]} &= h_n \mathsf{D}(\delta_{n+1})\mathsf{B} f\big(Y^{[n]}\big) + \mathsf{D}(\delta_{n+1})\mathsf{V}\mathsf{y}^{[n-1]}, \end{aligned} \tag{4.5.1}$$

where the diagonal $p \times p$ rescaling matrix $\mathsf{D}(\delta_{n+1})$ is

$$\mathsf{D}(\delta_{n+1}) = \begin{bmatrix} \delta_{n+1} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \delta_{n+1}^2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \delta_{n+1}^3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \delta_{n+1}^{p-2} & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \delta_{n+1}^{p-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \delta_{n+1}^{p} \end{bmatrix}.$$

This approach to changing stepsize is known as the rescale method. The only quantity that is different from the constant stepsize case is the Nordsieck vector. Before determining the effect of the variable stepsize, first note that the incoming Nordsieck vector has the form

$$\mathsf{y}^{[n-1]} = \mathsf{z}(x_{n-1}, h_n) - \beta_{n-1} h_n^{p+1} y^{(p+1)}(x_{n-1}) + O(h_n^{p+2}), \tag{4.5.2}$$

and the corresponding outgoing Nordsieck vector has the form

$$\mathsf{y}^{[n]} = \mathsf{z}(x_n, h_{n+1}) - \beta_n h_{n+1}^{p+1} y^{(p+1)}(x_n) + O(h_{n+1}^{p+2}). \tag{4.5.3}$$

The following lemma shows that β_n does not converge to a fixed point unless constant stepsizes are used.

Lemma 4.4 *The vector β_n of Nordsieck errors using the rescale approach to variable stepsize is*

$$\beta_n = \frac{\mathsf{D}(\delta_{n+1})}{\delta_{n+1}^{p+1}} \left(E_{p\_1} - \mathsf{B}\frac{c^p}{p!} + \mathsf{V}\beta_{n-1} \right).$$

Proof: Substituting (4.5.2), (4.5.3), (4.3.9) into the last equation of (4.5.1), then

$$\begin{aligned}\mathsf{z}(x_n, h_{n+1}) - \beta_n h_{n+1}^{p+1} y^{(p+1)}(x_n)\\ &= \mathsf{D}(\delta_{n+1})\mathsf{B} h_n y'(ex_{n-1} + ch_n)\\ &+ \mathsf{D}(\delta_{n+1})\mathsf{V}\big(\mathsf{z}(x_{n-1}, h_n) - \beta_{n-1} h_n^{p+1} y^{(p+1)}(x_{n-1})\big) + O(h^{p+2}),\end{aligned}$$

where $h = \max(h_n, h_{n+1})$. Using Taylor series expansions about x_{n-1} it can be shown that

$$\mathsf{z}(x_n, h_{n+1}) = \mathsf{D}(\delta_{n+1})\big(E\mathsf{z}(x_{n-1}, h_n) + E_{p\_1} h_n^{p+1} y^{(p+1)}(x_{n-1})\big) + O(h_n^{p+2}).$$

Now expanding $y'(ex_{n-1} + ch_n)$ into Taylor series around x_{n-1}

$$\begin{aligned}\mathsf{D}(\delta_{n+1})\big(\mathsf{E}\mathsf{z}(x_{n-1}, h_n) + E_{p\_1} h_n^{p+1} y^{(p+1)}(x_{n-1})\big) - \beta_n \delta_{n+1}^{p+1} h_n^{p+1} y^{(p+1)}(x_{n-1})\\ &= \mathsf{D}(\delta_{n+1})\mathsf{B}\left(\mathsf{C}\mathsf{z}(x_{n-1}, h_n) + \frac{c^p}{p!} h_n^{p+1} y^{(p+1)}(x_{n-1})\right)\\ &+ \mathsf{D}(\delta_{n+1})\mathsf{V}\big(\mathsf{z}(x_{n-1}, h_n) - \beta_{n-1} h_n^{p+1} y^{(p+1)}(x_{n-1})\big) + O(h_n^{p+2}).\end{aligned}$$

The terms up to $O(h^p)$ simplify to the order conditions for the Nordsieck vector, given by (4.3.4). The terms of order $p+1$ are

$$\mathsf{D}(\delta_{n+1}) E_{p\_1} - \beta_n \delta_{n+1}^{p+1} = \mathsf{D}(\delta_{n+1})\left(\mathsf{B}\frac{c^p}{p!} - \mathsf{V}\beta_{n-1}\right),$$

which gives the the required result when rearranged. Note this only converges if the stepsize is constant. □

This is a rather negative result because the values of β_{n-1} are part of the error propagated in the stages ξ_n and the approximate solution α_n. In order to retain the form of the error propagation given in Theorem 4.1, but with variable stepsizes, it is possible to modify the method (4.5.1) by using the information from the local error estimate $e(x_n)$. This technique, known as "rescale and modify", results in the method

$$\begin{aligned}Y^{[n]} &= h_n A f\big(Y^{[n]}\big) + e y_{n-1} + \mathsf{U}\mathsf{y}^{[n-1]},\\ y_n &= h_n \mathsf{b}^T f\big(Y^{[n]}\big) + e y_{n-1} + \mathsf{v}^T \mathsf{y}^{[n-1]},\\ \mathsf{y}^{[n]} &= h_n\big(\mathsf{D}(\delta_{n+1})\mathsf{B} + \theta(\delta_{n+1})\varphi^T\big) f\big(Y^{[n]}\big) + \big(\mathsf{D}(\delta_{n+1})\mathsf{V} + \theta(\delta_{n+1})\psi^T\big)\mathsf{y}^{[n-1]},\end{aligned} \tag{4.5.4}$$

where $\theta(\delta_{n+1})$ is a vector with a particular form, to be determined.

Theorem 4.5 *The vector β_n of Nordsieck errors using the rescale and modify approach to variable stepsize converges to the fixed point*

$$\beta = (I - \mathsf{V})^{-1}\left(E_{p\_1} - \mathsf{B}\frac{c^p}{p!}\right), \tag{4.5.5}$$

provided that φ^T and ψ^T satisfy the linear conditions (4.4.2) and $\theta(\delta_{n+1})$ has the following form

$$\theta(\delta_{n+1}) = \big(\mathsf{D}(\delta_{n+1}) - \delta_{n+1}^{p+1} I\big)\beta.$$

Proof: Substituting (4.5.2), (4.5.3), (4.3.9) into the last equation of (4.5.4), it follows that

$$\begin{aligned}\mathsf{z}(x_n, h_{n+1}) - \beta h_{n+1}^{p+1} y^{(p+1)}(x_n)\\ &= \big(\mathsf{D}(\delta_{n+1})\mathsf{B} + \theta(\delta_{n+1})\varphi^T\big) h_n y'(ex_{n-1} + ch_n)\\ &+ \big(\mathsf{D}(\delta_{n+1})\mathsf{V} + \theta(\delta_{n+1})\psi^T\big)\big(\mathsf{z}(x_{n-1}, h_n) - \beta h_n^{p+1} y^{(p+1)}(x_{n-1})\big) + O(h^{p+2}),\end{aligned}$$

where $h = \max(h_n, h_{n+1})$. Now expanding $\mathsf{z}(x_n, h_{n+1})$ and $y'(ex_{n-1} + ch_n)$ into Taylor series around x_{n-1}, then

$$\begin{aligned}\mathsf{D}(\delta_{n+1})\big(\mathsf{E}\mathsf{z}(x_{n-1}, h_n) + E_{p\_1} h_n^{p+1} y^{(p+1)}(x_{n-1})\big) - \beta\delta_{n+1}^{p+1} h_n^{p+1} y^{(p+1)}(x_{n-1})&\\ = \big(\mathsf{D}(\delta_{n+1})\mathsf{B} + \theta(\delta_{n+1})\varphi^T\big)\left(\mathsf{C}\mathsf{z}(x_{n-1}, h_n) + \frac{c^p}{p!} h_n^{p+1} y^{(p+1)}(x_{n-1})\right)&\\ + \big(\mathsf{D}(\delta_{n+1})\mathsf{V} + \theta(\delta_{n+1})\psi^T\big)\big(\mathsf{z}(x_{n-1}, h_n) - \beta h_n^{p+1} y^{(p+1)}(x_{n-1})\big) + O(h_n^{p+2}).&\end{aligned}$$

Collecting the terms up to $O(h^p)$ as coefficients of $\mathsf{D}(\delta_{n+1})$ and $\theta(\delta_{n+1})$ it follows that the first coefficient is equivalent to the order conditions for the Nordsieck vector and the second coefficient is equivalent to the first equation of (4.4.2) used to obtain an error estimate. Collecting the terms of order $p + 1$, it follows that

$$\mathsf{D}(\delta_{n+1})E_{p\_1} - \beta\delta_{n+1}^{p+1} = \mathsf{D}(\delta_{n+1})\left(\mathsf{B}\frac{c^p}{p!} - \mathsf{V}\beta\right) + \theta(\delta_{n+1})\left(\varphi^T\frac{c^p}{p!} - \psi^T\beta\right).$$

Rearranging (4.5.5) leads to

$$\mathsf{B}\frac{c^p}{p!} - \mathsf{V}\beta = E_{p\_1} - \beta,$$

and noting that the term inside the second bracket is the last condition of (4.4.2), then

$$\mathsf{D}(\delta_{n+1})E_{p\_1} - \beta\delta_{n+1}^{p+1} = \mathsf{D}(\delta_{n+1})(E_{p\_1} - \beta) + \theta(\delta_{n+1}).$$

Now rearranging in terms of $\theta(\delta_{n+1})$, gives

$$\theta(\delta_{n+1}) = \big(\mathsf{D}(\delta_{n+1}) - \delta_{n+1}^{p+1} I\big)\beta.$$

Therefore, β is fixed if $\theta(\delta_{n+1})$ is defined as above. □

As long as the rescale and modify approach is used the error propagation of the variable stepsize approach is equivalent to that of the fixed stepsize approach. Therefore the error propagation terms α_n, β_n and ξ_n are defined as in Theorem 4.1. The elements of the vector β_n are free parameters of the IRKS methods along with the error constant which is equal to α_n if α_{n-1} equals zero. This means that the IRKS methods have a lot of control over how the error propagates.

As a means of comparing different methods, it was suggested in [37] to construct a problem where the exact solution is known and subject the method to rapid step changes. To compare the rescale method (4.5.1) and the rescale and modify method (4.5.4), use the test equation

$$y'(x) = \lambda\big(y(x) - \exp(\mu x)\big) + \mu\exp(\mu x), \qquad y(x_0) = y_0,$$

where λ and μ are real parameters and y_0 is a given initial condition, with exact solution

$$y(x) = \exp(\mu x) - \big(\exp(\mu x) - y_0\big)\exp\big(\lambda(x - x_0)\big).$$

In this experiment $\lambda = \mu = -0.1$ and the new stepsize h_{n+1} is calculated according to the formula

$$h_{n+1} = \rho^{(-1)^n \sin\left(4\pi n/(x_N - x_0)\right)} h_n,$$

where $h_0 = (x_N - x_0)/N, x_0 = 0, y_0 = 1, x_N = 20$ and $N = 800$.

In the following figures, the local error estimate $e(x_n)$ denoted by $\cdot$ is compared with the local discretisation error $t(x_n)$ denoted by $\circ$. For ease of representation only every fifth step has been plotted. In Figure 4.2 the rescale method is compared with the rescale and modify method for $\rho = 2$.

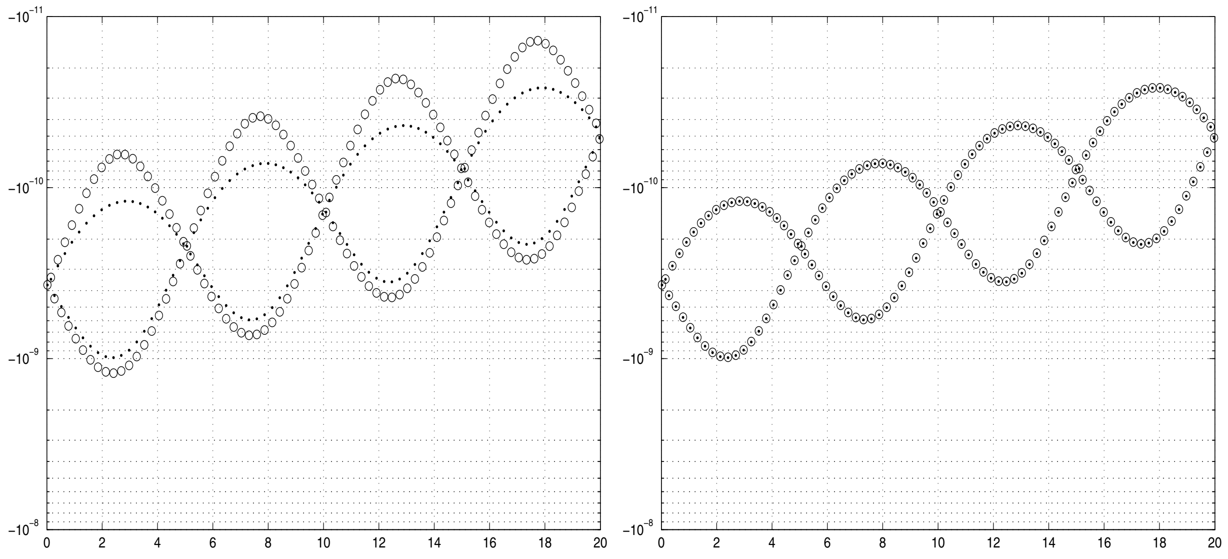


Figure 4.2: Rescale method (left) versus the rescale and modify method (right) with $\rho = 2$.

Notice that the rescale and modify method is considerably more accurate than the rescale method. In fact the rescale and modify method leads to almost exact agreement between the local truncation error and the local error estimate. In Figure 4.3 the rescale method is compared with the rescale and modify method for $\rho = 3$.

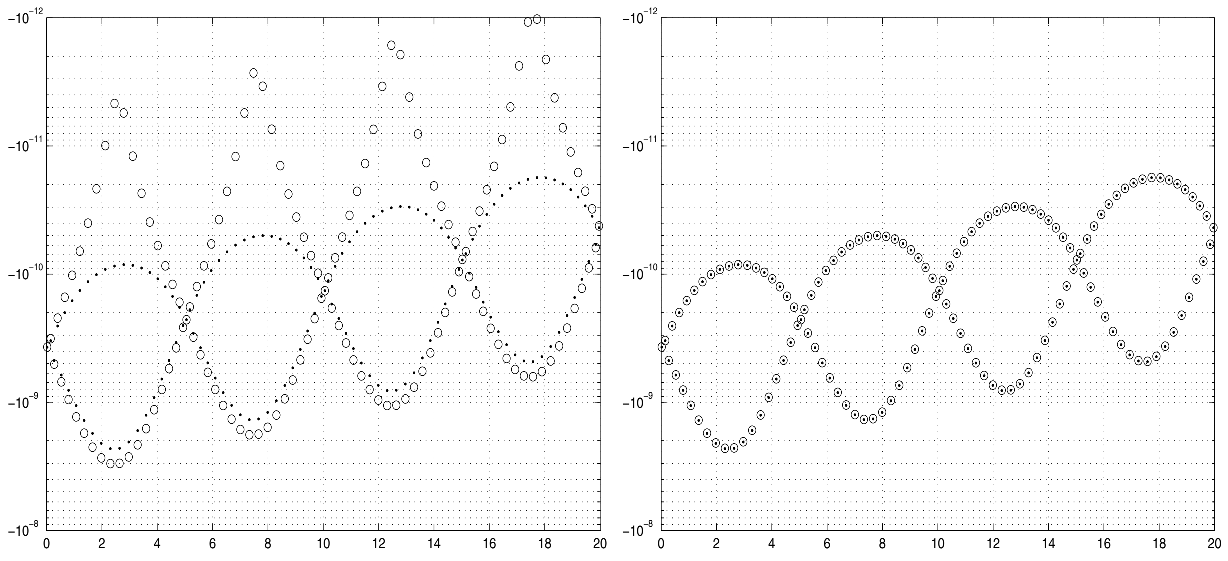


Figure 4.3: Rescale method (left) versus the rescale and modify method (right) with $\rho = 3$.

Again the rescale and modify method is considerably more accurate than the rescale method. Note that there is still almost perfect agreement between the estimate and true local truncation in error in the rescale and modify method. For the case when $\rho = 4$ the error in the rescale method becomes extremely large and is therefore not given. Figure 4.4 shows the rescale and modify method when $\rho = 8$ and $\rho = 16$.

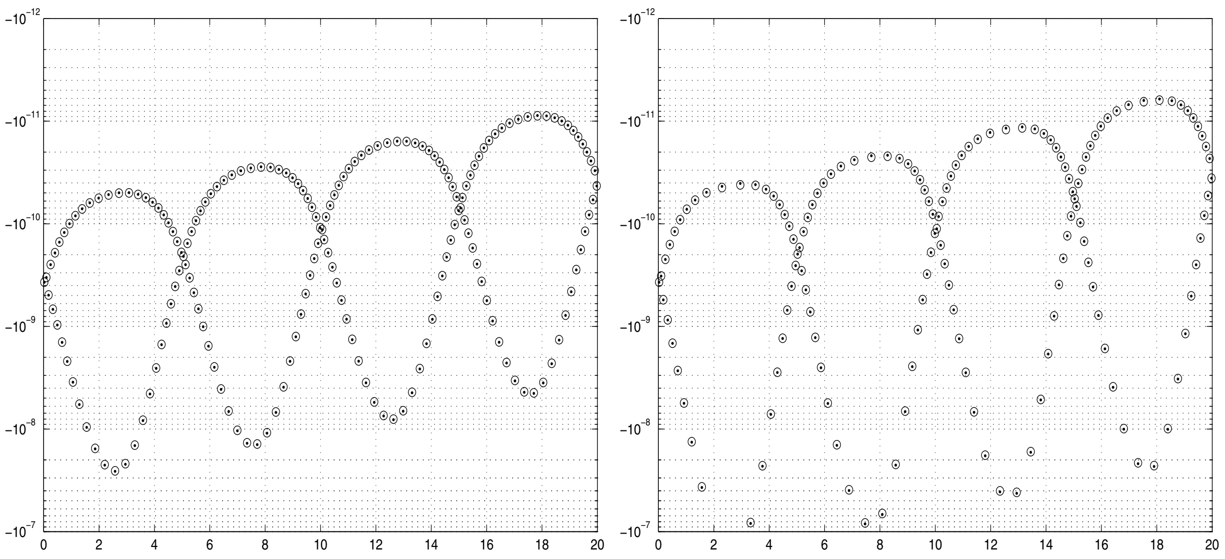


Figure 4.4: Rescale and modify methods with $\rho = 8$ (left) and $\rho = 16$ (right).

It is encouraging to see that even for the case when $\rho = 16$ there is still only a small difference between the estimate and true local discretisation error.

4.6 Stability analysis

It is significantly more difficult to understand the stability of the variable stepsize methods compared with the stability of the fixed stepsize methods. This is due to the fact that the stability matrix is different every time the stepsize is changed. The stability matrix of the rescale method is given by

$$M(z, \delta_{n+1}) = D(\delta_{n+1})\big(V + zB(I - zA)^{-1}U\big),$$

which depends on the stepsize ratio between the steps. The stability matrix of the rescale and modify method is given by

$$M(z, \delta_{n+1}) = D(\delta_{n+1})\big(V + zB(I - zA)^{-1}U\big) + \theta(\delta_{n+1})\big(\psi^T + z\varphi^T(I - zA)^{-1}U\big),$$

which is now dependent on the stepsize ratio between the steps and the local error estimate. To obtain zero-stability for one of these methods, it is not possible to only assume that the stability matrix $M(0, 1)$ is power bounded. It is necessary to take into account the fact that the stability matrix varies each time the stepsize is changed. Therefore, zero-stability is equivalent to requiring the product of matrices

$$M(0, \delta_N)M(0, \delta_{N-1}) \cdots M(0, \delta_2)M(0, \delta_1),$$

is bounded, where $M(0, \delta_i)$ for the rescale method is

$$M(0, \delta_i) = D(\delta_i)V,$$

or where $M(0, \delta_i)$ for the rescale and modify method is

$$M(0, \delta_i) = D(\delta_i)V + \theta(\delta_i)\psi^T.$$

Given that two matrices have spectral radius less than one does not imply that the product of the matrices has spectral radius less than one, so it is not possible just to check that $\rho(M(0, \delta_i)) < 1$. For this reason it is important to determine whether there exist a ratio δ_* such that $M(0, \delta)$ is bounded for δ satisfying $0 < \delta \le \delta_*$. To solve this problem Guglielmi and Zennaro [74] used the polytope norm $\| \cdot \|_*$ in $\mathbb{R}^2$ such that

$$\|M(0, \delta)\|_* < 1, \qquad \delta \in (0, \delta_*].$$

A polytope norm is a norm defined by its unit ball in $\mathbb{R}^2$. Some examples of polytope norms for general linear methods in Nordsieck form are given in [45]. Constructing these in general is a rather complicated task and is individual to each method. To overcome this difficulty it is possible to choose particular IRKS methods which are necessarily zero-stable in this general sense. If the matrix formed by removing the first row and column of V is strictly upper triangular and the local error estimate is computed using (4.4.3) then $\|M(0, \delta_i)\|_* = 0$. So it is possible to construct IRKS methods which are zero stable for any choice of variable mesh by requiring the matrix $T = I$. It is not claimed that methods of this form are necessarily better methods, but at this stage it is at least encouraging to know that such methods exist.

