HIGH-ORDER CONTINUOUS RUNGE-KUTTA METHODS

by

Thomas Sebastian Baker, B.Sc., M.Sc.

A thesis submitted in partial satisfaction for the degree of

Doctor of Philosophy

The University of Teesside

May, 1997

To Kate — my true director of studies.

Abstract

HIGH-ORDER CONTINUOUS RUNGE-KUTTA METHODS

Thomas Sebastian Baker
The University of Teesside, 1997

Director of Studies: Dr John R. Dormand
Second Supervisor: Dr Peter J. Prince

This thesis is concerned with the derivation of continuous Runge-Kutta and Runge-Kutta-Nyström processes. Specifically, the case of providing interpolants for existing high-order pairs is given most attention via the solution of general systems of continuous order conditions. Partially automated schemes have been adopted for this task making much use of specially developed algorithms and items of mathematical software, the principal tool presented being a general purpose Fortran 90 package, RK-AID. Extensive symbolic linear algebra has been performed in MACSYMA, for which a general strategy and notation for solution of over-determined algebraic systems is given. New optimisation strategies motivated by global error considerations are derived to aid the choice of free parameters, based upon linearisation of continuous truncation error terms. A modified 'direct search' method for performing multi-dimensional parameter searches has been coded and used for this purpose. Continuous extensions are derived in the minimum number of additional stages and optimised for each of the Dormand and Prince RKN6(4)6FD, RKN6(4)6FM, RK8(7)13M, RK8(6)12M and RKN12(10)17M pairs, up to a maximum order $p^* = q$ in every case. Experimental sixth-order triples are also presented that explore specific properties of the Nyström case.

Acknowledgments

I would like to thank Doctors John Dormand and Peter Prince for their time and encouragement over four years. Also, I owe my gratitude to the University of Teesside for employment, facilities and funding. Finally, thanks to my family for their support and patience while waiting for me to enter society.

Contents

Chapter 1

Introduction

1.1 Historical notes

Numerical techniques for the integration of differential quadrature problems of the form

$$y' = f(x),\ x \in [x_0, x_{end}],$$

have been known since the time of Newton. Approximate solution of general systems of initial value problems

$$\boldsymbol{y}'(x) = \boldsymbol{f}(x, \boldsymbol{y}(x)),\ \boldsymbol{y}(x_0) = \boldsymbol{y}_0,\ x \in [x_0, x_{end}],$$

where the function derivative is specified in terms of the independent variable also, is more recent — the first method being that of Euler [1] in 1768. Several centuries later, Euler's method was used to derive more accurate schemes, based upon classical quadrature rules, by Runge [2] and Heun [3] in the years 1895 and 1900, respectively. These methods belonged to a general class first characterised by Kutta [4] in 1901, and are now known as 'Runge-Kutta' processes.

These methods had been considered mostly as an abstraction until the theoretical developments of Butcher in the years 1963–1969 made their derivation more accessible. Popularity with practitioners increased with the availability of efficient and reliable methods of optimising the integration step size. Such a method is that of *embedding*, whereby a second Runge-Kutta process is formed using common function evaluations to the principal one enabling estimation of the local error introduced in each step.

In 1925, Nyström [5] produced direct methods for solving the second-order ordinary differential equation problem particular to its lack of a first derivative:

$$\boldsymbol{y}''(x) = \boldsymbol{f}(x, \boldsymbol{y}(x)),\ \boldsymbol{y}(x_0) = \boldsymbol{y}_0,\ \boldsymbol{y}'(x_0) = \boldsymbol{y}'_0,\ x \in [x_0, x_{end}].$$

Such methods are closely related to the Runge-Kutta formulation, and are usually known as Runge-Kutta-Nyström or Nyström methods. The technique of embedding, being immediately applicable to the Runge-Kutta-Nyström domain, enabled derivation of efficient variable step processes.

1.2 Continuous Runge-Kutta methods

A new generation of research in the 1980's and 1990's has been the addition of a continuous approximation to the initial value problem, including the possibility of derivative continuity and a global accuracy to match that of the discrete solution. An embedded pair equipped with such a 'dense output' facility is termed a *triple*, again utilising the same function evaluations as the main integrator for efficiency.

This thesis is concerned with the derivation of Runge-Kutta and Runge-Kutta-Nyström triples. Specifically, the case of providing an interpolant for existing high-order pairs that previously lacked such a facility is given most attention. General schemes have been developed for this task making much use of specially developed algorithms and items of mathematical software.

Basic concepts, notation and definitions are developed in Chapter 2, motivating the use of Runge-Kutta(-Nyström) processes and their derivation as efficient numerical integrators. Chapter 3 describes the principal tool for automation of common stages in the development of continuous formulae, called RK-AID (read *arcade*). The main function of the software presented is to aid the solution of order conditions and present them in a form convenient for porting into a computer algebra package (MACSYMA) for completion of the model. Before new processes are considered, a review of development strategies and existing formulae is contained in Chapter 4.

The new work on high-order interpolants has required a consistent general strategy and notation for solution of the over-determined algebraic linear systems involved in their derivation — this is contained in Chapter 5. Specification of any degrees of freedom for continuous processes has previously been done using minimal local error criteria, included here is a new requirement based on global error considerations. A modified 'direct search' method for performing multi-dimensional parameter searches is presented for this purpose.

The remaining chapters contain new RK(N) triples beginning with optimised, C^2 continuous, locally sixth-order Nyström methods in Chapter 6 — also containing new complete Nyström formulae with specific properties relating to their dense output ability. The Runge-Kutta approach to general first-order initial value problems is considered in Chapter 7 with C^1, eighth- and seventh-order interpolants for the Dormand and Prince discrete integrators RK8(7)13M and RK8(6)12M. Chapter 8 concludes the new work with C^2, tenth- to twelfth-order dense output for the RKN12(10)17M — a Dormand and Prince Nyström pair. Chapter 9 contains a concluding review with suggestions for future work.

A reference manual for the RK-AID package is included here as Appendix A, describing the user interface, arithmetic routines, and containing a complete glossary of commands. Also, a selection of new triples is included via RK-AID readable files of method coefficients in Appendix B, followed by a listing of the Fortran 90 optimisation code (presented in Chapter 5), as Appendix C.

Chapter 2

RK-OLOGY: Runge-Kutta and Runge-Kutta-Nyström theory

2.1 Introduction

Scientists and engineers frequently use mathematical models to describe the behaviour of objects or quantities under the action of external forces such as gravity, electromagnetism, pressure or chemical reaction. These models typically involve the rates of change of these values in terms of their quantity, position or other parameterisable measures, and are therefore termed *differential equations*. If this system of equations has only one independent variable, it is known as an *ordinary* differential equation, or 'ODE'.

Any one specific ODE may have a closed form analytical solution, but more generally it must be approximated (often in finite precision arithmetic), whereby no general solution is possible and a specific solution is commonly isolated by the provision of an initial value. Examples of such *numerical* methods are 'Runge-Kutta' (or RK) methods, and in a special case 'Runge-Kutta-Nyström' (or RKN) methods. The theoretical development of these processes, and their implementation, is described next, motivated by means of general convergence theory for a wider class of problems containing them, known as *single step methods*.

2.2 Initial value problems

For simplicity, the *scalar* first-order ODE initial value problem (IVP)

$$\frac{dy}{dx} = y' = f(x, y(x)),\ x \in [x_0, x_{end}],\ \text{where } y(x_0) = y_0, \tag{2.1}$$

may illustrate the basic principals. Numerical solution of (2.1) implies the process of obtaining a set of values $\{y_n\}$ corresponding to a discrete set of mesh points $\{x_n\}$ which are approximations to the specific solution curve $y(x)$ satisfying (2.1). Discrete variable methods of this type are most commonly used to propagate a series of approximations $y_n \approx y(x_n)$ on the set of points

$$x_{n+1} = x_0 + h_n,\ n = 0, \ldots, N-1,$$

where $x_N \leq x_{end}$. As the step size h_n is often variable, the idea of a maximum step size h can be used, writing $h_n = \Theta(x_n)h = \Theta_n h$, where $\Theta(x),\ x \in [x_0, x_{end}]$ is a piecewise constant function such that $\Theta_n \in (0, 1]$.

In general, the IVP can be regarded as a system of ODE's:

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}(x)),\ \boldsymbol{y}(x_0) = \boldsymbol{y_0}, \tag{2.2}$$

where $\boldsymbol{y}$, $\boldsymbol{y}_0$ and $\boldsymbol{f}$ are now *vectors* with components denoted by a preceding superscript, i.e., ${}^i y,\ i = 1, \ldots, m$, for $\boldsymbol{y} \in \mathbb{R}^m$. It should be noted that the problem definition (2.2) can be used to represent higher order systems (involving higher derivatives of $\boldsymbol{y}$ than the first) by extending the number of components of the system to include an equation for each derivative component also.

Definition 2.2.1 *A function $\boldsymbol{f}(x, \boldsymbol{y}(x))$ where $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$, satisfies a* <u>*Lipschitz condition*</u> *in the variable $\boldsymbol{y}$ on a region D, defined by*

$$x_0 \leq x \leq x_{end},\ -\infty < {}^i y < \infty,\ i = 1, \ldots, m,$$

where x_0 and x_{end} are finite, provided $\exists L > 0$ s.t.

$$\|\boldsymbol{f}(x, \boldsymbol{y}(x)) - \boldsymbol{f}(x, \boldsymbol{y}^*(x))\| \le L\|\boldsymbol{y}(x) - \boldsymbol{y}^*(x)\|,$$

wherever $(x, \boldsymbol{y}(x))$ and $(x, \boldsymbol{y}^*(x)) \in D$. The constant L is called a Lipschitz constant.

Theorem 2.2.1 (Picard's existence and uniqueness theorem) Suppose $\boldsymbol{f}(x, \boldsymbol{y}(x))$ where $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$, satisfies the following

1. $\boldsymbol{f}(x, \boldsymbol{y}(x))$ is defined and continuous for all $(x, \boldsymbol{y}(x))$ in the region D,
2. $\boldsymbol{f}(x, \boldsymbol{y}(x))$ satisfies a Lipschitz condition in $\boldsymbol{y}$ on D.

Then for every $\boldsymbol{\eta} \in \mathbb{R}^m$, $\exists$ a unique solution $\boldsymbol{y}(x)$ of the IVP

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}(x)), \;\; \boldsymbol{y}(x_0) = \boldsymbol{\eta},$$

where $\boldsymbol{y}(x)$ is defined and continuous $\forall (x, \boldsymbol{y}(x)) \in D$ (proof in Henrici [6]).

$\square$

Definition 2.2.2 *The initial value problem (2.2) is* <u>*well-posed*</u> *if, given any $\varepsilon > 0$, $\exists \delta > 0$ s.t. any other solution $\boldsymbol{v}(x)$ of the ODE*

$$\boldsymbol{v}' = \boldsymbol{f}(x, \boldsymbol{v}(x)), \;\; x > x_0$$

satisfying $\|\boldsymbol{y}(x_0) - \boldsymbol{v}(x_0)\| \le \delta$ also satisfies $\|\boldsymbol{y}(x) - \boldsymbol{v}(x)\| \le \varepsilon, \;\; \forall x > x_0$.

Theorem 2.2.2 (Gear [7]) The initial value problem (2.2) is well-posed if $\boldsymbol{f}(x, \boldsymbol{y}(x))$ satisfies the conditions of Theorem 2.2.1.

$\square$

Therefore, provided $\boldsymbol{f}(x, \boldsymbol{y}(x))$ satisfies Theorem 2.2.2 and hence Theorem 2.2.1, small perturbations in the stated problem will only lead to small changes in the answers. This is clearly necessary when attempting a numerical integration, and as such this condition will be assumed throughout.

2.3 Taylor and single RK processes

Assuming the existence of continuous bounded derivatives of $\boldsymbol{f}(x, \boldsymbol{y}(x))$ up to Q^{th} order, the true solution may be expanded in a Taylor series, therefore

$$\boldsymbol{y}(x_{n+1}) = \boldsymbol{y}(x_n + h_n) = \boldsymbol{y}(x_n) + h_n \boldsymbol{\Delta}(\boldsymbol{y}(x_n), h_n), \tag{2.3}$$

where

$$\boldsymbol{\Delta}(\boldsymbol{y}(x), h) = \sum_{i=0}^{Q-1} \frac{h^i}{(i+1)!} \boldsymbol{y}^{(i+1)}(x) + O(h^Q).$$

The Taylor algorithm of order $p < Q$ is obtained by truncating the series (2.3) at the h_n^p term, replacing the true value $\boldsymbol{y}(x_n)$ by the numerical approximation $\boldsymbol{y}_n$, thus taking the form

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h_n \boldsymbol{\Phi}(x_n, \boldsymbol{y}_n, h_n), \tag{2.4}$$

where

$$\boldsymbol{\Phi}(x, \boldsymbol{y}(x), h) = \sum_{i=0}^{p-1} \frac{h^i}{(i+1)!} \boldsymbol{y}^{(i+1)}(x).$$

The functions $\boldsymbol{\Delta}$ and $\boldsymbol{\Phi}$ in (2.3) and (2.4) are called the increment functions for the true solution and Taylor algorithm of order p, respectively. The Taylor algorithm (2.4) is termed a *single step* method, as it can propagate the solution in a step-by-step fashion requiring only a single back-value for the calculation of each approximation. It is therefore self-starting from the initial condition, and the step size can be variable. However, the requirement of total derivatives of $\boldsymbol{f}$, which are very complicated for even modest orders, makes the Taylor algorithm highly impractical.

Motivated by this difficulty of implementing a Taylor algorithm, another single step method without the need for any knowledge of the derivatives of $\boldsymbol{f}$ is the general linear Runge-Kutta method defined by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h_n \boldsymbol{\Phi}(x_n, \boldsymbol{y}_n, h_n) = \boldsymbol{y}_n + h_n \sum_{i=1}^{s} b_i \boldsymbol{g}_i, \tag{2.5}$$

where

$$\boldsymbol{g}_i = \boldsymbol{f}\left(x_n + c_i h_n, \boldsymbol{y}_n + h_n \sum_{j=1}^{s} a_{ij}\boldsymbol{g}_j \right),\ i = 1, \ldots, s.$$

The $\boldsymbol{g}_i,\ i = 1, \ldots, s$, are just function evaluations within the step, the increment function $\boldsymbol{\Phi}(x_n, \boldsymbol{y}_n, h_n)$ taking the form of a linear combination of these s slopes. If the $\boldsymbol{g}_i$ are to be explicitly calculated, it is required that

$$a_{ij} = 0,\ j \geq i,$$

and the process is then termed an *explicit* linear RK process. In this case, the usual enforcement of RK row-sum conditions

$$c_i = \sum_{j=1}^{s} a_{ij},\ i = 1, \ldots, s, \tag{2.6}$$

implies $c_1 = 0$ and the $\boldsymbol{g}_i$ of (2.5) can now be written

$$\boldsymbol{g}_i = \boldsymbol{f}\left(x_n + c_i h_n, \boldsymbol{y}_n + h_n \sum_{j=1}^{i-1} a_{ij}\boldsymbol{g}_j \right),\ i = 1, \ldots, s,$$

in which the summation is taken as zero when $i = 1$. It is conventional to represent the c_i (nodal), a_{ij} (internal weight) and b_i (external weight) parameters in the following tableau form (Butcher [8]):

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}$$

2.3.1 Global error, convergence and consistency

The numerical method should yield approximations of arbitrary accuracy. As the principal gauge of the accuracy of a process, the *global error* at a given mesh point $x_n \in [x_0, x_{end}]$ is defined as

$$\boldsymbol{\varepsilon}_n = \boldsymbol{y}_n - \boldsymbol{y}(x_n),\ n = 0, \ldots, N.$$

A single step method defined by (2.4) is convergent for the problem (2.2) if, for all fixed x_n in some region

$$\|\boldsymbol{\varepsilon}_n\| \to 0 \text{ as } h \to 0.$$

Thus, convergence of the method ensures the possibility of generating solutions of different accuracy by varying the step sizes used. Whether *arbitrary* accuracy can be achieved depends on the perturbing effect of floating point arithmetic on the numerical solution.

Definition 2.3.1 (Consistency) *The numerical method (2.4) is consistent with the problem (2.2) if*

$$\boldsymbol{\Phi}(x, \boldsymbol{y}(x), 0) = \boldsymbol{f}(x, \boldsymbol{y}(x)).$$

By inspection of (2.4), this is immediately true for the Taylor algorithm. For the RK algorithm (2.5),

$$\boldsymbol{\Phi}(x_n, \boldsymbol{y}_n, 0) = \boldsymbol{f}(x_n, \boldsymbol{y}_n) \sum_{i=1}^{s} b_i.$$

Thus the RK process is consistent if

$$\sum_{i=1}^{s} b_i = 1. \tag{2.7}$$

It can be shown (Henrici [6]) that consistency of a single step method leads to convergence, and hence Taylor algorithms and all RK methods satisfying (2.7) are convergent. To introduce the idea of *rate* of convergence, consider the local truncation error $\boldsymbol{t}_{n+1}$, defined as the amount by which the true solution fails to satisfy (2.4), i.e.,

$$\boldsymbol{t}_{n+1} = \boldsymbol{y}(x_n) + h_n \boldsymbol{\Phi}(x_n, \boldsymbol{y}(x_n), h_n) - \boldsymbol{y}(x_{n+1}). \tag{2.8}$$

As such, the local truncation error may be thought of as the error per step, and is another measure of the accuracy of a process. Substituting for $\boldsymbol{y}(x_{n+1})$ from the Taylor expansion (2.3),

$$\boldsymbol{t}_{n+1} = h_n \left[\boldsymbol{\Phi}(x_n, \boldsymbol{y}(x_n), h_n) - \boldsymbol{\Delta}(\boldsymbol{y}(x_n), h_n) \right]. \tag{2.9}$$

Assuming appropriate smoothness of $\boldsymbol{\Phi}$ and $\boldsymbol{f}$:

$$\begin{aligned} \boldsymbol{t}_{n+1} &= h_n \left[\sum_{i=0}^{Q-1} \frac{h_n^i}{i!} \frac{\partial^i}{\partial h_n^i} \boldsymbol{\Phi}(x_n, \boldsymbol{y}(x_n), 0) - \sum_{i=0}^{Q-1} \frac{h_n^i}{(i+1)!} \boldsymbol{y}^{(i+1)}(x_n) + O(h_n^Q) \right] \\ &= \sum_{i=1}^{Q} h_n^i \boldsymbol{\phi}_i(\boldsymbol{y}(x_n)) + O(h_n^{Q+1}). \end{aligned} \tag{2.10}$$

where

$$\boldsymbol{\phi}_i(\boldsymbol{y}(x)) = \frac{1}{(i-1)!} \frac{\partial^{(i-1)}}{\partial h_n^{i-1}} \boldsymbol{\Phi}(x, \boldsymbol{y}(x), 0) - \frac{1}{i!} \boldsymbol{y}^{(i)}(x), \ i = 1, \ldots, Q.$$

These $\boldsymbol{\phi}_i$ are termed *error functions*, and the order of a single step process can now be defined as the number of leading zero terms in the expansion (2.10), i.e., a p^{th} order process (2.4) is such that

$$\boldsymbol{\phi}_i = \boldsymbol{0}, \ i = 1, \ldots, p, \text{ and } \boldsymbol{\phi}_{p+1} \neq \boldsymbol{0} \ (p < Q).$$

The function $\boldsymbol{\phi}_{p+1}$ is called the principal error function. For the Taylor algorithm,

$$\boldsymbol{\phi}_i(\boldsymbol{y}(x)) = -\frac{1}{i!} \boldsymbol{y}^{(i)}(x), \ i = p+1, \ldots, Q.$$

For an RK process (2.5), following Butcher [9], the error functions can be expressed as combinations of the partial derivatives of $\boldsymbol{f}$:

$$\boldsymbol{\phi}_i(\boldsymbol{y}(x)) = \sum_{j=1}^{r_i} \tau_j^{(i)} \boldsymbol{F}_j^{(i)}(x_n, \boldsymbol{y}(x_n)),\ i = 1, \ldots, Q,$$

where the $\boldsymbol{F}_j^{(i)}$, $j = 1, \ldots, r_i$, are the i^{th} order *elementary differentials* of $\boldsymbol{f}$. Since the $\boldsymbol{F}_j^{(i)}$ form a basis, a p^{th} order RK process must have

$$\tau_j^{(i)} = 0,\ j = 1, \ldots, r_i,\ i = 1, \ldots, p.$$

These are called the RK equations of condition, each τ being a function of the RK parameters c_i, a_{ij} and b_i. An algorithm for generation of these τ's is described in Section 3.2. With advance knowledge of this, however, it is noted that the RK consistency condition (2.7) is the same condition as that for first-order, since

$$\tau_1^{(1)} = \sum_{i=1}^{s} b_i - 1.$$

All useful RK processes are therefore consistent, and the order of the method is the gauge used for the rate of its convergence in general.

Definition 2.3.2 *An s-stage p^{th} order single linear RK process will henceforth be denoted by* RK_s^p *(for brevity).*

2.3.2 Absolute stability of single RK methods

The previous convergence theory was developed using a limiting process as the step $h \to 0$. In practice, however, numerical integration of a given ODE is concerned with the use of a *finite* step length, requiring that perturbations introduced in any step of the calculation (due to floating point round-off or truncation) are not amplified unboundedly as the calculation proceeds. The term *absolutely stable* is used for any particular method whenever the global error $\|\boldsymbol{\varepsilon}_n\|$ remains bounded as $n \to \infty$. Many methods, including RK's, are absolutely stable only if the step length is less than a certain value. Following Lambert [10], for a system of ODE's, linearisation of the problem (2.2) gives

$$\boldsymbol{y}' = A\boldsymbol{y}, \tag{2.11}$$

where A is a square matrix of constants. For an s-stage linear RK, uncoupling is possible, and it is enough to consider the scalar equation

$$y' = \lambda y, \tag{2.12}$$

where λ is one of the eigenvalues of A and $Re(\lambda) < 0$. Owing to the linear nature of (2.12), all $F_j^{(i)}$ containing partial derivatives with respect to x, or with respect to y above the first order (i.e., f_{yy}, f_{yyy}, etc.) are zero in $\phi_i(y(x))$. Further details of such properties will be given in Section 4.2.1, mentioned here for convenience, as the condition for absolute stability is now easily characterised in terms of the single $\tau_j^{(i)}$ coefficient remaining in the local truncation error expansion for order i, which may be denoted here by $\tau_{l_i}^{(i)}$. Absolute stability of an *explicit* RK process is thus defined (modified from Harris [11]) wherever

$$\left| 1 + \sum_{i=1}^{s} \left(\tau_{l_i}^{(i)} + \frac{1}{i!} \right) (h\lambda)^i \right| < 1. \tag{2.13}$$

Definition 2.3.3 (Real negative stability limit) *The real negative stability limit of an explicit RK process is defined as S_R, the intersection of the boundary of validity of (2.13) with the negative real axis, i.e., when*

$$\left| 1 + \sum_{i=1}^{s} \left(\tau_{l_i}^{(i)} + \frac{1}{i!} \right) (S_R)^i \right| = 1, \ \textit{for } S_R < 0 \in \mathbb{R}.$$

It is conventional when designing RK methods to use the real negative stability limit as a gauge of absolute stability, despite its lack of applicability to nonlinear equations. Note that, due to the finite real negative stability of RK methods, they are not efficient when applied to stiff ODE systems[1], and the problem will from now on be assumed non-stiff.

2.4 RK pairs

The standard explicit RK_s^p algorithm forms a linear combination of s function evaluations within the step to give an approximation of order p at the end of the step. The technique of *embedding* utilises the ability to form a separate linear combination (using a different set of b_i parameters) of the same s stages (i.e., the same c_i and a_{ij}'s) to provide a concurrent approximation of higher order, q. Often, this requires special RK_s^p processes, using extra stages than necessary, in order to contain an embedded formula of higher order. Following Prince [12], the q^{th} order processes will be denoted by the addition of a cap $(\widehat{\cdot})$ onto relevant symbols in the notation. Usually, such a pair of formulae allows a choice of which approximation to use to propagate the solution. If the q^{th} order result continues the integration, it is said to be applied in *higher-order mode*. Otherwise, the *lower-order mode* may be selected.

Definition 2.4.1 *An s-stage linear RK pair of orders p and $q > p$ will be denoted as $\mathrm{RK}_s^{p(q)}$ or $\mathrm{RK}_s^{q(p)}$, designed for application in lower-order or higher-order mode, respectively.*

The $\mathrm{RK}_s^{q(p)}$ algorithm takes the form

$$\begin{aligned} \widehat{\boldsymbol{y}}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n \widehat{\boldsymbol{\Phi}}(x_n, \widehat{\boldsymbol{y}}_n, h_n) = \widehat{\boldsymbol{y}}_n + h_n \sum_{i=1}^{s} \widehat{b}_i \boldsymbol{g}_i, \\ \boldsymbol{y}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n \boldsymbol{\Phi}(x_n, \widehat{\boldsymbol{y}}_n, h_n) = \widehat{\boldsymbol{y}}_n + h_n \sum_{i=1}^{s} b_i \boldsymbol{g}_i, \end{aligned} \tag{2.14}$$

where

$$\boldsymbol{g}_i = \boldsymbol{f}\left(x_n + c_i h_n, \widehat{\boldsymbol{y}}_n + h_n \sum_{j=1}^{s} a_{ij} \boldsymbol{g}_j \right), \; i = 1, \ldots, s.$$

[1]ODE stiffness is hard to define exactly, but generally manifests itself as a limiting of the solver step size by factors other than asymptotic accuracy.

Now, from (2.9), and noting that $\widehat{\boldsymbol{t}}_{n+1} = O(h^{q+1})$ and $\boldsymbol{t}_{n+1} = O(h^{p+1})$,

$$\widehat{\boldsymbol{\Phi}}(x, \boldsymbol{y}(x), h) = \boldsymbol{\Delta}(\boldsymbol{y}(x), h) + O(h^q)$$

and

$$\boldsymbol{\Phi}(x, \boldsymbol{y}(x), h) = \boldsymbol{\Delta}(\boldsymbol{y}(x), h) + O(h^p).$$

Thus, from (2.14):

$$\boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1} = \boldsymbol{t}_{n+1} - \widehat{\boldsymbol{t}}_{n+1} + O(h^{p+q+1}),$$

giving

$$\boldsymbol{t}_{n+1} = \boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1} + O(h^{q+1}).$$

For the $\mathrm{RK}_s^{q(p)}$ process, $\boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1}$ can therefore be used as an estimate of the local truncation error in the p^{th} order formula. A similar result holds for an $\mathrm{RK}_s^{p(q)}$ process.

Definition 2.4.2 (Local error) *The local error, $\boldsymbol{l}_{n+1}$ at $x_{n+1} \in [x_0, x_{end}]$, of a method defined by (2.4) is given by*

$$\boldsymbol{l}_{n+1} = \boldsymbol{y}_{n+1} - \boldsymbol{v}(x_{n+1}) \tag{2.15}$$

where $\boldsymbol{v}(x)$ solves the IVP

$$\boldsymbol{v}'(x) = \boldsymbol{f}(x, \boldsymbol{v}(x)), \;\; \boldsymbol{v}(x_n) = \boldsymbol{y}_n. \tag{2.16}$$

When considering error estimates, the local error (2.15) is an alternative to the local truncation error, with the advantage that it can be computed by solving the IVP (2.16). By a similar analysis to that of the local truncation error, essentially replacing $\boldsymbol{y}_n$ by $\boldsymbol{y}(x_n)$, it can be shown that

$$\boldsymbol{l}_{n+1} = \boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1} + O(h^{q+1}),$$

and so for an $\mathrm{RK}_s^{q(p)}$, $\boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1}$ can also be regarded as an estimate of the local error in the p^{th} order formula. Note that in the case of higher-order mode, the local (truncation) error estimated is that in the lower-order formula, while the integration is progressed with the higher-order formula — effectively, the approximate solution is being *extrapolated* from the lower-order result, and so higher-order mode is often referred to as *local extrapolation mode.*

2.4.1 Step size control for RK pairs

With the availability of a cheap local (truncation) error estimate:

$$\boldsymbol{\delta}_{n+1} = \boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1}$$

for error per step control, or

$$\boldsymbol{\delta}_{n+1} = (\boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1})/h_n$$

for error per *unit* step control, an optimal next step h_{n+1} is traditionally predicted using the formula (Hull, et al. [13])

$$h_{n+1} = 0.9 h_n \left[T \times \min_i \left\{ \left| {}^iE / {}^i\delta_{n+1} \right| \right\} \right]^{\frac{1}{r}}, \tag{2.17}$$

where T is a tolerance parameter, $\boldsymbol{E}$ is a form of the solution vector, and r is either $p+1$ for error per step, or p for error per unit step. $\boldsymbol{E}$ is chosen from one of

1. Absolute error control

$$\boldsymbol{E} = \boldsymbol{u},$$

2. Relative error control

$$\boldsymbol{E} = \boldsymbol{Y}_{n+1}$$

3. Mixed error control

$$\boldsymbol{E} = e_1 \cdot \boldsymbol{u} + e_2 \cdot \boldsymbol{Y}_{n+1},$$

where $\boldsymbol{u}$ is a vector of 1's, and $\boldsymbol{Y}_{n+1}$ is the approximation on which the integration is advanced, i.e., $\widehat{\boldsymbol{y}}_{n+1}$ in local extrapolation mode. For mixed error control, the parameters e_1 and e_2 control the 'mix' of the other two forms[2].

In the case where h_{n+1} is predicted to be *less than* h_n, the step is commonly rejected and re-computed using the smaller step, which may be restricted to be no less than a certain ratio of h, 1/10 say. In certain circumstances, namely when

$$h_{n+1} < \left(\frac{1}{2}\right)^{p+1} h_n$$

(for error per step), the step size control is deemed to have failed, and the step is often arbitrarily halved. Also, a limit may be imposed on the largest proportional increase of h, e.g., that it may be no more than *doubled* in one step, say.

2.4.2 The FSAL stage for RK pairs

Embedded (explicit) RK pairs often make use of what is known as the FSAL technique (meaning 'first same as last' — Prince [12]). Here, the last function evaluation in any step is the same as the first function evaluation of the next step.

Definition 2.4.3 FSAL $\mathrm{RK}_s^{q(p)}$ *and* $\mathrm{RK}_s^{p(q)}$ *pairs will be denoted by the addition of a subscript f into the notation, i.e.,* $\mathrm{RK}_{sf}^{q(p)}$ *and* $\mathrm{RK}_{sf}^{p(q)}$*, respectively.*

The FSAL stage must clearly have $c_s = 1$, but also required for an $\mathrm{RK}_{sf}^{q(p)}$ is that

$$a_{si} = \widehat{b}_i, \;\; i = 1, \ldots, s-1, \tag{2.18}$$

and from the row-sum conditions (2.6), and consistency condition (2.7), this must require

$$\widehat{b}_s = 0. \tag{2.19}$$

[2]It is common to use $e_1 = e_2 = 1$.

$\text{RK}_{sf}^{p(q)}$ formulae similarly require

$$a_{si} = b_i, \; i = 1, \ldots, s-1, \; \text{and } b_s = 0.$$

For an $\text{RK}_{sf}^{q(p)}$ process to be practical, it is normal for the p^{th} order formula to use one more stage than for the q^{th} order result. The extra b_s parameter is, however, an extra degree of freedom, and provided that a step is not rejected, no extra cost is incurred over an $s-1$ stage, non-FSAL formula. It should be noted that a slight disadvantage is the loss of freedom as to whether higher-order or lower-order mode is used, without the need for an extra evaluation per step.

From now on, unless otherwise stated, local extrapolation mode can be assumed throughout.

2.5 RK triples

By adding a third RK algorithm to an embedded $\text{RK}_{s}^{q(p)}$ or $\text{RK}_{sf}^{q(p)}$ pair, the numerical solution of the initial value problem (2.2) may be determined at non-mesh points σh_n, where normally $0 < \sigma < 1$. This so-called *dense* formula may be expressed in the usual manner:

$$\boldsymbol{y}_{n+\sigma}^* = \widehat{\boldsymbol{y}}_n + \sigma h_n \boldsymbol{\Phi}^*(x_n, \widehat{\boldsymbol{y}}_n, \sigma h_n) = \widehat{\boldsymbol{y}}_n + \sigma h_n \sum_{i=1}^{s^*} b_i^* \boldsymbol{g}_i^*$$

where

$$\boldsymbol{g}_i^* = \boldsymbol{f}\left(x_n + c_i^* \sigma h_n, \widehat{\boldsymbol{y}}_n + \sigma h_n \sum_{j=1}^{i-1} a_{ij}^* \boldsymbol{g}_j^* \right), \; i = 1, \ldots, s^*.$$

Ideally, if $s^* = s$ and $\boldsymbol{g}_i^* = \boldsymbol{g}_i, \; i = 1, \ldots, s^*$, no extra function evaluations would be required to compute the approximation for $\boldsymbol{y}(x_n + \sigma h_n)$. Whether or not this is possible for given q, p and p^*, where p^* is the order of the dense process, common function evaluations are guaranteed if

$$c_i^* \;=\; c_i/\sigma, \; i = 1, \ldots, s,$$

$$a^*_{ij} \;=\; a_{ij}/\sigma,\ i = 2,\ldots,s,\ j = 1,\ldots,i-1,$$

where it is assumed that $s^* \geq s$, though this is not necessary in some circumstances, namely when a very low order ($p^* \ll q$) dense algorithm is involved. For simplicity it is also convenient to satisfy

$$\begin{aligned} c^*_i &= c_i/\sigma,\ i = s+1,\ldots,s^*, \\ a^*_{ij} &= a_{ij}/\sigma,\ i = s+1,\ldots,s^*,\ j = 1,\ldots,i-1, \end{aligned}$$

in order that all $\boldsymbol{g}_i,\ i = 1,\ldots,s^*$, are independent of σ to avoid possible singular cases, and as such will be assumed throughout.

Definition 2.5.1 *An* $\mathrm{RK}^{q(p)}_{s}$ *or* $\mathrm{RK}^{q(p)}_{sf}$ *pair with an* s^**-stage continuous extension of order* p^* *will be denoted as* $\mathrm{RK}^{q(p)}_{s}\mathrm{D}^{p^*}_{s^*}$ *or* $\mathrm{RK}^{q(p)}_{sf}\mathrm{D}^{p^*}_{s^*}$*, respectively.*

The general RK triple now takes the form

$$\begin{aligned} \widehat{\boldsymbol{y}}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n\widehat{\boldsymbol{\Phi}}(x_n,\widehat{\boldsymbol{y}}_n,h_n) \;=\; \widehat{\boldsymbol{y}}_n + h_n\sum_{i=1}^{s}\widehat{b}_i\boldsymbol{g}_i, \\ \boldsymbol{y}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n\boldsymbol{\Phi}(x_n,\widehat{\boldsymbol{y}}_n,h_n) \;=\; \widehat{\boldsymbol{y}}_n + h_n\sum_{i=1}^{s}b_i\boldsymbol{g}_i, \\ \boldsymbol{y}^*_{n+1} &= \widehat{\boldsymbol{y}}_n + \sigma h_n\boldsymbol{\Phi}^*(x_n,\widehat{\boldsymbol{y}}_n,\sigma h_n) \;=\; \widehat{\boldsymbol{y}}_n + \sigma h_n\sum_{i=1}^{s^*}b^*(\sigma)_i\boldsymbol{g}_i, \end{aligned} \tag{2.20}$$

where

$$\boldsymbol{g}_i = \boldsymbol{f}\left(x_n + c_ih_n, \widehat{\boldsymbol{y}}_n + h_n\sum_{j=1}^{i-1}a_{ij}\boldsymbol{g}_j\right),\ i = 1,\ldots,s^*,$$

and external weights for the discrete process are defined to be zero for stages $s+1$ to s^*, i.e.,

$$b_i = \widehat{b}_i = 0,\ i = s+1,\ldots,s^*.$$

This process may be represented by the extended Butcher tableau in Figure 2.1. Dense RK processes of order p^* require that a Taylor expansion in σh of the true solution matches that of the continuous approximation to order p^*, equivalent to the discrete case of Section 2.3.1. This therefore requires a set of continuous truncation error coefficients (τ^*'s) to be zero — the necessary form of which is considered in Section 3.2.

$$
\begin{array}{c|ccccccccc}
0 & & & & & & & & & \\
c_2 & a_{21} & & & & & & & & \\
c_3 & a_{31} & a_{32} & & & & & & & \\
\vdots & \vdots & \vdots & \ddots & & & & & & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & & & & & \\
\cline{1-6}
c_{s+1} & a_{s+1,1} & a_{s+1,2} & \cdots & a_{s+1,s-1} & a_{s+1,s} & & & & \\
c_{s+2} & a_{s+2,1} & a_{s+2,2} & \cdots & a_{s+2,s-1} & a_{s+2,s} & a_{s+2,s+1} & & & \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & & \\
c_{s^*} & a_{s^*,1} & a_{s^*,2} & \cdots & a_{s^*,s-1} & a_{s^*,s} & a_{s^*,s+1} & \cdots & a_{s^*,s^*-1} & \\
\hline
 & \widehat{b}_1 & \widehat{b}_2 & \cdots & \widehat{b}_{s-1} & \widehat{b}_s & 0 & \cdots & 0 & 0 \\
\cline{2-10}
 & b_1 & b_2 & \cdots & b_{s-1} & b_s & 0 & \cdots & 0 & 0 \\
\cline{2-10}
 & b_1^* & b_2^* & \cdots & b_{s-1}^* & b_s^* & b_{s+1}^* & \cdots & b_{s^*-1}^* & b_{s^*}^*
\end{array}
$$

Figure 2.1: The Butcher tableau of a Runge-Kutta triple.

2.5.1 Stability limits for an RK triple

The concept of real negative stability can be extended to each of the formulae of an RK triple, including the dense process. By replacing the τ's from the definition (2.3.3) with those of the desired process (and s by s^* for the dense), an equivalent definition can be obtained. Specifically, for the interpolant of an RK triple, the following definition applies:

Definition 2.5.2 (Continuous real negative stability limit) *The real negative stability limit of a continuous explicit RK process $S_R^*(\sigma)$, $0 \le \sigma \le 1$ is defined such that*

$$\left| 1 + \sum_{i=1}^{s^*} \left(\tau_{l_i}^{*(i)} + \frac{1}{i!} \right) S_R^*(\sigma)^i \right| = 1, \ \textit{for } S_R^*(\sigma) < 0 \in I\!R, \ 0 \le \sigma \le 1,$$

where $\tau_{l_i}^{*(i)}$ multiplies the single elementary differential $F_{l_i}^{(i)}$ for order i, as in Section 2.3.2.

The concept of absolute stability is usually only applied to discrete processes, the definition here still applies, however, in that the solution to the linearised problem (2.12) for $Re(\lambda) < 0$ is still required to decrease for any intermediate point $x_n + \sigma h_n$ from the dense output. This gives rise to the concept that an interpolant may exhibit absolute instability at intermediate points in the step, even though it is within the linear stability limit at $\sigma = 1$.

2.5.2 Continuity of dense RK processes

Concerning the global continuity of the RK interpolant, for a C^1 continuous solution it is required that

$$\boldsymbol{y}^*_{n+\sigma}\Big|_{\sigma=0} = \widehat{\boldsymbol{y}}_n, \;\; \frac{d\boldsymbol{y}^*_{n+\sigma}}{dx}\Bigg|_{\sigma=0} = \boldsymbol{f}(x_n, \widehat{\boldsymbol{y}}_n),$$

and

$$\boldsymbol{y}^*_{n+\sigma}\Big|_{\sigma=1} = \widehat{\boldsymbol{y}}_{n+1}, \;\; \frac{d\boldsymbol{y}^*_{n+\sigma}}{dx}\Bigg|_{\sigma=1} = \boldsymbol{f}(x_{n+1}, \widehat{\boldsymbol{y}}_{n+1}).$$

These conditions will be true if (Dormand and Prince [14]):

$$b^*_i(1) = \widehat{b}_i, \;\; b^*_i(0) = \delta_{i1}, \;\; \text{and} \;\; \widehat{b}_i + \frac{db^*_i}{d\sigma}\Bigg|_{\sigma=1} = \delta_{is^r}, \tag{2.21}$$

where

$$\delta_{ij} = \begin{cases} 0, & i \neq j, \\ 1, & i = j, \end{cases} \quad i, j = 1, \ldots, s^*, \tag{2.22}$$

and stage s^r is reusable as the first evaluation of the next step. In the case where $s^r = s$, this is an FSAL stage, but if the underlying pair is not of FSAL design, for C^1 continuity one of the function evaluations $s+1$ to s^* must be derived to satisfy

$$a_{s^r i} = \widehat{b}_i, \;\; i = 1, \ldots, s^r - 1, \;\; \text{where} \;\; s+1 \leq s^r \leq s^*.$$

It is usual, and will be assumed, that in such circumstances $s^r = s + 1$.

2.6 RKN pairs

The special second-order initial value problem in ODE's

$$\frac{d^2\boldsymbol{y}}{dx^2} = \boldsymbol{y}'' = \boldsymbol{f}(x, \boldsymbol{y}(x)),\ \boldsymbol{y}(x_0) = \boldsymbol{y}_0,\ \boldsymbol{y}'(x_0) = \boldsymbol{y}'_0, \tag{2.23}$$

frequently occurs in mechanics, applied mathematics and astronomy. Deemed 'special' due to the lack of $\boldsymbol{y}'(x)$ in the problem definition $\boldsymbol{f}$, it may be solved more efficiently using numerical methods developed specifically for the purpose than with general methods such as the Runge-Kutta algorithm. Similar to the case for general first-order systems (2.2), a unique solution to (2.23) is guaranteed (extended from Henrici [6]) if the function $\boldsymbol{f}(x, \boldsymbol{y}(x))$ satisfies a Lipschitz condition (2.2.1). One method for numerically determining this solution as a series of approximations $\boldsymbol{y}_n$ and $\boldsymbol{y}'_n$ is the explicit Runge-Kutta-Nyström method. The technique of embedding, i.e., adding a second algorithm for estimation of the local error, is parallel with the RK case, and as such the general $\mathrm{RKN}_s^{q(p)}$ or $\mathrm{RKN}_{sf}^{q(p)}$ pair is given by

$$\begin{aligned}
\widehat{\boldsymbol{y}}'_{n+1} &= \widehat{\boldsymbol{y}}'_n + h_n \widehat{\boldsymbol{\Phi}}'(x_n, \widehat{\boldsymbol{y}}_n, \widehat{\boldsymbol{y}}'_n, h_n) &&= \widehat{\boldsymbol{y}}'_n + h_n \sum_{i=1}^{s} \widehat{b}'_i \boldsymbol{g}_i, \\
\boldsymbol{y}'_{n+1} &= \widehat{\boldsymbol{y}}'_n + h_n \boldsymbol{\Phi}'(x_n, \widehat{\boldsymbol{y}}_n, \widehat{\boldsymbol{y}}'_n, h_n) &&= \widehat{\boldsymbol{y}}'_n + h_n \sum_{i=1}^{s} b'_i \boldsymbol{g}_i, \\
\widehat{\boldsymbol{y}}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n \widehat{\boldsymbol{\Phi}}(x_n, \widehat{\boldsymbol{y}}_n, \widehat{\boldsymbol{y}}'_n, h_n) &&= \widehat{\boldsymbol{y}}_n + h_n \widehat{\boldsymbol{y}}'_n + h_n^2 \sum_{i=1}^{s} \widehat{b}_i \boldsymbol{g}_i, \\
\boldsymbol{y}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n \boldsymbol{\Phi}(x_n, \widehat{\boldsymbol{y}}_n, \widehat{\boldsymbol{y}}'_n, h_n) &&= \widehat{\boldsymbol{y}}_n + h_n \widehat{\boldsymbol{y}}'_n + h_n^2 \sum_{i=1}^{s} b_i \boldsymbol{g}_i,
\end{aligned}$$

where

$$\boldsymbol{g}_i = \boldsymbol{f}\left(x_n + c_i h_n, \widehat{\boldsymbol{y}}_n + c_i h_n \widehat{\boldsymbol{y}}'_n + h_n^2 \sum_{j=1}^{i-1} a_{ij} \boldsymbol{g}_j\right),\ i = 1, \ldots, s.$$

Equivalent row-sum conditions for RKN methods are imposed as follows

$$\sum_{j=1}^{s} a_{ij} = \frac{c_i^2}{2},\ i = 1, \ldots, s, \tag{2.24}$$

again, implying $c_1 = 0$. A higher-order mode FSAL process here requires the conditions

$$c_s = 1,\ a_{si} = \widehat{b}_i,\ i = 1, \dots, s-1,\ \text{and } \widehat{b}_s = 0. \tag{2.25}$$

2.6.1 Convergence and consistency for single RKN methods

With global errors defined naturally as for the RK case, a *single* RKN process is said to be convergent for the problem (2.23) if, for all fixed x_n in some region,

$$\|\boldsymbol{\varepsilon}_n, \boldsymbol{\varepsilon}'_n\| \to 0 \text{ as } h \to 0.$$

Consistency with the problem (2.23) is ensured if

$$\boldsymbol{\Phi}(x, \boldsymbol{y}, \boldsymbol{y}', 0) = \boldsymbol{y}',$$

and

$$\boldsymbol{\Phi}'(x, \boldsymbol{y}, \boldsymbol{y}', 0) = \boldsymbol{f}(x, \boldsymbol{y}(x)).$$

These conditions are true when

$$\sum_{i=1}^{s} \widehat{b}'_i = 1, \tag{2.26}$$

and similar to RK consistency, expression (2.26) is the same condition as that for first-order. Again, this implies convergence under certain conditions (Henrici [6]), and obtaining a numerical solution to any desired accuracy should be possible.

The local truncation errors of the single RKN process may be written (Dormand and Prince [15]), assuming appropriate smoothness of $\boldsymbol{f}$, $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}'$:

$$\boldsymbol{t}_{n+1} = \sum_{i=1}^{Q-1} h^{i+1} \sum_{j=1}^{n_i} \tau_j^{(i+1)} \mathrm{F}_j^{(i)}(x_n, \boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)) + O(h_n^{Q+1}),$$

and

$$\boldsymbol{t}'_{n+1} = \sum_{i=0}^{Q-1} h^{i+1} \sum_{j=1}^{n_{i+1}} \tau_j'^{(i+1)} \mathrm{F}_j^{(i+1)}(x_n, \boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)) + O(h_n^{Q+1}).$$

The elementary differentials for the problem (2.23) are similar to those of the RK case, though for a given order p,

$$n_p \leq r_p$$

due to the lack of certain combinations of partial derivatives of the special problem (2.23). A p^{th} $(p < Q)$ order single RKN formula is such that

$$\boldsymbol{t}_{n+1} = O(h^{p+1}) \text{ and } \boldsymbol{t}'_{n+1} = O(h^{p+1}),$$

requiring

$$\tau_j^{(i)} = 0,\ j = 1, \ldots, n_{i-1},\ i = 2, \ldots, p,$$

$$\tau_j^{(p+1)} \neq 0, \text{ for at least one of } j = 1, \ldots, n_p,$$

and

$$\tau_j^{\prime(i)} = 0,\ j = 1, \ldots, n_i,\ i = 1, \ldots, p,$$

$$\tau_j^{\prime(p+1)} \neq 0, \text{ for at least one of } j = 1, \ldots, n_p,$$

which are functions of the RKN parameters b_i, b'_i, c_i and a_{ij}'s.

2.6.2 Step size control for RKN pairs

Similar to the RK case, local error estimates for $\boldsymbol{y}'_{n+1}$ $(\boldsymbol{\delta}'_{n+1})$ and for $\boldsymbol{y}_{n+1}$ $(\boldsymbol{\delta}_{n+1})$ are available using the embedded (lower-order) formula, i.e.,

$$\boldsymbol{\delta}'_{n+1} = \boldsymbol{y}'_{n+1} - \widehat{\boldsymbol{y}}'_{n+1},\ \boldsymbol{\delta}_{n+1} = \boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1}$$

for error per step control, or

$$\boldsymbol{\delta}'_{n+1} = (\boldsymbol{y}'_{n+1} - \widehat{\boldsymbol{y}}'_{n+1})/h_n,\ \boldsymbol{\delta}_{n+1} = (\boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1})/h_n$$

for error per *unit* step control. An optimal next step h_{n+1} can then be predicted using a form of (2.17) from Section 2.4.1:

$$h_{n+1} = 0.9h_n \left[T \times \min\left(\min_i \left\{ \left| {}^iE/{}^i\delta_{n+1} \right| \right\}, \min_i \left\{ \left| {}^iE'/{}^i\delta'_{n+1} \right| \right\} \right) \right]^{\frac{1}{r}},$$

where r is either $p+1$ for error per step or p for error per unit step. Assuming higher-order mode, $\boldsymbol{E}$ and $\boldsymbol{E}'$ are chosen from

1. Absolute error control
$$\boldsymbol{E}' = \boldsymbol{u}, \;\; \boldsymbol{E} = \boldsymbol{u},$$
2. Relative error control
$$\boldsymbol{E}' = \widehat{\boldsymbol{y}}'_{n+1}, \;\; \boldsymbol{E} = \widehat{\boldsymbol{y}}_{n+1},$$
3. Mixed error control
$$\boldsymbol{E}' = e_1 \cdot \boldsymbol{u} + e_2 \cdot \widehat{\boldsymbol{y}}'_{n+1}, \quad \boldsymbol{E} = e_1 \cdot \boldsymbol{u} + e_2 \cdot \widehat{\boldsymbol{y}}_{n+1},$$

where $\boldsymbol{u}$ is a vector of 1's. Special circumstances regarding the quality of the error estimate, step rejections, and choice of the next step size are the same as for the RK case.

It should be noted that step size control need not involve both $\boldsymbol{y}$ *and* $\boldsymbol{y}'$ solutions. This is considered in the general review of RKN pairs in Section 4.6.

2.7 RKN triples

Continuous approximation at non-mesh points $x_n + \sigma h_n$ can be done using a dense RKN process of the form:

$$\begin{aligned} \boldsymbol{y'}^*_{n+1} &= \widehat{\boldsymbol{y}}'_n + \sigma h_n \boldsymbol{\Phi}'^*(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, \sigma h_n) = \widehat{\boldsymbol{y}}'_n + \sigma h_n \sum_{i=1}^{s^*} b'^*_i \boldsymbol{g}^*_i \\ \boldsymbol{y}^*_{n+1} &= \widehat{\boldsymbol{y}}_n + \sigma h_n \boldsymbol{\Phi}^*(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, \sigma h_n) = \widehat{\boldsymbol{y}}_n + \sigma h_n \widehat{\boldsymbol{y}}'_n + \sigma^2 h_n^2 \sum_{i=1}^{s^*} b^*_i \boldsymbol{g}^*_i, \end{aligned}$$

where

$$\boldsymbol{g}_i^* = \boldsymbol{f}\left(x_n + \sigma c_i^* h_n, \widehat{\boldsymbol{y}}_n + \sigma c_i^* h_n \widehat{\boldsymbol{y}}_n' + \sigma^2 h_n^2 \sum_{j=1}^{i-1} a_{ij}^* \boldsymbol{g}_j^*\right),\ i = 1, \ldots, s^*.$$

Assuming $s^* \geq s$, $b_i = \widehat{b}_i = b_i' = \widehat{b}_i' = 0,\ i = s+1, \ldots, s^*$, and imposing

$$c_i^* = c_i/\sigma,\ i = 1, \ldots, s^*,$$

$$a_{ij}^* = a_{ij}/\sigma^2,\ i = 1, \ldots, s^*,\ j = 1, \ldots, i-1,$$

common function evaluations are guaranteed and all $\boldsymbol{g}_i$ are independent of σ as in the RK case. RKN pairs with an order p^* continuous extension are called RKN triples, denoted by $\mathrm{RKN}_s^{q(p)}\mathrm{D}_{s^*}^{p^*}$ (or $\mathrm{RKN}_{sf}^{q(p)}\mathrm{D}_{s^*}^{p^*}$ with an underlying FSAL discrete), defined in Figure 2.2.

2.7.1 Absolute stability of RKN methods

Following Horn [16], the test problem

$$y'' = \lambda^2 y, \text{ for } \lambda \text{ complex},$$

yields a definition of absolute stability as follows:

Definition 2.7.1 (Real negative stability limits) *The real negative stability limits of a discrete explicit RKN process are defined as S_R such that*

$$\left|1 + S_R + \sum_{i=2}^{2s} \left(\tau_{l_{i-1}^n}^{(i)} + \frac{1}{i!}\right)(S_R)^i\right| = 1,\ \textit{for } S_R < 0 \in I\!R,$$

and S_R' such that

$$\left|1 + \sum_{i=1}^{2s-1} \left(\tau_{l_i^n}'^{(i)} + \frac{1}{i!}\right)(S_R')^i\right| = 1,\ \textit{for } S_R' < 0 \in I\!R,$$

$$
\begin{aligned}
\widehat{\boldsymbol{y}}'_{n+1} &= \widehat{\boldsymbol{y}}'_n + h_n \widehat{\boldsymbol{\Phi}}'(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, h_n) = \widehat{\boldsymbol{y}}'_n + h_n \sum_{i=1}^{s} \widehat{b}'_i \boldsymbol{g}_i \\
\boldsymbol{y}'_{n+1} &= \widehat{\boldsymbol{y}}'_n + h_n \boldsymbol{\Phi}'(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, h_n) = \widehat{\boldsymbol{y}}'_n + h_n \sum_{i=1}^{s} b'_i \boldsymbol{g}_i \\
\boldsymbol{y}'^*_{n+1} &= \widehat{\boldsymbol{y}}'_n + \sigma h_n \boldsymbol{\Phi}'^*(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, \sigma h_n) = \widehat{\boldsymbol{y}}'_n + \sigma h_n \sum_{i=1}^{s^*} b'^*_i(\sigma) \boldsymbol{g}_i \\
\widehat{\boldsymbol{y}}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n \widehat{\boldsymbol{\Phi}}(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, h_n) = \widehat{\boldsymbol{y}}_n + h_n \widehat{\boldsymbol{y}}'_n + h_n^2 \sum_{i=1}^{s} \widehat{b}_i \boldsymbol{g}_i \\
\boldsymbol{y}_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n \boldsymbol{\Phi}(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, h_n) = \widehat{\boldsymbol{y}}_n + h_n \widehat{\boldsymbol{y}}'_n + h_n^2 \sum_{i=1}^{s} b_i \boldsymbol{g}_i \\
\boldsymbol{y}^*_{n+1} &= \widehat{\boldsymbol{y}}_n + h_n \boldsymbol{\Phi}^*(x_n, \boldsymbol{y}_n, \boldsymbol{y}'_n, \sigma h_n) = \widehat{\boldsymbol{y}}_n + \sigma h_n \widehat{\boldsymbol{y}}'_n + \sigma^2 h_n^2 \sum_{i=1}^{s^*} b^*_i(\sigma) \boldsymbol{g}_i
\end{aligned}
$$

$$
\boldsymbol{g}_i = \boldsymbol{f}\left(x_n + c_i h_n, \widehat{\boldsymbol{y}}_n + c_i h_n \widehat{\boldsymbol{y}}'_n + h_n^2 \sum_{j=1}^{i-1} a_{ij} \boldsymbol{g}_j \right), \ i = 1, \ldots, s^*
$$

$$
\begin{array}{c|ccccccccc}
0 & & & & & & & & & \\
c_2 & a_{21} & & & & & & & & \\
c_3 & a_{31} & a_{32} & & & & & & & \\
\vdots & \vdots & \vdots & \ddots & & & & & & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & & & & & \\
\hline
c_{s+1} & a_{s+1,1} & a_{s+1,2} & \cdots & a_{s+1,s-1} & a_{s+1,s} & & & & \\
c_{s+2} & a_{s+2,1} & a_{s+2,2} & \cdots & a_{s+2,s-1} & a_{s+2,s} & a_{s+2,s+1} & & & \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & & \\
c_{s^*} & a_{s^*,1} & a_{s^*,2} & \cdots & a_{s^*,s-1} & a_{s^*,s} & a_{s^*,s+1} & \cdots & a_{s^*,s^*-1} & \\
\hline
 & \widehat{b}_1 & \widehat{b}_2 & \cdots & \widehat{b}_{s-1} & \widehat{b}_s & 0 & \cdots & 0 & 0 \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s & 0 & \cdots & 0 & 0 \\
\hline
 & b^*_1 & b^*_2 & \cdots & b^*_{s-1} & b^*_s & b^*_{s+1} & \cdots & b^*_{s^*-1} & b^*_{s^*} \\
\hline
 & \widehat{b}'_1 & \widehat{b}'_2 & \cdots & \widehat{b}'_{s-1} & \widehat{b}'_s & 0 & \cdots & 0 & 0 \\
\hline
 & b'_1 & b'_2 & \cdots & b'_{s-1} & b'_s & 0 & \cdots & 0 & 0 \\
\hline
 & b'^*_1 & b'^*_2 & \cdots & b'^*_{s-1} & b'^*_s & b'^*_{s+1} & \cdots & b'^*_{s^*-1} & b'^*_{s^*}
\end{array}
$$

Figure 2.2: The general RKN triple and Butcher tableau.

where, as in the RK case, $\tau^{(i)}_{l^n_{i-1}}$ *and* $\tau'^{(i)}_{l^n_i}$ *are the only coefficients remaining in the local truncation error expansion for the* y *and* y' *solutions, respectively (see Section 4.5.1).*

This definition can naturally be applied to any of the processes of a triple formula. In the case of a dense process, the continuous real negative stability limit becomes the function $S^*_R(\sigma)$, by replacing s by s^*, $\tau^{(i)}_{l^n_{i-1}}$ by $\tau^{*(i)}_{l^n_{i-1}}$, and $\tau'^{(i)}_{l^n_i}$ by $\tau'^{*(i)}_{l^n_i}$ above.

2.7.2 Continuity of dense RKN processes

Following Dormand and Prince [15], for C^0 continuous solutions on both $\boldsymbol{y}^*_{n+\sigma}$ and $\boldsymbol{y}'^*_{n+\sigma}$, it is required that

$$b^*_i|_{\sigma=1} = \widehat{b}_i, \text{ and } b'^*_i|_{\sigma=1} = \widehat{b}'_i, \; i = 1, \ldots, s^*,$$

additionally for C^1 continuity

$$2\widehat{b}_i + \left.\frac{db^*_i}{d\sigma}\right|_{\sigma=1} = \widehat{b}'_i, \; b'^*_i|_{\sigma=0} = \delta_{i1}, \text{ and } \widehat{b}'_i + \left.\frac{db'^*_i}{d\sigma}\right|_{\sigma=1} = \delta_{is^r}, \; i = 1, \ldots, s^*,$$

where δ_{ij} is defined as in (2.22), and stage s^r is reusable (again, $s^r = s$ for FSAL). For RKN, C^2 continuity of the $\boldsymbol{y}^*_{n+\sigma}$ solution is possible by satisfying the following extra conditions:

$$b^*_i|_{\sigma=0} = \frac{\delta_{i1}}{2}, \text{ and } 2\widehat{b}_i + 4\left.\frac{db^*_i}{d\sigma}\right|_{\sigma=1} + \left.\frac{d^2 b^*_i}{d\sigma^2}\right|_{\sigma=1} = \delta_{is^r}, \; i = 1, \ldots, s^*.$$

Chapter 3

RK-AID: an RK(N) analysis and development package

3.1 Introduction

RK-AID is a software package written to aid development and analysis of RK and special RKN processes. In the following sections only the central algorithms will be discussed in detail, further abilities of the code being introduced when needed. A full description of the entry syntax, general usage, and a complete set of commands is included in Appendix A.

3.2 Generation of truncation error terms

The heart of RK-AID is its comprehensive error term analysis, via the ability to generate RK(N) truncation error terms algorithmically to *arbitrary* order owing to their relationship with the geometrical graph structures of topology known as *trees*[1] (Harris [11]). At present it is enough to know that each τ is unique to a particular tree and that the trees themselves are available. As such, conversion of a given tree to the expression for a single τ will be detailed with no regard as yet for *which* τ it is. The fundamental elements of this procedure (in the RK case) were motivated by the PASCAL code of Prince [12].

[1]From the class of rooted, non-ordered, multifurcating trees.

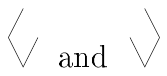
3.2.1 Trees to τ's

Each tree holds all the information necessary for generation of its related τ in a single recursive pass from the root upwards. Following Butcher [9] (with a slight modification to accommodate the special RKN case), a *general* (discrete, implicit) truncation error coefficient, corresponding to a single tree, is defined as:

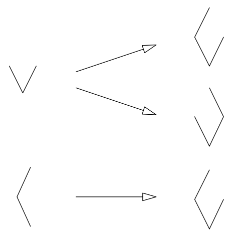
$$\frac{\alpha}{n!}\left(\gamma\sum_{i=1}^{s}\Phi_i^b - \frac{1}{m}\right). \tag{3.1}$$

Here, n is the number of nodes in the tree, γ and α are related to the tree only, and m and the Φ_i^b's are specific to the RK or RKN case, with the superscript on the Φ_i symbol denoting the particular set of weights involved (b_i in this case). Note that other representations are possible by using a different basis for the elementary differentials, for example, see Albrecht [17], [18].

The α factor reflects the degree of isomorphism of the tree, and therefore conveys the number of equivalent labellings of the non-ordered form. For example,



may be derived from trees with 3 nodes by addition of a new branch to each node as follows:



These 3 topologically equivalent occurrences of the same tree are represented by an α value of 3.

Calculation of α and γ (Butcher [9])

For a tree with n nodes, $t^{(n)}$, α may be calculated recursively from the α_i's of the N $(< n)$ sub-trees attached to the root, by the following relation:

$$\alpha = \frac{\alpha_1 \alpha_2 \ldots \alpha_N}{r_1! r_2! \ldots r_M!} \times \frac{(n-1)!}{n_1! n_2! \ldots n_N!}, \tag{3.2}$$

where n_i is the number of nodes in sub-tree i, and r_j represents the number of repetitions (at the root) of the distinct sub-tree j, M $(\leq N)$ being the number of distinct sub-trees.

The γ quantity is simply a product of n, the number of nodes in the tree, and the γ_i's for each of the N sub-trees:

$$\gamma = n \times \gamma_1 \times \gamma_2 \times \ldots \times \gamma_N, \tag{3.3}$$

where γ for the $n = 1$ $(N = 0)$ tree '$\cdot$' is defined as 1.

The Φ_i^b's hold the method dependent part of the error coefficient, depending on the process parameters (and RK or RKN type). These quantities are, again, easily obtained algorithmically from the trees in a recursive manner. The case for RK τ's is fundamental here, the RKN forms being closely related but requiring separate consideration. As such, the RK case will be described first, following Prince [12]. A basic notation is assumed throughout, the external method weights being represented by b_i's for both RK and RKN.

Determination of the Φ_i^b's for the RK case

Firstly, nodes of the tree are labelled in a left-centre-right manner up the tree beginning with i for the root, then placing j on the next node up to the left, k again up and to the left if the node exists, if not, the k goes on the first node up the sub-tree to the right of the one containing the j. All other nodes are labelled accordingly (see the fourth column of Tables 3.1 and 3.2).

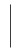
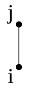
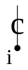
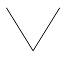
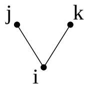
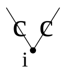
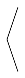
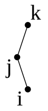
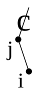
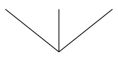
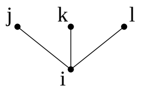
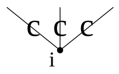
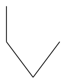
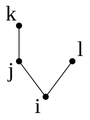
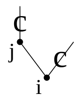
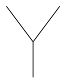
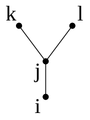
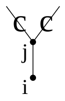
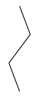
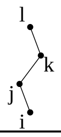
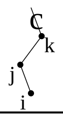
t	γ	α	Φ_i^b			
	1	1		b_i		—
	2	1		b_ia_{ij}		b_ic_i
	3	1		$b_ia_{ij}a_{ik}$		$b_ic_i^2$
	6	1		$b_ia_{ij}a_{jk}$		$b_ia_{ij}c_j$
	4	1		$b_ia_{ij}a_{ik}a_{il}$		$b_ic_i^3$
	8	3		$b_ia_{ij}a_{jk}a_{il}$		$b_ia_{ij}c_jc_i$
	12	1		$b_ia_{ij}a_{jk}a_{jl}$		$b_ia_{ij}c_j^2$
	24	1		$b_ia_{ij}a_{jk}a_{kl}$		$b_ia_{ij}a_{jk}c_k$

Table 3.1: Trees of up to 4 nodes with γ, α and RK Φ_i^b's.

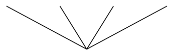
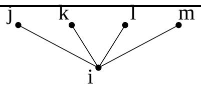
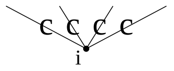
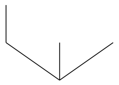
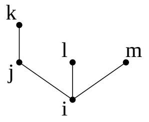
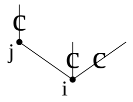
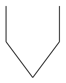
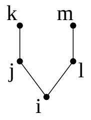
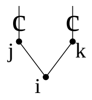
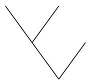
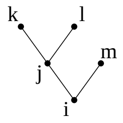
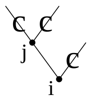
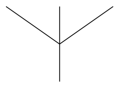
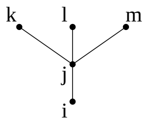
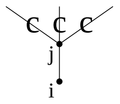
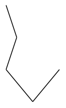
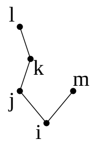
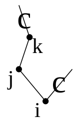
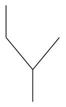
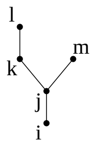
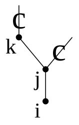
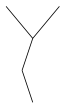
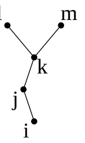
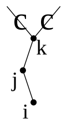
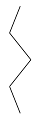
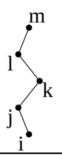
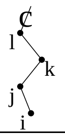
t	γ	α	Φ_i^b			
	5	1		$b_i a_{ij} a_{ik} a_{il} a_{im}$		$b_i c_i^4$
	10	6		$b_i a_{ij} a_{jk} a_{il} a_{im}$		$b_i a_{ij} c_j c_i^2$
	20	3		$b_i a_{ij} a_{jk} a_{il} a_{lm}$		$b_i a_{ij} c_j a_{ik} c_k$
	15	4		$b_i a_{ij} a_{jk} a_{jl} a_{im}$		$b_i a_{ij} c_j^2 c_i$
	20	1		$b_i a_{ij} a_{jk} a_{jl} a_{jm}$		$b_i a_{ij} c_j^3$
	30	4		$b_i a_{ij} a_{jk} a_{kl} a_{im}$		$b_i a_{ij} a_{jk} c_k c_i$
	40	3		$b_i a_{ij} a_{jk} a_{kl} a_{jm}$		$b_i a_{ij} a_{jk} c_k c_j$
	60	1		$b_i a_{ij} a_{jk} a_{kl} a_{km}$		$b_i a_{ij} a_{jk} c_k^2$
	120	1		$b_i a_{ij} a_{jk} a_{kl} a_{lm}$		$b_i a_{ij} a_{jk} a_{kl} c_l$

Table 3.2: Trees of 5 nodes with γ, α and RK Φ_i^b's.

All Φ_i^b's contain a b_i corresponding to the tree root. The $n = 1$ tree thus gives

$$\Phi_i^b = b_i.$$

Each branch of the tree then represents a double subscripted 'a' parameter linking the indices on the nodes at either end, implying a summation of the lexicographically highest index over the s stages of the process. For $n = 2$,

$$\Phi_i^b = b_i \sum_{j=1}^{s} a_{ij}.$$

The $n = 3$ case contains two trees giving

$$\Phi_i^b = b_i \sum_{j=1}^{s} a_{ij} \sum_{k=1}^{s} a_{ik} \text{ and } \Phi_i^b = b_i \sum_{j=1}^{s} a_{ij} \sum_{k=1}^{s} a_{jk},$$

respectively. These cases and those for $n = 4$ and $n = 5$ are shown in column four of Tables 3.1 and 3.2, where the summations are implied by presence of the associated indices.

By application of the row-sum conditions

$$\left(\overset{\mathrm{j}}{\underset{\mathrm{i}}{\vert}}\right) \sum_{j=1}^{s} a_{ij} = c_i \left(\underset{\mathrm{i}}{\not{\mathrm{c}}}\right) \tag{3.4}$$

to the expressions in column four of Tables 3.1 and 3.2, the usual form of representation is achieved as illustrated by column five. This process of summing over end nodes effectively *prunes* the tree skeleton, and it is now convenient to think of a 'c' parameter as being a feature of the subscripted node, and not part of the tree structure itself (as indicated by the tree diagrams).

Determination of the Φ_i^b's for the RKN case

Similar to the RK case, now following the notation adopted by Dormand and Prince (in [19] and [20], for example), labelling of the tree is done by attaching indices to each

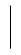
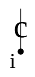
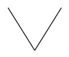
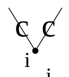
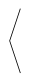
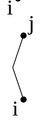
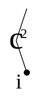
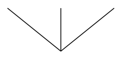
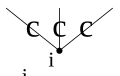
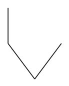
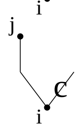
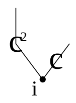
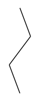
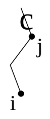
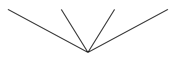
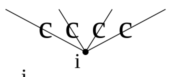
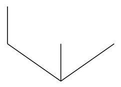
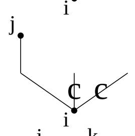
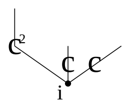
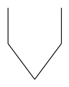
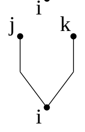
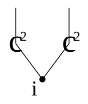
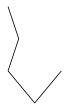
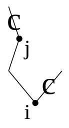
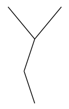
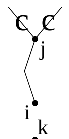
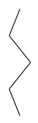
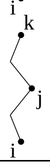
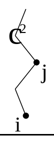
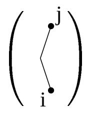
t	γ	α	Φ_i^b			
	1	1	$\bullet\, i$	b_i	—	
	2	1		$b_i c_i$	—	
	3	1		$b_i c_i^2$	—	
	6	1		$b_i a_{ij}$		$\frac{1}{2} b_i c_i^2$
	4	1		$b_i c_i^3$	—	
	8	3		$b_i a_{ij} c_i$		$\frac{1}{2} b_i c_i^3$
	24	1		$b_i a_{ij} c_j$	—	
	5	1		$b_i c_i^4$	—	
	10	6		$b_i a_{ij} c_i^2$		$\frac{1}{2} b_i c_i^4$
	20	3		$b_i a_{ij} a_{ik}$		$\frac{1}{4} b_i c_i^4$
	30	4		$b_i a_{ij} c_j c_i$	—	
	60	1		$b_i a_{ij} c_j^2$	—	
	120	1		$b_i a_{ij} a_{jk}$		$\frac{1}{2} b_i a_{ij} c_i^2$

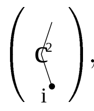
Table 3.3: RKN trees of up to 5 nodes with γ, α, and Φ_i^b's.

node adapting the same left-centre-right convention. Here, the effect of the special Nyström form is to skip odd 'levels' of the tree (calling the root level 0) during the recursive labelling process used previously — see column four of Table 3.3. Calling a node on an odd level of the tree an *odd node* (and similar for an even level) this relies on the fact that no Nyström truncation error expression is derived from a tree with multi-furcating odd nodes, i.e., a tree in which an odd node joins to *more than one* even node above. It is therefore assumed that no such tree will be considered.

Again, each Φ_i^b contains a b_i corresponding to the tree root, and branches linking labelled nodes become double subscripted 'a' parameters. Any odd node that ends its sub-tree is here a 'c' parameter associated with the its subscripted node, as in the RK case after row-summing. Here, 'a' parameters that end their sub-tree (ending on an even node) are summed out using the RKN row-sum conditions



$$\sum_{j=1}^{s} a_{ij} = \frac{c_i^2}{2} \qquad (3.5)$$



and the 'a' becomes '$\frac{1}{2}c^2$' on the node below. This representation is shown in column five of Table 3.3.

3.2.2 Generation of trees and specific τ's

Scoins [21] developed an algorithm, NORTS, for generation of the trees needed here (with a specified number of nodes and no repeat of isomorphisms). This is done using a 'canonical' form for providing a natural ordering of the trees, the term canonical indicating that each sub-tree is 'left-heavy' *recursively* up the tree, 'heaviness' being gauged by the height of the tree section (not the number of nodes). The sequence in which the trees are generated is such that, just as sub-trees must be higher to the left of each node of a tree, taller trees occur later in the generated sequence. Trees of the same height are generated in order of their lowest sub-tree, recursively. For example, the $n = 1$ tree has zero height (the first tree in any *natural* sequence), and the first tree for each of $n > 1$, i.e., the lowest tree, has height one, etc.

Now, associating any given tree with a specific τ for both RK and RKN methods, and also defining the quantity m from expression (3.1), the RK case is simple — p^{th} order τ's are generated with a one-to-one correspondence from trees with $n = p$ nodes, and $m = 1$. The Nyström case is complicated only by existence of two sets of coefficients (one for 'y' and one for 'y'') and that not all trees are relevant. The expressions for y' in the p^{th} order are similarly derived from all relevant trees with $n = p$ nodes, and $m = 1$, replacing Φ_i^b by $\Phi_i^{b'}$. Those for y in the p^{th} order are generated from all relevant trees with $n = p - 1$ nodes, and $m = p$.

It should be considered at this point that the approach used here for generation of truncation error terms is not in any way unique. One distinction from alternative methods (Hosea [22] for example) is that each τ is generated in isolation despite an obvious amount of recursive repetition with terms from previous orders. The advantage seen in the approach of RK-AID is that *any* individual terms from *any* order can be examined without the need to generate *and store* quantities from all preceding orders. This may be a slower approach in some circumstances, but does not make any more demands on storage as the order increases, and is more convenient for most of the analysis featured here.

A list of τ's up to sixth-order, for both RK and RKN, is given in Tables 3.1 and 3.2. It should be noted that for the RKN case, application of the row-sum conditions results in replication of τ's to within a scalar multiple, and are thus marked with an asterisk ($*$).

3.2.3 Extended row-sum and 'root' substitutions

In Butcher [8], it was first noted (for the RK case) that the truncation error basis can be written in an equivalent form whereby all terms derived from trees with n nodes of height > 1, i.e., all but the first, take the form

$$\frac{\alpha\gamma}{n!}\sum_{i=1}^{s}\Phi_i^{(q)b} + \Theta^q, \tag{3.6}$$

$$
\begin{aligned}
\tau_1^{(1)} &= \sum_i b_i - 1 \\
\tau_1^{(2)} &= \sum_i b_i c_i - \frac{1}{2!} \\
\tau_1^{(3)} &= \tfrac{1}{2} \sum_i b_i c_i^2 - \frac{1}{3!} \\
\tau_2^{(3)} &= \sum_{ij} b_i a_{ij} c_j - \frac{1}{3!} \\
\tau_1^{(4)} &= \tfrac{1}{6} \sum_i b_i c_i^3 - \frac{1}{4!} \\
\tau_2^{(4)} &= \sum_{ij} b_i c_i a_{ij} c_j - \frac{3}{4!} \\
\tau_3^{(4)} &= \tfrac{1}{2} \sum_{ij} b_i a_{ij} c_j^2 - \frac{1}{4!} \\
\tau_4^{(4)} &= \sum_{ijk} b_i a_{ij} a_{jk} c_k - \frac{1}{4!} \\
\tau_1^{(5)} &= \tfrac{1}{24} \sum_i b_i c_i^4 - \frac{1}{5!} \\
\tau_2^{(5)} &= \tfrac{1}{2} \sum_{ij} b_i c_i^2 a_{ij} c_j - \frac{6}{5!} \\
\tau_3^{(5)} &= \tfrac{1}{2} \sum_{ijk} b_i a_{ij} c_j a_{ik} c_k - \frac{3}{5!} \\
\tau_4^{(5)} &= \tfrac{1}{2} \sum_{ij} b_i c_i a_{ij} c_j^2 - \frac{4}{5!} \\
\tau_5^{(5)} &= \tfrac{1}{6} \sum_{ij} b_i a_{ij} c_j^3 - \frac{1}{5!} \\
\tau_6^{(5)} &= \sum_{ijk} b_i c_i a_{ij} a_{jk} c_k - \frac{4}{5!} \\
\tau_7^{(5)} &= \sum_{ijk} b_i a_{ij} c_j a_{jk} c_k - \frac{3}{5!} \\
\tau_8^{(5)} &= \tfrac{1}{2} \sum_{ijk} b_i a_{ij} a_{jk} c_k^2 - \frac{1}{5!} \\
\tau_9^{(5)} &= \sum_{ijkl} b_i a_{ij} a_{jk} a_{kl} c_l - \frac{1}{5!} \\
\tau_1^{(6)} &= \tfrac{1}{120} \sum_i b_i c_i^5 - \frac{1}{6!} \\
\tau_2^{(6)} &= \tfrac{1}{6} \sum_{ij} b_i c_i^3 a_{ij} c_j - \frac{10}{6!}
\end{aligned}
$$

$$
\begin{aligned}
\tau_3^{(6)} &= \tfrac{1}{2} \sum_{ijk} b_i c_i a_{ij} c_j a_{ik} c_k - \frac{15}{6!} \\
\tau_4^{(6)} &= \tfrac{1}{4} \sum_{ij} b_i c_i^2 a_{ij} c_j^2 - \frac{10}{6!} \\
\tau_5^{(6)} &= \tfrac{1}{2} \sum_{ijk} b_i a_{ij} c_j^2 a_{ik} c_k - \frac{10}{6!} \\
\tau_6^{(6)} &= \tfrac{1}{6} \sum_{ij} b_i c_i a_{ij} c_j^3 - \frac{5}{6!} \\
\tau_7^{(6)} &= \tfrac{1}{24} \sum_{ij} b_i a_{ij} c_j^4 - \frac{1}{6!} \\
\tau_8^{(6)} &= \tfrac{1}{2} \sum_{ijk} b_i c_i^2 a_{ij} a_{jk} c_k - \frac{10}{6!} \\
\tau_9^{(6)} &= \sum_{ijkl} b_i a_{ij} a_{jk} c_k a_{il} c_l - \frac{10}{6!} \\
\tau_{10}^{(6)} &= \sum_{ijk} b_i c_i a_{ij} c_j a_{jk} c_k - \frac{15}{6!} \\
\tau_{11}^{(6)} &= \tfrac{1}{2} \sum_{ijk} b_i a_{ij} c_j^2 a_{jk} c_k - \frac{6}{6!} \\
\tau_{12}^{(6)} &= \tfrac{1}{2} \sum_{ijkl} b_i a_{ij} a_{jk} c_k a_{jl} c_l - \frac{3}{6!} \\
\tau_{13}^{(6)} &= \tfrac{1}{2} \sum_{ijk} b_i c_i a_{ij} a_{jk} c_k^2 - \frac{5}{6!} \\
\tau_{14}^{(6)} &= \tfrac{1}{2} \sum_{ijk} b_i a_{ij} c_j a_{jk} c_k^2 - \frac{4}{6!} \\
\tau_{15}^{(6)} &= \tfrac{1}{6} \sum_{ijk} b_i a_{ij} a_{jk} c_k^3 - \frac{1}{6!} \\
\tau_{16}^{(6)} &= \sum_{ijkl} b_i c_i a_{ij} a_{jk} a_{kl} c_l - \frac{5}{6!} \\
\tau_{17}^{(6)} &= \sum_{ijkl} b_i a_{ij} c_j a_{jk} a_{kl} c_l - \frac{4}{6!} \\
\tau_{18}^{(6)} &= \sum_{ijkl} b_i a_{ij} a_{jk} c_k a_{kl} c_l - \frac{3}{6!} \\
\tau_{19}^{(6)} &= \tfrac{1}{2} \sum_{ijkl} b_i a_{ij} a_{jk} a_{kl} c_l^2 - \frac{1}{6!} \\
\tau_{20}^{(6)} &= \sum_{ijklm} b_i a_{ij} a_{jk} a_{kl} a_{lm} c_m - \frac{1}{6!}
\end{aligned}
$$

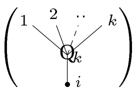
Figure 3.1: RK truncation error coefficients up to sixth-order.

$$\tau_1^{\prime(1)} = \sum_i b_i' - 1$$
$$\tau_1^{\prime(2)} = \sum_i b_i' c_i - \frac{1}{2!}$$
$$\tau_1^{\prime(3)} = \tfrac{1}{2}\sum_i b_i' c_i^2 - \frac{1}{3!}$$
$$\ast \quad \tau_2^{\prime(3)} = \tfrac{1}{2}\sum_i b_i' c_i^2 - \frac{1}{3!}$$
$$\tau_1^{\prime(4)} = \tfrac{1}{6}\sum_i b_i' c_i^3 - \frac{1}{4!}$$
$$\ast \quad \tau_2^{\prime(4)} = \tfrac{1}{2}\sum_i b_i' c_i^3 - \frac{3}{4!}$$
$$\tau_3^{\prime(4)} = \sum_{ij} b_i' a_{ij} c_j - \frac{1}{4!}$$
$$\tau_1^{\prime(5)} = \tfrac{1}{24}\sum_i b_i' c_i^4 - \frac{1}{5!}$$
$$\ast \quad \tau_2^{\prime(5)} = \tfrac{1}{4}\sum_i b_i' c_i^4 - \frac{6}{5!}$$
$$\ast \quad \tau_3^{\prime(5)} = \tfrac{1}{8}\sum_i b_i' c_i^4 - \frac{3}{5!}$$
$$\tau_4^{\prime(5)} = \sum_{ij} b_i' c_i a_{ij} c_j - \frac{4}{5!}$$
$$\tau_5^{\prime(5)} = \tfrac{1}{2}\sum_{ij} b_i' a_{ij} c_j^2 - \frac{1}{5!}$$
$$\ast \quad \tau_6^{\prime(5)} = \tfrac{1}{2}\sum_{ij} b_i' a_{ij} c_j^2 - \frac{1}{5!}$$
$$\tau_1^{\prime(6)} = \tfrac{1}{120}\sum_i b_i' c_i^5 - \frac{1}{6!}$$
$$\ast \quad \tau_2^{\prime(6)} = \tfrac{1}{12}\sum_i b_i' c_i^5 - \frac{10}{6!}$$
$$\ast \quad \tau_3^{\prime(6)} = \tfrac{1}{8}\sum_i b_i' c_i^5 - \frac{15}{6!}$$
$$\tau_4^{\prime(6)} = \tfrac{1}{2}\sum_{ij} b_i' c_i^2 a_{ij} c_j - \frac{10}{6!}$$
$$\ast \quad \tau_5^{\prime(6)} = \tfrac{1}{2}\sum_{ij} b_i' c_i^2 a_{ij} c_j - \frac{10}{6!}$$
$$\tau_6^{\prime(6)} = \tfrac{1}{2}\sum_{ij} b_i' c_i a_{ij} c_j^2 - \frac{5}{6!}$$
$$\tau_7^{\prime(6)} = \tfrac{1}{6}\sum_{ij} b_i' a_{ij} c_j^3 - \frac{1}{6!}$$
$$\ast \quad \tau_8^{\prime(6)} = \tfrac{1}{2}\sum_{ij} b_i' c_i a_{ij} c_j^2 - \frac{5}{6!}$$
$$\ast \quad \tau_9^{\prime(6)} = \tfrac{1}{2}\sum_{ij} b_i' a_{ij} c_j^3 - \frac{3}{6!}$$
$$\tau_{10}^{\prime(6)} = \sum_{ijk} b_i' a_{ij} a_{jk} c_k - \frac{1}{6!}$$

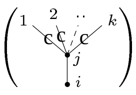
$$\tau_1^{(2)} = \sum_i b_i - \frac{1}{2!}$$
$$\tau_1^{(3)} = \sum_i b_i c_i - \frac{1}{3!}$$
$$\tau_1^{(4)} = \tfrac{1}{2}\sum_i b_i c_i^2 - \frac{1}{4!}$$
$$\ast \quad \tau_2^{(4)} = \tfrac{1}{2}\sum_i b_i c_i^2 - \frac{1}{4!}$$
$$\tau_1^{(5)} = \tfrac{1}{6}\sum_i b_i c_i^3 - \frac{1}{5!}$$
$$\ast \quad \tau_2^{(5)} = \tfrac{1}{2}\sum_i b_i c_i^3 - \frac{3}{5!}$$
$$\tau_3^{(5)} = \sum_{ij} b_i a_{ij} c_j - \frac{1}{5!}$$
$$\tau_1^{(6)} = \tfrac{1}{24}\sum_i b_i c_i^4 - \frac{1}{6!}$$
$$\ast \quad \tau_2^{(6)} = \tfrac{1}{4}\sum_i b_i c_i^4 - \frac{6}{6!}$$
$$\ast \quad \tau_3^{(6)} = \tfrac{1}{8}\sum_i b_i c_i^4 - \frac{3}{6!}$$
$$\tau_4^{(6)} = \sum_{ij} b_i c_i a_{ij} c_j - \frac{4}{6!}$$
$$\tau_5^{(6)} = \tfrac{1}{2}\sum_{ij} b_i a_{ij} c_j^2 - \frac{1}{6!}$$
$$\ast \quad \tau_6^{(6)} = \tfrac{1}{2}\sum_{ij} b_i a_{ij} c_j^2 - \frac{1}{6!}$$

Figure 3.2: RKN truncation error coefficients up to sixth-order.