4.7 Stepsize control

It is generally the case that the solution of an arbitrary ordinary differential equation varies considerably during the course of an integration. Therefore, certain parts of the trajectory need more attention than others. The most important means of controlling the amount of work done in certain parts of the trajectory is to vary the stepsize. In order to obtain an estimate of an appropriate stepsize for the next step, the maximum stepsize possible in the current step must be estimated. Given an approximation to the principal local truncation error

$$\sigma h_n^{p+1} y_{n-1}^{(p+1)} = t_n,$$

and given a user specified tolerance, the maximum stepsize $\overline{h}_n$ must satisfy

$$\sigma \overline{h}_n^{p+1} y_{n-1}^{(p+1)} = a + r|y_n|,$$

where a is the absolute tolerance and r is the relative tolerance required. It then follows that

$$\sigma y_{n-1}^{(p+1)} = \frac{1}{h_n^{p+1}} t_n = \frac{1}{\overline{h}_n^{p+1}} (a + r|y_n|).$$

As it is not possible to satisfy the above equation componentwise, since the only free parameter $\overline{h}_n$ is a scalar, it must be done using some norm. Therefore, the following equation must be satisfied

$$\left(\frac{\overline{h}_n}{h_n}\right)^{p+1} = \|\omega\|_N,$$

where the vector ω has components

$$\omega_i = \frac{r|y_{n,i}| + a}{t_{n,i}}.$$

Two commonly used norms are the root mean square norm and infinity norm given respectively by

$$\|\omega\|_{RMS} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \omega_i^2}, \qquad \|\omega\|_\infty = \max_i |w_i|. \tag{4.7.1}$$

Note that if the solution is likely to pass through zero, it is important that a does not equal zero, otherwise the stepsize will be reduced to zero. The maximum stepsize $\overline{h}_n$ can be represented in terms of h_n, as follows

$$\overline{h}_n = h_n \|\omega\|^{\frac{1}{p}}.$$

If $y^{(p+1)}(x)$ is sufficiently smooth, then it can be assumed that $y_{n-1}^{(p+1)} = y_n^{(p+1)}$ and therefore $\overline{h}_n = h_{n+1}$. Generally, to make this choice more feasible a safety factor γ is included to reduce the number of unnecessary rejected steps. Thus

$$h_{n+1} = \gamma h_n \|\omega\|^{\frac{1}{p+1}}.$$

The above analysis is concerned with monitoring the error per step, to control the stepsize. An alternative approach known as error per unit step is used sometimes. This is to say the error contributed in some parts of the trajectory shouldn't differ from other parts of the trajectory, due to the varying size of the steps taken during the integration. Error per unit step can be formulated in the same way as error per step by dividing the error contributed during the current step by the size of the step. It will always be assumed that the new stepsize is calculated using error per step rather than error per unit step.

Either method of stepsize control will encounter difficulties in certain circumstances.

- *Slow variation:* If $y^{(p+1)}(x)$ is not smooth then $y_{n-1}^{(p+1)}$ is often not a good approximation to $y_n^{(p+1)}$. This can occur especially if oscillatory behaviour is present in the solution.
- *Asymptotics:* If the method is of order p, but is not displaying order p qualities, then the method is said to exhibit order reduction.

When developing the stepsize control procedure it is necessary to use appropriately chosen constants r_{low}, r_{high} and h_{max}. The constant r_{low} is the smallest ratio by which the current stepsize can be reduced, which avoids unnecessarily having to increase the stepsize when the local error estimate is much smaller than the true local error. The constant r_{high} is the largest ratio that the current stepsize can increased by. The constant h_{max} is the largest size a stepsize can be anywhere in the integration. It is also convenient for reasons discussed in the following section that if the new stepsize is very close to the current stepsize, then it is convenient to keep the new stepsize the same as the old stepsize. This is referred to as a dead-zone and is often included in stepsize control algorithms. A dead-zone is almost always included for linear multistep methods due to the high cost of changing stepsize.

This sort of stepsize control is known as deadbeat control or integral control. The name deadbeat comes from the fact that at every step the controller tries to eliminate the whole deviation of the error from the tolerance at each step. This has lead to work mainly by Söderlind and his co-workers (see for example [76]), in which they use more sophisticated means of controlling the stepsize. Generally speaking these controllers, which are based on work from control theory, carefully weight information gained about the local discretisation error and stepsizes over at least the last two steps. The aim of this technique is not to damp out the difference in the error and tolerance at each step but to smoothly approach this situation. This eliminates problems such as repeated accept reject occurrences and $(-1)^n$ oscillations which often occur in the solution. For these more sophisticated controllers a dead-zone is not usually required. Even though it is very tempting to directly use these more sophisticated controllers for general linear methods, as these were developed for one step methods, the same sort of improvements could not be expected for general linear methods. So for the mean time the traditional controller is used, with the hope that more sophisticated controllers will be developed for general linear methods in the near future.

4.8 Order control

To obtain high efficiency it is well known that not only is the stepsize varied but also the order of the method. To appreciate this statement one looks at possible problems which occur with fixed order methods.

Problems with low order methods:

- If the error tolerance is severe, low fixed order methods require very small steps, thus increasing computational cost.
- Discontinuities can often exist and low order methods may not span the whole interval required for robustness.

Problems with high order methods:

- If the error tolerance is lax, the local error estimate is less accurate than the method estimates, thus much larger steps are attempted than should be, resulting in rejected steps.
- Higher order methods take larger steps and can step past discontinuities.

It is therefore important to include an order changing procedure, as certain parts of the trajectory require varying order methods to obtain the required efficiency while minimising the costs incurred. Changing the order of Runge-Kutta methods is easy as the only output approximation is the numerical solution. However, assessing the relative advantages of changing the order is inaccurate and inefficient. For general linear methods, the collection of methods available are of orders $p-1$, p and $p+1$ as it is difficult and inaccurate to approximate the scaled derivatives $h^{p+3}y^{(p+3)}(x)$, $h^{p+4}y^{(p+4)}(x)$, ..., required to increase the order by more than one unit. However, enough information is available to obtain a sufficiently accurate approximation to $h^{p+2}y^{(p+2)}(x)$ which is the critical calculation involved in order change.

The approximation to the solution at the point x_n using a method of order p, with stepsize h_n, is y_n. An approximation to the principal local truncation error, $t_{n,p}$, is used to estimate the stepsize h_{n+1}, using the method of order p. The principal local truncation error of methods for order $p-1$ and $p+1$ are

$$\begin{aligned} t_{n,p-1} &= \sigma_{p-1} h_n^p y_{n-1}^{(p)}, \\ t_{n,p+1} &= \sigma_{p+1} h_n^{p+2} y_{n-1}^{(p+2)}, \end{aligned}$$

where σ_{p-1} and σ_{p+1}, are the error constants of the methods of order $p-1$ and $p+1$ respectively. These are required to estimate the stepsize h_{n+1} if either of these methods were used instead. Using the traditional controller the predicted stepsizes $h_{n+1,p-1}$, $h_{n+1,p}$ and $h_{n+1,p+1}$ for the order $p-1$, p and $p+1$ methods respectively, are given by

$$\begin{aligned} h_{n+1,p-1} &= h_n \|\omega_{p-1}\|^{\frac{1}{p}}, \\ h_{n+1,p} &= h_n \|\omega_p\|^{\frac{1}{p+1}}, \\ h_{n+1,p+1} &= h_n \|\omega_{p+1}\|^{\frac{1}{p+2}}, \end{aligned}$$

where ω_{p-1}, ω_p and ω_{p+1} are

$$\omega_{p-1} = \frac{r|y_{n,i}| + a}{t_{n,p-1,i}}, \qquad \omega_p = \frac{r|y_{n,i}| + a}{t_{n,p,i}}, \qquad \omega_{p+1} = \frac{r|y_{n,i}| + a}{t_{n,p+1,i}}.$$

The most appropriate stepsize h_{n+1} is not chosen as the largest of $h_{n,p-1}$, $h_{n,p}$ and $h_{n,p+1}$ but the stepsize which minimises the computational cost. The standard strategy for the order change is to choose the order d in such a way that the distance integrated forward per unit of computational cost is maximised. This is represented by

$$\frac{h_{n+1,d}}{w_d} \to \max,$$

where $d = \{p-1, p, p+1\}$ and w_d is the computational cost of completing one step. It is difficult to estimate the computational cost of a given method, as it depends on the size and complexity of the differential equation system, the number of Newton iterations it requires and the general nature of the system. For Runge-Kutta methods the computational cost is often measured in terms of the number of function evaluations so for the IRKS methods the computational cost is therefore equivalent to $p+1$. This measures the distance forward the formula travels per stage. This has the downfall of assuming that each stage has the same cost. This is not such a problem if the method is diagonally implicit, but if the method is of DESIRE type then the cost of the singly implicit block is greater than that of the diagonal stage. From the various choices of order available, consider the distance forward per stage, for the orders $p-1$, p and $p+1$ methods. The ratios are therefore

$$\frac{h_{n+1,p-1}}{p}, \qquad \frac{h_{n+1,p}}{p+1}, \qquad \frac{h_{n+1,p+1}}{p+2}.$$

In order to balance the effects of assuming the computational cost is equivalent to the number of function evaluations safety factors δ_{p-1}, δ_p and δ_{p+1} are included. The inclusion of these factors prevent a choice of order $p-1$ or $p+1$ unless there is strong evidence that the choice will be beneficial, in terms of computational costs. The measure of efficiency for the methods is given by

$$\delta_{p-1} * \frac{h_{n,p-1}}{p}, \qquad \delta_p * \frac{h_{n,p}}{p+1}, \qquad \delta_{p+1} * \frac{h_{n,p+1}}{p+2}.$$

The main question that remains is how to best accurately approximate $\sigma_{p+1} h_n^{p+2} y^{(p+2)}(x_{n-1})$. It can be argued that it is unnecessary to change the order of the method upwards if the stepsize has not settled to an almost constant state. Therefore, a simple and reasonably robust solution to this problem is to wait until it is appropriate to force three steps to have identical sizes. The choice of three steps is of course heuristic however it may be a coincidence that two steps are almost constant but it is less likely that three steps are almost constant. It is also important not to wait too long before considering whether order change is a good idea. Using a scaled difference between consecutive principal local truncation errors gives

$$\begin{aligned}\frac{\sigma_{p+1}}{\sigma_p}(t_n - t_{n-1}) &= \sigma_{p+1} h^{p+1}\left(y_{n-1}^{(p+1)} - y_{n-1}^{(p+1)} + h y_{n-1}^{(p+2)}\right) + O(h^{p+3}) \\ &= \sigma_{p+1} h^{p+2} y_{n-1}^{(p+2)} + O(h^{p+3}).\end{aligned}$$

One of the clear advantages the IRKS methods have over other general linear methods is that the error constants of the method are free parameters. One of the difficulties faced with certain methods for example, the DESI methods (see [62]), is that the error constants do not always decrease as the order increases. This can make it especially difficult to increase the order past a method where the error constant is larger than the method of one order less. In the development of a code based on the DESI methods, to overcome this problem the solver jumped from order three to order five as the method of order four had this undesirable property.

If the order changing strategy suggests that an increase in order is appropriate, then the current output approximation

$$y^{[n]} = \begin{bmatrix} y_n \\ h_{n+1}y'_n \\ \vdots \\ h_{n+1}^p y_n^{(p)} \end{bmatrix},$$

is of order only p and needs to be modified to order $p+1$. Using the approximation $h_{n+1}^{p+1}y_n^{(p+1)}$, constants β_1, β_2, $\dots$, β_p must be chosen to make the change of order as smooth and stable as possible. Note that these constants are exactly the free parameters from the doubly companion matrix X this is due to the result from Theorem 4.2. Therefore $y^{[n]}$ becomes

$$y^{[n]} = \begin{bmatrix} y_n + \beta_1 h_{n+1}^{p+1}y_n^{(p+1)} \\ h_{n+1}y'_n + \beta_2 h_{n+1}^{p+1}y_n^{(p+1)} \\ \vdots \\ h_{n+1}^p y_n^{(p)} + \beta_p h_{n+1}^{p+1}y_n^{(p+1)} \\ h_{n+1}^{p+1}y_n^{(p+1)} \end{bmatrix}.$$

The principal part of the error incurred in each of the components is the local discretisation error. It has been noticed numerically that this correction does not affect the overall performance of the code very much at all. What is more likely to affect the performance of the code is an understanding of how the error propagates for $O(h^{p+2})$ terms. It is hoped that this will lead to certain conditions on the IRKS methods and determine whether it is possible to eliminate terms of the form $h^{p+2}\frac{\partial f}{\partial y}y^{(p+1)}$.

4.9 Computing the stages

Implicit methods are used for the solution of stiff problems and are much more complicated to use than explicit methods, because the stages are defined implicitly. It is generally regarded that to solve the stages accurately and efficiently the modified Newton's method should be used. The modified Newton's method uses the Jacobian matrix J evaluated at the beginning of the step for each iteration in one step. Often to further increase efficiency the Jacobian is used over several steps. The modified Newton iteration process is defined as

$$\begin{aligned} &(I - hAJ)\Delta Y_l^{[n]} = Z_l^{[n]} \\ &Y_{l+1}^{[n]} = Y_l^{[n]} + \Delta Y_l^{[n]}, \qquad l = 0, 1, 2, \dots \end{aligned}$$

where the Jacobian matrix $J = \frac{\partial f}{\partial y}$ is evaluated at the beginning of the step and

$$Z_l^{[n]} = -Y_l^{[n]} + hAf\big(Y_l^{[n]}\big) + Uy^{[n-1]}.$$

Note that the subscript l denotes iteration number not the component number. The nonlinear Jacobian matrix has $sm \times sm$ equations and to find the LU factors requires $s^3m^3 + O(m^2)$ operations and the back substitutions requires $s^2m^2 + O(m)$ operations. This process is very computationally expensive. To overcome this problem, Butcher [23] proposed using the Jordan canonical form of A rather than A itself in modified Newton iteration process. For any matrix A there exists a nonsingular matrix W such that $W^{-1}AW = \bar{A}$, where $\bar{A}$ is the Jordan canonical form of A. Transform $Y_l^{[n]}$ and $Z_l^{[n]}$, as

$$\bar{Y}_l^{[n]} = W^{-1}Y_l^{[n]}, \qquad \bar{Z}_l^{[n]} = W^{-1}Z_l^{[n]}.$$

Substituting into the first equation of the modified Newton iteration process gives

$$(I - h\bar{A}J)\Delta\bar{Y}_l^{[n]} = \bar{Z}_l^{[n]}.$$

Since only IRKS methods with a one point spectrum have been derived the block upper triangular matrix $I - h\bar{A}J$ has a special structure, that is the diagonal blocks are constant. This significantly reduces the computational process. The new modified Newton iteration process with the use of these transformations is defined as

$$\begin{aligned}
&\bar{Z}_l^{[n]} = W^{-1}Z_l^{[n]} \\
&(I - h\lambda J)\Delta\bar{Y}_{l,k}^{[n]} = \bar{Z}_{l,k}^{[n]} - \sum_{j=1}^{k-1}\Delta\bar{Y}_{l,j}^{[n]}, \qquad k = 1, 2, \ldots, s, \\
&\Delta\bar{Y}_l^{[n]} = W\Delta Y_l^{[n]}, \\
&Y_{l+1}^{[n]} = Y_l^{[n]} + \Delta Y_l^{[n]}, \qquad l = 0, 1, 2, \ldots.
\end{aligned}$$

The above iteration scheme requires the LU factorisation of $m \times m$ matrices $I - h\lambda J$ for each stage. Thus the total of the LU factorisation is sm^3 and sm^2 for the back substitutions. Note that there is the three additional transformations with each requiring s^2m operations. This transformation is especially useful if the system being integrated has high dimension. In order to reduce the costs even further it is possible to consider IRKS methods which are diagonally implicit. When this is the case the transformation matrix $W = I$. For Runge-Kutta methods diagonally implicit schemes have low stage order but for IRKS methods the stage order is equal to the order, therefore, the diagonally implicit methods are likely to be the most competitive with the traditional methods for solving stiff problems.

The problem which now arises is how $Y_0^{[n]}$ should be best chosen so as to minimise the number of iterations used when computing $Y^{[n]}$. One possible approach, which is based on the fact that the IRKS methods have high stage order, is now given. The stages of the current step $Y^{[n]}$ are

$$\begin{aligned}
Y^{[n]} &= y(x_{n-1} + ch_n) + O(h_n^{p+1}) \\
&= Cy^{[n-1]} + O(h_n^{p+1}).
\end{aligned}$$

The stages of the previous step $Y^{[n-1]}$, are

$$\begin{aligned}
Y^{[n-1]} &= y(x_{n-2} + ch_{n-1}) + O(h_{n-1}^{p+1}) \\
&= y(x_{n-1} + (c-e)h_{n-1}) + O(h_{n-1}^{p+1}) \\
&= G\widetilde{y}^{[n-1]} + O(h_{n-1}^{p+1}),
\end{aligned}$$

and the scaled Vandermonde matrix G is

$$G = [\ e \quad c-e \quad \tfrac{(c-e)^2}{2!} \quad \cdots \quad \tfrac{(c-e)^p}{p!}\ ].$$

Assume that the rescale and modify approach to variable stepsize is used. Recalling that

$$h_n f\big(Y^{[n]}\big) = CKy^{[n-1]},$$

it then follows from (4.5.4) and (4.4.3) that

$$\begin{aligned}
y^{[n-1]} &= D(\delta_n)\widetilde{y}^{[n-1]} + \theta(\delta_n)\psi^T h_n f\big(Y^{[n]}\big) \\
&= D(\delta_n)\widetilde{y}^{[n-1]} + \theta(\delta_n)\psi^T CKy^{[n-2]} \\
&= D(\delta_n)\widetilde{y}^{[n-1]}.
\end{aligned}$$

Therefore, the starting value for the Newton iterations can be accurately predicted by either approach to variable stepsize as

$$Y_0^{[n]} = CG^{-1}D\left(\frac{1}{\delta_n}\right)Y^{[n-1]} + O(h^{p+1}),$$

where $h = \min(h_{n-1}, h_n)$. When the method is diagonally implicit it is possible to use the stages already computed in the current step to obtain an even more accurate prediction of the stages. This is not possible for methods where the A matrix is full and is likely to increase the competitive advantage of the diagonally implicit IRKS methods.

4.10 Continuous extensions

After the integration is complete, continuous numerical methods are useful for plotting or event location. That is determining events such as local minimum or local maximum. Continuous methods are also useful during the integration in the strategies or heuristics of the method. Such as, defect control, discontinuity detection and mesh refinement. They are also very useful when solving boundary value problems, delay differential equations or differential algebraic equations.

Since as a design choice the methods have order and stage order p, interpolants can be constructed using only the information from the current step. The continuous interpolant is defined in each subinterval and takes the form

$$\eta(x_{n-1} + \theta h_n) = h_n \tau(\theta)^T f\big(Y^{[n]}\big) + \kappa(\theta)^T y^{[n-1]},$$

where $\eta(x_{n-1} + \theta h_n)$, which is a piecewise polynomial, is the approximation to the solution $y(x_{n-1} + \theta h_n)$ of uniform order p. The vectors $\tau(\theta)^T$ and $\kappa(\theta)^T$ have polynomial components of the form

$$\tau_i(\theta) = \sum_{j=0}^{p_*} \tau_{ij}\frac{\theta^j}{j!}, \qquad \kappa_i(\theta) = \sum_{j=0}^{p_*} \kappa_{ij}\frac{\theta^j}{j!}, \qquad i = 1, 2, \ldots, p+1,$$

where τ_{ij}, κ_{ij} and p_* are constants. The value of p_* must be at least equal to p.

To ensure the interpolant is of uniform order p, with respect to the incoming approximations, the order conditions can be represented by considering a step from x_{n-1} to x_n. The following result formalises this.

Theorem 4.6 *A general linear method in Nordsieck form with order and stage order p has an interpolant $\eta(x_{n-1} + \theta h_n)$ which approximates $y(x_{n-1} + \theta h_n)$ with uniform order p if and only if*

$$\exp(\theta z) = z\tau(\theta)^T \exp(cz) + \kappa(\theta)^T Z + O(z^{p+1}), \tag{4.10.1}$$

where the exp *function is applied componentwise to a vector and $\theta \in (0, 1]$.*

Proof: The interpolant $\eta(x_{n-1} + \theta h_n)$ and the scaled first derivatives $hf\big(Y^{[n]}\big)$ can be represented in terms of $y^{[n-1]}$ using Taylor series expansions about x_{n-1}. Therefore,

$$\begin{aligned}\eta(x_{n-1} + \theta h_n) &= \Theta^T y^{[n-1]} + O(h_n^{p+1}),\\ h_n f\big(Y^{[n]}\big) &= CKy^{[n-1]} + O(h_n^{p+1}),\end{aligned}$$

where the vector Θ^T is given by

$$\Theta^T = \begin{bmatrix} 1 & \theta & \frac{\theta^2}{2!} & \ldots & \frac{\theta^p}{p!} \end{bmatrix}.$$

Substitute the above expressions into the interpolant

$$\Theta^T y^{[n-1]} = \tau(\theta)^T CKy^{[n-1]} + \kappa(\theta)^T y^{[n-1]} + O(h_n^{p+1}).$$

By making the substitution $h_n^k y^{(k)}(x_{n-1})$ equals z^k, the interpolant is now

$$\Theta^T Z = \tau(\theta)^T CKZ + \kappa(\theta)^T Z + O(z^{p+1}).$$

Using the relation (3.2.2), the interpolant condition now becomes

$$\Theta^T Z = z\tau(\theta)^T CZ + \kappa(\theta)^T Z + O(z^{p+1}).$$

The result follows by substituting

$$\begin{aligned} \Theta^T Z &= \exp(\theta z) Z + O(z^{p+1}), \\ CZ &= \exp(cz) + O(z^{p+1}), \end{aligned}$$

into the above interpolant condition. □

To satisfy the uniform order condition (4.10.1), it turns out that $\kappa(\theta)^T$ is completely defined by the choice of $\tau(\theta)^T$. To show this is true, first define the vector Θ_*^T as

$$\Theta_*^T = \begin{bmatrix} 1 & \theta & \frac{\theta^2}{2!} & \cdots & \frac{\theta^{p*}}{p_*!} \end{bmatrix},$$

and the coefficient matrices τ and κ are

$$\tau = \begin{bmatrix} \tau_{1,0} & \tau_{2,0} & \cdots & \tau_{p,0} & \tau_{p+1,0} \\ \tau_{1,1} & \tau_{2,1} & \cdots & \tau_{p,1} & \tau_{p+1,1} \\ \vdots & \vdots & & \vdots & \vdots \\ \tau_{1,p_*-1} & \tau_{2,p_*-1} & \cdots & \tau_{p,p_*-1} & \tau_{p+1,p_*-1} \\ \tau_{1,p_*} & \tau_{2,p_*} & \cdots & \tau_{p,p_*} & \tau_{p+1,p_*} \end{bmatrix},$$

$$\kappa = \begin{bmatrix} \kappa_{1,0} & \kappa_{2,0} & \cdots & \kappa_{p,0} & \kappa_{p+1,0} \\ \kappa_{1,1} & \kappa_{2,1} & \cdots & \kappa_{p,1} & \kappa_{p+1,1} \\ \vdots & \vdots & & \vdots & \vdots \\ \kappa_{1,p_*-1} & \kappa_{2,p_*-1} & \cdots & \kappa_{p,p_*-1} & \kappa_{p+1,p_*-1} \\ \kappa_{1,p_*} & \kappa_{2,p_*} & \cdots & \kappa_{p,p_*} & \kappa_{p+1,p_*} \end{bmatrix}.$$

The vectors $\tau(\theta)^T$ and $\kappa(\theta)^T$ can be expressed as

$$\begin{aligned} \tau(\theta)^T &= \Theta_*^T \tau, \\ \kappa(\theta)^T &= \Theta_*^T \kappa. \end{aligned} \tag{4.10.2}$$

The uniform order condition (4.10.1) is now equivalent to

$$\Theta^T Z = \Theta_*^T \tau CKZ + \Theta_*^T \kappa Z + O(z^{p+1}).$$

After equating the coefficients $z^0, z^1, \ldots, z^p$, it follows that

$$\Theta_*^T \kappa = \Theta^T - \Theta_*^T \tau CK.$$

To eliminate Θ^T and Θ_*^T from the above equation, note that $\Theta^T = \Theta_*^T I_*$ where

$$I_* = \begin{bmatrix} I_{p+1} \\ 0_{p_*-p} \end{bmatrix}.$$

Equating the coefficients $\theta^0, \theta^1, \dots, \theta^{p_*}$, it then follows that

$$\kappa = I_* - \tau C K. \tag{4.10.3}$$

The matrix τ is chosen to obtain certain properties of the interpolant. For example, it may be desirable for the interpolant to give the same values as the method at the abscissae. This and other choices will be discussed.

As was the case for error propagation, see Section 4.3, the interpolant is often more conveniently expressed as

$$\eta(x_{n-1} + \theta h_n) = h_n \tau(\theta)^T f\big(Y^{[n]}\big) + y_{n-1} + \nu(\theta)^T \mathsf{y}^{[n-1]}.$$

This is because the solution component acts quite differently from the rest of the Nordsieck vector. The interpolant is continuous over the whole interval of integration if and only if

$$\eta(x_n-) = \eta(x_n+).$$

Using the rescale and modify approach to variable stepsize given in (4.5.4), the continuity requirement above leads to

$$\begin{aligned}\tau(1)^T h_n f\big(Y^{[n]}\big) &+ y_{n-1} + \nu(1)^T \mathsf{y}^{[n-1]} \\ &= \tau(0)^T h_{n+1} f\big(Y^{[n+1]}\big) + y_n + \nu(0)^T \mathsf{y}^{[n]} \\ &= \tau(0)^T h_{n+1} f\big(Y^{[n+1]}\big) + \big(\mathsf{b}^T + \nu(0)^T\big(\mathsf{D}(\delta_{n+1})\mathsf{B} + \theta(\delta_{n+1})\varphi^T\big)\big) h_n f\big(Y^{[n]}\big) + \\ &\quad y_{n-1} + \big(\mathsf{v}^T + \nu(0)^T\big(\mathsf{D}(\delta_{n+1})\mathsf{V} + \theta(\delta_{n+1})\psi^T\big)\big)\mathsf{y}^{[n-1]}.\end{aligned}$$

Collecting coefficients leads to the following relations

$$\begin{aligned}\tau(0)^T &= 0, \\ \tau(1)^T &= \mathsf{b}^T + \nu(0)^T\big(\mathsf{D}(\delta_{n+1})\mathsf{B} + \theta(\delta_{n+1})\varphi^T\big), \\ \nu(1)^T &= \mathsf{v}^T + \nu(0)^T\big(\mathsf{D}(\delta_{n+1})\mathsf{V} + \theta(\delta_{n+1})\psi^T\big).\end{aligned}$$

To ensure the interpolant is compatible with the discrete numerical solution when $\theta = 0$ then the interpolant should be equivalent to y_{n-1}. Likewise, when $\theta = 1$ then the interpolant should be equivalent to y_n. Therefore $\nu(0)^T = 0^T$ and the above relations reduce to

$$\begin{aligned}\tau(0)^T &= 0^T, \qquad & \nu(0)^T &= 0^T, \\ \tau(1)^T &= \mathsf{b}^T, \qquad & \nu(1)^T &= \mathsf{v}^T.\end{aligned} \tag{4.10.4}$$

These conditions are known as the compatibility conditions for interpolants. These conditions would of course be identical if the simpler rescale approach to variable stepsize was used.

The matrix τ can also be chosen to ensure that the consecutive derivatives of the interpolant are compatible with the Nordsieck vector. When $\theta = 0$ then derivative i of the interpolant should be equivalent to $y_{n-1}^{(i)}$. Likewise, when $\theta = 1$ then derivative i of the interpolant should be equivalent to $y_n^{(i)}$. To satisfy these requirements

$$\frac{1}{h_n^i}\left(\frac{d^i}{d\theta^i}\eta(x_n-)\right) = \frac{1}{h_{n+1}^i}\left(\frac{d^i}{d\theta^i}\eta(x_n+)\right),$$

for $i = 0, 1, \ldots, l$. Using the rescale approach to variable stepsize given by (4.5.1), the requirements above, lead to

$$\begin{aligned}
&\frac{1}{h_n^{i-1}}\tau^{(i)}(1)^T f\big(Y^{[n]}\big) + \frac{1}{h_n^i}\nu^{(i)}(1)^T \mathsf{y}^{[n-1]} \\
&\quad = \frac{1}{h_{n+1}^{i-1}}\tau^{(i)}(0)^T f\big(Y^{[n+1]}\big) + \frac{1}{h_{n+1}^i}\nu^{(i)}(0)^T \mathsf{y}^{[n]} \\
&\quad = \frac{1}{h_{n+1}^{i-1}}\tau^{(i)}(0)^T f\big(Y^{[n+1]}\big) + \frac{1}{h_{n+1}^i}\nu^{(i)}(0)^T \mathsf{D}(\delta_{n+1})\big(\mathsf{B}h_n f\big(Y^{[n]}\big) + \mathsf{V}\mathsf{y}^{[n-1]}\big).
\end{aligned}$$

Collecting coefficients leads to the relations

$$\begin{aligned}
\tau^{(i)}(0)^T &= 0, \\
\tau^{(i)}(1)^T &= \frac{1}{\delta_{n+1}^i}\nu^{(i)}(0)^T \mathsf{D}(\delta_{n+1})\mathsf{B}, \\
\nu^{(i)}(1)^T &= \frac{1}{\delta_{n+1}^i}\nu^{(i)}(0)^T \mathsf{D}(\delta_{n+1})\mathsf{V}.
\end{aligned} \tag{4.10.5}$$

When $\tau^{(i)}(0)^T = 0$ then $\nu^{(i)}(0)^T$ picks off the appropriate component of the Nordsieck vector. Therefore, $\nu^{(i)}(0)^T = e_i^T$, which leads to

$$\frac{1}{\delta_{n+1}^i}\nu^{(i)}(0)^T \mathsf{D}(\delta_{n+1}) = e_i^T.$$

The conditions (4.10.5) are therefore, equivalent to

$$\begin{aligned}
\tau^{(i)}(0)^T &= 0^T, & \nu^{(i)}(0)^T &= e_i^T, \\
\tau^{(i)}(1)^T &= \mathsf{B}_i^T, & \nu^{(i)}(1)^T &= \mathsf{V}_i^T.
\end{aligned} \tag{4.10.6}$$

Using the rescale and modify approach to variable stepsize given by (4.5.4), the above requirements lead to

$$\begin{aligned}
&\frac{1}{h_n^{i-1}}\tau^{(i)}(1)^T f\big(Y^{[n]}\big) + \frac{1}{h_n^i}\nu(1)^T \mathsf{y}^{[n-1]} \\
&\quad = \frac{1}{h_{n+1}^{i-1}}\tau^{(i)}(0)^T f\big(Y^{[n+1]}\big) + \frac{1}{h_{n+1}^i}\nu^{(i)}(0)^T \mathsf{y}^{[n]} \\
&\quad = \frac{1}{h_{n+1}^{i-1}}\tau^{(i)}(0)^T f\big(Y^{[n+1]}\big) + \frac{1}{h_{n+1}^i}\nu^{(i)}(0)^T h_n\big(\mathsf{D}(\delta_{n+1})\mathsf{B} + \theta(\delta_{n+1})\varphi^T\big) f\big(Y^{[n]}\big) + \\
&\qquad \frac{1}{h_{n+1}^i}\nu^{(i)}(0)^T\big(\mathsf{D}(\delta_{n+1})\mathsf{V} + \theta(\delta_{n+1})\psi^T\big)\mathsf{y}^{[n-1]}.
\end{aligned}$$

Collecting coefficients leads to the relations

$$\begin{aligned}
\tau^{(i)}(0)^T &= 0, \\
\tau^{(i)}(1)^T &= \frac{1}{\delta_{n+1}^i}\nu^{(i)}(0)^T\big(\mathsf{D}(\delta_{n+1})\mathsf{B} + \theta(\delta_{n+1})\varphi^T\big), \\
\nu^{(i)}(1)^T &= \frac{1}{\delta_{n+1}^i}\nu^{(i)}(0)^T\big(\mathsf{D}(\delta_{n+1})\mathsf{V} + \theta(\delta_{n+1})\psi^T\big).
\end{aligned} \tag{4.10.7}$$

The second part of the right hand side of the the last two equations above is

$$\begin{aligned}
\frac{1}{\delta_{n+1}^i}\nu^{(i)}(0)^T\theta(\delta_{n+1}) &= \frac{1}{\delta_{n+1}^i}\nu^{(i)}(0)^T\big(\mathsf{D}(\delta_{n+1}) - \delta_{n+1}^{p+1}I\big)\beta \\
&= (1 - \delta_{n+1}^p)e_i^T\beta \\
&= \beta_{p+1-i}(1 - \delta_{n+1}^p).
\end{aligned}$$

The conditions (4.10.7) are therefore, equivalent to

$$\begin{aligned} \tau^{(i)}(0)^T &= 0^T, & \nu^{(i)}(0)^T &= e_{i+1}^T, \\ \tau^{(i)}(1)^T &= \mathsf{B}_i^T + \beta_{p+1-i}(1-\delta_{n+1}^p)\varphi^T, & \nu^{(i)}(1)^T &= \mathsf{V}_i^T, \end{aligned}$$

if the local error estimate is given by (4.4.3).

The matrix τ can be chosen so as to ensure the interpolant is sufficiently continuous. This leads to the following theorem.

Theorem 4.7 *If the interpolant $\eta(x_{n-1}+\theta h_n)$ is l times continuously differentiable and*

$$\begin{aligned} \tau^{(i)}(0)^T &= 0^T, & \kappa^{(i)}(0)^T &= e_{i+1}^T, \\ \tau^{(i)}(1)^T &= B_{i+1}^T, & \kappa^{(i)}(1)^T &= V_{i+1}^T. \end{aligned} \tag{4.10.8}$$

for $i = 0, 1, \ldots, l$, then the first $l+1$ unscaled (that is the powers of h are removed) components of the Nordsieck vector are continuous. For this to hold

$$p_* \geq 2l+1,$$

and this implies the matrix τ can have the form

$$\tau = \begin{bmatrix} 0 \\ L^{-1}B \end{bmatrix}, \tag{4.10.9}$$

where the Toeplitz matrix L is given by

$$L = \begin{bmatrix} \frac{1}{(l+1)!} & \frac{1}{(l+2)!} & \frac{1}{(l+3)!} & \cdots & \frac{1}{(p_*-2)} & \frac{1}{(p_*-1)!} & \frac{1}{p_*!} \\ \frac{1}{l!} & \frac{1}{(l+1)!} & \frac{1}{(l+2)!} & \cdots & \frac{1}{(p_*-3)!} & \frac{1}{(p_*-2)!} & \frac{1}{(p_*-1)!} \\ \frac{1}{(l-1)!} & \frac{1}{l!} & \frac{1}{(l+1)!} & \cdots & \frac{1}{(p_*-4)!} & \frac{1}{(p_*-3)!} & \frac{1}{(p_*-2)!} \\ \vdots & \vdots & & & \vdots & \vdots & \vdots \\ \frac{1}{3!} & \frac{1}{4!} & \frac{1}{5!} & \cdots & \frac{1}{(l+1)!} & \frac{1}{(l+2)!} & \frac{1}{(l+3)!} \\ \frac{1}{2!} & \frac{1}{3!} & \frac{1}{4!} & \cdots & \frac{1}{l!} & \frac{1}{(l+1)!} & \frac{1}{(l+2)!} \\ \frac{1}{1!} & \frac{1}{2!} & \frac{1}{3!} & \cdots & \frac{1}{(l-1)!} & \frac{1}{l!} & \frac{1}{(l+1)!} \end{bmatrix}.$$

Proof: Assuming that (4.10.3) holds it is necessary only to consider the conditions (4.10.8) which involve τ. The conditions (4.10.8) are made up of the conditions (4.10.4) and (4.10.6). Note that $\tau^{(i)}(0)^T = 0^T$, for $i = 0, 1, \ldots, l$ ensures the first $l+1$ rows of τ are zero. Note that derivative i of Θ_* is

$$\left(\frac{d}{d\theta}\right)^i \Theta_*^T = \Theta_*^T K_*^i,$$

where K_* has the same form as K but is of size $(p_*+1)\times(p_*+1)$. The system of equations left to satisfy can be represented as follows

$$\begin{bmatrix} \Theta_*^T|_{\theta=1} \\ \Theta_*^T K_*|_{\theta=1} \\ \vdots \\ \Theta_*^T K_*^{l-1}|_{\theta=1} \\ \Theta_*^T K_*^l|_{\theta=1} \end{bmatrix} \tau = \begin{bmatrix} B_1^T \\ B_2^T \\ \vdots \\ B_l^T \\ B_{l+1}^T \end{bmatrix}.$$

Note that the matrix of the left hand side of the above equation is just the first $l+1$ rows of the matrix $\exp(K_*)$. Since the first $l+1$ rows of τ are zero, then if $p_*+1 = 2(l+1)$ there are sufficient conditions left in τ to satisfy the above system of equations. Let L be defined as above, which is equivalent to the matrix formed from the first $l+1$ rows and last $l+1$ columns of $\exp(K_*)$, then it follows that τ is given by (4.10.9). □

For the rescale and modify approach to variable stepsize, τ needs to be recomputed each time the stepsize is varied. This is somewhat unsatisfactory.

The matrix τ can be used to ensure that the interpolant matches the stage values at the abscissae points. Interpreting the interpolant in a component by component sense, this is the same as saying

$$\eta(ex_{n-1} + ch_n) = Ah_n f\big(Y^{[n]}\big) + Uy^{[n-1]}.$$

This implies that the matrices $\tau(c)$ and $\kappa(c)$, which have rows $\tau(c_i)^T$ and $\kappa(c_i)^T$, are

$$\tau(c) = A, \qquad \kappa(c) = U.$$

Using the expression (4.10.2), the above equations are

$$C_*\tau = A, \qquad C_*\kappa = U,$$

where the scaled Vandermonde matrix C_* is

$$C_* = \begin{bmatrix} e & c & \frac{c^2}{2!} & \cdots & \frac{c^{p-1}}{(p-1)!} & \frac{c^{p_*}}{p_*!} \end{bmatrix}.$$

Since only methods which are stiffly accurate are being considered, no problem arises when one of the abscissae equals one, because this stage is exactly equal to first output approximation and therefore (4.10.4) still holds. Since the IRKS method is stiffly accurate it is sufficient to choose $p_* = p$ and τ and κ as

$$\tau = C^{-1}A, \qquad \kappa = C^{-1}U.$$

Notice these conditions satisfy the expression (4.10.3).

CHAPTER 5

Numerical experiments

Comparing numerical methods is always a difficult process. This is due to the number of design decisions that need to be made when implementing a numerical method. These experiments performed below are of a preliminary nature. Their inclusion will hopefully verify that the IRKS methods have the potential to be the kernel of a competitive code.

5.1 Framework

Much effort has been placed in the development of numerical methods for ordinary differential equations. Comparing numerical methods is difficult because there is usually so many particular features or heuristics that need to be used when implementing any particular method. The paper by Hull, Enright, Fellen and Sedgwick [84] provided a basis for making comparisons. However, the DETEST program is really designed to compare implementations which have been well tested. It does not really explain which parts of the implementation may need to be improved. The approach used in this chapter is to compare the IRKS methods with traditional methods using a range of tests to determine which parts of the implementation need more emphasis. All the preliminary codes based on IRKS methods have been written in Matlab and since the program DETEST is written in Fortran it makes it difficult to compare anyway. The problems used in all the tests are those listed in [84]. There are five classes of problems A, B, C, D and E, each class containing five problems. For the convenience of the reader these problems have been listed in Appendix II. It should be noted that all the experiments performed in this chapter are on non-stiff problems. The thesis of Huang [83] contains more thorough analysis of the implementation questions and experiments for stiff problems.

The first experiment compares three well known Runge-Kutta methods with three IRKS methods using fixed stepsize. This should provide evidence as to whether the method in itself is likely to be competitive with traditional methods. The second experiment will consider the effect of variable stepsize on the method. This will be done in a controlled way, that is to say the method will be required to vary the stepsize to a pre-described pattern. Varying the stepsize in this way makes it possible to determine the effect variable stepsize has on the method. Fixed order, variable stepsize IRKS implementations will be compared with various fixed order, variable stepsize Runge-Kutta methods in experiment three. In order to avoid the effects of the heuristic choices that need to be made in any implementation the stepsize control procedure will be the same as that of the Runge-Kutta method being compared. In experiment four a suite of fixed order, variable stepsize methods will be compared with a variable order, variable stepsize implementation based on the same set of methods. Does the variable stepsize, variable order implementation choose the best method available? The final experiment compares the variable order, variable stepsize implementation with the code based on the Adams-Bashforth, Adams-Moulton predictor corrector method from the Matlab ODE suite [109].

5.2 Fixed stepsize

The first experiment compares three well known and competitive Runge-Kutta methods with three IRKS methods. The first Runge-Kutta method was developed by Bogacki and Shampine [6] and is of order three, it has the first same as last (FSAL) property and is the method used in the code `ode23` from Matlab. The second Runge-Kutta method was developed by Nørsett, the coefficients are included in [66] and is of order four with five stages. The third Runge-Kutta method is the well known method developed by Dormand and Prince [63]. The method is of order five has seven stages but has the FSAL property. This method forms the basis of the code `ode45` from Matlab. The three IRKS methods are of orders three, four and five and have one more stage than the order of the method. All are strictly stable and have been found by minimising the local error coefficients of the underlying one-step method. In Appendix III all the methods are given with some extra information concerning error constants and the like.

Since the Runge-Kutta method of order three has effectively three stages and the IRKS method of order three, has four stages then for fair comparisons the Runge-Kutta method uses a step $3/4h$, where h is the size of the stepsize of the IRKS method. For the same reasons the stepsize for the Runge-Kutta and IRKS methods of order four and five are the same.