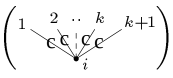
where Θ^q is a linear combination of τ's also derived from trees with n nodes[2]. The $\Phi_i^{(q)b}$'s here are modifications of the Φ_i^b's in the standard representation, and make use of the following additional RK and RKN method parameters, respectively:



$$Q_{ik} =$$

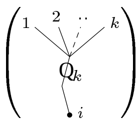


$$\sum_{j=1}^{s} a_{ij} c_j^k - \frac{c_i^{k+1}}{k+1}$$

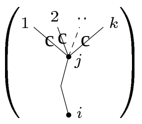


$$, \qquad (3.7)$$

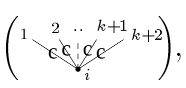
and



$$Q_{ik} =$$



$$\sum_{j=1}^{s} a_{ij} c_j^k - \frac{c_i^{k+2}}{(k+2)(k+1)}$$



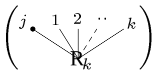
$$, \qquad (3.8)$$

which contain the row-sum conditions (3.4) and (3.5) — the $Q_{i0} = 0$ cases here. By making the substitution

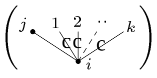
$$\sum_{j=1}^{s} a_{ij} c_j^k \to Q_{ik}$$

at least once wherever it occurs in a given Φ_i^b (for any $k > 0$), an index is removed from the labelled tree by each substitution, and hence serves to simplify the representation of each term. Note that, except for RKN τ's already dependent after row-summing (and the first from any order), every τ will contain *at least one* possible substitution. Where more than one is available for a specific Φ_i^b, there exists a set of alternative representations for the τ, with each option giving different $\Phi_i^{(q)b}$ and Θ^q permutations corresponding to the choice of substitutions made.

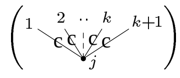
In [8], Butcher further prompted consideration of the following RK parameters:



$$R_{kj} =$$



$$\sum_{i=1}^{s} b_i c_i^k a_{ij} - (\bullet j)\,\frac{b_j}{k+1} + \frac{b_j c_j^{k+1}}{k+1}$$



$$. \qquad (3.9)$$

The *root* substitution

$$\sum_{i=1}^{s} b_i c_i^k a_{ij} \to R_{kj} \qquad (3.10)$$

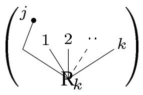
can be made once only for relevant Φ_i^b's, resulting in a form denoted by Φ_i^r and Θ^r,

[2]In the RKN case, some τ's will already be a scalar multiple of another after application of the row-sum conditions (3.5), in this case, all $\Phi_i^{(q)b}$'s are zero.

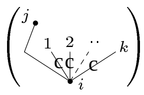
i.e.,

$$\frac{\alpha\gamma}{n!}\sum_{i=1}^{s}\Phi_i^r + \Theta^r, \tag{3.11}$$

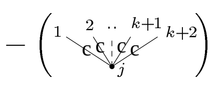
where the lack of n as a subscript refers to a linear combination of τ's from the n^{th} *and previous* tree orders. For RKN trees (see Hairer [23]), such a substitution is only relevant for y' equations, the parameters in this case taking the form



$$R_{kj} = $$



$$\sum_{i=1}^{s} b_i' c_i^k a_{ij} + (\dot{c}_j)\,\frac{b_j' c_j}{k+1}$$



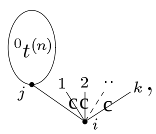
$$\frac{b_j' c_j^{k+2}}{(k+2)(k+1)} - \frac{b_j'}{k+2}\,(\bullet j)\,. \tag{3.12}$$

In some cases, a 'Q form' and 'R form' of the τ can co-exist, this representation being denoted by $\Phi_i^{(q)r}$ and Θ^{qr}, for a particular set of 'Q' substitutions[3]. Tables 3.4–3.6 illustrate the substitution process above for the RK case up to sixth-order, and Tables 3.7–3.8 the RKN case up to seventh-order in y' (again, the external method weights are represented as b_i, acknowledging that the forms containing them in columns one and two may relate to either y or y' terms). Note that in the RKN case only *independent* trees are shown, i.e., those trees which have not been row-summed and so give dependent truncation error terms. Also, the 'R form' expressions shown have had their summation indices lowered by one, i.e., $j \to i$, $k \to j$, etc., such that i is the first index as usual. The choice of $\Phi_i^{(q)b}$ and $\Phi_i^{(q)r}$ is that in which *all* possible 'Q' substitutions have been made.

3.2.4 Determination of the RK τ combinations Θ^r

Consider the following RK tree/τ pair:

$${}^1t^{(n+k+1)} =$$



$$\text{and } {}^1\tau^{(n+k+1)} = \frac{\alpha_1}{(n+k+1)!}\left(\gamma_1\sum_{i,j=1}^{s} b_i c_i^k a_{ij}\,{}^0\overline{\Phi}_j - 1\right),$$

[3]In the absence of any 'Q' substitutions, the 'QR form' is defined to be equal to the 'R form'.

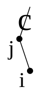
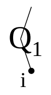
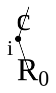
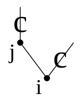
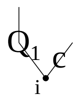
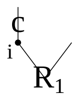
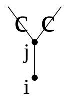
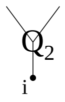
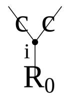
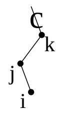
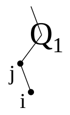
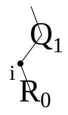
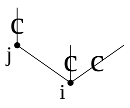
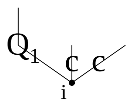
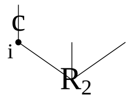
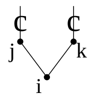
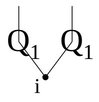
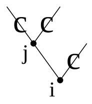
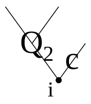
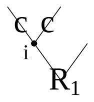
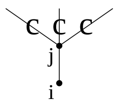
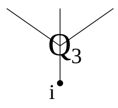
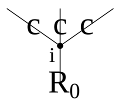
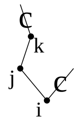
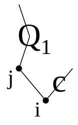
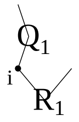
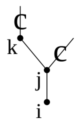
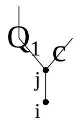
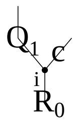
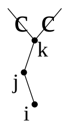
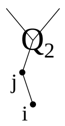
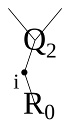
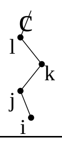
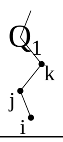
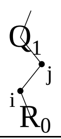
Φ_i^b		$\Phi_i^{(q)b}$		$\Phi_i^{(q)r}$	
	$b_i a_{ij} c_j$		$b_i Q_{i1}$		$R_{0i} c_i$
	$b_i c_i a_{ij} c_j$		$b_i c_i Q_{i1}$		$R_{1i} c_i$
	$b_i a_{ij} c_j^2$		$b_i Q_{i2}$		$R_{0i} c_i^2$
	$b_i a_{ij} a_{jk} c_k$		$b_i a_{ij} Q_{j1}$		$R_{0i} Q_{i1}$
	$b_i c_i^2 a_{ij} c_j$		$b_i c_i^2 Q_{i1}$		$R_{2i} c_i$
	$b_i a_{ij} c_j a_{ik} c_k$		$b_i Q_{i1}^2$		—
	$b_i c_i a_{ij} c_j^2$		$b_i c_i Q_{i2}$		$R_{1i} c_i^2$
	$b_i a_{ij} c_j^3$		$b_i Q_{i3}$		$R_{0i} c_i^3$
	$b_i c_i a_{ij} a_{jk} c_k$		$b_i c_i a_{ij} Q_{j1}$		$R_{1i} Q_{i1}$
	$b_i a_{ij} c_j a_{jk} c_k$		$b_i a_{ij} c_j Q_{j1}$		$R_{0i} c_i Q_{i1}$
	$b_i a_{ij} a_{jk} c_k^2$		$b_i a_{ij} Q_{j2}$		$R_{0i} Q_{i2}$
	$b_i a_{ij} a_{jk} a_{kl} c_l$		$b_i a_{ij} a_{jk} Q_{k1}$		$R_{0i} a_{ij} Q_{j1}$

Table 3.4: Height > 1 trees of up to 5 nodes with RK Φ_i^b, $\Phi_i^{(q)b}$ and $\Phi_i^{(q)r}$'s.

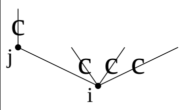
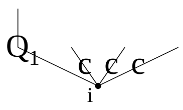
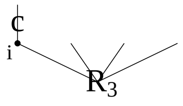
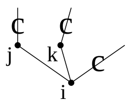
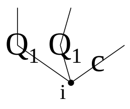
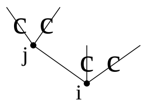
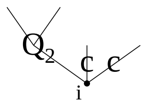
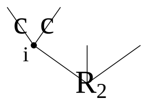
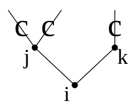
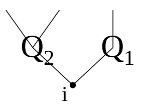
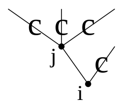
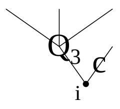
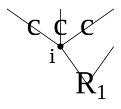
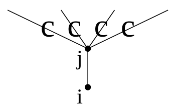
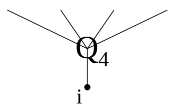
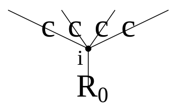
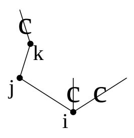
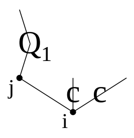
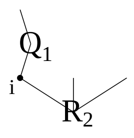
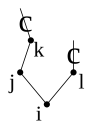
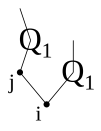
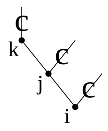
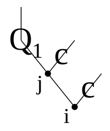
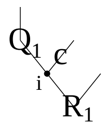
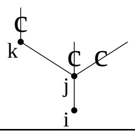
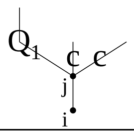
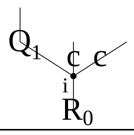
Φ_i^b		$\Phi_i^{(q)b}$		$\Phi_i^{(q)r}$	
	$b_ic_i^3a_{ij}c_j$		$b_ic_i^3Q_{i1}$		$R_{3i}c_i$
	$b_ic_ia_{ij}c_ja_{ik}c_k$		$b_ic_iQ_{i1}^2$		—
	$b_ic_i^2a_{ij}c_j^2$		$b_ic_i^2Q_{i2}$		$R_{2i}c_i^2$
	$b_ia_{ij}c_j^2a_{ik}c_k$		$b_iQ_{i2}Q_{i1}$		—
	$b_ic_ia_{ij}c_j^3$		$b_ic_iQ_{i3}$		$R_{1i}c_i^3$
	$b_ia_{ij}c_j^4$		b_iQ_{i4}		$R_{0i}c_i^4$
	$b_ic_i^2a_{ij}a_{jk}c_k$		$b_ic_i^2a_{ij}Q_{j1}$		$R_{2i}Q_{i1}$
	$b_ia_{ij}a_{jk}c_ka_{il}c_l$		$b_ia_{ij}Q_{j1}Q_{i1}$		—
	$b_ic_ia_{ij}c_ja_{jk}c_k$		$b_ic_ia_{ij}c_jQ_{j1}$		$R_{1i}c_iQ_{i1}$
	$b_ia_{ij}c_j^2a_{jk}c_k$		$b_ia_{ij}c_j^2Q_{j1}$		$R_{0i}c_i^2Q_{i1}$

Table 3.5: 6 node trees $2,\ldots,11$ with RK Φ_i^b, $\Phi_i^{(q)b}$ and $\Phi_i^{(q)r}$'s.

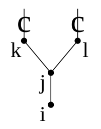
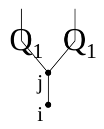
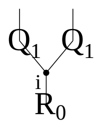
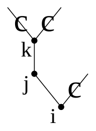
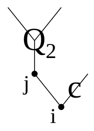
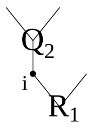
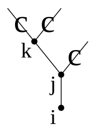
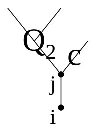
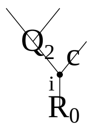
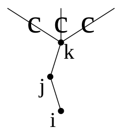
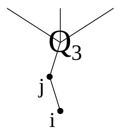
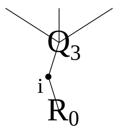
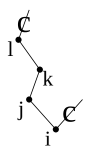
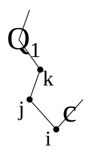
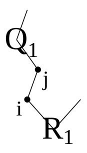
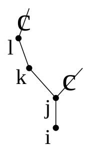
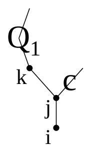
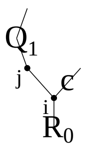
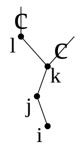
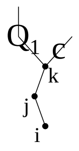
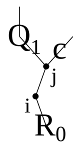
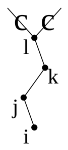
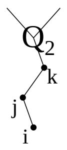
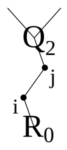
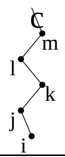
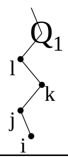
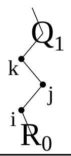
Φ_i^b	$\Phi_i^{(q)b}$	$\Phi_i^{(q)r}$
 $b_i a_{ij} a_{jk} c_k a_{jl} c_l$	 $b_i a_{ij} Q_{j1}^2$	 $R_{0i} Q_{i1}^2$
 $b_i c_i a_{ij} a_{jk} c_k^2$	 $b_i c_i a_{ij} Q_{j2}$	 $R_{1i} Q_{i2}$
 $b_i a_{ij} c_j a_{jk} c_k^2$	 $b_i a_{ij} c_j Q_{j2}$	 $R_{0i} c_i Q_{i2}$
 $b_i a_{ij} a_{jk} c_k^3$	 $b_i a_{ij} Q_{j3}$	 $R_{0i} Q_{i3}$
 $b_i c_i a_{ij} a_{jk} a_{kl} c_l$	 $b_i c_i a_{ij} a_{jk} Q_{k1}$	 $R_{1i} a_{ij} Q_{j1}$
 $b_i a_{ij} c_j a_{jk} a_{kl} c_l$	 $b_i a_{ij} c_j a_{jk} Q_{k1}$	 $R_{0i} c_i a_{ij} Q_{j1}$
 $b_i a_{ij} a_{jk} c_k a_{kl} c_l$	 $b_i a_{ij} a_{jk} c_k Q_{k1}$	 $R_{0i} a_{ij} c_j Q_{j1}$
 $b_i a_{ij} a_{jk} a_{kl} c_l^2$	 $b_i a_{ij} a_{jk} Q_{k2}$	 $R_{0i} a_{ij} Q_{j2}$
 $b_i a_{ij} a_{jk} a_{kl} a_{lm} c_m$	 $b_i a_{ij} a_{jk} a_{kl} Q_{l1}$	 $R_{0i} a_{ij} a_{jk} Q_{k1}$

Table 3.6: 6 node trees $12, \ldots, 20$ with RK Φ_i^b, $\Phi_i^{(q)b}$ and $\Phi_i^{(q)r}$'s.

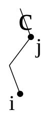
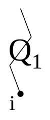
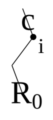
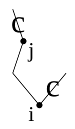
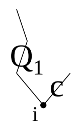
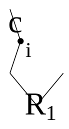
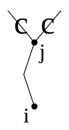
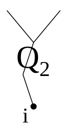
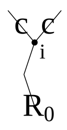
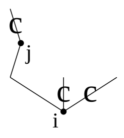
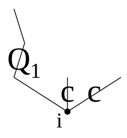
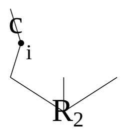
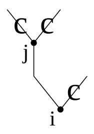
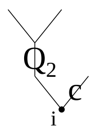
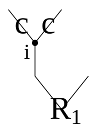
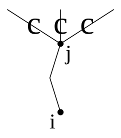
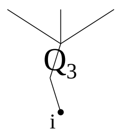
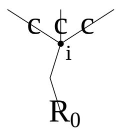
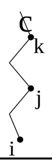
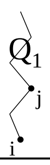
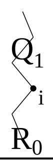
Φ_i^b	$\Phi_i^{(q)b}$	$\Phi_i^{(q)r}$
 $b_ia_{ij}c_j$	 b_iQ_{i1}	 $R_{0i}c_i$
 $b_ic_ia_{ij}c_j$	 $b_ic_iQ_{i1}$	 $R_{1i}c_i$
 $b_ia_{ij}c_j^2$	 b_iQ_{i2}	 $R_{0i}c_i^2$
 $b_ic_i^2a_{ij}c_j$	 $b_ic_i^2Q_{i1}$	 $R_{2i}c_i$
 $b_ic_ia_{ij}c_j^2$	 $b_ic_iQ_{i2}$	 $R_{1i}c_i^2$
 $b_ia_{ij}c_j^3$	 b_iQ_{i3}	 $R_{0i}c_i^3$
 $b_ia_{ij}a_{jk}c_k$	 $b_ia_{ij}Q_{j1}$	 $R_{0i}Q_{i1}$

Table 3.7: Height > 1 RKN trees of up to 6 nodes with Φ_i^b, $\Phi_i^{(q)b}$ and $\Phi_i^{(q)r}$'s.

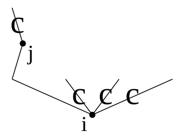
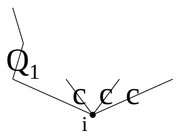
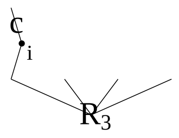
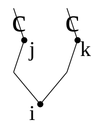
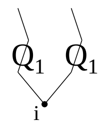
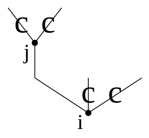
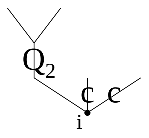
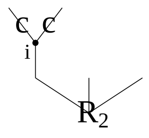
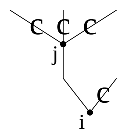
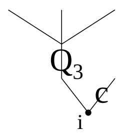
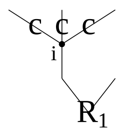
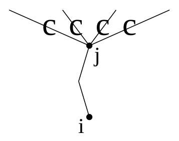
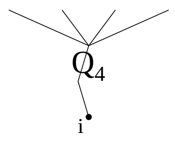
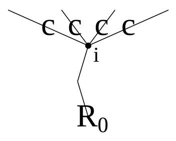
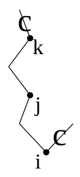
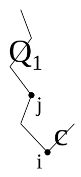
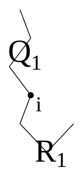
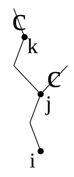
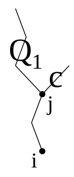
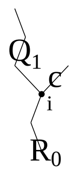
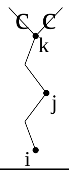
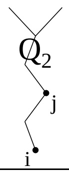
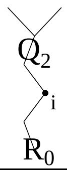
Φ_i^b		$\Phi_i^{(q)b}$		$\Phi_i^{(q)r}$	
	$b_i c_i^3 a_{ij} c_j$		$b_i c_i^3 Q_{i1}$		$R_{3i} c_i$
	$b_i a_{ij} c_j a_{ik} c_k$		$b_i Q_{i1}^2$		—
	$b_i c_i^2 a_{ij} c_j^2$		$b_i c_i^2 Q_{i2}$		$R_{2i} c_i^2$
	$b_i c_i a_{ij} c_j^3$		$b_i c_i Q_{i3}$		$R_{1i} c_i^3$
	$b_i a_{ij} c_j^4$		$b_i Q_{i4}$		$R_{0i} c_i^4$
	$b_i c_i a_{ij} a_{jk} c_k$		$b_i c_i a_{ij} Q_{j1}$		$R_{1i} Q_{i1}$
	$b_i a_{ij} c_j a_{jk} c_k$		$b_i a_{ij} c_j Q_{j1}$		$R_{0i} c_i Q_{i1}$
	$b_i a_{ij} a_{jk} c_k^2$		$b_i a_{ij} Q_{j2}$		$R_{0i} Q_{i2}$

Table 3.8: Height > 1 RKN trees of 7 nodes with Φ_i^b, $\Phi_i^{(q)b}$ and $\Phi_i^{(q)r}$'s.

where any n node tree ${}^{m}t^{(n)}$ has associated quantities α_m, γ_m, and ${}^{m}\Phi_i^b,\ i = 1, \ldots, s$, forming the truncation error term

$$ {}^{m}\tau^{(n)} = \frac{\alpha_m}{n!}\left(\gamma_m \sum_{i=1}^{s} {}^{m}\Phi_i^b - 1\right). $$

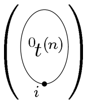
Note introduction of the quantity ${}^{m}\overline{\Phi}_i$, the bar $(\overline{\cdot})$ denoting lack of a b_i corresponding to the tree root, but otherwise is defined the same as ${}^{m}\Phi_i^b$, and therefore refers to a sub-tree. Now, expanding the j summation of ${}^{1}\tau^{(n+k+1)}$:

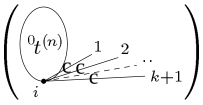
$$ \begin{aligned} {}^{1}\tau^{(n+k+1)} &= \frac{\alpha_1\gamma_1}{(n+k+1)!}\sum_{j=1}^{s}\left[\sum_{i=1}^{s} b_i c_i^k a_{ij} - \frac{b_j}{k+1} + \frac{b_j c_j^{k+1}}{k+1}\right]{}^{0}\overline{\Phi}_j - \frac{\alpha_1}{(n+k+1)!} \\ &\quad + \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!}\sum_{j=1}^{s} {}^{0}\Phi_j^b - \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!}\sum_{j=1}^{s} b_j c_j^{k+1}{}^{0}\overline{\Phi}_j \\ &= \frac{\alpha_1\gamma_1}{(n+k+1)!}\sum_{i=1}^{s} R_{ki}{}^{0}\overline{\Phi}_i + \beta^r \\ &= \frac{\alpha_1\gamma_1}{(n+k+1)!}\sum_{i=1}^{s} \Phi_i^r + \beta^r, \end{aligned} \tag{3.13} $$

where

$$ \begin{aligned} \beta^r &= \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!}\sum_{i=1}^{s} {}^{0}\Phi_i^b - \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!}\sum_{i=1}^{s} b_i c_i^{k+1}{}^{0}\overline{\Phi}_i \\ &\quad - \frac{\alpha_1}{(n+k+1)!}. \end{aligned} \tag{3.14} $$

The form of ${}^{1}\tau^{(n+k+1)}$ in (3.13) is that of (3.11) after the root substitution (3.10), where β^r needs to be represented as a linear combination of τ's, denoted by Θ^r. The τ's which will form the linear combination is clear from the summation sections in β^r, therefore writing

$\Theta^r = \lambda_0 {}^{0}\tau^{(n)} \Bigg($ 

$\Bigg) - \lambda_2 {}^{2}\tau^{(n+k+1)} \Bigg($ 

$\Bigg),$ (3.15)

where ${}^2\Phi_i^b$, relating to ${}^2\tau^{(n+k+1)}$, is given by ${}^2\Phi_i^b = b_i {}^0\overline{\Phi}_i c_i^{k+1}$. λ_0 and λ_2 can now be determined by equating β^r (3.14) to Θ^r (3.15) using the definitions of ${}^0\tau^{(n)}$ and ${}^2\tau^{(n+k+1)}$ in terms of α_0, α_2, γ_0, γ_2 and ${}^0\overline{\Phi}_i$, giving the following three conditions:

$$\begin{aligned}\frac{\lambda_0\alpha_0\gamma_0}{n!} &= \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!},\\ \frac{\lambda_2\alpha_2\gamma_2}{(n+k+1)!} &= \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!},\end{aligned}$$

and

$$\frac{\lambda_0\alpha_0}{n!} - \frac{\lambda_2\alpha_2}{(n+k+1)!} = \frac{\alpha_1}{(n+k+1)!}.$$

Solving the first two trivially,

$$\lambda_0 = \frac{\alpha_1\gamma_1 n!}{\alpha_0\gamma_0(k+1)(n+k+1)!}, \tag{3.16}$$

and

$$\lambda_2 = \frac{\alpha_1\gamma_1}{\alpha_2\gamma_2(k+1)}, \tag{3.17}$$

leaving the condition

$$\frac{\gamma_1}{k+1}\left(\frac{1}{\gamma_0} - \frac{1}{\gamma_2}\right) = 1. \tag{3.18}$$

Now, by inspection of the recursive definition of γ (3.3):

$$\gamma_0 = \frac{\gamma_1}{n+k+1}, \text{ and } \gamma_2 = \frac{\gamma_1}{n},$$

and so (3.18) is true after simple manipulation. Thus by the properties of the γ's of the three trees involved in the combination, the constant term is *absorbed* and the alternative form (3.6) is applicable. Tidying up, from the definition of α (3.2):

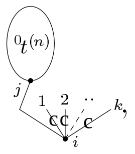
$$\alpha_0 = \frac{\alpha_1 n!k!}{(n+k)!}, \text{ and } \alpha_2 = \frac{\alpha_1 n!k!n_c!}{(n-1)!(n_c+k+1)},$$

where n_c is the number of single node principal sub-trees[4] of ${}^0t^{(n)}$, substituting into the expressions (3.16) and (3.17) yields

$$\lambda_0 = \frac{1}{(k+1)!}, \text{ and } \lambda_2 = \frac{(k+n_c+1)!}{n_c!(k+1)!}.$$

3.2.5 Determination of the RKN τ' combinations Θ^r

In an analogous manner to the previous section, the RKN Θ^r combinations may be derived by consideration of the tree/τ' pair

$${}^1t^{(n+k+2)} =$$


$$\text{and } {}^1\tau'^{(n+k+2)} = \frac{\alpha_1}{(n+k+2)!}\left(\gamma_1 \sum_{i,j=1}^{s} b'_i c_i^k a_{ij}\, {}^0\overline{\Phi}_j - 1\right).$$

As before, expanding the j summation of ${}^1\tau'^{(n+k+2)}$:

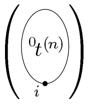
$$\begin{aligned}
{}^1\tau'^{(n+k+2)} &= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{j=1}^{s}\left[\sum_{i=1}^{s} b'_i c_i^k a_{ij} - \frac{b'_j}{k+2} + \frac{b'_j c_j}{k+1} - \frac{b'_j c_j^{k+2}}{(k+2)(k+1)}\right]{}^0\overline{\Phi}_j \\
&\quad + \frac{\alpha_1\gamma_1}{(k+2)(n+k+2)!}\sum_{j=1}^{s} {}^0\Phi_j^{b'} - \frac{\alpha_1\gamma_1}{(k+1)(n+k+2)!}\sum_{j=1}^{s} b'_j c_j\, {}^0\overline{\Phi}_j \\
&\quad + \frac{\alpha_1\gamma_1}{(k+2)(k+1)(n+k+2)!}\sum_{j=1}^{s} b'_j c_j^{k+2}\, {}^0\overline{\Phi}_j - \frac{\alpha_1}{(n+k+2)!} \\
&= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} R_{ki}\, {}^0\overline{\Phi}_i + \beta^r \\
&= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} \Phi_i^r + \beta^r,
\end{aligned}$$

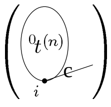
where

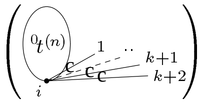
$$\begin{aligned}
\beta^r &= \frac{\alpha_1\gamma_1}{(k+2)(n+k+2)!}\sum_{j=1}^{s} {}^0\Phi_j^{b'} - \frac{\alpha_1\gamma_1}{(k+1)(n+k+2)!}\sum_{j=1}^{s} b'_j c_j\, {}^0\overline{\Phi}_j \\
&\quad + \frac{\alpha_1\gamma_1}{(k+2)(k+1)(n+k+2)!}\sum_{j=1}^{s} b'_j c_j^{k+2}\, {}^0\overline{\Phi}_j - \frac{\alpha_1}{(n+k+2)!}.
\end{aligned} \tag{3.19}$$

[4] i.e., the number of c's on the root.

Writing

$\Theta^r = \lambda_0\,{}^0\tau'^{(n)}$ 

$-\,\lambda_2\,{}^2\tau'^{(n+1)}$ 

$+\,\lambda_3\,{}^3\tau'^{(n+k+2)}$ 

, (3.20)

the λ's may be obtained by equating (3.19) to (3.20) as before, giving

$$\lambda_0 = \frac{\alpha_1\gamma_1 n!}{\alpha_0\gamma_0(k+2)(n+k+2)!}, \tag{3.21}$$

$$\lambda_2 = \frac{\alpha_1\gamma_1 (n+1)!}{\alpha_2\gamma_2(k+1)(n+k+2)!}, \tag{3.22}$$

and

$$\lambda_3 = \frac{\alpha_1\gamma_1}{\alpha_3\gamma_3(k+2)(k+1)}, \tag{3.23}$$

leaving the condition

$$\gamma_1\left(\frac{1}{\gamma_0} - \frac{1}{\gamma_2(k+1)} + \frac{1}{\gamma_3(k+2)(k+1)}\right) = 1. \tag{3.24}$$

However,

$$\gamma_0 = \frac{\gamma_1}{(n+k+2)(n+1)},\ \gamma_2 = \frac{\gamma_1}{n(n+k+2)},\ \text{and } \gamma_3 = \frac{\gamma_1}{n(n+1)},$$

and (3.24) is satisfied. Again, directly from the recursive definition of α (3.2):

$$\alpha_0 = \frac{\alpha_1(n+1)!k!}{(n+k+1)!},\ \alpha_2 = \frac{\alpha_1 n(n+1)!k!}{(n_c+1)(n+k+1)!},\ \text{and } \alpha_3 = \frac{\alpha_1 n(n+1)k!n_c!}{(n_c+k+2)!},$$

substituting into (3.21), (3.22) and (3.23) gives

$$\lambda_0 = \frac{k+1}{(k+2)!},\ \lambda_2 = \frac{1+n_c}{(k+1)!},\ \text{and } \lambda_3 = \frac{(k+n_c+2)!}{n_c!(k+2)!}, \tag{3.25}$$

where n_c is, again, the number of single node sub-trees at the root of ${}^0t^{(n)}$.

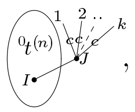
3.2.6 Generalisation to Θ^q and Θ^{qr} combinations

'Q' substitution is essentially equivalent for the RK and RKN case, and is more fundamental than the 'R' substitution of the previous sections as it is applicable to both RK and RKN y *and* y' forms. Whereas the root substitution can only be made once, and allows a simple derivation of the associated linear combination of τ's that arise, 'Q' substitution is not so easily treated in a closed form. The fundamental case of making a *single* substitution into the $\Phi_i^b/\Phi_i^{b'}$ form is, however, pre-requisite.

Single 'Q' substitutions

For the RK case, the relevant tree/τ pair is as follows:

$$^1t^{(n+k+1)} =$$



$$\text{, and } {}^1\tau^{(n+k+1)} = \frac{\alpha_1}{(n+k+1)!}\left(\gamma_1\sum_{i,J=1}^{s}{}^0\Phi_i^b a_{IJ}c_J^k - 1\right).$$

The I subscript is any already present in ${}^0\Phi_i^b$ defined by the n node tree ${}^0t^{(n)}$, with associated α_0, γ_0 and resulting ${}^0\tau^{(n)}$ as before. Now, expanding the I^{th} node of ${}^1\tau^{(n+k+1)}$:

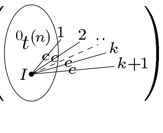
$$\begin{aligned}
{}^1\tau^{(n+k+1)} &= \frac{\alpha_1\gamma_1}{(n+k+1)!}\sum_{i=1}^{s}{}^0\Phi_i^b\left[\sum_{J=1}^{s}a_{IJ}c_J^k - \frac{c_I^{k+1}}{k+1}\right] - \frac{\alpha_1}{(n+k+1)!} \\
&\quad + \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!}\sum_{i=1}^{s}{}^0\Phi_i^b c_I^{k+1} \\
&= \frac{\alpha_1\gamma_1}{(n+k+1)!}\sum_{i=1}^{s}{}^0\Phi_i^b Q_{Ik} + \beta^q \\
&= \frac{\alpha_1\gamma_1}{(n+k+1)!}\sum_{i=1}^{s}\Phi_i^{(q)b} + \beta^q,
\end{aligned}$$

where

$$\beta^q = \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!}\sum_{i=1}^{s}{}^0\Phi_i^b c_I^{k+1} - \frac{\alpha_1}{(n+k+1)!}.$$

It is therefore sufficient to write

$$\Theta^q = \lambda_2{}^2\tau^{(n+k+1)}\left(\quad\right),$$



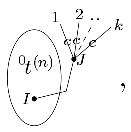
where ${}^2\Phi_i^b$, relating to ${}^2\tau^{(n+k+1)}$, is given by ${}^2\Phi_i^b = {}^1\Phi_i^b c_I^{k+1}$. Equating to β^q gives the two conditions

$$\frac{\lambda_2\alpha_2}{(n+k+1)!} = \frac{\alpha_1}{(n+k+1)!} \quad \text{and} \quad \frac{\lambda_2\alpha_2\gamma_2}{(n+k+1)!} = \frac{\alpha_1\gamma_1}{(k+1)(n+k+1)!}.$$

Solving the first gives

$$\lambda_2 = \frac{\alpha_1}{\alpha_2},$$

and since by definition $\gamma_1 = (k+1)\gamma_2$, the second condition is true. The RKN case is equivalent, with the above theory extending to both y and y' terms. The relevant tree is

$${}^1t^{(n+k+2)} =$$ 

,

where I is any *even* node, and having associated τ and τ' as follows

$${}^1\tau^{(n+k+3)} = \frac{\alpha_1}{(n+k+2)!}\left(\gamma_1 \sum_{i,J=1}^{s} {}^0\Phi_i^b a_{IJ} c_J^k - \frac{1}{n+k+3}\right), \tag{3.26}$$

and

$${}^1\tau'^{(n+k+2)} = \frac{\alpha_1}{(n+k+2)!}\left(\gamma_1 \sum_{i,J=1}^{s} {}^0\Phi_i^{b'} a_{IJ} c_J^k - 1\right). \tag{3.27}$$

Expanding (3.26) gives

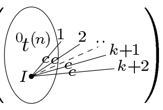
$$\begin{aligned}
{}^1\tau^{(n+k+3)} &= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^b \left[\sum_{J=1}^{s} a_{IJ}c_J^k - \frac{c_I^{k+2}}{(k+2)(k+1)}\right] - \frac{\alpha_1}{(n+k+3)!} \\
&\quad + \frac{\alpha_1\gamma_1}{(k+2)(k+1)(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^b c_I^{k+2} \\
&= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^b Q_{Ik} + \beta^q, \\
&= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} \Phi_i^{(q)b} + \beta^q,
\end{aligned}$$

where

$$\beta^q = \frac{\alpha_1\gamma_1}{(k+2)(k+1)(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^b c_I^{k+2} - \frac{\alpha_1}{(n+k+3)!}.$$

Clearly,

$$\Theta^q = \lambda_2 {}^2\tau^{(n+k+3)} \left(\quad \right),$$



and equating to β^q gives

$$\lambda_2 = \frac{\alpha_1}{\alpha_2}$$

as before. By an identical treatment, the same result holds for the y' forms, by expanding (3.27):

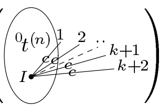
$$\begin{aligned}
{}^1\tau'^{(n+k+2)} &= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^{b'}\left[\sum_{J=1}^{s} a_{IJ}c_J^k - \frac{c_I^{k+2}}{(k+2)(k+1)}\right] - \frac{\alpha_1}{(n+k+2)!} \\
&\quad + \frac{\alpha_1\gamma_1}{(k+2)(k+1)(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^{b'} c_I^{k+2} \\
&= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^{b'} Q_{Ik} + \beta^{q'} \\
&= \frac{\alpha_1\gamma_1}{(n+k+2)!}\sum_{i=1}^{s} \Phi_i^{(q)b'} + \beta^{q'},
\end{aligned}$$

where

$$\beta^{q'} = \frac{\alpha_1\gamma_1}{(k+2)(k+1)(n+k+2)!}\sum_{i=1}^{s} {}^0\Phi_i^{b'} c_I^{k+2} - \frac{\alpha_1}{(n+k+2)!}.$$

With

$$\Theta^q = \lambda_2 {}^2\tau'^{(n+k+2)} \left(\quad \right),$$



and equating to $\beta^{q'}$ gives

$$\lambda_2 = \frac{\alpha_1}{\alpha_2}.$$

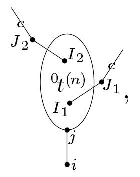
The situation is therefore clear for these *single* substitutions; the required scalar multiple of the relevant τ/τ' in Θ^q is the ratio of α_1 to α_2 in all cases. Unlike the root substitution theory, however, no easily characterisable closed form of this ratio can be given, even for this simple case. A general treatment needs to be derived for substitution of 'Q' parameters giving all available alternative representations, or at least a subset of the useful possibilities. As illustrated previously, many truncation error terms contain the possibility of 'root' and multiple 'Q' substitutions concurrently

— these cases will now be dealt with using a common, computational approach.

Computational general treatment of 'Q' substitution

An approach will now be given for the calculation of general Θ^q and Θ^{qr} combinations during the single recursive traversal of the relevant tree. The starting point is that of the classical Φ_i^b representation, or the Φ_i^r/Θ^r form after root substitution characterised above. Essentially, each 'Q' or 'R' parameter form encountered on substitution into the Φ_i^b quantity results in the manipulation of vectors of trees and associated scalar multiples that make up the relevant Θ combination. Each 'Q' substitution is considered for *all* trees currently held, and the α_1/α_2 ratio incorporated, with the specific α quantities calculated when needed by a separate process. This is best described by means of a suitably general example, so consider the following tree relating to the RK case:

$${}^1t^{(n+5)} =$$



,

the desired final form being

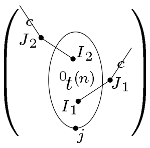
$${}^1\tau^{(n+5)} = \frac{\alpha_1\gamma_1}{(n+5)!}\sum_{j}^{s} R_{0j}\,{}^0\overline{\Phi}_j Q_{I_1,1}Q_{I_2,1} + \Theta^{qr},$$

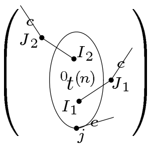
after having made an R_0 substitution, and two Q_1 substitutions at nodes I_1 and I_2. As the tree is processed recursively, the root substitution will be made first, giving the closed form intermediate result

$${}^1\tau^{(n+5)} = \frac{\alpha_1\gamma_1}{(n+5)!}\sum_{j,J_1,J_2}^{s} R_{0j}\,{}^0\overline{\Phi}_j a_{I_1J_1}c_{J_1}a_{I_2J_2}c_{J_2} + \Theta^r,$$

where

$$\Theta^r = \lambda_2\,{}^2\tau^{(n+4)}\left(\quad\right) - \lambda_3\,{}^3\tau^{(n+5)}\left(\quad\right),$$

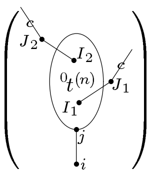


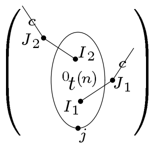


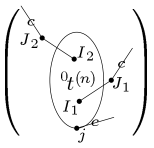
with

$$\lambda_2 = 1, \text{ and } \lambda_3 = 1 + n_c,$$

from (3.25) with $k = 0$ — as yet, no specific ${}^{0}t^{(n)}$ has been given, so n_c is unknown. Now, re-arranging:

$${}^{1}T^{(n+5)}$$


$$-\ {}^{2}T^{(n+4)}$$


$$+\ (1+n_c)\cdot {}^{3}T^{(n+5)}$$


$$= \frac{\alpha_1\gamma_1}{(n+5)!}\sum_{j,J_1,J_2}^{s} R_{0j}\,{}^{0}\overline{\Phi}_j\, a_{I_1J_1}c_{J_1}a_{I_2J_2}c_{J_2}, \qquad (3.28)$$

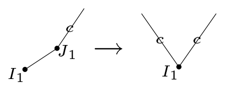
gives the form used internally by RK-AID during the left-centre-right recursive processing that generates all information from the fundamental tree ${}^{1}t^{(n+5)}$, and that in which 'Q' substitution is now done. The general structure in the iteration, for which the left-hand-side of expression (3.28) is the starting point, is best thought of as being *dynamic*. Internally it is stored as a p-vector of scalars V_p and a separate p-vector of trees T_p (in a Scoins height form) that make up the Θ combination as

$$\Theta = -\sum_{i=2}^{p} {}^{i}V_p\,{}^{i}T_p,$$

where in this case, $\Theta \equiv \Theta^{qr}$, $p = 3$, $V_p = \{1, -1, 1+n_c\}$, and the T_p vector of trees is as above. Now, assuming the I_1 node is the first of $\{I_1, I_2\}$ reached as the tree is traversed, the substitution

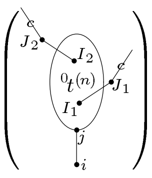
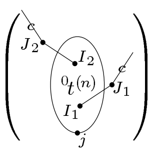
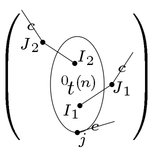
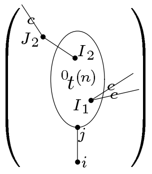
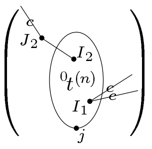
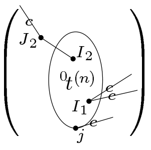
$$\sum_{J_1=1}^{s} a_{I_1J_1}c_{J_1} \rightarrow Q_{I_1,1}$$

in $\Phi_i^{(q)r}$ requires the modification



in the representation of all trees ${}^{i}T_p,\ i = 1, \ldots, p$, forming $\tilde{T}_p$ which is then *appended* onto T_p such that $p \rightarrow 2p,\ T_{2p} \rightarrow \left\{T_p, \tilde{T}_p\right\}$. Each of ${}^{i}V_p,\ i = 1, \ldots, p$, is multiplied by

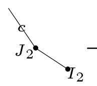
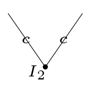
-$\alpha_i/\tilde{\alpha}_i$, where α_i and $\tilde{\alpha}_i$ are the α quantity calculated (or looked-up) for the trees ${}^iT_p$ and ${}^i\tilde{T}_p$, respectively, forming $\tilde{V}_p$, and then finishing the iteration with $V_{2p} \to \left\{V_p, \tilde{V}_p\right\}$. The resulting form in this case is therefore

$${}^1\mathcal{T}^{(n+5)}$$

$$- {}^2\mathcal{T}^{(n+4)}$$

$$+ (1+n_c)\cdot {}^3\mathcal{T}^{(n+5)}$$

$$- \left(\frac{\alpha_1}{\alpha_4}\right)\cdot {}^4\mathcal{T}^{(n+5)}$$

$$+ \left(\frac{\alpha_2}{\alpha_5}\right)\cdot {}^5\mathcal{T}^{(n+4)}$$

$$- \left(\frac{\alpha_3(1+n_c)}{\alpha_6}\right)\cdot {}^6\mathcal{T}^{(n+5)}$$

$$= \frac{\alpha_1\gamma_1}{(n+5)!}\sum_{j,J_2=1}^{s} R_{0j}\,{}^0\overline{\Phi}_j Q_{I_1,1} a_{I_2J_2} c_{J_2},$$

where

$$V_6 = \left\{1, -1, 1+n_c, -\frac{\alpha_1}{\alpha_4}, \frac{\alpha_2}{\alpha_5}, -\frac{\alpha_3(1+n_c)}{\alpha_6}\right\},$$

and T_6 is a vector of the six trees illustrated above. It is clear how the complexity of the Θ form can increase very rapidly with each substitution if the order is high (in this case ≥ 6). The final Q_1 substitution is made when node I_2 is reached by the algorithm, i.e.,

$$\sum_{J_2=1}^{s} a_{I_2J_2}c_{J_2} \to Q_{I_2,1}, \text{ and }$$

$$\longrightarrow$$


in $\Phi_i^{(q)r}$ and ${}^iT_6,\ i=1,\ldots,6$, respectively. This gives the required final form, shown in Figure 3.3. Only technical details now remain regarding acquisition of the α quantities needed in this example (for a specific ${}^0t^{(n)}$), and for any other case involving 'Q' substitutions. Primarily, α_1 is always calculated for general use, and is therefore available at the end of the iteration. All fundamental root substitution τ combinations have a closed form, the relevant α quantities (α_2 and α_3 here) being available as ratios of α_1 and are therefore also available. All other α's ($\alpha_i,\ i=4,\ldots,12$ here) are calculated as needed, and it is considered that such a process, if coded efficiently, is

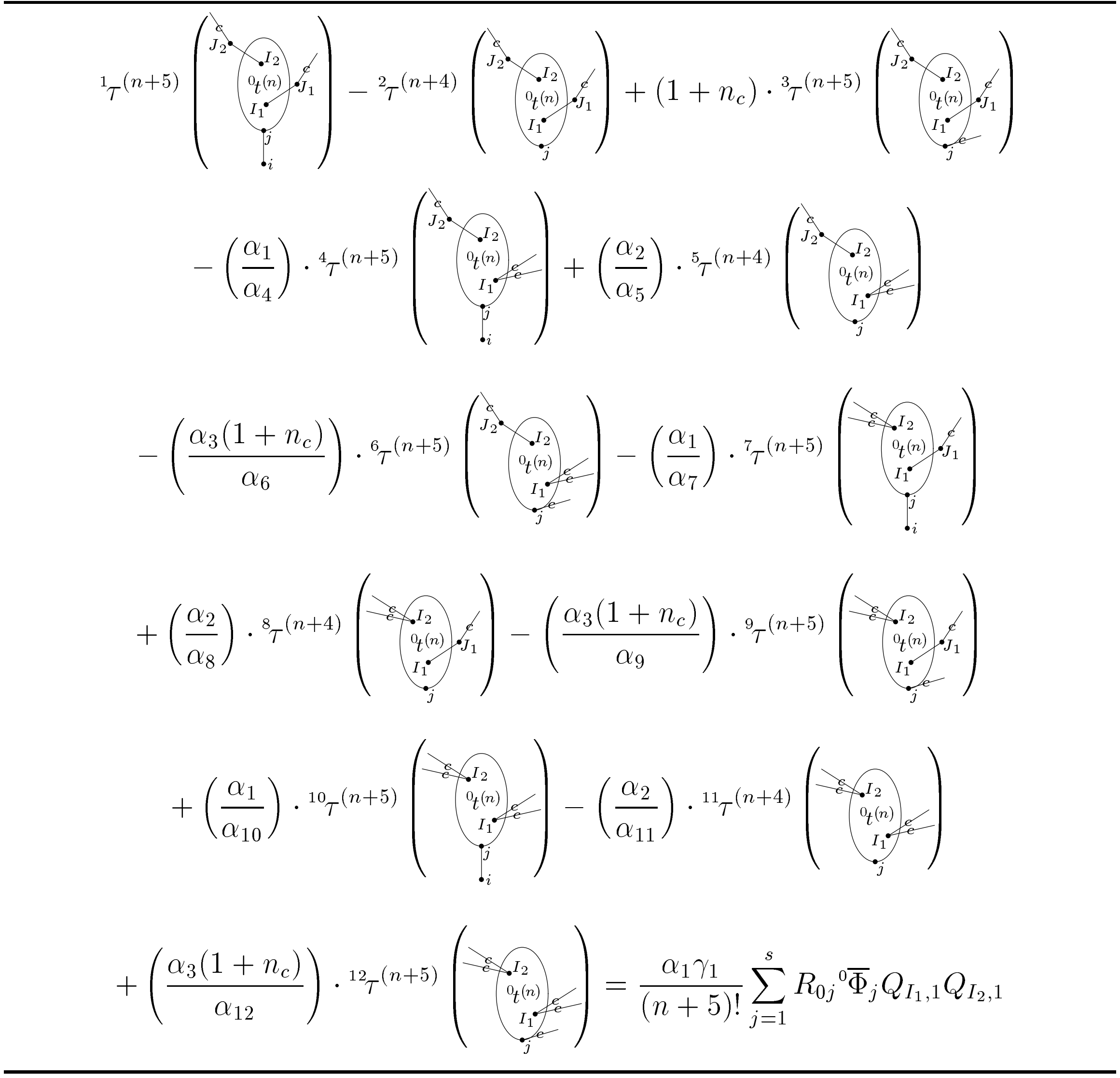
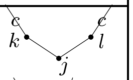
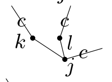
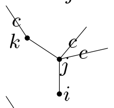
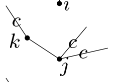
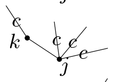
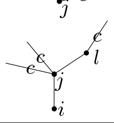


Figure 3.3: Example of a root and multiple 'Q' substitution.

a reasonable solution to the problem.

It is not obvious from the example above that trees involved in the linear combinations will often repeat in the sequence, and therefore that their relevant multiples must be summed. This can best be shown by specifying ${}^{0}t^{(n)}$ in the example here. For

τ	α	Tree
${}^{2}\tau^{(5)}$	3	
${}^{3}\tau^{(6)}$	15	
${}^{4}\tau^{(6)}$	6	
${}^{5}\tau^{(5)}$	6	
${}^{6}\tau^{(6)}$	10	
${}^{7}\tau^{(6)}$	6	

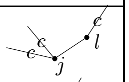
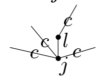
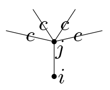
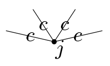
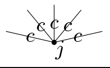
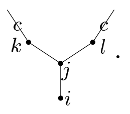
τ	α	Tree
${}^{8}\tau^{(5)}$	6	
${}^{9}\tau^{(6)}$	10	
${}^{10}\tau^{(6)}$	1	
${}^{11}\tau^{(5)}$	1	
${}^{12}\tau^{(6)}$	1	

Figure 3.4: Θ^{qr} trees for an R_{0j} and Q^2_{1j} substitution.

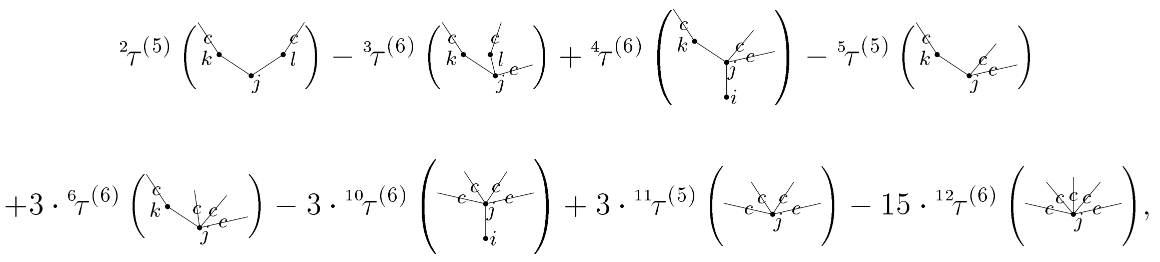
simplicity, $n = 1$, ${}^{0}t^{(n)} = {}^{0}t^{(1)} = \cdot_j$ is sufficient to illustrate the principals involved. In this case, therefore, $I_1 = I_2 = j$ and

$${}^{1}t^{(6)} =$$ 

From the definitions, $\alpha_1 = 3,\ \gamma_1 = 120$ and therefore

$${}^{1}\tau^{(6)} = \frac{1}{2}\sum_{ijkl=1}^{s} b_i a_{ij} a_{jk} c_k a_{jl} c_l - \frac{1}{240} = \frac{1}{2}\sum_{j=1}^{s} R_{0j} Q^2_{j1} + \Theta^{qr},$$

where Θ^{qr} is a linear combination of ${}^{2}\tau, \ldots, {}^{12}\tau$ with associated α quantities, given in Figure 3.4. Of these trees, ${}^{7}t^{(6)} \equiv {}^{4}t^{(6)}$, ${}^{8}t^{(5)} \equiv {}^{5}t^{(5)}$, and ${}^{9}t^{(6)} \equiv {}^{6}t^{(6)}$ (by isomorphisms), the scalar multiplying the trees ${}^{4}t^{(6)}$, ${}^{5}t^{(5)}$ and ${}^{6}t^{(6)}$ is therefore effectively doubled. The final form of Θ^{qr} is now


${}^{2}\tau^{(5)}$
$-\,{}^{3}\tau^{(6)}$
$+\,{}^{4}\tau^{(6)}$
$-\,{}^{5}\tau^{(5)}$
$+3\cdot{}^{6}\tau^{(6)}$
$-\,3\cdot{}^{10}\tau^{(6)}$
$+\,3\cdot{}^{11}\tau^{(5)}$
$-\,15\cdot{}^{12}\tau^{(6)}$
c
k
l
j
i
e

by substitution of ${}^{0}\!t^{(1)}$ and the α and n_c quantities into Figure 3.3, with reference to Figure 3.4.

The extended row-sum and root substitution process is now complete in terms of the scaling factors and trees required to generate possible alternative representations of the truncation error terms. However, it is the τ's that are inevitably required, needing a treatment of the tree forms to relate to the 'canonical' Scoins generation process, and also such that repeated isomorphisms of any resulting trees are dealt with appropriately (as in the example above).

3.2.7 Tree isomorphism and τ numbering

In the previous sections, 'Q' and 'R' parameter substitution was motivated by direct manipulation of trees. The resulting forms of this simple graphical approach do not necessarily fall into a canonical 'left heavy' representation, and as such it is not computationally trivial to recognise isomorphic repetitions. Also, since the τ sequence of interest corresponds to canonical trees, it is preferable to always *convert* from the natural form to a canonical form during the recursive process. In this way, repetition is obvious and it is simpler to relate the final form of the tree combinations to the required τ numbering scheme. In the previous example, the trees of the final Θ^{qr} form can be 'looked up' from Figure 3.1, the desired truncation error term is thus recognised as $\tau_{12}^{(6)}$, and can be written as

$$\frac{1}{2}\sum_i R_{0i}Q_{i1}^2+\tau_3^{(5)}-\tau_3^{(6)}+\tau_{11}^{(6)}-\tau_2^{(5)}+3\tau_2^{(6)}-3\tau_7^{(6)}+3\tau_1^{(5)}-15\tau_1^{(6)}.$$

$$\tau_1^{(1)} = \sum_i b_i - 1$$

$$\tau_1^{(2)} = \sum_i b_i c_i - \frac{1}{2!}$$

$$\tau_1^{(3)} = \tfrac{1}{2}\sum_i b_i c_i^2 - \frac{1}{3!}$$

$$\tau_2^{(3)} = \sum_i b_i Q_{i1} + \tau_1^{(3)} \text{ or } \sum_i R_{0i} c_i + \tau_1^{(2)} - 2\tau_1^{(3)}$$

$$\tau_1^{(4)} = \tfrac{1}{6}\sum_i b_i c_i^3 - \frac{1}{4!}$$

$$\tau_2^{(4)} = \sum_i b_i c_i Q_{i1} + 3\tau_1^{(4)} \text{ or } \sum_i R_{1i} c_i + \frac{1}{2}\tau_1^{(2)} - 3\tau_1^{(4)}$$

$$\tau_3^{(4)} = \tfrac{1}{2}\sum_i b_i Q_{i2} + \tau_1^{(4)} \text{ or } \tfrac{1}{2}\sum_i R_{0i} c_i^2 + \tau_1^{(3)} - 3\tau_1^{(4)}$$

$$\tau_4^{(4)} = \sum_i R_{0i} Q_{i1} + \tau_2^{(3)} - \tau_2^{(4)} + \tau_3^{(4)} - \tau_1^{(3)} + 3\tau_1^{(4)}$$

$$\tau_1^{(5)} = \tfrac{1}{24}\sum_i b_i c_i^4 - \frac{1}{5!}$$

$$\tau_2^{(5)} = \tfrac{1}{2}\sum_i b_i c_i^2 Q_{i1} + 6\tau_1^{(5)} \text{ or } \tfrac{1}{2}\sum_i R_{2i} c_i + \frac{1}{6}\tau_1^{(2)} - 4\tau_1^{(5)}$$

$$\tau_3^{(5)} = \tfrac{1}{2}\sum_i b_i Q_{i1}^2 + \tau_2^{(5)} - 3\tau_1^{(5)}$$

$$\tau_4^{(5)} = \tfrac{1}{2}\sum_i b_i c_i Q_{i2} + 4\tau_1^{(5)} \text{ or } \tfrac{1}{2}\sum_i R_{1i} c_i^2 + \frac{1}{2}\tau_1^{(3)} - 6\tau_1^{(5)}$$

$$\tau_5^{(5)} = \tfrac{1}{6}\sum_i b_i Q_{i3} + \tau_1^{(5)} \text{ or } \tfrac{1}{6}\sum_i R_{0i} c_i^3 + \tau_1^{(4)} - 4\tau_1^{(5)}$$

$$\tau_6^{(5)} = \sum_i R_{1i} Q_{i1} + \frac{1}{2}\tau_2^{(3)} - \tau_2^{(5)} + \tau_4^{(5)} - \frac{1}{2}\tau_1^{(3)} + 6\tau_1^{(5)}$$

$$\tau_7^{(5)} = \sum_i R_{0i} c_i Q_{i1} + \tau_2^{(4)} - 2\tau_2^{(5)} + 3\tau_5^{(5)} - 3\tau_1^{(4)} + 12\tau_1^{(5)}$$

$$\tau_8^{(5)} = \tfrac{1}{2}\sum_i R_{0i} Q_{i2} + \tau_3^{(4)} - \tau_4^{(5)} + \tau_5^{(5)} - \tau_1^{(4)} + 4\tau_1^{(5)}$$

$$\tau_9^{(5)} = \sum_{ij} R_{0i} a_{ij} Q_{j1} + \tau_4^{(4)} - \tau_6^{(5)} + \tau_8^{(5)} - \tau_3^{(4)} + \tau_4^{(5)}$$

Figure 3.5: RK 'QR form' truncation error coefficients up to fifth-order.

$$
\begin{aligned}
\tau_1^{(6)} &= \tfrac{1}{120}\sum_i b_i c_i^5 - \frac{1}{6!}\\
\tau_2^{(6)} &= \tfrac{1}{6}\sum_i b_i c_i^3 Q_{i1} + 10\tau_1^{(6)} \text{ or } \tfrac{1}{6}\sum_i R_{3i} c_i + \frac{1}{24}\tau_1^{(2)} - 5\tau_1^{(6)}\\
\tau_3^{(6)} &= \tfrac{1}{2}\sum_i b_i c_i Q_{i1}^2 + 3\tau_2^{(6)} - 15\tau_1^{(6)}\\
\tau_4^{(6)} &= \tfrac{1}{4}\sum_i b_i c_i^2 Q_{i2} + 10\tau_1^{(6)} \text{ or } \tfrac{1}{4}\sum_i R_{2i} c_i^2 + \frac{1}{6}\tau_1^{(3)} - 10\tau_1^{(6)}\\
\tau_5^{(6)} &= \tfrac{1}{2}\sum_i b_i Q_{i2} Q_{i1} + \tau_2^{(6)} + \tau_4^{(6)} - 10\tau_1^{(6)}\\
\tau_6^{(6)} &= \tfrac{1}{6}\sum_i b_i c_i Q_{i3} + 5\tau_1^{(6)} \text{ or } \tfrac{1}{6}\sum_i R_{1i} c_i^3 + \frac{1}{2}\tau_1^{(4)} - 10\tau_1^{(6)}\\
\tau_7^{(6)} &= \tfrac{1}{24}\sum_i b_i Q_{i4} + \tau_1^{(6)} \text{ or } \tfrac{1}{24}\sum_i R_{0i} c_i^4 + \tau_1^{(5)} - 5\tau_1^{(6)}\\
\tau_8^{(6)} &= \tfrac{1}{2}\sum_i R_{2i} Q_{i1} + \frac{1}{6}\tau_2^{(3)} - \tau_2^{(6)} + \tau_4^{(6)} - \frac{1}{6}\tau_1^{(3)} + 10\tau_1^{(6)}\\
\tau_9^{(6)} &= \sum_{ij} b_i a_{ij} Q_{j1} Q_{i1} + \tau_5^{(6)} + \tau_8^{(6)} - \tau_4^{(6)}\\
\tau_{10}^{(6)} &= \sum_i R_{1i} c_i Q_{i1} + \frac{1}{2}\tau_2^{(4)} - 3\tau_2^{(6)} + 3\tau_6^{(6)} - \frac{3}{2}\tau_1^{(4)} + 30\tau_1^{(6)}\\
\tau_{11}^{(6)} &= \tfrac{1}{2}\sum_i R_{0i} c_i^2 Q_{i1} + \tau_2^{(5)} - 3\tau_2^{(6)} + 6\tau_7^{(6)} - 6\tau_1^{(5)} + 30\tau_1^{(6)}\\
\tau_{12}^{(6)} &= \tfrac{1}{2}\sum_i R_{0i} Q_{i1}^2 + \tau_3^{(5)} - \tau_3^{(6)} + \tau_{11}^{(6)} - \tau_2^{(5)} + 3\tau_2^{(6)} - 3\tau_7^{(6)} + 3\tau_1^{(5)} - 15\tau_1^{(6)}\\
\tau_{13}^{(6)} &= \tfrac{1}{2}\sum_i R_{1i} Q_{i2} + \frac{1}{2}\tau_3^{(4)} - \tau_4^{(6)} + \tau_6^{(6)} - \frac{1}{2}\tau_1^{(4)} + 10\tau_1^{(6)}\\
\tau_{14}^{(6)} &= \tfrac{1}{2}\sum_i R_{0i} c_i Q_{i2} + \tau_4^{(5)} - 2\tau_4^{(6)} + 4\tau_7^{(6)} - 4\tau_1^{(5)} + 20\tau_1^{(6)}\\
\tau_{15}^{(6)} &= \tfrac{1}{6}\sum_i R_{0i} Q_{i3} + \tau_5^{(5)} - \tau_6^{(6)} + \tau_7^{(6)} - \tau_1^{(5)} + 5\tau_1^{(6)}\\
\tau_{16}^{(6)} &= \sum_{ij} R_{1i} a_{ij} Q_{j1} + \frac{1}{2}\tau_4^{(4)} - \tau_8^{(6)} + \tau_{13}^{(6)} - \frac{1}{2}\tau_3^{(4)} + \tau_4^{(6)}\\
\tau_{17}^{(6)} &= \sum_{ij} R_{0i} c_i a_{ij} Q_{j1} + \tau_6^{(5)} - 2\tau_8^{(6)} + \tau_{14}^{(6)} - \tau_4^{(5)} + 2\tau_4^{(6)}\\
\tau_{18}^{(6)} &= \sum_{ij} R_{0i} a_{ij} c_j Q_{j1} + \tau_7^{(5)} - \tau_{10}^{(6)} + 3\tau_{15}^{(6)} - 3\tau_5^{(5)} + 3\tau_6^{(6)}\\
\tau_{19}^{(6)} &= \tfrac{1}{2}\sum_{ij} R_{0i} a_{ij} Q_{j2} + \tau_8^{(5)} - \tau_{13}^{(6)} + \tau_{15}^{(6)} - \tau_5^{(5)} + \tau_6^{(6)}\\
\tau_{20}^{(6)} &= \sum_{ijk} R_{0i} a_{ij} a_{jk} Q_{k1} + \tau_9^{(5)} - \tau_{16}^{(6)} + \tau_{19}^{(6)} - \tau_8^{(5)} + \tau_{13}^{(6)}
\end{aligned}
$$

Figure 3.6: Sixth-order RK 'QR form' truncation error coefficients.

This process is much more difficult to do in general than for specific cases, and has been elusive as a programmable algorithm. The general approach of RK-AID is to allow truncation error term generation for isolated trees from any order, and therefore

$$\tau_1'^{(1)} = \sum_i b_i' - 1$$

$$\tau_1'^{(2)} = \sum_i b_i' c_i - \frac{1}{2!}$$

$$\tau_1'^{(3)} = \tfrac{1}{2}\sum_i b_i' c_i^2 - \frac{1}{3!}$$

$$* \quad \tau_2'^{(3)} = \tau_1'^{(3)}$$

$$\tau_1'^{(4)} = \tfrac{1}{6}\sum_i b_i' c_i^3 - \frac{1}{4!}$$

$$* \quad \tau_2'^{(4)} = 3\tau_1'^{(4)}$$

$$\tau_3'^{(4)} = \sum_i b_i' Q_{i1} + \tau_1'^{(4)} \text{ or } \sum_i R_{0i} c_i + \frac{1}{2}\tau_1'^{(2)} - 2\tau_1'^{(3)} + 3\tau_1'^{(4)}$$

$$\tau_1'^{(5)} = \tfrac{1}{24}\sum_i b_i' c_i^4 - \frac{1}{5!}$$

$$* \quad \tau_2'^{(5)} = 6\tau_1'^{(5)}$$

$$* \quad \tau_3'^{(5)} = 3\tau_1'^{(5)}$$

$$\tau_4'^{(5)} = \sum_i b_i' c_i Q_{i1} + 4\tau_1'^{(5)} \text{ or } \sum_i R_{1i} c_i + \frac{1}{3}\tau_1'^{(2)} - \tau_1'^{(3)} + 4\tau_1'^{(5)}$$

$$\tau_5'^{(5)} = \tfrac{1}{2}\sum_i b_i' Q_{i2} + \tau_1'^{(5)} \text{ or } \tfrac{1}{2}\sum_i R_{0i} c_i^2 + \frac{1}{2}\tau_1'^{(3)} - 3\tau_1'^{(4)} + 6\tau_1'^{(5)}$$

$$* \quad \tau_6'^{(5)} = \tau_5'^{(5)}$$

Figure 3.7: RKN y' 'QR form' truncation error coefficients up to fifth-order.

the obvious (and admittedly clumsy) solution is to step through every tree in the sequence for that order until the required one is found. This can be done very rapidly in software, and gives a maximum amount of freedom without storage requirements. A more refined treatment may be sought, but at present it is the one implemented. Figures 3.5–3.8 show the final form of truncation error coefficients up to sixth order, for both RK and RKN y' terms. The case shown is where a possible 'R' substitution and all 'Q' substitutions have been made (as in Tables 3.4–3.8), including an option whenever 'Q' and 'R' parameters *overlap*. The format of these truncation error terms

$$
\begin{array}{lll}
 & \tau_1'^{(6)} & = \frac{1}{120}\sum_i b_i' c_i^5 - \dfrac{1}{6!} \\
* & \tau_2'^{(6)} & = 10\tau_1'^{(6)} \\
* & \tau_3'^{(6)} & = 15\tau_1'^{(6)} \\
 & \tau_4'^{(6)} & = \frac{1}{2}\sum_i b_i' c_i^2 Q_{i1} + 10\tau_1'^{(6)} \text{ or } \frac{1}{2}\sum_i R_{2i} c_i + \dfrac{1}{8}\tau_1'^{(2)} - \dfrac{1}{3}\tau_1'^{(3)} + 5\tau_1'^{(6)} \\
* & \tau_5'^{(6)} & = \tau_4'^{(6)} \\
 & \tau_6'^{(6)} & = \frac{1}{2}\sum_i b_i' c_i Q_{i2} + 5\tau_1'^{(6)} \text{ or } \frac{1}{2}\sum_i R_{1i} c_i^2 + \dfrac{1}{3}\tau_1'^{(3)} - \dfrac{3}{2}\tau_1'^{(4)} + 10\tau_1'^{(6)} \\
 & \tau_7'^{(6)} & = \frac{1}{6}\sum_i b_i' Q_{i3} + \tau_1'^{(6)} \text{ or } \frac{1}{6}\sum_i R_{0i} c_i^3 + \dfrac{1}{2}\tau_1'^{(4)} - 4\tau_1'^{(5)} + 10\tau_1'^{(6)} \\
* & \tau_8'^{(6)} & = \tau_6'^{(6)} \\
* & \tau_9'^{(6)} & = 3\tau_7'^{(6)} \\
 & \tau_{10}'^{(6)} & = \sum_i R_{0i} Q_{i1} + \dfrac{1}{2}\tau_3'^{(4)} - \tau_4'^{(5)} + \tau_4'^{(6)} + \tau_7'^{(6)} - \dfrac{1}{2}\tau_1'^{(4)} + 4\tau_1'^{(5)} - 10\tau_1'^{(6)}
\end{array}
$$

Figure 3.8: Sixth-order RKN y' 'QR form' truncation error coefficients.

is now exactly as output by RK-AID (in fact, the typeset form shown derives from the file "*rkive.tex*", which is produced automatically). The commands given for their generation are:

```
[y] order :6
use Q's & R's
```

for the RK coefficients, and

```
Nystrom
[y'] order :6
use Q's & R's
```

for RKN y' terms. Explanation of the generalised form of these commands can be found in Appendix A. Note that for the RKN case, row-sum condition τ/τ' dependence, again marked with an asterisk $(*)$, is effectively handled by 'Q' substitution with $k = 0$, giving a single term in Θ^q (and $\Phi_i^{(q)b}/\Phi_i^{(q)b'} = 0,\ i = 1, \ldots, s$).

3.2.8 Error terms for embedded processes

Truncation error coefficients are of primary interest in the study of RK(N) processes, including embedded pairs or *triples* of explicit methods that share function evaluations. These separate formulae are handled by RK-AID concurrently as a single *model*. Error terms for the embedded processes differ primarily by the set of external weights, the previous treatment is therefore immediately extendible to these cases, replacing $b_i/b'_i/R_{ki}$ with the relevant parameters.

For the dense process of an RK(N) triple, only a very simple modification to the terms of the discrete case is necessary. Essentially, truncation error coefficients of the continuous process may be derived by substituting into the standard forms with $s \rightarrow s^*$, $b_i/\widehat{b}_i \rightarrow b_i^*$, and $c_i \rightarrow c_i/\sigma$. Also,

- $a_{ij} \rightarrow a_{ij}/\sigma$

for the RK case, and for RKN:

- $b'_i/\widehat{b}'_i \rightarrow b_i'^*$,
- $a_{ij} \rightarrow a_{ij}/\sigma^2$.

The σ dependence in Q^* parameters is removable as a factor for both RK and RKN, and so Q_{ik} is replaced by Q_{ik}/σ^{k+1} and Q_{ik}/σ^{k+2}, respectively. However, R_{kj} parameters become σ dependent, i.e.,

$$R_{kj}^* = \frac{1}{\sigma^{k+1}}\left(\sum_{i=1}^{s^*} b_i^* c_i^k a_{ij} + \frac{b_j^* c_j^{k+1}}{k+1}\right) - \frac{b_j^*}{k+1}$$

for RK, and

$$R^*_{kj} = \frac{1}{\sigma^{k+2}}\left(\sum_{i=1}^{s^*} b'^*_i c^k_i a_{ij} - \frac{b'^*_j c^{k+2}_j}{(k+2)(k+1)}\right) + \frac{b'^*_j c_j}{\sigma(k+1)} - \frac{b'^*_j}{k+2}$$

for RKN. An an example, RKN y'^* terms up to fifth-order may be written:

$$\begin{array}{lll}
 & \tau'^{(1)*}_1 & = \sum_{i=1}^{s^*} b'^*_i - 1 \\
 & \tau'^{(2)*}_1 & = \sum_{i=1}^{s^*} b'^*_i c_i/\sigma - \frac{1}{2!} \\
 & \tau'^{(3)*}_1 & = \frac{1}{2}\sum_{i=1}^{s^*} b'^*_i c^2_i/\sigma^2 - \frac{1}{3!} \\
* & \tau'^{(3)*}_2 & = \tau'^{(3)*}_1 \\
 & \tau'^{(4)*}_1 & = \frac{1}{6}\sum_{i=1}^{s^*} b'^*_i c^3_i/\sigma^3 - \frac{1}{4!} \\
* & \tau'^{(4)*}_2 & = 3\tau'^{(4)*}_1 \\
 & \tau'^{(4)*}_3 & = \sum_{i=1}^{s^*} b'^*_i Q_{i1}/\sigma^3 + \tau'^{(4)*}_1 \text{ or } \sum_{i=1}^{s^*} R^*_{0i} c_i/\sigma + \frac{1}{2}\tau'^{(2)*}_1 - 2\tau'^{(3)*}_1 + 3\tau'^{(4)*}_1 \\
 & \tau'^{(5)*}_1 & = \frac{1}{24}\sum_{i=1}^{s^*} b'^*_i c^4_i/\sigma^4 - \frac{1}{5!} \\
* & \tau'^{(5)*}_2 & = 6\tau'^{(5)*}_1 \\
* & \tau'^{(5)*}_3 & = 3\tau'^{(5)*}_1 \\
 & \tau'^{(5)*}_4 & = \sum_{i=1}^{s^*} b'^*_i c_i Q_{i1}/\sigma^4 + 4\tau'^{(5)*}_1 \text{ or } \sum_{i=1}^{s^*} R^*_{1i} c_i/\sigma + \frac{1}{3}\tau'^{(2)*}_1 - \tau'^{(3)*}_1 + 4\tau'^{(5)*}_1 \\
 & \tau'^{(5)*}_5 & = \frac{1}{2}\sum_{i=1}^{s^*} b'^*_i Q_{i2}/\sigma^4 + \tau'^{(5)*}_1 \text{ or } \frac{1}{2}\sum_{i=1}^{s^*} R^*_{0i} c^2_i/\sigma^2 + \frac{1}{2}\tau'^{(3)*}_1 - 3\tau'^{(4)*}_1 + 6\tau'^{(5)*}_1 \\
* & \tau'^{(5)*}_6 & = \tau'^{(5)*}_5
\end{array}$$

3.3 Manipulation of order conditions

Having developed the facility for generation of truncation error terms, emphasis now shifts to their manipulation via control of the method parameters. The primary concern is to aid solution of sets of equations of the form $\tau^{(i)}_j = 0$ (for a discrete process) by emulation of various algebraic techniques. The capabilities of RK-AID to this end are motivated by knowledge of existing RK(N) processes and their derivation — much of this is left to the next chapter, the main 'symbolic' functions of the software are, however, described in brief next.

3.3.1 Equation mode

The RK-AID equation mode is fundamental to the production of algebraic conditions for solution rather than terms for reference. The principal difference is the removal of leading constants and trailing τ combinations, producing an alternative *independent set* of conditions from the 'Q' and/or 'R' representations. For example, here are the RKN y'^* terms from above, using the equation mode (RK-AID commands precede the output):

```
Nystrom
[s'] order :5
equation mode
use Q's & R[s]'s
```

$1'^*.\quad \sum_i b_i'^* - 1$

$2'^*.\quad \sum_i b_i'^* c_i/\sigma - \frac{1}{2}$

$3'^*.\quad \sum_i b_i'^* c_i^2/\sigma^2 - \frac{1}{3}$

--- (3 / 4)

$5'^*.\quad \sum_i b_i'^* c_i^3/\sigma^3 - \frac{1}{4}$

$7'^*.\quad \sum_i b_i'^* Q_{i1}$ or $\sum_i R_{0i}^* c_i$

--- (5 / 7)

$8'^*.\quad \sum_i b_i'^* c_i^4/\sigma^4 - \frac{1}{5}$

$11'^*.\quad \sum_i b_i'^* c_i Q_{i1}$ or $\sum_i R_{1i}^* c_i$

$12'^*.\quad \sum_i b_i'^* Q_{i2}$ or $\sum_i R_{0i}^* c_i^2$

Total independent / Total : (8 / 13)

Note that in all cases where a subset of the equations are solved by the model, a statistic regarding the independent set is given following each order. The main tools for solution of equations depend on being able to reference parameters of the stages in a pseudo-algebraic manner, this is termed 'reduction' — facilities for which are detailed next.

3.3.2 'Free stage' reduction

A 'free stage' model is such that an unlimited number of stages is assumed which, especially for an explicit RK(N), means that there is no 'collapsing' of terms at high-order due to a restriction on the number of stages. The method for referencing the process parameters is described in Section A.3.2. Primarily, the ability to zero internal/external weight, 'Q' and 'R' parameter families enables simplifying conditions to be imposed — examples of which can be found in Section A.6.19 and throughout the remaining chapters where they are used extensively. Fundamental to this is the 'reduction mode', where the displayed equations and error terms will be modified algebraically to reflect patterns of zero parameters in the recursive summations.

3.3.3 Implied algebraic reduction

It is possible to equate implied summations of parameters to zero (and zero only) by means of the `:=` operator, e.g., `b(i)a(i,2):=0`. Since no facility is available for specifying over which range of indices the implied summation corresponds to, RK-AID takes the summation information from the first matching equation found. Note that conditions made in this way will be applied to any term also containing it as a *partial sum*, including in error terms (when the order of principal error terms is known via the `ERROR` feature of the `ORDER` command — see Section A.6.4). The primary use of this feature, however, is for *automatic* removal of common expressions such that no equation is repeated — described next.

3.3.4 Automatic equation 'splitting'

A form of automatic implied reduction has been implemented via the `SPLIT` command. For example, when the command `SPLIT [y]` is given, a mode is entered in which (lower-order) y equations output in equation mode are reduced to a more fundamental, independent set by means of basic assumptions. All equivalent equations will then be shown once only, and from then on eliminated from later terms containing that condition as a partial sum. Also, any equation that has become simplified to be the sum of two *or more* distinct parts, for example:

$$\lambda_1 \sum_{i=4}^{s} b_i a_{i2} + \lambda_2 \sum_{i=4}^{s} b_i a_{i3}, \tag{3.29}$$

will be 'split' into its components on the assumption that the only solution is for all Σ parts to be zero (the λ's are assumed constant and could represent any combination of parameters and scalars). On the first occurrence, the term above will be replaced by

$$\sum_{i=4}^{s} b_i a_{i2}, \tag{3.30}$$

which from then on will be taken as being zero. If the term (3.29) occurs again (or at least with the same summation sections), the first part will be eliminated, and assuming RK-AID is still in equation mode, will be given as

$$\sum_{i=4}^{s} b_i a_{i3}. \tag{3.31}$$

The conditions (3.30) $=0$ and (3.31) $=0$ will continue to be made for the rest of the model, including in error terms. It should be noted that the equations do not appear in any particular order under this scheme, and in some cases the model may be under-specified. As a simple example to illustrate this, consider applying the conditions

$$b_2 = b_3 = 0,\ Q_{i1} = 0,\ i = 4, \ldots, s,$$

to a set of RK order conditions up to fifth-order. This may be achieved with the RK-AID command file:

```
order :5, reduce
b(2:3)=0 & Q(4:s,1)=0
equation mode
use Q's
```

which produces the output:

1. $b_1+\sum_{i=4}^{s} b_i - 1$

---

2. $\sum_{i=4}^{s} b_i c_i - \frac{1}{2}$

---

3. $\sum_{i=4}^{s} b_i c_i^2 - \frac{1}{3}$

--- (3 / 4)

5. $\sum_{i=4}^{s} b_i c_i^3 - \frac{1}{4}$

7. $\sum_{i=4}^{s} b_i Q_{i2}$

8. $Q_{2,1}\sum_{i=4}^{s} b_i a_{i2}+Q_{3,1}\sum_{i=4}^{s} b_i a_{i3}$

--- (6 / 8)

9. $\sum_{i=4}^{s} b_i c_i^4 - \frac{1}{5}$

12. $\sum_{i=4}^{s} b_i c_i Q_{i2}$

13. $\sum_{i=4}^{s} b_i Q_{i3}$

14. $Q_{2,1}\sum_{i=4}^{s} b_i c_i a_{i2}+Q_{3,1}\sum_{i=4}^{s} b_i c_i a_{i3}$

15. $c_2 Q_{2,1}\sum_{i=4}^{s} b_i a_{i2}+c_3 Q_{3,1}\sum_{i=4}^{s} b_i a_{i3}$

16. $\sum_{i=4}^{s} b_i \sum_{j=2}^{i-1} a_{ij} Q_{j2}$

17. $\sum_{i=4}^{s} b_i \sum_{j=3}^{i-1} a_{ij} \sum_{k=2}^{j-1} a_{jk} Q_{k1}$

Total independent / Total : (13 / 17)

If this command set is re-run replacing `REDUCE` by `SPLIT`, expression (8) will be replaced by (3.30) and (15) replaced by (3.31). Equivalently, expression (14) will be given as

$$\sum_{i=4}^{s} b_i c_i a_{i2}.$$

The problem here is that no term is available for replacement by

$$\sum_{i=4}^{s} b_i c_i a_{i3},$$

and the model will be incorrect (the assumption that all summation parts must be zero is not valid) — this will be noticed, and a warning will be given. However, the first eight expressions (of which the fourth and sixth are dependent) will no longer represent a fourth-order model, for which no indication is output. In this way, care must be taken to check the equation subsets for validity — this problem does not, in fact, appear during the models of the later chapters. The 'split' form of the equation set above will take the form:

$$\begin{array}{rll}
1. & b_1 + \sum_{i=4}^{s} b_i - 1 & \\
\cline{1-2}
2. & \sum_{i=4}^{s} b_i c_i - \frac{1}{2} & \\
\cline{1-2}
3. & \sum_{i=4}^{s} b_i c_i^2 - \frac{1}{3} & \\
\cline{1-2}
 & & (3\ /\ 4) \\
5. & \sum_{i=4}^{s} b_i c_i^3 - \frac{1}{4} & \\
7. & \sum_{i=4}^{s} b_i Q_{i2} & \\
8. & \sum_{i=4}^{s} b_i a_{i2} & \\
\cline{1-2}
 & & (6\ /\ 8) \\
9. & \sum_{i=4}^{s} b_i c_i^4 - \frac{1}{5} & \\
12. & \sum_{i=4}^{s} b_i c_i Q_{i2} &
\end{array}$$

13. $\sum_{i=4}^{s} b_i Q_{i3}$
14. $\sum_{i=4}^{s} b_i c_i a_{i2}$
15. $\sum_{i=4}^{s} b_i a_{i3}$
16. $\sum_{i=5}^{s} b_i \sum_{j=4}^{i-1} a_{ij} Q_{j2}$
17. $\sum_{i=5}^{s} b_i \sum_{j=4}^{i-1} a_{ij} a_{j2}$

Total independent / Total : (13 / 17)

where equation (17) has been simplified by imposition of (3.31) to be

$$Q_{21} \sum_{i=5}^{s} b_i \sum_{j=4}^{i-1} a_{ij} a_{j2} + Q_{31} \sum_{i=5}^{s} b_i \sum_{j=4}^{i-1} a_{ij} a_{j3},$$

and has itself been split and replaced by its first summation section. This also contributes to an incomplete model specification, and is indicated (additional to the standard output warning) by the presence of a bracketed asterisk (`[*]`) alongside the missing expressions in the default "*rkive.out*" file:

```
b(i)a(i,2) := 0
b(i)c(i)a(i,2) := 0
b(i)a(i,3) := 0
b(i)a(i,j)a(j,2) := 0
b(i)c(i)a(i,3) := 0 [*]
b(i)a(i,j)a(j,3) := 0 [*]
```

where it can be seen that all automatic impositions have been logged in a form suitable for use in a subsequent command file.

Chapter 4

RK-IVE: an RK(N) formulae, usage and development history

4.1 Introduction

The use of RK and RKN methods for the approximate solution of general first-order and special second-order systems of initial value problems has been introduced. Chapter 2 was concerned with their applicability, and the conditions that must be considered for their use. Analysis and derivation of the fundamental governing (local) truncation error terms was made available through the algorithms of the RK-AID package in Chapter 3.

This chapter is intended as a review of existing RK and RKN processes, detailing the procedures involved in their derivation — as will be seen, acquisition of the order conditions is a necessary, but mainly routine, part of the work. The difficulty comes in their solution, which is far from clear except in simple cases. It will be assumed throughout that the abilities of mathematical software to reduce the need for mundane algebraic manipulation is accepted — not least by the quoting of RK-AID command files for presentation of typeset results and generation of relevant conditions and data. The single RK case is a fundamental and pre-requisite theory, and as such will be discussed first.

4.2 Single explicit RK processes

In 1895, Runge [2] first proposed that a sequence of s function evaluations $\boldsymbol{f}$ could be used to form a single-step numerical propagation of solutions to the general system of first-order ODE initial value problems (2.2) of Section 2.2. For an integer p, he proposed that a formula be derived such that a Taylor series expansion in the step length h of the approximation matches corresponding terms of the solution up to an order of h^p. In 1901, Kutta [4] derived the two Taylor expansions up to fifth-order, and was the first to solve the resulting *order conditions* for $p = 5$ using the method parameters of a six stage explicit RK. Butcher [9] established a unified approach to deriving the order conditions, which was widely accepted, and is now assumed via algorithms such as that in the RK-AID package.