All problems of the DETEST set were solved using each of the six methods. What is plotted in the three figures is the relative accuracy of the IRKS method over the Runge-Kutta method. Relative accuracy is measured as follows: the logarithm, to the base 10, of the absolute value of the ratio of the Runge-Kutta and the corresponding IRKS errors. Therefore $+1$ indicates that the IRKS method is exactly one decimal place more accurate than the Runge-Kutta method and -1 indicates that the Runge-Kutta method is exactly one decimal place more accurate than the IRKS method.

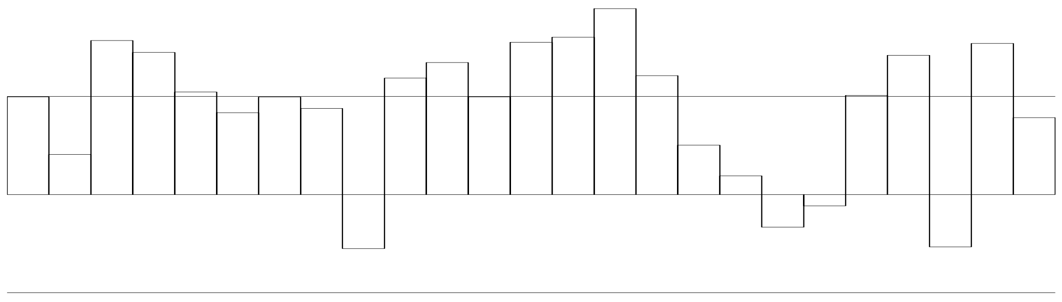


Figure 5.1: Relative accuracy of the IRKS method over the third order Runge-Kutta method.

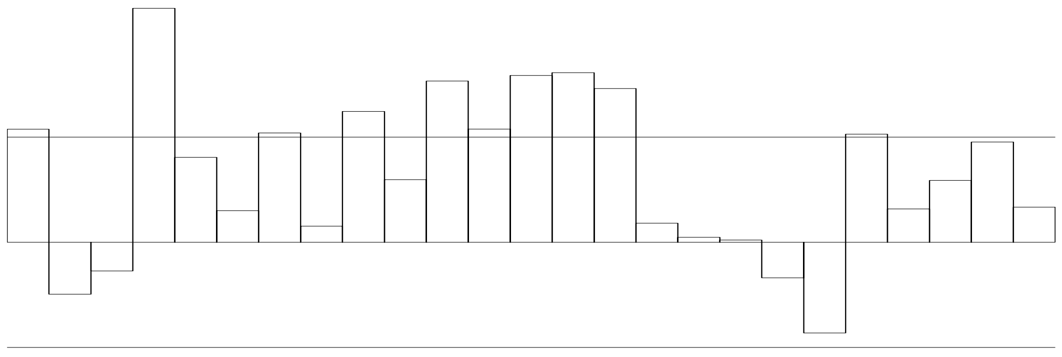


Figure 5.2: Relative accuracy of the IRKS method over the fourth order Runge-Kutta method.

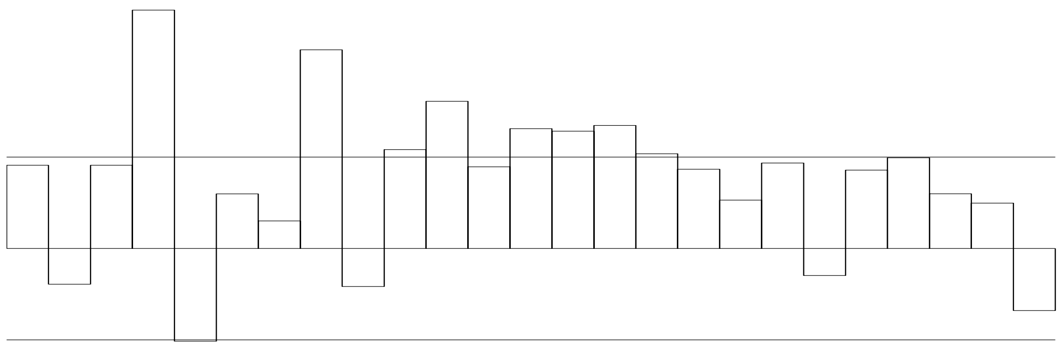


Figure 5.3: Relative accuracy of the IRKS method over the fifth order Runge-Kutta method.

On the majority of problems the IRKS methods outperformed the corresponding Runge-Kutta methods. It is now of interest to compare the effects when variable stepsize is used.

5.3 Prescribed variable stepsize

The second experiment compares exactly the same six methods discussed in the previous section. Again the DETEST set were solved by each of the six methods and again the relative accuracy of the IRKS method over the Runge-Kutta method is plotted in the six figures. The difference in this test is that constant stepsizes are not used but rather a particular stepsize pattern was performed. For each sequence of five steps, the steps are in the ratio $1 : r : r^2 : r : 1$, where $r = 2$ is chosen. Obviously this sort of stepsize pattern would never be used in practice but it is a good means of determining how a method reacts to stepsize change. The expected performance of an order p method using this stepsize changing pattern can be determined if asymptotic behaviour is assumed. The global truncation errors would be scaled up by the factor $F_p(r)$ if the number of steps is independent of r, where

$$F_p(r) = \left(\frac{2+2r+r^2}{5}\right)^{-(p+1)} \left(\frac{2+2r^{p+1}+r^{2(p+1)}}{5}\right).$$

To understand this formula first notice that the length of a five constant steps over the interval $1 + r + r^2 + r + 1$ is $(2 + 2r + r^2)/5$ so the error contributed in the constant stepsize case is

$$\left(\frac{2+2r+r^2}{5}\right)^{p+1}.$$

However in the variable stepsize case each step has an error; summing these errors and averaging gives

$$\left(\frac{2+2r^{p+1}+r^{2(p+1)}}{5}\right).$$

Taking the ratio gives the required formula. It would be expected that the factor $F_p(r)$ would be a minimum scale up factor as this factor does not take into account that additional errors are introduced by the stepsize changing process. Included in the six figures below is a dashed line this indicates where the relative accuracy of the IRKS method over the Runge-Kutta method would have been provided no error was introduced by the process of changing stepsize. This enables an easy comparison and determines the net effect of changing stepsize. For orders three, four and five $F_p(2)$ is 3.6250, 6.8125 and 13.206 respectively.

All problems of the DETEST set were again solved using each of the six methods. What is plotted in the three figures is the relative accuracy of the IRKS method over the Runge-Kutta method with constant stepsize with the dotted line and the relative accuracy of the IRKS method over the Runge-Kutta method with variable stepsize with the thick line. Thus it is easy to compare the effect of the variable stepsize.

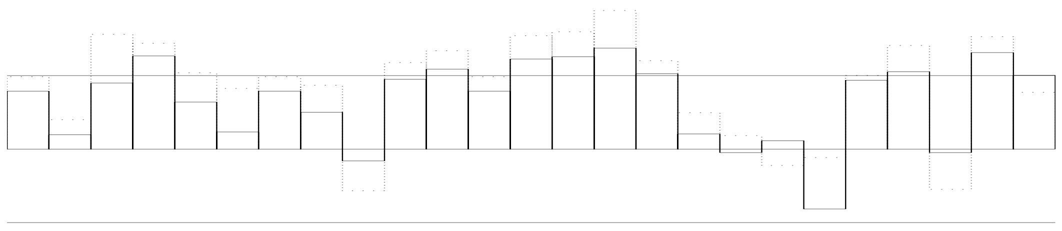


Figure 5.4: Relative accuracy of the IRKS method over the third order Runge-Kutta method.

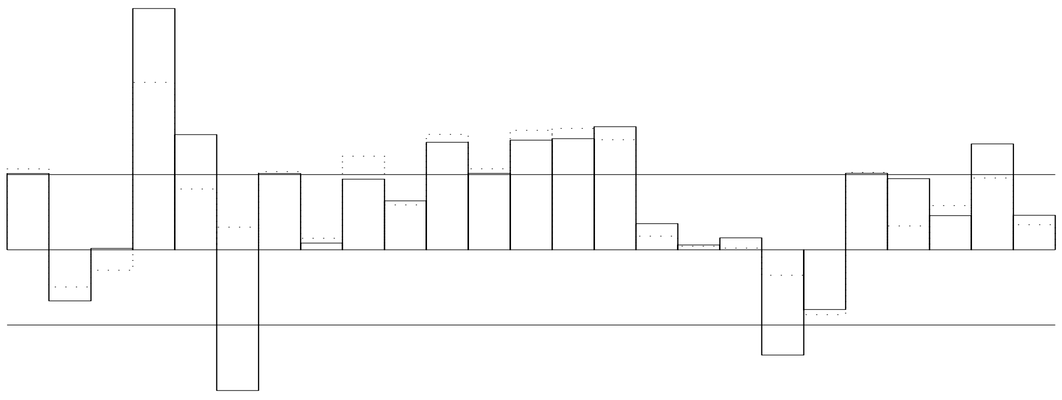


Figure 5.5: Relative accuracy of the IRKS method over the fourth order Runge-Kutta method.

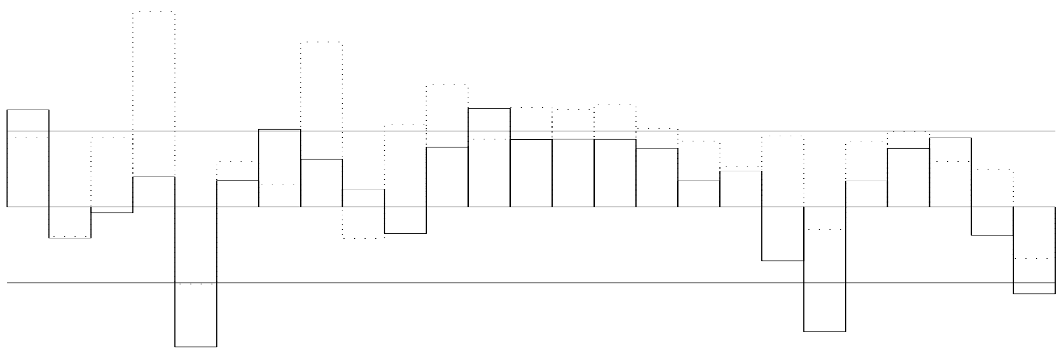


Figure 5.6: Relative accuracy of the IRKS method over the fifth order Runge-Kutta method.

On the majority of problems the IRKS methods still outperforms the corresponding Runge-Kutta methods. It is apparent from the figures that the negative effect of variable stepsize of the IRKS methods increases as order increases. It is also interesting to note that the IRKS methods are affected more by the change in stepsize than the corresponding Runge-Kutta methods. It turns out that this depends very much on the method but in what way is not clear. Determining what features a method should have in order to minimise the effects of variable stepsize will be considered in future work.

5.4 Variable stepsize

When solving, for example, Hamiltonian problems with symplectic methods it is necessary to keep the stepsize of the method constant. Otherwise certain properties of the method are destroyed. In fact it is quite common in the field of geometric numerical integration to require fixed stepsize. Unless the problem requires fixed stepsize, in practice ordinary differential equations are never solved using fixed stepsize schemes. This is because the solution can be rather smooth and easy to integrate in parts of the trajectory while, for example, very oscillatory in other parts requiring a much smaller stepsize. It is therefore important to compare different methods implemented in a variable stepsize scheme. This in some sense compares the effectiveness of the different local error estimates. As mentioned above, IRKS methods incur more error in the process of changing stepsize change than corresponding Runge-Kutta methods. This is because the higher derivative approximations need to be rescaled.

The following figures record the number of function evaluations rather than `etime` or `flops`. The reason for this is that with shared computer resources `etime` is a very unreliable measure. While `flops` and function evaluations should be relatively similar, function evaluations are easier to measure. The following three figures plot the number of function evaluations versus the global error in a double logarithm scale. Figure 5.7 compares the order three methods, Figure 5.4 compares the order four methods and finally Figure 5.9 compares the order five methods. There are fifteen plots in each figure. The five rows represent the problem classes A, B, C, D and E from the DETEST set, and the three columns represent the problems 1,3 and 5 within these classes. These tests are run with many different tolerances, `rtol`$= 10^{-i}$, $i = 2, 3 \ldots, 8$ and `atol`$=$`rtol`. Each tolerance has the same initial stepsize. The IRKS methods are plotted using a continuous line with $\times$ at the output points, whereas the Runge-Kutta method is plotted using the dot dashed line with $\circ$ at the output points.

The order three IRKS method outperforms the corresponding Runge-Kutta on almost all problems. The number of rejected steps is slightly less for the general linear method, this can be attributed to the fact that the local error estimate is asymptotically correct whereas the local error estimate of the Runge-Kutta is not. This situation is likely to improve when for example the effect of rescaling the Nordsieck vector is better understood. That is determine what properties a method should have, in order to minimise this effect. The order four and five IRKS methods outperforms the corresponding Runge-Kutta methods on almost all problems as well. It is worth noticing that the difference in the methods is narrowing as the order increases. This is partly due to the fact that the order five Runge-Kutta method is highly regarded as an optimal method and also the effect of the rescaling error is increased because more information is being passed from step to step.

Overall these results are very promising for the IRKS methods and encourage further investigation, namely the use of variable order as well as variable stepsize. Even though there is a negative effect on the IRKS methods due to the rescaling, the local error estimates perform well. Another use of high stage order is that order p continuous solutions can be obtained without any additional function evaluations being required. It is becoming more and more important to have continuous solutions rather than the discrete solutions, especially for plotting phase portraits and for the use in solving delay differential equations. The relative performance of the IRKS methods over the Runge-Kutta methods is likely to improve in these situations due to the high order continuous solutions which are available. In fact several preliminary numerical experiments were performed on delay differential equations and the results were very encouraging, even though no special care was taken about the discontinuities which are propagated throughout the interval. A more thorough investigation of the use of IRKS methods for delay differential equations will be conducted in the future.

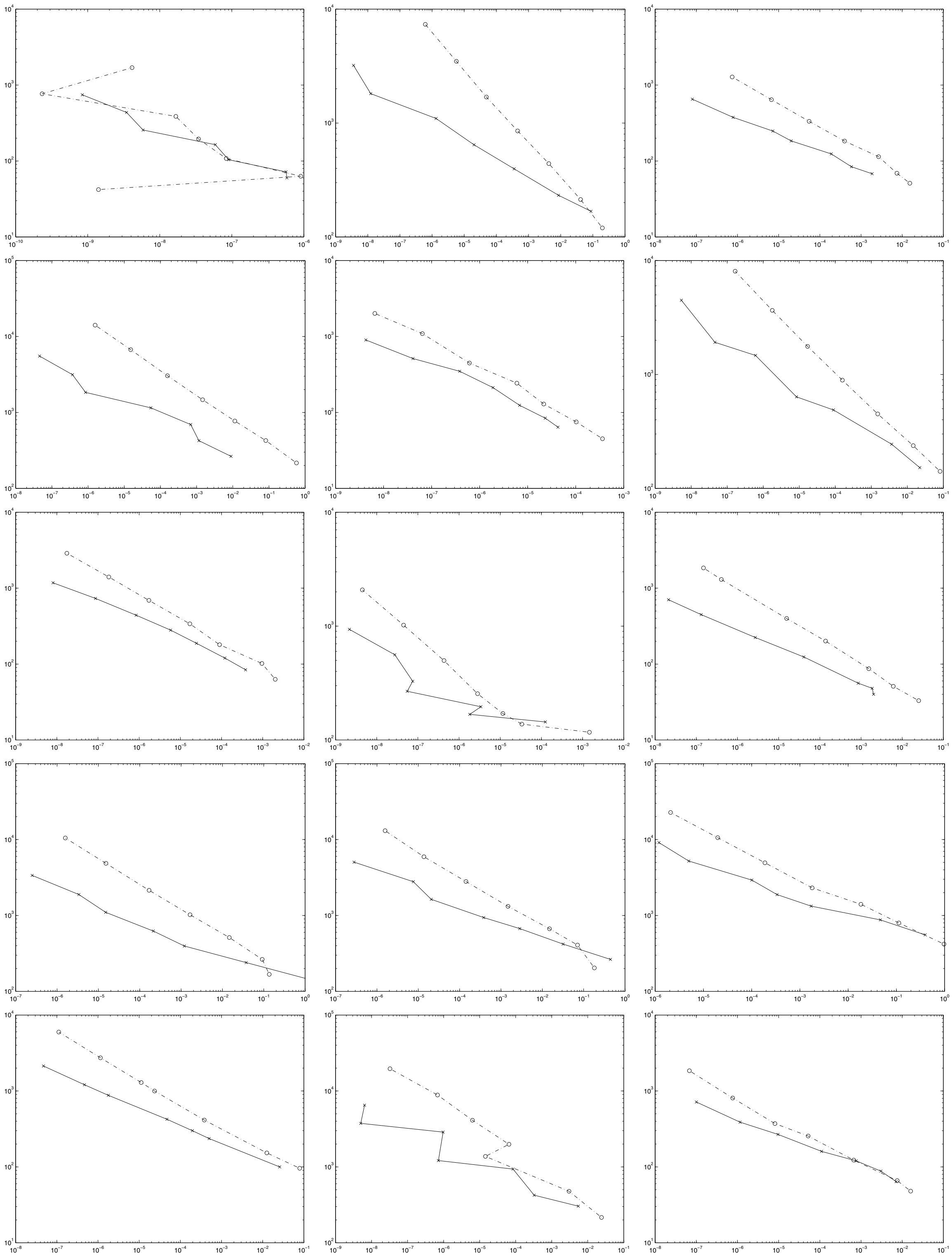


Figure 5.7: Order 3 Runge-Kutta and IRKS methods on all 1,3 and 5 DETEST problems.

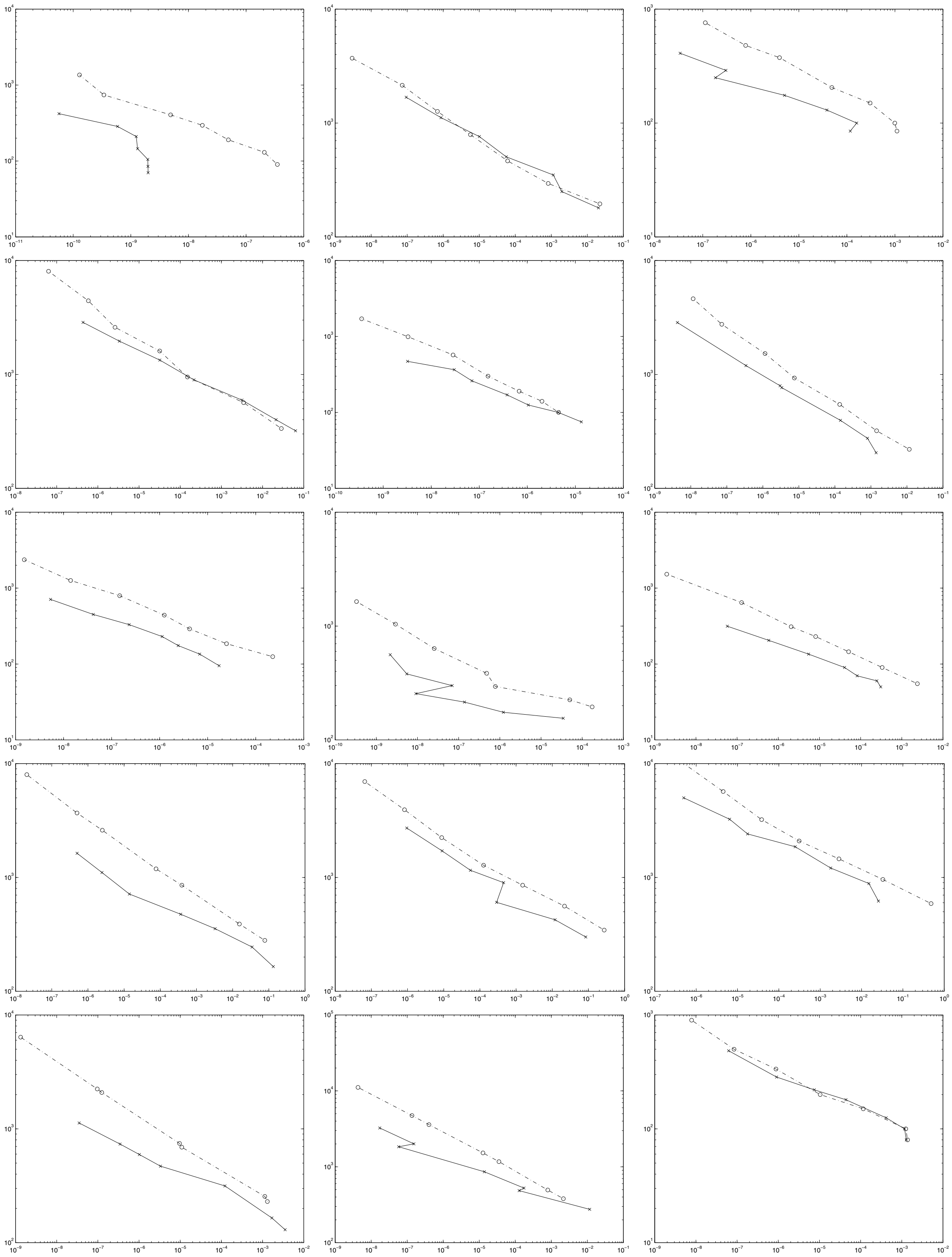


Figure 5.8: Order 4 Runge-Kutta and IRKS methods on all 1,3 and 5 DETEST problems.

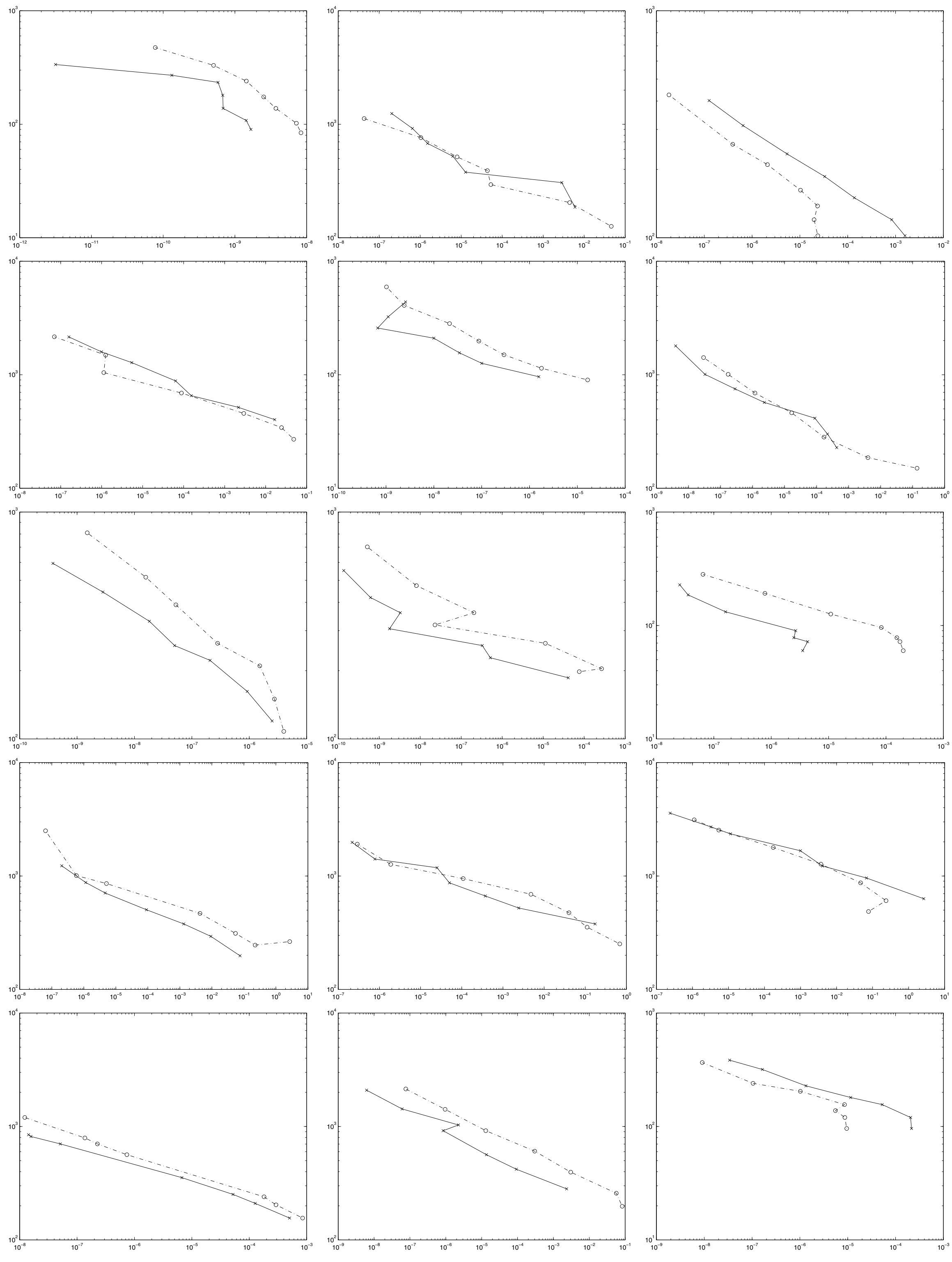


Figure 5.9: Order 5 Runge-Kutta and IRKS methods on all 1,3 and 5 DETEST problems.

5.5 Comparing fixed and variable order implementations

Given a suite of IRKS methods, a variable order implementation using these methods could be considered effective if given a tolerance the code performs exactly as the best code among the fixed order implementations. The following experiment compares the order two method plotted as a solid line, the order three method plotted as a dashed line, the order four method plotted as a dash dot line, the order five method plotted as a dotted line. All fixed order methods have $\circ$ plotted at the output points. The variable order code based on the same methods is plotted with solid line with $\times$ at the output points. This experiment compares the effectiveness of the order changing strategy. This of course is a very important implementation question. An effective code should not be integrating at a sub optimal order. Each test plots the number of function evaluations versus the global error in a double logarithm scale. The DETEST problems A2, B1, B4, C1, C5, D2, D4, E1 and E5 are plotted from left to right. These tests use tolerances `rtol`$= 10^{-i}$ where $i = 2, 3, \dots, 8$ and `atol=rtol`.

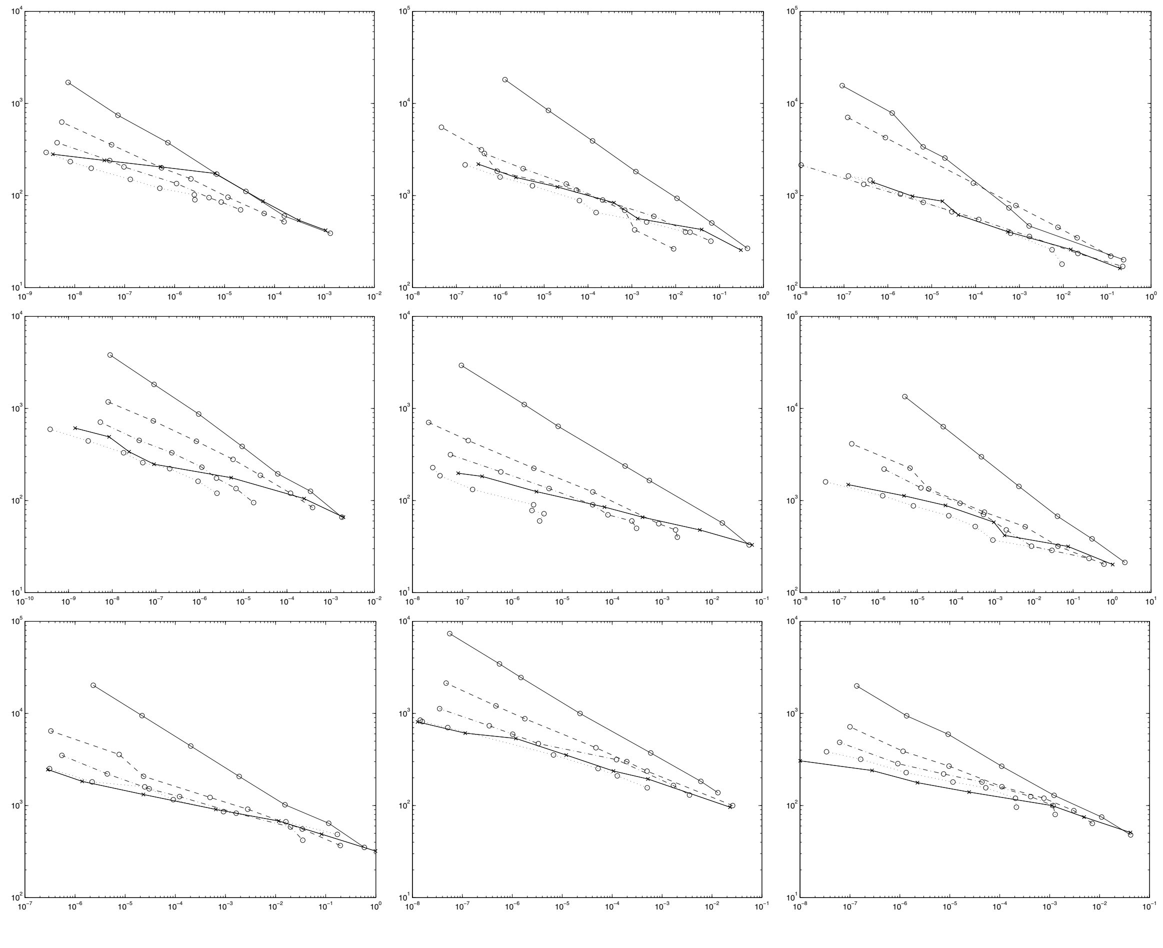


Figure 5.10: Order 2,3,4,5 and variable order IRKS methods on selected DETEST problems.

Generally the variable order code is integrating with the best method available. It could be said that in some cases it does not move up to the most efficient order quickly enough. What seems to be an important question for a variable stepsize, variable order implementation is to find a suite of methods which act well together. That is, if say a method of order g has bad error coefficients the code may choose to integrate with order $g - 1$ even though the optimal order may be greater than order g. In this case the order changing strategy seems to be working sufficiently well.

5.6 Variable stepsize, variable order

For the solution of non-stiff problems, variable order implementations based on Runge-Kutta methods are generally not available. This is primarily due to the fact that it is difficult to get accurate estimates of the local truncation error of a method of higher order than is currently in use. To obtain such an estimate for an explicit method would require a large number of additional function evaluations. For stiff problems, variable order implementations are available because methods exist with higher stage order. One such code STRIDE based on SIRK methods where the stage order is equal to the order is given in [14]. Another approach which has proved useful is to base the order changing strategy not on information about the higher order derivatives as in the case of STRIDE, but on information about the convergence rates from the Newton iteration process. This has been successfully used in the code RADAUP, see [80].

Generally all variable order implementations for non-stiff problems are based on Adams methods. Usually a PECE approach is used since this enables the use of the Milne's device as a means of estimating the local truncation error. There are two main techniques available for adapting the Adams methods to a variable stepsize environment. The first technique requires the coefficients of the method to be derived each time the stepsize is changed. The second technique is based on the use of Nordsieck vectors. This is the technique used for the IRKS methods.

The following experiment compares the IRKS methods with the `ode14` which is a fully variable stepsize, PECE implementation in terms of modified divided differences of the Adams-Bashforth-Moulton family of formulas of orders one through to four. Local extrapolation is done. This is exactly the same code as `ode113` from Matlab except the maximum order has been limited to four. The variable stepsize, variable order code based on the IRKS methods for orders one through five is used in the comparisons. The maximum order is five rather than four since the PECE code uses local extrapolation to increase the order. These tests will be run on all the DETEST problems with indices 2, 3 and 4. The tolerances for both codes are, `rtol`$= 10^{-i}$, $i = 2, 3 \ldots, 8$ and `atol=rtol`. Each tolerance does not result in the same initial stepsize. The IRKS methods are plotted using a continuous line with $\times$ at the output points, whereas the Runge-Kutta method is plotted using the dot dashed line with $\circ$ at the output points.

The PECE code `ode113` is a descendent of ODE/STEP, INTRP [110] which is regarded as a highly efficient code. It is surprising that at this early stage in the development of IRKS methods the IRKS and PECE implementations provide very similar results on almost all problems from the DETEST set. The IRKS code has not been optimised in any way and it is likely that further improvements can be made. Improvements are likely when more is understood about how best to vary stepsize, vary order, approximate local truncation errors and various other implementations issues of these methods.

The results in the chapter provide very encouraging results for the IRKS methods. It is likely that these methods will provide the kernel for an efficient implementation. All the experiments in this chapter have been concerned only with non-stiff problems. Since the implicit and explicit methods are so similar, which is contrary to the case for both the Runge-Kutta and linear multistep methods, it is likely that not only will it be possible to develop variable stepsize, variable order codes based on stiff methods (this is the subject of a thesis by Huang [83]) but it may also be possible to obtain variable stiffness. This would use a suite of explicit and implicit methods from order one through to say eight and swap between the explicit and implicit methods when the stiffness changing procedure suggests. It is hoped that this possibility can be investigated in the future.

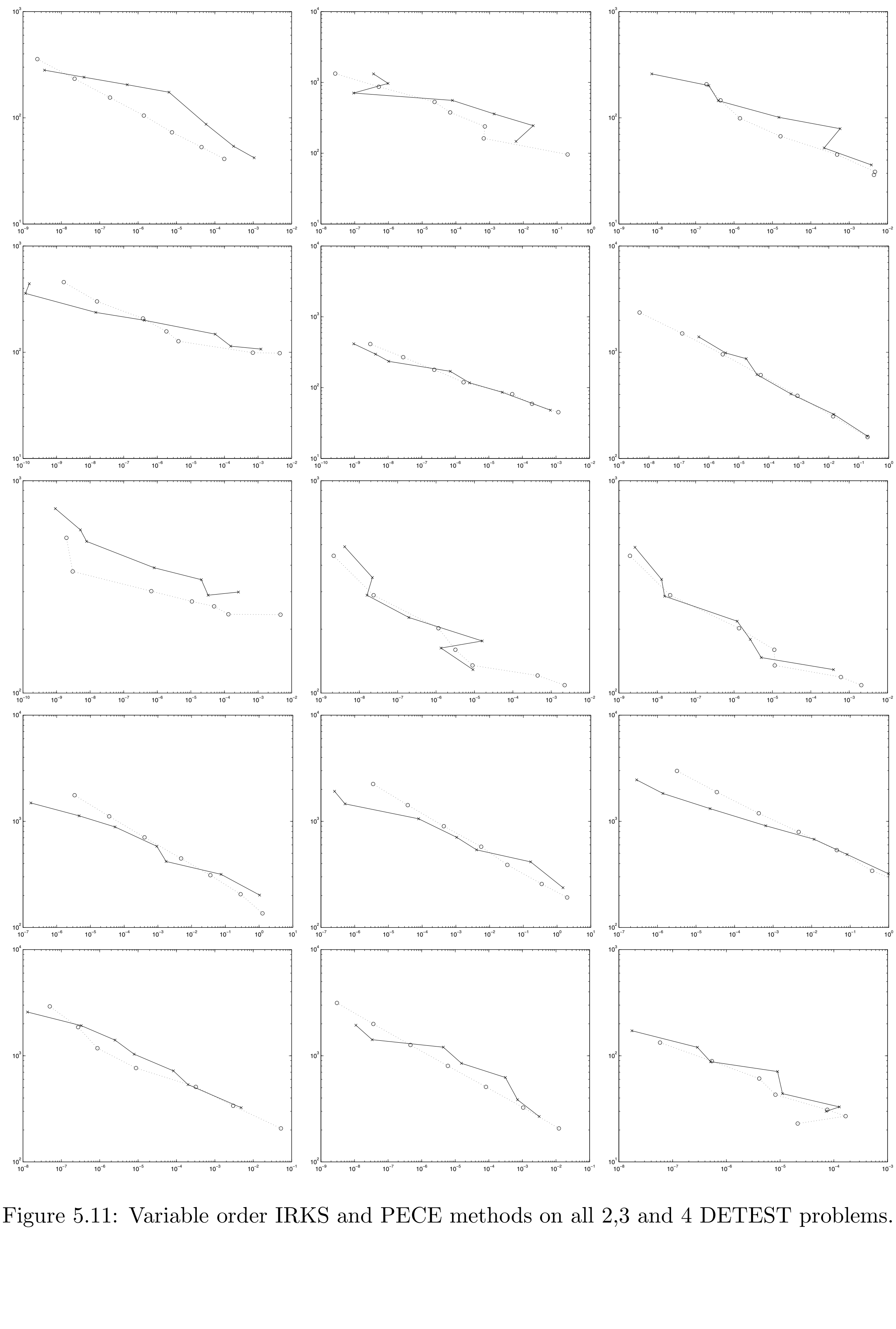


Figure 5.11: Variable order IRKS and PECE methods on all 2,3 and 4 DETEST problems.

CHAPTER 6

Conclusions and future work

The main aim of this thesis was to construct a class of general linear methods which has clear advantages over the traditional methods, while avoiding many of their disadvantages. The IRKS methods discussed in this thesis seem to provide an excellent balance between the advantages and disadvantages of the traditional methods.

6.1 Framework

In Chapter 1 a review of the traditional methods used for numerically solving ordinary differential equations was given. This showed the logical connections between the traditional methods and the more recent attempts at locating competitive general linear methods. The phenomenon of stiffness was discussed along with several other more recent problems which require either methods with special properties or, for example, continuous rather than discrete solutions.

A review of general linear methods was given in Chapter 2. The use of the B-series to understand the order of general linear methods in a general context was discussed. Also the concept of the underlying one step method was considered, which was later used to search for IRKS methods with minimal local truncation errors. Linear and non-linear stability for general linear methods was included. This was used to justify the decision that practical general linear methods should have Runge-Kutta stability. In order to develop practical general linear methods it is necessary to relate the defining parameters, that is, the order, the stage order,the number of approximations passed from step to step and the number of stages in a certain way. The main choice of these defining parameters discussed in this thesis is when the order equalled the stage order, the number of approximations passed from step to step was equal to the number of stages and the number of stages was one greater than the order. This choice allows a Nordsieck vector to be passed from step to step allowing a convenient means of varying the stepsize. When stage order equals order, methods used for stiff problems do not exhibit order reduction, and asymptotically correct error estimates and uniform order continuous solutions can be derived with no extra cost.

Using the particular choice of defining parameters and the desire for the methods to have Runge-Kutta stability, Chapter 3 considers the class of methods with IRKS. The IRKS conditions are sufficient conditions which relate the defining matrices and ensure the method has Runge-Kutta stability. In order to derive methods, a certain transformation is used to represent all the IRKS methods in a standard form. This allowed a procedure to be developed which resulted in the IRKS methods being able to be derived using only linear operations. There are several free parameters of the methods which must be chosen a-priori. These are the abscissae vector c, the coefficients $\beta_1, \beta_2, \ldots, \beta_p$, the $p \times p$ unit lower triangular matrix T, the z^{p+1} coefficient of the numerator of the stability function ϵ and the $(p+1) \times (p+1)$ matrix W. If the matrix W is chosen in a special way, the ESIRK and DESIRE methods are special cases of the IRKS methods. These new methods, where the number of stages was one greater than the order, may have advantages over the previously known methods where the number of stages and the order were equal. If the last row of the matrix T^{-1} was chosen in a particular way, methods which are analogous to the stiffly accurate Runge-Kutta methods can be derived. These methods may well have advantages for differential algebraic problems.

Chapter 4 discussed several issues regarding implementation. These are starting procedures, error estimation, constructing continuous solutions, techniques for variable stepsize and variable order, predicting the stages and computing the stages using a modified Newton's method. Also the error propagation of the IRKS methods was discussed when fixed or variable stepsize were used. The error propagation of the IRKS methods in variable stepsize can act as if the implementation was a fixed stepsize. What was surprising is that the free parameters of the IRKS methods, especially ϵ and $\beta_1, \beta_2, \ldots, \beta_p$, have a large control over how the errors propagate.

Several codes based on the techniques described above were implemented in Chapter 5. These were: fixed order, fixed stepsize; fixed order, variable stepsize; fixed order, fixed variable stepsize; variable order, variable stepsize. All implementations compared extremely well with the standard codes. The fact that these experimental implementations worked well seems to give sufficient evidence that the IRKS methods may in fact be a suitable kernel for an efficient code.

6.2 Contributions

In writing this thesis, a style which emphasises uniformity of exposition has been used. Since this is at the expense of a clear separation of new and existing ideas, the following summary lists the main achievements of this thesis:

1. The derivation of methods which are directly in Nordsieck form. This leads to the use of doubly companion matrices in the definition of IRKS methods.

2. Development of some results of doubly companion matrices, which were useful for the derivation of IRKS methods.

3. Construction of IRKS methods which used the doubly companion matrices in the definition, using a suitable transformation which represented all methods in a standard form. This results in the derivation of methods using only linear operations.

4. Showed how ESIRK and DESIRE methods can be considered as special cases of the IRKS methods. In doing so, a new class of ESIRK methods where $s = p + 1$ was developed. It was also shown how the free parameters must be chosen so that the method is a DESIRE method.

5. Conditions were found on the free parameters so that the resulting IRKS method has a property known as strongly stiffly accurate, which is a slight extension of the stiffly accurate Runge-Kutta methods.

6. The concept of the underlying one step method was used to search for IRKS methods in which the underlying one step method had minimal local truncation error coefficients. This approach of finding good general linear methods is only feasible if the class of methods in consideration can be determined using linear operations.

7. Showed that the composition of two IRKS methods is again an IRKS method if the number of approximations passed from step to step is the same, along with doubly companion matrices of the methods being the same, except for the first row.

8. Development of a set of fixed order, variable stepsize codes based on explicit IRKS methods. These codes performed extremely well when compared to fixed order implementations of well known Runge-Kutta methods.

9. A suite of IRKS methods of orders two through five was constructed using the underlying one step method approach. These were then implemented in a variable order, variable stepsize code, which performed at least as well as the implementations based on the well known linear multistep methods.