4.2.1 Order conditions

The set of RK order conditions ${}^\dagger\tau_j^{(i)} = 0$ relating to $\tau_j^{(i)}$, for $j = 1, \ldots, r_i,\ i = 1, \ldots, p$, may initially be separated into classes. The first τ from each order, and its associated order condition, contains only b_i and c_i parameters and relates to the quadrature problem $\boldsymbol{y}'(x) = \boldsymbol{f}(x)$, thus ${}^\dagger\tau_1^{(i)} = 0,\ i = 1, \ldots, p$, are termed *quadrature equations*, taking the form

$$ {}^\dagger\tau_1^{(k)} = \sum_{i=1}^{s} b_i c_i^{k-1} - \frac{1}{k} = 0,\ k = 1, \ldots, p. \tag{4.1}$$

For an explicit s-stage RK, remaining conditions

$$ {}^\dagger\tau_j^{(i)} = 0,\ i = 3, \ldots, p,\ j = 2, \ldots, r_i,$$

contain parameters

$$ b_i,\ i = s^b, \ldots, s, \tag{4.2}$$

$$ a_{ij},\ i = 3, \ldots, s,\ j = s_i^a, \ldots, i-1, \tag{4.3}$$

and

$$ c_i,\ i = 2, \ldots, s^c, \tag{4.4}$$

for some $s_i^a \geq 2$, $s^b \geq 2$ and $s^c \leq s$. The a_{i1}'s are only present in the row-sum conditions (2.6) of Section 2.3, and are thus determined trivially as

$$a_{i1} = c_i - \sum_{j=2}^{i-1} a_{ij}, \; i = 2, \ldots, s.$$

The last τ of each order $\tau_{r_i}^{(i)}$ relates to the linear, constant coefficient, problem

$$\boldsymbol{y}' = M\boldsymbol{y} + \boldsymbol{m}x,$$

for M a matrix, and $\boldsymbol{m}$ a vector of numerical constants, and as such is referred to as the i^{th} order *linear* τ. Note that this is the coefficient referred to before the τ ordering scheme was established in the absolute stability definitions (2.3.3) and (2.5.2), and so $l_i = r_i$ for completeness.

$$\begin{pmatrix} c_2 & c_3 & \ldots & c_s \\ c_2^2 & c_3^2 & \ldots & c_s^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_2^{s-1} & c_3^{s-1} & \ldots & c_s^{s-1} \end{pmatrix} \begin{pmatrix} b_2 \\ b_3 \\ \vdots \\ b_s \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{3} \\ \vdots \\ \frac{1}{s} \end{pmatrix}$$

Figure 4.1: RK quadrature system in the case $p = s$

4.2.2 Solution of order conditions for $p \leq 4$

For this range of orders, there are 2^{p-1} order conditions to be solved. A p stage formula contains $p(p+1)/2$ parameters, and with $p(p+1)/2 \geq 2^{p-1}$, there are enough parameters to solve for an RK_p^p process, noting that explicit RK processes cannot have order $> s$ due to the 'collapsing' of the linear τ in order $s+1$, i.e.,

$$\tau_{r_{s+1}}^{(s+1)} = -1/(s+1)!, \tag{4.5}$$

where $s_i^a = i,\ s^b = s+1$, and $s^c = 1$ in (4.2), (4.3) and (4.4). In these cases, the number of quadrature equations is equal to the number of stages and thus form an $(s-1)^{th}$ order Vandermode[1] system (Figure 4.1) which, assuming distinct matrix columns (c_i's), may commonly be solved for the b_i in terms of the $c_i,\ i = 2, \ldots, s$, with b_1 being determined from

$$b_1 = 1 - \sum_{i=2}^{s} b_i.$$

Remaining equations are then solved with the a_{ij}'s and c_i's. In the case of a $p = 1$ method, $b_1 = 1$ is sufficient and is the first-order Taylor algorithm

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h_n \boldsymbol{f}(x_n, \boldsymbol{y}_n),$$

known as Euler's method [1]. For $p > 1$, however, *families* of methods are determined in terms of free parameters, and processes must be chosen to have specific properties by a decision made on the values of these parameters. For example, any RK_2^2 process is a member of the one parameter family

$$b_1 = 1 - \frac{1}{2c_2}, \text{ and } b_2 = \frac{1}{2c_2},\ c_2 \neq 0,$$

where $a_{21} = c_2$. The classical formulae are

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array} \quad \text{and} \quad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

the latter being known as the improved Euler method, or Heun's method [3], which for a single ODE quadrature problem reduces to the Trapezoidal rule. Both of these formulae have as simple a coefficient set as is possible, this having been of great

[1]An $n \times n$ matrix formed by rows of increasing component-wise powers of a transposed vector $(x_1, x_2, \ldots, x_n)^T$.

importance to practitioners before the advent of electronic numerical computation. However, the family has $\tau_1^{(3)} = (3c_2 - 2)/10$, and $\tau_2^{(3)} = -\frac{1}{6}$, thus a choice of c_2 might better be based on the (absolute) size of these *principal* error terms.

A value of $c_2 = 2/3$ can be considered 'optimal' as it eliminates the third-order quadrature term and is minimal for any reasonable average of the two principal local truncation error terms. Other factors in a choice of c_2 may include secondary error terms (fourth-order in this case) or the region of absolute stability, though for any RK_p^p algorithm, from the definition (2.3.3) and property (4.5), the real negative stability limit is a function of p only, and so is not applicable here.

For $p = 3$, the general family of methods is in terms of c_2 and c_3, and so even more freedom of choice is available.

The 'classic' RK_4^4 process of Kutta [4] is defined by the tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{4.6}$$

and is the most famous of all RK formulae, perhaps due to its very simple appearance. The a_{ij} matrix being sub-diagonal makes the function evaluations very simple to calculate, and as such is probably the highest order method ever applied without the use of a computer nowadays. Truncation error analysis may conveniently be done using RK-AID as follows:

```
a(2,1)=1/2
a(3,1:2)=0; 1/2
a(4,1:3)=0; 0; 1
b(1:4)=1/6; 1/3; 1/3; 1/6
exact arithmetic, exact output
check, order :5
```

giving

1. $\tau_1^{(1)} = \sum_i b_i - 1 = 0$

---

2. $\tau_1^{(2)} = \sum_i b_i c_i - \dfrac{1}{2!} = 0$

---

3. $\tau_1^{(3)} = \frac{1}{2}\sum_i b_i c_i^2 - \dfrac{1}{3!} = 0$

4. $\tau_2^{(3)} = \sum_{ij} b_i a_{ij} c_j - \dfrac{1}{3!} = 0$

---

5. $\tau_1^{(4)} = \frac{1}{6}\sum_i b_i c_i^3 - \dfrac{1}{4!} = 0$

6. $\tau_2^{(4)} = \sum_{ij} b_i c_i a_{ij} c_j - \dfrac{3}{4!} = 0$

7. $\tau_3^{(4)} = \frac{1}{2}\sum_{ij} b_i a_{ij} c_j^2 - \dfrac{1}{4!} = 0$

8. $\tau_4^{(4)} = \sum_{ijk} b_i a_{ij} a_{jk} c_k - \dfrac{1}{4!} = 0$

---

9. $\tau_1^{(5)} = \frac{1}{24}\sum_i b_i c_i^4 - \dfrac{1}{5!} = \dfrac{1}{2880}$

10. $\tau_2^{(5)} = \frac{1}{2}\sum_{ij} b_i c_i^2 a_{ij} c_j - \dfrac{6}{5!} = \dfrac{1}{480}$

11. $\tau_3^{(5)} = \frac{1}{2}\sum_{ijk} b_i a_{ij} c_j a_{ik} c_k - \dfrac{3}{5!} = \dfrac{1}{160}$

12. $\tau_4^{(5)} = \frac{1}{2}\sum_{ij} b_i c_i a_{ij} c_j^2 - \dfrac{4}{5!} = -\dfrac{1}{480}$

13. $\tau_5^{(5)} = \frac{1}{6}\sum_{ij} b_i a_{ij} c_j^3 - \dfrac{1}{5!} = -\dfrac{1}{720}$

14. $\tau_6^{(5)} = \sum_{ijk} b_i c_i a_{ij} a_{jk} c_k - \dfrac{4}{5!} = \dfrac{1}{120}$

15. $\tau_7^{(5)} = \sum_{ijk} b_i a_{ij} c_j a_{jk} c_k - \dfrac{3}{5!} = -\dfrac{1}{240}$

16. $\tau_8^{(5)} = \frac{1}{2}\sum_{ijk} b_i a_{ij} a_{jk} c_k^2 - \dfrac{1}{5!} = \dfrac{1}{480}$

17. $\tau_9^{(5)} = \sum_{ijkl} b_i a_{ij} a_{jk} a_{kl} c_l - \dfrac{1}{5!} = -\dfrac{1}{120}$

For larger numbers of principal truncation error terms, as here, a measure of the accuracy of a formula commonly used is an average over the absolute values of $\tau_i^{(p+1)},\ i = 1, \ldots, r_{p+1}$. A variety of norms are applicable, the most common is that of the 2-norm, i.e.,

$$\|\boldsymbol{\tau}^{(p+1)}\|_2 = \sqrt{\sum_{i=1}^{r_{p+1}} (\tau_i^{(p+1)})^2},$$

where the symbol $\boldsymbol{\tau}^{(p+1)}$ refers to the vector comprised of $\tau_i^{(p+1)}$, $i = 1, \ldots, r_{p+1}$. By inclusion of the command `2 norm` to the RK-AID data-file above, the 2-norm of the principal terms (denoted by `T5`) is calculated[2] and displayed to the standard output as

```
 4'th order process
 ------------------
  (T5: 1.45E-02)
```

For order $p > 4$, $s > p$ stages are required for the derivation, and the complexity increases dramatically. In order to motivate the remaining discussion of this chapter, it is now convenient to review a selection from the entirety of known single RK's.

4.2.3 Review of single RK formulae

For an RK_s^p algorithm $p = 1, \ldots, 13$, Table 4.2 gives the number of elementary differentials r_p, the total number of elementary differentials N_p, an example of a number

p	1	2	3	4	5	6	7	8	9	10	11	12	13
r_p	1	1	2	4	9	20	48	115	286	719	1842	4766	12486
N_p	1	2	4	8	17	37	85	200	486	1205	3047	7813	20299
s	1	2	3	4	6	7	9	11	14	17	-	-	-
D_s	1	3	6	10	21	28	45	66	105	153	-	-	-

Table 4.2: r_p, N_p, s and D_s for an RK_s^p, $p = 1, \ldots, 13$.

of stages known for their solution s, and $D_s = s(s+1)/2$, the number of available parameters in s stages. The non-existence of the RK_5^5 and RK_{10}^8 processes (Butcher [9], [24] and [25]) means the RK_{11}^8 algorithms due to Curtis [26] and Cooper and Verner

[2]Such calculations are only done approximately, despite the exact arithmetic.

[27] are in the minimal number of stages. The RK_{17}^{10} algorithm is due to Hairer [28]. No single process of order > 10 is known at present.

For $p > 5$, it is clear that the required number of equations for each order can be solved with far fewer parameters than would be expected from a general nonlinear system. This suggests patterns or groupings of the equations and parameters that can simplify the solution of these rapidly increasing numbers of order conditions — such schemes are discussed next.

4.2.4 Butcher simplifying schemes

Butcher [29] characterised families of minimal s-stage *implicit* methods of order $2s$. This technique motivated the schemes used for developing high-order ($p > 5$) explicit RK's, so as an example, a three-stage implicit sixth-order formula can be derived as follows:

Example 4.2.1 (A three-stage, sixth-order implicit RK) A sufficient equation set can be obtained using the RK-AID commands

```
implicit, order :6
parameterise in 3 stages
```

which, by default, uses the Butcher 'Q' parameter families (3.7) of Section 3.2.3, giving:

1. $\sum_{i=1}^{3} b_i - 1$
2. $\sum_{i=1}^{3} b_i c_i^k - \frac{1}{(k+1)}, \quad k = 1:5$
3. $\sum_{i=1}^{3} b_i c_i^l Q_{i1}^k, \quad k = 1:2, \quad 2k + l < 6$
4. $\sum_{i=1}^{3} b_i c_i^k Q_{i2}, \quad k = 0:2$
5. $\sum_{i=1}^{3} b_i c_i^m \sum_{j=1}^{3} a_{ij} c_j^n Q_{j1}^l, \quad l = 1:2, \quad 2l + m + n < 5$

6. $\sum_{i=1}^{3} b_i c_i^k Q_{i3}, \quad k = 0:1$
7. $\sum_{i=1}^{3} b_i c_i^l \sum_{j=1}^{3} a_{ij} c_j^m Q_{j2}, \quad l+m<2$
8. $\sum_{i=1}^{3} b_i c_i^m \sum_{j=1}^{3} a_{ij} c_j^n \sum_{k=1}^{3} a_{jk} c_k^o Q_{k1}, \quad m+n+o<2$
9. $\sum_{i=1}^{3} b_i Q_{i2} Q_{i1}$
10. $\sum_{i=1}^{3} b_i Q_{i4}$
11. $\sum_{i=1}^{3} b_i Q_{i1} \sum_{j=1}^{3} a_{ij} Q_{j1}$
12. $\sum_{i=1}^{3} b_i \sum_{j=1}^{3} a_{ij} Q_{j3}$
13. $\sum_{i=1}^{3} b_i \sum_{j=1}^{3} a_{ij} \sum_{k=1}^{3} a_{jk} Q_{k2}$
14. $\sum_{i=1}^{3} b_i \sum_{j=1}^{3} a_{ij} \sum_{k=1}^{3} a_{jk} \sum_{l=1}^{3} a_{kl} Q_{l1}$

By using *all* of $a_{ij},\ i,j=1,2,3$, to satisfy

$$Q_{ik} = \sum_{j=1}^{3} a_{ij} c_j^k - \frac{c_i^{k+1}}{k+1} = 0, \text{ for } i=1,2,3,\ k=0,1,2, \tag{4.7}$$

where $Q_{i0}=0$ are the usual row-sum conditions, an alternative representation of remaining conditions using 'R' parameters (3.9) of Section 3.2.3, can be generated using

```
implicit, order :6
use Q's & R's
Q(1:s, 1:2)=0
reduce in 3 stages
equation mode, continuous numbering
```

giving the equation set in Figure 4.2. This highlights the dependence of 'Q' and 'R'

1. $\sum_{i=1}^{3} b_i - 1$

2. $\sum_{i=1}^{3} b_i c_i - \frac{1}{2}$

3. $\sum_{i=1}^{3} b_i c_i^2 - \frac{1}{3}$

$*$ 4. $\sum_{i=1}^{3} R_{0i} c_i$

(4 / 5)

5. $\sum_{i=1}^{3} b_i c_i^3 - \frac{1}{4}$

$*$ 6. $\sum_{i=1}^{3} R_{1i} c_i$

$*$ 7. $\sum_{i=1}^{3} R_{0i} c_i^2$

(7 / 11)

8. $\sum_{i=1}^{3} b_i c_i^4 - \frac{1}{5}$

$*$ 9. $\sum_{i=1}^{3} R_{2i} c_i$

$*$ 10. $\sum_{i=1}^{3} R_{1i} c_i^2$

11. $\sum_{i=1}^{3} b_i Q_{i3}$ or $\sum_{i=1}^{3} R_{0i} c_i^3$

(11 / 23)

12. $\sum_{i=1}^{3} b_i c_i^5 - \frac{1}{6}$

$*$ 13. $\sum_{i=1}^{3} R_{3i} c_i$

$*$ 14. $\sum_{i=1}^{3} R_{2i} c_i^2$

15. $\sum_{i=1}^{3} b_i c_i Q_{i3}$ or $\sum_{i=1}^{3} R_{1i} c_i^3$

16. $\sum_{i=1}^{3} b_i Q_{i4}$ or $\sum_{i=1}^{3} R_{0i} c_i^4$

Total independent / Total : (16 / 46)

Figure 4.2: A reduced set of order conditions for an implicit sixth-order RK.

parameters, with equations of the form

$$\sum_{i=1}^{3} b_i c_i^k Q_{ij} = 0$$

being alternatively written as

$$\sum_{i=1}^{3} R_{ki} c_i^j = 0.$$

For $j = 1, 2,\ k = 0, 1, \ldots$ these are solved by conditions (4.7) and so are marked with an asterisk (*). Remaining equations are then as follows:

$$\sum_{i=1}^{3} b_i c_i^k = \frac{1}{k+1}, \quad k = 0, \ldots, 5, \tag{4.8}$$

and

$$\begin{aligned} \sum_{i=1}^{3} b_i Q_{i3} = 0 \quad &\text{or} \quad \sum_{i=1}^{3} R_{0i} c_i^3 = 0, \\ \sum_{i=1}^{3} b_i c_i Q_{i3} = 0 \quad &\text{or} \quad \sum_{i=1}^{3} R_{1i} c_i^3 = 0, \\ \sum_{i=1}^{3} b_i Q_{i4} = 0 \quad &\text{or} \quad \sum_{i=1}^{3} R_{0i} c_i^4 = 0, \end{aligned} \tag{4.9}$$

where (4.8) may be solved uniquely using three-point Gaussian quadrature, i.e.,

$$c_1 = \frac{5-\sqrt{15}}{10},\ c_2 = \frac{1}{2},\ c_3 = \frac{5+\sqrt{15}}{10},\ \text{and } b_1 = \frac{5}{18},\ b_2 = \frac{4}{9},\ b_3 = \frac{5}{18}.$$

Equations (4.9) are now automatically solved, as

$$\begin{pmatrix} 1 & 1 & 1 \\ c_1 & c_2 & c_3 \\ c_1^2 & c_2^2 & c_3^2 \end{pmatrix} \begin{pmatrix} R_{k1} \\ R_{k2} \\ R_{k3} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},\ k = 0, 1,$$

implies $R_{0i} = R_{1i} = 0,\ i = 1, 2, 3$, by non-singularity of the Vandermode system. Note requirement of an order $k+2$ quadrature formula to give the assumed condition

$$\sum_{i=1}^{s} R_{ki} = 0.$$

After determination of the a_{ij}'s from (4.7), the final tableau becomes:

$$\begin{array}{c|ccc}
\frac{5-\sqrt{15}}{10} & \frac{5}{36} & \frac{10-3\sqrt{15}}{45} & \frac{25-6\sqrt{15}}{180} \\
\frac{1}{2} & \frac{10+3\sqrt{15}}{72} & \frac{2}{9} & \frac{10-3\sqrt{15}}{72} \\
\frac{5+\sqrt{15}}{10} & \frac{25+6\sqrt{15}}{180} & \frac{10+3\sqrt{15}}{45} & \frac{5}{36} \\
\hline
 & \frac{5}{18} & \frac{4}{9} & \frac{5}{18}
\end{array}$$

□

The scheme of Example 4.2.1 is easily extended to a method of any (even) order. In general, for an s stage implicit process, imposing the conditions

$$Q_{ik} = 0,\ i = 1, \ldots, s,\ k = 0, \ldots, s-1, \tag{4.10}$$

using the a_{ij}'s, and solving a Gaussian quadrature rule to determine the b_i's and c_i's, will always yield a unique process of order $2s$, due to the automatic satisfaction of

$$R_{kj} = 0,\ j = 1, \ldots, s,\ k = 0, 1, \ldots. \tag{4.11}$$

While such simplification schemes are not directly applicable to explicit methods, elements of this approach prompted the derivation of many explicit RK (and RKN) processes in common usage — not least, development of *embedded* formulae was made more accessible. The following section deals with the application of Butcher simplifying schemes to such pairs of RK methods.

4.3 Explicit RK pairs

The difficulty in deriving embedded pairs of methods is that two sets of order conditions must be solved for the main integrator and step size control formula concurrently, i.e., for an $\mathrm{RK}_s^{q(p)}$ pair, the order conditions are

$$^{\dagger}\tau_j^{(i)} = 0,\ j = 1, \ldots, r_i,\ i = 1, \ldots, p, \tag{4.12}$$

and

$$^{\dagger}\widehat{\tau}_j^{(i)} = 0,\ j = 1, \ldots, r_i,\ i = 1, \ldots, q. \tag{4.13}$$

The only difference between (4.12) and (4.13) for $i = 1, \ldots, p$, $(p < q)$ is the replacement of b_i by $\widehat{b}_i$ in each condition. The technique is therefore to derive a main integrator that contains the possibility of an alternative set of external weights that, with the same internal weights and nodes, gives a formula of a lower order, preferably $p = q - 1$ so the error estimate is that of an approximation not too far from the order of the integrator.

4.3.1 Simplification of embedded RK order conditions

The Butcher simplifying technique for implicit processes, i.e., using all a_{ij} parameters to satisfy (4.10), is independent of the external weights that are to provide the final approximation and so could ideally be used to solve non-quadrature order conditions for both of the two embedded processes of an explicit pair. The difficulty is that Q_{ik} parameters contain only a_{ij} for $j = 1, \ldots, i-1$, and so for a given i, the conditions $Q_{ik} = 0$ can only be made for a limited range of k values. For example, $Q_{1k} = -c_1^{k+1}/(k+1) = 0$ as $c_1 = 0$, but $Q_{2k} = -c_2^{k+1}/(k+1)$, and so cannot be zero unless $c_2 = 0$. Non-distinct c_i's of this kind will mean degeneracy of the quadrature systems for the b_i's and $\widehat{b}_i$'s, and so is restricting. It is possible, however, to make conditions of the form

$$Q_{ik} = 0,\ i = 3, \ldots, s,$$

for any one value of k. $Q_{3k} = a_{32}c_2^k - c_3^{k+1}/(k+1) = 0$ can be solved with either a_{32}, c_2 or c_3, and $Q_{4k} = a_{42}c_2^k + a_{43}c_3^k - c_4^{k+1}/(k+1) = 0$ with either a_{42}, a_{43}, c_2, c_3 or c_4, etc. If this simplification is made for $k = 1$, with $p > 2$ the conditions

$$\sum_{i=2}^{s} b_i c_i^k Q_{i1} = 0, \;\; k = 0, \ldots, p-3$$

reduce to $b_2 c_2^k Q_{21} = 0$, and b_2 must be zero (assuming $c_2 \neq 0$). Thus it is possible to use Butcher simplification schemes for elimination of groups of order conditions under certain restrictions, such as $b_2 = 0$ above. Assuming this restriction has been made for both processes, i.e., $b_2 = \widehat{b}_2 = 0$, order conditions are simplified for the order q and order p formulae, as required. Remaining a_{ij} parameters for each of the stages $i = 3, \ldots, s$, can be used to further impose $Q_{ik} = 0$ for $k > 1$. Difficulty arises for the low stages when more than one 'Q' parameter family is zeroed, for example, if $Q_{31} = Q_{32} = 0$ is applied, a_{32} is commonly used, and one of c_2 or c_3 must be constrained to the other, i.e., $a_{32} = c_3^2/(2c_2)$ solves $Q_{31} = 0$, and $c_2 = 2c_3/3$ solves $Q_{32} = 0$, and so $a_{32} = 3c_3/4$. For $p < 6$, no such simplification techniques are *necessary*, but for $p \geq 6$ they are required and invaluable.

Frequently, the $\widehat{R}_{0j}$, $j = 1, \ldots, s$, parameter family can be zeroed to solve many of the order conditions for the order q process. For example,

$$\widehat{R}_{0s} = -\widehat{b}_s(1 - c_s) = 0,$$

requires that $c_s = 1$ (assuming $\widehat{b}_s \neq 0$, i.e., no FSAL), and

$$\widehat{R}_{0,s-1} = \widehat{b}_s a_{s,s-1} - \widehat{b}_{s-1}(1 - c_{s-1}) = 0$$

can be solved with $\widehat{b}_s$, $\widehat{b}_{s-1}$, c_{s-1} or $a_{s,s-1}$. Conventionally, the 'high a_{ij}' parameters, with $i = s$, are used for this purpose, i.e., each of $\widehat{R}_{0j} = 0$, $j = 2, \ldots, s-1$ can be solved with a_{sj}, $j = 2, \ldots, s-1$. Equivalent conditions for the order p formula may often be solved concurrently, using 'R_{0j}' parameters, under certain restrictions such as $c_{s-1} = 1$, use of FSAL, etc. For a more detailed description of such techniques see, for example, Prince [12].

It should be noted that the use of 'R conditions' applied to *single* formulae are of primary concern when deriving high-order RK processes. The RK_{17}^{10} of Hairer [28] uses such schemes in an extremely complex manner, and the difficulty of embedding over such models is still a major challenge in the development of high-order RK pairs.

4.3.2 Review of embedded RK formulae

Runge [2] included an $\mathrm{RK}_2^{2(1)}$ pair in his introductory work, but for higher orders, Merson [30] was the first to illustrate the challenge by deriving an $\mathrm{RK}_5^{4(3)}$ pair, which was effectively an $\mathrm{RK}_5^{4(5)}$ for linear problems ($\tau_4^{(4)} = \tau_9^{(5)} = 0$), and as such was proposed to provide an error estimate for the fourth-order integrator. This is only 'correct' for linear problems, but the method has been used widely. Soon after the pioneering work of Butcher, further embedded methods without this restriction were constructed by Sarafyan [31], England [32] and Fehlberg ([33] and [34]). The Fehlberg pairs are all of $p = q - 1$ design, for the cases $q = 5, \ldots, 9$. Of these, the high-order processes, commonly known as RKF7 and RKF8 (designed for use in lower-order mode) are still in usage today, though all suffer from a deficiency in their error estimate under certain circumstances. Namely, the principal quadrature terms for the lower-order formula in all these cases is zero. The error estimate is therefore not valid for the quadrature or 'near-quadrature' problem, where the dependence on $\boldsymbol{y}(x)$ for the problem is zero or small. The number of stages he used for the derivation, however, has remained competitive.

The challenge in producing RK pairs has since not so much been the solution of order conditions, but doing so in *generality*, maintaining the number of free parameters for optimising the propagating formula and 'tuning' the error estimate. Much work has been done on producing minimal stage, accurate, efficient pairs (with asymptotically correct error estimates), mostly for application in higher-order mode (for example, Butcher [35], Prince [12] and Verner [36]). Such work has involved consideration of the following quantities (adopting the notation of [12]):

$$\begin{gathered} A_N^{(q+1)} = \|\hat{\boldsymbol{\tau}}^{(q+1)}\|_N, \; B_N^{(p+2)} = \frac{\|\boldsymbol{\tau}^{(p+2)}\|_N}{\|\boldsymbol{\tau}^{(p+1)}\|_N}, \\ C_N^{(p+2)} = \frac{\|\boldsymbol{\tau}^{(p+2)} - \hat{\boldsymbol{\tau}}^{(p+2)}\|_N}{\|\boldsymbol{\tau}^{(p+1)}\|_N}, \end{gathered} \tag{4.14}$$

where N is a variety of norm, i.e., 1, 2 or ∞. $A_N^{(q+1)}$ is thus the principal gauge of accuracy for a higher-order mode pair, providing that $A_N^{(q+2)}$ does not dominate (i.e., $A_N^{(q+2)}/A_N^{(q+1)}$ is 'small'). The quantities $B_N^{(p+2)}$ and $C_N^{(p+2)}$ are ratios of secondary to principal terms for the lower-order formula and error estimate, respectively[3]. These quantities should necessarily be 'small' to maintain validity of the error estimate. However, as the above are terms of an asymptotic expansion in h_n, 'small' in each case depends on the range of step sizes to be used when the method is applied. Thus, in this sense, formulae can be derived specifically to perform well for 'lax tolerance' or 'stringent tolerance' usage, by choice of these ratios for higher-order, lower-order, and error estimate norms. Also, the requirement for global errors to behave predictably with variation of a tolerance parameter, and hence range of step sizes, relies on $\|\boldsymbol{\tau}^{(p+1)}\|_N$ not being too big, although this would imply $B_N^{(p+2)}$ and $C_N^{(p+2)}$ being 'small'. Further considerations include the size of the real negative stability limits for the pair, avoidance of any of the $\tau_i^{(p+1)}$, $i = 1, \ldots, r_{p+1}$, being zero (the generalised 'quadrature problem'), and for the method parameters not to be too large ($\gg 1$), such that rounding effects in a finite precision machine are avoided. Owing to the obvious compromises that may have to be made, the final decision relating to choice of free parameters under these considerations tends to be done experimentally, solving test problems with an available true (or accurate) solution over ranges of a local error tolerance. Such tools are the DETEST package of Hull, et al. [13] and the test suite in RK-AID, based on a modified DETEST subset from Prince [12].

Specifically, two high-order RK pairs developed using these criterion are of most interest, as they are considered in the following chapters — the RK8(7)13M (an

[3]In the case where $q > p+1$, $B_N^{(p+2)}$ and $C_N^{(p+2)}$ are identical.

$\mathrm{RK}_{13}^{8(7)}$) of Prince [12] and its close relative, the Dormand and Prince RK8(6)12M (an $\mathrm{RK}_{12}^{8(6)}$ — published in Hairer, et al. [37]). Particular specification of these models is contained in Chapter 7.

4.4 Explicit RK triples

The derivation of RK pairs with the addition of a continuous order p^* approximation is a relatively new subject ([38], [39], [40], [41], [42], [43], [44], [45], [46], [47], and [48]). It is clear that in the case where an existing pair is to be used for the discrete approximation, the problem is equivalent to deriving an order p^* polynomial interpolant for the discrete data — over one or several steps. The problems of interpolating over multiple steps (to save extra function evaluations) are such that it will not be discussed any further (see, for example, Gladwell [38]).

The technique of using external linear combinations of existing, and extra, stages within each RK step was first presented by Horn [39], and has been formulated within a general RK framework by embedding (Section 2.5). It is therefore solution of

$$^{\dagger}\tau_j^{(i)*} = 0,\ j = 1, \ldots, r_i,\ i = 1, \ldots, p^*, \tag{4.15}$$

that presents the challenge. Following Dormand and Prince [14], this may be referred to as a 'mode A' approach, and derivation of an interpolant (directly) a 'mode B' approach. These two strategies are shown to be necessarily equivalent in certain cases. Where no existing pair is employed, the problem may best be thought of as solving (4.12), (4.13) and (4.15) concurrently.

4.4.1 Continuous RK order conditions

The conditions (4.15) may easily be obtained by substituting:

$$b_i^* \text{ for } b_i/\widehat{b}_i,\ c_i/\sigma \text{ for } c_i,\ a_{ij}/\sigma \text{ for } a_{ij}, \text{ and } Q_{ik}/\sigma^{k+1} \text{ for } Q_{ik}$$

in the standard discrete order conditions. For example, terms up to fourth-order take the form:

```
[s] order :4, parameterise
```

1. $\sum_{i=1}^{s^*} b_i^* - 1$
2. $\sum_{i=2}^{s^*} b_i^* c_i^k - \frac{\sigma^k}{k+1}, \quad k = 1:3$
3. $\sum_{i=2}^{s^*} b_i^* c_i^k Q_{i1}, \quad k = 0:1$
4. $\sum_{i=2}^{s^*} b_i^* Q_{i2}$
5. $\sum_{i=3}^{s^*} b_i^* \sum_{j=2}^{i-1} a_{ij} Q_{j1}$

Solution of such terms to zero may therefore be aided using Butcher simplifying techniques parallel to the discrete case. Specifically, when interpolating an existing discrete formula pair, if conditions of the form

$$b_i = \widehat{b}_i = 0,\ i = 2, \ldots, s^b - 1,\ Q_{ik} = 0,\ i = s^b, \ldots, s,$$

had been made (for some k and $s^b > 2$) in the discrete model, the same order conditions will be eliminated from the continuous model by making the natural conditions

$$b_i^* = 0,\ i = 2, \ldots, s^b - 1, \text{ and } Q_{ik} = 0,\ i = s + 1, \ldots, s^*.$$

In some cases this is not *necessary* to obtain a solution, but it serves to illustrate the technique. Due to the orthogonal nature of the non-quadrature conditions when using 'Q' forms, continuous quadrature conditions can only be solved using $b_i^*,\ i = 1, \ldots, s^*$. Remaining b_i^*'s can then be used to solve non-quadrature equations, and if others remain, they must be solved as consistency conditions using some of the a_{ij}'s and c_i's for $j = 2, \ldots, i-1,\ i = s+1, \ldots, s^*$. Assuming a reusable stage, C^1 continuity is guaranteed if all b^*'s are determined by the above procedure (see, for example, Owren and Zennaro [48], or Verner [45]). If this is not the case, continuity conditions must be solved for each free b_i^*, with reference to (2.21) from Section 2.5.2.

4.4.2 Review of continuous RK formulae

Techniques for providing dense output formulae have been primarily motivated by efficiency requirements. An RK interpolant will be globally order q if $p^* \geq q-1$ (Dormand and Prince [14]), and so this could be considered as a minimum requirement for its derivation irrespective of extra stages being required. The approach of Bogacki and Shampine [43] is to derive the highest order interpolant possible *without* adding extra (non-reusable) stages. An error estimate is provided for the continuous approximation which may then be used to implement the dense output for maximum efficiency. The RK8(7)13M (an $\mathrm{RK}_{13}^{8(7)}$) of Prince [12] was considered, a $p^*=5$ interpolant being found possible with $s^*=14$. For the discrete non-mesh values

$$\sigma = \frac{7 \pm \sqrt{7}}{14},$$

a sixth-order result with no extra function evaluations allows a free error estimate in the continuous fifth-order formula at these points.

In the more standard approach, $\mathrm{RK}_{7f}^{5(4)}$ pairs have been given fourth-order and fifth-order interpolants in up to two extra stages (Enright, et al. [41], Dormand and Prince [14], and Calvo, et al. [44]). Also, $\mathrm{RK}_8^{5(4)}\mathrm{D}_8^5$ and $\mathrm{RK}_{8f}^{5(4)}\mathrm{D}_8^5$ triples have been derived in the absence of an underlying pair by Owren and Zennaro ([47], [48]) in the minimal number of stages. A Dormand and Prince $\mathrm{RK}_{12}^{8(6)}\mathrm{D}_{16}^7$ triple is published in Hairer, et al. [37], based on the RK8(6)12M pair, dense output for which is also considered in Chapter 7. Interpolants up to ninth-order can be found in, for example, Enright, et al. [41] and Verner [45]. In all these cases, globally C^1 continuous dense output was considered as standard by inclusion of a reusable stage if not already present in an underlying FSAL pair.

The problem, as in the discrete case, is to derive the formula whilst keeping as many of the free parameters available for an optimisation procedure. Other than the order of the continuous process, accuracy of dense approximations can be gauged in different ways, those found in the literature are mainly based on a local error consideration. For example, Calvo, et al. [44] use the quantity

$$N_s = \int_0^1 \sigma^{p^*+1} \|\boldsymbol{\tau}^{(p^*+1)*}\|_2 \, d\sigma$$

as a gauge of the principal terms in the local error expansion of the continuous formula. This is essentially a double average over $\sigma \in [0,1]$ and the principal τ^*'s, and is therefore a local error gauge equivalent to the discrete case extended to all σ values. Bogacki and Shampine [43] require only that the integrand of N_s be 'small for $\sigma \in [0,1]$' (i.e., an average over σ is not specified). Whatever principal truncation error gauge is being used, secondary terms can be considered also, and again, the ratio of secondary to principal terms should not be overlooked. Equivalent to the discrete case, the size of the extra parameters should be kept as small as possible[4], though this is perhaps only of secondary importance, as any machine rounding errors are not propagated.

4.5 Single explicit RKN processes

In 1925, Nyström [5] first developed direct numerical methods for exploiting the special nature of ODE initial value problems lacking a first derivative. Derivation of such methods is similar to the RK case, complicated only by the *two* external approximations, and therefore having two sets of order conditions. It will be seen that, in the discrete case, imposition of a simplifying assumption relating the external weights of the two processes eliminates this complexity, and progress is familiar.

[4]Where the b_i^* polynomial *coefficients* are important, rather than the size of any continuous quantity.

4.5.1 Simplification of RKN order conditions

Derivation of an order p, s-stage process involves concurrent solution of the following order conditions

$$^{\dagger}\tau_j^{(i)} = 0,\ j = 1, \ldots, n_{i-1},\ i = 2, \ldots, p, \tag{4.16}$$

and

$$^{\dagger}\tau_j'^{(i)} = 0,\ j = 1, \ldots, n_i,\ i = 1, \ldots, p. \tag{4.17}$$

There exists a simple relationship between the $^{\dagger}\tau_j^{(i)}$ and $^{\dagger}\tau_j'^{(i)}$'s such that imposition of

$$b_i = (1 - c_i)b_i',\ i = 1, \ldots, s, \tag{4.18}$$

solves the conditions (4.16) if (4.17) are satisfied, thus it is never necessary to consider the 'b' formula and (4.18) will be assumed throughout. In this case, as the number of elementary differentials for the RKN case $n_i \leq r_i$ (for the RK case), and with the dependence of many τ's resulting from the row-sum condition assumption, in a given number of stages RKN processes may have a higher order than for an equivalent RK.

The order conditions (4.16) and (4.17) may be organised into several categories as for the RK case, and many parallels exist. Again, the terms derived from trees of height 1 relate to integration of $\boldsymbol{y}''(x) = \boldsymbol{f}(x)$, and $^{\dagger}\tau_1^{(i)} = 0$ and $^{\dagger}\tau_1'^{(i)} = 0$ are the quadrature equations. Also, the last coefficients from each order, i.e., $\tau_{n_{i-1}}^{(i)}$ and $\tau_{n_i}'^{(i)}$, relate to linear constant coefficient problems[5].

4.5.2 Review of single RKN formulae

In his introductory work, Nyström derived single RKN formulae of order p with $s = p - 1$, for $p = 3, 4$ and 5. Thirty years later, Albrecht [49] solved the equations of condition for an RKN$_5^6$. Battin [50] proved the non-existence of the RKN$_6^7$ algorithm, and provided the first RKN$_7^7$ and RKN$_8^8$ processes (see also Hairer [51]). The highest order, minimal stage, single RKN process is of order ten, due to Hairer [23]. For

[5]Therefore, $l_i^n = n_i$ in the absolute stability definition (2.7.1).

reference, the number of elementary differentials n_p, the total number of elementary differentials T_p, the number of independent truncation error coefficients i_p and the total number of independent truncation error coefficients N_p, are tabulated for the y' formula, for $p = 1, \ldots, 14$, in Table 4.5.

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14
n_p	1	1	2	3	6	10	20	36	72	137	275	541	1098	2208
T_p	1	2	4	7	13	23	43	79	151	288	563	1104	2202	4410
i_p	1	1	1	2	3	6	9	15	27	48	87	158	292	539
N_p	1	2	3	5	8	13	22	37	64	112	199	357	649	1188

Table 4.5: n_p, T_p, i_p and N_p for an RKN$_s^p$, $p = 1, \ldots, 14$.

4.6 Explicit RKN pairs

The theoretical development of RKN pairs was essentially concurrent with that of RK pairs, one technique naturally motivating the other. It can be argued that since the special second-order IVP does not explicitly involve the derivative, it is only the $\boldsymbol{y}$ solution that is of interest, and therefore the main initial consideration was the use of step size control, and whether monitoring the local error on both $\boldsymbol{y}$ and $\boldsymbol{y}'$ approximations was necessary. Such considerations, and a general review of the technique, are detailed next.

4.6.1 Embedded RKN order conditions

The general RKN embedded pair contains four separate sets of external weights, and requires

$$^{\dagger}\tau_j^{(i)} = 0, \; j = 1, \ldots, n_{i-1}, \; i = 2, \ldots, p, \tag{4.19}$$

$$^{\dagger}\tau_j^{\prime(i)} = 0, \; j = 1, \ldots, n_i, \; i = 1, \ldots, p, \tag{4.20}$$

and

$$^{\dagger}\widehat{\tau}_j^{(i)} = 0,\ j = 1, \ldots, n_{i-1},\ i = 2, \ldots, q, \tag{4.21}$$

$$^{\dagger}\widehat{\tau}_j^{\prime(i)} = 0,\ j = 1, \ldots, n_i,\ i = 1, \ldots, q. \tag{4.22}$$

Fehlberg [52] proposed RKN pairs with an embedding on the y formula only (i.e., conditions (4.20) were not solved), as did Dormand and Prince [20], and Filippi and Graf [53], the latter including an $\mathrm{RKN}_{18f}^{11(10)}$. However, if the rates of change of $\boldsymbol{y}$ and $\boldsymbol{y}'$ are greatly different for any one problem, it can be expected that error control on both is an advantage, thus it has since been thought preferable to perform step size control on $\boldsymbol{y}'$ also (see, for example, Horn [16]). From now on, the terms *pair* or *embedding* will refer to such 'complete' RKN embedded pairs, with an available error estimate on both $\boldsymbol{y}$ and $\boldsymbol{y}'$.

4.6.2 Review of embedded RKN formulae

'Complete' Nyström pairs were first derived by Bettis [54], [55] ($\mathrm{RKN}_{6f}^{5(4)}$ and $\mathrm{RKN}_{7f}^{6(5)}$ FSAL pairs) and Horn [16] ($\mathrm{RKN}_{8}^{7(6)}$, $\mathrm{RKN}_{10}^{8(7)}$, and $\mathrm{RKN}_{13f}^{9(8)}$ pairs). Dormand, et al. [19] reconsidered the fifth-order case, and generalising the model of Bettis [55] obtained an $\mathrm{RKN}_{6f}^{6(4)}$ (called 'RKN6(4)6FM'). This pair was found to be particularly efficient compared with other six-stage formulae owing to the extra order of accuracy, and having made a choice of the free parameters to reduce the quantities

$$\begin{gathered} A_N^{(q+1)} = \|\widehat{\boldsymbol{\tau}}^{(q+1)}\|_N,\ B_N^{(p+2)} = \frac{\|\boldsymbol{\tau}^{(p+2)}\|_N}{\|\boldsymbol{\tau}^{(p+1)}\|_N}, \\ C_N^{(p+2)} = \frac{\|\boldsymbol{\tau}^{(p+2)} - \widehat{\boldsymbol{\tau}}^{(p+2)}\|_N}{\|\boldsymbol{\tau}^{(p+1)}\|_N}, \end{gathered} \tag{4.23}$$

for the y formula, and

$$\begin{gathered} A_N^{\prime(q+1)} = \|\widehat{\boldsymbol{\tau}}^{\prime(q+1)}\|_N,\ B_N^{\prime(p+2)} = \frac{\|\boldsymbol{\tau}^{\prime(p+2)}\|_N}{\|\boldsymbol{\tau}^{\prime(p+1)}\|_N}, \\ C_N^{\prime(p+2)} = \frac{\|\boldsymbol{\tau}^{\prime(p+2)} - \widehat{\boldsymbol{\tau}}^{\prime(p+2)}\|_N}{\|\boldsymbol{\tau}^{\prime(p+1)}\|_N}, \end{gathered} \tag{4.24}$$

for y' (Prince [12]), which are natural generalisations of those for the RK case (4.14). The norms (4.23) and (4.24) were, in fact, compromised slightly by stability requirements.

Above sixth-order, derivation of RKN processes involves imposition of 'Q' conditions to reduce the number of independent equations (parallel to the RK case). The highest order 'complete' RKN pair known to date is the RKN12(10)17M (an $\mathrm{RKN}_{17}^{12(10)}$) of Dormand, et al. [56], further details of which may be found in Chapter 8. Such high-order pairs rely heavily on simplifying conditions, including use of 'R' parameters for the y' forms (see Section 3.2.3).

4.7 Explicit RKN triples

Continuous RKN methods have previously been considered by, for example, Horn [16], Fine [57], Dormand and Prince [15], Storer [58], and Tsitouras, et al. [59].

4.7.1 Continuous RKN order conditions

Solution to order p^* requires that

$$^{\dagger}\tau_j^{(i)*} = 0, \; j = 1, \ldots, n_{i-1}, \; i = 2, \ldots, p^*, \tag{4.25}$$

and

$$^{\dagger}\tau_j'^{(i)*} = 0, \; j = 1, \ldots, n_i, \; i = 1, \ldots, p^*. \tag{4.26}$$

Dormand and Prince [15] showed that, as for the RK case, two modes are available for provision of an RKN triple, one corresponding to forming an interpolant over a fixed discrete approximation, termed a 'mode B' approach, or the direct solution of (4.25) and (4.26) — the 'mode A' approach. In the absence of an underlying pair, (4.25) and (4.26) may be solved concurrently with the order conditions for an RKN pair (4.19), (4.20), (4.21) and (4.22) to produce the desired triple.

The term 'order' for an RKN triple may, in fact, not be a sufficient description due to the dense processes (i.e., the $\boldsymbol{y}^*$ and $\boldsymbol{y}'^*$ interpolants) being entirely distinct. Thus, they can have different orders, and providing they share the same function evaluations, they can use a different number of the stages. As such, the notation p^*/s^* will remain where the standard interpretation suffices, but the terms p^{y^*}/s^{y^*} and $p^{y'^*}/s^{y'^*}$ may be used where necessary in an obvious manner. It should be noted that the standard definition does not allow for the lack of either interpolant, and as such the term 'complete' will be extended for a triple to denote the existence of both interpolants.

4.7.2 Review of continuous RKN formulae

Favouring 'mode A', in [15] a $p^* = 3$ ('third-order') interpolant was provided over an existing $\mathrm{RKN}_{4f}^{4(3)}$, and a $p^* = 5$ ('fifth-order') formula derived over the RKN6(4)6FM, both pairs from Dormand, et al. [19].

These $p^* = q-1$ formulae both have $p^{y^*} = q$, and thus allow solution of the y^* terms to zero for an order above those of the y'^* formula, i.e., the methods have

$$^{\dagger}\tau_j^{(q)*} = 0,\ \ j = 1, \ldots, n_{q-1}, \tag{4.27}$$

in addition to (4.25) for $p^* = q-1$. This is, perhaps, not surprising since the number of equations to be solved is the same for the two formulae in this case. In fact (following [15]), a $p^{y'^*} = q-1$ formula may be obtained from a $p^{y^*} = q$ interpolant by determining each $b_i'^*$ parameter from simple differentiation of b_i^*, using

$$\sigma b_i'^* = \frac{d}{d\sigma}(\sigma^2 b_i^*),\ \ i = 1, \ldots, s^*. \tag{4.28}$$

Equivalently, assuming $p^{y'^*} = q-1$, a $p^{y^*} = q$ result is assured if the b_i^*'s are determined from (4.28) by an integration procedure.

It is due to this extra order for the $\boldsymbol{y}^*$ interpolant that it seems necessary to make the distinction from the standard definitions of Chapter 2 and Dormand and Prince [15], where a dense process is simply termed as having order p^*. In the case of, say, the $p^* = 5$ continuous extension to the RKN6(4)6FM ([15]), the $\boldsymbol{y}^*$ interpolant may be used in the absence of any continuous $\boldsymbol{y}'^*$ approximation, and so should itself be considered as being sixth-order. The main purpose of such generality in notation is the treatment of $p^* = p^{y'^*} = p^{y^*} = q$ RKN triples. In this case, the y^* formula may be derived to use fewer of the method stages than for the corresponding y'^* result, allowing for the possibility of saving function evaluations if only the $\boldsymbol{y}^*$ approximation was required, etc. Such considerations are developed in Chapters 6 and 8.

As in the RK case, the optimality of a continuous formula has previously been considered in terms of a local error criterion by employing some measure of the continuous principal local truncation error terms. For example, Dormand and Prince [15] use

$$M_s = \max_{\sigma \in [0,1]} \|\sigma^{p^{y^*}+1} \boldsymbol{\tau}^{(p^{y^*}+1)*}\|_2,$$

and

$$M'_s = \max_{\sigma \in [0,1]} \|\sigma^{p^{y'^*}+1} \boldsymbol{\tau}'^{(p^{y'^*}+1)*}\|_2.$$

Finally, the highest order RKN pairs with dense output are currently the 'mode A' $\mathrm{RKN}^{8(6)}_{10f}\mathrm{D}^{7}_{11}$ of Storer [58], and the $p^* = 7$, twelve-stage continuous extension to the Dormand and Prince RKN8(6)9FM pair (an $\mathrm{RKN}^{8(6)}_{9f}$) [56] due to Tsitouras, et al. [59] (both with $p^{y^*} = 8$). The approach of the latter authors is that corresponding to a 'mode B' approach. In this case, a Hermite form is used to interpolate the $\boldsymbol{y}$, $\boldsymbol{y}'$ and $\boldsymbol{y}''$ values at the mesh points, along with three intermediate values obtained from the unique sixth-order C^2 continuous formulae ($p^{y^*} = 7$) available using the 9 stages of the RKN8(6)9FM. The dense formula was optimised using an (unspecified) local truncation error norm. Taking a different approach, a two-step continuous formula for the RKN8(6)9FM, using no extra function evaluations per step, is shown to be more *efficient* (using a measure of global accuracy per function evaluation) in the case where the step size is increasing.

Chapter 5

RK-ING: Continuous RK(N) development strategies

5.1 Introduction

Low-order continuous approximation using simple polynomial interpolants has been established as convenient and 'cheap' for both explicit RK and RKN methods, a review of which is contained in Chapter 4. The following sections deal with the algebraic systems that must be satisfied to yield dense formulae of a desired order, and the error terms that are of interest for optimisation of the dense output under a range of practical considerations. A suitable subset of possible models is derived, and is such that a common systematic approach, using a restricted set of simplifying conditions, allows a convenient general framework to contain all the cases of interest.

5.2 Continuous order conditions

It will henceforth be assumed that all order conditions are presented in the alternative form implementing 'Q' parameters. Such an equation set may therefore be written in generality as

$$\sum_{i=1}^{s^*} w_i^* c_i^k = {}^k r, \; k = 0, \dots, n_q, \tag{5.1}$$

$$\sum_{i=2}^{s^*} w_i^* {}^k\overline{\Phi}_i^{(q)} = 0, \; k = 1, \dots, n, \tag{5.2}$$

where n_q and n are the number of quadrature and non-quadrature equations, respectively, and the 'w_i^*' represent a set of external polynomial weights, i.e., either $b_i^*(\sigma)$ or $b_i'^*(\sigma),\ i = 1, \ldots, s^*$. The ${}^k r$ form the right-hand side of the quadrature equations, and therefore

$${}^k r = \frac{\sigma^k}{k+1} \tag{5.3}$$

for RK and RKN y' equations, and

$${}^k r = \frac{\sigma^k}{(k+1)(k+2)} \tag{5.4}$$

for the RKN y case. The non-quadrature conditions (5.2) may be further parameterised recursively with

$${}^k\overline{\Phi}_i^{(q)} = c_i^{l_k} \prod_{j=1}^{n_k} \alpha_{i,{}^j m_k},\ k = 1, \ldots, n,\ i = 2, \ldots, s^*, \tag{5.5}$$

where ${}^j m_k \in \{1, \ldots, n_\alpha\},\ l_k \geq 0,\ n_k > 0$, and

$$\alpha_{ik} \in \left\{ Q_{ij_k},\ \sum_{t=2}^{i-1} a_{it} c_t^{v_k} \prod_{u=1}^{p_k} \alpha_{t,{}^u z_k} \right\},\ k = 1, \ldots, n_\alpha,\ i = 2, \ldots, s^*,$$

for $j_k \in \{1, \ldots, n_Q\}$, ${}^u z_k \in \{1, \ldots, k-1\}$, $v_k \geq 0$, and $p_k > 0$. Here, the quantity n_Q represents the number of 'Q' parameters in the model. Importantly, the set $\{\alpha_{i1}, \ldots, \alpha_{i,n_\alpha}\}$ now holds every (recursive) subsection of every equation from (5.2), and as such is convenient for an automatic parameterisation of the model. Usually, however, 'reduced' systems of equations will be considered that simplify the above characterisation, if not immediately in a clearly solvable form, and permit use of linear algebra techniques to generalise most of the work — such schemes will be detailed next.

5.3 Parameterised 'linear' systems

Simplifying conditions of the following types will be utilised:

$$w_i^* = 0,\ i = 2, \ldots, s^b - 1, \tag{5.6}$$

$$a_{ij} = 0,\ j = 2, \ldots, s_i^a - 1,\ i = 3, \ldots, s^*, \tag{5.7}$$

and

$$Q_{ik} = 0,\ k = 1, \ldots, q_i - 1,\ i = 3, \ldots, s^*. \tag{5.8}$$

Judicious use of (5.6), (5.7) and (5.8) will allow independent equation sets to be presented for solution that are linear in the 'α' quantities. In the characterisation of (5.5), this implies

$$n_i = p_j = 1,\ i = 1, \ldots, n,\ j = 1, \ldots, n_\alpha,$$

and the following representation will be assumed:

$${}^k\overline{\Phi}_i^{(q)} \equiv {}^k\nu_i^b = c_i^{l_k} \alpha_{i,m_k},\ k = 1, \ldots, n,\ i = s^b, \ldots, s^*, \tag{5.9}$$

$$\alpha_{ik} \in \left\{ Q_{ij_k},\ a_{if_k},\ \sum_{t=s_i^a}^{i-1} a_{it} c_t^{v_k} \alpha_{t,z_k} \right\},\ k = 1, \ldots, n_\alpha,\ i = 2, \ldots, s^*, \tag{5.10}$$

where $j_k \in \{q_i, \ldots, n_Q\}$, $f_k \in \{s_i^a, \ldots, s^a - 1\}$, $l_k \geq 0$, $v_k \geq 0$, $m_k \in \{1, \ldots, n_\alpha\}$, and $z_k \in \{1, \ldots, k-1\}$. Quantity s^a is thus defined as the smallest value such that, $\forall i$, $s^a \geq s_i^a$ and $\alpha_{ik} \not\equiv a_{i,s^a}$ for *any* $k \in \{1, \ldots, n_\alpha\}$. One final notational tool is the definition of an n_a-vector of integer values, $\boldsymbol{\eta}$, to contain the set of α_{ik} quantities not equal to 'Q' or 'a' parameters, i.e., such that

$$\alpha_{i,{}^m\eta} \notin \left\{ a_{i,s_i^a}, \ldots, a_{i,s^a-1},\ \ Q_{i,q_i}, \ldots, Q_{i,n_Q} \right\},\ m = 1, \ldots, n_a.$$

The convenience of this linearised form can now be seen by posing an equivalent set of order conditions, in addition to (5.6) and (5.7), as follows:

$$\sum_{i=1}^{s^*} w_i^* c_i^k = {}^k r, \; k = 0, \ldots, n_q, \tag{5.11}$$

$$\sum_{i=s^b}^{s^*} w_i^* {}^k\nu_i^b = \sum_{i=s^b}^{s^*} w_i^* c_i^{l_k} \alpha_{i,m_k} = 0, \; l_k \geq 0, \; m_k \in \{1, \ldots, n_\alpha\}, \; k = 1, \ldots, n, \tag{5.12}$$

$$\sum_{j=s^a}^{i-1} a_{ij} c_j^k = {}^k r_i, \; k = 0, \ldots, n_Q, \; i = 3, \ldots, s^*, \tag{5.13}$$

and

$$\sum_{j=s^a}^{i-1} a_{ij} {}^k\nu_j^d = {}^k\nu_i^r, \; k = 1, \ldots, n_a, \; i = 3, \ldots, s^*, \tag{5.14}$$

where ${}^k\nu_i^r = \alpha_{i,{}^k\eta}$, and the ${}^k\nu_i^d$ are defined from (5.10), for $k = 1, \ldots, n_a$. In this formulation, for a given $i \in \{3, \ldots, s^*\}$, (5.13) with $k = 0$ determines a_{i1}, i.e.,

$$a_{i1} = {}^0 r_i - \sum_{j=s_i^a}^{i-1} a_{ij}, \tag{5.15}$$

and (5.13) for $k = 1, \ldots, n_Q$, with (5.14) represents a linear system for a_{ij}, $j = s^a, \ldots, i-1$:

$$A_i \boldsymbol{a}_i = \boldsymbol{\alpha}_i, \tag{5.16}$$

where

$$A_i = \left(\begin{pmatrix} \boldsymbol{c}_{s^a}^{(Q)} \\ \boldsymbol{\nu}_{s^a}^d \end{pmatrix} \cdots \begin{pmatrix} \boldsymbol{c}_{i-1}^{(Q)} \\ \boldsymbol{\nu}_{i-1}^d \end{pmatrix} \right), \; \boldsymbol{a}_i = \begin{pmatrix} a_{i,s^a} \\ \vdots \\ a_{i,i-1} \end{pmatrix}, \; \boldsymbol{\alpha}_i = \begin{pmatrix} \boldsymbol{r}_i \\ \boldsymbol{\nu}_i^r \end{pmatrix},$$

$$\boldsymbol{c}_i^{(Q)} = \left(c_i, \; c_i^2, \ldots, c_i^{n_Q} \right)^T,$$

and all other vector components are given by the related symbol with a preceding superscript, as usual. System (5.16) will often be written in its augmented form $(\overline{A}_i)$, with the right-hand-side vector appended onto the coefficient matrix, i.e.,

$$\overline{A}_i = (A_i \,|\, \boldsymbol{\alpha}_i)\,. \tag{5.17}$$

Equivalently, (5.11) and (5.12) can be written:

$$w_1^* = {}^0r - \sum_{i=s^b}^{s^*} w_i^*, \tag{5.18}$$

and

$$W\boldsymbol{w}^* = \boldsymbol{\omega}, \tag{5.19}$$

where

$$W = \left(\begin{pmatrix} \boldsymbol{c}_{s^b}^{(q)} \\ \boldsymbol{\nu}_{s^b}^{b} \end{pmatrix} \cdots \begin{pmatrix} \boldsymbol{c}_{s^*}^{(q)} \\ \boldsymbol{\nu}_{s^*}^{b} \end{pmatrix} \right), \ \boldsymbol{w}^* = \begin{pmatrix} w_{s^b}^* \\ \vdots \\ w_{s^*}^* \end{pmatrix}, \ \boldsymbol{\omega} = \begin{pmatrix} \boldsymbol{r} \\ 0 \end{pmatrix}, \ \boldsymbol{c}_i^{(q)} = \left(c_i,\ c_i^2, \ldots, c_i^{n_q} \right)^T.$$

While such a representation is sufficient for all the examples of interest in Chapters 6–8, it should be reiterated that this framework does not contain *every* possible continuous RK(N) equation set, but that its linearity property is extremely desirable, and in fact *most* models will be as such. Solution of systems (5.16) and (5.19) may be trivial in certain circumstances, namely when none are *over-determined*, i.e., when $n_Q + n_a \le i - s^a$, $\forall i$, and $n_q + n < s^* - s^b$, taking into account that some of α_{ik}, $k = 1, \ldots, n_\alpha$, will be zero for low i by the explicit nature, and when no inconsistent linear dependencies exist between the rows of A_i or W. A very simple case where this is not true is considered by the following example.

Example 5.3.1 $(Q_{31} = Q_{32} = 0)$ Assuming $q_3 = n_Q = s_3^a = s^a = 2$, then

$$A_3 = \begin{pmatrix} c_2 \\ c_2^2 \end{pmatrix}, \text{ and } \boldsymbol{r}_3 = \begin{pmatrix} {}^1r_3 \\ {}^2r_3 \end{pmatrix}.$$

The augmented linear system (5.17) for a_{32} is thus:

$$\overline{A}_3 = \left(\begin{array}{c|c} c_2 & {}^{1}r_3 \\ c_2^2 & {}^{2}r_3 \end{array} \right).$$

Assuming $c_2 \neq 0$, applying one step of Gaussian-elimination using element $(1,1)$ as a pivot yields:

$$\overline{A}_3 = \left(\begin{array}{c|c} c_2 & {}^{1}r_3 \\ 0 & \rho_{31} \end{array} \right),$$

where $\rho_{31} = {}^{2}r_3 - c_2{}^{1}r_3$. The augmented system is therefore singular, but can be made consistent by constraining c_2 to make $\rho_{31} = 0$, i.e., by taking

$$c_2 = \frac{{}^{2}r_3}{{}^{1}r_3}.$$

For the RK case[1],

$${}^{k}r_3 = \frac{c_3^{k+1}}{k+1},$$

and $c_2 = 2c_3/3$. For RKN,

$${}^{k}r_3 = \frac{c_3^{k+2}}{(k+2)(k+1)},$$

and $c_2 = c_3/2$. The system $\overline{A}_3$ is now solvable for a_{32} in both cases. Note that in the case where $c_2 = 0$, consistency would require $c_3 = 0$ leaving a_{32} free.

$\square$

Solution of continuous order conditions in such over-determined cases as Example 5.3.1 is equivalent to forcing linear consistency on the systems (5.16) and (5.19). This may be approached in different ways depending on the nature and complexity of the matrix elements. While such consistency conditions can be presented formally as *determinants*, the more computational treatment utilising Gaussian-elimination will be used throughout. This is viable due to possible pivots in most cases being

[1]This standard result was presented earlier in Section 4.3.1.

functions of the 'c_i', only, as these will invariably be free parameters and polynomials involving them can be assumed to be non-zero in general.

While the order conditions presented in this section are of general interest, the case of most concern is that where a discrete method exists and a continuous extension is to be derived, commonly adding extra stages. Assuming the dense formula is not sought for an order higher than the discrete, the conditions (5.16) and (5.15) will already hold for $i = 3, \ldots, s$, and must be solved for $i = s+1, \ldots, s^*$, along with (5.18) and (5.19) to derive the continuous extension, and a systematic approach to solving consistency conditions may be considered. This will be discussed next, and forms the strategy on which the remaining work is based.

5.4 Continuous extensions

While the procedure for solving systems (5.16) and (5.19) must be the same independent of whether a fixed discrete formula is involved, certain features of this case can be exploited.

Definition 5.4.1 *An order p^*, s^*-stage continuous extension will be denoted by $D_{s^*}^{p^*}$, and this notation may conveniently be appended onto a discrete method name to define the relevant triple. Where lower-order ($\overline{p}^* < p^*$) dense output is available with $\overline{s}^*$-stages in common, this embedding may be written as $D_{s^*(\overline{s}^*)}^{p^*(\overline{p}^*)}$.*

A simple example may motivate essential features of the general case, and as such a family of fourth-order continuous extensions will now be derived for the 'classic' RK_4^4.

Example 5.4.1 (A D_6^4 extension for the Kutta RK_4^4 **[4])** Firstly, a C^1 continuous extension will require a reusable fifth stage — implying, in addition to the tableau (4.6):

$$a_{51} = b_1 = \frac{1}{6},\ a_{52} = b_2 = \frac{1}{3},\ a_{53} = b_3 = \frac{1}{3},\ a_{54} = b_4 = \frac{1}{6},\ b_5 = 0 \text{ and } c_5 = 1.$$

The following set of expressions are required to be solved to zero for fourth-order dense, keeping s^* free:

```
[s] order :4, parameterise
```

1. $\displaystyle\sum_{i=1}^{s^*} b_i^* - 1$
2. $\displaystyle\sum_{i=2}^{s^*} b_i^* c_i^k - \frac{\sigma^k}{k+1}, \quad k = 1:3$
3. $\displaystyle\sum_{i=2}^{s^*} b_i^* c_i^k Q_{i1}, \quad k = 0:1$
4. $\displaystyle\sum_{i=2}^{s^*} b_i^* Q_{i2}$
5. $\displaystyle\sum_{i=3}^{s^*} b_i^* \sum_{j=2}^{i-1} a_{ij} Q_{j1}$

Three b^*'s (of third-degree in σ) will be required to zero the quadrature terms (2) above, and (1) defines b_1^*. An equivalent 'α-parameterisation' for solution of the non-quadrature terms (3)–(5) to zero may be written:

$$\sum_{i=2}^{s^*} b_i^* \alpha_{i1} = 0, \ \sum_{i=2}^{s^*} b_i^* c_i \alpha_{i1} = 0, \ \sum_{i=2}^{s^*} b_i^* \alpha_{i2} = 0, \text{ and } \sum_{i=3}^{s^*} b_i^* \alpha_{i3} = 0,$$

where

$$\alpha_{i1} = Q_{i1}, \ \alpha_{i2} = Q_{i2}, \ \alpha_{i3} = \sum_{j=2}^{i-1} a_{ij} \alpha_{j1}. \tag{5.20}$$

In the framework of the previous section, this problem has

$$n = 4, \ n_q = n_\alpha = 3, \ n_Q = s_i^a = s^a = s^b = 2, \ n_a = 1, \ \boldsymbol{r}_i = \left(\frac{c_i^2}{2} + \alpha_{i1}, \ \frac{c_i^3}{3} + \alpha_{i2} \right)^T,$$

$$q_i = 1, \forall i, \ \boldsymbol{\nu}_i^b = (\alpha_{i1}, \ c_i \alpha_{i1}, \ \alpha_{i2}, \ \alpha_{i3})^T, \ \boldsymbol{\nu}_i^d = (\alpha_{i1}), \ \boldsymbol{\nu}_i^r = (\alpha_{i3}),$$

and ${}^{k}r$ as given in (5.3). The general $\boldsymbol{a}_i$ system may conveniently be abbreviated in its augmented form as

$$\overline{A}_i = \left(\begin{array}{ccccc|c} \frac{1}{2} & \frac{1}{2} & 1 & 1 & c_k & \frac{c_i^2}{2} + \alpha_{i1} \\ \frac{1}{4} & \frac{1}{4} & 1 & 1 & c_k^2 & \frac{c_i^3}{3} + \alpha_{i2} \\ -\frac{1}{8} & \frac{1}{8} & 0 & 0 & \alpha_{k1} & \alpha_{i3} \end{array}\right),$$

where the penultimate column should be expanded over the range $k = 6, \ldots, i-1$ (understanding that when $i = 6$ it collapses to the null case), and $\alpha_{k1},\ k = 2, \ldots, 5$, have been evaluated from the definition (5.20). The augmented $\boldsymbol{b}^*$ system may equivalently be written as

$$\overline{W} = \left(\begin{array}{ccccc|c} \frac{1}{2} & \frac{1}{2} & 1 & 1 & c_k & \frac{\sigma}{2} \\ \frac{1}{4} & \frac{1}{4} & 1 & 1 & c_k^2 & \frac{\sigma^2}{3} \\ \frac{1}{8} & \frac{1}{8} & 1 & 1 & c_k^3 & \frac{\sigma^3}{4} \\ -\frac{1}{8} & \frac{1}{8} & 0 & 0 & \alpha_{k1} & 0 \\ -\frac{1}{16} & \frac{1}{16} & 0 & 0 & c_k\alpha_{k1} & 0 \\ -\frac{1}{24} & \frac{1}{12} & -\frac{1}{12} & 0 & \alpha_{k2} & 0 \\ 0 & -\frac{1}{16} & \frac{1}{8} & 0 & \alpha_{k3} & 0 \end{array}\right),$$

where, again, the penultimate column is expanded over $k = 6, \ldots, s^*$. The $\overline{W}$ system may now be split into two by separating the quadrature and non-quadrature parts for convenience, i.e.,

$$\overline{X} = \left(\begin{array}{ccccc|c} \frac{1}{2} & \frac{1}{2} & 1 & 1 & c_k & \frac{\sigma}{2} \\ \frac{1}{4} & \frac{1}{4} & 1 & 1 & c_k^2 & \frac{\sigma^2}{3} \\ \frac{1}{8} & \frac{1}{8} & 1 & 1 & c_k^3 & \frac{\sigma^3}{4} \end{array}\right), \tag{5.21}$$

and

$$\overline{M} = \left(\begin{array}{ccccc|c} -\frac{1}{8} & \frac{1}{8} & 0 & 0 & \alpha_{k1} & 0 \\ -\frac{1}{16} & \frac{1}{16} & 0 & 0 & c_k\alpha_{k1} & 0 \\ -\frac{1}{24} & \frac{1}{12} & -\frac{1}{12} & 0 & \alpha_{k2} & 0 \\ 0 & -\frac{1}{16} & \frac{1}{8} & 0 & \alpha_{k3} & 0 \end{array}\right). \tag{5.22}$$

As mentioned previously, system (5.21) will determine three of b_i^*, $i = 2, \ldots, s^*$, in terms of the others. It is now of importance to solve (5.22) in as few stages as possible. The homogeneous form of $\overline{M}$ makes it convenient to consider an equivalent form after 'triangularising' the numerical part by Gaussian-elimination, i.e.,

$$\overline{M} = \left(\begin{array}{ccccc|c} -\frac{1}{8} & \frac{1}{8} & 0 & 0 & \alpha_{k1} & 0 \\ 0 & \frac{1}{24} & -\frac{1}{12} & 0 & \alpha_{k2} - \alpha_{k1}/3 & 0 \\ 0 & 0 & 0 & 0 & \alpha_{k1}\left(c_k - \frac{1}{2}\right) & 0 \\ 0 & 0 & 0 & 0 & \alpha_{k3} + 3\alpha_{k2}/2 - \alpha_{k1}/2 & 0 \end{array} \right),$$

and, introducing further notation,

$$M = \left(\begin{array}{c|c} T_b & \boldsymbol{\zeta}_k^b \\ \hline 0 & \boldsymbol{\zeta}_k^\beta \end{array} \right), \tag{5.23}$$

where

$$T_b = \begin{pmatrix} -\frac{1}{8} & \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{24} & -\frac{1}{12} & 0 \end{pmatrix}, \quad \boldsymbol{\zeta}_i^b = \left(\alpha_{i1},\ \alpha_{i2} - \frac{\alpha_{i1}}{3}\right)^T,$$

and

$$\boldsymbol{\zeta}_i^\beta = \left(\alpha_{i1}\left(c_i - \frac{1}{2}\right),\ \alpha_{i3} + \frac{3\alpha_{i2}}{2} - \frac{\alpha_{i1}}{2}\right)^T.$$

The top section of the coefficient matrix (5.23), i.e,

$$\left(\begin{array}{c|c} T_b & \boldsymbol{\zeta}_k^b \end{array} \right),$$

will determine r_b of the b^*'s, where $r_b = \mathrm{RANK}(T_b) = 2$, providing no linear dependencies exist between its rows and those of X. In this case, solution would only be possible for specific σ — for arbitrary α_{i1}, α_{i2} and c_i, however, this form of singularity will not appear. At present, five b^*'s have been accounted for in addition to b_1^*. All attention must now be given to the $\boldsymbol{\zeta}_i^\beta$ vector, which in this case forms the system

$$\beta = \begin{pmatrix} \beta_{6,1} & \cdots & \beta_{s^*,1} \\ \vdots & \ddots & \vdots \\ \beta_{6,n_\beta} & \cdots & \beta_{s^*,n_\beta} \end{pmatrix},$$

where

$$n_\beta = 2,\ \beta_{i1} \equiv {}^1\zeta_i^\beta = \alpha_{i1}\left(c_i - \frac{1}{2}\right),\ \text{and}\ \beta_{i2} \equiv {}^2\zeta_i^\beta = \alpha_{i3} + \frac{3\alpha_{i2}}{2} - \frac{\alpha_{i1}}{2}.$$

Ideally, $r_\beta = \text{RANK}(\beta) = 0$, requiring that $\beta_{i1} = \beta_{i2} = 0$, $\forall i$, and giving the desired minimal-stage solution with $s^* = 6$. This imposes either $c_6 = 1/2$ or $\alpha_{61} = 0$, and constrains one of α_{62} or α_{63} to the other, $\alpha_{62} = (\alpha_{61} - 2\alpha_{63})/3$, say. However, setting $c_6 = 1/2$ would make $\overline{X}$ inconsistent, and so $\alpha_{61} = 0$, $\alpha_{62} = -2\alpha_{63}/3$, for c_6 and α_{63} free, is the only available solution. The $\boldsymbol{a}_i$ system may also be written in an equivalent (partially eliminated) form as:

$$\left(\begin{array}{c|c} T_a & \boldsymbol{\zeta}_k^d \\ \hline 0 & \boldsymbol{\zeta}_k^\delta \end{array}\right) \boldsymbol{a}_i = \left(\begin{array}{c} \boldsymbol{\zeta}_i^r \\ \hline \boldsymbol{\zeta}_i^\rho \end{array}\right), \tag{5.24}$$

where

$$T_a = \begin{pmatrix} -\frac{1}{8} & \frac{1}{8} & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix},\ \boldsymbol{\zeta}_i^d = \left(\alpha_{i1},\ c_i + 4\alpha_{i1},\ c_i\left(c_i - \frac{1}{2}\right)\right)^T,$$

$$\boldsymbol{\zeta}_i^r = \left(\alpha_{i3},\ \frac{c_i^2}{2} + 4\alpha_{i3} - \alpha_{i1},\ \frac{c_i^3}{3} - \frac{c_i^2}{4} + \alpha_{i2} - \frac{\alpha_{i1}}{2}\right)^T,$$

and (5.24) is therefore under-determined — in this case $\boldsymbol{\zeta}_i^\delta$ and $\boldsymbol{\zeta}_i^\rho$ are undefined. The system has $r_a = \text{RANK}(T_a) = 3$, and $\overline{A}_6$ will determine a_{6i}, $i = 2, \ldots, 4$, in terms of a_{65} as another free parameter, and a row-sum condition forces

$$a_{61} = c_6 - \sum_{i=2}^{5} a_{6i}$$

to complete the three-parameter model. The final form of the resulting family of C^1 continuous, fourth-order, six-stage processes could now be presented, or the dense output optimised in some way to choose one member of the family and shown as a tableau. This was not felt to be necessary due to the simplicity of the case, and has thus served mainly to introduce some techniques for later development.

□

Example 5.4.1 was unnecessarily complex due to the introduction of new notation intended for models of higher order. Importantly, the general $\boldsymbol{a}_i$ system being underdetermined did not allow for a demonstration of forcing consistency on the $\overline{A}_i$ matrix. The general form (5.24) is more common, and requires consideration of the n_δ-vectors $\boldsymbol{\zeta}_i^\delta$ and $\boldsymbol{\zeta}_i^\rho$, where $n_\delta = n_Q + n_a - r_a$. Each $\overline{A}_i$ system will determine r_a of $a_{ij},\ j = s^a, \ldots, i-1$, and there will be $i + n_\delta - s^r - 1$ conditions to be solved resulting from required consistency of the augmented systems

$$\overline{D}_i = \left(\begin{array}{ccc|c} \delta_{s^r+1,1} & \cdots & \delta_{i-1,1} & \rho_{i,1} \\ \vdots & \ddots & \vdots & \vdots \\ \delta_{s^r+1,n_\delta} & \cdots & \delta_{i-1,n_\delta} & \rho_{i,n_\delta} \end{array} \right), \ i = s^r+1, \ldots, s^*, \tag{5.25}$$

for determination of $a_{ij},\ j = s^r+1, \ldots, i-1$, where

$$\delta_{ik} \equiv {}^k\zeta_i^\delta,\ i = s^r+1, \ldots, s^*-1, \text{ and } \rho_{ik} \equiv {}^k\zeta_i^\rho,\ i = s^r+1, \ldots, s^*,\ k = 1, \ldots, n_\delta.$$

Stage s^r in Example 5.4.1 is the fifth, and in general is defined as the (necessary) reusable stage. If the underlying discrete pair is of FSAL design, then $s^r = s$, and is assumed to be $s+1$ otherwise. For example, in the case $i = s^r+1$, all possible $a_{s^r+1,j},\ j = s^a, \ldots, s^r$, are determined from

$$T_a \boldsymbol{a}_{s^r+1} = \boldsymbol{\zeta}^r_{s^r+1},$$

and $\rho_{s^r+1,k} = 0,\ k = 1, \dots n_\delta$, must be solved for consistency. In cases where the T_a system is non-square, $i - r_a - s^a$ of the $a_{s^r,j}$ are forced to remain as free parameters, which ones depending on the position of rank-deficient columns in T_a.

5.4.1 Change of variables

The 'β_i', 'δ_i' and 'ρ_i' quantities just introduced are of fundamental importance to the solution of continuous extension models. All of the important properties of a given system may now be written in terms of these parameters, and in more complex cases it is convenient to make a change of variables in terms of them. This will therefore be done as standard. The 'α_i' quantities of the original system are now of only secondary importance, and may be removed from the system analogous to the way the a's and b^*/b'^*'s of the underlying process are not featured in the consistency conditions of the new representation. Consider the following conditions:

$$\beta_{ik} = {}^k\zeta_i^\beta,\ k = 1, \dots, n_\beta,\ i = s^r + 1, \dots, s^*, \tag{5.26}$$

$$\delta_{ik} = {}^k\zeta_i^\delta,\ k = 1, \dots, n_\delta,\ \ i = s^r + 1, \dots, s^* - 1, \tag{5.27}$$

and

$$\rho_{ik} = {}^k\zeta_i^\rho,\ k = 1, \dots, n_\delta,\ i = s^r + 1, \dots, s^*. \tag{5.28}$$

The ${}^k\zeta_i$'s form linear combinations of $\alpha_{ik},\ k = 1, \dots, n_\alpha$, the coefficients of which are simple polynomials in c_i. As such, (5.26)–(5.28) form the respective linear systems:

$$\Upsilon_i^\beta \boldsymbol{\alpha}_i = \boldsymbol{\beta}_i, \tag{5.29}$$

$$\widetilde{\Upsilon}_i^\delta \boldsymbol{\alpha}_i = \widetilde{\boldsymbol{\delta}}_i, \tag{5.30}$$

and

$$\widetilde{\Upsilon}_i^\rho \boldsymbol{\alpha}_i = \widetilde{\boldsymbol{\rho}}_i, \tag{5.31}$$

to make the required change of variables, where

$$\boldsymbol{\alpha}_i = (\alpha_{i1}, \ldots, \alpha_{i,n_\alpha})^T,\ \boldsymbol{\beta}_i = \left(\beta_{i1}, \ldots, \beta_{i,n_\beta}\right)^T,\ \widetilde{\boldsymbol{\delta}}_i = \boldsymbol{\delta}_i + \boldsymbol{p}_i^\delta,\ \widetilde{\boldsymbol{\rho}}_i = \boldsymbol{\rho}_i + \boldsymbol{p}_i^\rho,$$

$$\boldsymbol{\delta}_i = (\delta_{i1}, \ldots, \delta_{i,n_\delta})^T,\ \boldsymbol{\rho}_i = (\rho_{i1}, \ldots, \rho_{i,n_\delta})^T,$$

$$\boldsymbol{p}_i^\delta = \left(p_{i1}^\delta, \ldots, p_{i,n_\delta}^\delta\right)^T,\ \boldsymbol{p}_i^\rho = \left(p_{i1}^\rho, \ldots, p_{i,n_\delta}^\rho\right)^T,$$

and the p_{ik} quantities are simple polynomials in c_i of varying degree (or zero in special cases). The systems (5.29)–(5.31) may be combined to form a single problem

$$\Upsilon_i \boldsymbol{\alpha}_i = \boldsymbol{v}_i, \tag{5.32}$$

where

$$\Upsilon_i = \begin{pmatrix} \Upsilon_i^\beta \\ \widetilde{\Upsilon}_i^\delta \\ \widetilde{\Upsilon}_i^\rho \end{pmatrix},\ \boldsymbol{v}_i = \begin{pmatrix} \boldsymbol{\beta}_i \\ \widetilde{\boldsymbol{\delta}}_i \\ \widetilde{\boldsymbol{\rho}}_i \end{pmatrix},$$

determining the α_i's in terms of the β_i, δ_i and ρ_i's, removing them from the equation set. When $r_\Upsilon = \text{RANK}(\Upsilon_i) < n_\alpha$, some of $\alpha_{ik},\ k = 1, \ldots, n_\alpha$, will remain free, and when $r_\Upsilon < n_\beta + 2n_\delta$, consistency conditions will be imposed on some of $\beta_{ik},\ k = 1, \ldots, n_\beta$, and $\delta_{ik},\ \rho_{ik},\ k = 1, \ldots, n_\delta$. The general β matrix

$$\beta = \begin{pmatrix} \beta_{s^r+1,1} & \cdots & \beta_{s^*,1} \\ \vdots & \ddots & \vdots \\ \beta_{s^r+1,n_\beta} & \cdots & \beta_{s^*,n_\beta} \end{pmatrix}, \tag{5.33}$$

and $\overline{D}_i$ systems (5.25) are now more concisely represented in terms of the remaining β_i, δ_i and ρ_i's.

5.4.2 Summary of continuous extension strategy

The initial procedures involved in formation of a continuous extension are now as follows:

1. Form an independent set of continuous order conditions in terms of 'Q' parameters.

2. Parameterise in terms of 'α' quantities, forming systems for the $b_i^*/b_i'^*$ and a_{ij} quantities, i.e., (5.16) and (5.18), in terms of '$\boldsymbol{\nu}_i$', '$\boldsymbol{c}_i$' and '$\boldsymbol{r}_i$' vectors.

3. Partially eliminate all systems using the numerical columns made up of the fixed stages of the discrete method - forming '$\boldsymbol{\zeta}_i$' vectors.

4. Change variables from 'α_i' to 'β_i', 'δ_i' and 'ρ_i' quantities for the general i^{th} stage — removing the $\alpha_{ik},\ k = 1, \ldots, n_\alpha$, from the system.

5. Form β-matrix and $\overline{D}_i,\ i = s^r + 1, \ldots, s^*$, systems.

The end result of these steps is the simple characterisation of matrices holding all over-determined parts of the $\boldsymbol{b}^*/\boldsymbol{b}'^*$ and $\boldsymbol{a}_i$ systems. At this point, the number of extra stages required is not yet fixed, and will depend primarily on forcing rank-degeneracy of the β matrix. Strategies for achieving minimal stage solutions are mostly specific to the individual cases, and will not be discussed here any further. The $\overline{D}_i$ systems must also be made consistent to give a solution for *any* s^*, and in some cases this may require extra stages than would be necessary from consideration of β alone.

5.4.3 Practical considerations

Once all over-determined linear systems have been made consistent, the $b_i^*/b_i'^*$'s and a_{ij}'s may be solved for in terms of any unknowns and the 'α_i' quantities, which are also available as the solution of consistent systems. Importantly, this process may therefore be done numerically in a floating-point arithmetic environment for speed when consideration is given to an optimisation procedure. Free parameters may be chosen in this way before passing back into the algebraic model for completion.

All algebraic work will initially be done with RK-AID, and the resulting equation set ported into the *true* symbolic environment MACSYMA. Steps (1)–(5) in Section 5.4.2 may then be considered as essentially automatic assuming sufficiently general linear algebra facilities. The required systems will be made consistent in the minimum number of stages, and then passed into Fortran 90 for numerical, approximate, determination of the method tableau calculated in terms of free parameters. This enables some function of the error terms to be coded as a function of the free parameters for optimisation. A particular form of the error terms has yet to be established, but will be based on local or global error considerations — investigated fully in Section 5.6. Once free parameters have been determined, they may be fed back into the MACSYMA model for finalisation of the resulting tableau of extra parameters for stages $s^r + 1, \ldots, s^*$, and the external polynomial weights for all stages.

5.5 Specifics of the Nyström case

The procedure of Section 5.4 is applicable to the solution of any single set of order conditions, for both RK and RKN. The Nyström case is complicated by there being a set of external polynomial weights for each of the $\boldsymbol{y}$ and $\boldsymbol{y}'$ interpolants, though sharing the same a_{ij} and c_i parameters for efficiency (i.e., this is not *necessary*), non-quadrature equations involving 'Q' parameters are one order apart but identical in the two cases, just interchanging b_i^* and $b_i'^*$. The two fundamental cases of interest are provision of either $p^* = q - 1$ or $p^* = q$ solutions as usual, though the Nyström case gives more freedom due to its two distinct formulae. For $p^* = q - 1$ interpolants, it is possible to automatically generate the polynomial weights for either process having derived those for the other using formula (4.28) — see Section 4.7.2. In this case, $p^{y^*} = q$, and the continuous formula for y is one order higher than for y'. It seems irrelevant which set of order conditions are solved by the continuous extension scheme, provided both are optimised concurrently. Arbitrarily, the y' terms will be solved in every case, and the b^*'s calculated from (4.28) by integration.

For $p^* = p^{y^*} = p^{y'^*} = q$ processes, the two sets of weights may be derived to satisfy a continuous form of expression (4.18), namely:

$$b_i^* = \left(1 - \frac{c_i}{\sigma}\right) b_i'^*, \ i = 1, \ldots, s^*. \tag{5.34}$$

This gives only a C^1 formula, and as such it will not be considered any further. Now, as the rows of M for the b_i^*'s are a subset of those for the $b_i'^*$'s, it is convenient to solve for the y' case first — consistency of the $\boldsymbol{b}'^*$ system in the minimum number of stages will also give consistency of the $\boldsymbol{b}^*$ system in the same number of stages, resulting in free b_i^*'s. For this reason, in Chapters 6 and 8 which consider new Nyström interpolants, the y' continuous extension will be derived and optimised first, determining all free parameters that are common to both formulae, and the free b_i^*'s then used to optimise the y terms. An interesting feature may be considered in this $p^* = q$ case, namely that (4.28) may still be applied having derived the $b_i'^*$'s for $p^{y'^*} = q$, resulting in a $p^{y^*} = q + 1$ formula. It should be noted that problems occur in the notation here as $p^{y^*} = q + 1$, meaning that all truncation error term vectors $\boldsymbol{\tau}^{(i)*}$, $i = 2, \ldots, q + 1$, are zero, cannot produce local errors of order greater than q due to the y' formula being only of local order q. Such a formula will, if no confusion is possible, continue to be termed an 'order $q + 1$' process. This scheme will produce dense output lacking continuity with the standard discrete formula, which may be satisfactory in some circumstances, but is assumed to be a poor result in general. A procedure now termed *continuous extrapolation* solves this problem by propagating the $\sigma = 1$ result of the $\boldsymbol{y}^*$ approximation as the discrete $\widehat{\boldsymbol{y}}$ solution. In this case, $\widehat{b}_i$, $i = 1, \ldots, s^*$, are effectively replaced by

$$\overline{b}_i = b_i^*|_{\sigma=1}, \ i = 1, \ldots, s^*,$$

where

$$b_i^* = \left(\int_0^1 \sigma b'^* \, d\sigma\right)/\sigma^2, \ i = 1, \ldots, s^*, \tag{5.35}$$

and the continuous process is C^0 (with no FSAL). Step size control may now be done in several ways, the simplest is to use the standard embedded pair until a step is accepted, and then to propagate the $\overline{b}_i/\widehat{b}_i'$ linear combinations as the $\boldsymbol{y}$ and $\boldsymbol{y}'$

solutions, respectively. This should be satisfactory in that an error estimate is being used for an order $p < q$ result to choose the steps as usual, and that the discrete solution is still globally order $q-1$. An alternative method is to use the standard error estimate on y', but to use the extrapolated local order $q+1$ result to estimate the error in the order p solution for y. This would, however, require the full s^*-stage formulae at each (accepted or rejected) step. Having lost the reusable stage, it is now possible to re-introduce the FSAL property by addition of an extra function evaluation as stage s^*+1. This extra evaluation has

$$c_{s^*+1} = 1,\ a_{s^*+1,i} = \overline{b}_i,\ i = 1, \ldots, s^*,\ b_{s^*+1} = \widehat{b}_{s^*+1} = b'_{s^*+1} = \widehat{b}'_{s^*+1} = 0,$$

and $b'^*_{s^*+1}$ and $b^*_{s^*+1}$ constrained by continuity requirements. It is simple to show that for *any* $p^* = q$ Nyström method of order $q \geq 4$,

$$b^*_{s^*+1} = \frac{\sigma(\sigma-1)^2}{2},\ b'^*_{s^*+1} = \frac{\sigma(5\sigma-3)(\sigma-1)}{2}, \tag{5.36}$$

will give C^2 continuity if the previous reusable stage is modified by

$$b^*_{s^r} \longrightarrow b^*_{s^r} - b^*_{s^*+1},\ b'^*_{s^r} \longrightarrow b'^*_{s^r} - b'^*_{s^*+1},$$

stages s^r and s^*+1 will then satisfy (5.35) as required. For higher orders, expressions (5.36) *can* be polynomials of arbitrarily high degree, with coefficients of $\sigma^k,\ k > 3$, being free parameters, but for simplicity will be left as third-degree for the purposes of experimentation. Interestingly, this does not affect the *principal* continuous truncation error terms of the y' formula as stages s^r and s^*+1 are *linked* by having $c_{s^r} = c_{s^*+1} = 1$ and $\overline{\Phi}^{(q)}_{s^r} = \overline{\Phi}^{(q)}_{s^*+1} = 0$ for all trees relating to principal truncation error terms. This improves practicability of the technique, as methods may be optimised on the principal truncation error terms as usual, and the extrapolation done as an afterthought. In this way, continuous extrapolation formulae may be derived entirely automatically, and for experimentation purposes such a scheme is coded into

RK-AID as the `EXTRAPolate` command — see Section A.6.32. Example use of this function can be found for the sixth-order case in Chapter 6.