6.3 Future work

The process of finding IRKS methods using the underlying one step approach has led to some good methods; using more sophisticated searching routines may led to better methods of low orders and also make it possible to find methods of high order. The use of backward error analysis may lead to another approach of determining good methods. This approach is based on the assumption that only constant stepsizes are used. Whether or not the methods developed using a backward error analysis argument would be efficient for variable stepsizes would be interesting. The other main extension that may yield interesting methods is when the matrix A can have distinct eigenvalues, rather than the one point spectrum discussed in this thesis. Allowing the IRKS methods to have this more general structure will make it possible to derive methods where the first row is zero as was discussed for example in [40]. Another possibility is when the the matrix A is diagonal, parallel methods may exist with order greater than two.

As the numerical experiments performed in this thesis are of a preliminary nature there are several issues with regards to implementation that could be improved. For example the stepsize changing procedure uses the traditional dead-beat or integral controller rather than the more effective proportional integral controller which is used so successfully for Runge-Kutta methods. The emphasis of the numerical experiments in this thesis have been on the explicit methods; the thesis by Huang [83] addresses the implementation of the stiff methods thoroughly. It would be nice to combine the approaches of the non-stiff and stiff methods so that a possible production code could have available a stiffness switching option. Since the explicit and implicit methods are so similar it may be quite convenient to switch from explicit to implicit methods, or vice-versa, reasonably easily. As the needs of the users of numerical methods change it is important to provide features that are required by the user. For example, it is now commonplace that a user does not just want a highly accurate approximation of the solution at one point but rather continuous solutions over the whole integration interval, and even the possibility of event location. Certain events which are interesting are local minima or maxima and it also could be possible that the final solution is wanted when one of these events occur, rather than at a pre-decided final point.

Even though the methods constructed in this thesis provide many advantages over the traditional methods there is scope for extending the methods to the case where $s = p+2$. This will provide significantly more freedom and it would be interesting to see if there are relative advantages in using more stages. The resulting matrices of the IRKS methods where $s = p+2$ are however not square which makes computing the methods in a straight forward manner much more difficult than in the $s = p+1$ case. Some preliminary work has already been done on these methods and it is possible to obtain a very large class of methods, but it is not yet known to what extent this comprises the set of all methods. One likely advantage of methods where $s = p+2$ will be in the solution of oscillatory problems. For these problems it is possible to derive methods which have a phase order and amplitude order which is greater than the order of the method.

Systems which require information not only from the current state, but states in the past are often best modelled using delay differential equations. Most numerical solvers for delay differential equations are based on Runge-Kutta methods. For problems of this type it is essential for the numerical solution to be piecewise continuous throughout the interval of integration rather than solutions given only at certain discrete points. This is due to the need to evaluate the function at some point in the past which generally would not have been a grid point in the integration. To obtain continuous output Runge-Kutta methods require many additional stages to those required for the traditional order. It is for this reason that the IRKS methods may in fact provide excellent alternatives to the Runge-Kutta methods, as piecewise continuous solutions can be found without any additional evaluations of the function. The main consideration which must be taken into account is the effect on the Nordsieck vector when a discontinuity is reached. Many preliminary numerical experiments have been performed against the recently released `dde23` function in Matlab. The results showed an increased performance over the standard ordinary differential equation case even when no considerations of the Nordsieck vector were taken into account.

Differential equations with algebraic constraints, commonly known as differential algebraic equations were briefly discussed in Chapter 1. These equations are a very important application area, as they are often used to model problems arising for example in mechanics and electrical networks. According to Brenan, Campbell and Petzold [8], stiffly accurate implicit Runge-Kutta methods perform very well when applied to general index one and semi-explicit index two and three differential algebraic equations systems, even when there are small errors in the initial conditions. The index is roughly the number of times the algebraic constraint needs to be differentiated in order to be transformed to a well defined differential system. The IRKS methods, which have strong stiff accuracy, should be well suited for solving differential algebraic equation systems because the solution component will satisfy the algebraic constraint and because the IRKS methods have high stage order they will not be affected by the order reduction phenomenon. The main consideration will be the effect the rest of the Nordsieck vector has on the computation.

APPENDIX I

Matlab and Maple codes

The codes in this appendix are included so that the interested reader can verify the methods constructed within the text. The programs given have been written in such a way that they beer as much resemblance as possible to the text.

The following Maple code is used to generate the starting procedure needed to construct the initial Nordsieck vector. Section 2.3 gives relevant details about this starting procedure. When the IRKS method is implicit the solution of the stage derivatives should be found using a variant of Newton's method. In the case of non-stiff problems it is sufficient to find the stage derivatives using a fixed point iteration scheme. The vector c used in the following program is generally chosen so that the first abscissae is zero and the last abscissae is one. This means that the initial condition is used.

```
# ==============================================================================
start:=proc(p,c)
local J,C,A,B;
# ==============================================================================
# p       - Defines the order.
# c       - List of the abscissae.
# ==============================================================================
# J       - Lower triangular shifter matrix.
# C       - Scaled Vandermonde matrix with powers of c.
# A       - Singly implicit starting matrix.
# B       - Appropriate linear combinations of stage derivatives.
# ==============================================================================
J:=band([1,0,0],p+1):
C:=matrix(p+1,p+1,(i,j)->if j=1 then 1 else c[i]^(j-1)/(j-1)! fi):
A:=evalm(C&*J&*inverse(C));
B:=evalm(J&*inverse(C));
# ==============================================================================
end:
# ==============================================================================
```

The following Maple code is used to construct an IRKS method. The code is based on approach one developed in Section 3.7. It is possible to change the program so as to use approach two if this is desired. The notation used with the program is as close as possible to that used in the text.

```
# ===========================================================================
IRKS:=proc(p,l,eps,c,beta,iT,iW)
local J,K,P,E,C,betaK,Km,eKm,F,T,psi,X,Omega,Gamma,OTU,Btil,B,V,A,U,M;
# ===========================================================================
# p       - Defines the order.
# l       - The eigenvalue of the matrix A.
# eps     - Related linearly to the error constant or L-stability condition.
# c       - List of the abscissae.
# beta    - List of free parameters in the doubly companion matrix.
# iT      - pxp free matrix used to ensure V has zero spectral radius.
# iW      - (p+1)x(p+1) free matrix used to ensure A has desired structure.
# ===========================================================================
# J       - Lower triangular shifter matrix.
# K       - Upper triangular shifter matrix.
# P       - Permutation matrix.
# E       - Toeplitz matrix E=exp(K) where K is nilpotent.
# C       - Scaled Vandermonde matrix with powers of c.
# betaK   - Toeplitz matrix with elements from beta on diagonals.
# Km      - Differentiation matrix.
# eKm     - Upper triangular transformation matrix eKm=exp(lKm).
# F       - Upper triangular Toeplitz matrix.
# T       - Used to ensure spectral radius of V is zero.
# ===========================================================================
J:=band([1,0,0],p+1):
K:=band([0,0,1],p+1):
P:=band([1,seq(0,i=1..p),1,seq(0,i=1..p-1)],p+1):
E:=exponential(K):
C:=matrix(p+1,p+1,(i,j)->if j=1 then 1 else c[i]^(j-1)/(j-1)! fi):
betaK:=transpose(Lower(toeplitz(beta))):
Km:=matrix(p+1,p+1,(i,j)->if j=i+1 then p+1-i else 0 fi):
eKm:=subs(t=l,exponential(Km,t)):
F:=evalm(inverse(eKm)&*E&*eKm):
T:=stackmatrix(augment([1],matrix(1,p,0)),augment(matrix(p,1,0),inverse(iT)));
# ===========================================================================
# psi     - Upper triangular transformation matrix which links J and X.
# X       - Doubly companion matrix.
# iW      - Removing # gives methods of ESIRK type.
# Omega   - Lumps conditions together.
# Gamma   - Toeplitz except first column satisfying IRKS L-stability etc.
# Btil    - The transformed matrix B.
# ===========================================================================
psi:=evalm(betaK&*eKm):
X:=map(x->simplify(x),evalm(psi&*(J+band([l],p+1))&*inverse(psi)));
#iW:=evalm(inverse(psi)&*E&*inverse(C)):
Omega:=evalm(iW&*C&*(Reverse(beta)&*transpose([1,seq(0,i=1..p)])+K&*psi)):
Gamma:=evalm(F+(J&*F-F&*J)&*P):
Gamma[1,1]:=subs(x=l,simplify(eps+x*(-x)^p*LaguerreL(p,1/x))):
OTU:=LUdecomp(evalm(Omega&*T),L='OTL'):
Btil:=evalm(T&*Lower(inverse(T)&*Gamma&*T&*inverse(OTU))&*inverse(OTL)):
```

```
# ==============================================================================
# The Nordsieck formulation of the method is found.
# ==============================================================================
B:=evalm(psi&*Btil&*iW):
V:=evalm(E-B&*C&*K):
A:=evalm(inverse(B)&*X&*B):
U:=evalm(C-A&*C&*K):
M:=augment(stackmatrix(A,B),stackmatrix(U,V)):
# ==============================================================================
end:
# ==============================================================================
```

The following Maple code is required by the previous procedure. This procedure commutes the lower triangular part of a matrix see Section 3.7 for details.

```
#=============================================================================
Lower:=proc(M)
local Y, N;
#=============================================================================
N:=evalm(M);
Y:=matrix(p+1,p+1,(i,j)->if j>i then 0 else N[i,j] fi):
RETURN(Y);
end:
#=============================================================================
```

As a means of finding optimal methods using some searching routine, it is necessary to construct the method using a language with searching facilities. Matlab has been chosen at this early stage because it is easy to use. The following program constructs IRKS methods numerically. Approach one is used from Section 3.7 as in the Maple program above.

```
%=============================================================================
function [A,U,B,V]=IRKSppss(p,lambda,eps,c,beta,iT,iW)
% ==============================================================================
% p       - Defines the order.
% lambda  - The eigenvalue of the matrix A.
% eps     - Related linearly to the error constant or L-stability condition.
% c       - Column vector of the abscissae.
% beta    - Row vector of free parameters in the doubly companion matrix.
% iT      - pxp free matrix used to ensure V has zero spectral radius.
% iW      - (p+1)x(p+1) free matrix used to ensure A has desired structure.
% ==============================================================================
% J       - Lower triangular shifter matrix.
% K       - Upper triangular shifter matrix.
% P       - Permutation matrix.
% C       - Vandermonde matrix with powers of c.
% E       - Toeplitz matrix E=exp(K) where K is nilpotent.
% betaK   - Toeplitz matrix with elements from beta on diagonals.
% Km      - Differentiation matrix.
% eKm     - Upper triangular transformation matrix eKm=exp(lKm).
% F       - Upper triangular Toeplitz matrix.
% T       - Used to ensure spectral radius of V is zero.
% ==============================================================================
J=diag(ones(p,1),-1);
K=diag(ones(p,1),1);
P=diag(ones(p,1),1)+diag(1,-p);
E=expm(K);
C=fliplr(vander(c))*diag(E(1,:));
betaK=triu(toeplitz(beta));
Km=diag([p:-1:1],1);
eKm=expm(lambda*Km);
F=inv(eKm)*E*eKm;
T=[1,zeros(1,p);zeros(p,1),inv(iT)];
% ==============================================================================
% psi     - Upper triangular transformation matrix which links J and X.
% X       - Doubly companion matrix.
% iW      - Transformation used to force method to have effective order.
% Omega   - Lumps conditions together.
% Gamma   - Toeplitz except first column satisfying IRKS L-stability etc.
% Btil    - The transformed matrix B.
% ==============================================================================
psi=betaK*eKm;
X=psi*(J+lambda*eye(p+1))*inv(psi);
%iW=inv(psi)*E*inv(C);
Omega=iW*C*(betaK*diag(1,-p)+K*psi);
Gamma=F+(J*F-F*J)*P;
Gamma(1,1)=eps+lambda*fliplr(1./[1,cumprod(-[1:p])])*eKm(1,:)';
[OTL,OTU]=feval('LU',Omega*T)
Btil=T*tril(inv(T)*Gamma*T*inv(OTU))*inv(OTL);
% ==============================================================================
% The Nordsieck formulation of the method is found.
% ==============================================================================
B=psi*Btil*iW; B(abs(B)<1e-10)=0;
V=E-B*C*K;     V(abs(V)<1e-10)=0;
A=inv(B)*X*B;  A(abs(A)<1e-10)=0;
U=C-A*C*K;     U(abs(U)<1e-10)=0;
% ==============================================================================
```

Since the LU factorisation program in Matlab `lu` uses pivoting. Due to the nature of the way methods are constructed see Section 3.7 it is necessary not use any pivoting. The following Matlab program decomposes a matrix into the product of a unit lower triangular and an upper triangular matrix.

```
% ==============================================================================
function [L,U]=LU(A)
% ==============================================================================
% L       - Lower triangular part of matrix A.
% U       - Upper triangular part of matrix A.
% ==============================================================================
[n,m]=size(A);
L=eye(n);
U=zeros(n,m);
U(1,:)=A(1,:);
L(2:n,1)=A(2:n,1);
for i=1:n-1
    U(i,i)=A(i,i)-L(i,1:i-1)*U(1:i-1,i);
    for j=i+1:n
        U(i,j)=1/L(i,i)*(A(i,j)-L(i,1:i-1)*U(1:i-1,j));
        L(j,i)=1/U(i,i)*(A(j,i)-L(j,1:i-1)*U(1:i-1,i));
    end
end
U(n,n)=A(n,n)-L(n,1:n-1)*U(1:n-1,n);
% ==============================================================================
```

The following code computes the local errors of the underlying one step method see Section 2.6 for more details. This function is based on Algorithm 2.8 and is used in conjunction with the Matlab program which generates the IRKS methods to search for methods where the local errors of the underlying one step method are minimised. The IRKS methods generated and given in Appendix III used only the basic search routine from Matlab `fminsearch`. Using more sophisticated routines may well lead to more optimal methods.

```
% ==============================================================================
function le=UOSM(pmax,p,r,s,A,U,B,V)
% ==============================================================================
% pmax    - The order the local errors are calculated to.
% p       - The order of the method.
% r       - Number of output approximations of the method.
% s       - Number of stages of the method.
% A,U,B,V - Coefficient matrices of the method.
% le      - Local error coefficients of the underlying one step method.
% ==============================================================================
% t       - Trees referenced by integers.
% first   - Index of the first tree of order p.
% rho     - Order of the tree.
% dec1    - Standard decomposition of t.
% dec2    - Standard decomposition of t.
% xi      - Elementary weight function of the stages.
% igamma  - Inverse of gamma.
% igammaD - Inverse of the derivative of gamma.
% igammaD - Inverse of the derivative of gamma.
% phi     - Elementary weight function of the underlying one step method.
% S       - Elementary weight function of the starting procedure.
% xi      - Elementary weight function of the stages.
% xiD     - Elementary weight function of the stage derivatives.
% R       - Elementary weight function of the remainder.
% v       - Left eigenvector of V.
% Q       - Matrix (I-Vtil)^{-1}.
% ========================================================================
t=1;
first(1)=1;
rho(1)=1;
dec1(1)=1;
dec2(1)=0;
igamma(1)=1;
igammaD(1)=1;
phi(1)=1;
S(:,1)=[0;(V(2:r,2:r)-eye(r-1))\(-B(2:r,:)*ones(s,1))];
xiD(:,1)=ones(s,1);
xi(:,1)=A*ones(s,1)+U*S(:,1);
R(:,1)=zeros(r,1);
le(1)=0;
s=struct('disp',0,'tol',1e-20);
[evect,eval]=eigs(V',1,s);
v=evect'./evect(1);
Q=inv(eye(r)-V+eye(r,1)*v);
for p=2:pmax
    first(p)=t+1;
    for q=1:p-1
        for t1=first(q):first(q+1)-1
            l=max(first(p-q),dec2(t1));
            for t2=l:first(p-q+1)-1
                t=t+1;
                rho(t)=p;
                dec1(t)=t1;
                dec2(t)=t2;
                igammaD(t)=igammaD(dec1(t))*igamma(dec2(t));
                igamma(t)=igammaD(t)/rho(t);
                xiD(:,t)=xiD(:,dec1(t)).*xi(:,dec2(t));
                R(:,t)=Rt(t,phi,S,p);
                phi(t)=v*(B*xiD(:,t)-R(:,t));
                S(:,t)=Q*(B*xiD(:,t)-R(:,t));
                S(1,t)=0;
                xi(:,t)=A*xiD(:,t)+U*S(:,t);
                le(t)=phi(t)-igamma(t);
            end
        end
    end
end
% ========================================================================
```

The following program calculates the residual $R(t)$ of a composition of two B-series defined by (2.5.6). As discussed in Section 2.5 it is possible to use Theorem 2.13 to write a program which computes $R(t)$ recursively. The following program gives the expressions explicitly for all trees up to order five. It is possible to use the recursive program to generate these expressions for orders greater than five. The advantage of using explicit representations of $R(t)$ is that the computational time is significantly reduced. This is especially significant as the order increases.

```
% ================================================================================
function R=Rt(t,c,d,p)
% ================================================================================
% t       - Trees referenced by integers.
% c       - Scalar elementary weight function (underlying one step method).
% d       - Vector elementary weight function (starting method).
% p       - The order of the method.
% ================================================================================
if t==1
   R=zeros(p+1,1);
elseif t==2
   R=c(1)*d(:,1);
elseif t==3
   R=c(1)*d(:,2)+c(2)*d(:,1);
elseif t==4
   R=2*c(1)*d(:,2)+c(1)^2*d(:,1);
elseif t==5
   R=c(1)*d(:,3)+c(2)*d(:,2)+c(3)*d(:,1);
elseif t==6
   R=2*c(1)*d(:,3)+c(1)^2*d(:,2)+c(4)*d(:,1);
elseif t==7
   R=c(1)*d(:,3)+c(1)*d(:,4)+(c(1)^2+c(2))*d(:,2)+c(1)*c(2)*d(:,1);
elseif t==8
   R=3*c(1)*d(:,4)+3*c(1)^2*d(:,2)+c(1)^3*d(:,1);
elseif t==9
   R=c(1)*d(:,5)+c(2)*d(:,3)+c(3)*d(:,2)+c(5)*d(:,1);
elseif t==10
   R=2*c(1)*d(:,5)+c(1)^2*d(:,3)+c(4)*d(:,2)+c(6)*d(:,1);
elseif t==11
   R=c(1)*d(:,5)+c(1)*d(:,6)+(c(1)^2+c(2))*d(:,3)+c(1)*c(2)*d(:,2)+c(7)*d(:,1);
elseif t==12
   R=3*c(1)*d(:,6)+3*c(1)^2*d(:,3)+c(1)^3*d(:,2)+c(8)*d(:,1);
elseif t==13
   R=c(1)*d(:,5)+c(1)*d(:,7)+c(1)^2*d(:,3)+c(2)*d(:,4)+...
     (c(1)*c(2)+c(3))*d(:,2)+c(1)*c(3)*d(:,1);
elseif t==14
   R=c(1)*d(:,6)+2*c(1)*d(:,7)+2*c(1)^2*d(:,3)+c(1)^2*d(:,4)+...
     (c(1)^3+c(4))*d(:,2)+c(1)*c(4)*d(:,1);
elseif t==15
   R=2*c(1)*d(:,7)+2*c(2)*d(:,3)+c(1)^2*d(:,4)+2*c(1)*c(2)*d(:,2)+...
     c(2)^2*d(:,1);
elseif t==16
   R=2*c(1)*d(:,7)+c(1)*d(:,8)+c(1)^2*d(:,3)+(2*c(1)^2+c(2))*d(:,4)+...
     (2*c(1)*c(2)+c(1)^3)*d(:,2)+c(1)^2*c(2)*d(:,1);
elseif t==17
   R=4*c(1)*d(:,8)+6*c(1)^2*d(:,4)+4*c(1)^3*d(:,2)+c(1)^4*d(:,1);
% ================================================================================
```

APPENDIX II

Test problems

The following differential equations are from the DETEST set [84]. These are included for the convenience of unfamiliar readers. The differential equations with initial conditions are given along with the analytical solution where possible. There are four groups of problems with five differential equations or systems in each group.

Group A: Single equations

A1: The negative exponential.

$$y' = -y, \qquad y(0) = 1, \qquad y = \exp(-x).$$

A2: Special case of the Riccati equation.

$$y' = -\frac{y^3}{2}, \qquad y(0) = 1, \qquad y = \frac{1}{\sqrt{x+1}}.$$

A3: Oscillatory problem.

$$y' = y\cos(x), \qquad y(0) = 1, \qquad y = \exp(\sin(x)).$$

A4: Logistic curve.

$$y' = \frac{y}{4}\left(1 - \frac{y}{20}\right), \qquad y(0) = 1, \qquad y = \frac{20}{1 + 19\exp(-x^4)}.$$

A5: Spiral curve (solution in polar coordinates).

$$y' = \frac{y-x}{y+x}, \qquad y(0) = 4, \qquad r = 4\exp\left(\tfrac{\pi}{2} - \theta\right).$$

Group B: Small systems.

B1: The growth of two conflicting populations.

$$\begin{cases} y_1' = 2(y_1 - y_1 y_2), & y_1(0) = 1. \\ y_2' = -2(y_2 - y_1 y_2), & y_2(0) = 3. \end{cases}$$

B2: Linear chemical reaction.

$$\begin{cases} y_1' = -y_1 + y_2, & y_1(0) = 2. \\ y_2' = y_1 - 2y_2 + y_3, & y_2(0) = 0. \\ y_3' = y_2 - y_3, & y_3(0) = 1. \end{cases}$$

B3: Nonlinear chemical reaction.

$$\begin{cases} y_1' = -y_1, & y_1(0) = 1. \\ y_2' = y_1 - y_2^2, & y_2(0) = 0. \\ y_3' = \quad y_2^2, & y_3(0) = 0. \end{cases}$$

B4: The integral surface of a torus.

$$\begin{cases} y_1' = -y_2 - \dfrac{y_1 y_3}{\sqrt{y_1^2 + y_2^2}}, & y_1(0) = 3. \\ y_2' = \quad y_1 - \dfrac{y_2 y_3}{\sqrt{y_1^2 + y_2^2}}, & y_2(0) = 0. \\ y_3' = \quad \dfrac{y_1}{\sqrt{y_1^2 + y_2^2}}, & y_3(0) = 0. \end{cases}$$

B5: Euler equations of motion for a rigid body without external forces.

$$\begin{cases} y_1' = \quad y_2 y_3, & y_1(0) = 0. \\ y_2' = -y_1 y_3, & y_2(0) = 1. \\ y_3' = -0.51 y_1 y_2, & y_3(0) = 1. \end{cases}$$

Group C: Moderate systems.

C1: Radioactive decay chain.

$$\begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ \vdots \\ y_{10}' \end{bmatrix} = \begin{bmatrix} -1 & 0 & \cdots & \cdots & 0 \\ 1 & -1 & \ddots & & \vdots \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{10} \end{bmatrix}, \qquad y(0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}.$$

C2: Radioactive decay chain.

$$\begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ \vdots \\ y_{10}' \end{bmatrix} = \begin{bmatrix} -1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & \ddots & & \vdots \\ 0 & 2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -9 & 0 \\ 0 & \cdots & 0 & 9 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{10} \end{bmatrix}, \qquad y(0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}.$$

C3: Parabolic partial differential equation.

$$\begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ \vdots \\ y_{10}' \end{bmatrix} = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{10} \end{bmatrix}, \qquad y(0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}.$$

C4: Parabolic partial differential equation as in C3 with 51 equations.

C5: Five body problem: the motion of the five outer planets about the sun. The three spatial coordinates of body j are y_{1j}, y_{2j}, y_{3j} and $j = 1, 2, \ldots, 5$. Each satisfy the second order differential equation

$$y''_{ij} = k_2 \left(-(m_0 + m_j) \frac{y_{ij}}{r_j^3} + \sum_{\substack{k=1 \\ k \neq j}}^{5} m_k \left(\frac{y_{ik} - y_{ij}}{d_{jk}^3} - \frac{y_{ik}}{r_k^3} \right) \right),$$

where r_k^3 and d_{jk}^3 are

$$r_k^3 = \sum_{i=1}^{3} y_{ik}^2, \qquad d_{jk}^3 = \sum_{i=1}^{3} (y_{ij} - y_{ik})^2, \qquad k, j = 1, 2, \ldots, 5,$$

and the physical constants are

$$\begin{aligned} k_2 &= 2.95912208286, && \text{(gravitational constant)} \\ m_0 &= 1.00000597682, && \text{(mass of sun and four inner planets)} \\ m_1 &= 0.00095478610, && \text{(Jupiter)} \\ m_2 &= 0.00028558373, && \text{(Saturn)} \\ m_3 &= 0.00004372731, && \text{(Uranus)} \\ m_4 &= 0.00005177591, && \text{(Neptune)} \\ m_5 &= 0.00000277777, && \text{(Pluto).} \end{aligned}$$

The system is rewritten as a system of 30 first order differential equations with initial values

$$\begin{aligned}
y_{11} &= 3.4294741518, & y_{21} &= 3.3538695971, & y_{31} &= 1.3549401715, \\
y'_{11} &= -0.5571605704, & y'_{21} &= 0.5056967832, & y'_{31} &= 0.2305785439, \\[1ex]
y_{12} &= 6.6414554255, & y_{22} &= 5.9715695787, & y_{32} &= 2.1823149972, \\
y'_{12} &= -0.4155707763, & y'_{22} &= 0.3656827228, & y'_{32} &= 0.1691432132, \\[1ex]
y_{13} &= 11.2630437207, & y_{23} &= 14.6952576794, & y_{33} &= 6.2796052506, \\
y'_{13} &= -0.3253256691, & y'_{23} &= 0.1897060219, & y'_{33} &= 0.0877265322, \\[1ex]
y_{14} &= -30.1552268759, & y_{24} &= 1.6569996640, & y_{34} &= 1.4378575272, \\
y'_{14} &= -0.0240476254, & y'_{24} &= -0.2876595326, & y'_{34} &= -0.1172195431, \\[1ex]
y_{15} &= 21.1238353380, & y_{25} &= 28.4465098142, & y_{35} &= 15.3882659679, \\
y'_{15} &= -0.1768607531, & y'_{25} &= -0.2163934530, & y'_{35} &= -0.0148647893.
\end{aligned}$$

Group D: Orbital equations.

$$\begin{aligned}
y'_1 &= y_3, & y_1(0) &= 1 - \varepsilon, \\
y'_2 &= y_4, & y_2(0) &= 0, \\
y'_3 &= -\frac{y_1}{(y_1^2 + y_2^2)^{\frac{3}{2}}}, & y_3(0) &= 0, \\
y'_4 &= -\frac{y_2}{(y_1^2 + y_2^2)^{\frac{3}{2}}}, & y_4(0) &= \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}.
\end{aligned}$$

D1: Low eccentricity $\varepsilon = 0.1$.

D2: Small eccentricity $\varepsilon = 0.3$.

D3: Medium eccentricity $\varepsilon = 0.5$.

D4: Large eccentricity $\varepsilon = 0.7$.

D5: High eccentricity $\varepsilon = 0.9$.

All are derived from the orbit equations

$$x'' = -\frac{x}{(x^2+y^2)^3}, \qquad x(0) = 1-\varepsilon, \qquad x'(0) = 0,$$
$$y'' = -\frac{y}{(x^2+y^2)^3}, \qquad y(0) = 0, \qquad y'(0) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}},$$

with solution

$$x = \cos(u) - \varepsilon, \qquad x' = \frac{-\sin(u)}{1-\varepsilon\cos(u)},$$
$$y = \sqrt{1-\varepsilon^2}\sin(u), \qquad y' = \frac{\sqrt{1-\varepsilon^2}\cos(u)}{1-\varepsilon\cos(u)},$$

where $u - \epsilon\sin(u) - t = 0$.

Group E: Higher order equations.

E1: From Bessel's equation of order 1/2.

$$y_1' = y_2, \qquad y_1(0) = J_{\frac{1}{2}}(1),$$
$$y_2' = y_1\left(\frac{1}{4(x+1)^2} - 1\right) - \frac{y_2}{x+1}, \qquad y_2(0) = J'_{\frac{1}{2}}(1).$$

E2: From Van der Pol's equation.

$$y_1' = y_2, \qquad y_1(0) = 2,$$
$$y_2' = (1-y_1^2)y_2 - y_1, \qquad y_2(0) = 0.$$

E3: From Duffing's equation.

$$y_1' = y_2, \qquad y_1(0) = 0,$$
$$y_2' = \frac{y_1^3}{6} - y_1 + 2\sin(2.78533x), \qquad y_2(0) = 0.$$

E4: From the falling body equation.

$$y_1' = y_2, \qquad y_1(0) = 30,$$
$$y_2' = 0.032 - 0.4y_2^2, \qquad y_2(0) = 0.$$

E5: From a linear pursuit equation.

$$y_1' = y_2, \qquad y_1(0) = 0,$$
$$y_2' = \frac{\sqrt{1+y_2^2}}{(25-x)}, \qquad y_2(0) = 0.$$

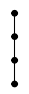
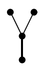
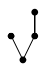
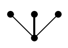
APPENDIX III

Methods

The following numerical methods are used in Chapter 5. Included in this appendix are three Runge-Kutta methods of orders three, four and five and a description of the Adams-Bashforth, Adams-Moulton methods. Also included are the coefficients of the IRKS methods of orders two, three, four and five used in all experiments. The first embedded Runge-Kutta method is of order three and was developed by Bogacki and Shampine [6], the method is given by the tableau

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{3}{4} & 0 & \frac{3}{4} & & \\
1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
\hline
 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & 0 \\
\hline
 & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8}
\end{array}$$

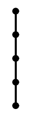
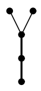
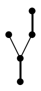
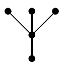
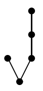
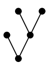
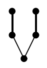
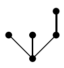
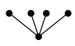
The embedded method has four stages but has the FSAL property so it effectively has only three stages. This method can be written as a general linear method with two output approximations. The first method is of order three but the second is of order two and is used only to provide an estimate of the local truncation error. This method forms the kernel of the code `ode23` from Matlab. The elementary weight functions and the density for the trees of order four are

t				
$\alpha(t)$	0	$\frac{1}{12}$	$\frac{1}{8}$	$\frac{11}{48}$
$\frac{1}{\gamma(t)}$	$\frac{1}{24}$	$\frac{1}{12}$	$\frac{1}{8}$	$\frac{1}{4}$

The error coefficients of the elementary differentials corresponding to the middle two trees are zero. Therefore contributions to the local truncation error only come from the tall and the bushy tree. The second embedded Runge-Kutta method is of order four and was developed by Nørsett and given in [66], the method is given by the tableau

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{3}{8} & \frac{3}{8} & & & & \\
\frac{9}{16} & 0 & \frac{9}{16} & & & \\
\frac{25}{32} & -\frac{125}{672} & \frac{325}{336} & 0 & & \\
1 & \frac{371}{891} & -\frac{200}{297} & \frac{1120}{891} & 0 & \\
\hline
 & \frac{25}{216} & \frac{32}{135} & \frac{256}{567} & 0 & \frac{11}{70} \\
\hline
 & \frac{37}{225} & \frac{44}{117} & 0 & \frac{448}{975} & 0
\end{array}$$

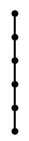
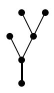
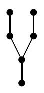
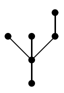
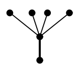
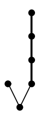
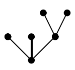
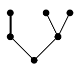
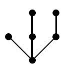
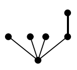
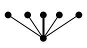
The embedded method has five stages. Since $b_2 = 0$ and $a_{43} = 0$ when the method is used with a fixed stepsize then there is no need for the second method and the first method can be represented with one less stage. This method is supposed to have good tolerance proportionality [111]. The elementary weight functions and the density for the trees of order five are

t									
$\alpha(t)$	0	$\frac{1}{64}$	$\frac{3}{128}$	$\frac{11}{256}$	$\frac{1}{24}$	$\frac{13}{192}$	$\frac{37}{704}$	$\frac{13}{128}$	$\frac{53}{256}$
$\frac{1}{\gamma(t)}$	$\frac{1}{120}$	$\frac{1}{60}$	$\frac{1}{40}$	$\frac{1}{20}$	$\frac{1}{30}$	$\frac{1}{15}$	$\frac{1}{20}$	$\frac{1}{10}$	$\frac{1}{5}$

None of the order five trees have an elementary weight the same as $\frac{1}{\gamma(t)}$, in saying that none of local errors associated with the trees of order five are that large. The last Runge-Kutta pair used for comparisons is the famous method of Dormand and Prince [63], called variously RK5(4)7FM, DOPRI5, DP(4,5) and DP54. This first method of this embedded pair is of order five, the second method which is used for error estimation purposes is of order four. The pair is represented by the tableau

$$\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{71}{57600} & 0 & -\frac{71}{16695} & \frac{71}{1920} & -\frac{17253}{339200} & \frac{22}{525} & -\frac{1}{40}
\end{array}$$

The number of stages is seven, one greater than required for order five, however this method also has the FSAL property and therefore effectively has only six stages. As for the first Runge-Kutta method, this method can be represented as a general linear method with two output approximations. This Runge-Kutta method is highly regarded as an optimal method, as the error terms of the higher order elementary differentials are minimised. This method forms the kernel of the code `ode45` from Matlab. The elementary weight functions and the density for the trees of order six which are not the same are

t											
$\alpha(t)$	$\frac{1}{600}$	$\frac{1}{16200}$	$\frac{89}{10621}$	$\frac{181}{10800}$	$\frac{181}{5400}$	$\frac{1}{150}$	$\frac{899}{32400}$	$\frac{899}{16200}$	$\frac{412}{9899}$	$\frac{899}{10800}$	$\frac{899}{5400}$
$\frac{1}{\gamma(t)}$	$\frac{1}{720}$	$\frac{1}{90}$	$\frac{1}{120}$	$\frac{1}{60}$	$\frac{1}{30}$	$\frac{1}{144}$	$\frac{1}{36}$	$\frac{1}{18}$	$\frac{1}{24}$	$\frac{1}{12}$	$\frac{1}{6}$

So it can be seen that there are only eleven trees above, therefore eight of the twenty trees satisfy the order conditions exactly. Only four trees differ from their order condition by more than $\frac{1}{10000}$, therefore this method is a highly optimal one.

The code `ode113` from Matlab is used to compare the variable stepsize, variable order implementation of the IRKS methods. The `ode113` code is a PECE implementation of Adams-Bashforth, Adams-Moulton methods of order one through twelve. Local extrapolation is used to increase accuracy of the solution. Since at this stage only IRKS methods of order up to five have been derived with minimised local error coefficients it is necessary to make the comparisons meaningful, the maximum order of the `ode113` code four (not five since local extrapolation is done).

The following IRKS methods of orders two through to five were found using the programs given in Appendix I. All methods are explicit that is $W = I$ and $\lambda = 0$ and have the strong strict stability properties described in Subsection 3.9.2. The abscissae vector c was chosen as

$$c = \begin{bmatrix} \frac{1}{p+1} & \frac{2}{p+1} & \cdots & \frac{p}{p+1} & 1 \end{bmatrix},$$

and $\epsilon = \frac{1}{(p+1)!}$. This means the error constant is zero and the method could be interpreted as one higher order, however as ϵ is a free parameter it will always be chosen to be very close to this value anyway. The coefficients $\beta_1, \beta_2, \ldots, \beta_p$ and the matrix T have been chosen to minimise the local error coefficients of the underlying one step method as described in Section 3.11. The IRKS methods given below are in traditional Nordsieck form that means quantity $i+1$ of the output approximation vector is $\frac{h^i}{i!} y_n^{(i)}$. This form makes magnitudes of the coefficients of the method closer.

The coefficients of the order two method which are not zero or one are

$$\begin{aligned}
A_{21} &= 0.471407662653622, & A_{31} &= -0.134639854910322, & A_{32} &= 0.804212616739007,\\
U_{12} &= 0.333333333333333, & U_{13} &= 0.111111111111111, & U_{22} &= 0.195259004013043,\\
U_{23} &= 0.130172669342029, & U_{32} &= 0.330427238171315, & U_{33} &= 0.017476414288205,\\
B_{11} &= -0.134639854910322, & B_{12} &= 0.804212616739007, & B_{31} &= 1.573932747069119,\\
B_{32} &= -1.471888137403692, & B_{33} &= 0.956614509246088, & V_{12} &= 0.330427238171315,\\
V_{13} &= 0.174764142882053, & V_{32} &= 1.058659118911515.
\end{aligned}$$

The coefficients of the order three method which are not zero or one are

$$\begin{aligned}
A_{21} &= 0.470846446710038, & A_{31} &= 0.261644146000365, & A_{32} &= 0.382303173507494,\\
A_{41} &= 0.802153740004660, & A_{42} &= -0.495396054365736, & A_{43} &= 0.721019675308796,\\
U_{12} &= 0.250000000000000, & U_{13} &= 0.062500000000000, & U_{14} &= 0.015625000000000,\\
U_{22} &= 0.029153553289961, & U_{23} &= 0.014576776644980, & U_{24} &= 0.036716291241867,\\
U_{32} &= 0.106052680492140, & U_{33} &= 0.049374753492322, & U_{34} &= 0.086089342494310,\\
U_{42} &= -0.027777360947721, & U_{43} &= 0.012789671400210, & U_{44} &= 0.004422512439833,\\
B_{11} &= 0.802153740004660, & B_{12} &= -0.495396054365736, & B_{13} &= 0.721019675308796,\\
B_{31} &= 1.446455234275734, & B_{32} &= -5.176148401701250, & B_{33} &= 2.583210132961478,\\
B_{34} &= 0.745416818553375, & B_{41} &= 3.640519215514662, & B_{42} &= -1.934848815659522,\\
B_{43} &= -1.681895828039472, & B_{44} &= 1.564670138888888, & V_{12} &= -0.027777360947721,\\
V_{13} &= 0.012789671400210, & V_{14} &= 0.004422512439833, & V_{32} &= 0.401066215910661,\\
V_{33} &= 0.087271948014413, & V_{34} &= 0.015483389816615, & V_{42} &= -1.588444710704555,\\
V_{43} &= -0.491907327816377, & V_{44} &= -0.087271948014413.
\end{aligned}$$

The coefficients of the order four method which are not zero or one are

$A_{21} = 0.2957311217040344$,	$A_{31} = -0.039934489325222$,	$A_{32} = 0.425901689047481$,
$A_{41} = 0.423835213302159$,	$A_{42} = -0.009436403626320$,	$A_{43} = 0.397304037858177$,
$A_{51} = -0.296305846538338$,	$A_{52} = 1.216363885601977$,	$A_{53} = -0.747215466333825$,
$A_{54} = 0.653600660017554$,	$U_{12} = 0.200000000000000$,	$U_{13} = 0.040000000000000$,
$U_{14} = 0.008000000000000$,	$U_{15} = 0.001600000000000$,	$U_{22} = 0.104268878295965$,
$U_{23} = 0.041707551318386$,	$U_{24} = 0.028512265395515$,	$U_{25} = 0.016136604105470$,
$U_{32} = 0.214032800277740$,	$U_{33} = 0.035252444492103$,	$U_{34} = 0.016359327976235$,
$U_{35} = 0.021847071262251$,	$U_{42} = -0.011702847534016$,	$U_{43} = 0.001250192150379$,
$U_{44} = 0.036580887257542$,	$U_{45} = 0.055182303793203$,	$U_{52} = 0.173556767252632$,
$U_{53} = -0.003671266293743$,	$U_{54} = 0.003781472902478$,	$U_{55} = 0.005112643571594$,
$B_{11} = -0.296305846538338$,	$B_{12} = 1.216363885601977$,	$B_{13} = -0.747215466333825$,
$B_{14} = 0.653600660017554$,	$B_{31} = 10.000016496975997$,	$B_{32} = -5.837474035968229$,
$B_{33} = -5.904712214292561$,	$B_{34} = 4.500681103257683$,	$B_{35} = 0.765845862194845$,
$B_{41} = 1.228682909898573$,	$B_{42} = 9.999954175745015$,	$B_{43} = -14.15295917400263$,
$B_{44} = 1.660756627244275$,	$B_{45} = 2.686339938592741$,	$B_{51} = -0.632617651189480$,
$B_{52} = 5.410153771892855$,	$B_{53} = -4.046215909571658$,	$B_{54} = -1.665749629987682$,
$B_{55} = 1.663894849785407$,	$V_{12} = 0.173556767252632$,	$V_{13} = -0.003671266293743$,
$V_{14} = 0.003781472902478$,	$V_{15} = 0.005112643571594$,	$V_{32} = -3.524357212167734$,
$V_{33} = 0.022845797533275$,	$V_{34} = 0.040229444224309$,	$V_{35} = -0.004714169797708$,
$V_{42} = -1.422774477477972$,	$V_{43} = 0.462224023471394$,	$V_{44} = 0.090103414290175$,
$V_{45} = -0.517738722736443$,	$V_{52} = -0.729465430929441$,	$V_{53} = 0.117792842856975$,
$V_{54} = 0.055508230191686$,	$V_{55} = -0.112949211823450$.	