5.6 Choice of free parameters

As reviewed in Chapter 4, many existing dense formulae have had free parameters chosen to minimise local truncation error norms over $\sigma \in [0,1]$. This is an extension of accepted local error considerations for the discrete case. However, it may be argued that as global error is the quantity of interest, and the dense approximation is not propagated, any local error introduced is of no *direct* concern. It therefore seems more natural to consider continuous behaviour of the global error within each step to motivate a choice of any free parameters.

5.6.1 Global error considerations for the RK case

Following Butcher [9] (for example), there is no loss of generality in considering an autonomous problem definition, where the independent variable x is included as a component of the $\boldsymbol{y}$ vector. Expressions for $\widehat{\boldsymbol{y}}_{n+1}$ and $\boldsymbol{y}(x_{n+1})$ in this case may be written ([9]):

$$\widehat{\boldsymbol{y}}_{n+1} = \widehat{\boldsymbol{y}}_n + \sum_{i=1}^{\infty} \frac{h_n^i}{i!} \sum_{j=1}^{r_i} \psi_j^{(i)} \boldsymbol{F}_j^{(i)}(\widehat{\boldsymbol{y}}_n),$$

$$\boldsymbol{y}(x_{n+1}) = \boldsymbol{y}(x_n) + \sum_{i=1}^{\infty} \frac{h_n^i}{i!} \sum_{j=1}^{r_i} \alpha_j^{(i)} \boldsymbol{F}_j^{(i)}(\boldsymbol{y}(x_n)).$$

where $\psi_j^{(i)} = \alpha_j^{(i)} \gamma_j^{(i)} \sum_{k=1}^{s} \widehat{b}_k \overline{\Phi}_k$, $\alpha_j^{(i)} = \alpha$, and $\gamma_j^{(i)} = \gamma$, all defined for the particular tree associated with $\widehat{\tau}_j^{(i)}$ (see Section 3.2). Using the definition of $\boldsymbol{\varepsilon}_{n+1}$,

$$\boldsymbol{\varepsilon}_{n+1} = \boldsymbol{\varepsilon}_n + \sum_{i=1}^{\infty} \frac{h_n^i}{i!} \sum_{j=1}^{r_i} \left\{ \psi_j^{(i)} \boldsymbol{F}_j^{(i)}(\boldsymbol{y}(x_n) + \boldsymbol{\varepsilon}_n) - \alpha_j^{(i)} \boldsymbol{F}_j^{(i)}(\boldsymbol{y}(x_n)) \right\},$$

and expansion to first order in $\boldsymbol{\varepsilon}_n$, using the definition of $\widehat{\boldsymbol{t}}_{n+1}$, gives

$$\boldsymbol{\varepsilon}_{n+1} = \boldsymbol{\varepsilon}_n + \sum_{i=1}^{\infty} \frac{h_n^i}{i!} \sum_{j=1}^{r_i} \psi_j^{(i)} \boldsymbol{F}_{jk}^{(i)}(\boldsymbol{y}(x_n))^k \varepsilon_n + \widehat{\boldsymbol{t}}_{n+1} + O(\|\boldsymbol{\varepsilon}_n\|^2),$$

where tensor notation has been employed in relation to the derivatives of $\boldsymbol{F}_j^{(i)}$ (see, for example, Dormand, et al. [60]). Similarly for $\boldsymbol{\varepsilon}_{n+\sigma}^*$:

$$\boldsymbol{\varepsilon}_{n+\sigma}^* = \boldsymbol{\varepsilon}_n + \sum_{i=1}^{\infty} \frac{(\sigma h_n)^i}{i!} \sum_{j=1}^{n_i} \psi_j^{(i)*} \boldsymbol{F}_{jk}^{(i)}(\boldsymbol{y}(x_n))^k \varepsilon_n + \boldsymbol{t}_{n+\sigma}^* + O(\|\boldsymbol{\varepsilon}_n\|^2), \tag{5.37}$$

where $\psi_j^{(i)*} = \alpha_j^{(i)} \gamma_j^{(i)} \sum_{k=1}^{s^*} b_k^* \overline{\Phi}_k / \sigma^{i-1}$. Noting that $r_1 = \psi_1^{(1)*} = 1$, and $\boldsymbol{\varepsilon}_n$ is $O(h^q)$, equation (5.37) yields

$$\boldsymbol{\varepsilon}_{n+\sigma}^* = \boldsymbol{\varepsilon}_n + \boldsymbol{G}_{n+\sigma}^* + \boldsymbol{t}_{n+\sigma}^* + O(h^{q+2}), \tag{5.38}$$

where $\boldsymbol{G}_{n+\sigma}^* = \sigma h_n \boldsymbol{f}_k(\boldsymbol{y}(x_n))^k \varepsilon_n$, and therefore as $\boldsymbol{t}_{n+1}^* = \widehat{\boldsymbol{t}}_{n+1}$ from continuity requirements, $\boldsymbol{\varepsilon}_{n+1}^* = \boldsymbol{\varepsilon}_{n+1}$.

In practice, if each component, $\varepsilon_{n+\sigma}^*$, of $\boldsymbol{\varepsilon}_{n+\sigma}^*$ (omitting the superscript denoting an individual component) lies between the corresponding components of $\boldsymbol{\varepsilon}_n$ and $\boldsymbol{\varepsilon}_{n+1}$, so that

$$\min\{\varepsilon_n, \varepsilon_{n+1}\} \le \varepsilon_{n+\sigma}^* \le \max\{\varepsilon_n, \varepsilon_{n+1}\}, \tag{5.39}$$

then the global error is bounded by that at the interval end points. Failure of (5.39) for a single component may be quantified by integration of the overshoot, or 'box deviation', as

$$\int\limits_{\varepsilon_{n+\sigma}^* < u} |\varepsilon_{n+\sigma}^* - u| d\sigma + \int\limits_{\varepsilon_{n+\sigma}^* > v} |\varepsilon_{n+\sigma}^* - v| d\sigma$$

where $u = \min\{\varepsilon_n, \varepsilon_{n+1}\}$ and $v = \max\{\varepsilon_n, \varepsilon_{n+1}\}$.

Considering (5.38) component-wise, it is clear that if $t_{n+\sigma}^*$ is small compared to $G_{n+\sigma}^*$, then the deviation $\varepsilon_{n+\sigma}^* - \varepsilon_n$ is well behaved, being approximately linear for small h. When $t_{n+\sigma}^*$ is comparable in size to $G_{n+\sigma}^*$, consideration should be given to the interaction of the two throughout the step $[x_n, x_{n+1}]$. The situation to be given closest attention is that in which G^*_{n+1} and t^*_{n+1} have opposite signs. Figures 5.1 and 5.2 illustrate the case, and show a (scalar) situation in which a small (absolute) continuous local error can give a poorly behaved $\varepsilon^*_{n+\sigma}$, and how, in order to satisfy (5.39), the ideal form of $t^*_{n+\sigma}$ would be linear in σ to complement the linear variation

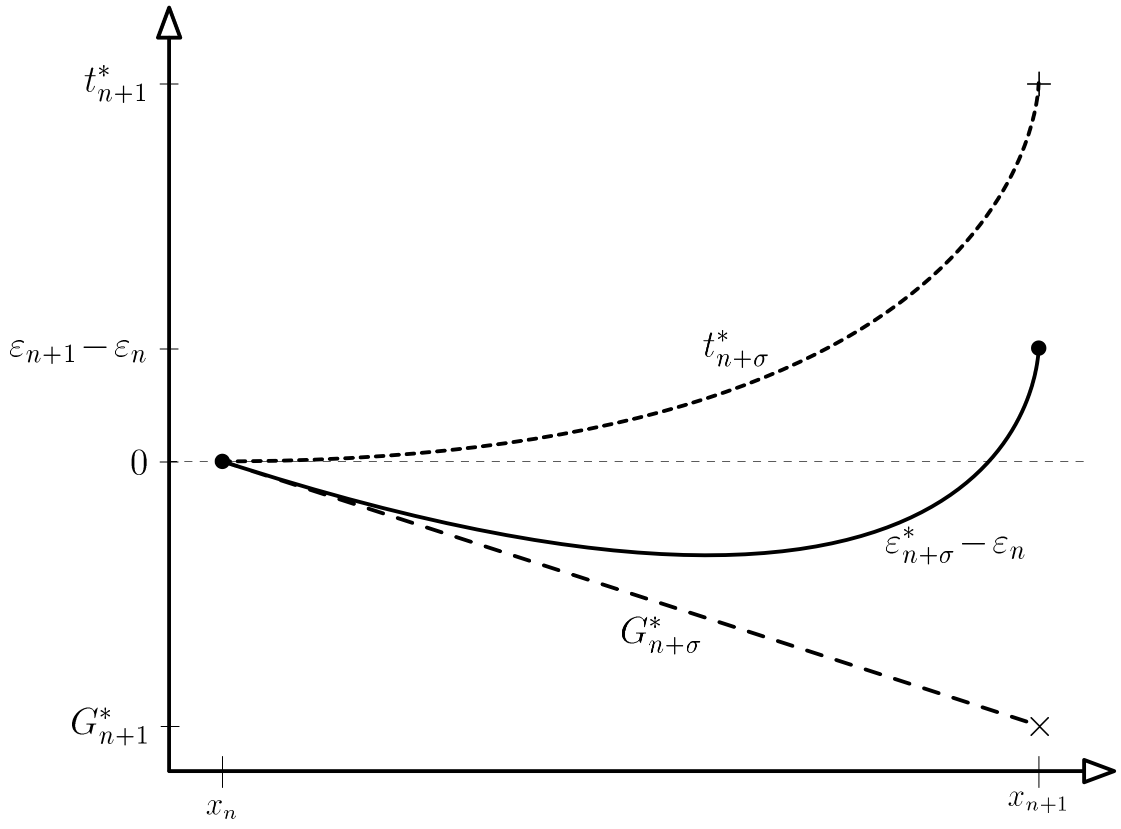

t^*_{n+1}
$\varepsilon_{n+1} - \varepsilon_n$
0
G^*_{n+1}
$t^*_{n+\sigma}$
$\varepsilon^*_{n+\sigma} - \varepsilon_n$
$G^*_{n+\sigma}$
x_n
x_{n+1}

Figure 5.1: Large 'box deviation' resulting from 'small' continuous local error.

of $G^*_{n+\sigma}$. Additionally, if t^*_{n+1} dominates G^*_{n+1}, then on average a linear variation for $t^*_{n+\sigma}$ also seems reasonable, bearing in mind that in general the relative sizes of ε_n and ε_{n+1} are unknown. Thus it is preferable to choose any free parameters to 'minimise'

$$\boldsymbol{T}^*_{n+\sigma} = \boldsymbol{t}^*_{n+\sigma} - \sigma \widehat{\boldsymbol{t}}_{n+1}. \tag{5.40}$$

By continuity, $\boldsymbol{T}^*_{n+0} = \boldsymbol{T}^*_{n+1} = \mathbf{0}$, and substituting non-autonomous expressions for

$\boldsymbol{t}^*_{n+\sigma}$ and $\widehat{\boldsymbol{t}}_{n+1}$ in (5.40) yields

$$\boldsymbol{T}^*_{n+\sigma} = \sum_{i=p^*+1}^{\infty} h_n^i \sum_{j=1}^{r_i} D_j^{(i)*} \boldsymbol{F}_j^{(i)}(x_n, \boldsymbol{y}(x_n)),$$

where $D_j^{(i)*} = \sigma^i \tau_j^{(i)*} - \sigma \widehat{\tau}_j^{(i)}$. Concentrating on the principal term in $\boldsymbol{T}^*_{n+\sigma}$, it is

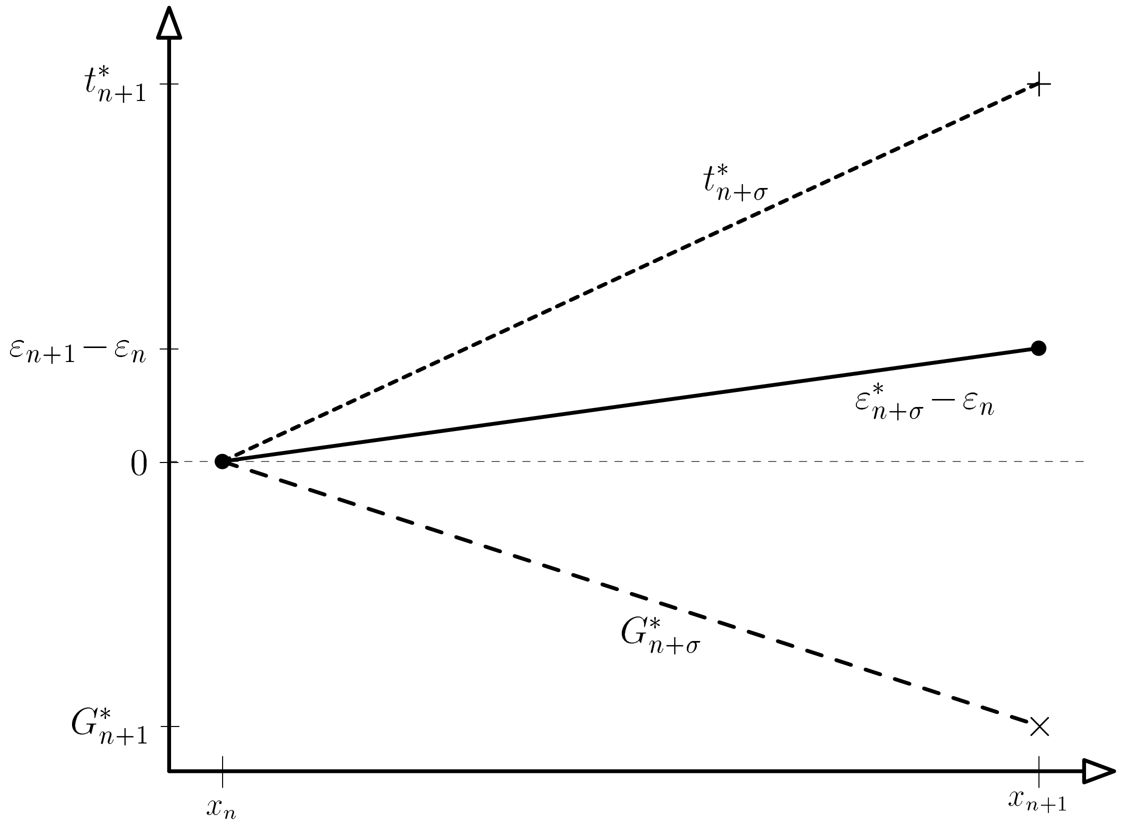

t^*_{n+1}
$t^*_{n+\sigma}$
$\varepsilon_{n+1} - \varepsilon_n$
$\varepsilon^*_{n+\sigma} - \varepsilon_n$
0
$G^*_{n+\sigma}$
G^*_{n+1}
x_n
x_{n+1}

Figure 5.2: Small 'box deviation' resulting from linear continuous local error.

reasonable to choose any free parameters to make a suitable norm of $D_j^{(p^*+1)*}$, $j = 1, 2, \ldots, r_{p^*+1}$, small for $\sigma \in [0,1]$. It should be noted that for the $p^* = q-1$ case, where $\tau_j^{(q)*} = 0$, $j = 1, \ldots, r_q$, this is the same criterion as that for minimising principal local truncation error over σ. Where secondary terms are considered, however, it can be argued that the D^*'s should be controlled rather than the τ^*'s. This can be made clearer by an example where, in a particular step, the principal terms in the local error expansion sum to be negligible for all $\sigma \in [0,1]$. In this case, the secondary terms would be the dominating effect on the global error and a linear variation for all of $\tau_j^{(q+1)*}$, $j = 1, \ldots, r_{q+1}$, would be 'safest' in general by the same argument.

5.6.2 Global error considerations for the RKN case

Similar to the RK case, considering an autonomous definition yields

$$\widehat{\boldsymbol{y}}_{n+1} = \widehat{\boldsymbol{y}}_n + \sum_{i=2}^{\infty} \frac{h_n^i}{(i-1)!} \sum_{j=1}^{n_{i-1}} \psi_j^{(i)} \boldsymbol{F}_j^{(i-1)}(\widehat{\boldsymbol{y}}_n, \widehat{\boldsymbol{y}}'_n),$$

$$\boldsymbol{y}(x_{n+1}) = \boldsymbol{y}(x_n) + h_n \boldsymbol{y}'(x_n) + \sum_{i=2}^{\infty} \frac{h_n^i}{i!} \sum_{j=1}^{n_{i-1}} \alpha_j^{(i)} \boldsymbol{F}_j^{(i-1)}(\boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)),$$

and

$$\widehat{\boldsymbol{y}}'_{n+1} = \widehat{\boldsymbol{y}}'_n + \sum_{i=1}^{\infty} \frac{h_n^i}{i!} \sum_{j=1}^{n_i} \psi_j^{'(i)} \boldsymbol{F}_j^{(i)}(\widehat{\boldsymbol{y}}_n, \widehat{\boldsymbol{y}}'_n),$$

$$\boldsymbol{y}'(x_{n+1}) = \boldsymbol{y}'(x_n) + \sum_{i=1}^{\infty} \frac{h_n^i}{i!} \sum_{j=1}^{n_i} \alpha_j^{'(i)} \boldsymbol{F}_j^{(i)}(\boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)),$$

where $\psi_j^{(i)} = \alpha_j^{(i)} \gamma_j^{(i)} \sum_{k=1}^{s} \widehat{b}_k \overline{\Phi}_k$, $\alpha_j^{(i)} = \alpha$, and $\gamma_j^{(i)} = \gamma$, all defined for the particular tree associated with $\widehat{\tau}_j^{(i)}$, and $\psi_j^{'(i)} = \alpha_j^{'(i)} \gamma_j^{'(i)} \sum_{k=1}^{s} \widehat{b}'_k \overline{\Phi}_k$, $\alpha_j^{'(i)} = \alpha$, and $\gamma_j^{'(i)} = \gamma$, defined for the tree associated with $\widehat{\tau}_j^{'(i)}$. Using the definition of $\boldsymbol{\varepsilon}_{n+1}$ and $\boldsymbol{\varepsilon}'_{n+1}$,

$$\boldsymbol{\varepsilon}_{n+1} = \boldsymbol{\varepsilon}_n + h_n \boldsymbol{\varepsilon}'_{n+1} + \sum_{i=2}^{\infty} h_n^i \sum_{j=1}^{n_{i-1}} \boldsymbol{E}_j^{(i)}, \ \boldsymbol{\varepsilon}'_{n+1} = \boldsymbol{\varepsilon}'_n + h_n \boldsymbol{\varepsilon}'_{n+1} + \sum_{i=1}^{\infty} h_n^i \sum_{j=1}^{n_i} \boldsymbol{E}_j^{'(i)},$$

where

$$\boldsymbol{E}_j^{(i)} = \frac{\psi_j^{(i)}}{(i-1)!} \boldsymbol{F}_j^{(i-1)}(\boldsymbol{y}(x_n) + \boldsymbol{\varepsilon}_n, \boldsymbol{y}'(x_n) + \boldsymbol{\varepsilon}'_n) - \frac{\alpha_j^{(i)}}{i!} \boldsymbol{F}_j^{(i-1)}(\boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)),$$

and

$$\boldsymbol{E}_j^{'(i)} = \frac{\psi_j^{'(i)}}{i!} \boldsymbol{F}_j^{(i)}(\boldsymbol{y}(x_n) + \boldsymbol{\varepsilon}_n, \boldsymbol{y}'(x_n) + \boldsymbol{\varepsilon}'_n) - \frac{\alpha_j^{'(i)}}{i!} \boldsymbol{F}_j^{(i)}(\boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)).$$

Expansion to first order in $\boldsymbol{\varepsilon}_n$, $\boldsymbol{\varepsilon}'_n$, using the definitions for $\widehat{\boldsymbol{t}}_{n+1}$ and $\widehat{\boldsymbol{t}}'_{n+1}$ yields

$$\boldsymbol{\varepsilon}_{n+1} = \boldsymbol{\varepsilon}_n + h_n\boldsymbol{\varepsilon}'_n + \sum_{i=2}^{\infty} \frac{h_n^i}{(i-1)!} \sum_{j=1}^{n_{i-1}} \psi_j^{(i)} \boldsymbol{H}_j^{(i-1)} + \widehat{\boldsymbol{t}}_{n+1} + O(\|\boldsymbol{\varepsilon}_n, \boldsymbol{\varepsilon}'_n\|^2),$$

$$\boldsymbol{\varepsilon}'_{n+1} = \boldsymbol{\varepsilon}'_n + \sum_{i=1}^{\infty} \frac{h_n^i}{(i-1)!} \sum_{j=1}^{n_i} \psi_j^{'(i)} \boldsymbol{H}_j^{'(i)} + \widehat{\boldsymbol{t}}'_{n+1} + O(\|\boldsymbol{\varepsilon}_n, \boldsymbol{\varepsilon}'_n\|^2),$$

where

$$\boldsymbol{H}_j^{(i)} = \boldsymbol{F}_{jk}^{(i)}(\boldsymbol{y}(x_n), \boldsymbol{y}'(x_n))^k \varepsilon_n + \boldsymbol{F}_{jk'}^{(i)}(\boldsymbol{y}(x_n), \boldsymbol{y}'(x_n))^{k'} \varepsilon'_n,$$

and tensor notation has again been employed in relation to the derivatives of $\boldsymbol{F}_j^{(i)}$. Similarly for $\boldsymbol{\varepsilon}^*_{n+\sigma}$ and $\boldsymbol{\varepsilon}'^*_{n+\sigma}$:

$$\begin{aligned}\boldsymbol{\varepsilon}^*_{n+1} &= \boldsymbol{\varepsilon}_n + h_n\boldsymbol{\varepsilon}'_n + \sum_{i=2}^{\infty} \frac{(\sigma h_n)^i}{(i-1)!} \sum_{j=1}^{n_{i-1}} \psi_j^{(i)*} \boldsymbol{H}_j^{(i-1)} + \boldsymbol{t}^*_{n+1} + O(\|\boldsymbol{\varepsilon}_n, \boldsymbol{\varepsilon}'_n\|^2),\\ \boldsymbol{\varepsilon}'^*_{n+1} &= \boldsymbol{\varepsilon}'_n + \sum_{i=1}^{\infty} \frac{(\sigma h_n)^i}{(i-1)!} \sum_{j=1}^{n_i} \psi_j^{'(i)*} \boldsymbol{H}_j^{(i)} + \boldsymbol{t}'^*_{n+1} + O(\|\boldsymbol{\varepsilon}_n, \boldsymbol{\varepsilon}'_n\|^2),\end{aligned} \tag{5.41}$$

where $\psi_j^{(i)*} = \alpha_j^{(i)}\gamma_j^{(i)} \sum_{k=1}^{s^*} b_k^* \overline{\Phi}_k/\sigma^{i-2}$ and $\psi_j^{'(i)*} = \alpha_j^{'(i)}\gamma_j^{'(i)} \sum_{k=1}^{s^*} b_k'^* \overline{\Phi}_k/\sigma^{i-1}$. Now, as $n_1 = \psi_1^{'(1)*} = \psi_2^{(1)*} = 1$, and $\boldsymbol{\varepsilon}_n$, $\boldsymbol{\varepsilon}'_n$ are $O(h^q)$, equations (5.41) yield

$$\begin{aligned}\boldsymbol{\varepsilon}^*_{n+\sigma} &= \boldsymbol{\varepsilon}_n + \sigma h_n \boldsymbol{\varepsilon}'_n + \boldsymbol{t}^*_{n+\sigma} + O(h^{q+2}),\\ \boldsymbol{\varepsilon}'^*_{n+\sigma} &= \boldsymbol{\varepsilon}'_n + \boldsymbol{G}'^*_{n+\sigma} + \boldsymbol{t}'^*_{n+\sigma} + O(h^{q+2}),\end{aligned} \tag{5.42}$$

where $\boldsymbol{G}'^*_{n+\sigma} = \sigma h_n \boldsymbol{f}_k(\boldsymbol{y}(x_n))^k \varepsilon_n$. To first order, the 'global error deviation' functions $\varepsilon'^*_{n+\sigma} - \varepsilon'_n$ and $\varepsilon^*_{n+\sigma} - \varepsilon_n$ are given by

$$\boldsymbol{G}'^*_{n+\sigma} + \boldsymbol{t}'^*_{n+\sigma}, \text{ and } \sigma h_n \boldsymbol{\varepsilon}'_n + \boldsymbol{t}^*_{n+\sigma},$$

respectively, and a linear behaviour for $\boldsymbol{t}^*_{n+\sigma}$ and $\boldsymbol{t}'^*_{n+\sigma}$ should be considered as ideal by an argument identical to that in the RK case. Considering the local truncation error linear deviation functions

$$\boldsymbol{T}^*_{n+\sigma} = \boldsymbol{t}^*_{n+\sigma} - \sigma\widehat{\boldsymbol{t}}_{n+1}, \tag{5.43}$$

and

$$\boldsymbol{T}'^*_{n+\sigma} = \boldsymbol{t}'^*_{n+\sigma} - \sigma\widehat{\boldsymbol{t}}'_{n+1}, \tag{5.44}$$

substituting the non-autonomous expressions for $\boldsymbol{t}^*_{n+\sigma}$, $\widehat{\boldsymbol{t}}_{n+1}$, $\boldsymbol{t}'^*_{n+\sigma}$, and $\widehat{\boldsymbol{t}}'_{n+1}$ in (5.43) and (5.44) yields

$$\boldsymbol{T}^*_{n+\sigma} = \sum_{i=p^{y^*}+1}^{\infty} h_n^i \sum_{j=1}^{n_{i-1}} D_j^{(i)*}\boldsymbol{F}_j^{(i)}(x_n, \boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)), \tag{5.45}$$

and

$$\boldsymbol{T}'^*_{n+\sigma} = \sum_{i=p^{y'^*}+1}^{\infty} h_n^i \sum_{j=1}^{n_i} D_j^{'(i)*}\boldsymbol{F}_j^{(i)}(x_n, \boldsymbol{y}(x_n), \boldsymbol{y}'(x_n)), \tag{5.46}$$

where $D_j^{(i)*} = \sigma^i\tau_j^{(i)*} - \sigma\widehat{\tau}_j^{(i)}$, and $D_j^{'(i)*} = \sigma^i\tau_j^{'(i)*} - \sigma\widehat{\tau}_j^{'(i)}$. An optimisation procedure for choice of free parameters by this global error criterion therefore should be directed towards the principal and secondary terms of (5.45) and (5.46), as in the RK case, by minimisation of norms of the 'D^*' and 'D'^*' functions.

5.6.3 Additional considerations

With reference to Chapter 4, the best choice of free parameters is rarely determined solely in terms of truncation error norms. Additional considerations may involve an aesthetic requirement for 'tidy' rational representations for the method coefficients. This is mainly to allow the possibility of presenting the method tableau in print, but also as an item of software may then contain the exact coefficients with the minimum of storage. Neither of these considerations are particularly relevant in respect of very high-order processes, but low- to high-order methods may still be sought with concise rational, or rational/surd forms.

A very real necessity when finalising a method is to ensure that no coefficients are too large (in magnitude) such that it may perform badly when implemented using a floating-point arithmetic. This is especially important for dense processes using simple polynomials, as high-degree interpolants will often require very large coefficients. Any degrees of freedom in choice of the b_i^* and $b_i'^*$'s should therefore be used to ensure that the polynomial coefficients are no bigger than necessary, possibly at cost of the truncation error norms.

In the remaining chapters, all these considerations will be taken into account during the derivation of new dense processes. Methods for obtaining optimum choices of free parameters are discussed in the following section.

5.7 Optimisation methods

For the purposes of optimising truncation error norms by the linear deviation criterion of the previous sections, it can be assumed that an 'objective function' $f(\boldsymbol{x})$ is conceivable that embodies a value to be minimised in terms of a vector of parameters $\boldsymbol{x}$. Ignoring any aesthetic requirements for concise coefficients, this would ideally also include a component that controls the size of the method parameters. If such a function is linear in its parameters, it may be minimised algebraically using a 'least squares' linear systems of equations. This is an ideal situation that does arise in practice, but more commonly can be forced to arise by making certain compromises. For control of general non-linear objective functions that may contain constraints, and possibly discontinuities, it is preferable to have available an optimisation tool that is applicable to all the cases of interest from a floating-point arithmetic environment. Choice of such a method may be based on the following possible difficulties:

- Large numbers of parameters may be involved in the search.
- A general objective function may not be continuous or differentiable at all points in the function space.
- Functions containing high-order truncation error norms, especially of a dense process, may only be calculable to very low accuracy.

These points have led to the development of an optimisation algorithm that can perform under very basic assumptions of smoothness and accuracy of its objective function. It is based upon a 'direct search' methodology whereby no attempt is made to use or approximate derivatives of the objective function, and is an overlying modification to the MDS (Multi-directional search) simplex[2] method of Torczon [61]. Assuming the availability of fast machines in which to perform the searches, the MDS method is capable of convergence onto single local minima with acceptable efficiency for reasonably high dimensional spaces (≤ 50), and is very robust in the presence of high 'noise' and discontinuities in the function (Torczon [61], [62] and [63]). The method will, however, converge onto points of non-differentiability or discontinuity and has no method of restarting without outside interaction. The following section deals with the new method, designed to allow a more automated approach to finding multiple local minima with MDS.

5.7.1 GLOBS: a modified MDS optimisation code

Specifics of the MDS 'descent method' will be left to the literature ([61], [62] and [63]). It is sufficient to understand that it will search the parameter space from a given starting position and converge onto a point satisfying specified stopping criterion. These are based on having found a region of space where the objective function $f(\boldsymbol{x})$ (not to be confused with the f from an ODE problem definition) is changing sufficiently slowly to indicate the presence of a local minima. An important characteristic of MDS, however, is that it never requires function values in any sense other than for *comparison* purposes — a major factor in the robustness of the algorithm. The challenge, therefore, was to extend practicability of the method in finding multiple local minima without loss of this desirable feature. The way in which GLOBS achieves this automated search with MDS is to log all points at which the method converges, and in some sense 'avoid' them whilst still searching for function descent.

[2]A simplex is a non-degenerate geometrical form used by the optimisation, and should not be confused with the simplex method for linear programming.

It is convenient to introduce the term 'stopping point' (or just 'stop') to be used instead of 'local minima', because the MDS method itself does not guarantee such generality. As such, GLOBS contains two distinct, but related, schemes for avoiding the MDS stops. Both are based on intuitive analogies. The first is an idea akin to 'plugging' holes in the function space with localised decaying exponential functions that are placed at each stop. The second is analogous to a physical situation involving 'repelling' objects. Consider the MDS simplex to be a magnetically charged particle acting under gravity, such that it will find its lowest level with no other interaction. If a similarly charged particle is placed underneath it each time it stops, it will be forced away with an inverse-square law repulsion until it is balanced by the increasing function value. These two ideas form a (loosely termed) 'penalty function' $g(\boldsymbol{x})$ at every point in the space — the optimisation is then performed on $h(\boldsymbol{x}) = g(\boldsymbol{x}) + f(\boldsymbol{x})$ by MDS as usual. More formally, if $f(\boldsymbol{x}) \in \mathbb{R}^n \rightarrow \mathbb{R}$, and the stops are dynamically represented by $\boldsymbol{m}_i,\ i = 1, \ldots, p$, the penalty function in the two cases is given by

1. *Exponential plugging*

$$\boldsymbol{g}(x) = G_1 \sum_{i=1}^{p} e^{G_2\sqrt{X_i}},\ G_1 > 0,\ G_2 < 0,$$

2. *Repulsion*

$$\boldsymbol{g}(x) = G \sum_{i=1}^{p} 1/X_i,\ G > 0,$$

where, in both cases,

$$X_i = ||\boldsymbol{x} - \boldsymbol{m}_i||_2^2.$$

The constants G, G_1 and G_2 must be specified at the onset of each run, but are modified at relevant points in the execution. For example, after any stop that does not improve on f for the previous one, G (or G_1) is doubled and G_2 is halved such that both methods provide more effective propulsion from previous stops. Specific implementation of these ideas may be gleaned from the code itself, provided in full in Appendix C.

A construction that aids initial choice of the G and G_2 quantities is the idea of an 'effective width'. Considering the action of a single stopping point, i.e., with $p = 1$ above, both forms of $g(\boldsymbol{x})$ decay as $X = X_1$ increases, and there is a point, $w/2$, at which g becomes equal to the function tolerance being used by MDS, T. For the decaying exponential case,

$$w = \frac{2\ln(T/G_1)}{G_2}, \tag{5.47}$$

and for repulsion,

$$w = \frac{2G}{T}. \tag{5.48}$$

The quantity w is therefore the 'width' of the effective region of the schemes, and for this reason is given to GLOBS on initialisation instead of G or G_2 (see the code for more details). G_1 is the height of each plug when exponentials are being used, and is specified in this form. There now follows a set of simple examples to demonstrate the new method, followed by a short discussion.

Example 5.7.1 (A bounded scalar test function) In order to allow a graphical representation of the optimisation procedure, a scalar example is ideal as progress can easily be plotted. A suitable test example is the objective function

$$f(x) = \begin{cases} |\sin(\pi x)| - |x|, & |x| \le 4 \\ \infty, & |x| > 4, \end{cases}$$

which has non-differentiable local minima of height $-|x|$ at $|x| = 0, \ldots, 4$, and is effectively bounded by the 'walls' at $|x| = 4$. The infinite function value may, of course, be specified as just $\gg 1$ for this purpose. GLOBS was run on the above function using a maximum of 50 stops, function and step tolerances of 1×10^{-6} (see the code), starting point $x = 0$, and a variety of global schemes, with varying G, G_1 and G_2. The results are shown graphically in Figures 5.4–5.7, the Fortran 90 calling program given in Figure 5.3 having been used in all cases. The scalar case has $n = 1$ in the code (the function being more general to accommodate the next example) with height $h = G_1$ and effective width w being shown in the figures. G_2 or G is calculated

```
PROGRAM Globs_Demo

Use Search

Double precision, allocatable :: V(:), Z(:)
Double precision :: fm, F, h, w

external F

!NB: not necessary to set N here - done for use by F(x).

n = ????? ; Allocate (V(n), Z(n)) ; V = 0.0D0 ; h = ????? ; w = ?????

!GLOBS(F, R(2), V, M, G(2), TOL(2), SS, FMIN, Z, T, USE, INT, D, NAME, P)

call Globs(F, (/0, 0/), V, 50, (/h, w/), (/1.0D-6, 1.0D-6/), 1.0D-4, fm, &
  & Z, -1.0D0, .false., .false., 0.0D0, "demo", 1)

!NB: output in Z & fm.

END PROGRAM Globs_Demo

FUNCTION F(x)

Use Search

Double precision, parameter :: pi = 3.14159265358979323846264 3D0
Double precision :: x(n), F

If (Maxval(Abs(x)) <= 4) then
  F = Sum(Abs(sin(x * pi)) - Abs(x))
else
  F = Huge(1.0D0)
End If

END FUNCTION F
```

Figure 5.3: The Fortran 90 GLOBS demo code.

internally from (5.47) or (5.48), respectively, with $T = 1{\times}10^{-6}$.

Examples using the inverse-square law repulsion scheme in Figures 5.6 and 5.7 show that the method has become 'stuck' in the minima around $x = 1$ and $x = -4$, respectively. This is not a flaw of the scheme, and can be accounted for by again considering the electromagnetic analogy. If the scalar case is thought of as a charged *ring* on a wire it can be imagined that, if the ring gets caught between two similarly charged particles, it will be pushed with a larger and larger force into the space between them, indefinitely. Therefore, if the ring *ever* slips between two stopping points on the wire this blockage can occur. This situation will arise frequently in the scalar case, as immediately after a stop it is never certain which way the ring will go due to the discrete nature of the MDS process. This is therefore unavoidable. Such a failing is not easily conceived in higher dimensional examples, and has never been seen in practice when $n > 1$. The method of adding decaying exponentials to the scalar function does not contain any possibility of getting stuck, and progress under this scheme is more predictable (Figures 5.4-5.5).

$\square$

Example 5.7.2 (A Multi-variate test function) The function from Example 5.7.1 can easily be extended to higher dimensions by simply adding a scalar function for each dimension, i.e.,

$$f(\boldsymbol{x}) = \begin{cases} \displaystyle\sum_{i=1}^{n} |\sin(\pi\,{}^{i}x)| - |{}^{i}x|\,, & \max_i |{}^{i}x| \le 4 \\ \infty & , \max_i |{}^{i}x| > 4 \end{cases} \quad , \text{ for } \boldsymbol{x} = \{{}^{1}x, {}^{2}x, \ldots, {}^{n}x\}\,.$$

Such a function has local minima with $f(\boldsymbol{x}) \in \{-4n, \ldots, 0\}$ for $\boldsymbol{x}$ having all integer components, and is non-differentiable along all n orthogonal axis directions leading away from each minima. Tests have been performed at many values of n, the case of $n = 8$, however, shows enough detail for demonstration purposes. The actual parameter space can not be shown (easily ?) and only plots of the objective function

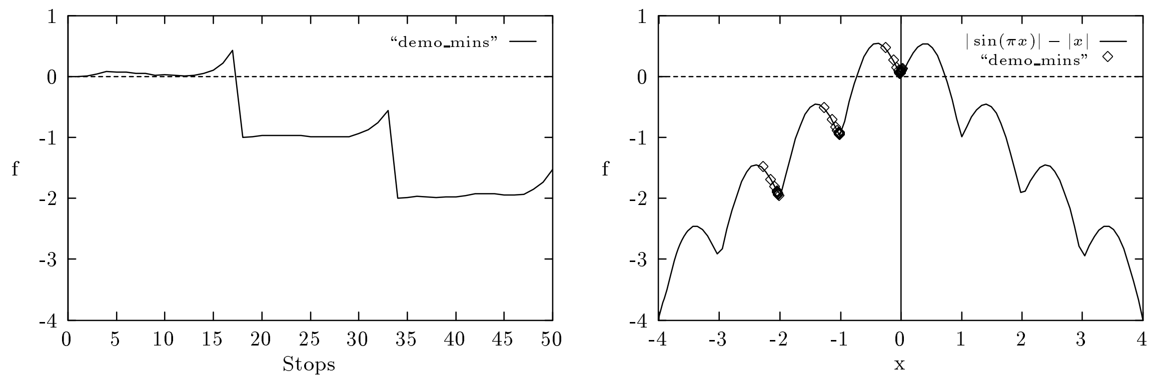


Figure 5.4: Exponential demo 1 — $h = G_1 = 5{\times}10^{-2}, w = 5{\times}10^{-2}$.

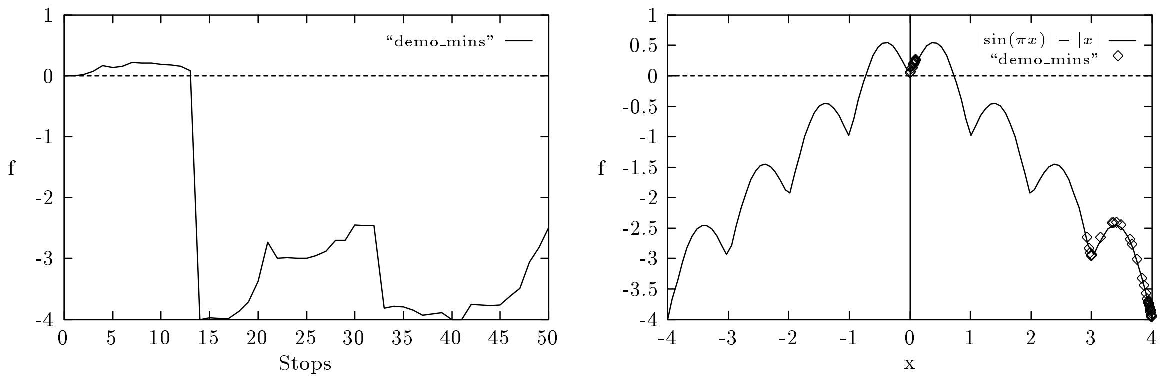


Figure 5.5: Exponential demo 2 — $h = G_1 = 1{\times}10^{-2}, w = 1{\times}10^{-2}$.

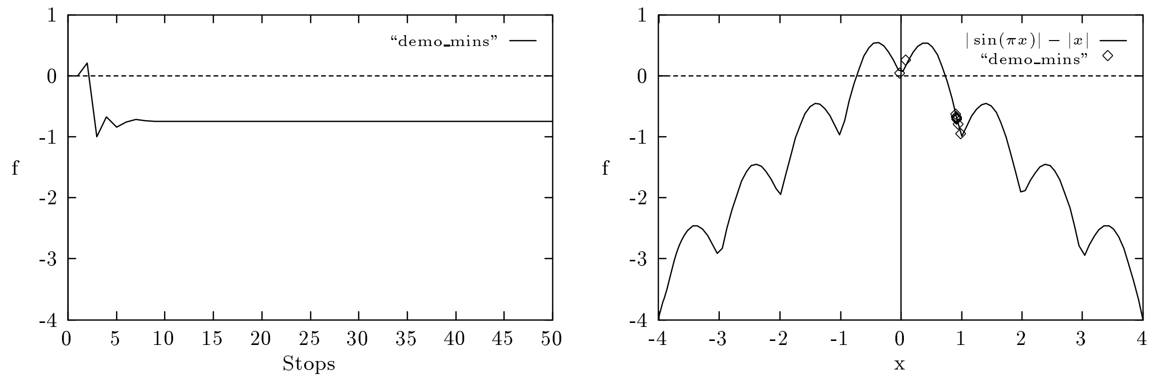


Figure 5.6: Repulsion demo 1 — $w = 2{\times}10^{3}$.

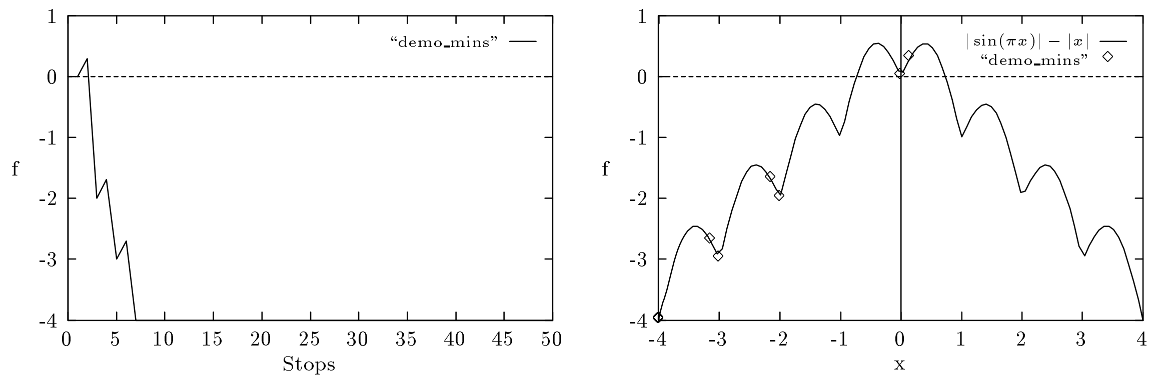


Figure 5.7: Repulsion demo 2 — $w = 6{\times}10^3$.

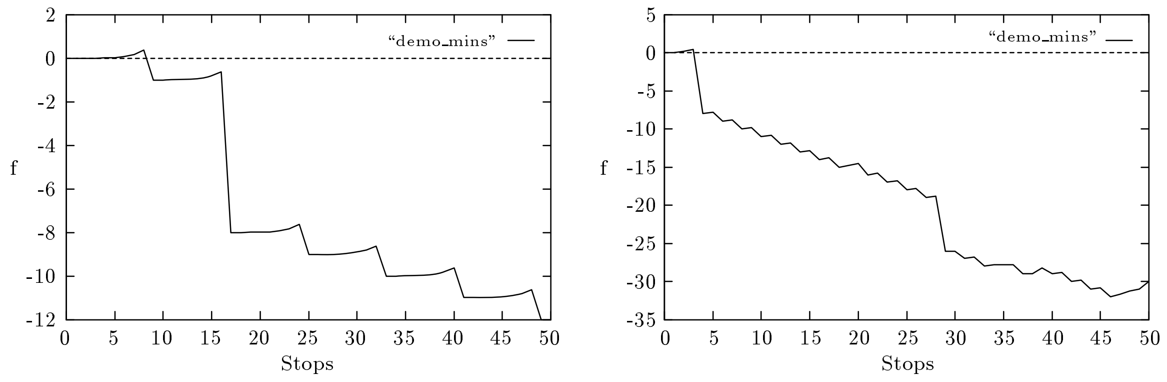


Figure 5.8: Exponential demos — $h = w = 1{\times}10^{-2}$, and $h = w = 1$, respectively.

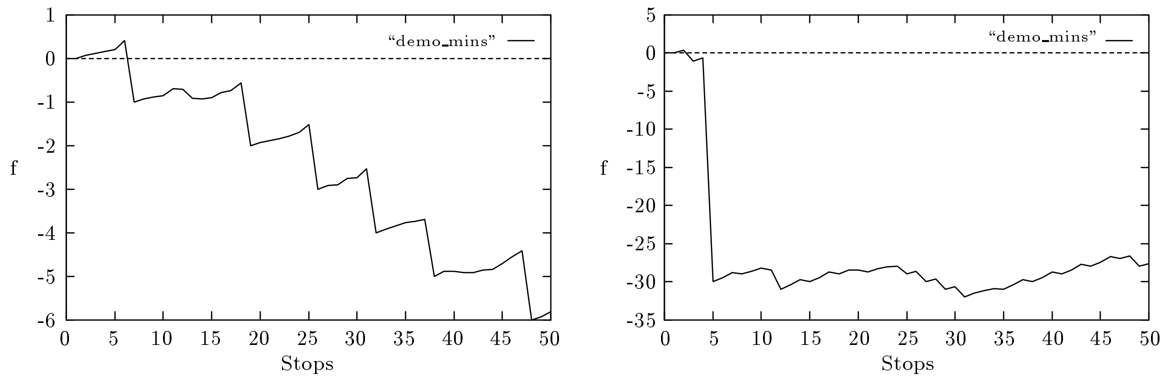


Figure 5.9: Repulsion demos — $w = 1{\times}10^2$ and $w = 1{\times}10^4$, respectively.

at each of 50 stopping points are given. All implementation details are as in Example 5.7.1, using the same code, but with $n = 8$. Graphs of four different tests are shown in Figures 5.8 and 5.9, noting that in the second case for each of the two schemes, the global minimum of $f(\boldsymbol{x}) \approx -32$ was successfully found.

$\square$

It should be clear that choice of G, G_1 and G_2 is problem dependent, and best done from experimentation or practice. The optimisation of RK(N) dense processes in this work, being a large scale task, has allowed the experimentation needed to use GLOBS effectively. The multi-variate example (5.7.2) suggests no preference for either search scheme, both showing predictable behaviour. Other than for the scalar case, the repulsion scheme has, in fact, provided excellent results in practice. It is this form of the method that has been used for the optimisation of all dense processes from the remaining chapters, and has proved to be most useful as a tool in conjunction with specific techniques tailored to individual processes, mainly through the graphing of intermediate results and occasional human interaction — such strategies and general practical notes are described next.

5.7.2 Practical notes

While the GLOBS code acts as a convenient tool for achieving automated descent of a given objective function, the nature of RK(N) continuous truncation error terms and parameters enables individually tailored usage to be more fruitful. Such techniques may be summarised as follows:

- *Starting parameters.* It is always required that an initial set of free parameters is provided as a first approximation. For RK(N) parameter searches, the most useful technique for such a choice has been to derive a complete formula using a random, but simple rational, set of free parameters (in MACSYMA from the algebraic model) and have its truncation error characteristics noted. Principal and secondary norms of local error and linear deviation quantities can be generated with RK-AID for this purpose. If this set of free parameters is then used as a starting position for the GLOBS search, the numerical model can be validated by calculation of error norms through *separate* means, and is possibly the only way that a lack of programming errors can be ensured.

- *Restarts.* When performing many MDS runs using GLOBS, it is possible that the simplex may take an unsuitable path where subsequent stops are of increasingly *worse* values of $f(\boldsymbol{x})$ than at a previous stop. Of course, this does not discount future descent, but after some time the path may be considered a failure and the code restarted. In this case, GLOBS will allow the run to continue from the best point in the path. All previous stopping points are still held (see the code in Appendix C for more details), and will be avoided as usual — the simplex is therefore unlikely to move in the same direction as it did previously at this point, and a better path is possible.

- *Constraints.* In some cases it may be convenient to impose a maximum size on a norm quantity, especially $b_i^*/b_i'^*$ polynomial coefficient sizes. Also, the ratio of secondary to principal linear deviation norms may be restricted to be under a certain size. Such constraints can be imposed by simply setting the objective function to a value $\gg 1$ such that the simplex is 'boxed' into the required region of parameter space. This can only be done when the norms/ratios are already under the desired size, and this may initially be achieved by minimising such quantities in isolation (see *control of parameter size* and *secondary norms* below). A constraint may also be imposed on the minimum separation of additional and discrete c_i values, $1{\times}10^{-3}$, say, such that the model does not become singular. While this would tend to produce high truncation error norms, and so is unlikely to be acceptable, the objective function will become ill-conditioned and problems may occur. A final need for constraint is when solution of the linear systems for 'α_i', a_{ij} and $b_i^*/b_i'^*$ parameters fails due to ill-conditioning (trapped by the presence of small pivots during Gaussian-elimination). In this case a function value of $\gg 1$ may also be used, and such points will be avoided by the simplex.

- *Node partitions.* When small numbers of additional stages are concerned (specifically for the sixth-order Nyström cases of Chapter 6), a separate minimisation may be performed for all permutations of the c_i ordering such that $c_i,\ i = s+1, \ldots, s^*$, are not allowed to cross boundaries formed by the discrete c_i values ($i = 1, \ldots, s$). This may be achieved via the imposition of constraints as above.

- *Control of parameter size.* In conjunction with the enforcement of constraints, a preliminary minimisation may be done on absolute parameter size in isolation, i.e., with no component of truncation error norms or ratios, until they are of a desired size. This is especially relevant with respect to $b_i^*/b_i'^*$ polynomial coefficients, where large values may cause rounding errors in calculation of truncation error norms to an unacceptable level. In some cases, for the very high-degree polynomial work (of Chapter 8, for example), only a couple of decimal places of accuracy are possible in double-precision ($\approx$ 16 figure) arithmetic, and control of the polynomial coefficients is of primary importance.

- *Parameter subsets.* Good results have occasionally been obtained from minimisation over subsets of the parameters, just the extra c_i's, for example, such that a quicker descent may be achieved in early stages of the optimisation. Also, c_i's and $b_i^*/b_i'^*$ quantities may be 'frozen' and just a_{ij} coefficients modified. In general, however, minimisation over complete free parameter sets has given the best results, even when their numbers are 40–50. A large number of free parameters may often be considered as a disadvantage when treated algebraically, but given general routines as here, this has only been seen to improve the possibility of finding an acceptable final model.

- *Continuous norm discretisations.* The integration of all continuous norms of truncation error and linear deviation quantities requires use of a quadrature rule. In all cases, a simple trapezoidal scheme will be used over a varying number of samples, depending on the accuracy required. The number of discrete points taken over $\sigma \in [0,1]$ will also affect the speed of the numerical process, and so a compromise must usually be found. Since at least 2–3 decimal places of accuracy is always required, numbers of points in the range 15–25 is reasonable for most cases, but may be increased when the optimisation gets to such a stage where fine detail in the objective function is necessary.

- *Secondary norms.* Secondary norms are considered in every case by monitoring the ratio of secondary to principal (linear deviation) quantities. At the onset of an optimisation procedure, it has been useful to minimise this ratio in isolation in order that a constraint can be imposed (see above). An interesting alternative solution to the control of secondary terms has also been to initially minimise the secondary terms *only*. In most cases, the principal terms will also be controlled as a consequence and rarely will they increase to an unacceptable level.

- *Descent thresholds.* Where objective functions are comprised of truncation error norms of finite precision, it is possible that progress of the MDS simplex could be hindered by it descending only in *noise*. This could be caused by numerical rounding errors or the approximate nature of the integration quadrature rule. To combat this, a descent threshold has been imposed in every case to be above the expected level of accuracy. If an objective function of $O(1{\times}10^{-6})$ is relevant, and only the third decimal place is of concern (or known to be accurate), a descent threshold of $1{\times}10^{-10}$ would be used. In cases where descent is not found, this threshold can be decreased — the objective function gradient may just be very slight in the current region of parameter space.

- *Reduction of continuous norm oscillations.* In some cases, the search for *monotonic* continuous truncation error norms has helped the eventual search for small linear deviation quantities. This can be accounted to reduction in oscillations of the continuous quantities about the zero axis, as truncation error coefficients of constant *sign* imply a simpler behaviour of linear deviation objective functions being minimised.

- *Completion.* The nature of the indefinite search that GLOBS can perform (in an unbounded, infinite parameter space) means that it could run forever assuming enough machine storage to keep all the stopping points. It should therefore be clear that as a global minimum of any chosen objective function is unknown, a real difficulty is knowing when to stop. This can only be gauged by the rate at which descent is being achieved, and the amount of time available. The chosen method has been to derive *and test numerically* a completed formula at various stages of the optimisation, and to stop when test results appear not to reflect the improvement in theoretical characteristics (see Section 5.8).

In the remaining chapters, no specific details will be given, and phrases such as "a GLOBS optimisation was performed" will be used freely on the understanding that many hours of experimentation with GLOBS parameters, varying objective functions, and combinations of the above techniques may have been required.

5.8 Numerical testing

As the numerical testing of dense output methods is fundamental to their derivation, a test suite has been coded and included into the RK-AID package. Development of the global error criterion derived in this chapter was motivated by testing in an attempt to explain the observed behaviour of continuous global error within each step. As such, numerical tests will be based on the degree of global error 'box deviation' control achieved by the criterion presented.

5.8.1 Scaled box deviation (SBD) data

In a single step, a box deviation vector $\boldsymbol{\beta}_n$ is defined to have components

$$^{i}\beta_n = \int\limits_{^{i}\varepsilon^*_{n+\sigma} < {}^{i}u_n} |{}^{i}\varepsilon^*_{n+\sigma} - {}^{i}u_n| d\sigma + \int\limits_{^{i}\varepsilon^*_{n+\sigma} > {}^{i}v_n} |{}^{i}\varepsilon^*_{n+\sigma} - {}^{i}v_n| d\sigma, \; i = 1, \ldots, m,$$

where m is the number of problem components, ${}^iu_n = \min\{{}^i\varepsilon_n, {}^i\varepsilon_{n+1}\}$ and ${}^iv_n = \max\{{}^i\varepsilon_n, {}^i\varepsilon_{n+1}\}$. Equivalently for a continuous RKN, define $\boldsymbol{\beta}'_n$ to have components

$$ {}^i\beta'_n = \int\limits_{{}^i\varepsilon'^*_{n+\sigma} < {}^iu'_n} |{}^i\varepsilon'^*_{n+\sigma} - {}^iu'_n| d\sigma + \int\limits_{{}^i\varepsilon'^*_{n+\sigma} > {}^iv'_n} |{}^i\varepsilon'^*_{n+\sigma} - {}^iv'_n| d\sigma, \; i = 1, \ldots, m, $$

where ${}^iu'_n = \min\left\{{}^i\varepsilon'_n, {}^i\varepsilon'_{n+1}\right\}$ and ${}^iv'_n = \max\left\{{}^i\varepsilon'_n, {}^i\varepsilon'_{n+1}\right\}$. The scaled (accumulated) box deviation of a continuous integration (for a single component) is now defined as

$$ {}^i\overline{\beta} = \frac{\sum\limits_n {}^i\beta_n}{\sum\limits_n |{}^i\varepsilon_{n+1}|}, \; {}^i\overline{\beta}' = \frac{\sum\limits_n {}^i\beta'_n}{\sum\limits_n |{}^i\varepsilon'_{n+1}|}, $$

for y and RKN y' processes, respectively. Scaling by the mean absolute discrete global error in each case is done to give a higher level of problem independence. A one-norm (mean) may now be taken over the problem components to give single statistics SBD $= ||\overline{\boldsymbol{\beta}}||_1$ and SBD$' = ||\overline{\boldsymbol{\beta}}'||_1$. Note that the RK-AID command for such a quantity is `1 NORM`, `BOX DEVIATION`, and the integration over $\sigma \in [0, 1]$ in each step must be done via a quadrature rule. In the Nyström case, SBD *or* SBD$'$ quantities will often simply be referred to by the acronym SBD.

It is useful here to anticipate behaviour of the SBD device. Clearly, for two methods of the same continuous order applied to the same discrete points, a comparison of experimental SBD values should on average reflect success of the theoretical linear deviation criteria. When the continuous local order is different between two methods to be compared, again with the same discrete formula, asymptotic effects should dominate. Being scaled by mean discrete global errors, SBD in effect measures a local quantity. For example, $p^* = q - 1$ output will give a continuous global error the same order as the discrete points, i.e., order h^q, and the accumulated SBD will be $O(1)$. Equivalently, $p^* = q$ output will yield SBD $= O(h)$, and decreasing tolerance runs should give increasingly better continuous global error (smaller SBD).

Chapter 6

Continuous sixth-order RKN processes

6.1 Introduction

This chapter considers a useful class of moderate order Nyström formulae that are in common usage. Sixth-order processes may be derived as minimal-stage FSAL methods with $s = 6$, containing fourth-order error estimating formulae as embedded processes. Two particular formula pairs of interest are the 'RKN6(4)6FM' (Dormand, et al. [19]) and the 'RKN6(4)6FD' (Dormand and Prince [15]), for which sixth-order dense output is now considered.

6.2 Order conditions

There are 21 relevant continuous order conditions to be considered when deriving a sixth-order RKN model. These are best shown via a parameterised 'Q' form, containing only 10 'classes' of expression (for solution to zero):

```
Nystrom model
[s'] order :6, parameterise
[s] order :6, parameterise
```

1. $\sum_{i=1}^{s^*} b_i^* - \frac{1}{2}$

2. $\sum_{i=2}^{s^*} b_i^* c_i^k - \frac{\sigma^k}{(k+2)(k+1)}, \quad k = 1:4$
3. $\sum_{i=2}^{s^*} b_i^* c_i^k Q_{i1}, \quad k = 0:1$
4. $\sum_{i=2}^{s^*} b_i^* Q_{i2}$

---

5. $\sum_{i=1}^{s^*} b_i'^* - 1$
6. $\sum_{i=2}^{s^*} b_i'^* c_i^k - \frac{\sigma^k}{k+1}, \quad k = 1:5$
7. $\sum_{i=2}^{s^*} b_i'^* c_i^k Q_{i1}, \quad k = 0:2$
8. $\sum_{i=2}^{s^*} b_i'^* c_i^k Q_{i2}, \quad k = 0:1$
9. $\sum_{i=2}^{s^*} b_i'^* Q_{i3}$
10. $\sum_{i=3}^{s^*} b_i'^* \sum_{j=2}^{i-1} a_{ij} Q_{j1}$

Only the y' terms (5)–(10) above will be considered in detail, as those for y may consequently be dealt with trivially. Six b'^*'s are needed to zero the quadrature expressions — (5) and (6) above. An equivalent form for the non-quadrature conditions may be written:

$$\sum_{i=2}^{s^*} b_i'^* c_i^k \alpha_{i1} = 0,\ k = 0,1,2,\ \ \sum_{i=2}^{s^*} b_i'^* c_i^k \alpha_{i2} = 0,\ k = 0,1, \\ \sum_{i=2}^{s^*} b_i'^* \alpha_{i3} = 0,\ \sum_{i=3}^{s^*} b_i'^* \alpha_{i4} = 0, \tag{6.1}$$

where

$$\alpha_{i1} = Q_{i1},\ \alpha_{i2} = Q_{i2},\ \alpha_{i3} = Q_{i3},\ \alpha_{i4} = \sum_{j=2}^{i-1} a_{ij} \alpha_{i1}.$$

In the notation of Chapter 5, this problem has

$$n = s^r = 7,\ n_q = 5,\ n_\alpha = 4,\ n_Q = 3,\ s^a = s_i^a = 2,\ n_a = 1,\ \boldsymbol{\nu}_i^d = (\alpha_{i1}),\ \boldsymbol{\nu}_i^r = (\alpha_{i4}),$$

$\boldsymbol{r}_i = \left(\frac{1}{6}c_i^3 + \alpha_{i1}, \frac{1}{12}c_i^4 + \alpha_{i2}, \frac{1}{20}c_i^5 + \alpha_{i3}\right)^T, \ \boldsymbol{\nu}_i^b = \left(\alpha_{i1}, c_i\alpha_{i1}, c_i^2\alpha_{i1}, \alpha_{i2}, c_i\alpha_{i2}, \alpha_{i3}, \alpha_{i4}\right)^T.$

The s^b and q_i quantities are specific to the model, depending on simplifying conditions imposed in the two cases.

It is now required to solve (6.1) in as few extra stages as possible ($s^* \geq 6$), particularly for each of the two formula pairs of interest as the first six stages. At this point it is therefore worth summarising the properties of these method pairs, as follows:

1. RKN6(4)6FM (Dormand, et al. [19])

   - A six-stage discrete FSAL formula making no use of any simplifying conditions, i.e., no Q_{ik} parameters are zero ($q_i = 1, \ \forall i$).
   - Chosen from a three parameter family.
   - One extra stage is required for fifth-order dense ($p^{y^*} = 6$) [15].

2. RKN6(4)6FD (Dormand and Prince [15])

   - A six-stage discrete FSAL formula employing simplifying conditions

     $$b_2' = \widehat{b}_2' = \widehat{b}_2 = 0, \ Q_{i1} = Q_{i2} = 0 \ (q_i = 3), \ i = 3, \dots, 6.$$

     Two of $c_i, \ i = 2, \dots, 5$, were required to be constrained for this purpose.
   - Chosen from a one parameter family.
   - Fifth-order dense is possible with $s^* = 6$ (again, $p^{y^*} = 6$) [15].
   - This triple is contained in a commercial RKN code, Brankin, et al. [64] — the coefficients for which are presented in Example A.3.3.

These formulae highlight how the underlying discrete pair influences any dense process to be added as a continuous extension. Specifically, that more stages may be required to yield a dense formula of a desired order when less simplifying conditions have been made. As such, it may be expected that a sixth-order continuous extension would require less additional stages for the RKN6(4)6FD. In fact, this is not the case — though derivation of the RKN6(4)6FD dense formula is much *simpler*.

6.3 Sixth-order dense for the RKN6(4)6FD

6.3.1 Continuous output on y'

Initially, it is important to consider whether the simplifying conditions for the discrete need to be extended to the dense model. It will at present be assumed that

$$b_2'^* = 0,\ \alpha_{i1} \equiv Q_{i1} = \alpha_{i2} \equiv Q_{i2} = 0,\ i = 7, \ldots, s^*, \tag{6.2}$$

need to be imposed, and the possibility of doing otherwise discounted later. In this case, $s^b = q_i = 3,\ i = 7, \ldots, s^*$, and from (6.1) the (abbreviated) non-quadrature system for $b_i'^*,\ i = 3, \ldots, s^*$, is now

$$\overline{M} = \left(\begin{array}{ccccc|c} \frac{835128061\,R-78694563299}{138396093750000} & -\frac{197712061\,R+25940272424}{23066015625000} & \frac{9793\,R+6784963}{1200000000} & 0 & \alpha_{k3} & 0 \\ \frac{835128061\,R-78694563299}{166075312500000} & \frac{112814\,R-263549}{123018750000} & -\frac{142757\,R-11443813}{7200000000} & 0 & \alpha_{k4} & 0 \end{array} \right),$$

where $R = \sqrt{8581}$, and the penultimate column should be expanded over $k = 7, \ldots, s^*$. Equivalently, the $a_{ij},\ j = 2, \ldots, i-1$, systems are now written

$$\overline{A}_i = \left(\begin{array}{cccccc|c} -\frac{R-209}{900} & -\frac{R-209}{450} & \frac{R+209}{450} & \frac{9}{10} & 1 & c_k & \frac{1}{6}c_i^3 \\ -\frac{209\,R-26131}{405000} & -\frac{209\,R-26131}{101250} & \frac{209\,R+26131}{101250} & \frac{81}{100} & 1 & c_k^2 & \frac{1}{12}c_i^4 \\ -\frac{17453\,R-1813702}{91125000} & -\frac{17453\,R-1813702}{11390625} & \frac{17453\,R+1813702}{11390625} & \frac{729}{1000} & 1 & c_k^3 & \frac{1}{20}c_i^5 + \alpha_{i3} \\ \frac{17453\,R-1813702}{546750000} & 0 & 0 & 0 & 0 & 0 & \alpha_{i4} \end{array} \right),$$

with k subscripts expanded over $7, \ldots, i-1$. As none of $\overline{M}$ or $\overline{A}_i,\ i = 7, \ldots, s^*$, are over-determined, no further analysis needs to be done and the model is essentially complete with $s^* = 9$ (seven stages for the quadrature equations with $b_2'^* = 0$, and two for $\overline{M}$). This determines $b_i'^*,\ i = 3, \ldots, 9$, and $b_1'^*$ as fifth-degree polynomials in σ in terms of $c_i, \alpha_{i3}, \alpha_{i4},\ i = 7, \ldots, 9$. $\overline{A}_i$ will determine four of $a_{ij},\ j = 2, \ldots, i-1$, in terms of the others, c_i, α_{i3}, and α_{i4}, for $i = 7, \ldots, 9$. For the purposes of a simple optimisation procedure, two a_{ij}'s will be reclaimed as free parameters using α_{i3} and α_{i4}. As such, $\overline{A}_i$ will determine $a_{i,i-2}$ and $a_{i,i-1}$, requiring choice of α_{i3} and α_{i4} for consistency[1]. The 15 free parameters for this model are therefore:

$$c_7, c_8, c_9, a_{72}, a_{73}, a_{74}, a_{82}, a_{83}, a_{84}, a_{85}, a_{92}, a_{93}, a_{94}, a_{95}, \text{ and } a_{96}. \tag{6.3}$$

Briefly, it is interesting to consider not making the conditions (6.2). Firstly, $b_2'^* = 0$ must clearly be imposed for solution of the equation $b_2'^* Q_{21} = 0$, as $Q_{21} \neq 0$. For distinct c_i, $i = 7, \ldots, 9$, the three equations

$$\sum_{i=7}^{9} b_i'^* c_i^k \alpha_{i1} = 0, \;\; k = 0, 1, 2,$$

imply $\alpha_{i1} \equiv Q_{i1} = 0$, $i = 7, \ldots, 9$, and at least one extra stage would be required otherwise. For no c_i's do α_{i2}, $i = 7, \ldots, 9$, *have* to be zero, though again an extra stage would be needed to solve

$$\sum_{i=7}^{9} b_i'^* c_i^k \alpha_{i2} = 0, \;\; k = 0, 1,$$

otherwise, as required. In future, all simplifying conditions for the discrete will be extended to the continuous model in this way without justification.

Choice of free parameters

Given the availability of consistent systems for $b_i'^*$ and a_{ij} parameters, it is a simple matter to pass these systems to Fortran 90 and solve them by Gaussian-elimination and back substitution having specified all parameters in (6.3). It is now required to form an objective function for minimisation using GLOBS. For this purpose, a set of reduced principal and secondary truncation error coefficients for y'^* can be generated using the RK-AID command set

[1]Clearly, $\alpha_{i4} = a_{i2}(17453\,R - 1813702)/546750000$.

```
title: FD69_err
!
Nystrom model
[s'] order 7:8, full reduction
use Q's
Q(3:s, 1;2) = 0 & b[s'](2) = 0
```

giving the LaTeX output in Figures 6.1 and 6.2. While such output is useful for refer-

$$\tau_1'^{(7)*} = \tfrac{1}{720}\sum_{i=3}^{s^*} b_i'^* c_i^6/\sigma^6 - \frac{1}{7!}$$

$* \quad \tau_2'^{(7)*} = 15\tau_1'^{(7)*}$

$* \quad \tau_3'^{(7)*} = 45\tau_1'^{(7)*}$

$* \quad \tau_4'^{(7)*} = 15\tau_1'^{(7)*}$

$* \quad \tau_5'^{(7)*} = 20\tau_1'^{(7)*}$

$* \quad \tau_6'^{(7)*} = 60\tau_1'^{(7)*}$

$* \quad \tau_7'^{(7)*} = 10\tau_1'^{(7)*}$

$* \quad \tau_8'^{(7)*} = 15\tau_1'^{(7)*}$

$* \quad \tau_9'^{(7)*} = 15\tau_1'^{(7)*}$

$$\tau_{10}'^{(7)*} = \tfrac{1}{6}\sum_{i=3}^{s^*} b_i'^* c_i Q_{i3}/\sigma^6 + 6\tau_1'^{(7)*}$$

$$\tau_{11}'^{(7)*} = \tfrac{1}{24}\sum_{i=3}^{s^*} b_i'^* Q_{i4}/\sigma^6 + \tau_1'^{(7)*}$$

$* \quad \tau_{12}'^{(7)*} = 15\tau_1'^{(7)*}$

$* \quad \tau_{13}'^{(7)*} = 15\tau_1'^{(7)*}$

$* \quad \tau_{14}'^{(7)*} = 3\tau_{10}'^{(7)*}$

$* \quad \tau_{15}'^{(7)*} = 6\tau_{11}'^{(7)*}$

$* \quad \tau_{16}'^{(7)*} = 3\tau_{11}'^{(7)*}$

$$\tau_{17}'^{(7)*} = Q_{2,1}\sum_{i=3}^{s^*} b_i'^* c_i a_{i2}/\sigma^6 + \tau_{10}'^{(7)*}$$

$$\tau_{18}'^{(7)*} = c_2 Q_{2,1}\sum_{i=3}^{s^*} b_i'^* a_{i2}/\sigma^6 + 4\tau_{11}'^{(7)*}$$

$$\tau_{19}'^{(7)*} = \tfrac{1}{2}Q_{2,2}\sum_{i=3}^{s^*} b_i'^* a_{i2}/\sigma^6 + \tau_{11}'^{(7)*}$$

$* \quad \tau_{20}'^{(7)*} = \tau_{19}'^{(7)*}$

Figure 6.1: 7th order error terms for the RKN6(4)6FD$_9^6$ continuous extension on y'.

ence purposes, reduction data files are also output that contain information useful for construction of the error norms. The files "*tds_FD69_err.rd*" and "*tds_FD69_err.tab*" will now contain data equivalent to Figure 6.3. The information contained in these files is a breakdown of the set of independent τ'^*'s in their standard ('*abc*') representation. For each truncation error term of the form

$$\alpha\Big(\frac{\gamma\Theta}{n!} - \frac{1}{i!}\Big),$$

$$\tau_1^{\prime(8)*} = \tfrac{1}{5040}\sum_{i=3}^{s^*} b_i^{\prime*} c_i^7/\sigma^7 - \frac{1}{8!}$$

$$*\quad \tau_2^{\prime(8)*} = 21\tau_1^{\prime(8)*}$$

$$*\quad \tau_3^{\prime(8)*} = 105\tau_1^{\prime(8)*}$$

$$*\quad \tau_4^{\prime(8)*} = 105\tau_1^{\prime(8)*}$$

$$*\quad \tau_5^{\prime(8)*} = 35\tau_1^{\prime(8)*}$$

$$*\quad \tau_6^{\prime(8)*} = 210\tau_1^{\prime(8)*}$$

$$*\quad \tau_7^{\prime(8)*} = 105\tau_1^{\prime(8)*}$$

$$*\quad \tau_8^{\prime(8)*} = 70\tau_1^{\prime(8)*}$$

$$*\quad \tau_9^{\prime(8)*} = 35\tau_1^{\prime(8)*}$$

$$*\quad \tau_{10}^{\prime(8)*} = 105\tau_1^{\prime(8)*}$$

$$*\quad \tau_{11}^{\prime(8)*} = 35\tau_1^{\prime(8)*}$$

$$\tau_{12}^{\prime(8)*} = \tfrac{1}{12}\sum_{i=3}^{s^*} b_i^{\prime*} c_i^2 Q_{i3}/\sigma^7 + 21\tau_1^{\prime(8)*}$$

$$*\quad \tau_{13}^{\prime(8)*} = \tau_{12}^{\prime(8)*}$$

$$\tau_{14}^{\prime(8)*} = \tfrac{1}{24}\sum_{i=3}^{s^*} b_i^{\prime*} c_i Q_{i4}/\sigma^7 + 7\tau_1^{\prime(8)*}$$

$$\tau_{15}^{\prime(8)*} = \tfrac{1}{120}\sum_{i=3}^{s^*} b_i^{\prime*} Q_{i5}/\sigma^7 + \tau_1^{\prime(8)*}$$

$$*\quad \tau_{16}^{\prime(8)*} = 35\tau_1^{\prime(8)*}$$

$$*\quad \tau_{17}^{\prime(8)*} = 105\tau_1^{\prime(8)*}$$

$$*\quad \tau_{18}^{\prime(8)*} = 35\tau_1^{\prime(8)*}$$

$$*\quad \tau_{19}^{\prime(8)*} = 3\tau_{12}^{\prime(8)*}$$

$$*\quad \tau_{20}^{\prime(8)*} = 3\tau_{12}^{\prime(8)*}$$

$$*\quad \tau_{21}^{\prime(8)*} = 6\tau_{14}^{\prime(8)*}$$

$$*\quad \tau_{22}^{\prime(8)*} = 10\tau_{15}^{\prime(8)*}$$

$$*\quad \tau_{23}^{\prime(8)*} = 3\tau_{14}^{\prime(8)*}$$

$$*\quad \tau_{24}^{\prime(8)*} = 15\tau_{15}^{\prime(8)*}$$

$$\tau_{25}^{\prime(8)*} = \tfrac{1}{2}Q_{2,1}\sum_{i=3}^{s^*} b_i^{\prime*} c_i^2 a_{i2}/\sigma^7 + \tau_{12}^{\prime(8)*}$$

$$*\quad \tau_{26}^{\prime(8)*} = \tau_{25}^{\prime(8)*}$$

$$\tau_{27}^{\prime(8)*} = c_2 Q_{2,1}\sum_{i=3}^{s^*} b_i^{\prime*} c_i a_{i2}/\sigma^7 + 4\tau_{14}^{\prime(8)*}$$

$$\tau_{28}^{\prime(8)*} = \tfrac{1}{2}c_2^2 Q_{2,1}\sum_{i=3}^{s^*} b_i^{\prime*} a_{i2}/\sigma^7 + 10\tau_{15}^{\prime(8)*}$$

$$*\quad \tau_{29}^{\prime(8)*} = \tau_{28}^{\prime(8)*}$$

$$\tau_{30}^{\prime(8)*} = \tfrac{1}{2}Q_{2,2}\sum_{i=3}^{s^*} b_i^{\prime*} c_i a_{i2}/\sigma^7 + \tau_{14}^{\prime(8)*}$$

$$\tau_{31}^{\prime(8)*} = \tfrac{1}{2}c_2 Q_{2,2}\sum_{i=3}^{s^*} b_i^{\prime*} a_{i2}/\sigma^7 + 5\tau_{15}^{\prime(8)*}$$

$$\tau_{32}^{\prime(8)*} = \tfrac{1}{6}\sum_{i=3}^{s^*} b_i^{\prime*}\sum_{j=2}^{i-1} a_{ij} Q_{j3}/\sigma^7 + \tau_{15}^{\prime(8)*}$$

$$*\quad \tau_{33}^{\prime(8)*} = \tau_{30}^{\prime(8)*}$$

$$*\quad \tau_{34}^{\prime(8)*} = \tau_{31}^{\prime(8)*}$$

$$*\quad \tau_{35}^{\prime(8)*} = 3\tau_{32}^{\prime(8)*}$$

$$\tau_{36}^{\prime(8)*} = Q_{2,1}\sum_{i=4}^{s^*} b_i^{\prime*}\sum_{j=3}^{i-1} a_{ij} a_{j2}/\sigma^7 + \tau_{32}^{\prime(8)*}$$

Figure 6.2: 8th order error terms for the RKN6(4)6FD$_9^6$ continuous extension on y'.

where i is an integer, the ".tab" file (represented on the right of Figure 6.3) gives the associated 'tree' in Scoins height form, α and γ quantities, plus the ratio $n!/\alpha\gamma$. Note

m	τ		Σ	Σ^2	∞
11	7	1	226	7476	60
2	7	10	4	10	3
3	7	11	10	46	6
1	7	17	1	1	1
1	7	18	1	1	1
2	7	19	2	2	1
14	8	1	1002	110692	210
4	8	12	8	20	3
3	8	14	10	46	6
3	8	15	26	326	15
2	8	25	2	2	1
1	8	27	1	1	1
2	8	28	2	2	1
2	8	30	2	2	1
2	8	31	2	2	1
2	8	32	4	10	3
1	8	36	1	1	1

τ		l	Tree	α	γ	$n!/\alpha\gamma$
7	1	7	0 1 1 1 1 1 1	1	7	720
7	10	6	0 1 2 2 2 1 0	6	140	6
7	11	6	0 1 2 2 2 2 0	1	210	24
7	17	5	0 1 2 3 1 0 0	6	840	1
7	18	5	0 1 2 3 2 0 0	4	1260	1
7	19	5	0 1 2 3 3 0 0	1	2520	2
8	1	8	0 1 1 1 1 1 1 1	1	8	5040
8	12	7	0 1 2 2 2 1 1 0	21	160	12
8	14	7	0 1 2 2 2 2 1 0	7	240	24
8	15	7	0 1 2 2 2 2 2 0	1	336	120
8	25	6	0 1 2 3 1 1 0 0	21	960	2
8	27	6	0 1 2 3 2 1 0 0	28	1440	1
8	28	6	0 1 2 3 2 2 0 0	10	2016	2
8	30	6	0 1 2 3 3 1 0 0	7	2880	2
8	31	6	0 1 2 3 3 2 0 0	5	4032	2
8	32	6	0 1 2 3 3 3 0 0	1	6720	6
8	36	5	0 1 2 3 4 0 0 0	1	40320	1

Figure 6.3: ".rd" and ".tab" error term data for the RKN6(4)6FD$_9^6$.

that in the Nyström case, the tree is given with all odd nodes removed — for this reason, in the RKN case *only*, a quantity l is given for the number of nodes in the tree as contained in the file (padded with zeros up to the maximum length for each order). The ".rd" file (on the left of Figure 6.3) gives quantities necessary for calculation of $||\cdot||_1$, $||\cdot||_2$ and $||\cdot||_\infty$ norms for each order. Column m gives the number of occurrences of the truncation error term, Σ gives the sum of the occurring multiples of the the τ, and column Σ^2 the sum of the squares of the multiples. Column ∞ gives the maximum occurring multiple. When calculating truncation error norms in an environment such as Fortran 90, given the ".tab" and ".rd" data along with an algorithm for calculating Θ, norms of local truncation error or linear deviation quantities can easily be worked out quickly by convenient vector and matrix operations.

Two models are now proposed, with the choice of free parameters (6.3) being as follows:

1. $\underline{\text{FD}_9^6\text{L}}$ *(linear deviation model)* Optimisation based on a principal linear deviation norm:

$$f(\boldsymbol{x}) = N_l^{(7)} \equiv \int_0^1 \|\boldsymbol{D}'^{(7)*}\|_2 \; d\sigma,$$

where $\boldsymbol{D}'^{(7)*}$ is the vector composed of elements $D_i'^{(7)*}, \; i = 1, \ldots, 20$, and

$$\boldsymbol{x} = \{c_7, c_8, c_9, a_{73}, a_{83}, a_{84}, a_{93}, a_{94}, a_{95}\}.$$

For this model, extra zeros have been put into $\overline{M}$ by imposing

$$\alpha_{i3} \equiv Q_{i3} = 0 \text{ (using } a_{i,i-3}), \text{ and } \alpha_{i4} \equiv Q_{21} a_{i2} = 0 \; (a_{i2} = 0), \; i = 7, \ldots, 9.$$

In this case, the b'^*'s contain only c_i's (and σ), and if small rational expressions can then be found for c_7, c_8 and c_9, the b'^*'s are likely to have a much simpler representation. A model was also optimised without this simplification, but was found not to merit losing the simple coefficient representation.

2. $\underline{\text{FD}_9^6\text{M}}$ *(minimised local error model)* Optimisation of a continuous principal local truncation error function:

$$f(\boldsymbol{x}) = N_m^{(7)} \equiv \int_0^1 \sigma^7 \|\boldsymbol{\tau}'^{(7)*}\|_2 \; d\sigma,$$

where $\boldsymbol{\tau}'^{(7)*}$ is the vector of $\tau_i'^{(7)*}, \; i = 1, \ldots, 20$, and $\boldsymbol{x}$ is the complete parameter list in (6.3). It should be noted that this model is only considered for comparison with the above case, and is hoped to give credence to the global error criterion of Chapter 5.

In both of the above cases, the objective function $f(\boldsymbol{x})$ was only the primary component of that optimised. It has been standard practice to include into $f(\boldsymbol{x})$ a component of the secondary quantities $N_l^{(8)}/N_m^{(8)}$, and norms of the 'a' parameter and 'b'^*'

polynomial coefficients to keep them 'small' (see Section 5.7.2). Final choices of the free parameters in the two cases are as follows:

1. $\underline{\text{FD}_9^6\text{L}}$

$$c_7 = \frac{3}{13},\ c_8 = \frac{1}{3},\ c_9 = \frac{6}{7},\ a_{73} = \frac{1}{80},$$

$$a_{83} = 0,\ a_{84} = -\frac{1}{260},\ a_{93} = 0,\ a_{94} = \frac{1}{16},\ a_{95} = 0.$$

2. $\underline{\text{FD}_9^6\text{M}}$

$$c_7 = \frac{1}{14},\ c_8 = \frac{7}{15},\ c_9 = \frac{1}{46},\ a_{72} = -\frac{1}{2399},\ a_{73} = \frac{1}{389},\ a_{74} = -\frac{1}{278},$$

$$a_{82} = \frac{1}{18},\ a_{83} = \frac{1}{85},\ a_{84} = \frac{1}{255},\ a_{85} = \frac{1}{312},$$

$$a_{92} = \frac{2}{55},\ a_{93} = -\frac{1}{54},\ a_{94} = \frac{1}{427},\ a_{95} = \frac{1}{6621},\ a_{96} = -\frac{1}{2399}.$$

6.3.2 Continuous output on y

An interesting point here is that sixth-order dense on y is possible with *no extra stages* ($s^* = 6$), the unique set of b^*'s for which is already presented in [15] and Example A.3.3. For this model, no optimisation procedure was relevant as all free parameters were chosen with respect to the discrete formula only. With $s^* = 9$, expression (5.34) could be used to trivially determine $b_i^*,\ i = 1, \ldots, 9$, to give a C^1 formula. However, there are 3 free b^*'s available to optimise the C^2 y^* formula here. It is conceivable in an application of the RKN6(4)6FD that dense output may only be required for y in any one step — in this situation, no extra evaluations need to be done after the step is accepted. If a continuous sixth-order result is required on both y and y', the three extra evaluations must be done and it is reasonable that they should *all* be used for the y formula also. After satisfying continuity requirements the free b^*'s must take the form:

$$b_i^* = \lambda_i(\sigma - 1)^3\sigma,\ i = 7, \ldots, 9,$$

where fourth-degree polynomials have been assumed adequate. Having made the natural condition $b_2^* = 0$, only quadrature equations remain, and

$$\sum_{i=1}^{9} b_i^* = \frac{1}{2}, \quad \sum_{i=3}^{9} b_i^* c_i^k = \frac{\sigma^k}{(k+2)(k+1)}, \quad k = 1, \ldots, 4,$$

can now be solved for b_1^* and b_i^*, $i = 3, \ldots, 6$, in terms of λ_7, λ_8 and λ_9. A GLOBS

	$N'^{(6)}$	$N_l'^{(7)}$	$N_l'^{(8)}$	$N_m'^{(7)}$	$N_m'^{(8)}$
FD_6^5	3.42×10^{-4}	3.85×10^{-4}	3.16×10^{-4}	3.55×10^{-5}	2.75×10^{-4}
$\mathrm{FD}_9^6\mathrm{L}$	0	1.52×10^{-5}	2.47×10^{-5}	6.29×10^{-5}	9.71×10^{-5}
$\mathrm{FD}_9^6\mathrm{M}$	0	4.45×10^{-5}	7.09×10^{-5}	1.96×10^{-5}	3.04×10^{-5}

Table 6.2: Integrated y'^* error norms for three RKN6(4)6FD triples.