The coefficients of the order five method which are not zero or one are

$A_{21} = 0.263372972232338$,	$A_{31} = -0.263053563397940$,	$A_{32} = 0.441750625998773$,
$A_{41} = -0.807101161142095$,	$A_{42} = 0.576841125717985$,	$A_{43} = 0.260736476477041$,
$A_{51} = -0.300357326545973$,	$A_{52} = -0.012056490728257$,	$A_{53} = 0.493481701798725$,
$A_{54} = 0.196335346754935$,	$A_{61} = 0.584293901912358$,	$A_{62} = -2.917005275237307$,
$A_{63} = 4.669954930744773$,	$A_{64} = -2.597765649735092$,	$A_{65} = 0.931826640861966$,
$U_{12} = 0.166666666666666$,	$U_{13} = 0.027777777777777$,	$U_{14} = 0.004629629629629$,
$U_{15} = 0.000771604938271$,	$U_{16} = 0.000128600823045$,	$U_{22} = 0.069960361100994$,
$U_{23} = 0.023320120366998$,	$U_{24} = 0.015089289351008$,	$U_{25} = 0.007468401748783$,
$U_{26} = 0.003099126907539$,	$U_{32} = 0.321302937399166$,	$U_{33} = 0.043184103800131$,
$U_{34} = -0.000329078383096$,	$U_{35} = 0.001926825100143$,	$U_{36} = 0.004996309982320$,
$U_{42} = 0.636190225613734$,	$U_{43} = 0.068180937869444$,	$U_{44} = -0.024278006205638$,
$U_{45} = -0.003349000792875$,	$U_{46} = 0.017713433211216$,	$U_{52} = 0.455930102053902$,
$U_{53} = 0.047339049639966$,	$U_{54} = -0.024139427530338$,	$U_{55} = 0.010166810706614$,
$U_{56} = 0.055656100424013$,	$U_{62} = 0.328695451453301$,	$U_{63} = -0.009406249680509$,
$U_{64} = -0.056440233621147$,	$U_{65} = 0.008178456392172$,	$U_{66} = 0.037260018923659$,

$$
\begin{array}{llllll}
B_{11} = & 0.584293901912358, & B_{12} = & -2.917005275237307, & B_{13} = & 4.669954930744773, \\
B_{14} = & -2.597765649735092, & B_{15} = & 0.931826640861966, & B_{31} = & 15.915061239241016, \\
B_{32} = & -14.176319893310838, & B_{33} = & 11.538908201124006, & B_{34} = & -12.148596781040178, \\
B_{35} = & 3.452169061907394, & B_{36} = & 2.148869773618636, & B_{41} = & 17.683612455307627, \\
B_{42} = & -9.137326426005964, & B_{43} = & 12.896934882903027, & B_{44} = & -16.658023145198315, \\
B_{45} = & 0.659357867701929, & B_{46} = & 4.313338591509030, & B_{51} = & -8.513792491139199, \\
B_{52} = & 38.844682504222445, & B_{53} = & -38.279871145369852, & B_{54} = & 18.712314794358732, \\
B_{55} = & -11.425398233410481, & B_{56} = & 4.811300746781537, & B_{61} = & -10.957591802735312, \\
B_{62} = & 27.674473356035389, & B_{63} = & -28.855294895329248, & B_{64} = & 16.511267662248411, \\
B_{65} = & -6.741133239149936, & B_{66} = & 1.683285163776493, & V_{12} = & 0.328695451453301, \\
V_{13} = & -0.009406249680509, & V_{14} = & -0.056440233621147, & V_{15} = & 0.008178456392172, \\
V_{16} = & 0.037260018923659, & V_{32} = & -6.730091601540036, & V_{33} = & -0.246274294359811, \\
V_{34} = & 0.304504551881114, & V_{35} = & -0.152256173640385, & V_{36} = & 0.137941274718748, \\
V_{42} = & -9.757894226217335, & V_{43} = & -0.214830852925117, & V_{44} = & 0.796459469907389, \\
V_{45} = & -0.459067237437802, & V_{46} = & -0.238694721809408, & V_{52} = & -4.149236175443180, \\
V_{53} = & -0.308677190755958, & V_{54} = & 0.890416251997090, & V_{55} = & -0.432249547872500, \\
V_{56} = & -0.390648241144361, & V_{62} = & 0.684993755154203, & V_{63} = & -0.088195219750214, \\
V_{64} = & 0.308927910281421, & V_{65} = & -0.166931550695128, & V_{66} = & -0.117935627675077.
\end{array}
$$

References

[1] Abramowitz, M. & Stegun, I. A. *Handbook of Mathematical Functions,* Dover Publications, (1964).

[2] Albrecht, P. *Numerical treatment of ODEs, the theory of A-methods,* Numer. Math. **33**, (1985), 59-87.

[3] Alexander, R. *Diagonally implicit Runge-Kutta methods for stiff ODEs,* SIAM J. Numer. Anal. **14**, (1977), 1006-1021.

[4] Bashforth, F. & Adams, J. C. *An attempt to test the theories of capillary action by comparing the theoritical and measured forms of drops of fluid, with an explanation of method of integration employed in constructing the tables which give the theoretical forms of such drops,* Cambridge University, (1883).

[5] Beaudet, P. R. *Multi-off-grid methods in multi-step integration of ordinary differential equations,* Lecture Notes in Math. **362**, (1972), 128-148.

[6] Bogacki, P. & Shampine, L. F. *A 3(2) pair of Runge-Kutta formulas,* Appl. Math. Lett. **2**, (1989), 1-9.

[7] Brankin, R. W., Gladwell, I. & Shampine, L. F. *RKSUITE: a suite of Runge-Kutta codes for the initial value problem for ODEs,* Report 92, Southern Methodist University, (1991).

[8] Brenan, K. E., Campbell, S. L. & Petzold, L. R. *Numerical Solution of Intial Value Problems in Differential Algebraic Equations,* North Holland, (1989).

[9] Brown, P. N., Byrne, G. D. & Hindmarsh, A. C. *VODE: a variable-coefficient ODE solver,* SIAM J. Sci. Comput. **10**, (1989), 1038-1051.

[10] Brush, D. G., Kohfeld, J. J. & Thompson, G. T. *Solution of ordinary differential equations using two 'off-step' points,* J. Assoc. Comput. Mach. **14**, (1967), 769-784.

[11] Burrage, K. *A special family of Runge-Kutta methods for solving stiff differential equations,* BIT **18**, (1978), 43-69.

[12] Burrage, K. *Order properties of implicit multivalue methods,* IMA J. Numer. Anal. **8**, (1988), 385-400.

[13] Burrage, K. & Butcher, J. C. *Non-linear stability for a general class of differential equation methods,* BIT **20**, (1980), 185-203.

[14] Burrage, K., Butcher, J. C. & Chipman, F. H. *An implementation of singly-implicit Runge-Kutta methods,* BIT **20**, (1980), 326-340.

[15] Burrage, K. & Moss, P. *Simplifying assumptions for the order of partitioned multivalue methods,* BIT **20**, (1980), 452-465.

[16] Butcher, J. C. *Coefficients for the study of Runge-Kutta integration processes,* J. Aust. Math. Soc. **3**, (1963), 185-201.

[17] Butcher, J. C. *Implicit Runge-Kutta processes,* Math. Comp. **18**, (1964), 50-64.

[18] Butcher, J. C. *A modified multistep method for the numerical integration of ordinary differential equations,* J. Assoc. Comput. Mach. **12**, (1965), 124-135.

[19] Butcher, J. C. *On the convergence of numerical solutions to ordinary differential equations,* Math. Comp. **20**, (1966), 1-10.

[20] Butcher, J. C. *A multistep generalization of Runge-Kutta methods with four or five stages,* J. Assoc. Comput. Mach. **14**, (1967), 84-99.

[21] Butcher, J. C. *The effective order of Runge-Kutta methods,* Lecture Notes in Math. **109**, (1969), 133-139.

[22] Butcher, J. C. *An algebraic theory of integration methods,* Math. Comp. **26**, (1972), 79-106.

[23] Butcher, J. C. *On the implementation of implicit Runge-Kutta methods,* BIT **16**, (1976), 237-240.

[24] Butcher, J. C. *A transformed implicit Runge-Kutta method,* J. Assoc. Comput. Mach. **26**, (1979), 731-738.

[25] Butcher, J. C. *Linear and non-linear stability for general linear methods,* Report 28, The University of Auckland, (1982).

[26] Butcher, J. C. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods,* John Wiley & Sons, (1987).

[27] Butcher, J. C. *Towards efficient implementation of singly-implicit methods,* ACM Trans. Math. Software **14**, (1988), 68-75.

[28] Butcher, J. C. *Diagonally implicit multistage integration methods,* Appl. Numer. Math. **11**, (1993), 347-363.

[29] Butcher, J. C. *An introduction to DIMSIMs,* Mat. Apl. Comput. **14**, (1995), 59-72.

[30] Butcher, J. C. *An introduction to "Almost Runge-Kutta" methods,* Appl. Numer. Math. **24**, (1997), 331-342.

[31] Butcher, J. C. *ARK methods up to order 5,* Numer. Algorithms **17**, (1998), 193-221.

[32] Butcher, J. C. *General linear methods for stiff differential equations,* BIT **41**, (2001), 240-264.

[33] Butcher, J. C. *Numerical Methods for Ordinary Differential Equations,* John Wiley & Sons, (2003).

[34] Butcher, J. C. & Cash, J. R. *Towards efficient Runge-Kutta methods for stiff systems,* SIAM J. Numer. Anal. **27**, (1990), 753-761.

[35] Butcher, J. C. & Chartier, P. *A generalisation of singly implicit Runge-Kutta methods,* Appl. Numer. Math. **24**, (1997), 343-350.

[36] Butcher, J. C. & Chartier, P. *The effective order of singly-implicit Runge-Kutta methods,* Numer. Algorithms **20**, (1999), 269-284.

[37] Butcher, J. C., Chartier, P. & Jackiewicz, Z. *Nordsieck representation of DIMSIMs,* Numer. Algorithms **16**, (1997), 209-230.

[38] Butcher, J. C., Chartier, P. & Jackiewicz, Z. *Experiments with a variable-order type 1 DIMSIM code,* Numer. Algorithms **22**, (1999), 237-261.

[39] Butcher, J. C. & Chen, D. J. L *ESIRK methods and variable stepsize,* Appl. Numer. Math. **28**, (1998), 193-207.

[40] Butcher, J. C. & Chen, D. J. L *A new type of singly-implicit Runge-Kutta method,* Appl. Numer. Math. **34**, (2000), 179-188.

[41] Butcher, J. C. & Diamantakis, M. T. *DESIRE: diagonally extended singly implicit Runge-Kutta effective order methods,* Numer. Algorithms **17**, (1998), 121-145.

[42] Butcher, J. C. & Jackiewicz, Z. *Construction of diagonally implicit general linear methods of type 1 and 2 for ordinary differential equations,* Appl. Numer. Math. **21**, (1996), 385-415.

[43] Butcher, J. C. & Jackiewicz, Z. *Implementation of diagonally implicit multi-stage integration methods for ordinary differential equations,* SIAM J. Numer. Anal. **34**, (1997), 2119-2141.

[44] Butcher, J. C. & Jackiewicz, Z. *Construction of high order DIMSIMs for ordinary differential equations,* Appl. Numer. Math. **27**, (1998), 1-12.

[45] Butcher, J. C. & Jackiewicz, Z. *A new appoach to error estimation for general linear methods,* (to appear Numer. Math.).

[46] Butcher, J. C., Jackiewicz, Z. & Mittelmann, H. D. *A nonlinear optimization approach to the construction of general linear methods of high order,* J. Comput. Appl. Math. **81**, (1997), 181-196.

[47] Butcher, J. C. & Moir, N. *Experiments with a new fifth order method,* (to appear Numer. Algorithms).

[48] Butcher, J. C. & O'Sullivan, A. E. *Nordsieck methods with an off-step point,* Numer. Algorithms **31**, (2002), 87-101.

[49] Butcher, J. C. & Tracogna, S. *Order conditions for two-step Runge-Kutta methods,* Appl. Numer. Math. **24**, (1997), 351-364.

[50] Butcher, J. C. & Wright, W. M. *A transformation relating explicit and diagonally-implicit general linear methods,* Appl. Numer. Math. **44**, (2003), 313-327.

[51] Butcher, J. C. & Wright, W. M. *The construction of practical general linear methods,* (to appear BIT).

[52] Byrne, G. D. & Lambert, R. J. *Pseudo Runge-Kutta methods involving two points,* J. Assoc. Comput. Mach. **13**, (1966), 114-123.

[53] Cash, J. R. *A class of implicit Runge-Kutta methods for the numerical integration of stiff ordinary differential equations,* J. Assoc. Comput. Mach. **22**, (1975), 504-511.

[54] Cash, J. R. & Sighal, A. *High order methods for the numerical solution of two-point boundary value problems,* BIT **22**, (1982), 184-199.

[55] Chan, T. M. H. *Algebraic Structures for the Analysis of Numerical Methods,* PhD Thesis, The University of Auckland, (1998).

[56] Chartier, P. *The potential of parallel multi-value methods for the simulation of large real life problems,* CWI Quarterly **11**, (1998), 7-32.

[57] Chipman, F. H. *A-stable Runge-Kutta processes,* BIT **11**, (1971), 384-388.

[58] Curtiss, C. F. & Hirschfelder, J. O. *Integration of stiff equations,* Proc. Nat. Acad. Sci. **38**, (1952), 235-243.

[59] Dahlquist, G. *Convergence and stability in the numerical integration of ordinary differential equations,* Math. Scand. **4**, (1956), 33-53.

[60] Dahlquist, G. *A special stability property for linear multistep methods,* BIT **3**, (1963), 27-43.

[61] Dahlquist, G. *Error analysis for a class of methods for stiff non-linear initial value problems,* Lecture Notes in Math. **506**, (1976), 60-72.

[62] Diamantakis, M. *Implementation of the DESI Formulae,* PhD Thesis, London Imperial College, (1995).

[63] Dormand, J. R. & Prince, P. J. *A family of embedded Runge-Kutta formulae,* J. Comput. Appl. Math. **6**, (1980), 19-26.

[64] Ehle, B. L. *On Páde approximations to the exponential function and A-stable methods for the numerical solution of initial value problems,* Report 2010, University of Waterloo, (1969).

[65] Eirola, T. & Sanz-Serna, J. M. *Conservation of integrals and symplectic structure in the integration of differential equations by multistep methods,* Numer. Math. **61**, (1992), 281-290.

[66] Enright, W. H., Jackson, K. R., Nørsett, S. P. & Thomsen, P. G. *Interpolants for Runge-Kutta formulas,* ACM Trans. Math. Software **12**, (1986), 193-218.

[67] Franck, R., Schneid, J. & Überhuber, C. W. *Order results for implicit Runge-Kutta methods applied to stiff systems,* SIAM J. Numer. Anal. **22**, (1983), 497-514.

[68] Gear, C. W. *Hybrid methods for initial value problems in ordinary differential equations,* SIAM J. Numer. Anal. **2**, (1965), 69-86.

[69] Gear, C. W. *The numerical integration of ordinary differential equations,* Math. Comp. **21**, (1967), 146-156.

[70] Gear, C. W. *DIFSUB for solution of ordinary differential equations,* Comm. ACM **14**, (1971), 185-190.

[71] Gear, C. W. *Runge-Kutta starters for multistep methods,* ACM Trans. Math. Software **6**, (1980), 263-279.

[72] Gill, S. *A process for the step-by-step integration of differential equations in an automatic digital computing machine,* Proc. Cambridge Philos. Soc. **47**, (1951). 96-108.

[73] Gragg, W. B. & Stetter, H. J. *Generalized multistep predictor-corrector methods,* J. Assoc. Comput. Mach. **11**, (1964), 188-209.

[74] Guglielmi, N. & Zennaro, M. *On asymptotic properties of a family of matrices,* Linear Algebra Appl. **322**, (2001), 169-192.

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

[76] Gustafsson, K. & Söderlind, G. *Control strategies for the iterative solution of nonlinear equations in ODE solvers,* SIAM J. Sci. Comput. **18**, (1997), 23-40.

[77] Hairer, E. & Wanner, G. *On the Butcher group and general multi-value methods,* Computing (Arch. Elektron. Rechnen) **13**, (1974), 1-15.

[78] Hairer, E. & Wanner, G. *Solving Ordinary Differential Equations, II: Stiff and Differential Algebraic Problems,* Springer-Verlag, (1991).

[79] Hairer, E. & Wanner, G. *Order conditions for general two-step Runge-Kutta methods,* SIAM J. Numer. Anal. **34**, (1997), 2087-2089.

[80] Hairer, E. & Wanner, G. *Stiff differential equations solved by Radau methods,* ACM Trans. Math. Software **111**, (1999), 93-111.

[81] Heun, K. *Neue Methoden zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen,* Z. Math. Phys. **45**, (1900), 23-38.

[82] Hindmarsh, A. C. *LSODE and LSODI, two new initial value ordinary differential equation solves,* ACM SIGNUM Newsletter **15**, (1980), 10-11.

[83] Huang, S. *Solving Stiff Differential Systems with Inherent Runge-Kutta Stable Methods,* PhD Thesis, The University of Auckland, (2004).

[84] Hull, T. E., Enright, W. H., Fellen, B. M. & Sedgwick, A. E. *Comparing numerical methods for ordinary differential equations,* SIAM J. Numer. Anal. **9**, (1972), 603-637.

[85] Jackiewicz, Z., Renaut, R. & Feldstein, A. *Two-step Runge-Kutta methods,* SIAM J. Numer. Anal. **28**, (1991), 1165-1182.

[86] Jackiewicz, Z., Renaut, R. & Zennaro, M. *Explicit two-step Runge-Kutta methods,* Appl. Math. **40**, (1995), 433-456.

[87] Jackiewicz, Z. & Tracogna, S. *A general class of two-step Runge-Kutta methods for ordinary differential equations,* SIAM J. Numer. Anal. **32**, (1995), 1390-1427.

[88] Jackiewicz, Z. & Tracogna, S. *Variable stepsize continuous two-step Runge-Kutta methods for ordinary differential equations,* Numer. Algorithms **12**, (1996), 347-368.

[89] Jackiewicz, Z., Vermiglio, R. & Zennaro, M. *Variable stepsize diagonally implicit multistage integration methods for ordinary differential equations,* Appl. Numer. Math. **16**, (1995), 343-367.

[90] Jeltsch, R. & Nevanlinna, O. *Stability of explicit time discretizations for solving initial value problems,* Numer. Math. **37**, (1981), 61-91.

[91] Kirchgraber, U. *Multistep methods are essentially one step methods,* Numer. Math. **48**, (1986), 85-90.

[92] Kohfeld, J. J. & Thompson, G. T. *Multistep methods with modified predictors and correctors,* J. Assoc. Comput. Mach. **14**, (1967), 155-166.

[93] Kohfeld, J. J. & Thompson, G. T. *A modification of Nordsieck's method using an 'off-step' point,* J. Assoc. Comput. Mach. **15**, (1968), 390-401.

[94] Krogh, F. T. *A variable step variable order multistep method for the numerical solution of ordinary differential equations,* Information Processing **68**, (1969), 194-199.

[95] Kutta, W. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen,* Z. Math. Phys. **46**, (1901), 435-453.

[96] Merson, R. H. *An operational method for the study of integration processes,* Proc. Symp. Data Processing, (1957), 1-25.

[97] Milne, W. E. *Numerical integration of ordinary differential equations,* Amer. Math. Monthly **33**, (1926), 455-460.

[98] Milne, W. E. *A note on the numerical integration of differential equations,* J. Research Nat. Bur. Standards **43**, (1949), 537-542.

[99] Moulton, F. R. *New Methods in Exterior Ballistics,* University of Chicago, (1926).

[100] Muir, P. H. & Enright, W. H. *Relationships amoung some classes of implicit Runge-Kutta methods and their stability functions,* BIT **27**, (1987), 403-423.

[101] Murua, A. *Formal series and numerical integrators, Part I: Systems of ODEs and symplectic integrators,* Appl. Numer. Math. **29**, (1999), 221-251.

[102] Nordsieck, A. *On numerical integration of ordinary differential equations,* Math. Comp. **16**, (1962), 22-49.

[103] Nørsett, S. P. *Runge-Kutta methods with a multiple real eigenvalue only,* BIT **16**, (1976), 388-393.

[104] Nyström, E. J. *Über die numerische Integration von Differentialgleichungen,* Acta Soc. Sci. Fennicae **50**, (1925), 1-54.

[105] Prothero, A. & Robinson, A. *On the stability and accuracy of one step methods for solving stiff systems of ordinary differential equations,* Math. Comp. **28**, (1974), 145-162.

[106] Renaut, R. *Two-step Runge-Kutta methods and hyperbolic partial differential equations,* Math. Comp. **55**, (1990), 563-579.

[107] Runge, C. *Über die numerische Auflösung von Differentialgleichungen,* Math. Ann. **46**, (1895), 167-178.

[108] Sanz-Serna, J. M. *Runge-Kutta schemes for Hamiltonian systems,* BIT **28**, (1988), 877-883.

[109] Shampine, L. F. & Reichelt, M. W. *The Matlab ODE suite,* SIAM J. Sci. Comput. **18**, (1997), 1-22.

[110] Shampine, L. F. & Gordon, M. K. *Computer Solution of Ordinary Differential Equations: the Initial Value Problem,* W. H. Freeman, (1975).

[111] Sharp, P. W. *Numerical comparisons of some explicit Runge-Kutta pairs of orders 4 through 8,* ACM Trans. Math. Software **17**, (1991), 387-409.

[112] Stoffer, D. *General linear methods: connection to one step methods and invariant curves,* Numer. Math. **64**, (1993), 395-408.

[113] Tracogna, S. *Implementation of two-step Runge-Kutta methods for ordinary differential equations,* J. Comput. Appl. Math. **76**, (1996), 113-136.

[114] Welfert, B. D. (private communication).

[115] Wright, W. M. *Practical General Linear Methods,* MSc Thesis, The University of Auckland, (1998).

[116] Wright, W. M. *The construction of order 4 DIMSIMs for ordinary differential equations,* Numer. Algorithms **26**, (2001), 123-130.

[117] Wright, W. M. *Explicit general linear methods with inherent Runge-Kutta stability,* Numer. Algorithms **31**, (2002), 381-399.

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

[119] Verner, J. H. *Explicit Runge-Kutta methods with estimates of the local truncation error,* SIAM J. Numer. Anal. **15**, (1978), 772-790.