	$N_l^{(7)}$	$N_l^{(8)}$	$N_m^{(7)}$	$N_m^{(8)}$
FD_6^5	7.48×10^{-5}	7.33×10^{-5}	7.93×10^{-5}	9.73×10^{-5}
$\mathrm{FD}_9^6\mathrm{L}$	1.71×10^{-5}	1.65×10^{-5}	5.25×10^{-5}	6.84×10^{-5}
$\mathrm{FD}_9^6\mathrm{M}$	2.16×10^{-5}	2.34×10^{-5}	4.13×10^{-5}	4.77×10^{-5}

Table 6.3: Integrated y^* error norms for three RKN6(4)6FD triples.

optimisation process has been performed on the principal truncation error terms of the y^* formula (generated in an identical way to the y'^* terms) using the same criterion in the two cases. Reasonable choices of the free λ parameters are as follows:

1. $\underline{\mathrm{FD}_9^6\mathrm{L}}$

$$\lambda_7 = -\frac{1}{5}, \; \lambda_8 = \frac{5}{2}, \; \lambda_9 = -1.$$

2. $\underline{\mathrm{FD}_9^6\mathrm{M}}$

$$\lambda_7 = -1, \; \lambda_8 = \frac{5}{7}, \; \lambda_9 = -3.$$

Principal/secondary linear deviation and local truncation error norms for the three RKN6(4)6FD dense formulae are presented in Tables 6.2 and 6.3 (calculated accurate

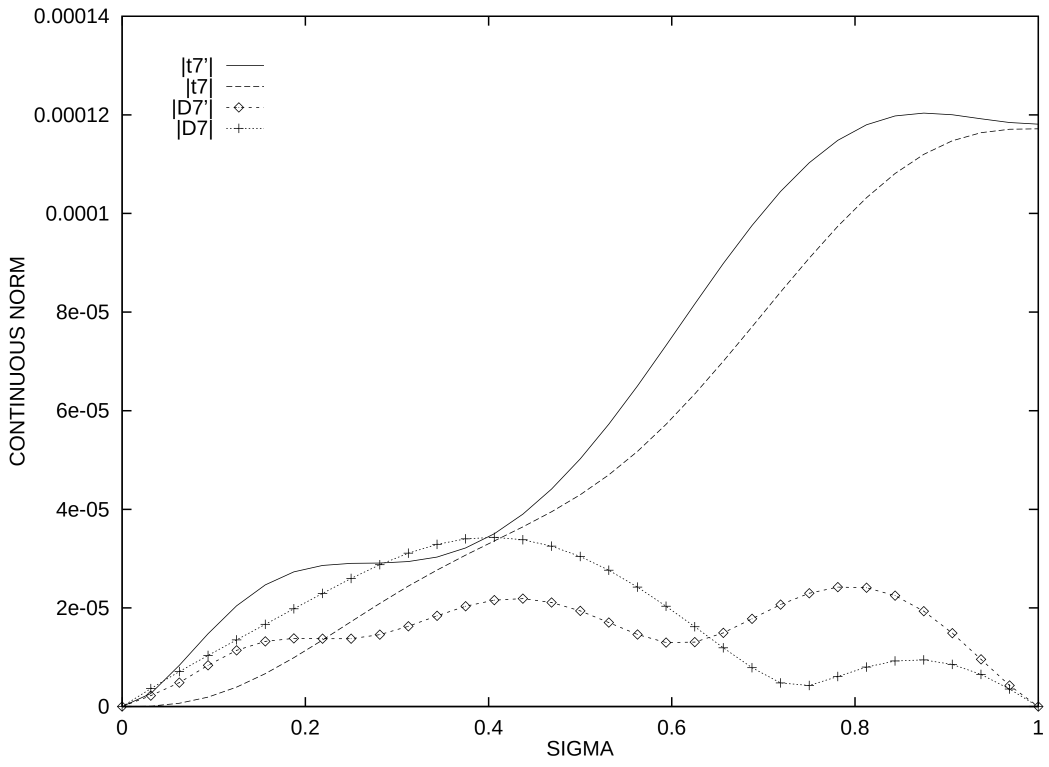


Figure 6.4: Principal truncation error and linear deviation norms for the FD_9^6L.

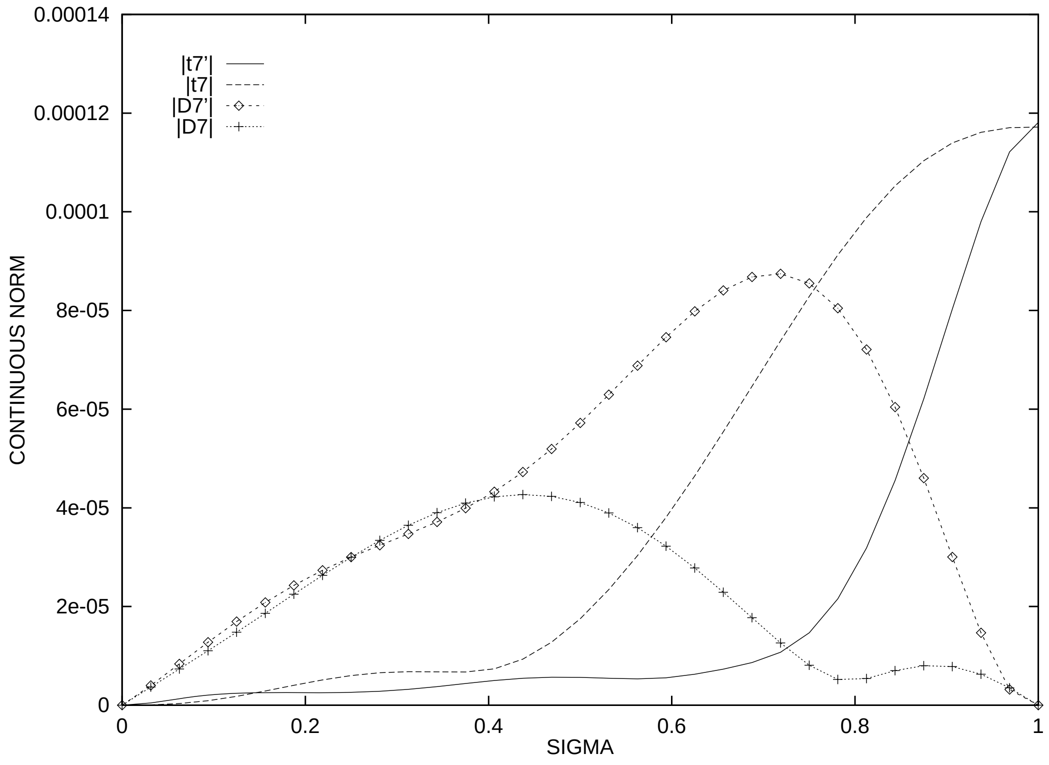


Figure 6.5: Principal truncation error and linear deviation norms for the FD_9^6M.

to two decimal places using a trapezium rule over 32 steps), where in the case of principal y' terms for the FD_6^5, the notation $N'^{(6)}$ is used for $N_l'^{(6)} \equiv N_m'^{(6)}$. Plots of $N_l'^{(7)}$, $N_m'^{(7)}$, $N_l^{(7)}$ and $N_m^{(7)}$ for the $\mathrm{FD}_9^6\mathrm{L}$ and $\mathrm{FD}_9^6\mathrm{M}$ are shown in Figures 6.4 and 6.5.

6.3.3 Numerical tests

For comparison of the two new sixth-order dense processes with the existing FD_6^5, six numerical tests will be used to collect continuous global error data. The test problems are numbered 10–13 in the RK-AID routines (see Section A.6.28), using default values, the orbit problem 10 being repeated with three different eccentricities — 0.1, 0.5 and 0.9 (corresponding to DETEST problems D1, D3 and D5, respectively). While it is not relevant to draw direct conclusions about the three formulae from this very limited test set, it is interesting to see whether the results fit predicted behaviour. Fundamentally, either of the two new $p^{y'^*} = 6$ formulae should yield better global error behaviour of their continuous y' solution, as compared with the FD_6^5. Also, the three extra stages used to give the sixth-order dense result on y may be hoped to benefit the new formulae, understanding that the y interpolant of the FD_6^5 could be used instead. The following RK-AID batch-file can be used to gather the required results:

```
batch, echo           ! Batch mode with ‘echo’
tol 1D-1              ! Starting tolerance
no tols 8             ! No. of tolerances
title                 ! Name output files by method
watts                 ! Step size control
dense output          ! Activate dense output
lump all problems     ! The six problem average
1 norm, box deviation ! SBD data average
!
test 646fd5 | go
test 646fd6m | go
test 646fd6l | go
```

which, when directed into the standard input of RK-AID, produces a set of output files for each problem (files ending “_#.eff”, for # an integer problem number), and

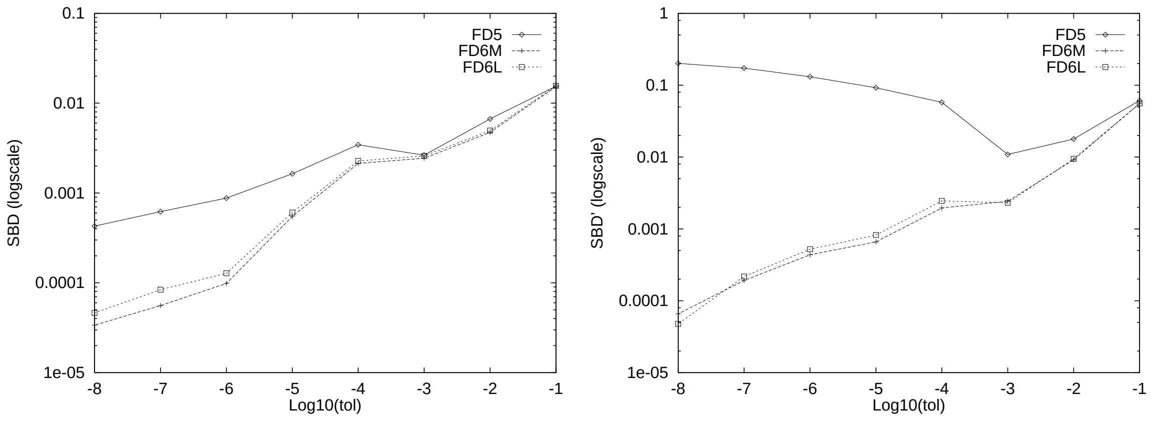


Figure 6.6: SBD/SBD′ on orbit D1 for three RKN6(4)6FD triples.

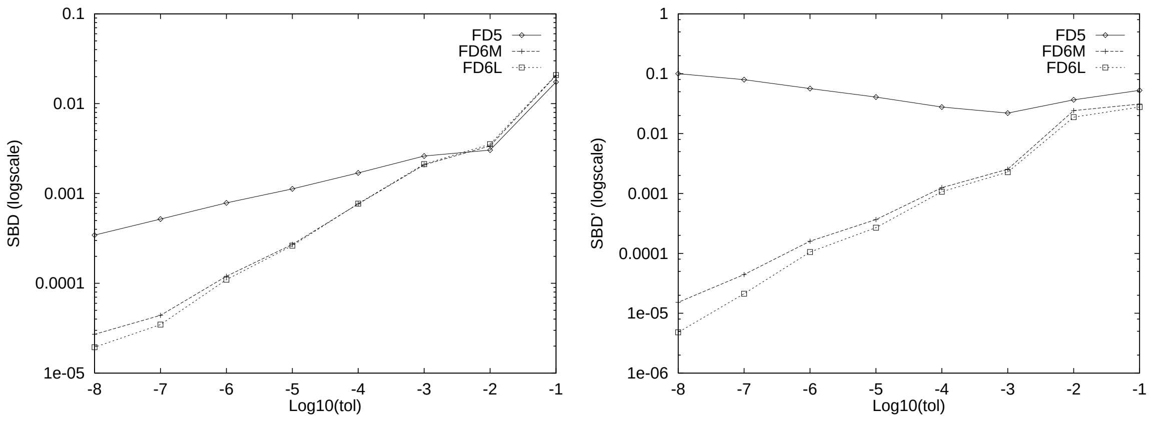


Figure 6.7: SBD/SBD′ on RK-AID problem 11 for three RKN6(4)6FD triples.

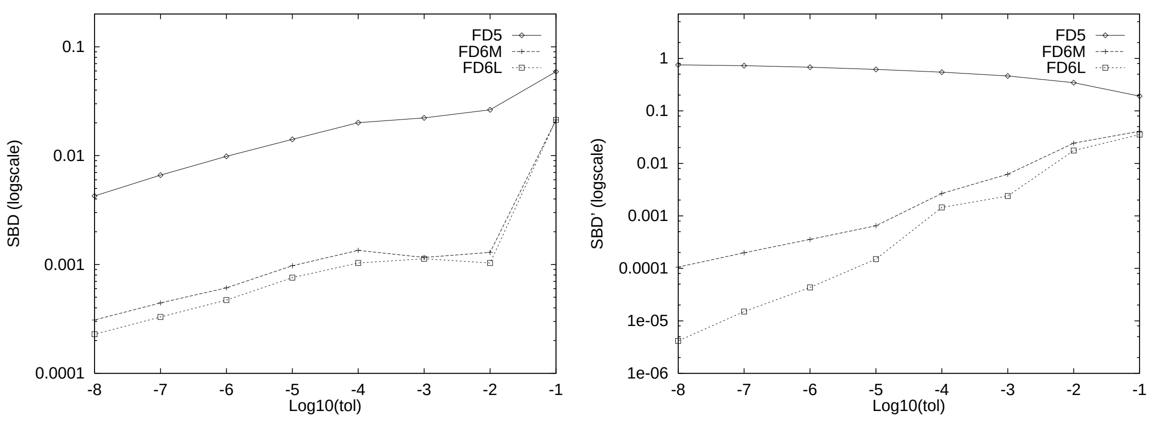


Figure 6.8: SBD/SBD′ on RK-AID problem 12 for three RKN6(4)6FD triples.

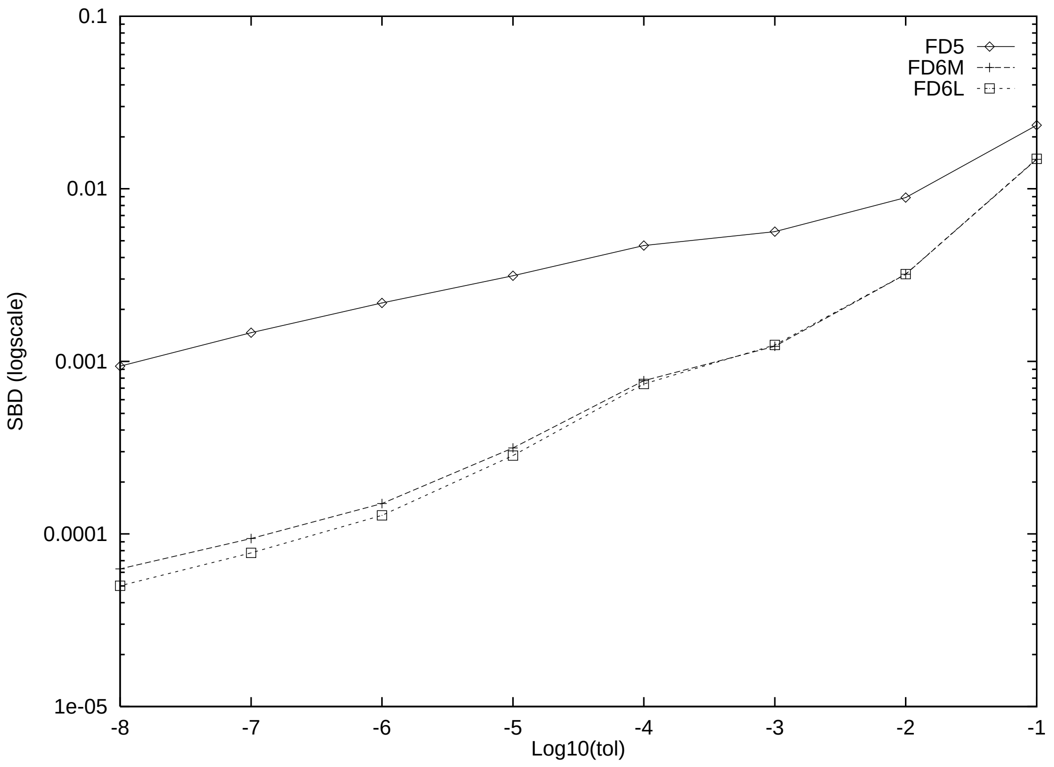


Figure 6.9: $\overline{\text{SBD}}$ over six problems for three RKN6(4)6FD triples.

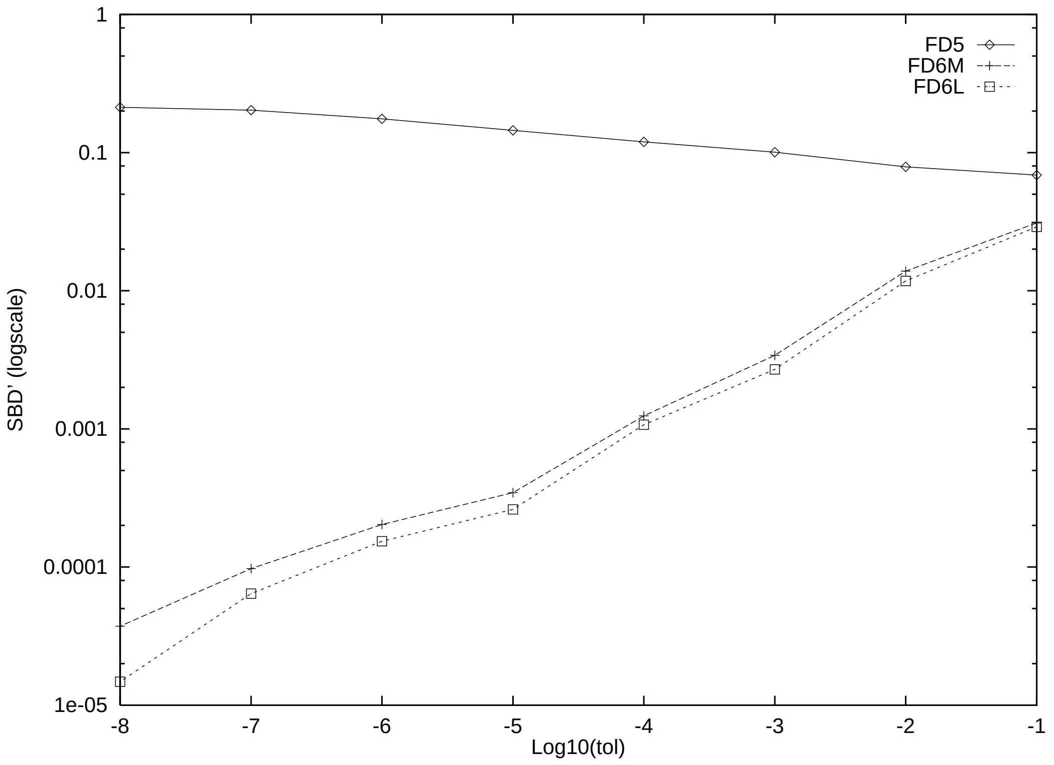


Figure 6.10: $\overline{\text{SBD}}'$ over six problems for three RKN6(4)6FD triples.

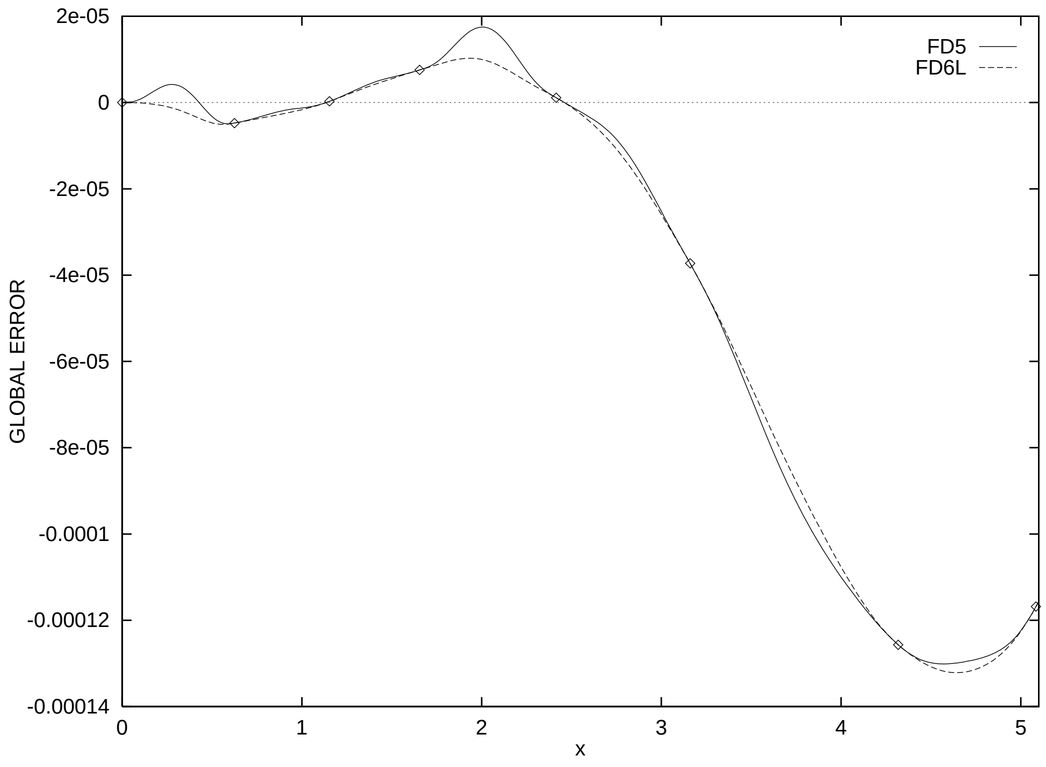


Figure 6.11: Global error on ${}^1y^*$ for a D1 orbit using two RKN6(4)6FD triples.

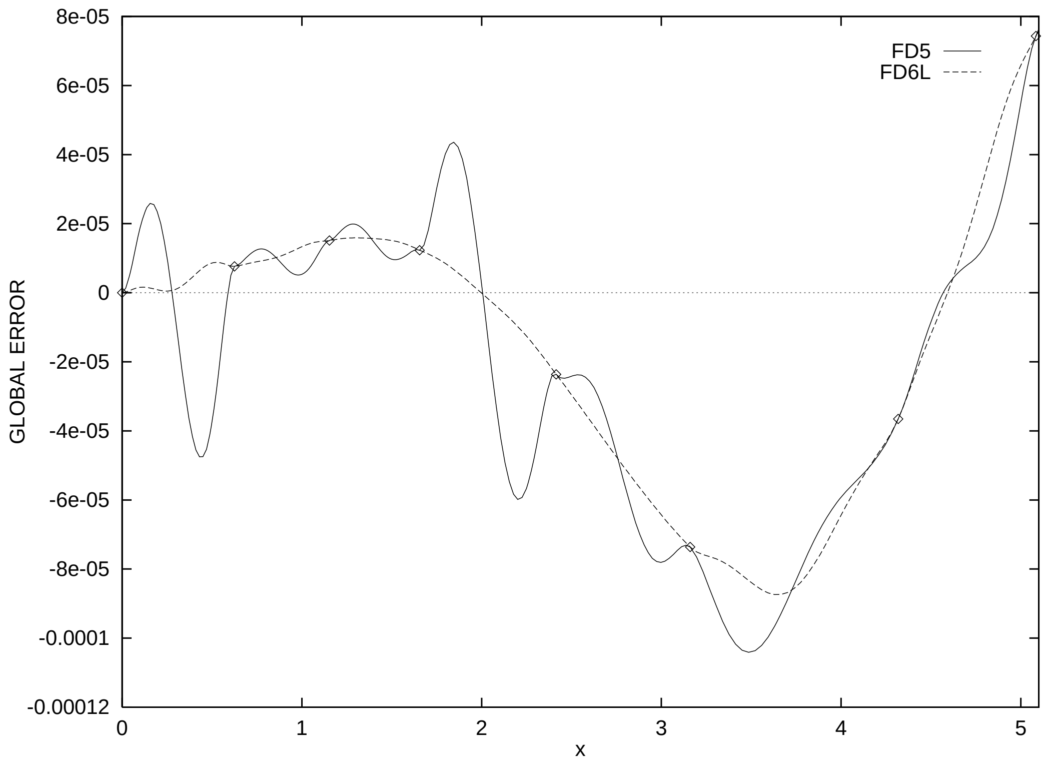


Figure 6.12: Global error on ${}^1y'^*$ for a D1 orbit using two RKN6(4)6FD triples.

for each TEST command. A representative set of logscale SBD/SBD$'$ against tolerance plots from the six test problems are given individually in Figures 6.6–6.8, with SBD$'$ data on the right in each case. Note that an *average* SBD/SBD$'$ ($\overline{\text{SBD}}/\overline{\text{SBD}}'$) data set for each method is also now available in a file ending "_a.eff", plotted in Figures 6.9 and 6.10. All results for y' show clear superiority of the two sixth-order interpolants over that of the FD$^5_6$, the order difference being qualitatively obvious from non-improvement in its SBD$'$ values for decreasing tolerances. On y, the extra 3 stages utilised for a sixth-order interpolant (over the minimum 6 of the FD$^5_6$) also give an advantage for all tolerances. Of the two new sixth-order methods, the FD$^6_9$L does yield smaller *average* box deviation quantities, the difference being slight but consistent. Specifically, the reverse was true for problem D1 on y', only, though the average quantities were affected largely in favour of the FD$^6_9$L from problem 13 (not plotted). It is therefore difficult to see any recommendation for either new method from y^* results alone, but in general the FD$^6_9$L can be adopted as *the* FD$^6_9$. Extra coefficients for the FD$^6_9$L are given exactly in Section B.1.1.

To provide a more graphic display of interpolant quality, continuous global errors ${}^1\varepsilon^*(x)$, ${}^1\varepsilon'^*(x)$ for the orbit problem D1 results are shown in Figures 6.11 and 6.12, respectively. A tolerance of $1{\times}10^{-3}$ was chosen for the FD$^5_6$ and FD$^6_9$L processes, and only the initial part of the integration has been shown. This plot illustrates the overall average behaviour, and it should be clear that the $p^{y'^*} = 6$ interpolation is excellent in every case. Note that discrete points are marked, but do not appear on the graph 'keys'.

6.4 Sixth-order dense for the RKN6(4)6FM

6.4.1 Continuous output on y'

With no simplifying conditions involved in derivation of the RKN6(4)6FM, $s^b = 2$, $q_i = 1$, $\forall i$, and the general a_{ij}, $j = 2, \ldots, i-1$, and non-quadrature $\boldsymbol{b}'^*$ systems may be written:

$$\overline{A}_i = \left(\begin{array}{cccccc|c} \frac{1}{10} & \frac{3}{10} & \frac{7}{10} & \frac{17}{25} & 1 & c_k & \frac{1}{6}c_i^3 + \alpha_{i1} \\ \frac{1}{100} & \frac{9}{100} & \frac{49}{100} & \frac{289}{625} & 1 & c_k^2 & \frac{1}{12}c_i^4 + \alpha_{i2} \\ \frac{1}{1000} & \frac{27}{1000} & \frac{343}{1000} & \frac{4913}{15625} & 1 & c_k^3 & \frac{1}{20}c_i^5 + \alpha_{i3} \\ -\frac{1}{6000} & \frac{1}{22000} & \frac{7}{66000} & \frac{17}{93750} & 0 & \alpha_{k1} & \alpha_{i4} \end{array}\right),$$

and

$$\overline{M} = \left(\begin{array}{cccccc|c} -\frac{1}{6000} & \frac{1}{22000} & \frac{7}{66000} & \frac{17}{93750} & 0 & \alpha_{k1} & 0 \\ -\frac{1}{60000} & \frac{3}{220000} & \frac{49}{660000} & \frac{289}{2343750} & 0 & c_k\alpha_{k1} & 0 \\ -\frac{1}{600000} & \frac{9}{2200000} & \frac{343}{6600000} & \frac{4913}{58593750} & 0 & c_k^2\alpha_{k1} & 0 \\ -\frac{1}{120000} & -\frac{97}{440000} & -\frac{2051}{1320000} & -\frac{1003}{390625} & 0 & \alpha_{k2} & 0 \\ -\frac{1}{1200000} & -\frac{291}{4400000} & -\frac{14357}{13200000} & -\frac{17051}{9765625} & 0 & c_k\alpha_{k2} & 0 \\ -\frac{1}{2000000} & -\frac{1673}{22000000} & -\frac{60277}{22000000} & -\frac{636599}{146484375} & 0 & \alpha_{k3} & 0 \\ 0 & -\frac{1}{132000} & \frac{49}{2420000} & \frac{21199}{773437500} & 0 & \alpha_{k4} & 0 \end{array}\right).$$

Again, it can be seen that the $\overline{A}_i$ systems are under-determined, and so no further parameterisation is necessary. The numerical part of M has rank 3 (i.e., $r_b = 3$), therefore eliminating with three pivoting operations (partitioning as in Chapter 5):

$$\overline{M} = \left(\begin{array}{c|c|c} T_b & \boldsymbol{\zeta}_k^b & 0 \\ \hline 0 & \boldsymbol{\zeta}_k^\beta & 0 \end{array}\right) =$$

$$\left(\begin{array}{ccccc|c|c} -\frac{1}{6000} & \frac{1}{22000} & \frac{7}{66000} & \frac{17}{93750} & 0 & \alpha_{k1} & 0 \\ 0 & \frac{1}{110000} & \frac{7}{110000} & \frac{493}{4687500} & 0 & \frac{\alpha_{k1}(10\,c_k-1)}{10} & 0 \\ 0 & 0 & \frac{7}{275000} & \frac{9367}{234375000} & 0 & \frac{\alpha_{k1}\left(100\,c_k^2-40\,c_k+3\right)}{100} & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & \frac{\alpha_{k1}49(c_k-5)+2\,\alpha_{k2}}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{c_k(\alpha_{k1}49(c_k-5)+2\,\alpha_{k2})}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\alpha_{k1}\left(4335\,c_k^2-1315\,c_k+88\right)+50\,\alpha_{k3}}{50} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{\alpha_{k1}\left(950\,c_k^2-655\,c_k+56\right)-330\,\alpha_{k4}}{330} & 0 \end{array}\right).$$

With reference to Section 5.4.1, the 'α_i' may now be removed from the system by forcing

$$\boldsymbol{\zeta}_i^\beta = \boldsymbol{\beta}_i = (\beta_{i1}, \beta_{i2}, \beta_{i3}, \beta_{i4})^T,$$

requiring solution of $\Upsilon_i^\beta \boldsymbol{\alpha}_i = \boldsymbol{\beta}_i$, where

$$\Upsilon_i^\beta = \begin{pmatrix} \frac{49\,c_i - 5}{2} & 1 & 0 & 0 \\ \frac{c_i (49\,c_i - 5)}{2} & c_i & 0 & 0 \\ \frac{4335\,c_i^2 - 1315\,c_i + 88}{50} & 0 & 1 & 0 \\ -\frac{950\,c_i^2 - 655\,c_i + 56}{330} & 0 & 0 & 1 \end{pmatrix}.$$

This yields

$$\alpha_{i1} = \frac{330\,\alpha_{i4} - 330\,\beta_{i4}}{950\,c_i^2 - 655\,c_i + 56},\ \ \alpha_{i3} = -\frac{A}{4750\,c_i^2 - 3275\,c_i + 280},$$

$$\alpha_{i2} = -\frac{\alpha_{i4}\,(8085\,c_i - 825) - 950\,\beta_{i1}\,c_i^2 + (655\,\beta_{i1} - 8085\,\beta_{i4})\,c_i + 825\,\beta_{i4} - 56\,\beta_{i1}}{950\,c_i^2 - 655\,c_i + 56},$$

where

$$\begin{aligned} A \ = \ & \alpha_{i4}\left(143055\,c_i^2 - 43395\,c_i + 2904\right) + (-143055\,\beta_{i4} - 4750\,\beta_{i2})\,c_i^2 + \\ & \qquad (43395\,\beta_{i4} + 3275\,\beta_{i2})\,c_i - 2904\,\beta_{i4} - 280\,\beta_{i2}, \end{aligned}$$

and $\beta_{i3} = c_i \beta_{i2}$ for consistency. A minimal-stage solution to the problem is now guaranteed if $\beta_{ik} = 0,\ k = 1, \ldots, 4$, by setting $\beta_{i1} = \beta_{i2} = \beta_{i4} = 0$. In this case,

$$\alpha_{i1} = \frac{330\,\alpha_{i4}}{(10\,c_i - 1)\,(95\,c_i - 56)},\ \ \alpha_{i2} = -\frac{165\,\alpha_{i4}\,(49\,c_i - 5)}{(10\,c_i - 1)\,(95\,c_i - 56)},$$

$$\alpha_{i3} = -\frac{33\,\alpha_{i4}\,(4335\,c_i^2 - 1315\,c_i + 88)}{5\,(10\,c_i - 1)\,(95\,c_i - 56)},$$

and

$$\boldsymbol{\zeta}_i^b = \left(\frac{330\,\alpha_{i4}}{(10\,c_i - 1)\,(95\,c_i - 56)}, \frac{33\,\alpha_{i4}}{95\,c_i - 56}, \frac{33\,\alpha_{i4}\,(10\,c_i - 3)}{10\,(95\,c_i - 56)}\right)^T.$$

Solution of $\overline{M}$ now requires three b'^*'s, plus six for the quadrature equations, yielding a solution with $s^* = 9$. Weights $b_i'^*$, $i = 1, \ldots, 9$, are thus specified in terms of c_i, α_{i4}, $i = 7, \ldots, 9$, and σ (as fifth-degree polynomials). $\overline{A}_i$ will determine a_{ij}, $j = 2, 3, 4$, and a_{i6} in terms of a_{i5}, a_{ij}, $j = 7, \ldots, i-1$, α_{i4} and c_i. The 12 parameters for optimisation purposes are therefore

$$c_7, c_8, c_9, a_{75}, a_{85}, a_{87}, a_{95}, a_{97}, a_{98}, \alpha_{74}, \alpha_{84}, \text{ and } \alpha_{94}. \tag{6.4}$$

Choice of free parameters

The relevant truncation error coefficients will not be given in detail for this case, being essentially a general set of seventh- and eighth-order terms. Two optimisation procedures have been performed in an identical manner with those for the RKN6(4)6FD to generate FMD$_9^6$L and FMD$_9^6$M models. In both these cases, simple rational coefficients were sought by experimenting with the conditions $\alpha_{74} = \alpha_{84} = \alpha_{94} = 0$ in order to simplify the $\boldsymbol{b}'^*$ system. This was found not to compromise the results for either formula, and the 9 remaining parameters in the two cases have been chosen as follows:

1. FMD$_9^6$L

$$c_7 = \frac{9}{31},\ c_8 = \frac{14}{17},\ c_9 = \frac{8}{41},\ a_{75} = \frac{1}{24},\ a_{85} = -\frac{1}{21},\ a_{87} = \frac{3}{19},$$

$$a_{95} = -\frac{1}{20},\ a_{97} = -\frac{5}{36},\ a_{98} = -\frac{1}{23}.$$

2. FMD$_9^6$M

$$c_7 = \frac{1}{10},\ c_8 = \frac{4}{7},\ c_9 = \frac{1}{3},\ a_{75} = \frac{1}{61},\ a_{85} = -\frac{1}{27},\ a_{87} = \frac{1}{15},$$

$$a_{95} = -\frac{1}{34},\ a_{97} = \frac{1}{65},\ a_{98} = -\frac{1}{58}.$$

An additional consideration for this case over the RKN6(4)6FD work of the previous section is that fifth-order dense on y' requires a seventh stage. For convenience when applying the new methods, it may be sought that this formula is common to the 9-stage models. The FMD$_7^5$ of [15] does not, however, allow sixth-order dense on y' with $s^* = 9$ using its seventh stage. The FMD$_9^6$L and FMD$_9^6$M models above were monitored during optimisation for the possibility of containing 'good' fifth-order formulae in their first seven stages. These formula will be denoted as FMD$_7^5$L and FMD$_7^5$M, and the complete embedded triples (i.e., containing the *two* dense formulae) as FMD$_{9(7)}^{6(5)}$L and FMD$_{9(7)}^{6(5)}$M.

	$N'^{(6)}$	$N_l'^{(7)}$	$N_l'^{(8)}$	$N_m'^{(7)}$	$N_m'^{(8)}$
FMD$_7^5$	2.19×10^{-4}	2.68×10^{-4}	2.38×10^{-4}	2.55×10^{-4}	2.11×10^{-4}
FMD$_7^5$L	1.80×10^{-4}	2.25×10^{-4}	2.11×10^{-4}	2.12×10^{-4}	1.81×10^{-4}
FMD$_7^5$M	9.38×10^{-5}	1.65×10^{-4}	1.76×10^{-4}	1.49×10^{-4}	1.41×10^{-4}
FMD$_9^6$L	0	6.32×10^{-6}	1.65×10^{-5}	3.98×10^{-5}	6.52×10^{-5}
FMD$_9^6$M	0	2.99×10^{-5}	4.84×10^{-5}	1.27×10^{-5}	1.89×10^{-5}

Table 6.4: Integrated y'^* error norms for five RKN6(4)6FM triples.

	$N_l^{(7)}$	$N_l^{(8)}$	$N_m^{(7)}$	$N_m^{(8)}$
FMD$_7^5$	4.68×10^{-5}	5.24×10^{-5}	5.97×10^{-5}	6.65×10^{-5}
FMD$_7^5$L	3.70×10^{-5}	4.20×10^{-5}	5.31×10^{-5}	5.99×10^{-5}
FMD$_7^5$M	1.93×10^{-5}	3.08×10^{-5}	2.68×10^{-5}	4.15×10^{-5}
FMD$_9^6$L	1.23×10^{-5}	1.74×10^{-5}	3.83×10^{-5}	5.86×10^{-5}
FMD$_9^6$M	2.18×10^{-5}	3.44×10^{-5}	2.58×10^{-5}	4.07×10^{-5}

Table 6.5: Integrated y^* error norms for five RKN6(4)6FM triples.

6.4.2 Continuous output on y

The two extra stages available for $p^{y^*} = 6$ interpolants here may now be employed to improve on the principal and secondary local truncation error/linear deviation norms

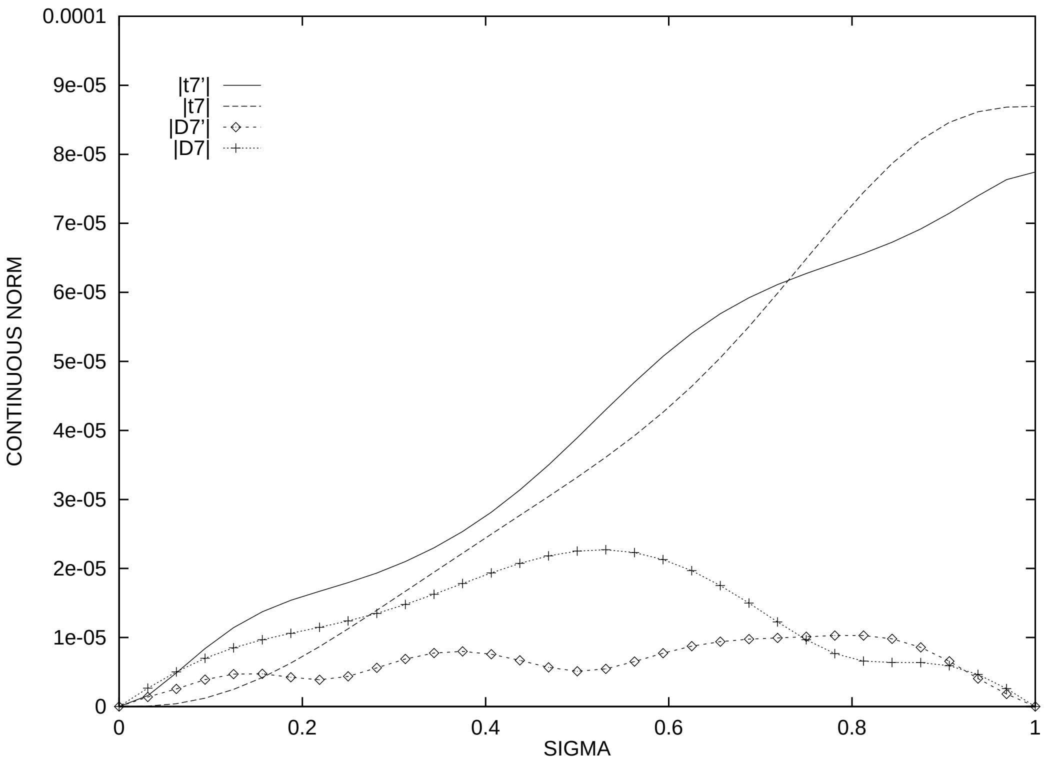


Figure 6.13: Principal truncation error and linear deviation norms for the FMD_9^6L.

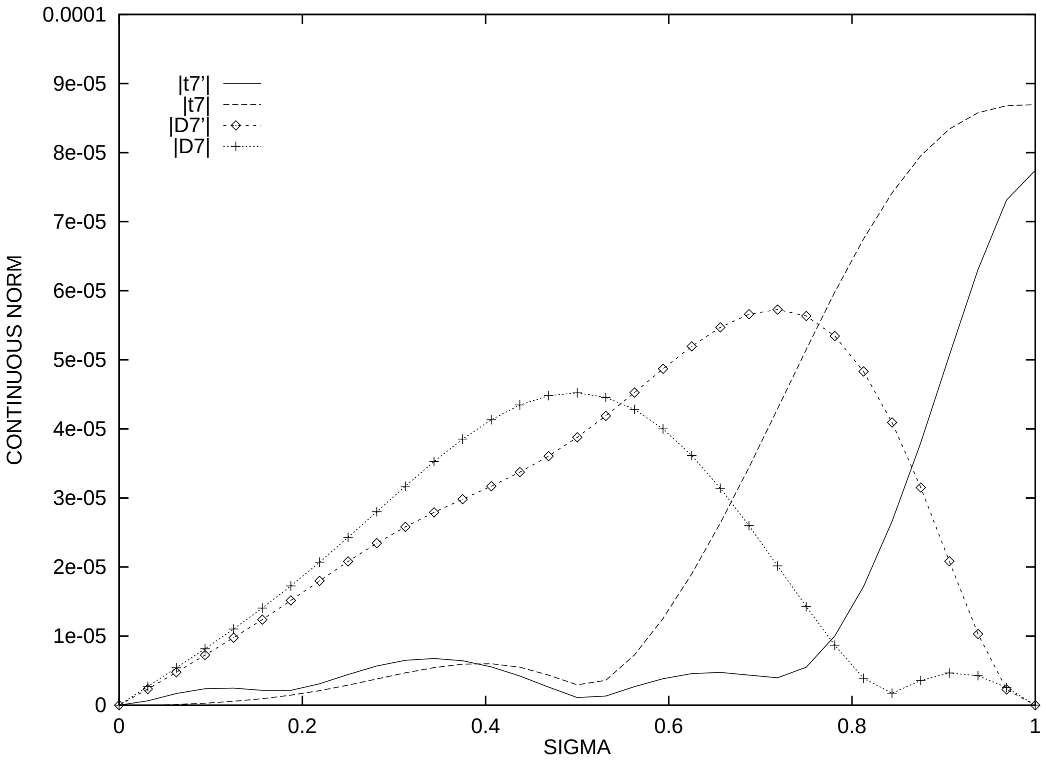


Figure 6.14: Principal truncation error and linear deviation norms for the FMD_9^6M.

as before. The two free b^*'s must be constrained for C^2 continuity, and have again been chosen as fourth-degree:

$$b_i^* = \lambda_i(\sigma - 1)^3\sigma,\ \ i = 8, 9.$$

An optimisation was performed on λ_8 and λ_9, giving the following choices:

1. $\underline{\text{FMD}_{9(7)}^{6(5)}\text{L}}$

$$\lambda_8 = -1,\ \ \lambda_9 = -\frac{5}{2}.$$

2. $\underline{\text{FMD}_{9(7)}^{6(5)}\text{M}}$

$$\lambda_8 = -\frac{1}{87},\ \ \lambda_9 = -2.$$

Principal/secondary linear deviation and local truncation error norms for the two sixth-order, and three fifth-order RKN6(4)6FM dense formulae are presented in Tables 6.4 and 6.5 (again accurate to two decimal places), and plots of $N_l^{\prime(7)}$, $N_m^{\prime(7)}$, $N_l^{(7)}$ and $N_m^{(7)}$ for the FMD$_9^6$L and FMD$_9^6$M are shown in Figures 6.13 and 6.14.

6.4.3 Numerical tests

In an identical use of the six RK-AID Nyström test problems to the previous section, a set of SBD statistics has been collected for all five methods. As before, results for three of these problems has been graphed using a logscale against tolerance plot in Figures 6.15–6.17, and an *average* over all six problems is given in Figures 6.18 and 6.19, with SBD$'$ data on the right in every case. With only *two* extra stages needed to increase from $p^* = q-1$ to $p^* = q$, it may have been expected that improvement in dense output quality would not be as marked as in the RKN6(4)6FD case, especially for continuous y output. This is certainly suggested from the results presented. However, a *significant* improvement in the $p^* = 5$ formula has been indicated by the new embedded FMD$_7^5$L and FMD$_7^5$M cases over the existing FMD$_7^5$ (which was 'partially' optimised on a purely local error criterion). Note that the FMD$_7^5$M is statistically a better 'linear deviation' formula than the FMD$_7^5$L (from Table 6.5), and inferiority

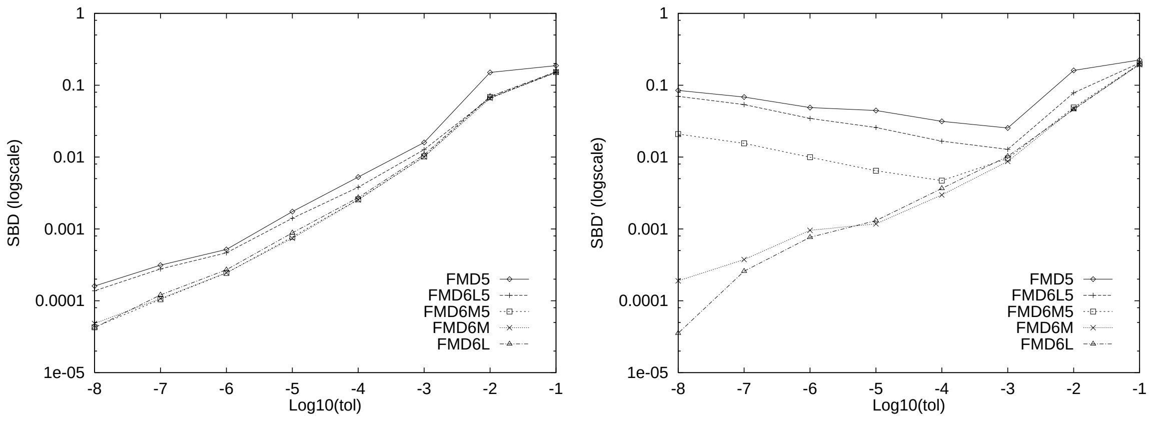


Figure 6.15: SBD/SBD′ on orbit D1 for five RKN6(4)6FM triples.

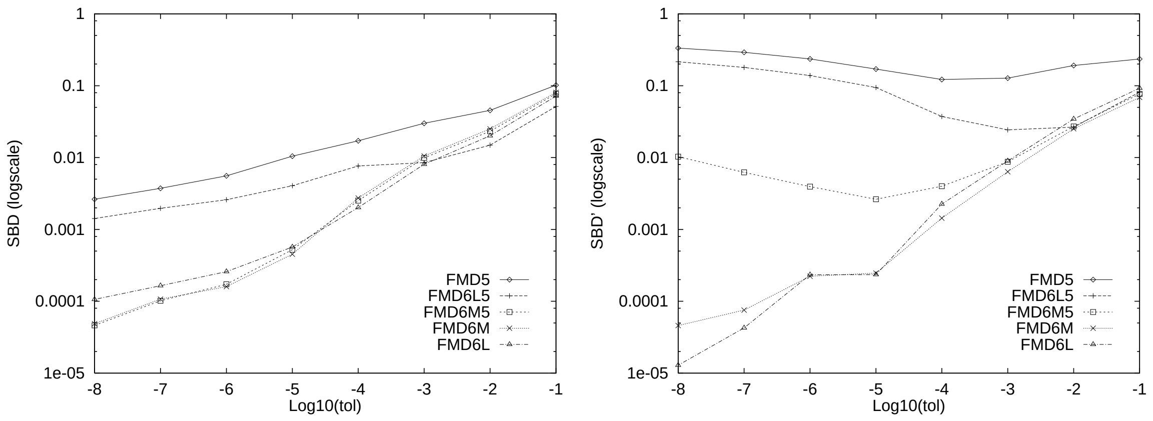


Figure 6.16: SBD/SBD′ on RK-AID problem 11 for five RKN6(4)6FM triples.

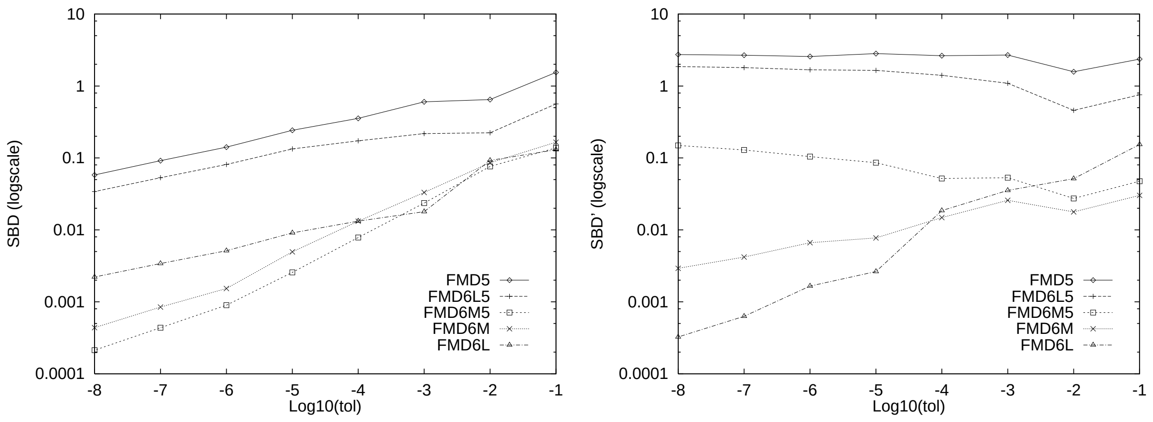


Figure 6.17: SBD/SBD′ on RK-AID problem 12 for five RKN6(4)6FM triples.

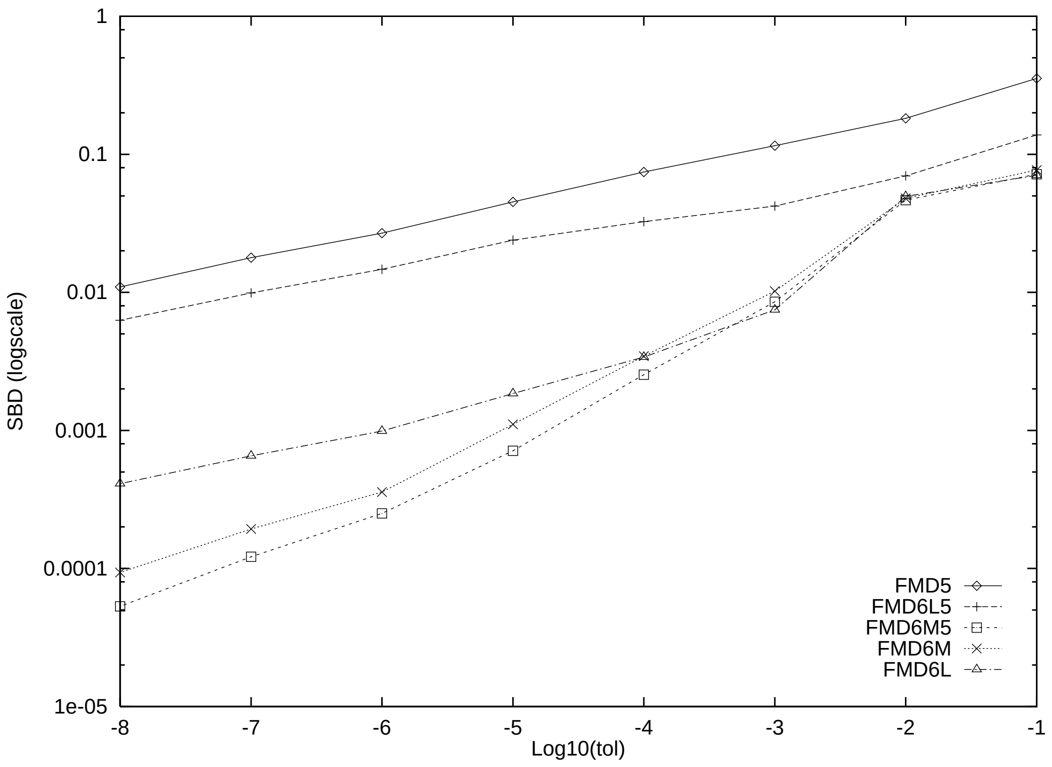


Figure 6.18: $\overline{\mathrm{SBD}}$ over six problems for five RKN6(4)6FM triples.

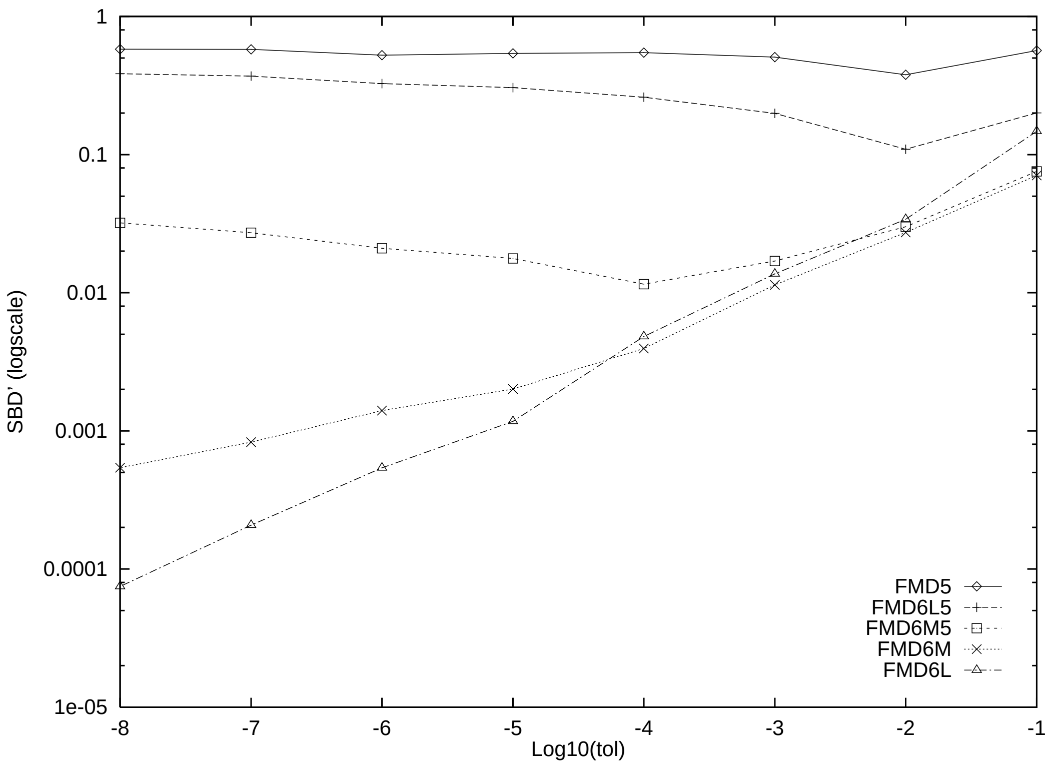


Figure 6.19: $\overline{\mathrm{SBD}}'$ over six problems for five RKN6(4)6FM triples.

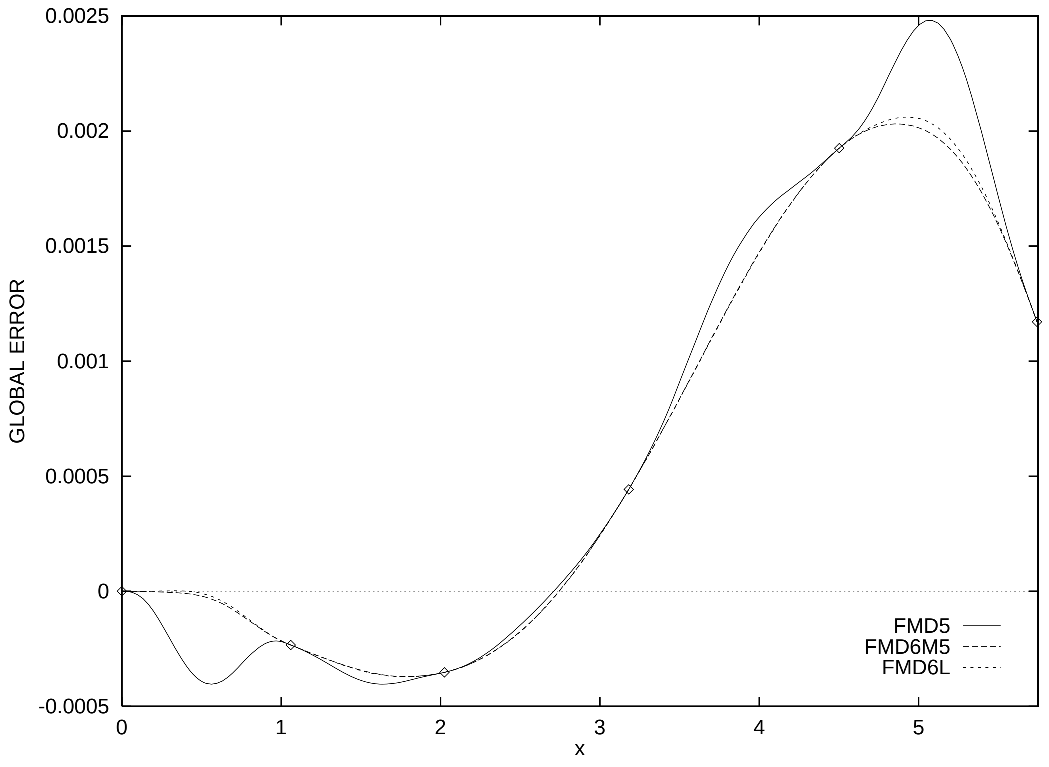


Figure 6.20: Global error on ${}^{1}y^{*}$ for a D1 orbit using three RKN6(4)6FM triples.

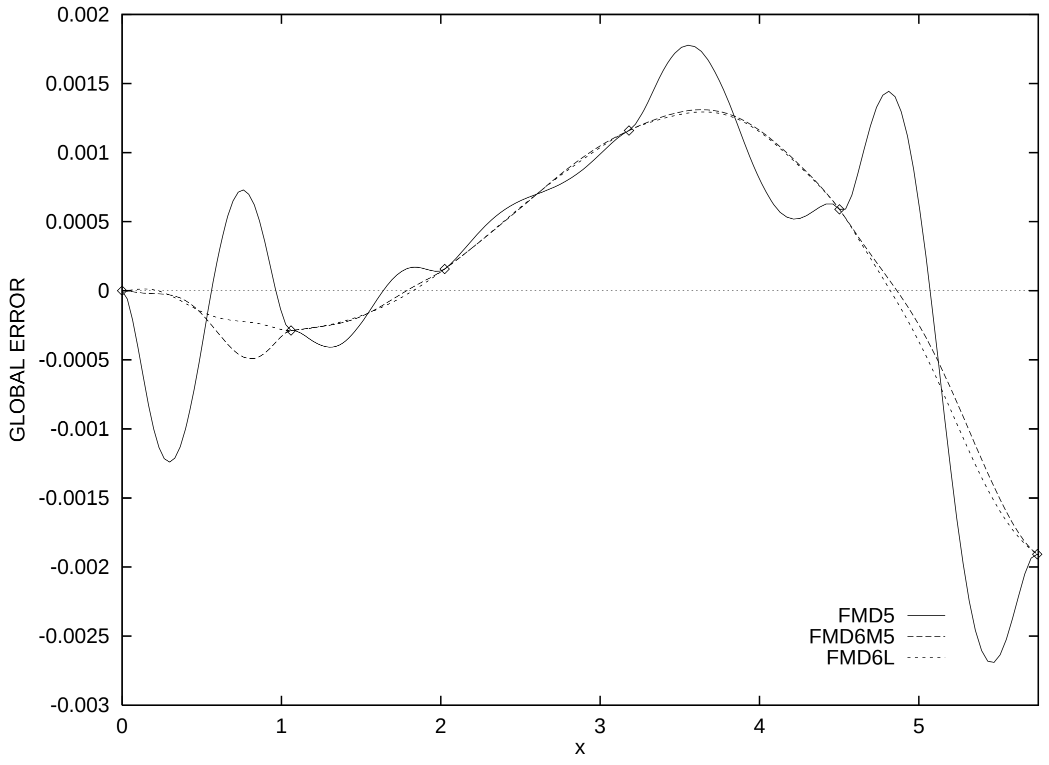


Figure 6.21: Global error on ${}^{1}y'^{*}$ for a D1 orbit using three RKN6(4)6FM triples.

of the FMD$^5_7$L can clearly be seen in all cases. Now, with only a marginal difference between $N_l^{(7)}$ quantities for the FMD$^6_9$M, FMD$^6_9$L and FMD$^5_7$M, experimental differences on y are unpredictable. For example, the seven-stage formula performs superbly and advantages of $s^* = 9$ seem negligible if only dense output on y were required. Also, the FMD$^6_9$L does not show its statistical advantage until y'^* output is considered, where for the more stringent tolerances superior global error behaviour is seen. The reason for such unpredictable behaviour on the y^* output may be inferred as being due to the power of a small number of free parameters for optimisation of the $p^* = 5$, seven-stage methods. Advantages gained from the two extra stages for $p^* = 6$ are then small, compared with the RKN6(4)6FD case, where the $p^* = 5$ case was unique.

It is unfortunate that the sixth-order linear deviation formula does not contain the best embedded fifth-order method, but plenty of scope is possible between the FMD$^{6(5)}_{9(7)}$L and FMD$^{6(5)}_{9(7)}$M models. While it is difficult to exclusively recommend the FMD$^6_9$L from these results, the theoretical basis formed by the global error criterion is still to be preferred. Extra coefficients for the FMD$^6_9$L and FMD$^5_7$M are given in Sections B.1.2 and B.1.2. Again, for illustration purposes a continuous global error plot from the D1 orbit problem results (with tolerance $1{\times}10^{-3}$) is shown in Figures 6.20 and 6.21 for the FMD$^5_7$, FMD$^5_7$M and FMD$^6_9$L.

6.5 New and extrapolated triples

The simplicity of the sixth-order Nyström case enables experimentation regarding further possibilities for efficient triples. In addition to derivation of alternative processes, the suggestion in Chapter 5 of extrapolating a continuous order $q+1$ result on y to propagate the discrete solution for *any* Nyström triple with $p^{y'^*} = q$ will be considered for the new formulae.

6.5.1 New sixth-order Nyström triples

The purpose of this section is to investigate the possibility of $s^* = 8$, $p^* = 6$ triples. With reference to the RKN6(4)6FD model employing $s^b = q_i = 3$ simplifying conditions, it would initially be expected that an eight-stage model for sixth-order dense might be available. Considering the $\overline{M}$ ($\boldsymbol{b}'^*$) system, it is conceivable that a discrete model might be possible such that the rank of the numerical part of the its two rows is 1 (not 2 in the RKN6(4)6FD case). The RKN6(4)6FD was chosen from a one-parameter family, its derivation being essentially straightforward barring the solution of $\tau_{10}'^{(6)}$ to zero. In Dormand and Prince [15], c_4 was used to solve

$$^{\dagger}\tau_{10}'^{(6)} = \sum_{i=3}^{5} b_i' a_{i2} = 0, \tag{6.5}$$

leaving c_5 as the only free parameter. If, however, (6.5) is solved by linear dependence on the equation

$$^{\dagger}\tau_{7}'^{(6)} = \sum_{i=3}^{5} b_i' Q_{i3} = 0, \tag{6.6}$$

solution of the $\overline{M}$ system for sixth-order dense will need only eight stages, as required. This is achieved by forcing

$$a_{42} = \lambda Q_{43}, \; a_{52} = \lambda Q_{53}, \tag{6.7}$$

for $\lambda = a_{32}/Q_{33}$, using c_4 and c_5, giving a unique higher-order formula (i.e., with no free parameters). Details of the derivation will not be given as conditions (6.7) require solution of a sixth-degree polynomial with only a single applicable real solution, which is non-radical, and can therefore only be approximated. This discrete model has been provided with a fourth-order embedding (with b_4, b_5, b_6 and b_6' as 'tuning' parameters), and shall be denoted as the RKN6(4)6FU ('U' for *unique*). In an identical manner to derivation of the FD$_9^6$L and FMD$_9^6$L linear deviation criterion formulae, FUD$_6^5$L and FUD$_8^6$L models have been obtained and optimised (no local error minisation formula have been derived for comparison purposes), the coefficients for which are given in

Section B.1.3.

An alternative sixth-order triple has also been derived with $s^* = 8$, but using all 8 stages for the discrete formula (eighth stage is FSAL). This method uses a simplifying scheme with $s^b = q_i = 4,\ i > 3$, and is denoted by RKN6(5)8F8 — an $\mathrm{RKN}_{8f}^{6(5)}\mathrm{D}_8^6$ triple (note the fifth-order embedding), given in Section B.1.4. The optimisation for

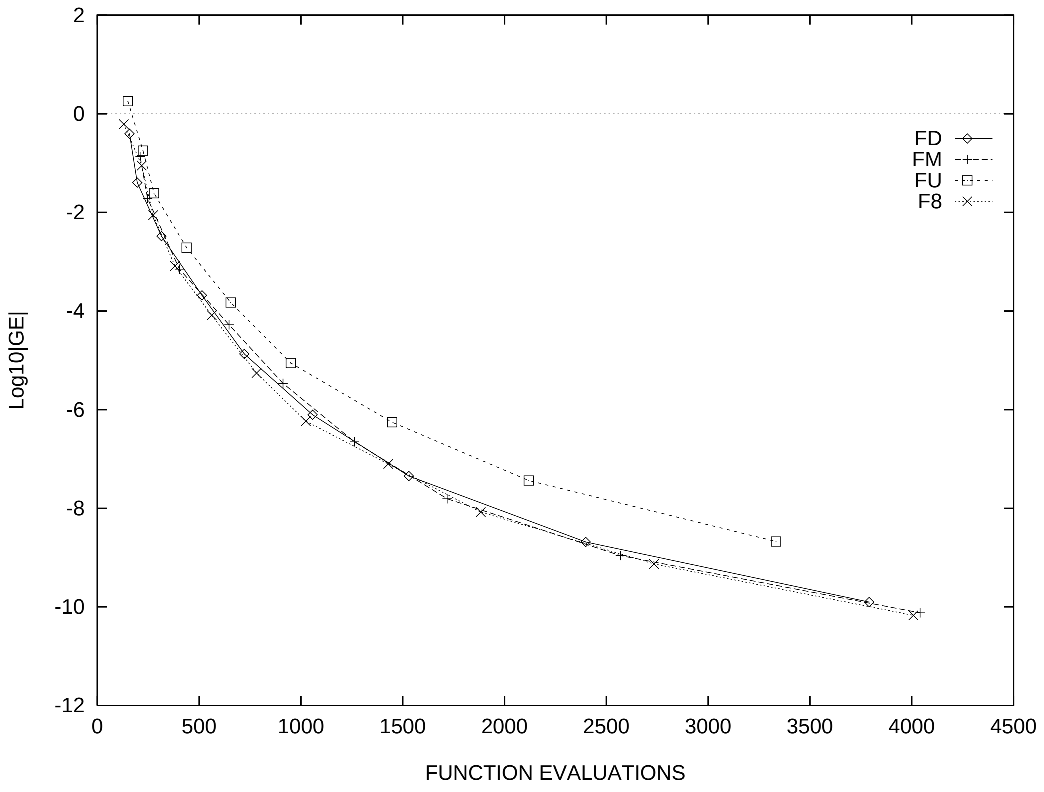


Figure 6.22: Average efficiency of four sixth-order RKN methods on six test problems.

the dense formula in this case was not done using any definite criterion because the discrete process was not fixed, i.e., optimisation was performed on the continuous process with the $\sigma = 1$ case as the discrete formula for both y and y'. Numerical comparison of the four discrete models has been done using the following RK-AID batch-file (noting that no dense output is done by these tests):

```
batch, echo
lump all problems
title
!
tol 1D-1 | no. tols 11 | test 658f8 | go
tol 1D-2 | no. tols 9 | test 646fm | go
!
tol 1D-1 | no. tols 9
test 646fd | go
test 646fu | go
```

which uses six problems (including three orbits with different eccentricities) to form mean maximum global error against mean number of function evaluation results for each of the TEST commands. The combined "_a.eff" plot for all four methods is given in Figure 6.22. While this shows the RKN6(4)6FU to be clearly less efficient than the two published pairs, the RKN6(5)8F8 performs equally well, and it may be hoped that advantages gained by lack of a ninth stage for sixth-order dense output purposes may be significant for either method.

It is not clear how continuous processes with different discrete formulae can be compared fairly. The scaled box deviation (SBD) scheme, being scaled by an average of discrete global errors, is one possible quantity. As such, it is interesting to plot SBD data against the number of function evaluations used to achieve some form of dense output efficiency measure for a collection of the methods contained in this chapter, considering the y' interpolants only. A D1 orbit problem, being fairly representative, was chosen to gather this information over a range of tolerances, and is presented in Figure 6.23. For low tolerances (small numbers of function evaluations), this data indicates that the fifth-order cases are competitive. For the sixth-order processes it is interesting that for lower tolerances the $\mathrm{FD}_9^6\mathrm{L}$ and $\mathrm{FMD}_9^6\mathrm{L}$ perform better, but are eventually equaled in efficiency by both of the eight-stage models. The RKN6(4)6FU actually appears more efficient than all others for tolerances $< 1{\times}10^{-5}$. Better global error behaviour (small SBD's) may, of course, be due to larger discrete global errors dominating the local errors, but in practice such a property *may* be acceptable. The difficulty at the derivation stage is knowing whether a practitioner would favour small discrete global errors and poor dense output, or vice versa. It is only possible to make a *compromise* for the likely user. It can not be ruled out that the minimal-stage models may be useful in certain circumstances, but it is difficult to make general recommendations. In the following chapters, it will be assumed that existing discrete

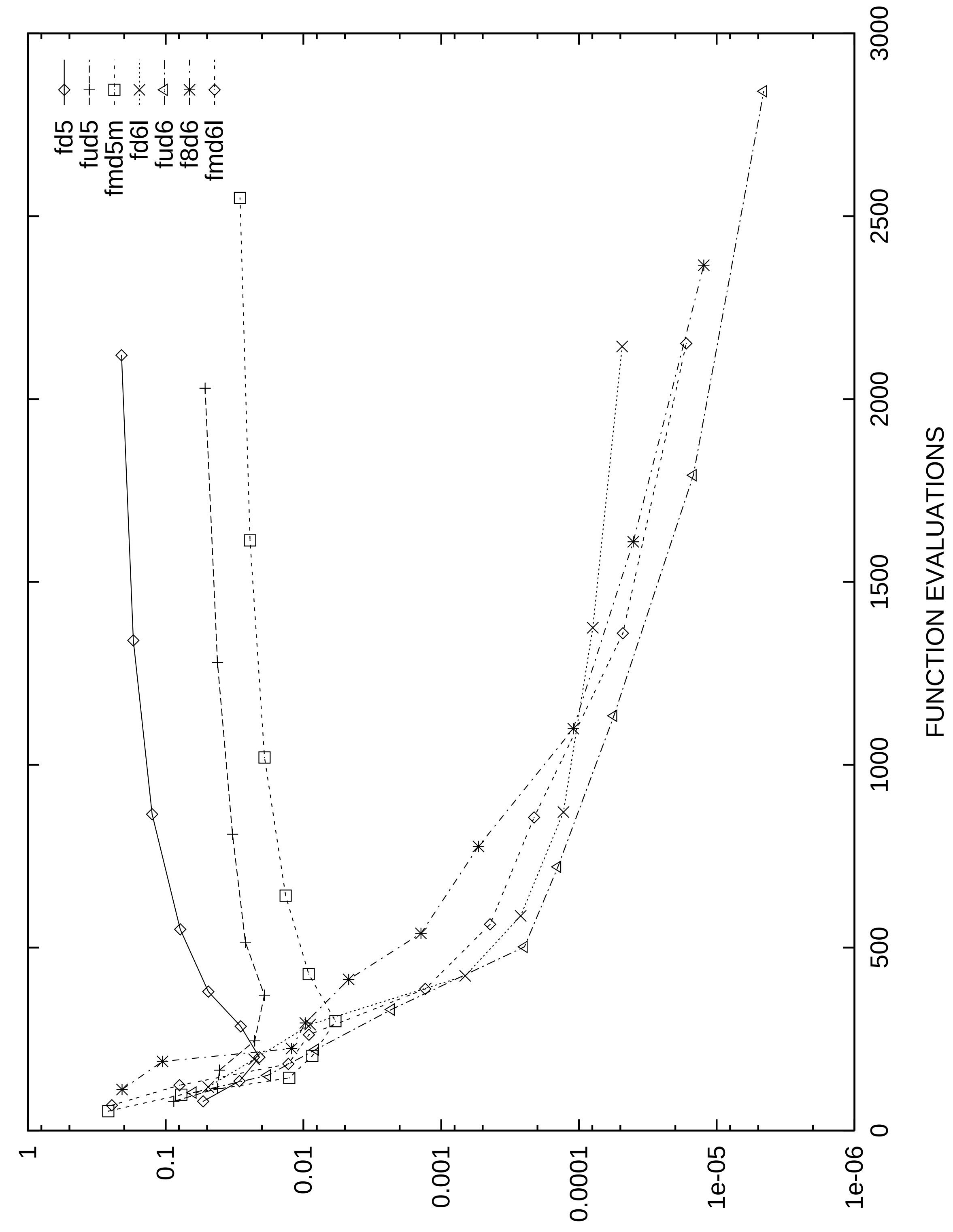


Figure 6.23: SBD′ against function evaluation plots for a D1 orbit.

models are the only starting point, requiring $p^* = q$ or $p^* = q - 1$ interpolation in the minimum number of stages. This removes the added complexity of compromising discrete output quality for the purposes of better, or cheaper, dense output.

6.5.2 Continuous extrapolation formulae

New $p^* = q$ continuous extensions derived in this chapter have highlighted how continuity requirements force an equation set for the y^* formula to contain fewer conditions than could clearly be solved otherwise — free b^*'s can be used no more usefully than in an optimisation process. Nyström $p^* = q - 1$ triples do not present this problem, being naturally one order higher for y^* ($p^{y^*} = q$), and therefore satisfying exactly the same number of order conditions. As mentioned previously, the b_i^* and $b_i'^*$ polynomial weights are simply related by calculus in this case. It would therefore seem ideal for a b_i^* polynomial set to also be automatically available, with no free parameters, in the $p^* = q$ case. The process of continuous extrapolation is exactly this, enabling the same integration law (5.35) to be used in generating a unique $p^{y^*} = q + 1$ interpolant. As detailed in Section 5.5, an extra stage may be added for C^2 continuity, and step size control can be done in two ways. These depend on whether the extrapolated process is considered to be a new triple, with a discrete y solution requiring s^* stages, or as a simple modification to the y solution to accommodate the discontinuous interpolant. In the latter case, a more efficient process is assured being that a rejected step only 'wastes' s stages. In order that fluctuations in the continuous global error can be seen, the usual D1 orbit problem is not used for demonstration purposes here. Instead, RK-AID problem 12 (the periodic second-order linear system) has been run with tolerance $1{\times}10^{-3}$ and the results plotted for four sixth-order methods. For each formula, results from the standard and extrapolated processes are shown for comparison. Use was made of the following batch-file, in this case for the RKN6(5)8F8:

```
batch, echo
problem 12
dense output
tol 1D-3
```

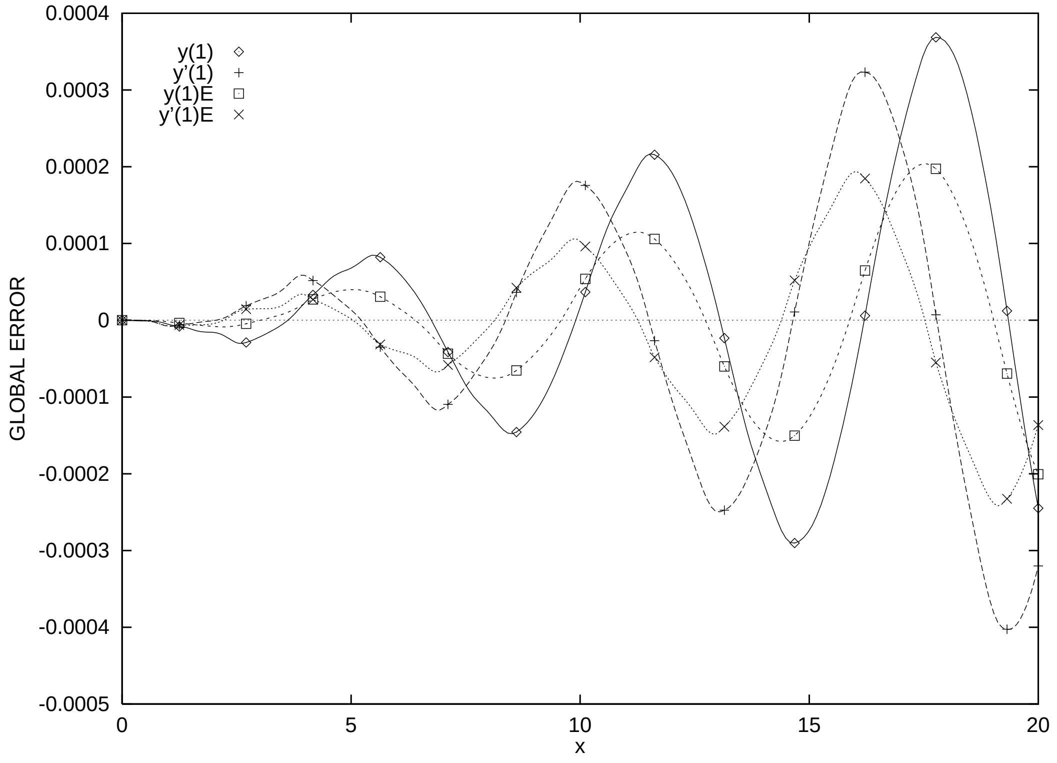


Figure 6.24: F8D$_8^6$ global error for RK-AID problem 12 ('E': extrapolated formula).

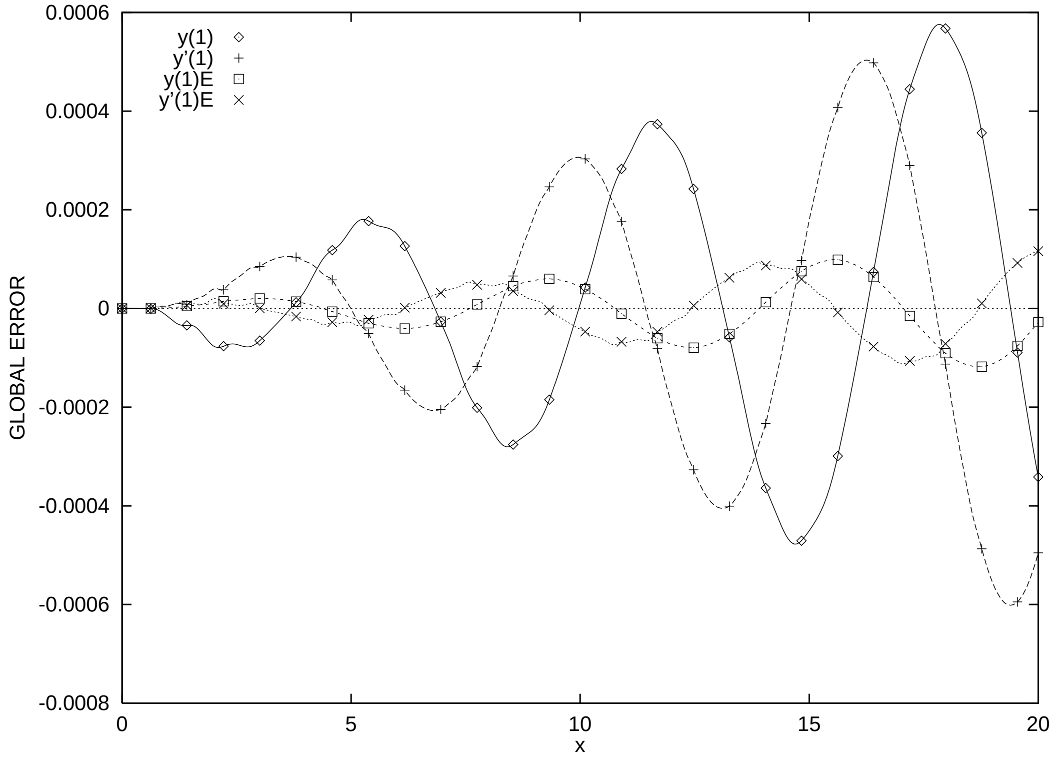


Figure 6.25: FUD$_8^6$L global error for RK-AID problem 12 ('E': extrapolated formula).

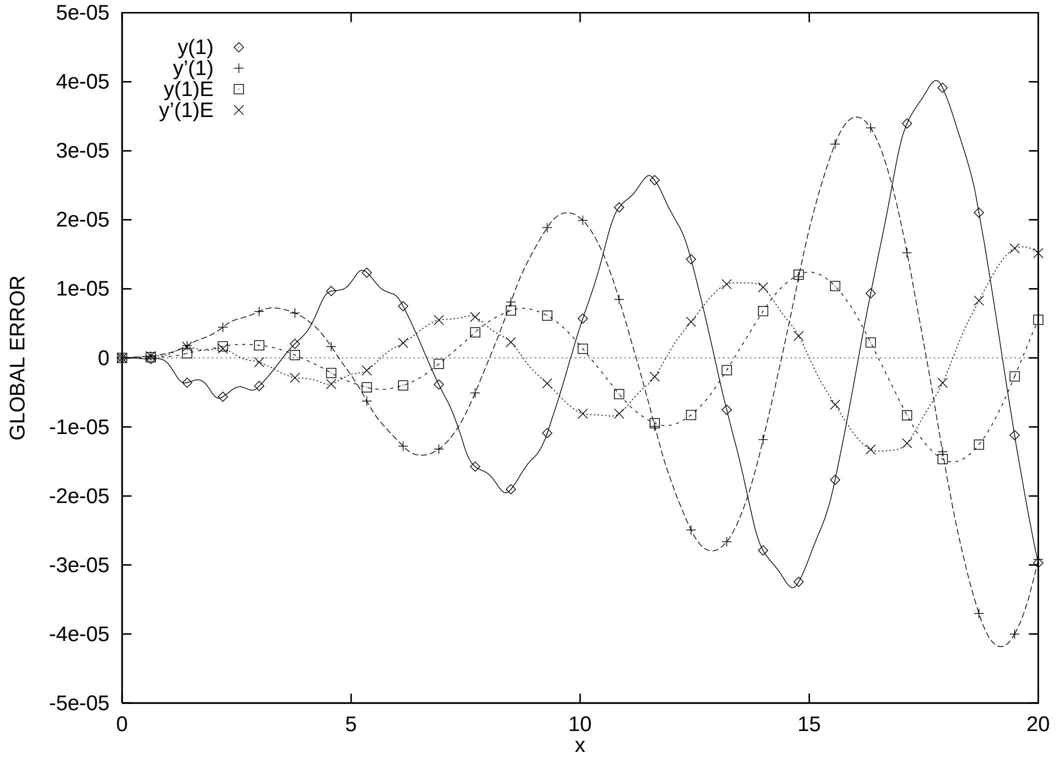


Figure 6.26: FD_9^6L global error for RK-AID problem 12 ('E': extrapolated formula).

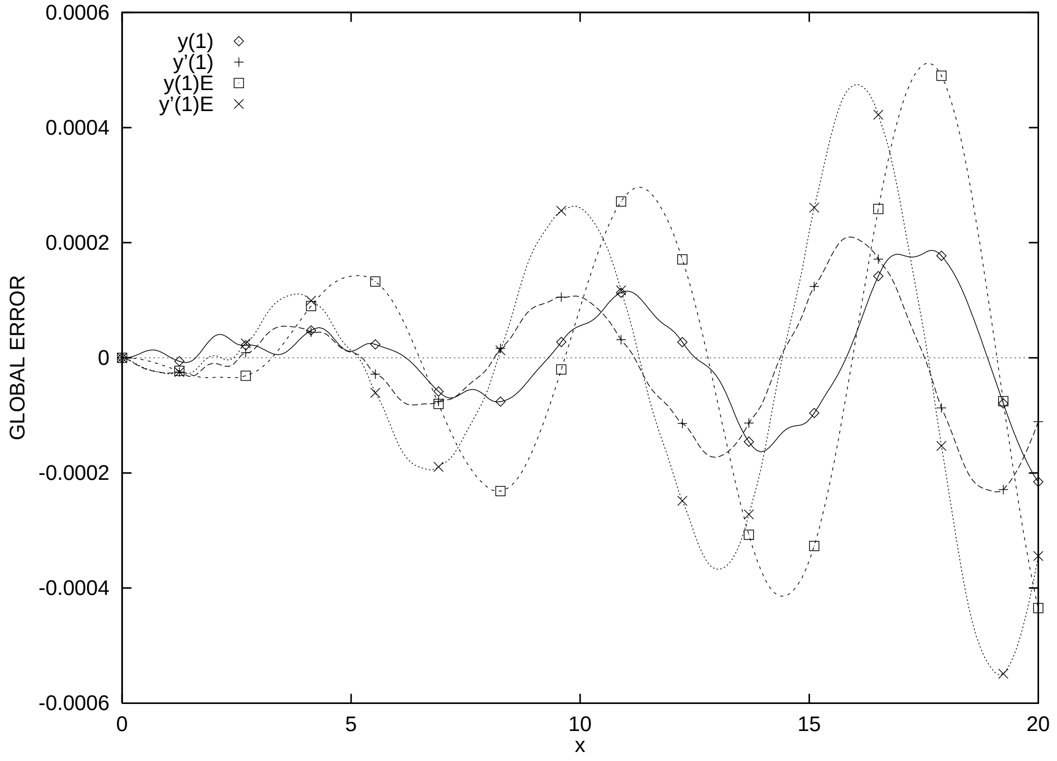


Figure 6.27: FMD_9^6L global error for RK-AID problem 12 ('E': extrapolated formula).

```
title
test 658f8d6 | go
postfix _extra | extrapolate | go
```

results for which are shown in Figures 6.24–6.27. These plots demonstrate viability of the continuous extrapolation technique, global error size in most cases is competitive, with the notable exception of the extrapolated RKN6(4)6FM formula. Dense output is excellent (in a box deviation sense) in all cases. This example serves mostly to demonstrate that the idea *can* work — problems may be envisaged relating to the step-size control 'tuning', for example. Note that, except for $N_l^{\prime(7)}/N_m^{\prime(7)}$, error term

	$N_l^{\prime(7)}$	$N_l^{\prime(8)}$	$N_l^{(8)}$	$N_l^{(9)}$
$\mathrm{FD}_9^6\mathrm{L}$	1.52×10^{-5}	2.48×10^{-5}	1.19×10^{-5}	1.74×10^{-5}
$\mathrm{FMD}_9^6\mathrm{L}$	6.32×10^{-6}	1.70×10^{-5}	6.95×10^{-6}	9.91×10^{-6}

Table 6.6: Integrated linear deviation norms for two extrapolated triples.

quantities are different for the extrapolated formulae, not least that $N^{(7)} = 0$. Eighth-order terms appear to have been controlled by the optimisation criterion applied to the non-extrapolated coefficients, for example, $N_l^{\prime(8)}/N_m^{\prime(8)}$ barely change. Principal/secondary linear deviation quantities for the extrapolated $\mathrm{FD}_9^6\mathrm{L}$ and $\mathrm{FMD}_9^6\mathrm{L}$ are given in Table 6.6 (compare with Tables 6.2–6.5). In future, extrapolation formulae could be derived more effectively by optimisation of the order $q+2$ and $q+3$ error terms for y concurrently with the y' norms as in the $p^* = q-1$ case. No further examples will be given, but it should be reiterated that *any* Nyström $p^* = q$ formula can be modified as an afterthought by use of the continuous extrapolation technique.

Chapter 7

Continuous eighth-order RK processes

7.1 Introduction

This chapter considers high-order Runge-Kutta embedded pairs for which a dense output facility either does not exist, or is currently unsatisfactory. With reference to Section 4.3.2, $\mathrm{RK}_{13}^{8(7)}$ pairs exist (no FSAL), and as a special case, $\mathrm{RK}_{12}^{8(6)}$ pairs are possible. Two particular formulae to be considered are the RK8(7)13M of Prince [12], and the Dormand and Prince RK8(6)12M published in Hairer, et al. [37], for which eighth-order continuous extensions are now developed. Seventh-order dense formulae will also be derived for both methods — such a process exists for the RK8(6)12M [37], but was not developed under any contemporary optimisation criterion. The following analysis assumes availability of the exact rational coefficients for each of the two method pairs. These are not published in either case, but have been obtained (Prince [65]).

7.2 Order conditions

Derivation of eighth-order dense RK formulae requires solution of 200 continuous truncation error terms to zero (see Table 4.2). This large jump in complexity from the sixth-order Nyström case of the previous chapter requires heavy use of simplifying conditions, essentially continuous extensions of those for the discrete models — see

Prince [12] and Hairer, et al. [37]. For convenience, the s_i^a, $i = 3, \ldots, s^*$, quantities may now be represented naturally by a vector, $\boldsymbol{s}^a$. Simplifying conditions to be employed for the continuous methods then have

$$s^b = 6,\ \boldsymbol{s}^a = (2, 3, 3, 4, 4, \ldots)^T.$$

The q_i, $i = 3, \ldots, s^*$, quantities may also be represented by a vector, $\boldsymbol{q}$, which for the RK8(7)13M and RK8(6)12M are as follows:

$$\boldsymbol{q} = (3, 3, 3, 4, 4, \ldots)^T, \text{ and } \boldsymbol{q} = (3, 3, 3, 5, 5, \ldots)^T,$$

respectively. Derivation of the higher-order formulae for both eighth-order models made use of the conditions

$$\widehat{R}_{0i} = 0,\ i = 1, \ldots, s,$$

equivalent conditions cannot, however, be imposed for the dense model and the equation set is subsequently more complicated. The reduction mode of RK-AID is now invaluable, with the two processes sharing a fundamental set of simplifying conditions allowing a parameterised set of preliminary ${}^\dagger\tau^*$'s to be generated with the following command set:

```
[s] order :8, parameterise
!
b[s](2:5) = 0 & Q(3:s, 1;2) = 0 & Q(6:s, 3) = 0
a(4:s, 2) = 0 & a(6:s, 3) = 0
```

giving

1. $b_1^* + \sum_{i=6}^{s^*} b_i^* - 1$
2. $\sum_{i=6}^{s^*} b_i^* c_i^k - \frac{\sigma^k}{k+1}, \quad k = 1:7$
3. $\sum_{i=6}^{s^*} b_i^* c_i^k Q_{i4}, \quad k = 0:2$
4. $c_4^k Q_{4,3} \sum_{i=6}^{s^*} b_i^* c_i^l a_{i4} + c_5^k Q_{5,3} \sum_{i=6}^{s^*} b_i^* c_i^l a_{i5}, \quad k + l < 3$

5. $c_4^k a_{4,3} \sum_{i=6}^{s^*} b_i^* c_i^l a_{i4} + c_5^k a_{5,3} \sum_{i=6}^{s^*} b_i^* c_i^l a_{i5}, \quad k+l<3$

6. $\sum_{i=6}^{s^*} b_i^* c_i^k Q_{i5}, \quad k=0:1$

7. $\sum_{i=6}^{s^*} b_i^* c_i^l \sum_{j=4}^{i-1} a_{ij} c_j^m Q_{j4}, \quad l+m<2$

8. $\sum_{i=6}^{s^*} b_i^* c_i^m \sum_{j=4}^{i-1} a_{ij} c_j^n \sum_{k=3}^{j-1} a_{jk} c_k^o Q_{k3}, \quad m+n+o<2$

9. $c_4^k a_{4,3} \sum_{i=6}^{s^*} b_i^* c_i^l a_{i4} + c_5^k a_{5,3} \sum_{i=6}^{s^*} b_i^* c_i^l a_{i5}, \quad k+l<2$

10. $\sum_{i=6}^{s^*} b_i^* c_i^m \sum_{j=5}^{i-1} a_{ij} c_j^n \sum_{k=4}^{j-1} a_{jk} c_k^o a_{k3}, \quad m+n+o<2$

11. $\sum_{i=6}^{s^*} b_i^* Q_{i6}$

12. $\sum_{i=6}^{s^*} b_i^* \sum_{j=4}^{i-1} a_{ij} Q_{j5}$

13. $\sum_{i=6}^{s^*} b_i^* \sum_{j=4}^{i-1} a_{ij} \sum_{k=3}^{j-1} a_{jk} Q_{k4}$

14. $\sum_{i=6}^{s^*} b_i^* \sum_{j=4}^{i-1} a_{ij} \sum_{k=3}^{j-1} a_{jk} \sum_{l=2}^{k-1} a_{kl} Q_{l3}$

15. $\sum_{i=6}^{s^*} b_i^* \sum_{j=5}^{i-1} a_{ij} \sum_{k=4}^{j-1} a_{jk} a_{k3}$

16. $\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} \sum_{k=5}^{j-1} a_{jk} \sum_{l=4}^{k-1} a_{kl} a_{l3}$

While these expressions could be solved in this form, it should be noticed that (4), (5) and (9) have been 'split' by RK-AID to suggest possible linear independence of the summation sections they contain. For example, consider two of the $l=1$ forms of these equations, i.e.,

$$c_4^k Q_{4,3} A_4 + c_5^k Q_{5,3} A_5 = 0, \text{ and } c_4^k a_{4,3} A_4 + c_5^k a_{5,3} A_5 = 0, \quad k=0,1, \tag{7.1}$$

from (4) and (5) above, where

$$A_j = \sum_{i=6}^{s^*} b_i^* c_i a_{ij}.$$

Solution of (7.1) may be written

$$\begin{pmatrix} Q_{4,3} & Q_{5,3} \\ c_4 Q_{4,3} & c_5 Q_{5,3} \\ a_{4,3} & a_{5,3} \\ c_4 a_{4,3} & c_5 a_{5,3} \end{pmatrix} \begin{pmatrix} A_4 \\ A_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

and unless a linear dependence exists between the columns of the coefficient matrix, $A_4 = A_5 = 0$ is the only solution. This is therefore assumed by the model, and expressions (4), (5) and (9) imply the equivalent conditions

$$\sum_{i=6}^{s^*} b_i^* c_i^k a_{i4} = \sum_{i=6}^{s^*} b_i^* c_i^k a_{i5} = 0, \ \ k = 0, 1, 2. \tag{7.2}$$

If these are now worked into the model, expressions (8), (10) and (15) may be shown to be equivalent to

$$\sum_{i=7}^{s^*} b_i^* c_i^k \sum_{j=6}^{i-1} a_{ij} c_j^l a_{j4} = \sum_{i=7}^{s^*} b_i^* c_i^k \sum_{j=6}^{i-1} a_{ij} c_j^l a_{j5} = 0, \ \ k + l < 2, \tag{7.3}$$

again assuming a non-singular system for determination of the two summation sections. Finally, expressions (14) and (16) may be simplified with both of (7.2) and (7.3), then split to give

$$\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k4} = \sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k5} = 0.$$

An identical strategy was required for solution of the discrete model (Prince [12]), and may be emulated by the automatic implied algebraic reduction (or 'equation splitting') facility of RK-AID, by addition of the command `SPLIT [s]` (see Sections 3.3.4

and A.6.22). A final equation set can be generated in this way to give the following (non-parameterised) expressions, where quadrature terms have been removed:

$$\begin{array}{l}
\sum_{i=6}^{s^*} b_i^* Q_{i4} \\
\sum_{i=6}^{s^*} b_i^* a_{i4} \\
\sum_{i=6}^{s^*} b_i^* a_{i5} \\
\sum_{i=6}^{s^*} b_i^* c_i Q_{i4} \\
\sum_{i=6}^{s^*} b_i^* Q_{i5} \\
\sum_{i=6}^{s^*} b_i^* c_i a_{i4} \\
\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} Q_{j4} \\
\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} a_{j4}
\end{array}
\quad
\begin{array}{|l|}
\sum_{i=6}^{s^*} b_i^* c_i a_{i5} \\
\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} a_{j5} \\
\hline
\sum_{i=6}^{s^*} b_i^* c_i^2 Q_{i4} \\
\sum_{i=6}^{s^*} b_i^* Q_{i6} \\
\sum_{i=6}^{s^*} b_i^* c_i Q_{i5} \\
\sum_{i=6}^{s^*} b_i^* c_i^2 a_{i4} \\
\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} Q_{j4} \\
\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j Q_{j4} \\
\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} Q_{j5}
\end{array}
\quad
\begin{array}{l}
\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} a_{j4} \\
\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j a_{j4} \\
\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} Q_{k4} \\
\sum_{i=6}^{s^*} b_i^* c_i^2 a_{i5} \\
\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k4} \\
\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} a_{j5} \\
\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j a_{j5} \\
\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k5}
\end{array}$$

the horizontal line being to separate the seventh- and additional eighth-order terms (reading from top-to-bottom and left-to-right). These 25 non-quadrature order conditions can now be parameterised with $n_\alpha = 15$ to yield the equivalent equation set in Figure 7.1, where

$$\alpha_{i1} = Q_{i4},\ \alpha_{i2} = a_{i4},\ \alpha_{i3} = a_{i5},\ \alpha_{i4} = Q_{i5},\ \alpha_{i5} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j1},\ \alpha_{i6} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j1},$$

$$\alpha_{i7} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j2},\ \alpha_{i8} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j2},\ \alpha_{i9} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j3},\ \alpha_{i10} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j3}{}^{\dagger},\ \alpha_{i11} = Q_{i6}{}^{\dagger},$$

$$\alpha_{i12} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j4}{}^{\dagger},\ \alpha_{i13} = \sum_{j=7}^{i-1} a_{ij}\alpha_{j5}{}^{\dagger},\ \alpha_{i14} = \sum_{j=7}^{i-1} a_{ij}\alpha_{j7}{}^{\dagger},\ \alpha_{i15} = \sum_{j=7}^{i-1} a_{ij}\alpha_{j9}{}^{\dagger},$$

$$\sum_{i=6}^{s^*} b_i^* \alpha_{i1} = 0$$
$$\sum_{i=6}^{s^*} b_i^* c_i \alpha_{i1} = 0$$
$$\sum_{i=6}^{s^*} b_i^* c_i^2 \alpha_{i1} = 0^\dagger$$
$$\sum_{i=6}^{s^*} b_i^* \alpha_{i2} = 0$$
$$\sum_{i=6}^{s^*} b_i^* c_i \alpha_{i2} = 0$$
$$\sum_{i=6}^{s^*} b_i^* c_i^2 \alpha_{i2} = 0^\dagger$$
$$\sum_{i=6}^{s^*} b_i^* \alpha_{i3} = 0$$
$$\sum_{i=6}^{s^*} b_i^* c_i \alpha_{i3} = 0$$

$$\sum_{i=6}^{s^*} b_i^* c_i^2 \alpha_{i3} = 0^\dagger$$
$$\sum_{i=6}^{s^*} b_i^* \alpha_{i4} = 0$$
$$\sum_{i=6}^{s^*} b_i^* c_i \alpha_{i4} = 0^\dagger$$
$$\sum_{i=7}^{s^*} b_i^* \alpha_{i5} = 0$$
$$\sum_{i=7}^{s^*} b_i^* c_i \alpha_{i5} = 0^\dagger$$
$$\sum_{i=7}^{s^*} b_i^* \alpha_{i6} = 0^\dagger$$
$$\sum_{i=7}^{s^*} b_i^* \alpha_{i7} = 0$$
$$\sum_{i=7}^{s^*} b_i^* c_i \alpha_{i7} = 0^\dagger$$
$$\sum_{i=7}^{s^*} b_i^* \alpha_{i8} = 0^\dagger$$

$$\sum_{i=7}^{s^*} b_i^* \alpha_{i9} = 0$$
$$\sum_{i=7}^{s^*} b_i^* c_i \alpha_{i9} = 0^\dagger$$
$$\sum_{i=7}^{s^*} b_i^* \alpha_{i10} = 0^\dagger$$
$$\sum_{i=7}^{s^*} b_i^* \alpha_{i11} = 0^\dagger$$
$$\sum_{i=7}^{s^*} b_i^* \alpha_{i12} = 0^\dagger$$
$$\sum_{i=8}^{s^*} b_i^* \alpha_{i13} = 0^\dagger$$
$$\sum_{i=8}^{s^*} b_i^* \alpha_{i14} = 0^\dagger$$
$$\sum_{i=8}^{s^*} b_i^* \alpha_{i15} = 0^\dagger$$

Figure 7.1: Parameterised continuous non-quadrature equations for the RK8(7)13M.

and terms specific to the eighth-order systems are marked with a dagger (†). These expressions form the final model for a D^8 extension on the RK8(7)13M, with the additional conditions

$$Q_{i4} = 0 \ (q_i = 5), \ i = 6, \ldots, s^*,$$

for the RK8(6)12M model giving

$$\alpha_{i1} = \alpha_{i5} = \alpha_{i6} = \alpha_{i13} = 0, \ \forall i, \tag{7.4}$$

simplifying the equation set substantially. Seventh-order interpolants on either formula are relatively simple to provide, and will also be shown to exist naturally as embedded processes within the continuous eighth-order models. In each case, a reusable stage must be added with $s^r = s+1$, and the $\overline{A}_i$ systems have $s^a = 6$ due to the presence of parameterised a_{ij} parameters, i.e., $\alpha_{i2} = a_{i4}$ and $\alpha_{i3} = a_{i5}$ (see the definition of s^a in Section 5.3).

7.3 Dense output for the RK8(7)13M

A small addition to the notation may be introduced at this point to distinguish between the eighth- and seventh-order models, by use of a bracketed superscript integer on any symbol to denote the order of the model. For example, the $\boldsymbol{b}^*$ matrix for seventh-order dense may be denoted as $W^{(7)}$, having $n_q^{(7)} = 6$ quadrature terms, its non-quadrature part being $M^{(7)}$, with internal vector $\boldsymbol{\nu}_i^{b(7)}$.

7.3.1 Seventh- and eighth-order equation systems

The non-quadrature $\boldsymbol{b}^*$ matrix for eighth-order dense, $M^{(8)}$, in this case has

$$n_q^{(8)} = 7, \, n^{(8)} = 25, \ \boldsymbol{\nu}_i^{b(8)} = \Big(\alpha_{i1}, c_i\alpha_{i1}, c_i^2\alpha_{i1}, \alpha_{i2}, c_i\alpha_{i2}, c_i^2\alpha_{i2}, \alpha_{i3}, c_i\alpha_{i3}, c_i^2\alpha_{i3}, \alpha_{i4}, c_i\alpha_{i4},$$
$$\alpha_{i5}, c_i\alpha_{i5}, \alpha_{i6}, \alpha_{i7}, c_i\alpha_{i7}, \alpha_{i8}, \alpha_{i9}, c_i\alpha_{i9}, \alpha_{i10}, \alpha_{i11}, \alpha_{i12}, \alpha_{i13}, \alpha_{i14}, \alpha_{i15}\Big)^T.$$

The system for $a_{ij},\ j = s^a, \ldots, i-1,\ \overline{A}_i^{(8)}$, has

$$n_Q^{(8)} = 6,\ n_a^{(8)} = 10,\ \boldsymbol{\nu}_i^{d(8)} = (\alpha_{i1}, c_i\alpha_{i1}, \alpha_{i2}, c_i\alpha_{i2}, \alpha_{i3}, c_i\alpha_{i3}, \alpha_{i4}, \alpha_{i5}, \alpha_{i7}, \alpha_{i9})^T,$$

$$\boldsymbol{\nu}_i^{r(8)} = (\alpha_{i5}, \alpha_{i6}, \alpha_{i7}, \alpha_{i8}, \alpha_{i9}, \alpha_{i10}, \alpha_{i12}, \alpha_{i13}, \alpha_{i14}, \alpha_{i15})^T,$$

and

$$\boldsymbol{r}_i^{(8)} = \left(\frac{c_i^2}{2} - \frac{5\,\alpha_{i3}}{16} - \frac{\alpha_{i2}}{8},\ \frac{c_i^3}{3} - \frac{25\,\alpha_{i3}}{256} - \frac{\alpha_{i2}}{64},\ \frac{c_i^4}{4} - \frac{125\,\alpha_{i3}}{4096} - \frac{\alpha_{i2}}{512},\right.$$
$$\frac{c_i^5}{5} + \alpha_{i1} - \frac{625\,\alpha_{i3}}{65536} - \frac{\alpha_{i2}}{4096},\ \frac{c_i^6}{6} + \alpha_{i4} - \frac{3125\,\alpha_{i3}}{1048576} - \frac{\alpha_{i2}}{32768},$$
$$\left.\frac{c_i^7}{7} + \alpha_{i11} - \frac{15625\,\alpha_{i3}}{16777216} - \frac{\alpha_{i2}}{262144}\right)^T,$$

where it can be seen that $\alpha_{i2} \equiv a_{i4}$ and $\alpha_{i3} \equiv a_{i5}$ have been taken to the left-hand-side of $\overline{A}_i^{(8)}$ (N.B., $s^a = 6$), and thus multiplied by the relevant powers of $c_4 = 1/8$ and $c_5 = 5/16$ as required. Additionally for seventh-order dense,

$$n_q^{(7)} = 6,\ n^{(7)} = 10,\ \boldsymbol{\nu}_i^{b(7)} = (\alpha_{i1}, c_i\alpha_{i1}, \alpha_{i2}, c_i\alpha_{i2}, \alpha_{i3}, c_i\alpha_{i3}, \alpha_{i4}, \alpha_{i5}, \alpha_{i7}, \alpha_{i9})^T,$$

$$n_Q^{(7)} = 5,\ n_a^{(7)} = 3,\ \boldsymbol{\nu}_i^{d(7)} = (\alpha_{i1}, \alpha_{i2}, \alpha_{i3})^T,\ \boldsymbol{\nu}_i^{r(7)} = (\alpha_{i5}, \alpha_{i7}, \alpha_{i9})^T,$$

and

$$\boldsymbol{r}_i^{(7)} = \left(\frac{c_i^2}{2} - \frac{5\,\alpha_{i3}}{16} - \frac{\alpha_{i2}}{8},\ \frac{c_i^3}{3} - \frac{25\,\alpha_{i3}}{256} - \frac{\alpha_{i2}}{64},\ \frac{c_i^4}{4} - \frac{125\,\alpha_{i3}}{4096} - \frac{\alpha_{i2}}{512},\right.$$
$$\left.\frac{c_i^5}{5} + \alpha_{i1} - \frac{625\,\alpha_{i3}}{65536} - \frac{\alpha_{i2}}{4096},\ \frac{c_i^6}{6} + \alpha_{i4} - \frac{3125\,\alpha_{i3}}{1048576} - \frac{\alpha_{i2}}{32768}\right)^T.$$

The reason for having $s^a = s^b$ here (and in every other model considered) should now become clear — as $n_q^{(7)} \equiv n_Q^{(8)}$ and $\boldsymbol{\nu}_i^{b(7)} \equiv \boldsymbol{\nu}_i^{d(8)}$, the abbreviated seventh-order coefficient matrix for $b_i^*,\ i = s^b \ldots, s^*,\ M^{(7)}$, is identical to the abbreviated eighth-order $a_{ij},\ j = s^a, \ldots, i-1$, matrix, $A_i^{(8)}$. Such a feature will vastly simplify analysis of the eighth-order model, and will subsequently allow trivial determination of seventh-order formulae. The structure of systems $\overline{W}^{(7)}$ and $\overline{A}_i^{(8)}$ is outlined in Figures 7.2 and 7.3, wherein rational integers have been replaced by the symbol '$\times$'. It has

$$\overline{W}^{(7)} = \left(\begin{array}{cccccccccc|c}
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k & \frac{\sigma}{2} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^2 & \frac{\sigma^2}{3} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^3 & \frac{\sigma^3}{4} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^4 & \frac{\sigma^4}{5} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^5 & \frac{\sigma^5}{6} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^6 & \frac{\sigma^6}{7} \\
\hline
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k1} & 0 \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & c_k\alpha_{k1} & 0 \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k2} & 0 \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & c_k\alpha_{k2} & 0 \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k3} & 0 \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & c_k\alpha_{k3} & 0 \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k4} & 0 \\
0 & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k5} & 0 \\
0 & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k7} & 0 \\
0 & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k9} & 0
\end{array}\right)$$

Figure 7.2: The $b_i^*,\ i = 6, \ldots, s^*$, system for 7^{th} order dense on the RK8(7)13M.

$$\overline{A}_i^{(8)} = \left(\begin{array}{cccccccccc|c}
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k & \frac{c_i^2}{2} - \frac{5\,\alpha i_3}{16} - \frac{\alpha_{i2}}{8} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^2 & \frac{c_k^3}{3} - \frac{25\,\alpha_{i3}}{256} - \frac{\alpha i_2}{64} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_i^3 & \frac{c_k^4}{4} - \frac{125\,\alpha_{i3}}{4096} - \frac{\alpha_{i2}}{512} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^4 & \frac{c_i^5}{5} + \alpha_{i1} - \frac{625\,\alpha_{i3}}{65536} - \frac{\alpha_{i2}}{4096} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^5 & \frac{c_i^6}{6} + \alpha_{i4} - \frac{3125\,\alpha i_3}{1048576} - \frac{\alpha_{i2}}{32768} \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & c_k^6 & \frac{c_i^7}{7} + \alpha_{i11} - \frac{15625\,\alpha_{i3}}{16777216} - \frac{\alpha_{i2}}{262144} \\
\hline
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k1} & \alpha_{i5} \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & c_k\alpha_{k1} & \alpha_{i6} \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k2} & \alpha_{i7} \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & c_k\alpha_{k2} & \alpha_{i8} \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k3} & \alpha_{i9} \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & c_k\alpha_{k3} & \alpha_{i10} \\
\times & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k4} & \alpha_{i12} \\
0 & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k5} & \alpha_{i13} \\
0 & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k7} & \alpha_{i14} \\
0 & \times & \times & \times & \times & \times & \times & \times & 0 & \alpha_{k9} & \alpha_{i15}
\end{array}\right)$$

Figure 7.3: The $a_{ij},\ j = 6, \ldots, i-1$, system for 8^{th} order dense on the RK8(7)13M.

been assumed that detail required for such systems does not extend to the rationals themselves — showing the 'true' system would not (easily) be possible in print due to the number of digits involved. For example,

$$a_{11,9} = \frac{-847205714160239289113307424793539077951658318917591980262304042838612275700008766\,01}{633587043839807269984161128303227064853003326302890606270194592859608259795885606\,9} \cdots$$

$$\cdots \frac{695770093019554505337422084139866018794462110706582931060886539402641825835\,5}{746043830625361109589149156559097143238748941588410373201257425589787832182\,4},$$

and evaluated 'α_i' quantities become even larger (in their number of rational digits rather than absolute numerical size). In future, all large systems are assumed to be adequately specified by the coefficients of the discrete model and '$\boldsymbol{\nu}_i$' vectors, and may not (necessarily) be given — as is the case for $\overline{W}^{(8)}$ (and $\overline{A}_i^{(7)}$) here.

7.3.2 Pre-analysis for eighth-order dense

Progress may now be made in the eighth-order case by determining the partially eliminated non-quadrature $\boldsymbol{b}^*$ and $\boldsymbol{a}_i$ system forms. The rank of the numerical part of the $M^{(8)}$ system ($r_b^{(8)}$) is 7, and performing 7 pivoting operations produces the equivalent sectioned form:

$$M^{(8)} = \left(\begin{array}{c|c} T_b^{(8)} & \boldsymbol{\zeta}_k^{b(8)} \\ \hline 0 & \boldsymbol{\zeta}_k^{\beta(8)} \end{array}\right), \tag{7.5}$$

where the size of the $\boldsymbol{\zeta}_i^{b(8)}$ and $\boldsymbol{\zeta}_i^{\beta(8)}$ vectors are $r_b^{(8)} = 7$ and $n_\beta^{(8)} = n^{(8)} - r_b^{(8)} = 18$, respectively. The minimum number of stages for the model is now

$$s^* = s^b + n_q^{(8)} + r_b^{(8)} - 1 = 19,$$

which would be the case if each element of $\boldsymbol{\zeta}_i^{\beta(8)}$ could be made zero. Keeping other options available, however, a 'β-parameterisation' will be done after equivalently treating the $\overline{A}_i^{(8)}$ system (see Figure 7.3). This has $r_a^{(8)} = 8$, and 8 pivoting operations yields

$$\overline{A}_i^{(8)} = \left(\begin{array}{c|c|c} T_a^{(8)} & \boldsymbol{\zeta}_k^{d(8)} & \boldsymbol{\zeta}_i^{r(8)} \\ \hline 0 & \boldsymbol{\zeta}_k^{\delta(8)} & \boldsymbol{\zeta}_i^{\rho(8)} \end{array}\right), \tag{7.6}$$

where the size of the $\boldsymbol{\zeta}_i^{d(8)}/\boldsymbol{\zeta}_i^{r(8)}$ vectors is $r_a^{(8)} = 8$ and $\boldsymbol{\zeta}_i^{\delta(8)}/\boldsymbol{\zeta}_i^{\rho(8)}$ have size $n_\delta^{(8)} = n_a^{(8)} + n_Q^{(8)} - r_a^{(8)} = 8$. Note that an equivalent form for the seventh-order $\boldsymbol{b}^*$ system (Figure 7.2) is then

$$\overline{W}^{(7)} = \left(\begin{array}{c|c|c} T_a^{(8)} & \boldsymbol{\zeta}_k^{d(8)} & \\ \cline{1-2} 0 & \boldsymbol{\zeta}_k^{\delta(8)} & \raisebox{1.5ex}{$\boldsymbol{p}_\sigma$} \end{array}\right), \tag{7.7}$$

for $\boldsymbol{p}_\sigma$ a vector of polynomials in σ, and the two systems share their abbreviated coefficient matrices. This will henceforth be referred to as *duality*, and will prove invaluable in the avoidance of possible singular cases in the following analysis. Importantly, $M^{(7)}$ has $r_b^{(7)} = 5$, therefore 5 pivoting operations yields

$$M^{(7)} = \left(\begin{array}{c|c} T_b^{(7)} & \boldsymbol{\zeta}_k^{b(7)} \\ \hline 0 & \boldsymbol{\zeta}_k^{\beta(7)} \end{array}\right),$$

and the vectors $\boldsymbol{\zeta}_i^{\beta(8)}$ from (7.5) and $\boldsymbol{\zeta}_i^{\delta(8)}$ from (7.6) should contain all $n_\beta^{(7)} = n^{(7)} - r_b^{(7)} = 5$ elements of $\boldsymbol{\zeta}_i^{\beta(7)}$. This would not *necessarily* occur naturally out of the elimination processes, and has been ensured in this case by performing the Gaussian-elimination on particular subsets of the matrix rows prior to treating the complete systems. For example, where $\overline{A}_i^{(8)}$ is concerned, the last 10 rows were dealt with separately (with 5 pivoting operations) before adding the remaining rows and eliminating 8 times as required. A similar trick has been used on the $M^{(8)}$ matrix. The common expressions of $\boldsymbol{\zeta}_i^{\delta(8)}$ and $\boldsymbol{\zeta}_i^{\beta(8)}$ have been forced into the first five places of each vector for convenience in the next section.

7.3.3 Change of variables for eighth-order dense

Based on the treatment outlined in Section 5.4.1, the α_i's may be removed from the system using

$$\beta_{ik} = {}^k\zeta_i^{\beta(8)},\ k = 1,\dots,n_\beta,\ \delta_{ik} = {}^k\zeta_i^{\delta(8)},\ k = 1,\dots,n_\delta,\ \rho_{ik} = {}^k\zeta_i^{\rho(8)},\ k = 1,\dots,n_\delta,$$

to form the linear systems:

$$\Upsilon_i^\beta \boldsymbol{\alpha}_i = \boldsymbol{\beta}_i, \tag{7.8}$$

$$\Upsilon_i^\delta \boldsymbol{\alpha}_i = \boldsymbol{\delta}_i,\ \widetilde{\Upsilon}_i^\delta \boldsymbol{\alpha}_i = \widetilde{\boldsymbol{\delta}}_i, \tag{7.9}$$

and

$$\Upsilon_i^\rho \boldsymbol{\alpha}_i = \boldsymbol{\rho}_i,\ \widetilde{\Upsilon}_i^\rho \boldsymbol{\alpha}_i = \widetilde{\boldsymbol{\rho}}_i, \tag{7.10}$$

where

$$\boldsymbol{\alpha}_i = (\alpha_{i1},\dots,\alpha_{i15})^T,\ \boldsymbol{\beta}_i = (\beta_{i1},\dots,\beta_{i18})^T,\ \widetilde{\boldsymbol{\delta}}_i = \overline{\boldsymbol{\delta}}_i + \boldsymbol{p}_i^\delta,\ \widetilde{\boldsymbol{\rho}}_i = \overline{\boldsymbol{\rho}}_i + \boldsymbol{p}_i^\rho,$$

$$\boldsymbol{\delta}_i = (\delta_{i1},\dots,\delta_{i5})^T,\ \overline{\boldsymbol{\delta}}_i = (\delta_{i6},\dots,\delta_{i8})^T,\ \boldsymbol{\rho}_i = (\rho_{i1},\dots,\rho_{i5})^T,\ \overline{\boldsymbol{\rho}}_i = (\rho_{i6},\dots,\rho_{i8})^T,$$

$$\boldsymbol{p}_i^\delta = \left(p_{i6}^\delta,\dots,p_{i8}^\delta\right)^T,\ \boldsymbol{p}_i^\rho = (p_{i6}^\rho,\dots,p_{i8}^\rho)^T,$$

and the p_{ik} quantities are simple polynomials in c_i. Note that systems (7.9) and (7.10) have been split into two to embody the fact that the first five elements of $\boldsymbol{\zeta}_i^{\delta(8)}$ and $\boldsymbol{\zeta}_i^{\rho(8)}$ are linear combinations of 'α_i' quantities only, i.e., not containing polynomials in c_i as is the general case. This feature is due to equivalence of the $A_i^{(8)}$ and $M^{(7)}$ systems as mentioned previously, and enables a composite system

$$\Upsilon_i \boldsymbol{\alpha}_i = \boldsymbol{\upsilon}_i, \tag{7.11}$$

to be formed, where

$$\Upsilon_i = \begin{pmatrix} \Upsilon_i^{\beta} \\ \Upsilon_i^{\delta} \\ \Upsilon_i^{\rho} \end{pmatrix}, \ \boldsymbol{v}_i = \begin{pmatrix} \boldsymbol{\beta}_i \\ \boldsymbol{\delta}_i \\ \boldsymbol{\rho}_i \end{pmatrix},$$

that does not contain any c_i polynomials in its right-hand-side vector. This may loosely be termed the 'linear' Υ_i system, solution of which in isolation simplifies the change of variables procedure tailored to forcing linear dependence on the β matrix. Additionally, the 'nonlinear' Υ_i system is as follows:

$$\widetilde{\Upsilon}_i \boldsymbol{\alpha}_i = \widetilde{\boldsymbol{v}}_i, \tag{7.12}$$

where

$$\widetilde{\Upsilon}_i = \begin{pmatrix} \widetilde{\Upsilon}_i^{\delta} \\ \widetilde{\Upsilon}_i^{\rho} \end{pmatrix}, \ \widetilde{\boldsymbol{v}}_i = \begin{pmatrix} \widetilde{\boldsymbol{\delta}}_i \\ \widetilde{\boldsymbol{\rho}}_i \end{pmatrix},$$

and the complete system for specification of the $\alpha_{ik},\ k = 1, \ldots, 15$, is thus:

$$\widehat{\Upsilon}_i \boldsymbol{\alpha}_i = \widehat{\boldsymbol{v}}_i, \tag{7.13}$$

where

$$\widehat{\Upsilon}_i = \begin{pmatrix} \Upsilon_i \\ \widetilde{\Upsilon}_i \end{pmatrix}, \ \widehat{\boldsymbol{v}}_i = \begin{pmatrix} \boldsymbol{v}_i \\ \widetilde{\boldsymbol{v}}_i \end{pmatrix}.$$

Now considering system (7.11) in its augmented form $\overline{\Upsilon}_i$, equivalence of

$$ {}^{k}\zeta_i^{\beta(7)} = {}^{k}\zeta_i^{\beta(8)} = {}^{k}\zeta_i^{\delta(8)}, \ k = 1, \ldots, 5,$$

is best treated by introducing the general purpose quantities $\varphi_{ik},\ k = 1, \ldots, 5$, to replace $\beta_{ik},\ \delta_{ik},\ k = 1, \ldots, 5$. The '$\varphi_i$' are key parameters that control:

1. the $\beta^{(7)}$ matrix,

2. five rows of the $\beta^{(8)}$ matrix, and

3. five rows of the $D_i^{(8)}$ matrices.

It is worth reiterating that (7.11) need not be *solved*, but just made consistent such that the α_i's can be calculated from Fortran 90. This will remove a large amount of complexity from the expressions. Structure of the augmented system $\overline{\Upsilon}_i$ is indicated in Figure 7.4, where, again, rational integers have been replaced by '$\times$' and polynomials in c_i are represented by $P_i^{l:m}$, where l and m are the lower and upper range of the powers of c_i. For example, the coefficient of α_{i1} for the equation

$$\varphi_{i1} = {}^1\zeta_i^{\beta(7)} \equiv {}^1\zeta_i^{\beta(8)} \equiv {}^1\zeta_i^{\delta(8)},$$

(the top-left element of Υ_i) is

$$-\frac{596016934812826419200000\,c_i - 42754021546065120399936}{2346276711242964431 91}. \tag{7.14}$$

In determining the consistency conditions of $\overline{\Upsilon}_i$, a careful choice of pivots to include the elements of 1, or the rational entries, will vastly simplify the process. While this could be done 'by hand' very easily here, an automatic method would be preferable for more complicated models. An ideal alternative to standard Gaussian-elimination is to perform 'full pivoting'. This term is usually reserved for solution of numerical systems in a floating-point environment, referring to the choice of pivots so as to reduce rounding error. In this case, pivots can be chosen to be the 'simplest' elements to keep as much of the structure as possible. For example, the system has many non-polynomial entries (0's, 1's, $\times$'s and c_i's) which should be preserved by the pivoting operations. The rank of Υ_i is 13, and therefore 13 (full) pivoting operations yields a reduced system (Figure 7.5) with consistency conditions denoted as $\gamma_k,\ k = 1, \ldots, 10$, linear in the β_i, ρ_i and φ_i's. Note that row, but not column, interchanges have been done by the pivoting algorithm, and that the sparseness of the system has been preserved. The equations $\gamma_k = 0,\ k = 1, \ldots, 10$, are then solved for

$$\beta_{i6}, \beta_{i7}, \beta_{i8}, \beta_{i9}, \beta_{i10}, \beta_{i11}, \beta_{i12}, \beta_{i13}, \beta_{i15}, \text{ and } \beta_{i16},$$

$$\overline{\Upsilon}_i = \left(\begin{array}{ccccccccccccccc|c}
P_i^{0:1} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i1} \\
P_i^{0:1} & P_i^{0:1} & \times & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i2} \\
P_i^{0:1} & P_i^{0:1} & \times & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i3} \\
P_i^{0:1} & P_i^{0:1} & \times & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i4} \\
P_i^{0:1} & P_i^{0:1} & \times & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i5} \\
P_i^{0:2} & P_i^{0:2} & P_i^{0:2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i6} \\
P_i^{0:2} & P_i^{0:2} & \times & c_i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i7} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & c_i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i8} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i9} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & c_i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i10} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i11} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & c_i & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i12} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \beta_{i13} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \beta_{i14} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \beta_{i15} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \beta_{i16} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \beta_{i17} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \beta_{i18} \\
0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 1 & 0 & \rho_{i1} \\
0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 & 1 & \rho_{i2} \\
0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & 1 & 0 & 0 & 0 & 0 & 0 & \rho_{i3} \\
0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & 0 & 0 & 1 & 0 & 0 & 0 & \rho_{i4} \\
0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & 0 & 0 & 0 & 1 & 0 & 0 & \rho_{i5}
\end{array}\right)$$

Figure 7.4: 'Linear' $\boldsymbol{\alpha}_i$ system for eighth-order dense on the RK8(7)13M.

$$\overline{\Upsilon}_i = \left(\begin{array}{ccccccccccccccc|c}
0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 1 & 0 & \rho_{i1} \\
0 & 0 & 0 & 0 & \times & 0 & \times & \times & \times & 0 & 0 & 0 & 0 & \times & 1 & \times \cdot \varphi_{i1} + \rho_{i2} \\
P_i^{0:1} & P_i^{0:1} & \times & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i2} \\
P_i^{0:1} & P_i^{0:1} & \times & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i4} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{i11} \\
P_i^{0:1} & P_i^{0:1} & \times & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i5} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \beta_{i13} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \beta_{i14} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \beta_{i15} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \beta_{i16} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \beta_{i17} \\
P_i^{0:2} & P_i^{0:2} & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \beta_{i18} \\
P_i^{0:1} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{i1} \\
 & & & & & & & & & & & & & & & \gamma_1 \\
 & & & & & & & & & & & & & & & \gamma_2 \\
 & & & & & & & & & & & & & & & \gamma_3 \\
 & & & & & \ddots & & & & & & & & & & \gamma_4 \\
 & & & & & & \Large 0 & \ddots & & & & & & & & \gamma_5 \\
 & & & & & & & & & & & & & & & \gamma_6 \\
 & & & & & & & & & & & & & & & \gamma_7 \\
 & & & & & & & & & & & & & & & \gamma_8 \\
 & & & & & & & & & & & & & & & \gamma_9 \\
 & & & & & & & & & & & & & & & \gamma_{10}
\end{array}\right)$$

Figure 7.5: Eliminated 'linear' $\boldsymbol{\alpha}_i$ system for eighth-order dense on the RK8(7)13M.

in terms of

$$c_i, \beta_{14i}, \beta_{17i}, \beta_{18i}, \text{ and } \varphi_{ik}, \rho_{ik}, \; k = 1, \ldots, 5,$$

as free parameters. The structure of the resulting β_i's is indicated in Figure 7.6, where

$$\boldsymbol{\beta}_i = \begin{pmatrix} \varphi_{i1} \\ \varphi_{i2} \\ \varphi_{i3} \\ \varphi_{i4} \\ \varphi_{i5} \\ \times \cdot \varphi_{i1} + \varphi_{i1} c_i \\ \times \cdot \varphi_{i1} + \varphi_{i2} c_i \\ \times \cdot \varphi_{i1} + \varphi_{i3} c_i \\ \{\varphi_{i3}, \varphi_{i4}, \varphi_{i5}, (\beta_{i17} - \rho_{i1}), (\beta_{i18} - \rho_{i2})\} \\ \times \cdot \varphi_{i1} + \varphi_{i4} c_i \\ \{\varphi_{i3}, \varphi_{i4}, \varphi_{i5}, (\beta_{i17} - \rho_{i1}), (\beta_{i18} - \rho_{i2})\} \\ \times \cdot \varphi_{i1} + \varphi_{i5} c_i \\ \rho_{i3} + \{\varphi_{i3}, \varphi_{i4}, \varphi_{i5}, (\beta_{i17} - \rho_{i1}), (\beta_{i18} - \rho_{i2})\} \\ \beta_{i14} \\ \rho_{i4} + \{\varphi_{i3}, \varphi_{i4}, \varphi_{i5}, (\beta_{i17} - \rho_{i1}), (\beta_{i18} - \rho_{i2})\} \\ \rho_{i5} + \{\varphi_{i3}, \varphi_{i4}, \varphi_{i5}, (\beta_{i17} - \rho_{i1}), (\beta_{i18} - \rho_{i2})\} \\ \beta_{i17} \\ \beta_{i18} \end{pmatrix}$$

Figure 7.6: Parameterised 'β_i' vector for eighth-order dense on the RK8(7)13M.

terms in braces ($\{\ldots\}$) refer to a numerical (rational integer) linear combination of the quantities indicated. It should be noted that β_{i17} is linked to ρ_{i1} in these expressions, represented by $(\beta_{i17} - \rho_{i1})$ in the linear combination lists (likewise for β_{i18} and ρ_{i2}). For example,

$$\beta_{i16} = \rho_{i5} + \frac{213297882840269493 81\,(\beta_{i18} - \rho_{i2})}{5418335771025694720000} + \frac{11645535751997913\,(\beta_{i17} - \rho_{i1})}{2328191151612603200000}$$

$$+ \frac{3425559002934499161\,\varphi_{i5}}{12930536890854539264000} - \frac{130879780692057\,\varphi_{i4}}{717470308663360000} + \frac{4543802890512753\,\varphi_{i3}}{5637266710926400}.$$

Now forming the complete $\boldsymbol{\alpha}_i$ system (7.13), two 'α_i' are available for solution of $\widetilde{\Upsilon}_i^{\rho} \boldsymbol{\alpha}_i = \widetilde{\boldsymbol{\rho}}_i$ and $\widetilde{\Upsilon}_i^{\delta} \boldsymbol{\alpha}_i = \widetilde{\boldsymbol{\delta}}_i$. Appending the augmented form of system (7.12) to the bottom of the eliminated, consistent form of $\overline{\Upsilon}_i$ gives full-rank (15), and may then be eliminated by 15 full pivoting operations to give 4 further consistency conditions, $\overline{\gamma}_k = 0,\ k = 1, \ldots, 4$. These are solved for

$$\delta_{6i}, \delta_{7i}, \delta_{8i}, \text{ and } \rho_{6i},$$

in terms of

$$c_i, \rho_{7i}, \rho_{8i}, \beta_{14i}, \beta_{17i}, \beta_{18i}, \text{ and } \varphi_{ik}, \rho_{ik},\ k = 1, \ldots, 5. \tag{7.15}$$

The system $\widehat{\Upsilon}_i \boldsymbol{\alpha_i} = \widehat{\boldsymbol{v}}_i$ is then consistent for determination of all $\alpha_{ik},\ k = 1, \ldots, 15$, and the free parameters (7.15) are available for solution of the model.

7.3.4 Solution for eighth-order dense

Solution of the model now requires consideration of the $\beta^{(8)}$ matrix and $\overline{D}_i^{(8)}$ systems, of fundamental importance being to force rank-deficiency on the over-determined part of the $\boldsymbol{b}^*$ system, i.e.,

$$\beta^{(8)} = \begin{pmatrix} \beta_{15,1} & \cdots & \beta_{s^*,1} \\ \vdots & \ddots & \vdots \\ \beta_{15,18} & \cdots & \beta_{s^*,18} \end{pmatrix}.$$

Further to this, the over-determined part of the $\overline{A}_i^{(8)}$ systems, i.e.,

$$D_i^{(8)} \overline{\boldsymbol{a}}_i = \boldsymbol{\rho}_i,$$

must be made consistent for $i = 15, \ldots, s^*$, where

$$D_i^{(8)} = \begin{pmatrix} \varphi_{15,1} & \cdots & \varphi_{i-1,1} \\ \vdots & \ddots & \vdots \\ \varphi_{15,5} & \cdots & \varphi_{i-1,5} \\ \delta_{15,6} & \cdots & \delta_{i-1,6} \\ \delta_{15,7} & \cdots & \delta_{i-1,7} \\ \delta_{15,8} & \cdots & \delta_{i-1,8} \end{pmatrix},\ \overline{\boldsymbol{a}}_i = (a_{i15}, \ldots, a_{i,i-1})^T,$$

and

$$\boldsymbol{\rho}_i = (\rho_{i1}, \ldots, \rho_{i8})^T.$$

19-stage model

A 19-stage solution requires $r_\beta^{(8)} = 0$, i.e., $\boldsymbol{\beta}_i = 0$, $i = 15, \ldots, s^*$. In this case, examination of Figure 7.6 clearly requires

$$\beta_{i14} = \beta_{i17} = \beta_{i18} = 0, \ \varphi_{ik} = 0, \ k = 1, \ldots, 5,$$

and solution of five homogeneous conditions for ρ_{ik}, $k = 1, \ldots, 5$, coefficients of which are rational integers. This *square* system is non-singular, and the only solution is therefore

$$\rho_{ik} = 0, \ k = 1, \ldots, 5. \tag{7.16}$$

Note that for every $\delta_{ik} = 0$ (for some k), the corresponding ρ_{ik} *must* also be zero from consistency of any $\overline{D}_i^{(8)}$ system, and (7.16) was necessary in this case.

With $s^* = 19$ apparently viable, the $\overline{D}_i^{(8)}$ systems must now be considered, non-zero rows being formed by

$$\delta_{i6} = \frac{14\,[14\,\rho_{i8}\,(c_i + 10) - 3\,\rho_{i7}\,(4\,c_i^2 + 10\,c_i + 75)]}{15\,c_i\,(c_i - 1)\,(14\,c_i + 15)},$$

$$\delta_{i7} = \frac{2\,[28\,\rho_{i8}\,(7\,c_i + 15) - 3\,\rho_{i7}\,(c_i + 3)\,(14\,c_i + 75)]}{3\,c_i\,(c_i - 1)\,(14\,c_i + 15)},$$

$$\delta_{i8} = \frac{14\,\rho_{i8}\,(7\,c_i^2 + 18\,c_i + 30) - 15\,\rho_{i7}\,(14\,c_i^2 + 30\,c_i + 45)}{c_i\,(c_i - 1)\,(14\,c_i + 15)},$$

$$\rho_{i6} = \frac{14\,\{\rho_{i7}\,(6\,c_i + 18) - 7\,\rho_{i8}\}}{15\,(14\,c_i + 15)},$$

and ρ_{i7}, ρ_{i8} free. Consistency of the $\overline{A}_{15}^{(8)}$ system requires $\rho_{15k} = 0$, $k = 6, \ldots, 8$, and imposing $\rho_{15,7} = \rho_{15,8} = 0$ is sufficient. In this case $\delta_{15,6} = \delta_{15,7} = \delta_{15,8} = 0$, and

$$\overline{D}^{(8)}_{16} = \left(\begin{array}{c|c} 0 & \frac{14\{\rho_{16,7}(6\,c_{16}+18)-7\,\rho_{16,8}\}}{15(14\,c_{16}+15)} \\ 0 & \rho_{16,7} \\ 0 & \rho_{16,8} \end{array} \right),$$

consistency of which requires $\rho_{16,7} = \rho_{16,8} = 0$. Now $\delta_{16,6} = \delta_{16,7} = \delta_{16,8} = 0$, and it should be clear that $\boldsymbol{\delta}_i = \boldsymbol{\rho}_i = 0$, $i = 15, \ldots, 19$, is the only possible solution after subsequent consideration of $\overline{D}^{(8)}_{17}$, $\overline{D}^{(8)}_{18}$ and $\overline{D}^{(8)}_{19}$. Hence, $D^{(8)}_i$ has rank zero $\forall i$, and from duality of the $A^{(8)}_i$ and $W^{(7)}$ coefficient matrices, it transpires that $\overline{W}^{(7)}$ can never be consistent ($\forall \sigma$), and that no seventh-order solution can be possible using *any* number of the 19 stages of the eighth-order model. This clearly implies non-existence of the $p^* = 8$, $s^* = 19$ case. Such a solution must be avoided in the remaining work, and is characterised by the following theorem:

Theorem 7.3.1 (Dual Rank Theorem) For an order q, s^*-stage RK(N) model, and where a duality exists between $A^{(q)}_i$ and $W^{(p)}$ matrices for $p < q$, no order q problem solution is possible $\forall \sigma$ if $D^{(q)}_i$ does not have full row-rank *for some* $i \leq s^*$.

Proof. Consider that $A^{(q)}_i$ is in standard partially eliminated form

$$A^{(q)}_i = \left(\begin{array}{c|c} T_a & \Delta^{(q)}_i \\ \hline 0 & D^{(q)}_i \end{array} \right),$$

for $\Delta^{(q)}_i$ the matrix formed from columns $\boldsymbol{\zeta}^{d(q)}_k$, for $k = s^r + 1, \ \ldots, i-1$. By duality,

$$\overline{W}^{(p)} = \left(\begin{array}{c|c|c} T_a & \Delta^{(q)}_i & \\ \cline{1-2} 0 & D^{(q)}_i & \boldsymbol{p}_\sigma \end{array} \right),$$

where $\boldsymbol{p}_\sigma$ is a vector of polynomials in σ containing no zero entries by independence of the differing powers of σ from each quadrature equation of a continuous RK(N) process. Existence of a solution to $\overline{W}^{(p)}$ is now equivalent to there being at least one

i such that $\overline{W}^{(p)}$ is consistent (or can be made consistent). If $D_i^{(q)}$ does not have full row-rank for *any* i ($\leq s^*$), $\overline{W}^{(p)}$ will always have at least one consistency condition of the form $x = p_\sigma$ (for p_σ an element of $\boldsymbol{p}_\sigma$). Since x can contain no quantity that may be constrained as a function of σ, $\overline{W}^{(p)}$ will always be inconsistent (except for any particular σ satisfying $p_\sigma = 0$). In this case, no order p solution can exist $\forall\sigma$ (using any number of the s^* stages of the order q model), and hence $\overline{W}^{(q)}$ has no continuous solution by the sharing of order conditions.

$\square$

The eighth-order model considered here had been *apparently* solved in 19 stages by forcing $r_\beta^{(8)} = 0$ on the $M^{(8)}$ system (with $\boldsymbol{\beta}_i = 0$, $\forall i$), and consistency on all $\overline{D}_i^{(8)}$ systems (with $\boldsymbol{\delta}_i = \boldsymbol{\rho}_i = 0$, $\forall i$). Solution failure is due to inconsistency of $\overline{W}^{(8)}$ formed from $\overline{M}^{(8)}$ after addition of $n_q^{(8)} = 7$ quadrature equation rows, $\overline{X}^{(8)}$. Such a failure can only be possible if rows of $M^{(8)}$ are linearly dependent on rows of $X^{(8)}$. To illustrate this, the consistent, fully-eliminated equation system $\widehat{\Upsilon}_i\boldsymbol{\alpha}_i = \widehat{\boldsymbol{\upsilon}}_i$ (from Section 7.3.3) may now be back-substituted for α_{ik}, $k = 1, \ldots, 15$. The resulting expressions are of the following form:

$$\alpha_{ik} = \sum_{j=1}^{7} \lambda_j^k c_i^j, \; k = 1, \ldots, 15.$$

For example,

$$\alpha_{i1} \equiv Q_{i4} = -c_i \Big(71970741696\, c_i^4 - 146352554390\, c_i^3 + 99577595425\, c_i^2 \\ -25540221445\, c_i + 1885808445)\, /\, 915527268480,$$

and thus $\lambda_6^1 = \lambda_7^1 = 0$. Now, by substituting with

$$c_6 = \frac{3}{8}, c_7 = \frac{59}{400}, c_8 = \frac{93}{200}, c_9 = \frac{5490023248}{9719169821}, c_{10} = \frac{13}{20} \text{ and } c_{11} = \frac{30992876149296355}{33518267164510641},$$

it can be shown that α_{i1} evaluates to Q_{i4} for $i = 6, \ldots, 11$. An equivalent result holds for all α_{ik}, $k \in \{1, \ldots, 15\}$, and each row of $M^{(8)}$ is therefore dependent on the rows of $X^{(8)}$ in all but stages 12, 13 and 14 (where $c_{12} = c_{13} = c_{14} = 1$). Thus, $\overline{W}^{(8)}$ can be written in the following reduced form:

$$\overline{W}^{(8)} = \left(\begin{array}{cccccccccccccc|c}
\times & \times & \times & \times & \times & \times & 1 & 1 & 1 & c_{15} & c_{16} & c_{17} & c_{18} & c_{19} & \frac{\sigma}{2} \\
\times & \times & \times & \times & \times & \times & 1 & 1 & 1 & c_{15}^2 & c_{16}^2 & c_{17}^2 & c_{18}^2 & c_{19}^2 & \frac{\sigma^2}{3} \\
\times & \times & \times & \times & \times & \times & 1 & 1 & 1 & c_{15}^3 & c_{16}^3 & c_{17}^3 & c_{18}^3 & c_{19}^3 & \frac{\sigma^3}{4} \\
\times & \times & \times & \times & \times & \times & 1 & 1 & 1 & c_{15}^4 & c_{16}^4 & c_{17}^4 & c_{18}^4 & c_{19}^4 & \frac{\sigma^4}{5} \\
\times & \times & \times & \times & \times & \times & 1 & 1 & 1 & c_{15}^5 & c_{16}^5 & c_{17}^5 & c_{18}^5 & c_{19}^5 & \frac{\sigma^5}{6} \\
\times & \times & \times & \times & \times & \times & 1 & 1 & 1 & c_{15}^6 & c_{16}^6 & c_{17}^6 & c_{18}^6 & c_{19}^6 & \frac{\sigma^6}{7} \\
\times & \times & \times & \times & \times & \times & 1 & 1 & 1 & c_{15}^7 & c_{16}^7 & c_{17}^7 & c_{18}^7 & c_{19}^7 & \frac{\sigma^7}{8} \\
0 & 0 & 0 & 0 & 0 & 0 & \times & \times & 0 & 0 & 0 & 0 & 0 & 0 & {}^1P_\sigma \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 & 0 & {}^2P_\sigma \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {}^3P_\sigma \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {}^4P_\sigma \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {}^5P_\sigma \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {}^{25}P_\sigma
\end{array}\right),$$

with rank 9. Polynomials ${}^kP_\sigma$, $k = 3, \ldots, 25$, all contain a factor $\sigma(\sigma-1)^2$ (by continuity), but the system is inconsistent for $\sigma \in (0,1)$. Under the conditions of Theorem 7.3.1, such analysis will be unnecessary for the order q model in future. While it is, of course, possible for $\overline{W}^{(q)}$ to be inconsistent *without* the conditions of Theorem 7.3.1, it is not clear how this would naturally occur out of the solution strategy.

20-stage model

Now seeking $r_\beta^{(8)} = 1$, for maximum generality β must contain a single independent column or row. A column-rank of 1 may be forced by requiring:

$$\boldsymbol{\beta}_i = \lambda_i \boldsymbol{\beta}_j, \text{ for all } i \neq j \in \{15, \ldots, 20\}, \tag{7.17}$$

where $j \in \{15, \dots, 20\}$, and the λ_i are stage independent. This also implies a row-rank of 1 by linear algebra, and can easily be shown by a single step of Gaussian-elimination with β_{j1} as the pivot. A simple solution to (7.17) may now be proposed allowing analysis for a general i^{th} stage to be maintained. Examination of Figure 7.6 shows that coefficients of c_i in $\boldsymbol{\beta}_i$ are $\varphi_{ik},\ k = 1, \dots, 5$, only. Making the conditions

$$\varphi_{ik} = 0,\ k = 1, \dots, 5, \tag{7.18}$$

will therefore leave $\boldsymbol{\beta}_i$ stage independent provided none of the remaining free 'β_i' are specified to contain c_i parameters. It seems natural to leave the column comprised of $i = 15$ quantities in $\beta^{(8)}$ free, and (7.17) may be satisfied for $j = 15$ by also imposing $\lambda_{15} = 1$,

$$\rho_{ik} = \lambda_i \rho_{15k},\ k = 1, \dots, 5, \tag{7.19}$$

and

$$\beta_{i14} = \lambda_i \beta_{15,14},\ \beta_{i17} = \lambda_i \beta_{15,17},\ \beta_{i18} = \lambda_i \beta_{15,18}. \tag{7.20}$$

However, from consistency of $\overline{D}_{15}^{(8)}$, $\rho_{15k} = 0,\ k = 1, \dots, 5$, and (7.19) must require

$$\rho_{ik} = 0,\ k = 1, \dots, 5. \tag{7.21}$$

The natural consequence of conditions (7.18) is that seventh-order dense is now guaranteed in the minimum number of stages, as in this case $r_\beta^{(7)} = 0$. The 20-stage model proposed for eighth-order dense will therefore also yield a seventh-order formula using common function evaluations.

Consistency of $\overline{D}_{15}^{(8)}$ now requires $\rho_{15k} = 0,\ k = 6, \dots, 8$, so setting $\rho_{15,7} = \rho_{15,8} = 0$, $\rho_{15,6} = 0$ can be solved for $\beta_{15,14}$ in terms of $\beta_{15,17}$ and $\beta_{15,18}$. The first column of the $\overline{D}_i^{(8)}$ systems now have

$$\delta_{15,6} = \frac{27\ (2\,c_{15} - 1)\,\gamma}{29800846740641320960},\ \delta_{15,7} = \frac{27\ (7\,c_{15} - 3)\,\gamma}{41721185436897849344},$$

$$\delta_{15,8} = \frac{81\,(8\,c_{15} - 3)\,\gamma}{83442370873795698688},$$

where

$$\gamma = \frac{62899700234170992\,\beta_{15,17} + 3193957319545988875\,\beta_{15,18}}{(c_{15} - 1)\,c_{15}},$$

and additionally,

$$\rho_{16,6} = \frac{14\,\{6\,\rho_{16,7}\,(\,c_{16} + 3) - 7\,\rho_{16,8}\}}{15\,(14\,c_{16} + 15)}.$$

The $\overline{D}_{16}^{(8)}$ system may now be made consistent by considering the form after a single step of Gaussian-elimination. Note that because $\beta_{15,17}$ and $\beta_{15,18}$ occur only as a factor (γ) in $\delta_{15,6}$, $\delta_{15,7}$ and $\delta_{15,8}$, the ratio of any two elements of the first column of $\overline{D}_{16}^{(8)}$ will be a rational polynomial of c_{15}, only. In this way, consistency conditions arising from a single pivoting operation will be independent of $\beta_{15,17}$ and $\beta_{15,18}$. For more complicated models, this form of 'δ-dependency' will be extremely beneficial (see Chapter 8). If γ were zero, $\boldsymbol{\delta}_{15} = \mathbf{0}$ would result in a singular case similar to that of the 19-stage model (by Theorem 7.3.1), without having $\beta_{15,17} = \beta_{15,18} = 0$. After elimination of $\overline{D}_{16}^{(8)}$, two resulting consistency conditions are as follows:

$$\gamma_1 = -\frac{14\,\rho_{16,8}\,(7\,c_{15} - 3) - 3\,\rho_{16,7}\,(2\,c_{16} + 54\,c_{15} - 21)}{3\,(2\,c_{15} - 1)\,(14\,c_{16} + 15)},$$

and

$$\gamma_2 = \frac{2\,[\rho_{16,8}\,\{7\,(2\,c_{15} - 1)\,c_{16} + 43\,c_{15} - 18\} - 3\,\rho_{16,7}\,(8\,c_{15} - 3)\,(c_{16} + 3)]}{(2\,c_{15} - 1)\,(14\,c_{16} + 15)},$$

forming a homogeneous square system for $\rho_{16,7}$ and $\rho_{16,8}$ which is non-singular for $c_{15} \neq c_{16}$ giving $\rho_{16,7} = \rho_{16,8} = 0$, or

$$c_{16} = c_{15},\;\; \rho_{16,7} = \frac{2\,\rho_{16,8}\,(7\,c_{15} - 3)}{3\,(8\,c_{15} - 3)},$$

otherwise (with $\rho_{16,8}$ free). The remaining analysis must now be done for two distinct models:

1. Degenerate case ($c_{16} = c_{15}$). Considering $\overline{D}_{17}^{(8)}$, two pivoting operations yields a single consistency condition, solution of which gives

$$\rho_{17,7} = \frac{2\,\rho_{8,17}\,(42\,c_{17} - 49\,c_{15} + 3)}{3\,(46\,c_{17} - 54\,c_{15} + 3)}.$$

2. Non-degenerate case (c_{16} free). Again, two pivoting operations on $\overline{D}_{17}^{(8)}$ yields a consistency condition:

$$\gamma = \frac{3\,\rho_{17,7}\,(3 - 8\,c_{17}) + 2\,\rho_{17,8}\,(7\,c_{17} - 3)}{14\,c_{17} + 15},$$

which may be solved with

$$\rho_{17,7} = \frac{2\,\rho_{17,8}\,(7\,c_{17} - 3)}{3\,(8\,c_{17} - 3)}.$$

Finally, $\overline{D}_i^{(8)}$, $i = 18, \ldots, 20$, are already consistent, with $\overline{D}_{19}^{(8)}$ a single row under-determined, and $\overline{D}_{20}^{(8)}$ two rows under-determined. Either of the two solutions above are valid, the large number of free parameters (the same number in each case) make a choice of either one unlikely to affect the performance of a final optimised process. Arbitrarily, the non-degenerate case has been chosen based on it having an extra free node (c_{16}).

With all systems now consistent, it is only left to determine the set of free parameters and perform an optimisation. The numerical part of the $\overline{A}_i^{(8)}$ system has $r_a^{(8)} = 8$, and so only 8 of a_{ij}, $j = 6, \ldots, i-1$, can be determined. A rank-deficiency exists in column 8, leaving a_{i13} free in every stage. As $r_\beta^{(7)} = 0$, the number of stages required for the embedded seventh-order model *would* be 16 ($s^b + r_b^{(7)} + n_q^{(7)} - 1$). However, the duality of $A_i^{(8)}$ and $M^{(7)}$ will also yield b_{13}^* as free when solving $\overline{W}^{(7)}$. Thus, extra stages $i = 15, 16$ and 17 will be required for the seventh-order dense, using common evaluations to the eighth-order model, if required. Free parameters for the complete (non-degenerate) eighth-order model are therefore:

$$\begin{gathered} c_{15}, c_{16}, c_{17}, c_{18}, c_{19}, c_{20},\ \ \lambda_{16}, \lambda_{17}, \lambda_{18}, \lambda_{19}, \lambda_{20}, \\ \beta_{15,17}, \beta_{15,18},\ \ \rho_{17,8}, \rho_{18,7}, \rho_{18,8}, \rho_{19,7}, \rho_{19,8}, \rho_{20,7}, \rho_{20,8}, \\ a_{15,13}, a_{16,13}, a_{17,13}, a_{18,13}, a_{19,13}, a_{19,18}, a_{20,13}, a_{20,18}, a_{20,19}. \end{gathered} \tag{7.22}$$

7.3.5 Solution for (dedicated) seventh-order dense

As an alternative to the seventh-order dense formulae embedded within the model of the previous section, a dedicated model may be derived by consideration of the $\overline{W}^{(7)}$ and $\overline{A}_i^{(7)}$ systems. This may be necessary due to the previous model being over-specified — extra free parameters are possible by the specific treatment here.

From specification of the system vectors in Section 7.3.1, $\overline{A}_i^{(7)}$ may be formed and eliminated with $r_a^{(7)} = 7$ pivoting operations. This gives one over-determined row ($n_\delta = 1$) with

$$\begin{aligned} \boldsymbol{\zeta}_i^{\delta(7)} = \Bigg(& c_i^5 - \frac{5\,c_i^4}{2} + \frac{15\,c_i^3}{7} - \frac{5\,c_i^2}{7} + \frac{c_i}{14} + \frac{836210130911807901914 5\,\alpha_{i3}}{44332726080029997917798 4} \\ & - \frac{238702508366861272 9\,\alpha_{i2}}{230899615000156239155 2} - \frac{327055577322682555\,\alpha_{i1}}{49137800180685489} \Bigg), \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{\zeta}_i^{\rho(7)} = \Bigg(& \frac{c_i^6}{6} - \frac{c_i^5}{2} + \frac{15\,c_i^4}{28} - \frac{5\,c_i^3}{21} + \frac{c_i^2}{28} + \frac{8362101309118079019145\,\alpha_{i9}}{443327260800299979177984} \\ & - \frac{2387025083668612729\,\alpha_{i7}}{2308996150001562391552} - \frac{327055577322682555\,\alpha_{i5}}{49137800180685489} + \alpha_{i4} \\ & + \frac{21285\,\alpha_{i3}}{7340032} - \frac{45\,\alpha_{i2}}{32768} - \frac{5\,\alpha_{i1}}{2} \Bigg). \end{aligned}$$

The non-quadrature $\boldsymbol{b}^*$ system, $\overline{M}^{(7)}$ was considered in Section 7.3.2, having $n_\beta^{(7)} = 5$ over-determined rows specified by $\boldsymbol{\zeta}_i^{\beta(7)}$. A rank-zero $\beta^{(7)}$ matrix is now sought and these expressions can be zeroed with no need for parameters $\beta_{ik},\ k = 1, \ldots, 5$. The

α_i's may now be used for this purpose and to parameterise for ρ_{i1}, δ_{i1} with

$$ {}^k\zeta_i^{\beta(7)} = 0,\ k = 1, \ldots, 5,\ \delta_{i1} = {}^1\zeta_i^{\delta(7)},\ \rho_{i1} = {}^1\zeta_i^{\rho(7)}, $$

forming a single linear system

$$ \widehat{\Upsilon}_i \boldsymbol{\alpha}_i = \widehat{\boldsymbol{v}}_i,\ \text{where } \boldsymbol{\alpha}_i = (\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, \alpha_{i4}, \alpha_{i5}, \alpha_{i7}, \alpha_{i9})^T, $$

noting that the simplicity of this case allows a complete treatment with no need to split the 'linear' and 'nonlinear' rows. The $\widehat{\Upsilon}_i$ system has rank 7 and therefore specifies the 7 α_i's above, leaving δ_{i1} and ρ_{i1} free. It is now only required to force consistency on $\overline{D}_{15}^{(7)}$ with $\rho_{15,1} = 0$, and the model is complete. The $A_i^{(7)}$ matrix is rank-deficient in columns 6 and 8 ($r_a^{(7)} = 7$), leaving a_{i11} and a_{i13} free for each stage. $\overline{D}_{16}^{(7)}$ is already consistent, and $\overline{D}_{17}^{(7)}$ is one row under-determined leaving $a_{17,16}$ free. Finally, the seventeen-stage model has b_{13}^* free (see Section 7.3.4), and the total list of free parameters is as follows:

$$ \begin{gathered} b_{13}^*,\ c_{15}, c_{16}, c_{17},\ \delta_{15,1}, \delta_{16,1}, \delta_{17,1},\ \rho_{16,1}, \rho_{17,1}, \\ a_{15,11}, a_{15,13}, a_{16,11}, a_{16,13}, a_{17,11}, a_{17,13}, a_{17,16}. \end{gathered} \tag{7.23} $$

Examination of the free parameters from the eighth-order case (7.22) shows 12 relating to the first three extra stages (including b_{13}^*). Thus, by reducing requirements of the $\overline{A}_i$ system, three extra degrees of freedom have been gained in this model. Whether this is significant can only be gauged from the degree of improvement in truncation error properties possible using the extra parameters in an optimisation process.

7.3.6 Choice of free parameters

The standard procedure for optimisation is now to pass all (now consistent) systems to Fortran 90 for determination of the method tableau given all free parameters. This is done by back-substituting the $\widehat{\Upsilon}_i \boldsymbol{\alpha}_i = \widehat{\boldsymbol{v}}_i$ system for the α_i's, allowing all

'$\boldsymbol{\zeta}_i$' vectors to be calculated and thus (partially eliminated) $\boldsymbol{b}^*$ and $\boldsymbol{a}_i$ systems to be formed and solved. This clearly requires matrices T_a and T_b to also be available from Fortran. Having coded the method coefficients as a function of the free parameters, this can then be used in an optimisation procedure for minimising some norm of local truncation error properties. For this purpose, ".tab" and ".rd" 'reduction data' files can be created with the following RK-AID command file:

```
title 87_#1
!
[s] order :10, error terms #2:, split
!
b[s](2:5) = 0 & Q(3:s, 1;2) = 0 & Q(6:s, 3) = 0
a(4:s, 2) = 0 & a(6:s, 3) = 0
!
use Q's
equation mode, full reduction
```

where '`#1`'/'`#2`' should be replaced by 7/8 and 8/9 in the seventh- and eighth-order cases, respectively. This will specify an independent set of eighth-order, ninth-order, and tenth-order truncation error coefficients for the seventh- and eighth-order models. The number of independent terms i_p / n_p, where n_p is the total number of terms in order p from Table 4.2, are given in the following table for $p^* = 7$ and $p^* = 8$:

p^*	i_8 / 115	i_9 / 286	i_{10} / 719
7	16	52	148
8	0	32	104

These error terms are too numerous to show here in any detail. However, the process of forming one, two, or infinity norms of principal, secondary and tertiary terms (for the seventh-order case only) can be automated having the resulting reduction information available in a form easily read from Fortran 90. The relevant files will be named "*ts_87_7.rd*", *ts_87_7.tab*", "*ts_87_8.rd*" and "*ts_87_8.tab*" for the two runs. As a simple example, reduction information in "*ts_87_7.rd*" relating to principal terms, only, may be represented as in Table 7.2. This information is sufficient for the formation of one-, two-, or infinity-norms of the eighth-order set. For this purpose, the independent terms may be written in their standard 'abc' form or in the fully reduced (and 'split')

m	τ		Σ	Σ^2	∞
29	8	1	2472	340992	315
10	8	12	36	156	6
11	8	14	96	1026	15
19	8	15	456	16476	60
6	8	46	10	22	3
5	8	51	18	78	6
5	8	52	18	78	6
11	8	53	96	1026	15
3	8	86	5	11	3
3	8	87	5	11	3
5	8	89	18	78	6
2	8	90	2	2	1
3	8	108	5	11	3
1	8	109	1	1	1
1	8	110	1	1	1
1	8	115	1	1	1

Table 7.2: Principal ".rd" data for the RK8(7)13MD^7 cases.

form as output by RK-AID, this being shown in Figure 7.7. In either case, the two-norm of all 115 eighth-order terms can now be written

$$\begin{aligned}||\boldsymbol{\tau}^{(8)*}||_2 = \Big\{ & 340992\,\tau_1^{(8)*} + 156\,\tau_{12}^{(8)*} + 1026\,\tau_{14}^{(8)*} + 16476\,\tau_{15}^{(8)*} + 22\,\tau_{46}^{(8)*} + 78\,\tau_{51}^{(8)*} \\ & +78\,\tau_{52}^{(8)*} + 1026\,\tau_{53}^{(8)*} + 11\,\tau_{86}^{(8)*} + 11\,\tau_{87}^{(8)*} + 78\,\tau_{89}^{(8)*} + 2\,\tau_{90}^{(8)*} \\ & +11\,\tau_{108}^{(8)*} + \tau_{109}^{(8)*} + \tau_{110}^{(8)*} + \tau_{115}^{(8)*} \Big\}^{\frac{1}{2}},\end{aligned}$$

noting that the quantity of usual interest in this case would be $\sigma^8||\boldsymbol{\tau}^{(8)*}||_2$. The same information can clearly be used for formation of linear deviation norms.

A single (dedicated) seventh-order and three eighth-order formulae will now be specified by GLOBS optimisations, first considering a practical problem. As perhaps expected, the high-order nature of the simple polynomial interpolation can introduce considerable rounding error into the dense output calculations if not properly safe-

$$\tau_1^{(8)*} = \tfrac{1}{5040}\sum_{i=6}^{s^*} b_i^* c_i^7/\sigma^7 - \frac{1}{8!}$$

$$\tau_{12}^{(8)*} = \tfrac{1}{48}\sum_{i=6}^{s^*} b_i^* c_i^2 Q_{i4}/\sigma^7 + 21\tau_1^{(8)*}$$

$$\tau_{14}^{(8)*} = \tfrac{1}{120}\sum_{i=6}^{s^*} b_i^* c_i Q_{i5}/\sigma^7 + 7\tau_1^{(8)*}$$

$$\tau_{15}^{(8)*} = \tfrac{1}{720}\sum_{i=6}^{s^*} b_i^* Q_{i6}/\sigma^7 + \tau_1^{(8)*}$$

$$\tau_{46}^{(8)*} = \tfrac{1}{12}\{Q_{4,3}\sum_{i=6}^{s^*} b_i^* c_i^2 a_{i4} + Q_{5,3}\sum_{i=6}^{s^*} b_i^* c_i^2 a_{i5}\}/\sigma^7 + \tau_{12}^{(8)*}$$

$$\tau_{51}^{(8)*} = \tfrac{1}{24}\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} Q_{j4}/\sigma^7 + \tau_{14}^{(8)*}$$

$$\tau_{52}^{(8)*} = \tfrac{1}{24}\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j Q_{j4}/\sigma^7 + 6\tau_{15}^{(8)*}$$

$$\tau_{53}^{(8)*} = \tfrac{1}{120}\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} Q_{j5}/\sigma^7 + \tau_{15}^{(8)*}$$

$$\tau_{86}^{(8)*} = \tfrac{1}{6}\{Q_{4,3}\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} a_{j4} + Q_{5,3}\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} a_{j5}\}/\sigma^7 + \tau_{51}^{(8)*}$$

$$\tau_{87}^{(8)*} = \tfrac{1}{6}\{Q_{4,3}\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j a_{j4} + Q_{5,3}\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j a_{j5}\}/\sigma^7 + \tau_{52}^{(8)*}$$

$$\tau_{89}^{(8)*} = \tfrac{1}{24}\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} Q_{k4}/\sigma^7 + \tau_{53}^{(8)*}$$

$$\tau_{90}^{(8)*} = \tfrac{1}{2}Q_{2,1}a_{3,2}\{a_{4,3}\sum_{i=6}^{s^*} b_i^* c_i^2 a_{i4} + a_{5,3}\sum_{i=6}^{s^*} b_i^* c_i^2 a_{i5}\}/\sigma^7 + \tau_{46}^{(8)*}$$

$$\tau_{108}^{(8)*} = \tfrac{1}{6}\{Q_{4,3}\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k4} + Q_{5,3}\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k5}\}/\sigma^7 + \tau_{89}^{(8)*}$$

$$\tau_{109}^{(8)*} = Q_{2,1}a_{3,2}\{a_{4,3}\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} a_{j4} + a_{5,3}\sum_{i=7}^{s^*} b_i^* c_i \sum_{j=6}^{i-1} a_{ij} a_{j5}\}/\sigma^7 + \tau_{86}^{(8)*}$$

$$\tau_{110}^{(8)*} = Q_{2,1}a_{3,2}\{a_{4,3}\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j a_{j4} + a_{5,3}\sum_{i=7}^{s^*} b_i^* \sum_{j=6}^{i-1} a_{ij} c_j a_{j5}\}/\sigma^7 + \tau_{87}^{(8)*}$$

$$\tau_{115}^{(8)*} = Q_{2,1}a_{3,2}\{a_{4,3}\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k4} + a_{5,3}\sum_{i=8}^{s^*} b_i^* \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k5}\}/\sigma^7 + \tau_{108}^{(8)*}$$

Figure 7.7: Independent principal error terms for any RK8(7)13MD^7.

guarded against. Such high-precision work really needs to be accompanied by high numerical accuracy of the computational platform, but compromises can be made to reduce this requirement. The usual recommendation for small (absolute) parameter size should be strictly maintained, especially as regards the 'b^*' polynomial coefficients, but above this more drastic measures may be taken. The third of the following list of process specifications contains such a consideration:

1. $\underline{87\mathrm{D}_{17}^{7}}$ *(dedicated seventh-order model).* This will be based upon an optimisation of

$$f(\boldsymbol{x}) = N^{(8)} \equiv \int_0^1 \sigma^8 \|\boldsymbol{\tau}^{(8)*}\|_2 \, d\sigma,$$

where $\boldsymbol{\tau}^{(8)*}$ is the vector of $\tau_i^{(8)*}$, $i = 1, \ldots, 115$, and

$$\boldsymbol{x} = \{p_3, p_4, p_5, p_6,\ c_{15}, c_{16}, c_{17},\ \delta_{15,1}, \delta_{16,1}, \delta_{17,1},\ \rho_{16,1}, \rho_{17,1},$$
$$a_{15,11}, a_{15,13}, a_{16,11}, a_{16,13}, a_{17,11}, a_{17,13}, a_{17,16}\}.$$

Comparison of the above with (7.23) shows that the free b_{13}^* has been split into its remaining polynomial coefficients after satisfying continuity requirements (with $\widehat{b}_{13} = \frac{1}{4}$), which demand the form

$$\begin{aligned} b_{13}^* = {} & p_6\sigma^6 + p_5\sigma^5 + p_4\sigma^4 + p_3\sigma^3 - \\ & \frac{2(10p_6 + 8p_5 + 6p_4 + 4p_3 + 1)\sigma^2 - (16p_6 + 12p_5 + 8p_4 + 4p_3 + 3)\sigma}{4}, \end{aligned} \tag{7.24}$$

assuming 'natural' sixth-degree polynomials.

2. $\underline{87\mathrm{D}_{20}^{8}\mathrm{L}}$ *(linear deviation model).* Here, a 20-stage formula with linearised principal truncation error terms is derived by optimisation of the objective function:

$$f(\boldsymbol{x}) = N_l^{(9)} \equiv \int_0^1 \|\boldsymbol{D}^{(9)*}\|_2 \, d\sigma,$$

where $\boldsymbol{D}^{(9)*}$ is the vector composed of elements $D_i^{(9)*}$, $i = 1, \ldots, 286$, and $\boldsymbol{x}$ is the parameter list in (7.22).

3. $\underline{87\mathrm{D}^8_{20}\mathrm{L}_2}$ *(alternative linear deviation model).* This formula is to be based on the same objective function as the $87\mathrm{D}^8_{20}\mathrm{L}$ case, but is constrained by numerical stability requirements, and uses the conditions

$$a_{18,13} = a_{19,13} = a_{19,18} = a_{20,13} = a_{20,18} = a_{20,19} = 0,$$

to reduce possible rounding effects in the higher stages. In this case,

$$\boldsymbol{x} = \{c_{15}, c_{16}, c_{17}, c_{18}, c_{19}, c_{20},\ \lambda_{16}, \lambda_{17}, \lambda_{18}, \lambda_{19}, \lambda_{20},\ \beta_{15,17}, \beta_{15,18},$$
$$\rho_{17,8}, \rho_{18,7}, \rho_{18,8}, \rho_{19,7}, \rho_{19,8}, \rho_{20,7}, \rho_{20,8},\ a_{15,13}, a_{16,13}, a_{17,13}\}.$$

4. $\underline{87\mathrm{D}^8_{20}\mathrm{M}}$ *(minimised local error model).* This formula is sought for comparison with the linear deviation models above, in order to check the validity of predicted continuous global error behaviour. The objective function to be considered is therefore:

$$f(\boldsymbol{x}) = N_m^{(9)} \equiv \int_0^1 \sigma^9 \|\boldsymbol{\tau}^{(9)*}\|_2 \ d\sigma,$$

where $\boldsymbol{\tau}^{(9)*}$ is the vector of $\tau_i^{(9)*},\ i = 1, \ldots, 286$, and $\boldsymbol{x}$ is the parameter list in (7.22).

In all cases, secondary (linear deviation) and method parameter norms were also considered (see Section 5.7.2).

The three eighth-order models allow a set of b^*'s to be derived using the first seventeen stages to give an (embedded) seventh-order formula with b^*_{13} as a free parameter. This must take the form (7.24), leaving $p_i,\ i = 3, \ldots, 6$, free for an additional optimisation having specified all other parameters. These polynomial coefficients are chosen as in the $87\mathrm{D}^7_{17}$ case, but as all truncation error terms will be linear in the 'p_i', principal and secondary terms may be minimised exactly using linear algebra if the alternative quantity

$$(\overline{N}^{(8)})^2 = \int_0^1 \sigma^8 \|\boldsymbol{\tau}^{(8)*}\|_2^2 \ d\sigma,$$

is used for the eighth-order truncation error term (double) average. Similarly, a norm of ninth-order linear deviation terms $(\overline{N}_l^{(9)})^2$ can be considered. In this way, the truncation error norms can be partially differentiated with respect to $p_i,\ i = 3, \ldots, 6$, and concurrently solved to zero as a linear system for the p_i. Such systems have been found to be ill-conditioned, thus MACSYMA has been used for minimisation of $(\overline{N}^{(8)})^2$, with additional components for $(\overline{N}_l^{(9)})^2$, and a b_i^* coefficient norm.

Free parameters for the four models now follow, where eighth-order formulae also specify polynomial coefficients for b_{13}^* relating to the embedded seventh-order model. No provision has been made for possible 'easy numbers' formulae — all free parameters being determined approximately in Fortran 90 and then replaced by an accurate continued fraction representation. This typically requires rationals with numerator and denominator comprising of 3 or 4 digits, though some are considerably larger, and have arisen from optimisation over *equivalent* sets of parameters. It is enough to say that the specific formulation just presented has only been possible following an improvement in the MACSYMA implementation available since the work began. The original model was therefore 'broken down' into much smaller elements to enable solution, resulting in a different, but entirely equivalent parameter set from the more automated treatment here. This is unfortunate as the specifications have become 'messy' in some cases (the free δ's and β's, for example).

1. $\underline{87\mathrm{D}_{17}^{7}}$. This formula is presented in Baker, et al. [66].

$$p_3 = \frac{23}{8}, p_4 = -\frac{1}{55}, p_5 = \frac{9}{4}, p_6 = -\frac{5}{2}, c_{15} = \frac{10}{39}, c_{16} = \frac{29}{39}, c_{17} = \frac{19}{33}, \delta_{15,1} = -\frac{5}{3808},$$

$$\delta_{16,1} = -\frac{17865688038598544210339188318872 5}{109180844900385004461549764933161704},$$

$$\delta_{17,1} = -\frac{1366005955588823587443681757 55}{29652742463056330386312568781757},$$

$$\rho_{16,1} = \frac{5}{38864}, \rho_{17,1} = \frac{15}{2947}, a_{15,11} = \frac{1}{40}, a_{15,13} = \frac{2}{29},$$

$$a_{16,11} = \frac{1}{275}, a_{16,13} = -\frac{1}{32}, a_{17,11} = \frac{31}{54}, a_{17,13} = \frac{2248}{1249}, a_{17,16} = -\frac{391}{237}.$$

2. $\underline{87\mathrm{D}^{8(7)}_{20(17)}\mathrm{L}}$. The $87\mathrm{D}^{8}_{20}\mathrm{L}$ here is contained in [66] as the RK8(7)13M '8DL'.

$$c_{15} = \frac{1}{4}, c_{16} = \frac{1}{2}, c_{17} = \frac{3}{7}, c_{18} = \frac{16}{25}, c_{19} = \frac{37}{53}, c_{20} = \frac{17}{24},$$

$$\lambda_{16} = -\frac{14}{9}, \lambda_{17} = -\frac{1}{282}, \lambda_{18} = \frac{193}{61572}, \lambda_{19} = \frac{511}{596}, \lambda_{20} = \frac{403}{441},$$

$$\beta_{15,17} = \frac{931276460645041280}{37043992206662078601}, \beta_{15,18} = 0,$$

$$\rho_{17,8} = \frac{5}{16016}, \rho_{18,7} = \frac{157001}{12595744}, \rho_{18,8} = \frac{473675}{25191488}, \rho_{19,7} = -\frac{3154409}{17756522},$$

$$\rho_{19,8} = -\frac{11517393}{35513044}, \rho_{20,7} = -\frac{720950}{15135253}, \rho_{20,8} = -\frac{372765}{4324358},$$

$$a_{15,13} = \frac{90}{487}, a_{16,13} = -\frac{58}{197}, a_{17,13} = \frac{3}{242}, a_{18,13} = -\frac{27}{152}, a_{19,13} = -\frac{57}{28},$$

$$a_{19,18} = -\frac{1495}{718}, a_{20,13} = -\frac{27}{94}, a_{20,18} = -\frac{99}{184}, a_{20,19} = -\frac{1}{1006},$$

and free b^*_{13} polynomial coefficients for the embedded seventh-order dense:

$$p_3 = \frac{83185196983}{1123786504}, p_4 = -\frac{212670633025}{1438733731}, p_5 = \frac{137984619547}{1016847665}, p_6 = -\frac{131270741929}{2838381017}.$$

3. $\underline{87\mathrm{D}^{8(7)}_{20(17)}\mathrm{L}_2}$.

$$c_{15} = \frac{1}{4}, c_{16} = \frac{7}{12}, c_{17} = \frac{23}{58}, c_{18} = \frac{64}{101}, c_{19} = \frac{219}{322}, c_{20} = \frac{377}{565},$$

$$\lambda_{16} = -\frac{331}{78}, \lambda_{17} = -\frac{276}{4441}, \lambda_{18} = \frac{3}{24325}, \lambda_{19} = \frac{446}{2697}, \lambda_{20} = \frac{247}{501},$$

$$\beta_{15,17} = \frac{232819115161260320}{9871321705500210057}, \beta_{15,18} = 0,$$

$$\rho_{17,8} = \frac{50}{91931}, \rho_{18,7} = \frac{536747}{17872512}, \rho_{18,8} = \frac{5129347}{107235072}, \rho_{19,7} = -\frac{9641}{55772},$$

$$\rho_{19,8} = -\frac{620069}{1952020}, \rho_{20,7} = -\frac{133174}{5474301}, \rho_{20,8} = -\frac{90222}{1824767},$$

$$a_{15,13} = \frac{46}{245}, a_{16,13} = -\frac{208}{237}, a_{17,13} = \frac{11}{225},$$

and

$$p_3 = \frac{7329198128}{90718581}, p_4 = -\frac{317922252337}{2008385941}, p_5 = \frac{708700166655}{4934598118}, p_6 = -\frac{708212992779}{14581109260}.$$

4. $\underline{87\mathrm{D}^{8(7)}_{20(17)}\mathrm{M}}$. The $87\mathrm{D}^{8}_{20}\mathrm{M}$ also contained in [66] as the RK8(7)13M '8DM'.

$$c_{15} = \frac{133}{547}, c_{16} = \frac{203}{521}, c_{17} = \frac{157}{433}, c_{18} = \frac{667}{1022}, c_{19} = \frac{127}{200}, c_{20} = \frac{103}{157},$$

$$\lambda_{16} = -\frac{1047}{50}, \lambda_{17} = -\frac{24}{5045}, \lambda_{18} = \frac{215}{178}, \lambda_{19} = \frac{3}{755}, \lambda_{20} = \frac{537}{307},$$

$$\beta_{15,17} = \frac{23809448351549976952466135}{339078500232180977050612356}, \beta_{15,18} = -\frac{12}{16897},$$

$$\rho_{17,8} = \frac{7955}{1079218}, \rho_{18,7} = \frac{9198724}{283810501}, \rho_{18,8} = \frac{938123156}{17880061563}, \rho_{19,7} = -\frac{145695467}{1219347066},$$

$$\rho_{19,8} = -\frac{54282155}{232256584}, \rho_{20,7} = -\frac{3750707}{91205676}, \rho_{20,8} = -\frac{10940537}{141875496},$$

$$a_{15,13} = -\frac{19}{1507}, a_{16,13} = \frac{89}{1487}, a_{17,13} = -\frac{35}{479}, a_{18,13} = -\frac{68}{915}, a_{19,13} = -\frac{486}{1069},$$

$$a_{19,18} = -\frac{253}{267}, a_{20,13} = -\frac{116}{2899}, a_{20,18} = -\frac{344}{1185}, a_{20,19} = -\frac{12}{1897},$$

and

$$p_3 = \frac{603142726207}{8845893645}, p_4 = -\frac{229394900016}{1748636767}, p_5 = \frac{239523364459}{2029248159}, p_6 = -\frac{85253728740}{2138166601}.$$

Error norms for the dedicated seventh-order formula, three eighth-order formulae, and the three seventh-order processes embedded within them are given accurate to two decimal places in Table 7.3. These were calculated using the following RK-AID commands for each method:

```
check #method#
[s] order #1, using 87_#2
2 norm, linear deviation
```

where '#1' is replaced by the order required, and '#2' is 7 or 8 to employ the ".rd" files produced by the model batch-file in Section 7.3.6 — this is not necessary, but can be used for speed as only an independent set is evaluated. Error norms can,

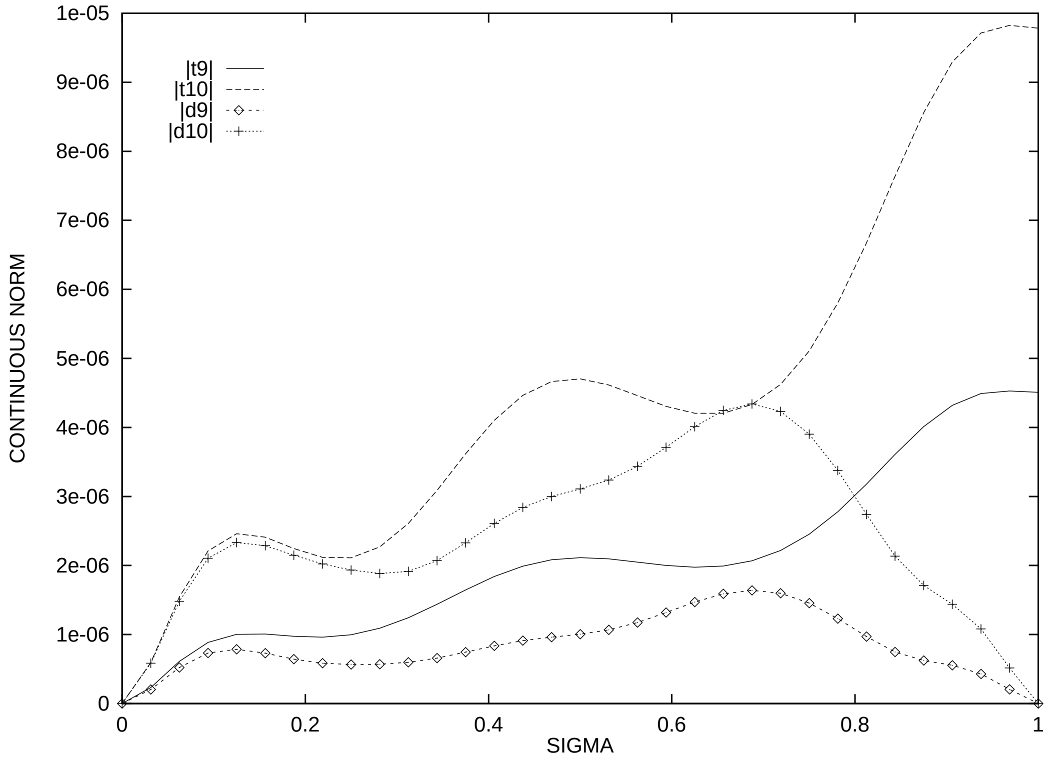


Figure 7.8: Principal/secondary error and linear deviation norms for the $87D^{8}_{20}L_2$.

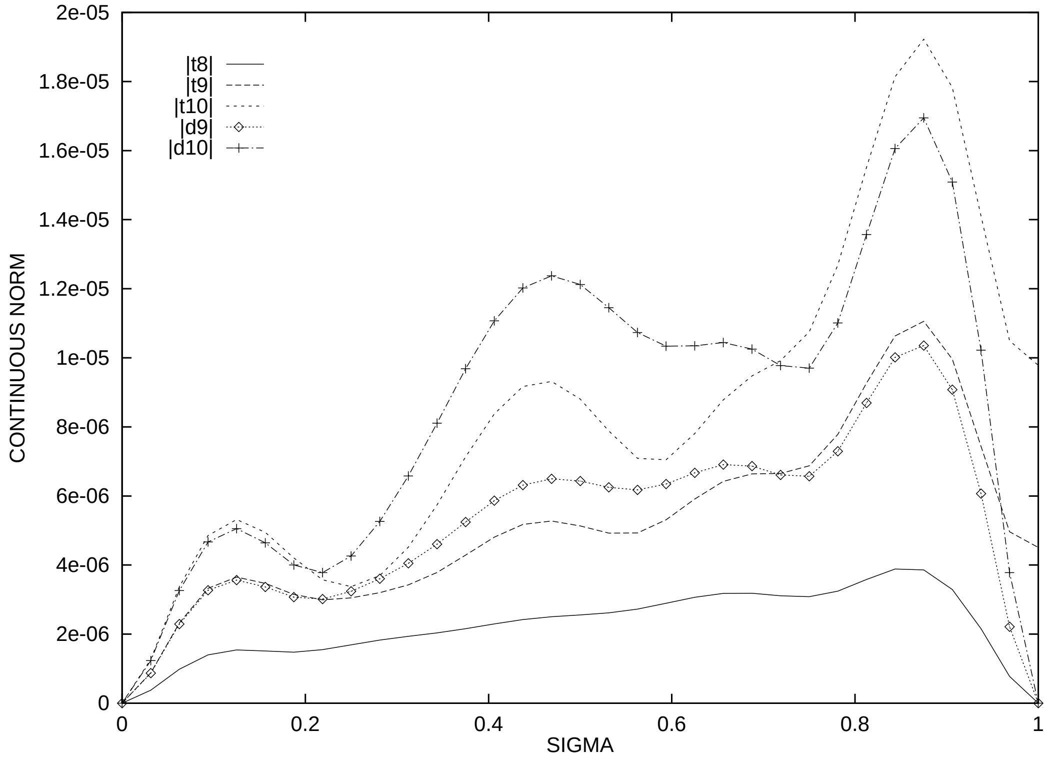


Figure 7.9: Principal to tertiary error and linear deviation norms for the $87D^{7}_{17}L_2$.

in fact, be displayed for two orders in each run, and so '`#1`' can be replaced by a range, e.g., '`9:10`', for the principal and secondary norms of an eighth-order process. Note that, of the seventh-order formulae, the general model has yielded the smallest principal error norm, but that of the 17-stage embedding within the $87\mathrm{D}^{8(7)}_{20(17)}\mathrm{L}_2$ is only *slightly* larger, and the three extra degrees of freedom for the dedicated model appear to have been negligible. Continuous principal and secondary truncation error and linear deviation norms for the $87\mathrm{D}^8_{20}\mathrm{L}_2$ and $87\mathrm{D}^7_{17}\mathrm{L}_2$, including tertiary terms for the latter, are plotted in Figures 7.8 and 7.9. Similar plots for the $87\mathrm{D}^8_{20}\mathrm{L}$ and $87\mathrm{D}^8_{20}\mathrm{M}$ are contained in Baker, et al. [66]. Files containing data for such plots are produced automatically by RK-AID when calculating the error norms in Table 7.3.

	$N^{(8)}$	$N_l^{(9)}$	$N_l^{(10)}$	$N_m^{(9)}$	$N_m^{(10)}$
$87\mathrm{D}^7_{17}$	2.27×10^{-6}	5.08×10^{-6}	8.28×10^{-6}	4.76×10^{-6}	7.09×10^{-6}
$87\mathrm{D}^7_{17}\mathrm{L}$	2.42×10^{-6}	5.68×10^{-6}	9.09×10^{-6}	5.54×10^{-6}	8.65×10^{-6}
$87\mathrm{D}^7_{17}\mathrm{L}_2$	2.28×10^{-6}	5.36×10^{-6}	8.68×10^{-6}	5.28×10^{-6}	8.42×10^{-6}
$87\mathrm{D}^7_{17}\mathrm{M}$	3.26×10^{-6}	7.51×10^{-6}	1.13×10^{-5}	7.40×10^{-6}	1.07×10^{-5}
$87\mathrm{D}^8_{20}\mathrm{L}$	0	5.33×10^{-7}	1.52×10^{-6}	2.29×10^{-6}	4.97×10^{-6}
$87\mathrm{D}^8_{20}\mathrm{L}_2$	0	8.48×10^{-7}	2.46×10^{-6}	2.07×10^{-6}	4.48×10^{-6}
$87\mathrm{D}^8_{20}\mathrm{M}$	0	1.85×10^{-6}	4.13×10^{-6}	9.63×10^{-7}	2.37×10^{-6}

Table 7.3: Integrated continuous error norms for seven RK8(7)13M triples.

7.3.7 Numerical tests

For RK methods, the full set of fifteen RK-AID test problems (see Section A.6.28) will be used for collection of SBD statistics. The relevant RK-AID batch-file is as follows:

```
batch, echo          ! Batch mode with 'echo'
tol 1D-4             ! Starting tolerance
no tols 13           ! No. of tolerances
title                ! Name output files by method
dense output         ! Activate dense output
lump all problems    ! The fifteen problem average
```

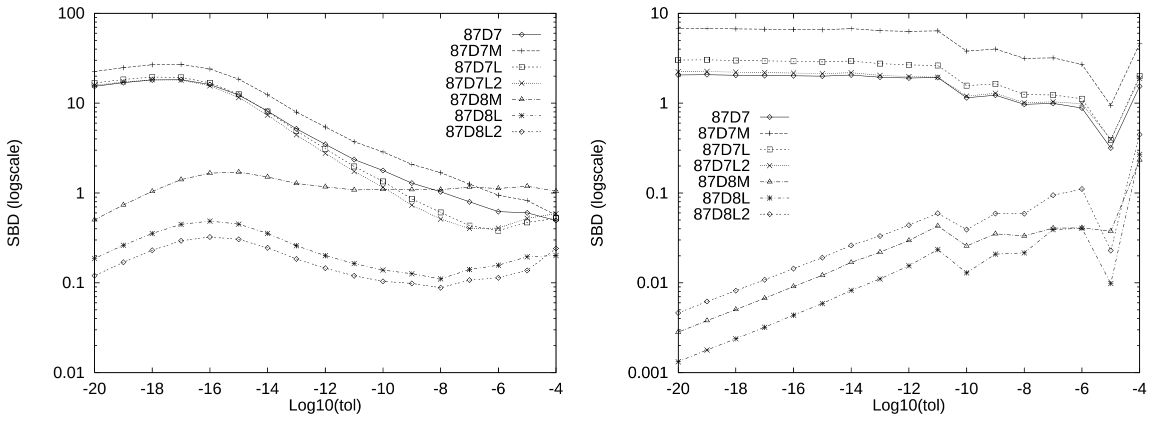


Figure 7.10: SBD on DETEST problems A1 and E5 for seven RK8(7)13M triples.

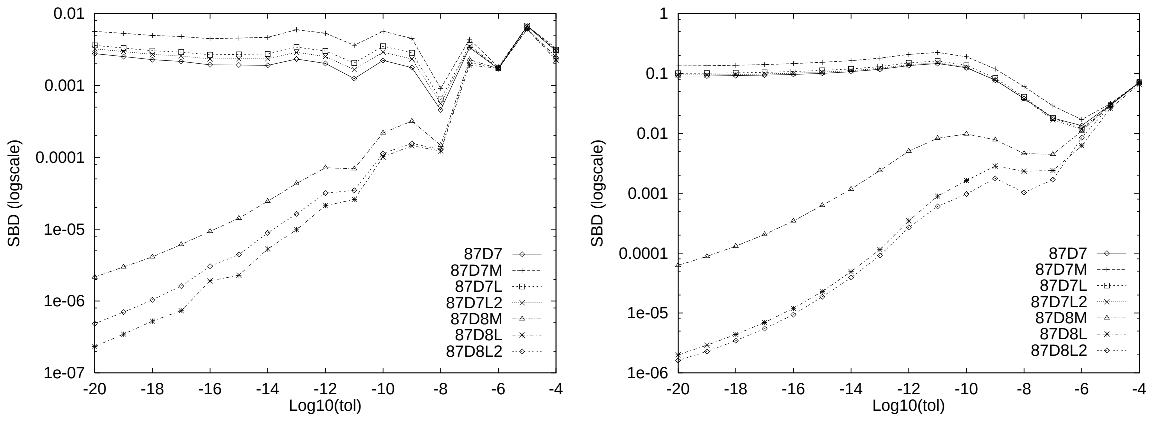


Figure 7.11: SBD on orbit D3 and RK-AID problem 11 for seven RK8(7)13M triples.

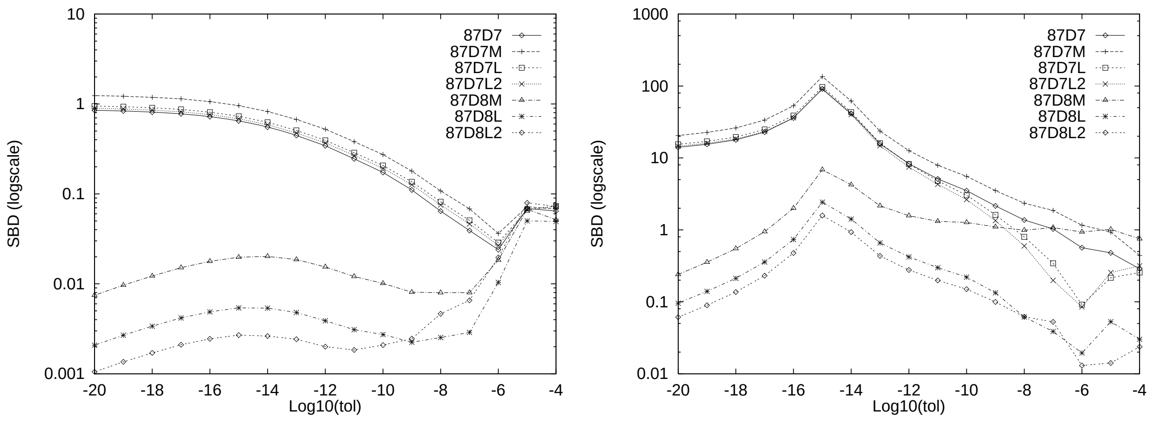


Figure 7.12: SBD on RK-AID problems 12 and 13 for seven RK8(7)13M triples.

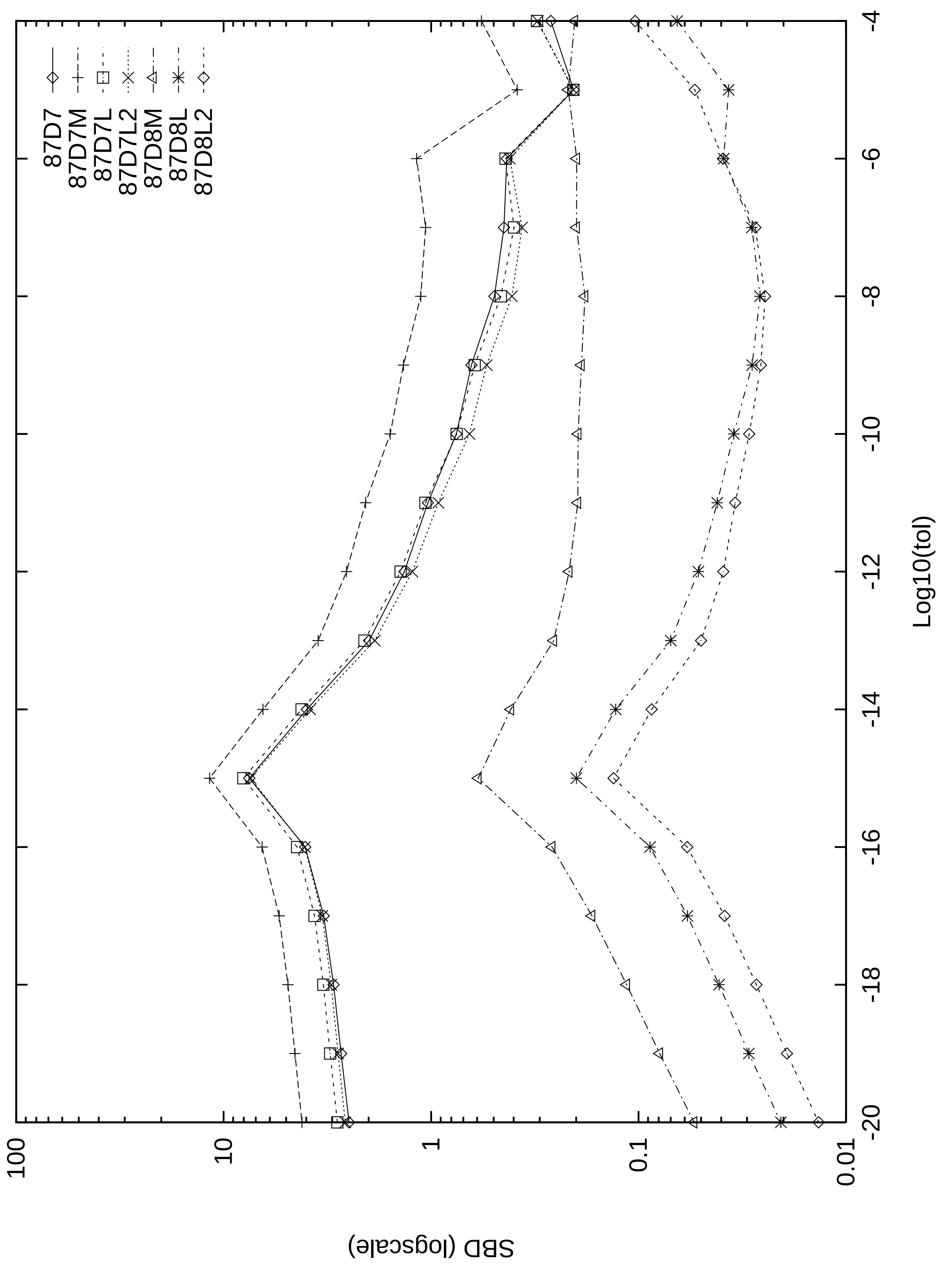


Figure 7.13: $\overline{\text{SBD}}$ over 15 problems for seven RK8(7)13M triples.

```
1 norm, box deviation ! SBD data average
!
test 8713d7 | go
test 8713d7l | go
test 8713d7l2 | go
test 8713d7m | go
test 8713d8l | go
test 8713d8l2 | go
test 8713d8m | go
```

The Fortran 90 double-precision arithmetic used in these tests allows only (approximately) 16 decimal figures of accuracy in the calculations. For stringent tolerances, a source of numerical rounding error in calculated SBD values will be the underlying discrete formula. Errors introduced in each step from calculation of extra stages for dense output are not propagated, and continuity of the polynomial weights at least ensures C^0 behaviour of the global errors. While the conditions

$$b_i^*(\sigma) \to \widehat{b}_i \text{ as } \sigma \to 1,\; i = 1, \ldots, s^*, \tag{7.25}$$

will be true in exact arithmetic, coefficients of the powers of σ are typically large at high-order. Therefore, to satisfy (7.25) to within machine-precision requires extra numerical accuracy than is available, and failure of this is not a *rounding* effect, as such, but simply a problem of finite precision calculation. Given method coefficients to a higher accuracy than the machine arithmetic will use, the `FLOAT` command, if added to the batch-file above, will ensure negligible precision effects in determination of the external polynomials at required non-mesh values of σ (see Sections A.5 and A.6.15). As such, results have been obtained with and without this scheme for comparison. In addition to this, it is instructive to perform the same tests in the absence of rounding error. For tolerances down to 1×10^{-20} this can be safely assumed when implemented in quadruple-precision (≈ 32 figure) arithmetic. While not available for the machines having been used during this work[1], the RK-AID test suite has been

[1]A SUN 'SPARC station IPX', and IBM PC's.

coded as a *separate* application for a recently available machine[2] allowing the desired calculations.

Six problems from the fifteen have been chosen for illustration of the general trends, the quadruple-precision results for which are shown via logscale SBD plots in Figures 7.10–7.12. An *average* over all problems ($\overline{\text{SBD}}$) is given in Figure 7.13, again from the quadruple-precision results. These processes have been developed with a view to implementation on less accurate platforms, and the effect of rounding error in the RK-AID (double-precision) results has been tabulated in Figure 7.14. Assuming the average quadruple-precision SBD data (that plotted in Figure 7.13) is free from rounding errors, a comparison with the RK-AID results will detect the onset of varying percentage rounding errors. For example, a 10% error in $\overline{\text{SBD}}$ using 'pure' double precision calculations first appears at tolerances between 10^{-11} and 10^{-14}, as indicated by column two of the upper table in Figure 7.14. Clearly, numerical instability of the continuous extensions can be gauged by a comparison over these quantities, discrete errors being the same in each case. Effects due to determination of the external weights at non-mesh points can be discounted by considering the `FLOAT`'ed data (the lower table in Figure 7.14).

Of the three eighth-order methods, the minimised local error model is clearly the worst in most cases (by the SBD global error gauge), and is a good justification of the linear deviation criterion. The two linearised truncation error formulae perform well, the $87\text{D}^8_{20}\text{L}_2$ appearing slightly better in most cases, more consistently so at stringent tolerances. This is not predicted theoretically, but is a good result from a compromised formula. Also, numerical stability does seem to have been improved in absence of the `FLOAT` command (as expected), with appearance of 10% rounding errors one tolerance later than for the $87\text{D}^8_{20}\text{L}$. The seventh-order dense output formulae are indistinguishable in most cases, though the $87\text{D}^7_{17}\text{M}$ can be seen to be consistently inferior throughout. The $87\text{D}^8_{20}\text{L}$ and 87D^7_{17}, having minimal truncation error properties, have been presented in Baker, et al. [66]. As an alternative based

[2]A Silicon Graphics 'O2'.

Double precision:

	10 %	10^2 %	10^3 %	10^4 %
$87\mathrm{D}^7_{17}$	10^{-13}	10^{-14}	–	–
$87\mathrm{D}^7_{17}\mathrm{M}$	10^{-14}	–	–	–
$87\mathrm{D}^7_{17}\mathrm{L}$	10^{-14}	–	–	–
$87\mathrm{D}^7_{17}\mathrm{L}_2$	10^{-14}	–	–	–
$87\mathrm{D}^8_{20}\mathrm{M}$	10^{-12}	10^{-12}	10^{-13}	–
$87\mathrm{D}^8_{20}\mathrm{L}$	10^{-11}	10^{-12}	10^{-13}	10^{-14}
$87\mathrm{D}^8_{20}\mathrm{L}_2$	10^{-12}	10^{-12}	10^{-13}	–

Double precision with polynomial precision extension:

	10 %	10^2 %	10^3 %	10^4 %
$87\mathrm{D}^7_{17}$	10^{-14}	–	–	–
$87\mathrm{D}^7_{17}\mathrm{M}$	10^{-14}	–	–	–
$87\mathrm{D}^7_{17}\mathrm{L}$	10^{-14}	–	–	–
$87\mathrm{D}^7_{17}\mathrm{L}_2$	10^{-14}	–	–	–
$87\mathrm{D}^8_{20}\mathrm{M}$	10^{-13}	–	–	–
$87\mathrm{D}^8_{20}\mathrm{L}$	10^{-13}	10^{-16}	–	–
$87\mathrm{D}^8_{20}\mathrm{L}_2$	10^{-13}	10^{-16}	–	–

Figure 7.14: Tol. for onset of %'age $\overline{\mathrm{SBD}}$ rounding error for seven RK8(7)13M triples.

on numerical stability, the $87\mathrm{D}^{8(7)}_{20(17)}\mathrm{L}_2$ may here be recommended as an ideal general 'quadruple', containing both a good seventh- and eighth-order dense formula in three and six extra (common) stages, respectively. Coefficients for this method are given approximately in Section B.2.1.

To give an illustration of dense output quality, and also to compare the local error and linear deviation criteria, continuous output has been done on problem A1 with a tolerance of 1×10^{-5} for the $87\mathrm{D}^{8(7)}_{20(17)}\mathrm{M}$ and $87\mathrm{D}^{8(7)}_{20(17)}\mathrm{L}_2$. The scalar, linear, nature of

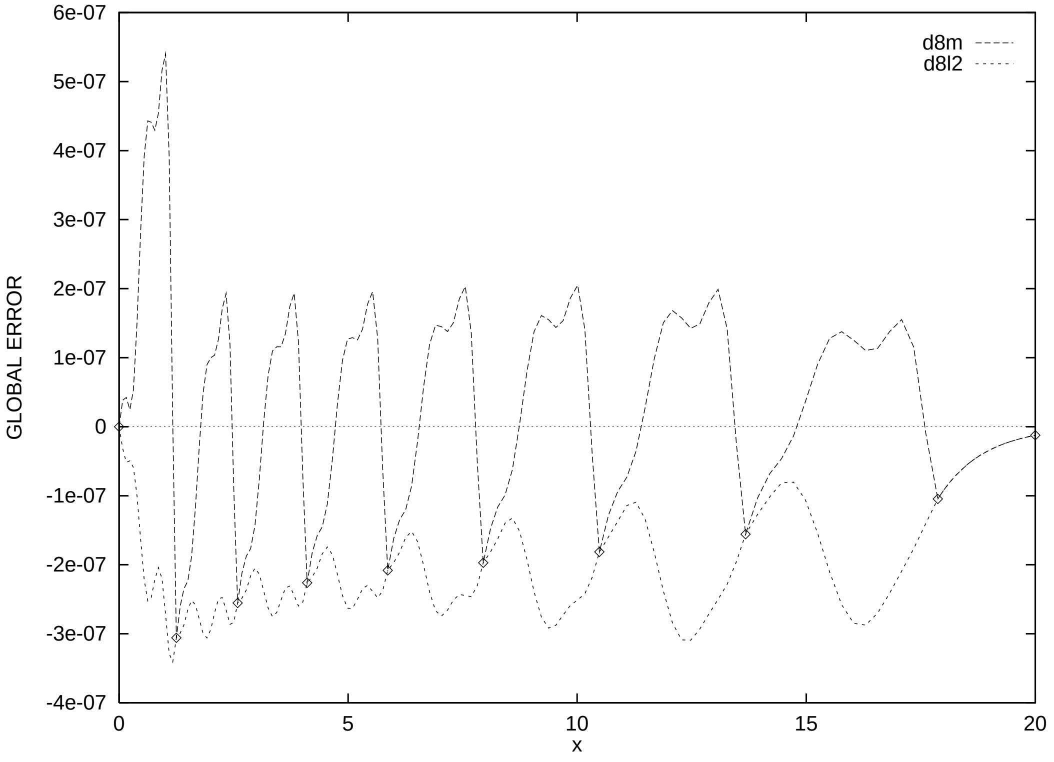


Figure 7.15: Continuous global error for problem A1 using two $\mathrm{RK8(7)13MD}^{8}_{20}$ triples.

the test allows a fair prediction of global error behaviour using expression (5.38), with $\boldsymbol{G}^*_{n+\sigma} = \sigma h_n \boldsymbol{f}_k(\boldsymbol{y}(x_n))^k \varepsilon_n$. Problem A1 (in autonomous form) has $\boldsymbol{f}(\boldsymbol{y}) = (1, -y)^T$, for $\boldsymbol{y} = (x, y)^T$, and so in this case for the y component (omitting the superscripts)

$$G^*_{n+\sigma} = -\sigma h_n \varepsilon_n. \tag{7.26}$$

Figure 7.15 contains the plot of dense output results for y, where it can clearly be seen that the linear deviation model 'checks' the continuous global error in each step predicted by (7.26), as compared with the minimised local error model, producing a smaller overshoot from the bounds of the discrete points.

7.4 Dense output for the RK8(6)12M

The equation systems in this case are simply obtained by imposing the conditions (7.4) to the model for the RK8(7)13M. This results in smaller linear systems, the solution of which is similar to that in the previous sections, and as such is described more briefly here.

7.4.1 Seventh- and eighth-order equation systems

The non-quadrature $\boldsymbol{b}^*$ matrix for eighth-order dense, $M^{(8)}$, can be formed from the following information:

$$n_q^{(8)} = 7,\ n^{(8)} = 18,\ \boldsymbol{\nu}_i^{b(8)} = \Big(\alpha_{i2}, c_i\alpha_{i2}, c_i^2\alpha_{i2}, \alpha_{i3}, c_i\alpha_{i3}, c_i^2\alpha_{i3}, \alpha_{i4}, c_i\alpha_{i4},$$

$$\alpha_{i7}, c_i\alpha_{i7}, \alpha_{i8}, \alpha_{i9}, c_i\alpha_{i9}, \alpha_{i10}, \alpha_{i11}, \alpha_{i12}, \alpha_{i14}, \alpha_{i15}\Big)^T.$$

The system for $a_{ij},\ j = s^a, \ldots, i-1,\ \overline{A}_i^{(8)}$, has

$$n_Q^{(8)} = 6,\ n_a^{(8)} = 7,\ \boldsymbol{\nu}_i^{d(8)} = (\alpha_{i2}, c_i\alpha_{i2}, \alpha_{i3}, c_i\alpha_{i3}, \alpha_{i4}, \alpha_{i7}, \alpha_{i9})^T,$$

$$\boldsymbol{\nu}_i^{r(8)} = (\alpha_{i7}, \alpha_{i8}, \alpha_{i9}, \alpha_{i10}, \alpha_{i12}, \alpha_{i14}, \alpha_{i15})^T,$$

and

$$\boldsymbol{r}_i^{(8)} = \left(\frac{15\,c_i^2 - \left(\sqrt{6}+6\right)\alpha_{i3} + \left(\sqrt{6}-6\right)\alpha_{i2}}{30}, \frac{50\,c_i^3 - \left(2\sqrt{6}+7\right)\alpha_{i3} + \left(2\sqrt{6}-7\right)\alpha_{i2}}{150},\right.$$

$$\frac{1125\,c_i^4-\left(19\sqrt{6}+54\right)\alpha_{i3}+\left(19\sqrt{6}-54\right)\alpha_{i2}}{4500},$$

$$\frac{4500\,c_i^5-\left(28\sqrt{6}+73\right)\alpha_{i3}+\left(28\sqrt{6}-73\right)\alpha_{i2}}{22500},$$

$$\frac{112500\,c_i^6+675000\,\alpha_{i4}-\left(241\sqrt{6}+606\right)\alpha_{i3}+\left(241\sqrt{6}-606\right)\alpha_{i2}}{675000},$$

$$\left.\frac{3375000\,c_i^7+23625000\,\alpha_{i11}-\left(2394\sqrt{6}+5929\right)\alpha_{i3}+\left(2394\sqrt{6}-5929\right)\alpha_{i2}}{23625000}\right)^T.$$

Equivalently, systems for seventh-order dense are very simple:

$$n_q^{(7)}=6,\ n^{(7)}=7,\ \boldsymbol{\nu}_i^{b(7)}=\left(\alpha_{i2},c_i\alpha_{i2},\alpha_{i3},c_i\alpha_{i3},\alpha_{i4},\alpha_{i7},\alpha_{i9}\right)^T,$$

$$n_Q^{(7)}=5,\ n_a^{(7)}=2,\ \boldsymbol{\nu}_i^{d(7)}=\left(\alpha_{i2},\alpha_{i3}\right)^T,\ \boldsymbol{\nu}_i^{r(7)}=\left(\alpha_{i7},\alpha_{i9}\right)^T,$$

and $\boldsymbol{r}_i^{(7)}$ as $\boldsymbol{r}_i^{(8)}$ but removing the last element.

7.4.2 Pre-analysis for eighth-order dense

Here, $r_b^{(8)}=6$, producing $\boldsymbol{\zeta}_i^{b(8)}$ and $\boldsymbol{\zeta}_i^{\beta(8)}$ vectors of length 6 and $n^{(8)}-r_b^{(8)}=12$, respectively. The minimum possible number of stages for the model is now 18 (seven for the quadrature equations plus six for $\overline{M}^8$), one less than for the RK8(7)13M. The dual systems $A_i^{(8)}$ and $W^{(7)}$ have $r_a^{(8)}=8$ and $r_b^{(7)}=5$, thus vectors $\boldsymbol{\zeta}_i^{d(8)}/\boldsymbol{\zeta}_i^{r(8)}$ and $\boldsymbol{\zeta}_i^{\delta(8)}/\boldsymbol{\zeta}_i^{\rho(8)}$ have size 8 and $n_\delta^{(8)}=n_a^{(8)}+n_Q^{(8)}-r_a^{(8)}=5$, respectively. Now, $M^{(7)}$ in this case yields $r_b^{(7)}=2$, and the two over-determined rows of $\overline{M}^{(7)}$ contained in $\boldsymbol{\zeta}_i^{\beta(7)}$ will be common to the $\boldsymbol{\zeta}_i^{\delta(8)}$ and $\boldsymbol{\zeta}_i^{\beta(8)}$ vectors (forced into the first two elements as before). Consideration of these quantities, again to be parameterised as $\varphi_i,\ i=1,2$, will be the key to trivial determination of seventh-order formulae as embedded processes within the eighth-order methods.

7.4.3 Change of variables for eighth-order dense

The 'linear' $\overline{\Upsilon}_i$ system is formed from

$$\varphi_{ik} = {}^{k}\zeta_i^{\beta(8)} \equiv {}^{k}\zeta_i^{\delta(8)}, \rho_{ik} = {}^{k}\zeta_i^{\rho(8)},\ k=1,2,\ \beta_{ik} = {}^{k}\zeta_i^{\beta(8)},\ k=3,\ldots,12, \tag{7.27}$$

and the 'nonlinear' form with

$$\delta_{ik} = {}^{k}\zeta_i^{\delta(8)},\ \rho_{ik} = {}^{k}\zeta_i^{\rho(8)},\ k=3,\ldots,5. \tag{7.28}$$

To provide a convenient derivation of embedded seventh-order formulae in the minimum number of stages, $\varphi_{i1} \equiv \delta_{i1} \equiv \beta_{i1} = \varphi_{i2} \equiv \delta_{i2} \equiv \beta_{i2} = 0$ will be imposed now. This may be considered as a simplifying condition, but would have arisen naturally as in the RK8(7)13M case. Consistency of the $\overline{A}_i^{(8)}$ systems now enforce the extra simplification

$$\rho_{i1} = \rho_{i2} = 0. \tag{7.29}$$

For this model,

$$\boldsymbol{\alpha}_i = (\alpha_{i2}, \alpha_{i3}, \alpha_{i4}, \alpha_{i8}, \alpha_{i9}, \alpha_{i10}, \alpha_{i11}, \alpha_{i12}, \alpha_{i14}, \alpha_{i15})^T,$$

and the augmented 'linear' $\boldsymbol{\alpha}_i$ system for equations (7.27) is outlined in Figure 7.16, where '$\times$' now represents a rational/surd of the form

$$\frac{a+\sqrt{6}}{b},$$

for integer a and b. Matrix Υ_i has rank 10, and even with full pivoting it is not possible to reduce the augmented system without using a c_i polynomial as a pivot. The resulting consistency conditions $\gamma_i,\ i=1,\ldots,4$, become linear combinations of the β_i's, all but one of which have third-degree c_i polynomial coefficients. To provide a simple solution, the 'γ_i' may be zeroed using only stage independent 'β_i'

specifications. This can be done by considering the form

$$\gamma_i = \sum_{j=0}^{3} p_{ij} c_i^j, \ i = 1, \ldots, 4,$$

where the p_{ij} will be constant coefficient linear combinations of $\beta_{ik}, \ k = 3, \ldots, 12$. The equations

$$\overline{\Upsilon}_i = \left(\begin{array}{ccccccccccc|c}
P_i^{0:1} & P_i^{0:1} & \times & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (\varphi_{i1} =)\, 0 \\
P_i^{0:1} & P_i^{0:1} & \times & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & (\varphi_{i2} =)\, 0 \\
0 & 0 & 0 & \times & \times & \times & \times & 0 & \times & 1 & 0 & (\rho_{i1} =)\, 0 \\
0 & 0 & 0 & \times & \times & \times & \times & 0 & \times & 0 & 1 & (\rho_{i2} =)\, 0 \\
P_i^{0:2} & P_i^{0:2} & P_i^{0:1} & 0 & 0 & 0 & 0 & \times & \times & 0 & 0 & \beta_{i3} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 0 & \times & \times & 0 & 0 & \beta_{i4} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & c_i & 0 & 0 & 0 & \times & \times & 0 & 0 & \beta_{i5} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 1 & 0 & 0 & \times & \times & 0 & 0 & \beta_{i6} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & c_i & 0 & \times & \times & 0 & 0 & \beta_{i7} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 1 & \times & \times & 0 & 0 & \beta_{i8} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 0 & \times & \times & 0 & 0 & \beta_{i9} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 0 & \times & \times & 0 & 0 & \beta_{i10} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 0 & \times & \times & 1 & 0 & \beta_{i11} \\
P_i^{0:2} & P_i^{0:1} & P_i^{0:1} & 0 & 0 & 0 & 0 & \times & \times & 0 & 1 & \beta_{i12}
\end{array} \right)$$

Figure 7.16: Linear 'α_i' system for eighth-order dense on the RK8(6)12M.

$$p_{ij} = 0, \ j = 0, \ldots, 3, \ i = 1, \ldots, 4,$$

then form a rank-4 homogeneous system for the β_i's, determining $\beta_{i3}, \beta_{i4}, \beta_{i8}$, and β_{i9} as rational/surd linear combinations of

$$\beta_{i5}, \beta_{i6}, \beta_{i7}, \beta_{i10}, \beta_{i11}, \beta_{i12}. \tag{7.30}$$

The 'nonlinear' equations (7.28) may now be appended onto the (consistent) $\overline{\Upsilon}_i$ system, and fully eliminated to give five consistency conditions. While these expressions are linear in all but the c_i, an unseen a difficulty appears with the large degree of the rational polynomials that arise. The presence of the radical $\sqrt{6}$ also increases the complexity of the expressions, and huge demands are made on the algebraic manipulator MACSYMA as a result. The five consistency conditions for complete removal of the α_i's from the system may now be solved for $\delta_{i3}, \delta_{i4}, \delta_{i5}, \rho_{i3}, \rho_{i4}$ in terms of c_i, ρ_{i5} and the 'β_i' in (7.30). It is highly impractical to show these, or any of the following expressions due to their complexity, and it is hoped that the detail given is sufficient for the theoretical development here, or to make replication possible.

7.4.4 Solution for eighth-order dense

A rank zero $\beta^{(8)}$ matrix ($r_\beta^{(8)} = 0$, $s^* = 18$) solution will now require

$$\beta_{i5} = \beta_{i6} = \beta_{i7} = \beta_{i10} = \beta_{i11} = \beta_{i12} = 0, \tag{7.31}$$

and, identical to the RK8(7)13M case, a solution with $r_\beta^{(8)} = 1$ in 19 stages can be achieved with $\lambda_{14} = 1$,

$$\beta_{ik} = \lambda_i \beta_{14k}, \; k \in \{5, 6, 7, 10, 11, 12\}, \tag{7.32}$$

and arbitrary (stage independent) λ_i, $i = 15, \ldots, s^*$. These two cases will now be examined by considering the $\overline{D}_i^{(8)}$ systems.

18-stage model

Having made the conditions (7.31), the $\overline{D}_i^{(8)}$ systems are formed from the free parameter ρ_{i5} and c_i polynomials, only. Consistency of $\overline{D}_{14}^{(8)}$ requires $\rho_{14,5} = 0$ and the solution of two high-degree polynomials in c_{14} for $\rho_{14,3} = \rho_{14,4} = 0$. No non-singular simultaneous solution could be found for these expressions.

19-stage model

Imposing (7.32) leaves $\beta_{14,5}, \beta_{14,6}, \beta_{14,7}, \beta_{14,10}, \beta_{14,11}, \beta_{14,12}$, and

$$c_i,\ i = 14, \ldots, 19,\ \lambda_i,\ i = 15, \ldots, 19,\ \rho_{i5},\ i = 14, \ldots, 19,$$

as free parameters, and solving for consistency of the $\overline{D}_i^{(8)}$ systems is simple to conceive. Again, the remaining work produces a 'contagion' of high-degree polynomial expressions for elements of the $\overline{D}_i^{(8)}$, and the following steps are *very* tedious:

1. $\overline{D}_{14}^{(8)}$ consistency. Set $\rho_{14,5} = 0$ and solve $\rho_{14,3} = \rho_{14,4} = 0$ with $\beta_{14,5}$ and $\beta_{14,6}$.
2. $\overline{D}_{15}^{(8)}$ consistency. A single step of Gaussian-elimination results in two consistency conditions that are rational and quadratic in the remaining β_{14}'s, but may be solved linearly for $\rho_{15,5}$ and λ_{15}.
3. $\overline{D}_{16}^{(8)}$ consistency. A single *huge* equation results from two pivoting operations, and may be solved with $\rho_{16,5}$.

System $\overline{D}_{17}^{(8)}$ is now automatically consistent, $\overline{D}_{18}^{(8)}$ may be solved with $a_{18,17}$ free, and $\overline{D}_{19}^{(8)}$ with $a_{19,17}$ and $a_{19,18}$ free.

7.4.5 Choice of free parameters

The model is complete save the transferring of all consistent systems into Fortran 90 for the GLOBS optimisation. Due to the large expressions involved, this process is difficult and results in a very slow function for calculation of the method tableau in terms of the free parameters. The ninth- and tenth-order truncation error terms are determined in exactly the same way as for the RK8(7)13M, using a modified version of the RK-AID command file in Section 7.3.6. Having the resulting reduction data files, and after many hours of computing time, two linear deviation models were obtained (with no comparison 'M' formula), having 20 free parameters given as follows:

- $\underline{86D^{8}_{19}L}$. Presented in Baker, et al. [66] as the '86DL'.

$$c_{14} = \frac{3}{19}, c_{15} = \frac{113}{480}, c_{16} = \frac{17}{38}, c_{17} = \frac{6}{17}, c_{18} = \frac{11}{16}, c_{19} = \frac{3}{4},$$

$$\lambda_{16} = \frac{371}{527}, \lambda_{17} = \frac{285}{559}, \lambda_{18} = \frac{231}{43}, \lambda_{19} = \frac{278}{243},$$

$$a_{18,17} = -\frac{174}{155}, a_{19,17} = -\frac{343}{114}, a_{19,18} = -\frac{7}{284},$$

$$\beta_{14,7} = -1.3899197122744593838338573221283255539476,$$

$$\beta_{14,10} = -1.4953707689429207111211757186363177761253,$$

$$\beta_{14,11} = -1.4566966501184324033421855390843566 1521 \times 10^{-1},$$

$$\beta_{14,12} = -1.0645770653827454920147517858060089 6024 \times 10^{-1},$$

$$\rho_{17,5} = 6.8182520187792330192470690000016807 47908 \times 10^{-3},$$

$$\rho_{18,5} = 4.6027421686926774945680702258190163 37567 \times 10^{-1},$$

$$\rho_{19,5} = 1.4156772352498080573172714734648573 42668 \times 10^{-1}.$$

- $\underline{86D^{8}_{19}L_2}$.

$$c_{14} = \frac{249}{1940}, c_{15} = \frac{31}{130}, c_{16} = \frac{1005}{2117}, c_{17} = \frac{13}{28}, c_{18} = \frac{31}{44}, c_{19} = \frac{407}{743},$$

$$\lambda_{16} = \frac{2109}{689}, \lambda_{17} = \frac{1331}{845}, \lambda_{18} = \frac{1997}{277}, \lambda_{19} = \frac{8995}{2412589},$$

$$a_{18,17} = -\frac{1177}{2501}, a_{19,17} = \frac{116}{1185}, a_{19,18} = \frac{42}{5165},$$

$$\beta_{14,7} = 1.3755881418968160910883550231297886 69599,$$

$$\beta_{14,10} = 2.8114561445394283905874843725588461 12972,$$

$$\beta_{14,11} = 1.4258153488378717286557166697902418 89788 \times 10^{-1},$$

$$\beta_{14,12} = 1.0485930334709781889464352460784199 98206 \times 10^{-1},$$

$$\rho_{17,5} = -7.43892503325907656977364774203591762995\,7 \times 10^{-2},$$

$$\rho_{18,5} = 3.82017151590329385350165473954135318599\,8 \times 10^{-1},$$

$$\rho_{19,5} = 6.91477520080964114653878461713489466823\,6 \times 10^{-2}.$$

These partially approximate specifications are again due to the optimisation having been performed over an equivalent set of parameters (as in the RK8(7)13M case), the rational expressions when converted to this form involve impracticably large numbers of digits and the radical $\sqrt{6}$. Note that the exact quantities are available, but are given

	$N^{(8)}$	$N_l^{(9)}$	$N_l^{(10)}$	$N_m^{(9)}$	$N_m^{(10)}$
86D$^7_{16}$P	1.42×10^{-5}	2.41×10^{-5}	2.90×10^{-5}	2.36×10^{-5}	2.77×10^{-5}
86D$^7_{16}$	5.21×10^{-6}	1.16×10^{-5}	1.70×10^{-5}	1.06×10^{-5}	1.45×10^{-5}
86D$^7_{16}$L	5.21×10^{-6}	1.15×10^{-5}	1.67×10^{-5}	1.05×10^{-5}	1.42×10^{-5}
86D$^7_{16}$L$_2$	5.35×10^{-6}	1.18×10^{-5}	1.69×10^{-5}	1.08×10^{-5}	1.46×10^{-5}
86D$^8_{19}$L	0	1.14×10^{-6}	2.68×10^{-6}	2.56×10^{-6}	5.53×10^{-6}
86D$^8_{19}$L$_2$	0	1.54×10^{-6}	3.74×10^{-6}	2.50×10^{-6}	5.03×10^{-6}

Table 7.4: Integrated continuous error norms for six RK8(6)12M triples.

approximately here for convenience. The latter of these specifications gives a compromised formula intended as an alternative to that in [66], which has slightly worse error norm characteristics but smaller b_i^* polynomial coefficients to benefit numerical stability. These eighth-order processes contain seventh-order methods using their first 16 stages, and the complete embedded triples will be denoted by 86D$^{8(7)}_{19(16)}$L and 86D$^{8(7)}_{19(16)}$L$_2$. A dedicated seventh-order formula has also been derived and optimised, presented in Baker, et al. [66], the derivation of which is very simple (with reference to the seventh-order case for the RK8(7)13M), and as such will not be detailed here.

Six dense output formulae are now available, namely the seventh-order formula of [37], to be denoted as the 86D$^7_{16}$P, the 86D$^8_{19}$L and 86D$^8_{19}$L$_2$ with associated (embedded) 86D$^7_{16}$L and 86D$^7_{16}$L$_2$, and the dedicated model 86D$^7_{16}$ from [66]. Continuous error norms for these methods are given (integrated) in Table 7.4, again accurate to

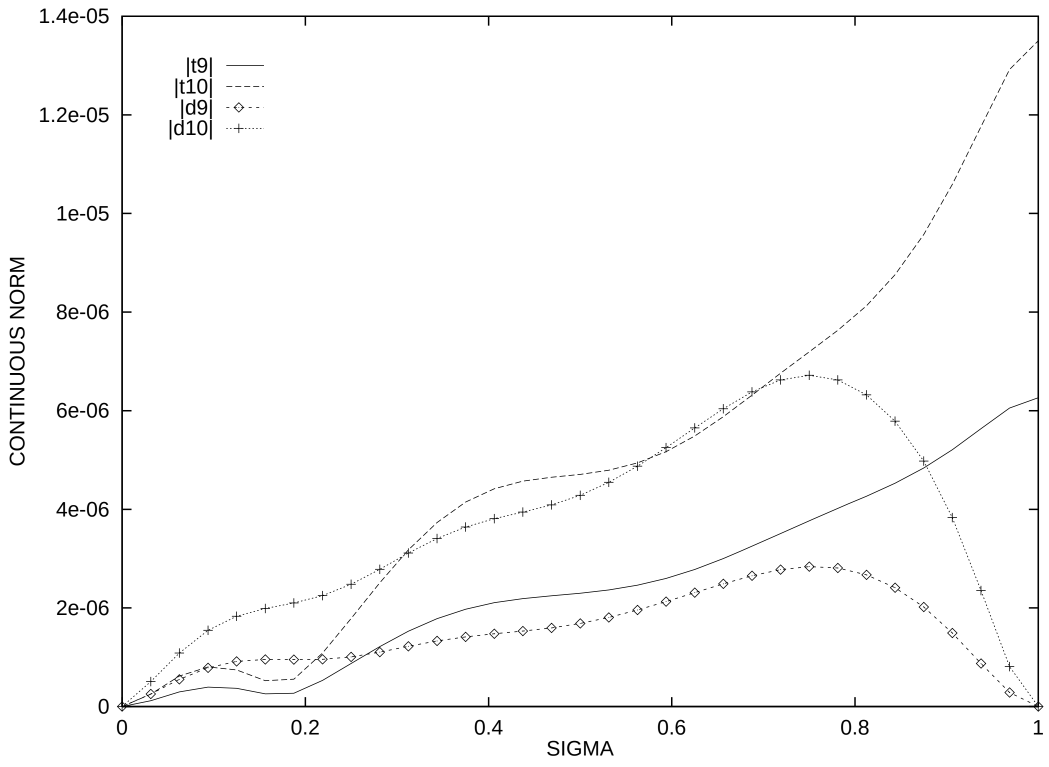


Figure 7.17: Principal/secondary error and linear deviation norms for the $86D^8_{19}L_2$.

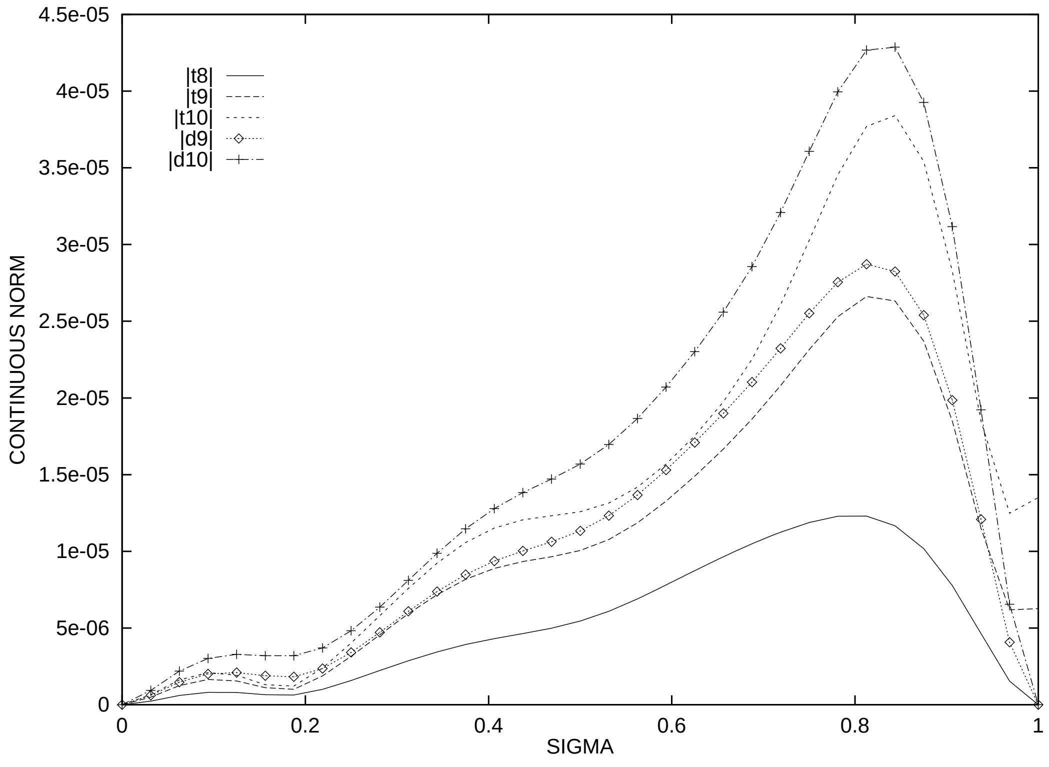


Figure 7.18: Principal to tertiary error and linear deviation norms for the $86D^7_{16}L_2$.

two decimal places, and plotted for the $86D^8_{19}L_2$ and $86D^7_{16}L_2$ in Figures 7.17 and 7.18. A similar plot for the $86D^8_{19}L$ is given in [66]. The norm values show significant improvement in truncation error characteristics of the new seventh-order formulae over the $86D^7_{16}P$. Also, extra degrees of freedom available for the new dedicated model (not derived here) are insignificant where the principal error norm is concerned, as compared with the $86D^8_{19}L$, and have actually been chosen to the slight disadvantage of secondary and tertiary terms.

7.4.6 Numerical tests

The four seventh-order and two eighth-order methods have been compared with SBD statistics on the fifteen RK-AID test problems, a selection of which are given in Figures 7.19–7.21 (all using quadruple-precision software). An equivalent treatment of rounding errors as for the RK8(7)13M is given in Figure 7.23. A large SBD inferiority is noticeable for the $86D^7_{16}P$ in all cases, and slightly worse average results for the $86D^7_{16}$ at lax tolerances. Otherwise, it is unreasonable to make any further conclusions from the seventh-order results, giving indistinguishable SBD plots. However, the $86D^7_{16}P$ has considerably better numerical stability, which for tolerances between 10^{-12} and 10^{-14} may be significant with only double-precision calculations available. This advantage disappears when using the `FLOAT` scheme, and such instability can be solely accredited to polynomial weight coefficient sizes.

Surprisingly, eighth-order results show a considerable advantage from the compromised $86D^8_{19}L_2$ formula, which has marginally worse theoretical truncation error properties than the $86D^8_{19}L$. Also, the $86D^8_{19}L_2$ does appear slightly more numerically stable with, for example, a 100% $\overline{\text{SBD}}$ error occurring one tolerance later. This difference again disappears when using the `FLOAT` scheme. With the optimal truncation error property formulae ($86D^7_{16}$ and $86D^8_{19}L$) already recommended in [66], it is here reasonable to present the compromised $86D^{8(7)}_{19(16)}L_2$ for all purposes with no need for a *separate* 16-stage model for seventh-order dense, and is given approximately in Section B.2.2.

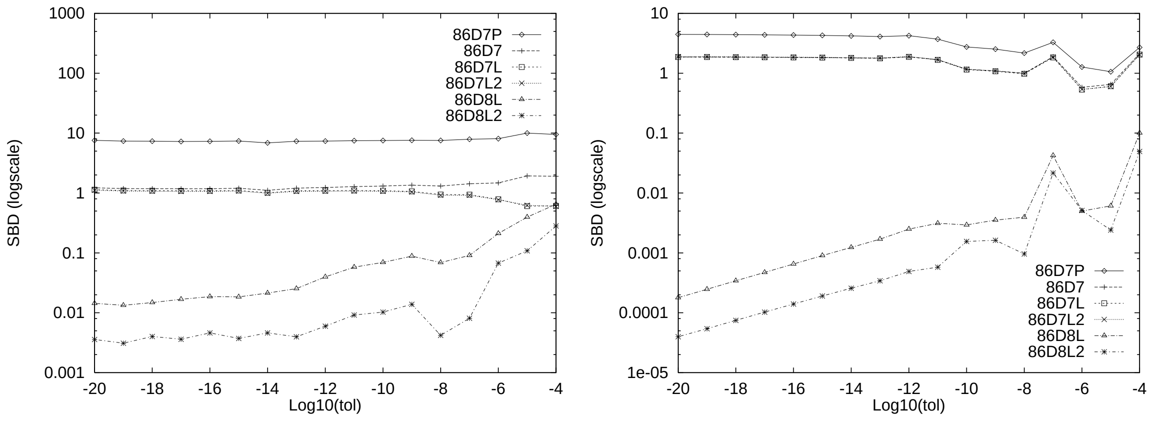


Figure 7.19: SBD on DETEST problems A1 and E5 for six RK8(6)12M triples.

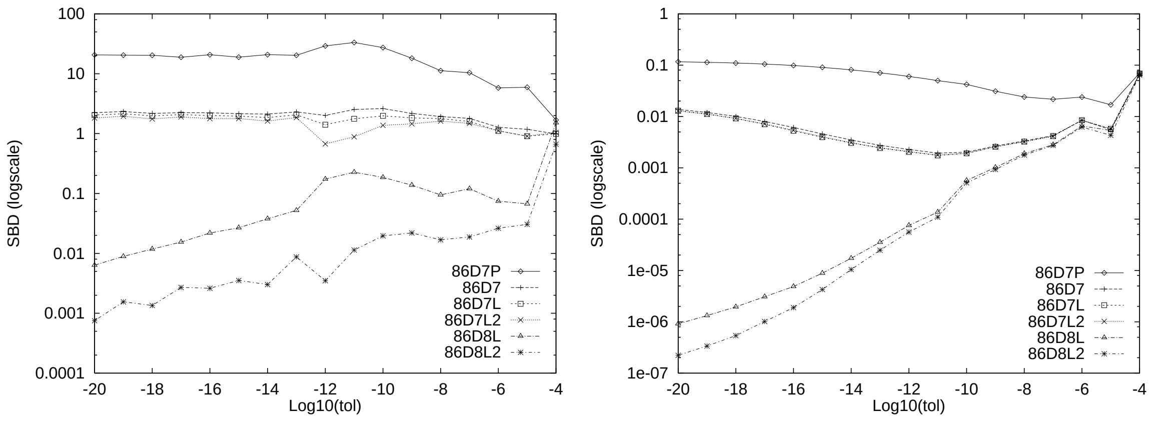


Figure 7.20: SBD on RK-AID problems 9 and 11 for six RK8(6)12M triples.

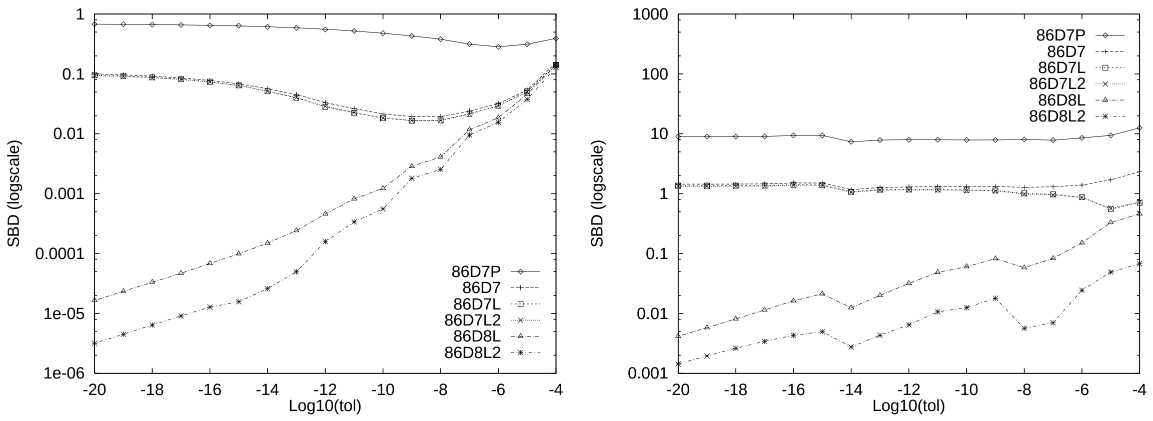


Figure 7.21: SBD on RK-AID problems 12 and 13 for six RK8(6)12M triples.

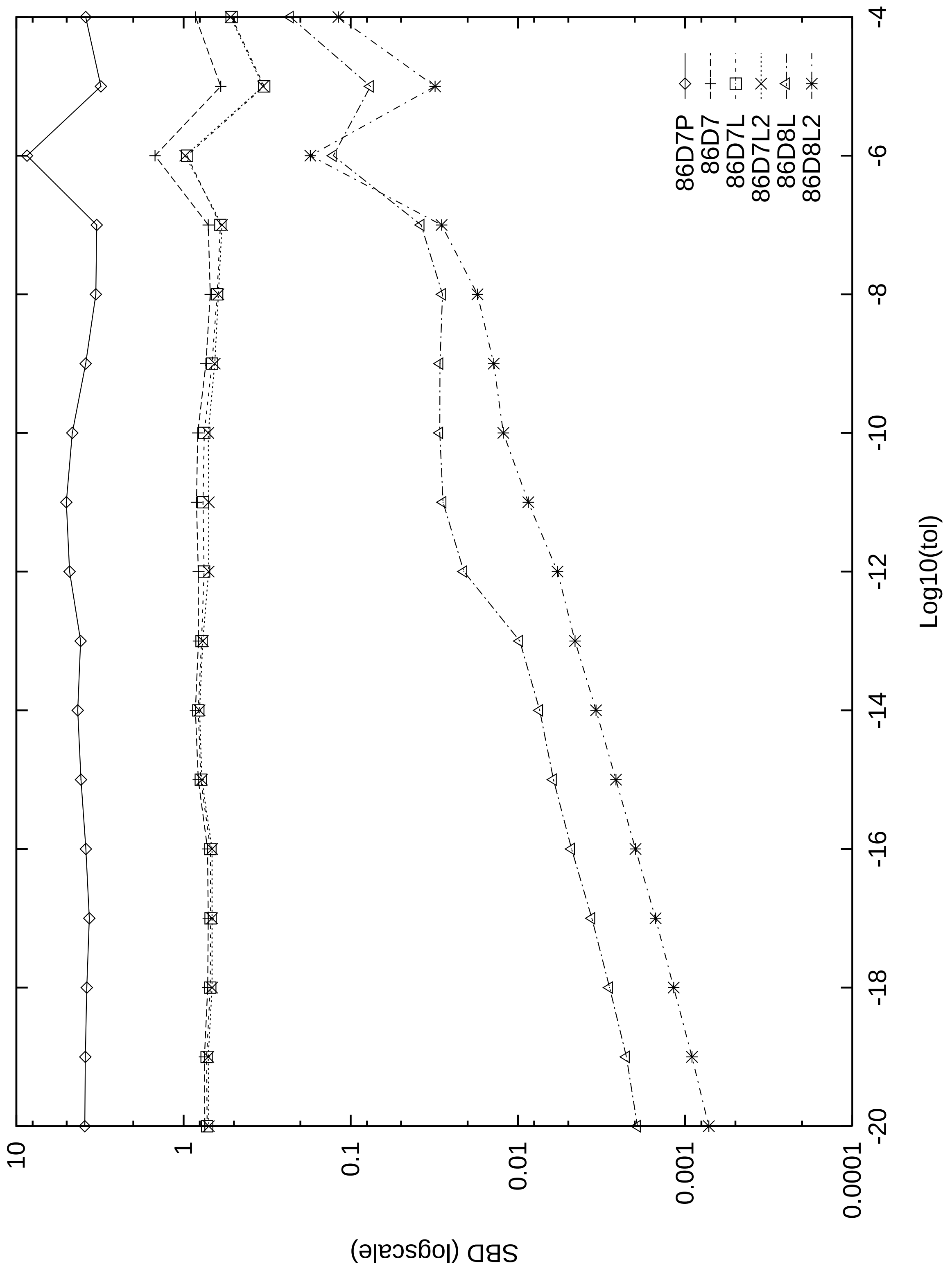


Figure 7.22: $\overline{\mathrm{SBD}}$ over 15 problems for six RK8(6)12M triples.

Double precision:

	10 %	10^2 %	10^3 %	10^4 %	10^5 %	10^6 %
$86D^7_{16}P$	10^{-14}	–	–	–	–	–
$86D^7_{16}$	10^{-12}	10^{-13}	10^{-16}	–	–	–
$86D^7_{16}L$	10^{-12}	10^{-13}	–	–	–	–
$86D^7_{16}L_2$	10^{-12}	10^{-13}	–	–	–	–
$86D^8_{19}L$	10^{-10}	10^{-10}	10^{-11}	10^{-12}	10^{-13}	10^{-14}
$86D^8_{19}L_2$	10^{-10}	10^{-11}	10^{-11}	10^{-12}	10^{-13}	10^{-15}

Double precision with polynomial precision extension:

	10 %	10^2 %	10^3 %	10^4 %	10^5 %	10^6 %
$86D^7_{16}P$	10^{-14}	–	–	–	–	–
$86D^7_{16}$	10^{-14}	–	–	–	–	–
$86D^7_{16}L$	10^{-14}	–	–	–	–	–
$86D^7_{16}L_2$	10^{-14}	–	–	–	–	–
$86D^8_{19}L$	10^{-12}	10^{-13}	10^{-14}	–	–	–
$86D^8_{19}L_2$	10^{-12}	10^{-13}	10^{-14}	–	–	–

Figure 7.23: Tol. for onset of %'age $\overline{\text{SBD}}$ rounding error for six RK8(6)12M triples.

Chapter 8

Continuous twelfth-order RKN processes

8.1 Introduction

This chapter considers the highest order Runge-Kutta-Nyström method currently available, no form of continuous extension for which exists at present. This formula is the RKN12(10)17M (an $\text{RKN}_{17}^{12(10)}$ pair) of Dormand, et al. [56], for which a range of dense output processes are now to be derived, up to a maximum of twelfth-order on both y and y' ($p^* = 12$). The exact rationals for the RKN12(10)17M method tableau are required for the following analysis, and can be found in the production code of Brankin, et al. [64], in which the method is incorporated.

8.2 Order conditions

Derivation of single twelfth-order RKN formulae requires 1104 τ''s and 563 τ's to be solved to zero. Of these, only 357 and 199 are independent, respectively (see Table 4.5). Unlike the other formulae considered previously in this work, the model for the RKN12(10)17M is not published in any detail, and so must be recreated for determination of the continuous equations of condition up to order 12. This can be done automatically with RK-AID by generating the higher-order discrete formula model on y' using the commands

```
read 121017m
```

```
!
[h'] order :12, split
!
use Q's
equation mode
!
float, threshold 1.0D-30
```

assuming the formula coefficients are contained exactly or to at least 30–40 decimal places of accuracy in the file "*121017m*". This will compare $\widehat{b}'_i$, a_{ij} and evaluated Q_{ik} parameters with the zero threshold, and work through the model 'splitting' the equations, and displaying a set of independent conditions. Note that although the discrete formula satisfies $\widehat{R}_{0i} = 0,\ i = 1, \ldots, 17$ (easily seen by addition of the command `use R[h]'s` to the above script), this is of no interest in determination of a likely set of dense order conditions, and as such has been ignored here. Examination of the method coefficients and the "*rkive.out*" file from the above run show the formula to have

$$\widehat{b}'_i = 0,\ i = 2, \ldots, 6,\ a_{i2} = 0,\ i = 5, \ldots, 17,$$

$$Q_{i1} = Q_{i2} = 0,\ i = 3, \ldots, 17,\ Q_{i3} = 0,\ i = 5, \ldots, 17,\ Q_{i4} = 0,\ i = 6, \ldots, 17,$$

and a possible independent set of non-quadrature equations shown in Figure 8.1. The two lines are to separate terms generated from 10th, 11th and 12th order truncation error coefficients (reading them from top-to-bottom and left-to-right). It should be noted that 'split' terms containing a_{i5} occur one order later than those containing a_{i3} and a_{i4} as their end node. This may initially appear confusing, but no warning is given by RK-AID to suggest an incomplete model (see Section 3.3.4), and the terms

$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 \sum_{j=6}^{i-1} a_{ij} a_{j5},\ \sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} c_j a_{j5}, \sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j^2 a_{j5},\ \text{and}\ \sum_{i=8}^{17} \widehat{b}'_i \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k5},$$

do not appear as a result. The usual case is therefore maintained, where of the complete equation set for twelfth-order dense, terms up to the first line relate to a tenth-order model, and those between the lines may be additionally included for an eleventh-

$$\sum_{i=7}^{17} \widehat{b}'_i Q_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i a_{i3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i a_{i4}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i Q_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i Q_{i6}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i a_{i3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i a_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i a_{i4}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 Q_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i Q_{i6}$$
$$\sum_{i=7}^{17} \widehat{b}'_i Q_{i7}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 a_{i3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i a_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} Q_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 a_{i4}$$

$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} a_{j3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} a_{j4}$$

$$\sum_{i=7}^{17} \widehat{b}'_i c_i^3 Q_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 Q_{i6}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i Q_{i7}$$
$$\sum_{i=7}^{17} \widehat{b}'_i Q_{i8}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^3 a_{i3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 a_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} Q_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j Q_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} Q_{j6}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^3 a_{i4}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} a_{j3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j a_{j3}$$

$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} a_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} a_{j4}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j a_{j4}$$

$$\sum_{i=7}^{17} \widehat{b}'_i c_i^4 Q_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^3 Q_{i6}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 Q_{i7}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i Q_{i8}$$
$$\sum_{i=7}^{17} \widehat{b}'_i Q_{i9}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^4 a_{i3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^3 a_{i5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 \sum_{j=6}^{i-1} a_{ij} Q_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} c_j Q_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j^2 Q_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} Q_{j6}$$

$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j Q_{j6}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} Q_{j7}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^4 a_{i4}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 \sum_{j=6}^{i-1} a_{ij} a_{j3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} c_j a_{j3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j^2 a_{j3}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} a_{j5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j a_{j5}$$
$$\sum_{i=8}^{17} \widehat{b}'_i \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} Q_{k5}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i^2 \sum_{j=6}^{i-1} a_{ij} a_{j4}$$
$$\sum_{i=7}^{17} \widehat{b}'_i c_i \sum_{j=6}^{i-1} a_{ij} c_j a_{j4}$$
$$\sum_{i=7}^{17} \widehat{b}'_i \sum_{j=6}^{i-1} a_{ij} c_j^2 a_{j4}$$
$$\sum_{i=8}^{17} \widehat{b}'_i \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k3}$$
$$\sum_{i=8}^{17} \widehat{b}'_i \sum_{j=7}^{i-1} a_{ij} \sum_{k=6}^{j-1} a_{jk} a_{k4}$$

Figure 8.1: Non-quadrature expressions for the $\widehat{y}'$ formula of the RKN12(10)17M.

order model. Assuming $s^b = 7$, $\boldsymbol{s}^a = (2,2,3,3,\ldots)^T$ and $\boldsymbol{q} = (3,3,4,5,5,\ldots)^T$, an independent set of order conditions for (up to) a twelfth-order continuous extension on y' may now be inferred by simply replacing $\widehat{b}'_i$ by b'^*_i, and the number of stages from 17 to a free s^*. Dense output on y requires the same (non-quadrature) conditions as for y', but one order down, replacing b'^*_i by b^*_i — these formulae will be derived trivially in most cases as for the sixth-order Nyström cases of Chapter 6. The resulting 57 continuous order conditions may be parameterised with $n_\alpha = 25$ to give the equation set in Figure 8.2, where

$$\alpha_{i1} = Q_{i5},\ \alpha_{i2} = a_{i3},\ \alpha_{i3} = a_{i4},\ \alpha_{i4} = Q_{i6},\ \alpha_{i5} = a_{i5},\ \alpha_{i6} = Q_{i7},\ \alpha_{i7} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j1},$$

$$\alpha_{i8} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j2},\ \alpha_{i9} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j3},\ \alpha_{i10} = Q_{i8}{}^{\dagger},\ \alpha_{i11} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j1}{}^{\dagger},\ \alpha_{i12} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j4}{}^{\dagger},$$

$$\alpha_{i13} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j2}{}^{\dagger},\ \alpha_{i14} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j5}{}^{\dagger},\ \alpha_{i15} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j3}{}^{\dagger},\ \alpha_{i16} = Q_{i9}{}^{\ddagger},$$

$$\alpha_{i17} = \sum_{j=6}^{i-1} a_{ij}c_j^2\alpha_{j1}{}^{\ddagger},\ \alpha_{i18} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j4}{}^{\ddagger},\ \alpha_{i19} = \sum_{j=6}^{i-1} a_{ij}\alpha_{j6}{}^{\ddagger},\ \alpha_{i20} = \sum_{j=6}^{i-1} a_{ij}c_j^2\alpha_{j2}{}^{\ddagger},$$

$$\alpha_{i21} = \sum_{j=6}^{i-1} a_{ij}c_j\alpha_{j5}{}^{\ddagger},\ \alpha_{i22} = \sum_{j=7}^{i-1} a_{ij}\alpha_{j7}{}^{\ddagger},\ \alpha_{i23} = \sum_{j=6}^{i-1} a_{ij}c_j^2\alpha_{j3}{}^{\ddagger},\ \alpha_{i24} = \sum_{j=7}^{i-1} a_{ij}\alpha_{j8}{}^{\ddagger},$$

and

$$\alpha_{i25} = \sum_{j=7}^{i-1} a_{ij}\alpha_{j9}{}^{\ddagger},$$

terms specific to the $p^{y'^*} = 11$ and $p^{y'^*} = 12$ models being marked with dagger symbols (†) and (‡), respectively. A reusable stage must be added with $s^r = 18$ to give C^2 continuity, and the $\overline{A}_i$ systems naturally have $s^a = 6$. However, to force a dual nature between the A_i and W_i matrices, s^a will be set to $s^b = 7$, and $\alpha_{i26} \equiv a_{i6}$ introduced as a further parameter. This is a slight modification to the definitions of Chapter 5, but seems more natural in this case, giving $n_\alpha = 26$.

$\sum_{i=7}^{s^*} b_i'^* \alpha_{i1} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i5} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i7} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i6} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^4 \alpha_{i3} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* \alpha_{i2} = 0$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i7} = 0$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i11} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i10} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i8} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* \alpha_{i3} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i3} = 0$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i12} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i16} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i13} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i1} = 0$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i8} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i^3 \alpha_{i3} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^4 \alpha_{i2} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i20} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* \alpha_{i4} = 0$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i9} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i8} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^3 \alpha_{i5} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i14} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i2} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i^3 \alpha_{i1} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i13} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i7} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i21} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* \alpha_{i5} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i4} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i14} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i11} = 0^\ddagger$	$\sum_{i=8}^{s^*} b_i'^* \alpha_{i22} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i3} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i6} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i9} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i17} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i9} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i1} = 0$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i10} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i15} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i12} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i15} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* c_i \alpha_{i4} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i^3 \alpha_{i2} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^4 \alpha_{i1} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i18} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i23} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* \alpha_{i6} = 0$	$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i5} = 0^\dagger$	$\sum_{i=7}^{s^*} b_i'^* c_i^3 \alpha_{i4} = 0^\ddagger$	$\sum_{i=7}^{s^*} b_i'^* \alpha_{i19} = 0^\ddagger$	$\sum_{i=8}^{s^*} b_i'^* \alpha_{i24} = 0^\ddagger$
$\sum_{i=7}^{s^*} b_i'^* c_i^2 \alpha_{i2} = 0$				$\sum_{i=8}^{s^*} b_i'^* \alpha_{i25} = 0^\ddagger$

Figure 8.2: Parameterised non-quadrature y'^* equations for the RKN12(10)17M.

8.3 Equation systems

Linear systems for twelfth-order and dedicated eleventh-order dense output models will now be given, with embedded tenth-order formulae also considered. The non-quadrature $\boldsymbol{b}'^*$ matrices, $M^{(p)}$ for $p = 10, 11$ and 12 have $n_q^{(p)} = p - 1$, $n^{(12)} = 57$ and

$$\boldsymbol{\nu}_i^{b(12)} = (\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, c_i\alpha_{i1}, \alpha_{i4}, c_i\alpha_{i2}, \alpha_{i5}, c_i\alpha_{i3}, c_i^2\alpha_{i1}, c_i\alpha_{i4}, \alpha_{i6}, c_i^2\alpha_{i2}, c_i\alpha_{i5}, \alpha_{i7},$$
$$c_i^2\alpha_{i3}, \alpha_{i8}, \alpha_{i9}, c_i^3\alpha_{i1}, c_i^2\alpha_{i4}, c_i\alpha_{i6}, \alpha_{i10}, c_i^3\alpha_{i2}, c_i^2\alpha_{i5}, c_i\alpha_{i7}, \alpha_{i11}, \alpha_{i12}, c_i^3\alpha_{i3}, c_i\alpha_{i8}, \alpha_{i13},$$
$$\alpha_{i14}, c_i\alpha_{i9}, \alpha_{i15}, c_i^4\alpha_{i1}, c_i^3\alpha_{i4}, c_i^2\alpha_{i6}, c_i\alpha_{i10}, \alpha_{i16}, c_i^4\alpha_{i2}, c_i^3\alpha_{i5}, c_i^2\alpha_{i7}, c_i\alpha_{i11}, \alpha_{i17}, c_i\alpha_{i12},$$
$$\alpha_{i18}, \alpha_{i19}, c_i^4\alpha_{i3}, c_i^2\alpha_{i8}, c_i\alpha_{i13}, \alpha_{i20}, c_i\alpha_{i14}, \alpha_{i21}, \alpha_{i22}, c_i^2\alpha_{i9}, c_i\alpha_{i15}, \alpha_{i23}, \alpha_{i24}, \alpha_{i25})^T,$$

the first $n^{(11)} = 32$ elements of which form $\boldsymbol{\nu}_i^{b(11)}$, and of those, the first $n^{(10)} = 17$ elements form $\boldsymbol{\nu}_i^{b(10)}$. The $\overline{A}_i$ systems have $n_Q^{(p)} = p - 3$, $n_\alpha^{(12)} = 17$,

$$\boldsymbol{\nu}_i^{d(12)} = \Big(\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, c_i\alpha_{i1}, \alpha_{i4}, c_i\alpha_{i2}, \alpha_{i5}, c_i\alpha_{i3}, c_i^2\alpha_{i1}, c_i\alpha_{i4},$$
$$\alpha_{i6}, c_i^2\alpha_{i2}, c_i\alpha_{i5}, \alpha_{i7}, c_i^2\alpha_{i3}, \alpha_{i8}, \alpha_{i9}\Big)^T,$$

and

$$\boldsymbol{\nu}_i^{r(12)} = \Big(\alpha_{i7} + \frac{1472\alpha_{i26}}{1922607421875}, \alpha_{i8} - \frac{1408\alpha_{i26}}{196875}, \alpha_{i9} - \frac{2048\alpha_{i26}}{703125},$$
$$\alpha_{i11} + \frac{5888\alpha_{i26}}{48065185546875}, \alpha_{i12} + \frac{41696\alpha_{i26}}{144195556640625}, \alpha_{i13} - \frac{5632\alpha_{i26}}{4921875}, \alpha_{i14} - \frac{432\alpha_{i26}}{546875},$$
$$\alpha_{i15} - \frac{8192\alpha_{i26}}{17578125}, \alpha_{i17} + \frac{23552\alpha_{i26}}{1201629638671875}, \alpha_{i18} + \frac{166784\alpha_{i26}}{3604888916015625},$$
$$\alpha_{i19} + \frac{108464\alpha_{i26}}{1544952392578125}, \alpha_{i20} - \frac{22528\alpha_{i26}}{123046875}, \alpha_{i21} - \frac{1728\alpha_{i26}}{13671875},$$
$$\alpha_{i22}, \alpha_{i23} - \frac{32768\alpha_{i26}}{439453125}, \alpha_{i24}, \alpha_{i25}\Big)^T,$$

the first $n_\alpha^{(11)} = 8$ elements of which form $\boldsymbol{\nu}_i^{d(11)}$ and $\boldsymbol{\nu}_i^{r(11)}$. Also,

$$\boldsymbol{r}_i^{(12)} = \Big(c_i^3 - \alpha_{i2}c_3 - \alpha_{i3}c_4 - \alpha_{i5}c_5 - \alpha_{i26}c_6,\ c_i^4 - \alpha_{i2}c_3^2 - \alpha_{i3}c_4^2 - \alpha_{i5}c_5^2 - \alpha_{i26}c_6^2,$$

$$c_i^5 - \alpha_{i2}c_3^3 - \alpha_{i3}c_4^3 - \alpha_{i5}c_5^3 - \alpha_{i26}c_6^3,\ c_i^6 - \alpha_{i2}c_3^4 - \alpha_{i3}c_4^4 - \alpha_{i5}c_5^4 - \alpha_{i26}c_6^4,$$

$$c_i^7 + \alpha_{i1} - \alpha_{i2}c_3^5 - \alpha_{i3}c_4^5 - \alpha_{i5}c_5^5 - \alpha_{i26}c_6^5,\ c_i^8 + \alpha_{i4} - \alpha_{i2}c_3^6 - \alpha_{i3}c_4^6 - \alpha_{i5}c_5^6 - \alpha_{i26}c_6^6,$$

$$c_i^9 + \alpha_{i6} - \alpha_{i2}c_3^7 - \alpha_{i3}c_4^7 - \alpha_{i5}c_5^7 - \alpha_{i26}c_6^7, c_i^{10} + \alpha_{i10} - \alpha_{i2}c_3^8 - \alpha_{i3}c_4^8 - \alpha_{i5}c_5^8 - \alpha_{i26}c_6^8,$$

$$c_i^{11} + \alpha_{i16} - \alpha_{i2}c_3^9 - \alpha_{i3}c_4^9 - \alpha_{i5}c_5^9 - \alpha_{i26}c_6^9\Big)^T,$$

for $c_3 = 1/25$, $c_4 = 1/10$, $c_5 = 2/15$ and $c_6 = 4/25$, all but the last element of which (that relating to Q_{i9}) forming $\boldsymbol{r}_i^{(11)}$. Note that the rational expressions for the powers of $c_i,\ i = 3, \ldots, 6$, become large and so have not been incorporated explicitly here.

The abbreviated $A_i^{(12)}$ and $W^{(10)}$ matrices are identical, this having been referred to as 'duality' in the RK case, but here the systems are *two* orders apart (as compared with a single order in the RK case). This will again prove invaluable for avoidance of singular cases in the following analysis, and for the embedding of tenth-order dense formulae within the $p^* = 11$ and $p^* = 12$ models.

8.4 Pre-analysis

The rank of the numerical part of $M^{(12)}$ is $r_b^{(12)} = 10$, thus eliminating yields a $\boldsymbol{\zeta}_i^{\beta(12)}$ vector of length $n_\beta^{(12)} = n^{(12)} - r_b^{(12)} = 47$. As previously, this vector can be forced to

p	$n_Q^{(p)}$	$n_a^{(p)}$	$r_a^{(p)}$	$n_\delta^{(p)}$	$n_q^{(p)}$	$n^{(p)}$	$r_b^{(p)}$	$n_\beta^{(p)}$
12	9	17	11	15	11	57	10	47
11	8	8	10	6	10	32	9	23
10	–	–	–	–	9	17	7	10

Table 8.1: Pre-analysis summary for the RKN12(10)17M continuous extensions.

contain $\boldsymbol{\zeta}_i^{\beta(11)}$ as its first $n_\beta^{(11)} = n^{(11)} - r_b^{(11)} = 32 - 9 = 23$ elements. Each of $\boldsymbol{\zeta}_i^{\beta(12)}$ and $\boldsymbol{\zeta}_i^{\beta(11)}$ may also contain $\boldsymbol{\zeta}_i^{\beta(10)}$ as their first $n_\beta^{(10)} = n^{(10)} - r_b^{(10)} = 17 - 7 = 10$ elements, as will $\boldsymbol{\zeta}_i^{\delta(12)}$ by duality. The $\overline{A}_i^{(12)}$ system has $r_a^{(12)} = 11$, giving complete $\boldsymbol{\zeta}_i^{\delta(12)}$ and $\boldsymbol{\zeta}_i^{\rho(12)}$ vectors of length $n_\delta^{(12)} = n_a^{(12)} + n_Q^{(12)} - r_a^{(12)} = 15$. Also, $\overline{A}_i^{(11)}$ has $r_a^{(11)} = 10$ forming $\boldsymbol{\zeta}_i^{\delta(11)}/\boldsymbol{\zeta}_i^{\rho(11)}$ vectors of length $n_\delta^{(11)} = n_a^{(11)} + n_Q^{(11)} - r_a^{(11)} = 6$.

With no intention of solving a dedicated tenth-order model, this information may be summarised as in Table 8.1. A treatment will now be given for the solution of eleventh- and twelfth-order systems in a slightly more uniform way than for previous models. This is done due to the increased complexity of the $p^* = q-1$ case over that in Chapters 6 and 7.

8.5 Change of variables

In order to change variables to 'β_i', 'δ_i' and 'ρ_i' quantities for both $p^* = 11$ and $p^* = 12$ cases, a single

$$\Upsilon_i \boldsymbol{\alpha}_i = \boldsymbol{\upsilon}_i$$

system containing all parameters for both models will be formed to have only linear terms in the $\boldsymbol{\upsilon}_i$ vector, i.e., the 'linear' form of the previous chapters. This will contain equations

$$\beta_{ik} = {}^k\zeta_i^{\beta(12)},\ k = 1,\dots,47,\ \delta_{ik} = {}^k\zeta_i^{\delta(12)},\ k = 1,\dots,10,\ \rho_{ik} = {}^k\zeta_i^{\rho(12)},\ k = 1,\dots,10,$$

noting that $n_\beta^{(10)} = 10$ is the number of dual elements between the $W^{(10)}$ and $A_i^{(12)}$ matrices. Multi-purpose parameters

$$\varphi_i,\ i = 1,\dots,10, \text{ and } \psi_i,\ i = 11,\dots,23,$$

will be used to replace quantities representing common elements of $\boldsymbol{\zeta}_i^{\beta(12)}$, $\boldsymbol{\zeta}_i^{\beta(11)}$, $\boldsymbol{\zeta}_i^{\beta(10)}$, $\boldsymbol{\zeta}_i^{\delta(12)}$ and $\boldsymbol{\zeta}_i^{\delta(11)}$ to give the following equivalent 'linear' equation set:

$$\begin{gathered}\varphi_{ik} = {}^k\zeta_i^{\beta(12)} \equiv {}^k\zeta_i^{\beta(11)} \equiv {}^k\zeta_i^{\beta(10)} \equiv {}^k\zeta_i^{\delta(12)},\ k = 1,\dots,10,\\ \psi_{ik} = {}^k\zeta_i^{\beta(12)} \equiv {}^k\zeta_i^{\beta(11)},\ i = 11,\dots,23, \qquad (8.1)\\ \beta_{ik} = {}^k\zeta_i^{\beta(12)},\ k = 24,\dots,47,\ \rho_{ik} = {}^k\zeta_i^{\rho(12)},\ k = 1,\dots,10.\end{gathered}$$

The 'φ_i' then control:

1. the $\beta^{(10)}$ matrix,
2. rows 1–10 of the $\beta^{(11)}$ and $\beta^{(12)}$ matrices,
3. rows 1–10 of the $D_i^{(12)}$ matrices.

Also, $\varphi_i,\ i = 1, \ldots, 3$, are equivalent to rows 1–3 of $D_i^{(11)}$, and $\rho_i,\ i = 1, \ldots, 3$, are elements corresponding to the right-hand-side vector. Finally, 'ψ_i' control rows 11–23 of the $\beta^{(11)}$ and $\beta^{(12)}$ matrices. Guided by previous models, the *simplifying conditions*

$$\varphi_{ik} = \rho_{ik},\ k = 1, \ldots, 10, \tag{8.2}$$

will be imposed understanding that:

1. an embedded tenth-order model is implied in the minimum number of stages for both eleventh- and twelfth order cases,
2. eleventh- and twelfth-order 'β' matrices will be simplified, and
3. ten of the fifteen over-determined rows of $\overline{D}_i^{(12)}$ are removed, as are three of the six over-determined rows of $\overline{D}_i^{(11)}$.

Application of (8.2) thus makes the task of forcing consistency on the $\overline{A}_i^{(12)}$ systems viable, though still a considerably more complicated task than for any of the previous models. Consistency of $\overline{A}_i^{(11)}$ appears equivalent in complexity to the eighth-order cases of Chapter 7. Experience has shown that $r_\beta = 0$ solutions are not always possible (e.g., eighth-order dense for either of the models of Chapter 7), and (1) above suggests the possibility of this simplification forcing a singular case (characterised by Theorem 7.3.1). The following analysis requires a solution strategy more general than for any of the previous cases, and necessity of (8.2) has been validated in *most* cases. With such lack of rigour, this must unfortunately remain as a *simplification*, only.

System $\overline{\Upsilon}_i$ formed from equations (8.1) may now conveniently be considered in two stages, corresponding to the eleventh- and twelfth order models. Rows relating to $\beta_{ik},\ k = 24, \ldots, 47$, and $\rho_{ik}\ (= 0),\ k = 4, \ldots, 10$, may initially be removed for

$$\left(\begin{array}{cccccccccccccccccccccccccc|c}
0&0&0&0&0&0&\times&\times&\times&0&\times&\times&0&0&1&0&0&0&0&0&0&0&0&0&0&\times&(\rho_{i1}=)\,0\\
0&0&0&0&0&0&\times&\times&\times&0&\times&\times&1&0&0&0&0&0&0&0&0&0&0&0&0&\times&(\rho_{i2}=)\,0\\
0&0&0&0&0&0&\times&\times&\times&0&\times&\times&0&1&0&0&0&0&0&0&0&0&0&0&0&\times&(\rho_{i3}=)\,0\\
P_i^{0:1}&\times&P_i^{0:1}&\times&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i1}=)\,0\\
P_i^{0:1}&P_i^{0:1}&\times&\times&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i2}=)\,0\\
P_i^{0:1}&\times&\times&\times&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i3}=)\,0\\
P_i^{0:2}&\times&\times&P_i^{0:1}&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i4}=)\,0\\
P_i^{0:2}&P_i^{0:2}&\times&P_i^{0:1}&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i5}=)\,0\\
P_i^{0:2}&\times&\times&P_i^{0:1}&c_i&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i6}=)\,0\\
P_i^{0:2}&\times&\times&P_i^{0:1}&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i7}=)\,0\\
P_i^{0:2}&\times&P_i^{0:2}&P_i^{0:1}&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i8}=)\,0\\
P_i^{0:2}&\times&\times&P_i^{0:1}&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{ii9}=)\,0\\
P_i^{0:2}&\times&\times&P_i^{0:1}&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&(\varphi_{i10}=)\,0\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&c_i&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i11}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i12}\\
P_i^{0:3}&P_i^{0:3}&\times&P_i^{0:2}&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i13}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&P_i^{2:2}&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i14}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&c_i&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i15}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i16}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i17}\\
P_i^{0:3}&\times&P_i^{0:3}&P_i^{0:2}&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i18}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&c_i&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i19}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i20}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i21}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&0&c_i&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\psi_{i22}\\
P_i^{0:3}&\times&\times&P_i^{0:2}&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&\psi_{i23}
\end{array}\right)$$

Figure 8.3: ‘Linear’ $\boldsymbol{\alpha}_i$ system $(\overline{\Upsilon}_i)$ for eleventh-order dense on the RKN12(10)17M.

treatment of the eleventh-order dense method. Noting that zero-columns relating to 'α_i' not present are left in for later appending of the remaining rows, the structure of $\overline{\Upsilon}_i$ is shown in Figure 8.3, where a polynomial notation is used as in Chapter 7. In this case, the rank of $\overline{\Upsilon}_i$ is 14, and of its 26 rows, 12 are over-determined giving

$$\boldsymbol{\psi}_i = \begin{pmatrix} 0 \\ \psi_{i12} \\ 0 \\ 0 \\ 0 \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}\} \\ \psi_{i17} \\ 0 \\ 0 \\ \{\psi_{i21}, \psi_{i23}\} \\ \psi_{i21} \\ 0 \\ \psi_{i23} \end{pmatrix} \qquad \boldsymbol{\beta}_i = \begin{pmatrix} \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \{\psi_{i12}c_i, (\psi_{i23}c_i - \beta_{i44})\} \\ \beta_{i27} \\ \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \{\psi_{i17}c_i, \psi_{i21}c_i, \psi_{i23}c_i, \beta_{i44}\} \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}, \beta_{i46}, \beta_{i47}\} \\ \{\psi_{i17}c_i, (\psi_{i23}c_i - \beta_{i44})\} \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}, \beta_{i46}, \beta_{i47}\} \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}, \beta_{i46}, \beta_{i47}\} \\ \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \{\psi_{i21}c_i, \psi_{i23}c_i, \beta_{i44}\} \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}, \beta_{i46}, \beta_{i47}\} \\ \{\psi_{i21}c_i, (\psi_{i23}c_i - \beta_{i44})\} \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}, \beta_{i46}, \beta_{i47}\} \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}, \beta_{i46}, \beta_{i47}\} \\ \times \cdot (\psi_{i23}c_i - \beta_{i44}) \\ \beta_{i44} \\ \{\psi_{i17}, \psi_{i21}, \psi_{i23}, \beta_{i46}, \beta_{i47}\} \\ \beta_{i46} \\ \beta_{i47} \end{pmatrix}$$

Figure 8.4: 'ψ_i' and 'β_i' vectors for $p^* = 11$ and $p^* = 12$ on the RKN12(10)17M.

consistency conditions $\gamma_i,\ i = 1, \ldots, 12$, as linear combinations of the 'ψ_i'. Solution of these to zero requires

$$\psi_{i11} = \psi_{i13} = \psi_{i14} = \psi_{i15} = \psi_{i18} = \psi_{i19} = \psi_{i22} = 0, \tag{8.3}$$

with ψ_{i16} and ψ_{i20} determined as linear combinations of

$$\psi_{i12}, \psi_{i17}, \psi_{i21} \text{ and } \psi_{i23}. \tag{8.4}$$

The structure of the resulting $\boldsymbol{\psi}_i$ vector is indicated on the left of Figure 8.4, where terms in braces ($\{\ldots\}$) again denote a rational integer linear combination of the quantities indicated. The (now consistent) $\overline{\Upsilon}_i$ system may be termed $\overline{\Upsilon}_i^{(11)}$ for future reference, having two 'α_i' free.

Now continuing to treat the complete $\overline{\Upsilon}_i$ system, containing all of (8.1) with the 'ψ_i' specified as above, rows relating to β_{ik}, $k = 24, \ldots, 47$, and ρ_{ik} $(= 0)$, $k = 4, \ldots, 10$, may now be appended onto $\overline{\Upsilon}_i^{(11)}$, the system then having rank 25. This can be made consistent as before by determining the β_i's as linear combinations of

$$\psi_{i12}, \psi_{i17}, \psi_{i21}, \psi_{i23},\ \beta_{i27}, \beta_{i44}, \beta_{i46} \text{ and } \beta_{i47}. \tag{8.5}$$

The structure of the resulting $\boldsymbol{\beta}_i$ vector is indicated on the right of Figure 8.4, where '$\times$' now refers to a rational integer. The consistent form of $\overline{\Upsilon}_i$ for the twelfth-order case may be termed $\overline{\Upsilon}_i^{(12)}$, having a single '$\alpha_i$' free. The final p^* specific part of the analysis is in the treatment of 'nonlinear' equations to be appended onto the consistent $\overline{\Upsilon}_i^{(11)}$ and $\overline{\Upsilon}_i^{(12)}$ systems. This is therefore handled separately in the following sections.

8.6 Solution for eleventh-order dense

Appending the 'nonlinear' equations

$$\delta_{ik} = {}^{k}\zeta_i^{\delta(11)},\ \rho_{ik} = {}^{k}\zeta_i^{\rho(11)},\ k = 4, \ldots, 6,$$

to $\overline{\Upsilon}_i^{(11)}$, performing a full 16 pivoting operations specifies the final two 'α_i' (by acquisition of a full-rank system to be solved numerically), leaving 4 consistency conditions which may be solved linearly for

$$\delta_{i4}, \delta_{i5}, \delta_{i6} \text{ and } \rho_{i4},$$

in terms of (8.4), c_i, ρ_{i5} and ρ_{i6}. A solution with $r_\beta^{(11)} = 0$ would clearly require $\psi_{i12} = \psi_{i17} = \psi_{i21} = \psi_{i23} = 0$, and yield a solution with $s^* = s^b + n_q^{(11)} + r_b^{(11)} - 1 = 25$. In this case, however, $\boldsymbol{\rho}_{19} = \mathbf{0}$ is required for consistency of $\overline{D}_{19}^{(11)}$ and forces $\boldsymbol{\delta}_{19} = \mathbf{0}$, therefore $\boldsymbol{\rho}_{20} = \mathbf{0}$ from consideration of $\overline{D}_{20}^{(11)}$, giving $\boldsymbol{\delta}_{20} = 0$. Inevitably, $\boldsymbol{\delta}_i = 0$ and $D_i^{(12)}$ has rank zero $\forall i$ — hence no solution is possible for $\overline{W}^{(12)}$ by Theorem 7.3.1 with $q = 12$ and $p = 10$. This is exactly the same form of singularity found for the 19-stage model for eighth-order dense on the RK8(7)13M. Also, note this discounts an $r_\beta^{(12)} = 0$ solution for $p^* = 12$ by the consistent approach.

Now adding a stage, giving $s^* = 26$, $r_\beta^{(11)} = 1$ can be forced with $\lambda_{19} = 1$ and

$$\psi_{ik} = \lambda_i \psi_{19k}, \ k \in \{12, 17, 21, 23\}.$$

For consistency of $\overline{D}_{19}^{(11)}$, setting $\rho_{19,5} = \rho_{19,6} = 0$ leaves only $\rho_{19,4}$ to be zeroed with $\psi_{19,12}$, say. A simplification will now be imposed to reduce the complexity of consistency conditions for $\overline{D}_{20}^{(11)}$ and $\overline{D}_{21}^{(11)}$. Writing $\delta_{19,4}$ in terms of c_{19}, $\delta_{19,5}$ and $\delta_{19,6}$ (which themselves are rational c_{19} polynomial linear combinations of $\psi_{19,17}$, $\psi_{19,21}$ and $\psi_{19,23}$) as:

$$\delta_{19,4} = \frac{\delta_{19,5}\,(140\,c_{19} + 21) - \delta_{19,6}\,(420\,c_{19} + 140)}{21\,c_{19} + 1},$$

it should be clear that if one of $\delta_{19,4}$, $\delta_{19,5}$ or $\delta_{19,6}$ were zero, any ratio of those remaining would be a rational of c_{19} polynomials only, i.e., would not contain any of the ψ_{19}'s. This is highly beneficial, as the Gaussian-elimination process for determining consistency conditions will not propagate free parameters other than c_{19}. So, solving $\delta_{19,6}$ to zero with $\psi_{19,17}$,

$$\delta_{19,4} = 660591053039006127269\,(20\,c_{19} + 3)\,\gamma,$$

and

$$\delta_{19,5} = 94370150434143732467\,(21\,c_{19}+1)\,\gamma,$$

where

$$\gamma = \frac{253665771121401521861 7\,\psi_{19,23} + 181273004847705164 8\,\psi_{19,21}}{4220623582313080008951994428126361350 00\,(c_{19}^2+19\,c_{19}+1)}.$$

Note that $\gamma = 0$ would give $\boldsymbol{\delta}_{19} = 0$, and cause 'collapsing' to a singular case as for the 25-stage model[1]. Now considering $\overline{D}_{20}^{(11)}$, as $\delta_{19,6} = 0$, consistency must require $\rho_{20,6} = 0$. A single step of Gaussian-elimination then gives a single condition to be solved with $\rho_{20,5}$. Finally, $\overline{D}_{21}^{(11)}$ may be made consistent by choosing $\rho_{6,21}$, and the model is complete.

At present, 54 free parameters are available for an optimisation procedure. While such a large number is, in principal, an advantage, this led to a simplification being made for 'prototype' formulae which, in fact, was never removed and is thus common to both of the formulae to be presented. Choice of such a simplification was essentially arbitrary, but was based on the zeroing of 'α' quantities. To do this, ρ_{i5} and ρ_{i6} for $i > 21$ were chosen to zero $\alpha_{ik},\ k = 1,\ldots,9,\ i = 22,\ldots,26$ (see Section 8.2). Note that this was done primarily to zero $\boldsymbol{\zeta}_i^{b(11)},\ i = 22,\ldots,26$, and that further choice of $\lambda_i = 0,\ i = 22,\ldots,26$, would also yield $\alpha_{ik},\ k = 10,\ldots,26,\ i = 22,\ldots,26$. Such a choice of ρ_{i5} and ρ_{i6} results in large polynomial expressions containing both c_{19} and general c_i, as well as free parameters $\psi_{19,21}$ and $\psi_{19,23}$, and so will not be shown here. The final list of 44 free parameters is now

$$c_{19}, c_{20}, c_{21}, c_{22}, c_{23}, c_{24}, c_{25}, c_{26},\ \lambda_{20}, \lambda_{21}, \lambda_{22}, \lambda_{23}, \lambda_{24}, \lambda_{25}, \lambda_{26},\ \psi_{19,21}, \psi_{19,23},\ \rho_{21,5},$$

$$a_{19,15}, a_{19,17}, a_{20,15}, a_{20,17}, a_{21,15}, a_{21,17}, a_{22,15}, a_{22,17}, a_{23,15}, a_{23,17}, a_{23,22}, a_{24,15}, a_{24,17},$$

$$a_{24,22}, a_{24,23}, a_{25,15}, a_{25,17}, a_{25,22}, a_{25,23}, a_{25,24}, a_{26,15}, a_{26,17}, a_{26,22}, a_{26,23}, a_{26,24}, a_{26,25},$$

[1]Many comparisons can be drawn between this and the 20-stage model for eighth-order dense on the RK8(7)13M (Section 7.3.4).

where it can be seen that rank-deficiency of the $T_a^{(11)}$ matrix yields a_{i15} and a_{i17} free $\forall i$. Choice of these parameters for finalisation of a $p^* = 11$ dense output formula will be left to Section 8.8, where equivalent treatment will be given for the $p^* = 12$ case — described next.

8.7 Solution for twelfth-order dense

Appending the 'nonlinear' equations

$$\delta_{ik} = {}^k\zeta_i^{\delta(12)}, \;\; \rho_{ik} = {}^k\zeta_i^{\rho(12)}, \;\; k = 11, \ldots, 15,$$

to $\overline{\Upsilon}_i^{(12)}$, performing a full 26 pivoting operations specifies the final 'α_i', leaving 9 consistency conditions which may be solved linearly for

$$\delta_{i11}, \delta_{i12}, \delta_{i13}, \delta_{i14}, \delta_{i15}, \;\; \rho_{i11}, \rho_{i12}, \rho_{i13}, \text{ and } \rho_{i14},$$

in terms of (8.5), c_i and ρ_{i15}.

An $r_\beta^{(12)} = 0$ solution, requiring $s^* = s^b + n_q^{(12)} + r_b^{(12)} - 1 = 27$, must necessarily fail as did the eleventh-order case (Section 8.6). With reference to Figure 8.4, $r_\beta^{(12)} = 1$ ($s^* = 28$) must require conditions for a single independent column, i.e., $\lambda_{19} = 1$,

$$\psi_{ik} = \lambda_i \psi_{19k}, \;\; k \in \{12, 17, 21, 23\}, \;\; \beta_{ik} = \lambda_i \beta_{19k}, \;\; k \in \{27, 44, 46, 47\}.$$

Viability of this case can be tested by examining residuals formed by $\boldsymbol{\beta}_i - \lambda_i \boldsymbol{\beta}_{19}$. Such a vector must be zero for $r_\beta^{(12)} = 1$, and simply contains $\lambda_i(c_i - c_{19})$ multiplied by the c_i component of $\boldsymbol{\beta}_i$, thus requiring either $\psi_{19,12} = \psi_{19,17} = \psi_{19,21} = \psi_{19,23} = 0$, or one of

$$\lambda_i = 0 \text{ or } c_i = c_{19}, \;\; \forall i. \tag{8.6}$$

The former condition must be discounted as above, but (8.6) is a possible case. Setting $\rho_{19,15} = 0$, $\overline{D}_{19}^{(12)}$ consistency requires $\rho_{19,k}$, $k = 11, \ldots, 14$ to be solved to zero. This can be done linearly with $\psi_{19,12}$, $\beta_{19,27}$, $\beta_{19,44}$, and $\beta_{19,46}$ constrained in terms of c_{19}, $\psi_{19,17}$, $\psi_{19,21}$, $\psi_{19,23}$ and $\beta_{19,47}$. Forming $\overline{D}_{20}^{(12)}$ and eliminating once gives four consistency conditions, one of which may be solved for $\rho_{20,15}$. The remaining three, γ_i, $i = 1, \ldots, 3$, say, are then linear in λ_{20}, and quadratic in $\psi_{19,17}$, $\psi_{19,21}$, $\psi_{19,23}$ with high-degree c_{19} and c_{20} polynomial coefficients. Additional to these three conditions, (8.6) must be considered. Possible cases are:

1. $\underline{\lambda_{20} = 0}$. In this case, γ_i, $i = 1, \ldots, 3$, may be solved for $\psi_{19,17}$ and $\psi_{19,21}$ in terms of c_{19} and $\psi_{19,23}$, only. This case gives $\boldsymbol{\delta}_{19} = 0$ and is therefore disallowed.

2. $\underline{c_{20} = c_{19}}$. Here, the '$\gamma_i$' may be solved exactly as above $\forall \lambda_{20}$ (and is therefore disallowed) or with $\lambda_{20} = 1$. In this case, $\rho_{20,15} = 0$, and $\boldsymbol{\delta}_{20} = \boldsymbol{\delta}_{19}$, which will cause consistency of $\overline{D}_i^{(12)}$, $i > 20$ to proceed identically, i.e.,

$$c_i = c_{19}, \ \lambda_i = 1, \ \boldsymbol{\delta}_i = \boldsymbol{\delta}_{19}, \ \forall i,$$

   is the only possible solution. In this case, each $D_i^{(12)}$ has rank 1 $\forall i$, and the solution must fail by Theorem 7.3.1.

Now proceeding to seek $r_\beta^{(12)} > 1$ solutions, $s^* = 29$ is optimal, and the $\beta^{(12)}$ matrix will require a more general treatment than for previous models. The ordering of the following sections into 29- and 30-stage models is not true to the chronological order of development, as all prototype twelfth-order work was done with $s^* = 30$ but could not be abandoned in the light of subsequent practical results.

8.7.1 29-stage model

Requiring $r_\beta^{(12)} = 2$, it is necessary for both the $\boldsymbol{\psi}_i$ and $\boldsymbol{\beta}_i$ vectors to be considered. First treating $\boldsymbol{\beta}_i$ (with reference to Figure 8.4), for $\beta^{(12)}$ to have *two* independent columns must require

$$\beta_{ik} = \mu_i \beta_{19k} + \nu_i \beta_{20k}, \ k \in \{27, 44, 46, 47\}.$$

Due to the c_i component of $\boldsymbol{\beta}_i$ containing only the free ψ_i's, the residual vector

$$\boldsymbol{\beta}_i - \mu_i \boldsymbol{\beta}_{19} - \nu_i \boldsymbol{\beta}_{20},$$

in this case contains elements in only two forms, namely,

$$\{\}_i - \mu_i \{\}_{19} - \nu_i \{\}_{20} \text{ and } \{\}_i \, c_i - \mu_i \{\}_{19} \, c_{19} - \nu_i \{\}_{20} \, c_{20}, \tag{8.7}$$

where $\{\}_i$ refers to some rational integer linear combination of ψ_{i12}, ψ_{i17}, ψ_{i21} and ψ_{i23}. Expressions in either of these forms can be made zero by setting $\lambda_{19} = 1$, and

$$\psi_{ik} = \lambda_i \psi_{19k}, \; k \in \{12, 17, 21, 23\}, \tag{8.8}$$

as for the eleventh-order case. This is such that the free 'ψ_{19}' may be factored from expressions (8.7), giving only two possible residual forms, i.e.,

$$\{\}_{19} \, (\lambda_i - \mu_i - \nu_i \lambda_{20}) \text{ and } \{\}_{19} \, (c_i - \mu_i c_{19} - \nu_i \lambda_{20} c_{20}).$$

These may be solved with μ_i and ν_i (for *any* $\{\}_{19}$) to give

$$\mu_i = \frac{(c_{20} - c_i)\lambda_i}{c_{20} - c_{19}}, \; \nu_i = \frac{(c_i - c_{19})\lambda_i}{(c_{20} - c_{19})\lambda_{20}}.$$

Of course, (8.8) will yield an embedded eleventh-order formulae using the minimum possible number of 26 common function evaluations, as expected. The $\overline{D}_i^{(12)}$ systems will now be considered, making use of the 'δ-dependency' technique used in the previous section. Thus, setting $\rho_{19,15} = 0$, $\boldsymbol{\rho}_{19}$ can be zeroed with $\beta_{19,27}$, $\beta_{19,44}$, $\beta_{19,46}$ and $\psi_{19,12}$. Also, $\delta_{19,11}$ will be solved to zero with $\psi_{19,17}$ in order to yield the ratio of any pair of elements of $\boldsymbol{\delta}_{19}$ as a rational polynomial in c_{19}, only:

$$\delta_{19,11} = 0, \; \delta_{19,12} = \times \cdot \frac{(490 \, c_{19}^3 + 70 \, c_{19}^2 + 10 \, c_{19} + 1)\gamma}{(c_{19} - 2)c_{19}^2},$$

$$\delta_{19,13} = \times \cdot \frac{(70\,c_{19}^2 + 10\,c_{19} + 1)\gamma}{(c_{19}-2)c_{19}},\ \delta_{19,14} = \times \cdot \frac{(10\,c_{19}+1)\gamma}{(c_{19}-2)c_{19}},\ \delta_{19,15} = \times \cdot \frac{c_{19}\gamma}{(c_{19}-2)},$$

where symbols '×' represent rational integers, and

$$\gamma = 2536657711214015218617\,\psi_{19,23} + 1812730048477051648\,\psi_{19,21}. \tag{8.9}$$

Again, $\gamma = 0$ will inevitably yield $r_{D_i}^{(12)} = \operatorname{rank}(D_i^{(12)})$ to be zero $\forall i$, and requires the same combination of $\psi_{19,21}$ and $\psi_{19,23}$ to be zero as for the singular case of the eleventh-order model. Now, for consistency of $\overline{D}_{20}^{(12)}$, $\rho_{20,11}$ must be zeroed (as $\delta_{19,11} = 0$) by a constraint on $\rho_{20,15}$, and the remaining three consistency conditions can be solved for $\beta_{20,27}$, $\beta_{20,44}$ and $\beta_{20,46}$. The δ_{20k}, $k = 11,\ldots,15$, now contain rational polynomials in c_{19}, c_{20}, and all have a 'singular' factor of $\gamma\lambda_{20}$. If γ were zero, such a singular case would have $r_{D_i}^{(12)} = 0$, $\forall i$, or if $\lambda_{20} = 0$ would have $r_{D_i}^{(12)} = 1$, $\forall i$. Note that the ratio of any two elements of $\boldsymbol{\delta}_{20}$ contains only c_{19} and c_{20} without having to impose any *extra* simplifying condition. Now, of the three consistency conditions of $\overline{D}_{21}^{(12)}$, two may be solved with $\rho_{15,21}$ and λ_{21}. The third is a rational polynomial in c_{19}, c_{20} and c_{21}, also containing $\gamma\lambda_{20}$ as a factor. With no solution for γ or λ_{20} viable, all that remains is a constraint on one of the c_i, $i = 19,\ldots,21$. Such a condition is extremely difficult to derive, hence dismissal of the $s^* = 29$ case during initial stages in development of the $p^* = 12$ model. However, remarkably, a *linear* factor for c_{21} may be found in this huge expression and can be solved with

$$c_{21} = \frac{c_{20}\ \left({}^1P_{19}\,c_{20}^2 + {}^2P_{19}\,c_{20} + {}^3P_{19}\right)}{{}^4P_{19}\,c_{20}^2 + {}^5P_{19}\,c_{20} + 2\cdot{}^3P_{19}},$$

where

$${}^1P_{19} = 910\,c_{19}^4 + 680\,c_{19}^3 - 363\,c_{19}^2 - 9\,c_{19} - 8,$$

$${}^2P_{19} = -2100\,c_{19}^5 + 680\,c_{19}^4 - 852\,c_{19}^3 + 342\,c_{19}^2 + 11\,c_{19} + 6,$$

$${}^3P_{19} = 2\,c_{19}\,(10\,c_{19}+1)\left(70\,c_{19}^3 - 46\,c_{19}^2 + 9\,c_{19} - 2\right),$$

$${}^4P_{19} = 490\,c_{19}^4 + 720\,c_{19}^3 - 309\,c_{19}^2 - 5\,c_{19} - 6,$$

$$^{5}P_{19} = -1400\, c_{19}^5 + 710\, c_{19}^4 - 848\, c_{19}^3 + 277\, c_{19}^2 + 5\, c_{19} + 4.$$

Finally, consistency of $\overline{D}_{22}^{(12)}$ and $\overline{D}_{23}^{(12)}$ can be forced with $\rho_{15,22}$, λ_{22}, and $\rho_{15,23}$, respectively. The final set of 54 free parameters for this model is now:

$$c_{19}, c_{20}, c_{22}, c_{23}, c_{24}, c_{25}, c_{26}, c_{27}, c_{28}, c_{29},\ \ \psi_{19,21}, \psi_{19,23},\ \ \beta_{19,47}, \beta_{20,47},\ \ \lambda_{20}, \lambda_{23},$$

$$\lambda_{24}, \lambda_{26}, \lambda_{27}, \lambda_{28}, \lambda_{29},\ \ \rho_{24,15}, \rho_{25,15}, \rho_{26,15}, \rho_{27,15}, \rho_{28,15}, \rho_{29,15},\ \ a_{19,17}, a_{20,17},$$

$$a_{21,17}, a_{22,17}, a_{23,17}, a_{24,17}, a_{25,17}, a_{25,24}, a_{26,17}, a_{26,24}, a_{26,25}, a_{27,17}, a_{27,24}, a_{27,25},$$

$$a_{27,26}, a_{28,17}, a_{28,24}, a_{28,25}, a_{28,26}, a_{28,27}, a_{29,17}, a_{29,24}, a_{29,25}, a_{29,26}, a_{29,27}, a_{29,28}.$$

noting that rank-deficiency of $T_a^{(12)}$ yields a_{i17} free $\forall i$.

8.7.2 30-stage model

In this case, $r_\beta^{(12)} = 3$ can be obtained by imposing

$$\beta_{22k} = \lambda\beta_{19k} + \mu\beta_{20k} + \nu\beta_{21k},\ \ \beta_{ik} = 0,\ \ i = 23, \ldots, 30,\ \ k \in \{27, 44, 46, 47\},$$

and

$$\psi_{22k} = \lambda\psi_{19k} + \mu\psi_{20k} + \nu\psi_{21k},\ \ \psi_{ik} = 0,\ \ i = 23, \ldots, 30,\ \ k \in \{12, 17, 21, 23\}.$$

This gives

$$\boldsymbol{\psi}_i = \boldsymbol{\beta}_i = 0,\ \ i = 23, \ldots, 30, \tag{8.10}$$

and the column combination need only be imposed for stage 22. Rows of $\beta^{(12)}$ corresponding to $\boldsymbol{\psi}_i$ are now rank 3, as required, and examination of residuals

$$\boldsymbol{\beta}_{22} - \lambda\boldsymbol{\beta}_{19} - \mu\boldsymbol{\beta}_{20} - \nu\boldsymbol{\beta}_{21}$$

show 6 classes of expressions needing constraints between the 'ψ_i' for $i = 19, \ldots, 21$, all of the form

$$\{\}_{22}\, c_{22}^l - \lambda\, \{\}_{19}\, c_{19}^l - \mu\, \{\}_{20}\, c_{20}^l - \nu\, \{\}_{21}\, c_{21}^l,\ l \in \{0, 1\}\,. \tag{8.11}$$

These conditions can be solved in most generality by setting

$$\psi_{21,k} = \frac{\psi_{19,k}(c_{19} - c_{22})\lambda + \psi_{20,k}(c_{20} - c_{22})\mu}{(c_{22} - c_{21})\nu},\ k \in \{12, 17, 21, 23\}, \tag{8.12}$$

and it should be clear that (8.10) was necessary due to c_{22} appearing in expressions (8.12). Note that this implies $r_\beta^{(11)} = 2$, giving an embedded seventh-order dense in 27 common function evaluations. It has since been shown possible to derive a 30-stage $p^* = 12$ formula with $r_\beta^{(11)} = 1$ by imposing the more natural conditions (8.8) — thus not *requiring* (8.10), but instead needing a rank 3 dependency on columns of $\beta^{(12)}$ generated from $\boldsymbol{\beta}_i$ for $i = 23, \ldots, 30$. This is of no *real* concern having now shown existence of the $\mathrm{D}_{29(26)}^{12(11)}$ case.

Due to its experimental nature, this 30-stage model was developed using a slightly different, but in most respects *equivalent* formulation than that presented. This causes slight ambiguity with regard to a choice of simplification that is now to be imposed. In the early model, certain β_{ik} were chosen to zero some of the c_i coefficients in the $\boldsymbol{\beta}_i$ vector. This was thought to simplify the derivation, but was in hindsight merely an arbitrary reduction in the number of free parameters. The effect in this formulation is that ψ_{i12} and ψ_{i17} are constrained as rational integer linear combinations of ψ_{i21} and ψ_{i23}. It should be reiterated that this combination is now meaningless other than for the removal of unnecessary parameters in the following derivation. Consistency of $\overline{D}_{19}^{(12)}$ constrains $\beta_{19,27}$, $\beta_{19,44}$, $\beta_{19,46}$ and $\psi_{19,21}$ in terms of c_{19}, $\psi_{19,23}$ and $\beta_{19,47}$. In this case, $\boldsymbol{\delta}_{19}$ contains just c_{19} and $\psi_{19,23}$, and any ratio of its elements contains only c_{19}. Consistency of $\overline{D}_{20}^{(12)}$ plus $\delta_{20,11} = 0$ can be achieved with $\rho_{20,15}$, $\beta_{20,27}$, $\beta_{20,44}$, $\beta_{20,46}$ and $\psi_{20,21}$. Similarly, consistency of $\overline{D}_{21}^{(12)}$ plus $\delta_{21,11} = 0$ can be done with $\rho_{21,15}$, $\beta_{21,27}$, $\beta_{21,44}$, $\beta_{21,46}$. Finally, consistency of $\overline{D}_{22}^{(12)}$ and $\overline{D}_{23}^{(12)}$ can be solved for with

$\rho_{22,15}$, $\psi_{20,23}$ and $\rho_{23,15}$, respectively.

Similar to the $p^* = 11$ case, a simplification was made to aid the optimisation of prototype formulae. The conditions

$$\rho_{i15} = \frac{11\,c_i^{10} - 40\,c_i^9 + 54\,c_i^8 - 32\,c_i^7 + 7\,c_i^6}{210}, \; i = 24, \ldots, 30,$$

have the effect that $\alpha_{ik} = 0, \; k = 1, \ldots, 26, \; i = 24, \ldots, 30$. The resulting 52 free parameters are therefore:

$$c_{19}, c_{20}, c_{21}, c_{22}, c_{23}, c_{24}, c_{25}, c_{26}, c_{27}, c_{28}, c_{29}, c_{30}, \; \lambda, \mu, \nu, \; \psi_{19,23}, \beta_{19,47},$$

$$\beta_{20,47}, \beta_{21,47}, \; a_{19,17}, a_{20,17}, a_{21,17}, a_{22,17}, a_{23,17}, a_{24,17}, a_{25,17}, a_{25,24}, a_{26,17}, a_{26,24},$$

$$a_{26,25}, a_{27,17}, a_{27,24}, a_{27,25}, a_{27,26}, a_{28,17}, a_{28,24}, a_{28,25}, a_{28,26}, a_{28,27}, a_{29,17}, a_{29,24},$$

$$a_{29,25}, a_{29,26}, a_{29,27}, a_{29,28}, a_{30,17}, a_{30,24}, a_{30,25}, a_{30,26}, a_{30,27}, a_{30,28}, a_{30,29}.$$

8.8 Choice of free parameters

Optimisation procedures will now be detailed for dense output processes up to twelfth-order on y^* and y'^*. All eleventh-order models will also yield embedded tenth-order processes, and all twelfth-order models will allow embedded tenth- and eleventh-order methods. Except for the $p^* = 12$ case, $p^{y^*} = p^{y'^*} + 1$, and derivation of y^* formulae is automatic. The following RK-AID command file can be used to derive reduction data for the three cases:

```
Nystrom
!
title n12_#1
!
[s'] order :14, error terms #2:, split
[s] order :14, error terms #3:, split
!
b[all][2:6] = 0 & Q(3:s, 1;2) = 0 & Q(5:s, 3) = 0 & Q(6:s, 4) = 0
a(5:s, 2) = 0
!
```

```
use Q's
equation mode, full reduction
```

where '`#1`'/'`#2`'/'`#3`' should be replaced by 10/11/12, 11/12/13 and 12/13/13, for the tenth-, eleventh- and twelfth-order cases, respectively. The resulting data is summarised by the following table, the top line being of the forms i'_p/n_p and i_p/n_{p-1}, where i_p and i'_p are the number of independent terms in order p:

p^*	$i'_{11}/275$	$i'_{12}/541$	$i'_{13}/1098$	$i'_{14}/2208$	$i_{12}/275$	$i_{13}/541$	$i_{14}/1098$
10	16	40	85	167	16	40	85
11	0	26	65	138	0	26	65
12	0	0	42	105	0	26	65

Motivated by success in previous chapters, only linear deviation (continuous global error) criteria will be used for the optimisation of complete formulae. Also, continuous approximations derived for the RK8(7)13M in Chapter 7 showed an advantage in relinquishing high a_{ij}'s as free parameters by setting them to zero in order to benefit numerical stability characteristics. Preliminary work has shown this to be unsuccessful in the $p^* = 12$ cases here, but applicable to $p^* < 12$ work. It is highly unlikely that such high-accuracy formulae will be implemented only in double-precision arithmetic (assuming this to be ≈ 16 decimal figures), but some care can be taken to ensure the best possible numerical stability behaviour as always.

The dedicated eleventh-order model has been optimised to form $\mathrm{D}^{11}_{26}\mathrm{L}$ and $\mathrm{D}^{11}_{26}\mathrm{L}_2$ extensions, both using principal and secondary linear deviation quantities as in the sixth-order case of Chapter 6, the second having all free a_{ij} set to zero. Since the y^* formula can be derived automatically from calculus to give $p^{y^*} = 12$, both y^* and y'^* truncation error norms were optimised simultaneously using a composite objective function. This procedure proved to be hindered by rounding errors, in some cases converging extremely slowly, and involved large amounts of computing time — final choices of free parameters are given in Figures 8.5 and 8.6. These formulae contain $p^{y'^*} = 10$ formulae in their first 23 stages with b'^*_{17} free (from duality of $A^{(12)}_i$ and

$$c_{19} = 2.077555126745104 \times 10^{-1} \quad c_{20} = 1.118067336899909 \times 10^{-1}$$
$$c_{21} = 7.431286641981875 \times 10^{-1} \quad c_{22} = 4.503573028840555 \times 10^{-1}$$
$$c_{23} = 6.376923198006114 \times 10^{-1} \quad c_{24} = 5.276338202473543 \times 10^{-1}$$
$$c_{25} = 8.907823094198624 \times 10^{-1} \quad c_{26} = 1.345161783415166 \times 10^{-1}$$
$$\lambda_{20} = 6.101294696854401 \times 10^{1} \quad \lambda_{21} = -1.825785795356162 \times 10^{1}$$
$$\lambda_{22} = -6.321620473414337 \times 10^{-1} \quad \lambda_{23} = -6.927899097634118 \times 10^{-2}$$
$$\lambda_{24} = 5.398003512222964 \times 10^{-1} \quad \lambda_{25} = -1.055922063543048 \times 10^{-1}$$
$$\lambda_{26} = 2.533533353373047 \times 10^{-1} \quad \psi_{19,21} = -8.625199313069101 \times 10^{-3}$$
$$\psi_{19,23} = 3.795414463953944 \times 10^{-6} \quad \rho_{21,5} = -4.731030096409804 \times 10^{-6}$$

$$a_{19,15} = -1.000433318616335 \times 10^{-5} \quad a_{19,17} = 4.882027993099862 \times 10^{-6}$$
$$a_{20,15} = -2.210882338293867 \times 10^{-5} \quad a_{20,17} = -1.118224184911934 \times 10^{-5}$$
$$a_{21,15} = -8.695333982296235 \times 10^{-5} \quad a_{21,17} = 6.246754106301047 \times 10^{-5}$$
$$a_{22,15} = 1.381937749081287 \times 10^{-7} \quad a_{22,17} = 1.264255255600534 \times 10^{-7}$$
$$a_{23,15} = -6.102713743195909 \times 10^{-8} \quad a_{23,17} = 1.650593998899283 \times 10^{-7}$$
$$a_{23,22} = 6.563462102808245 \times 10^{-3} \quad a_{24,15} = -1.313558723632225 \times 10^{-7}$$
$$a_{24,17} = -3.240237734768944 \times 10^{-7} \quad a_{24,22} = -9.361299210945053 \times 10^{-3}$$
$$a_{24,23} = -1.445629922774984 \times 10^{-3} \quad a_{25,15} = -2.006828203659304 \times 10^{-7}$$
$$a_{25,17} = 2.104456531438268 \times 10^{-7} \quad a_{25,22} = 1.418783744549517d \times 10^{-3}$$
$$a_{25,23} = 7.239797479229506 \times 10^{-3} \quad a_{25,24} = -1.481131359882035 \times 10^{-3}$$
$$a_{26,15} = -1.563048969565983 \times 10^{-7} \quad a_{26,17} = 3.342632770935417 \times 10^{-8}$$
$$a_{26,22} = -2.489255740043502 \times 10^{-3} \quad a_{26,23} = 1.241020215996109 \times 10^{-3}$$
$$a_{26,24} = 2.183880161249337 \times 10^{-3} \quad a_{26,25} = 5.170658563069779 \times 10^{-5}$$

$$p_4 = -9.0767020754695520485696601263906357968 2 \times 10^{2}$$
$$p_5 = 1.2256404863802833507189496655018565219 38 \times 10^{3}$$
$$p_6 = -3.9889118889362819263514440695828569851 44 \times 10^{2}$$
$$p_7 = -8.0505829199435358685643321165242480289 88 \times 10^{2}$$
$$p_8 = 9.0097052403419567516459266105867271202 18 \times 10^{2}$$
$$p_9 = -2.7896806723310452423372217830806896856 25 \times 10^{2}$$

Figure 8.5: $\mathrm{RKN12}^{(10)}_{17}\mathrm{D}^{11(10)}_{26(23)}\mathrm{L}$ formula specification.

$$c_{19} = 2.014879378181408 \times 10^{-1} \quad c_{20} = 1.084689396510368 \times 10^{-1}$$
$$c_{21} = 7.514830210292604 \times 10^{-1} \quad c_{22} = 4.373465289514149 \times 10^{-1}$$
$$c_{23} = 6.450755241132182 \times 10^{-1} \quad c_{24} = 5.281796199242585 \times 10^{-1}$$
$$c_{25} = 8.945165528359746 \times 10^{-1} \quad c_{26} = 1.308750556048654 \times 10^{-1}$$
$$\lambda_{20} = 2.334611058998921 \times 10^{2} \quad \lambda_{21} = -1.449117101370823 \times 10^{1}$$
$$\lambda_{22} = -3.460081210447549 \times 10^{-1} \quad \lambda_{23} = -8.98540316997094 \times 10^{-2}$$
$$\lambda_{24} = 4.736311908183883 \times 10^{-1} \quad \lambda_{25} = -9.570758189433875 \times 10^{-2}$$
$$\lambda_{26} = 1.71863588898375 \times 10^{-1} \quad \psi_{19,21} = -9.95107409438961 \times 10^{-3}$$
$$\psi_{19,23} = 4.378849598582156 \times 10^{-6} \quad \rho_{21,5} = -8.568091445239221 \times 10^{-6}$$

$$p_4 = -1.193618438650662103450400058002112387268 \times 10^{3}$$
$$p_5 = 1.901516510751829644346740878578081271195 \times 10^{3}$$
$$p_6 = -1.362438891823676409226299484471121360779 \times 10^{3}$$
$$p_7 = 1.615979694079873987054164625341807425544 \times 10^{1}$$
$$p_8 = 5.150247051770655521578864897849761365564 \times 10^{2}$$
$$p_9 = -2.019707326214701808795070233414774565275 \times 10^{2}$$

Figure 8.6: $\mathrm{RKN12}_{17}^{(10)}\mathrm{D}_{26(23)}^{11(10)}\mathrm{L}_2$ formula specification.

$M^{(10)}$), a $p^{y^*} = 11$ formula being available automatically. For C^2 continuity in the 'natural' ninth-degree polynomial, b'^{*}_{17} must take the following form (with $\widehat{b}'_{17} = \frac{1}{40}$):

$$p_9\,\sigma^9 + p_8\,\sigma^8 + p_7\,\sigma^7 + p_6\,\sigma^6 + p_5\,\sigma^5 + p_4\,\sigma^4 - \frac{A^{(10)}\,\sigma^3 - B^{(10)}\,\sigma^2 + C^{(10)}\,\sigma}{9240}, \quad (8.13)$$

where

$$A^{(10)} = 117600\,p_9 + 97020\,p_8 + 77000\,p_7 + 57750\,p_6 + 39600\,p_5 + 23100\,p_4 + 3465,$$

$$B^{(10)} = 161280\,p_9 + 129360\,p_8 + 98560\,p_7 + 69300\,p_6 + 42240\,p_5 + 18480\,p_4 + 6468,$$

and

$$C^{(10)} = 52920\,p_9 + 41580\,p_8 + 30800\,p_7 + 20790\,p_6 + 11880\,p_5 + 4620\,p_4 + 2772,$$

giving six free parameters $p_i,\ i = 4, \ldots, 9$. As with the seventh-order dense on the RK8(7)13M, this embedded tenth-order formula was optimised over 'smooth' two-norms of $\tau'^{(11)*}_j$ and $\tau'^{(12)*}_j$ quantities using algebraic techniques only (see Section 7.3.6). This gave an exact optimal choice for the six free polynomial coefficients with regard to the y'^* formula, which were then compromised slightly to give smaller 'b^*' and 'b'^*' polynomial coefficients. This procedure satisfactorily controlled the automatic y^* formula, which could have been optimised concurrently using the same algebraic technique. The chosen free $p_i,\ i = 4, \ldots, 9$, are included in Figures 8.5 and 8.6.

Optimisation of the two twelfth-order models proved a much more complicated task than for any other method considered. This was not due to the large number of principal and secondary truncation error terms involved, but to the extremely high-degree polynomials involved from solution of the $\overline{D}^{(12)}_i$ systems. The 30-stage model, being experimental at the time of derivation, was hugely simplified by specification of $c_i,\ i = 19, \ldots, 22$, to simple rational values *prior* to coding. This was an unfortunate loss of generality, but felt necessary to allow any form of numerical coding

$$c_{22} = 1.656045991357656820{\times}10^{-2} \quad c_{23} = 4.556989520510072440{\times}10^{-1}$$
$$c_{24} = 1.982522519676483960{\times}10^{-1} \quad c_{25} = 7.911131732271137200{\times}10^{-1}$$
$$c_{26} = 8.848723759717065370{\times}10^{-1} \quad c_{27} = 6.806807386257398560{\times}10^{-1}$$
$$c_{28} = 4.053465529875516780{\times}10^{-1} \quad c_{29} = 9.664315201187114740{\times}10^{-1}$$
$$\psi_{19,21} = -9.718965563895949340{\times}10^{-01} \quad \psi_{19,23} = 7.393374044986880840{\times}10^{-4}$$
$$\beta_{19,47} = -1.863706644607852210{\times}10^{-2} \quad \beta_{20,47} = 2.757029376190415010{\times}10^{-4}$$
$$\lambda_{20} = -1.885020305901503190{\times}10^{-2} \quad \lambda_{23} = -3.821651850864279120{\times}10^{-4}$$
$$\lambda_{24} = 1.561124909875597490{\times}10^{-1} \quad \lambda_{25} = 5.056729684379144220{\times}10^{-5}$$
$$\lambda_{26} = -5.625384231552909180{\times}10^{-5} \quad \lambda_{27} = -5.554932661855161500{\times}10^{-5}$$
$$\lambda_{28} = -2.668563279256761370{\times}10^{-4} \quad \lambda_{29} = 7.840559500487236380{\times}10^{-5}$$
$$\rho_{24,15} = -1.162650477274668570{\times}10^{-6} \quad \rho_{25,15} = -1.482471816732256800{\times}10^{-5}$$
$$\rho_{26,15} = -1.119465830453138880{\times}10^{-5} \quad \rho_{27,15} = -9.531024961696759260{\times}10^{-6}$$
$$\rho_{28,15} = 7.801786358381010940{\times}10^{-6} \quad \rho_{29,15} = -1.548543022216709860{\times}10^{-6}$$

Figure 8.7: Partial RKN12$^{(10)}_{17}$D$^{12}_{29}$LM y'^{*} formula specification.

$$c_{21} = \tfrac{3}{10} \quad c_{22} = \tfrac{2}{5} \quad c_{23} = 4.887741983886613 \times 10^{-1}$$
$$c_{24} = 3.236510340156346 \times 10^{-2} \quad c_{25} = 4.159830577759178 \times 10^{-1}$$
$$c_{26} = 1.134282349596857 \times 10^{-1} \quad c_{27} = 8.441056902359744 \times 10^{-1}$$
$$c_{28} = 7.106911188459770 \times 10^{-1} \quad c_{29} = 9.327720966865529 \times 10^{-1}$$
$$c_{30} = 9.839329419804503 \times 10^{-1} \quad \lambda = 1.146422660161057 \times 10^{1}$$
$$\mu = -2.318907975859266 \quad \nu = 5.884242753076186$$
$$\psi_{19,23} = 7.781915839504719 \times 10^{-6} \quad \beta_{19,47} = 2.773312182805975 \times 10^{-4}$$
$$\beta_{20,47} = 5.976011713628374 \times 10^{-4} \quad \beta_{21,47} = -9.010344063055291 \times 10^{-4}$$

Figure 8.8: Partial RKN12$^{(10)}_{17}$D$^{12}_{30}$L y'^{*} formula specification.

given the machine hardware available. A larger working memory for the MACSYMA implementation, for example, may have enabled this complex process to be done in general. During optimisation of the 29-stage model, it was found viable (at great length) to release c_{22} as a further free parameter, and despite having one less stage, comparable formulae were found. Due to the availability of 3 free b^*'s relating to the y^* formula in each case, only the thirteenth- and fourteenth-order y'^* error terms were optimised by this procedure. With $b_i^*,\ i = s^* - 2, \ldots, s^*$, as free parameters, continuity requirements suggest the tenth-degree form

$$\begin{aligned} b_i^* = p_{i10}\,\sigma^{10} + p_{i9}\,\sigma^9 + p_{i8}\,\sigma^8 + p_{i7}\,\sigma^7 + p_{i6}\,\sigma^6 + p_{i5}\,\sigma^5 + p_{i4}\,\sigma^4 \\ -A^{(12)}\,\sigma^3 + B^{(12)}\,\sigma^2 - C^{(12)}\,\sigma, \end{aligned} \tag{8.14}$$

where

$$A^{(12)} = 36\,p_{i10} + 28\,p_{i9} + 21\,p_{i8} + 15\,p_{i7} + 10\,p_{i6} + 6\,p_{i5} + 3\,p_{i4},$$

$$B^{(12)} = 63\,p_{i10} + 48\,p_{i9} + 35\,p_{i8} + 24\,p_{i7} + 15\,p_{i6} + 8\,p_{i5} + 3\,p_{i4},$$

and

$$C^{(12)} = 28\,p_{i10} + 21\,p_{i9} + 15\,p_{i8} + 10\,p_{i7} + 6\,p_{i6} + 3\,p_{i5} + p_{i4},$$

thus the 21 free parameters $p_{ik},\ i = s^* - 2, \ldots, s^*,\ k = 4, \ldots, 10$, are available for a further optimisation process. Optimal choices were found from minimisation of *principal* y^* terms, only, and were in both cases compromised to give smaller 'b^*' polynomial coefficient norms. Secondary terms were found to be controlled satisfactorily by this procedure. The two formulae, denoted by D_{29}^{12}LM (for *minimal*) and D_{30}^{12}L, respectively, are specified by the free parameters given in Figures 8.7 and 8.8. In both cases, $c_{19} = \frac{1}{8}$ and $c_{20} = \frac{1}{6}$ were the common arbitrarily chosen nodes. For brevity here, free a_{ij} and p_{ik} parameters should be read from the complete method data-files in Sections B.3.3 and B.3.4. It should be noted that, for the first time in this work, all method coefficients relating to the RKN12(10)17M formula have not been derived as exact rationals, but only to a high-precision — 60 decimal places

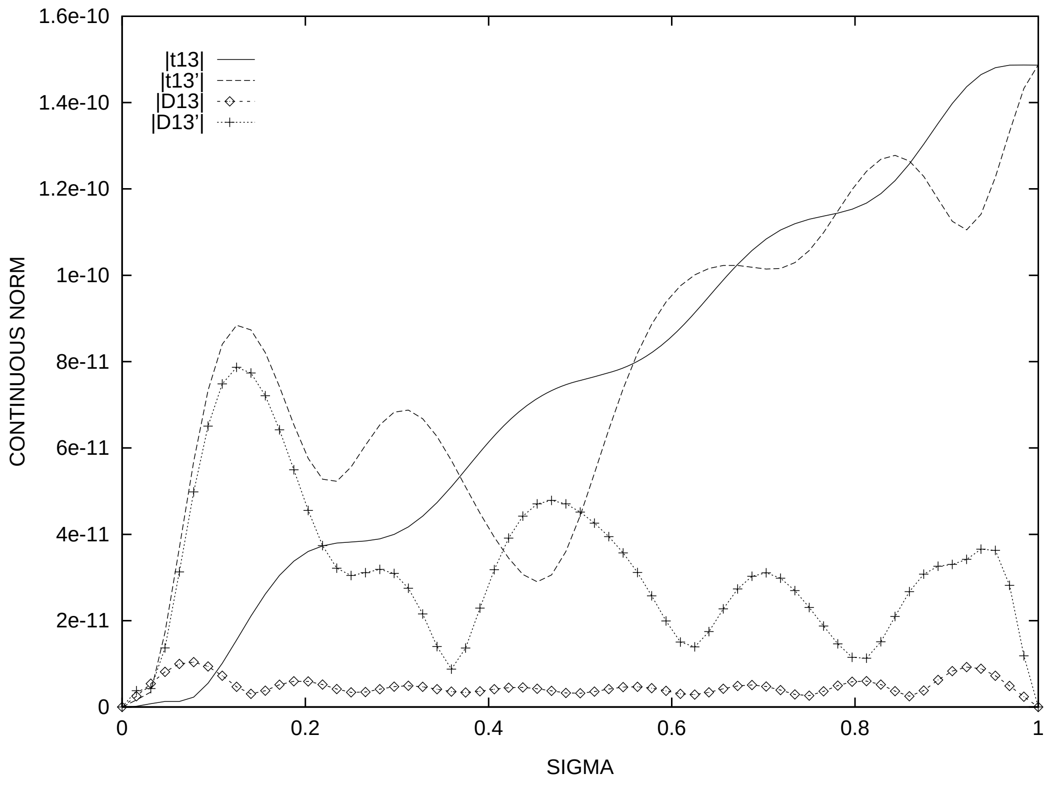


Figure 8.9: Principal truncation error and linear deviation norms for the D_{29}^{12}LM.

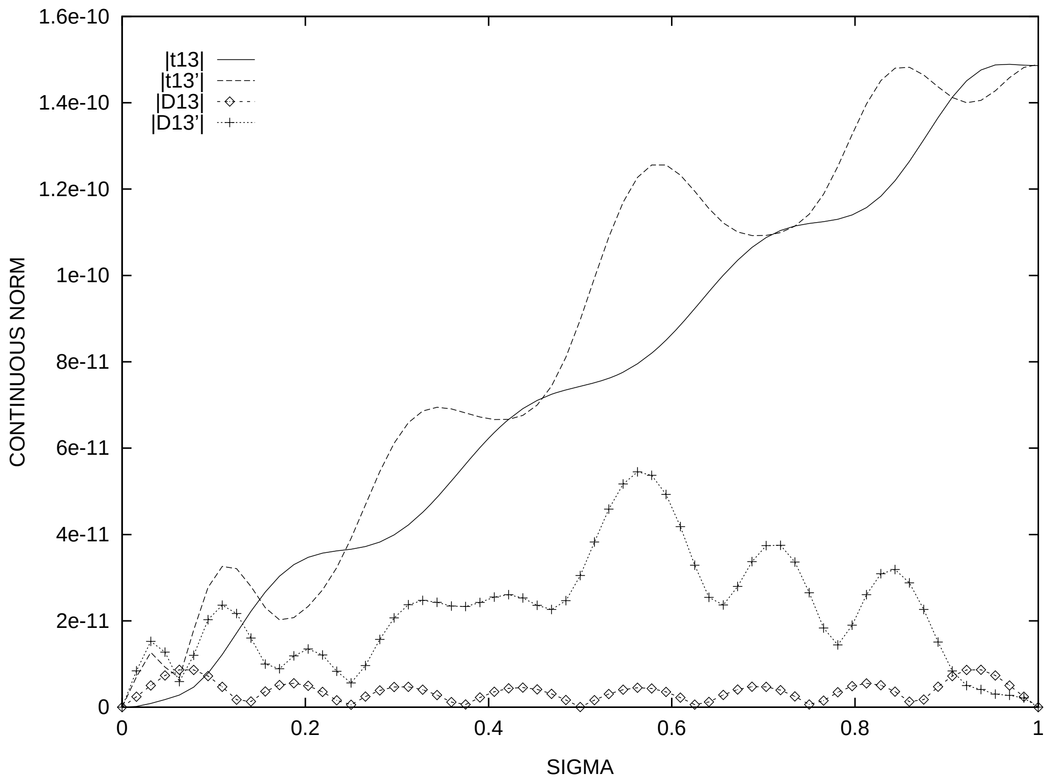


Figure 8.10: Principal truncation error and linear deviation norms for the D_{30}^{12}L.

throughout (given to 40 decimal places in the files of Appendix B). Note that no tenth- and eleventh-order formulae embedded within the twelfth-order cases have been given here. The eleventh-order formulae in 26- and 27-stages for the two cases

	$N'^{(11)}$	$N'^{(12)}$	$N_l'^{(13)}$	$N_l'^{(14)}$
$\mathrm{D}_{23}^{10}\mathrm{L}$	1.49×10^{-9}	3.55×10^{-9}	6.01×10^{-9}	7.62×10^{-9}
$\mathrm{D}_{23}^{10}\mathrm{L}_2$	1.51×10^{-9}	3.57×10^{-9}	6.27×10^{-9}	8.01×10^{-9}
$\mathrm{D}_{26}^{11}\mathrm{L}$	0	2.36×10^{-10}	6.15×10^{-10}	9.61×10^{-10}
$\mathrm{D}_{26}^{11}\mathrm{L}_2$	0	3.17×10^{-10}	8.18×10^{-10}	1.26×10^{-9}
$\mathrm{D}_{29}^{12}\mathrm{LM}$	0	0	3.18×10^{-11}	1.09×10^{-10}
$\mathrm{D}_{30}^{12}\mathrm{L}$	0	0	2.21×10^{-11}	7.64×10^{-11}

Table 8.2: Integrated y'^* linear deviation norms for six RKN12(10)17M triples.

	$N^{(12)}$	$N_l^{(13)}$	$N_l^{(14)}$
$\mathrm{D}_{23}^{10}\mathrm{L}$	1.78×10^{-10}	3.98×10^{-10}	6.47×10^{-10}
$\mathrm{D}_{23}^{10}\mathrm{L}_2$	1.79×10^{-10}	4.10×10^{-10}	6.88×10^{-10}
$\mathrm{D}_{26}^{11}\mathrm{L}$	0	3.11×10^{-11}	7.92×10^{-11}
$\mathrm{D}_{26}^{11}\mathrm{L}_2$	0	4.10×10^{-11}	1.03×10^{-10}
$\mathrm{D}_{29}^{12}\mathrm{LM}$	0	4.74×10^{-12}	1.78×10^{-11}
$\mathrm{D}_{30}^{12}\mathrm{L}$	0	3.70×10^{-12}	1.88×10^{-11}

Table 8.3: Integrated y^* linear deviation norms for six RKN12(10)17M triples.

were found not to be competitive with the truncation error norms of the (optimised) dedicated methods, despite great lengths to code software for optimisation of the embedded cases concurrently with the 29- and 30-stage models. Testing during development verified the inferiority of these embedded processes, and unfortunately the twelfth-order models should only be considered when using *all* the 29- and 30-stages. Similar results were found from the optimised tenth-order formulae available using the first 23 stages of the $p^* = 12$ models, though differences were much less defined. Note that all these embedded processes are available, but will not be presented or tested here. Integrated linear deviation norms up to 14th order are given for the six finished processes in Figures 8.2 and 8.3, and principal terms are shown plotted for $\mathrm{D}^{12}_{30}\mathrm{L}$ and $\mathrm{D}^{12}_{29}\mathrm{LM}$ cases in Figures 8.9 and 8.10.

8.9 Numerical tests

The same problem set will be used here as in Chapter 6, where, as for the RK case in Chapter 7, the RK-AID (Nyström) test suite has been coded as a separate application to allow quadruple-precision (≈ 32 decimal figure) arithmetic. SBD results assumed free of any significant rounding errors have thus been obtained, and for a selection of three problems are plotted using a logscale in Figures 8.11–8.13. Average results over all six test problems are shown in Figures 8.14 and 8.15. Also, effects of rounding error in double-precision arithmetic (using the RK-AID test suite) can be found in Figures 8.16 and 8.17.

These results show a good correlation between (experimental) box deviation and (theoretical) linear deviation quantities, and illustrate clear qualitative differences between the varying orders. Note that the tenth-order methods have $p^{y^*} = q-1 = 11$, and so give global accuracy to match the discrete points on y.

The $p^* = 11$ models can be seen as competitive with the $p^* = 12$ cases on y output, for tolerances $> 1{\times}10^{-12}$, and any benefit from the three extra stages appears negligible for these accuracy requirements. Of the two eleventh-order methods, extra degrees of freedom in not having free a_{ij} zero give slight improvement in output quality, but appear to benefit numerical stability for the $\mathrm{D}^{11}_{26}\mathrm{L}_2$ and embedded $\mathrm{D}^{10}_{23}\mathrm{L}_2$, on y'. Unfortunately, the $\mathrm{D}^{11}_{26}\mathrm{L}$ is slightly more stable on y output (without the `FLOAT` scheme), an effect not predicted by theory. Tentatively, the $\mathrm{D}^{11(10)}_{26(23)}\mathrm{L}_2$ may be preferred from numerical stability properties, and is presented in Section B.3.2.

From the average results, twelfth-order dense output on y' is clearly superior for tolerances $< 1{\times}10^{-9}$, and in the case of the $\mathrm{D}^{12}_{30}\mathrm{L}$ is better (or comparable) for *all* tolerances except $1{\times}10^{-4}$. A difficulty arises with the need for a thirtieth stage for this method, as the two twelfth-order formulae can not be compared fairly. Note that all continuous processes can theoretically be improved by the addition of (unnecessary)

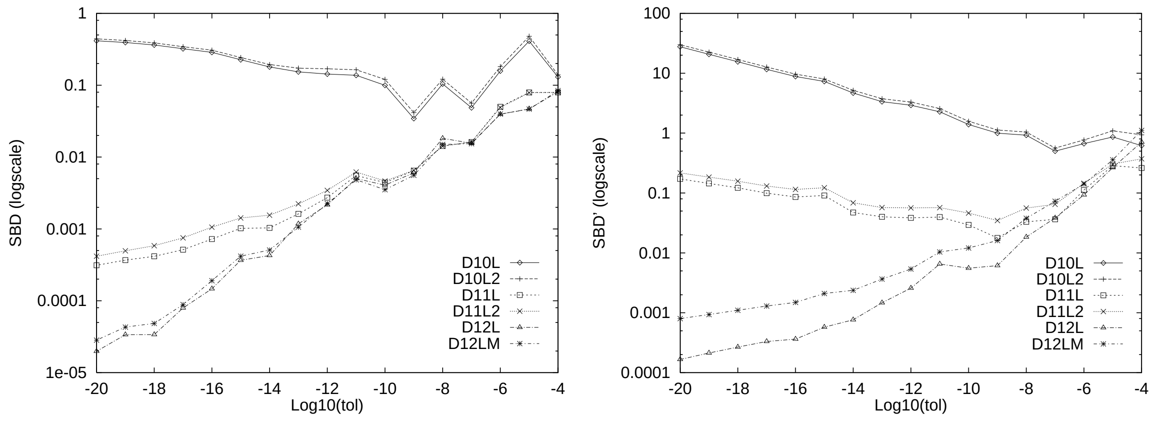


Figure 8.11: SBD/SBD′ on orbit D1 for six RKN12(10)17M triples.

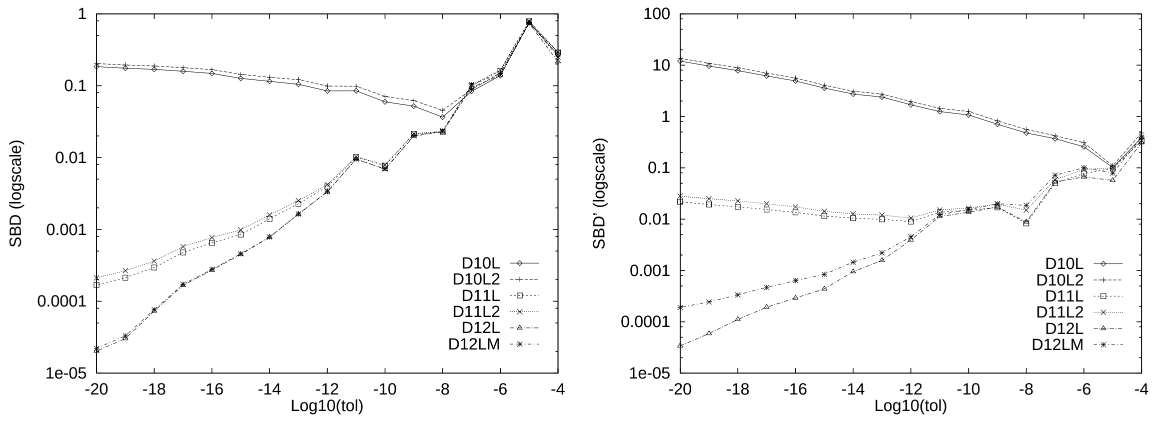


Figure 8.12: SBD/SBD′ on RK-AID problem 11 for six RKN12(10)17M triples.

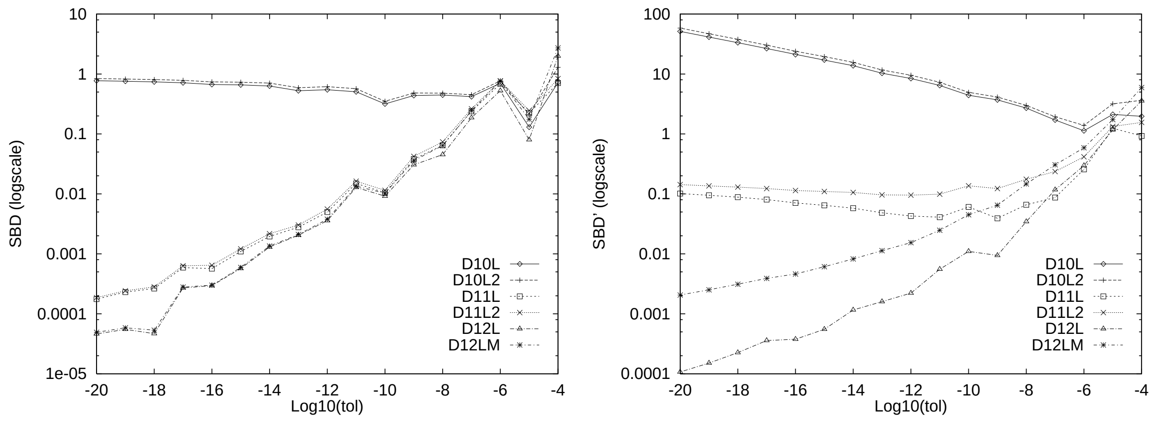


Figure 8.13: SBD/SBD′ on RK-AID problem 12 for six RKN12(10)17M triples.

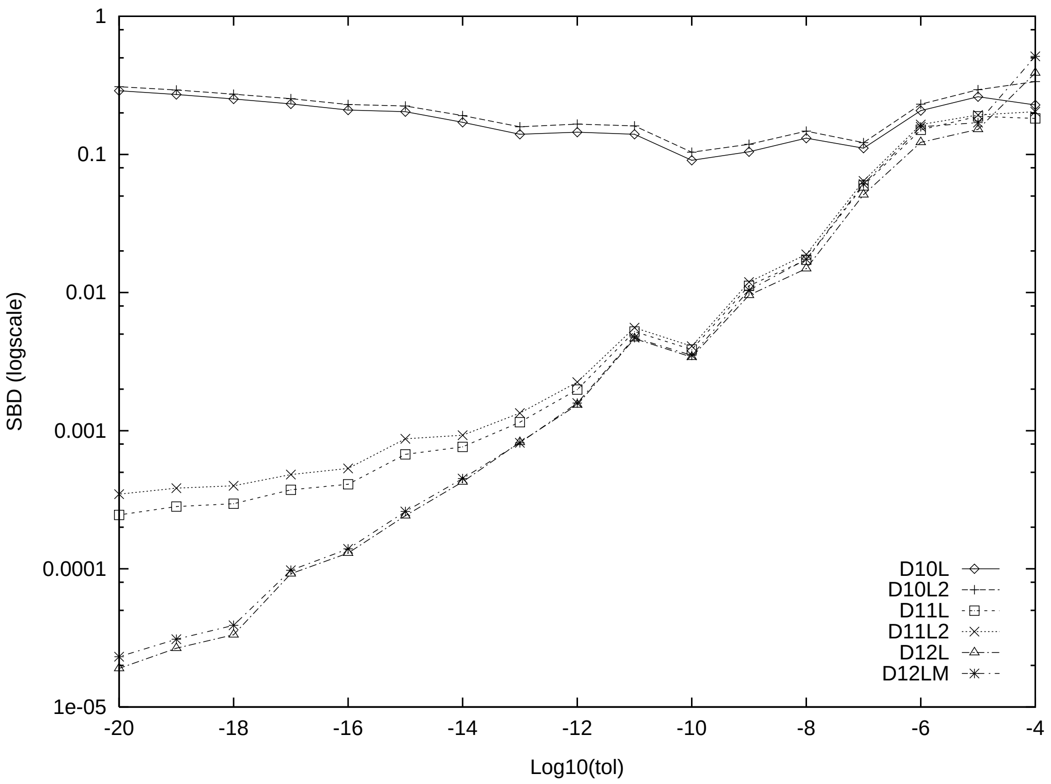


Figure 8.14: $\overline{\text{SBD}}$ over six test problems for six RKN12(10)17M triples.

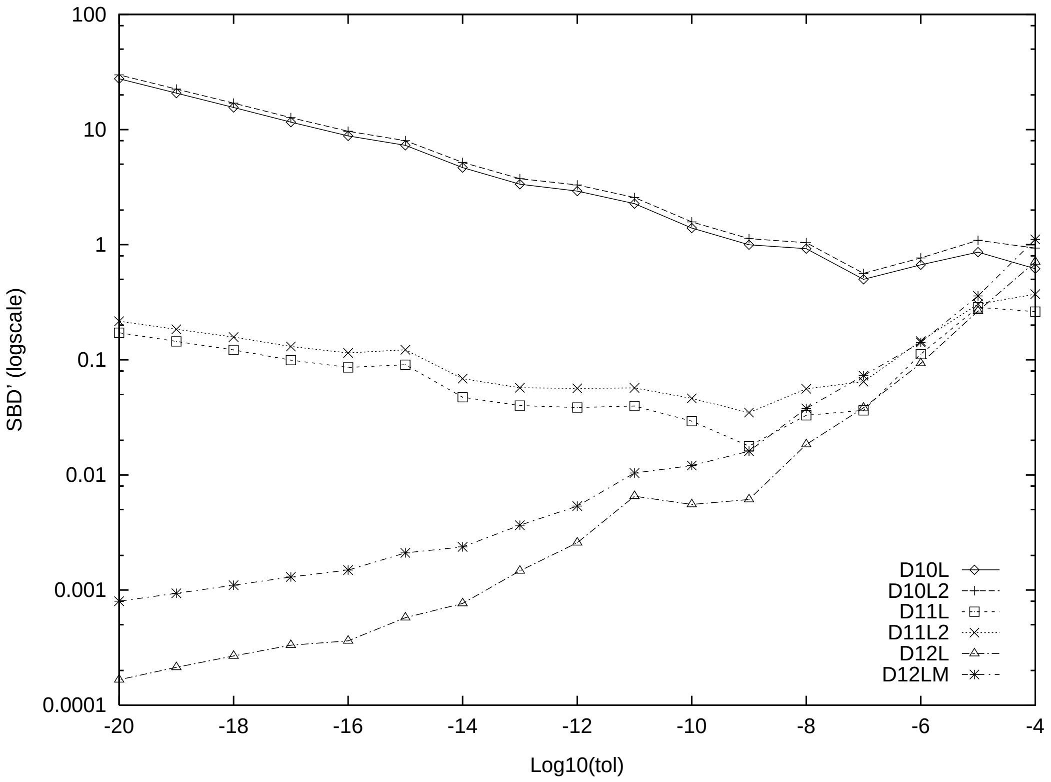


Figure 8.15: $\overline{\text{SBD}}'$ over six test problems for six RKN12(10)17M triples.

Double precision:

	10 %	10^2 %	10^3 %	10^4 %	10^5 %	10^6 %	10^7 %	10^8 %
$D_{23}^{10}L$	10^{-9}	10^{-10}	10^{-11}	10^{-13}	–	–	–	–
$D_{23}^{10}L_2$	10^{-10}	10^{-11}	10^{-12}	10^{-14}	–	–	–	–
$D_{26}^{11}L$	10^{-9}	10^{-10}	10^{-10}	10^{-11}	10^{-12}	10^{-14}	–	–
$D_{26}^{11}L_2$	10^{-8}	10^{-9}	10^{-10}	10^{-10}	10^{-11}	10^{-12}	10^{-14}	–
$D_{30}^{12}L$	10^{-9}	10^{-9}	10^{-10}	10^{-10}	10^{-11}	10^{-12}	10^{-14}	–
$D_{29}^{12}LM$	10^{-8}	10^{-9}	10^{-10}	10^{-10}	10^{-11}	10^{-12}	10^{-14}	–

Double precision with polynomial precision extension:

	10 %	10^2 %	10^3 %	10^4 %	10^5 %	10^6 %	10^7 %	10^8 %
$D_{23}^{10}L$	10^{-12}	–	–	–	–	–	–	–
$D_{23}^{10}L_2$	10^{-12}	–	–	–	–	–	–	–
$D_{26}^{11}L$	10^{-12}	10^{-12}	10^{-12}	10^{-15}	–	–	–	–
$D_{26}^{11}L_2$	10^{-12}	10^{-12}	10^{-12}	10^{-15}	–	–	–	–
$D_{30}^{12}L$	10^{-12}	10^{-12}	10^{-12}	10^{-13}	10^{-13}	–	–	–
$D_{29}^{12}LM$	10^{-12}	10^{-12}	10^{-12}	10^{-13}	10^{-15}	–	–	–

Figure 8.16: Onset of %'age $\overline{\text{SBD}}$ rounding error for six RKN12(10)17M triples.

Double precision:

	10 %	10^2 %	10^3 %	10^4 %	10^5 %	10^6 %	10^7 %	10^8 %
$D_{23}^{10}L$	10^{-10}	10^{-11}	10^{-11}	–	–	–	–	–
$D_{23}^{10}L_2$	10^{-10}	10^{-11}	10^{-13}	–	–	–	–	–
$D_{26}^{11}L$	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}	10^{-14}	–	–
$D_{26}^{11}L_2$	10^{-9}	10^{-9}	10^{-10}	10^{-11}	10^{-12}	10^{-14}	–	–
$D_{30}^{12}L$	10^{-8}	10^{-8}	10^{-9}	10^{-9}	10^{-10}	10^{-11}	10^{-12}	10^{-14}
$D_{29}^{12}LM$	10^{-8}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}	10^{-12}	10^{-14}

Double precision with polynomial precision extension:

	10 %	10^2 %	10^3 %	10^4 %	10^5 %	10^6 %	10^7 %	10^8 %
$D_{23}^{10}L$	10^{-13}	–	–	–	–	–	–	–
$D_{23}^{10}L_2$	10^{-13}	–	–	–	–	–	–	–
$D_{26}^{11}L$	10^{-12}	10^{-13}	–	–	–	–	–	–
$D_{26}^{11}L_2$	10^{-12}	10^{-15}	–	–	–	–	–	–
$D_{30}^{12}L$	10^{-12}	10^{-12}	10^{-12}	10^{-15}	10^{-15}	–	–	–
$D_{29}^{12}LM$	10^{-12}	10^{-12}	10^{-13}	10^{-15}	–	–	–	–

Figure 8.17: Onset of %'age $\overline{\text{SBD}}'$ rounding error for six RKN12(10)17M triples.

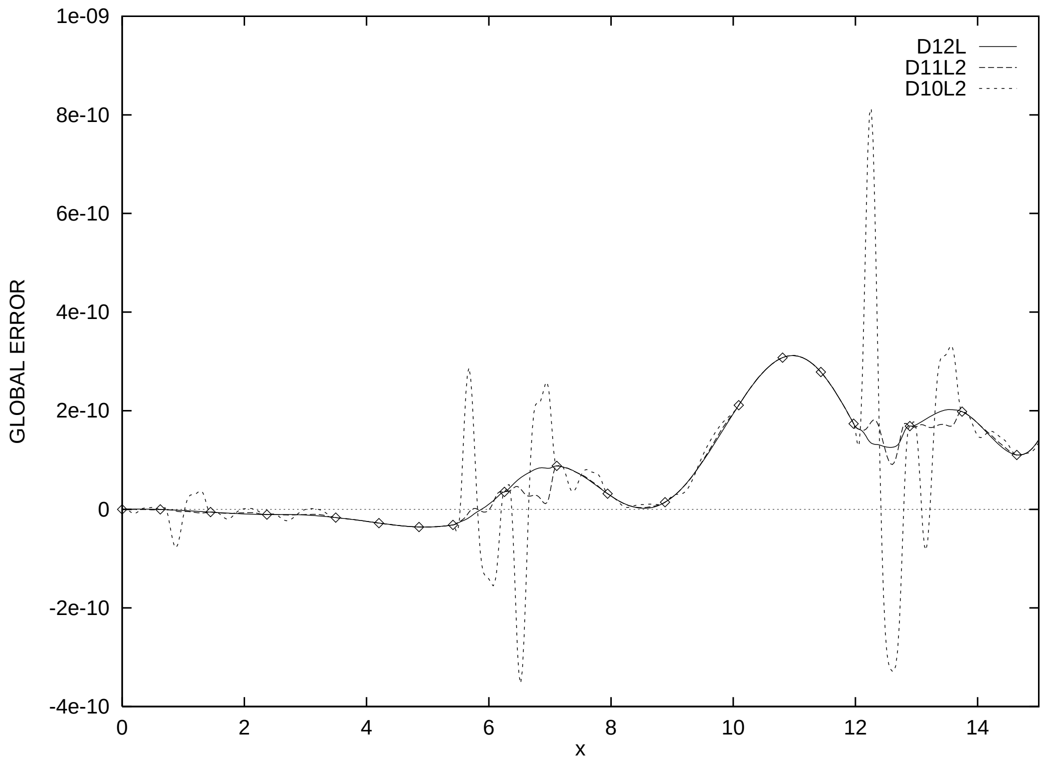


Figure 8.18: Global error on ${}^{1}y^{*}$ for a D1 orbit using three RKN12(10)17M triples.

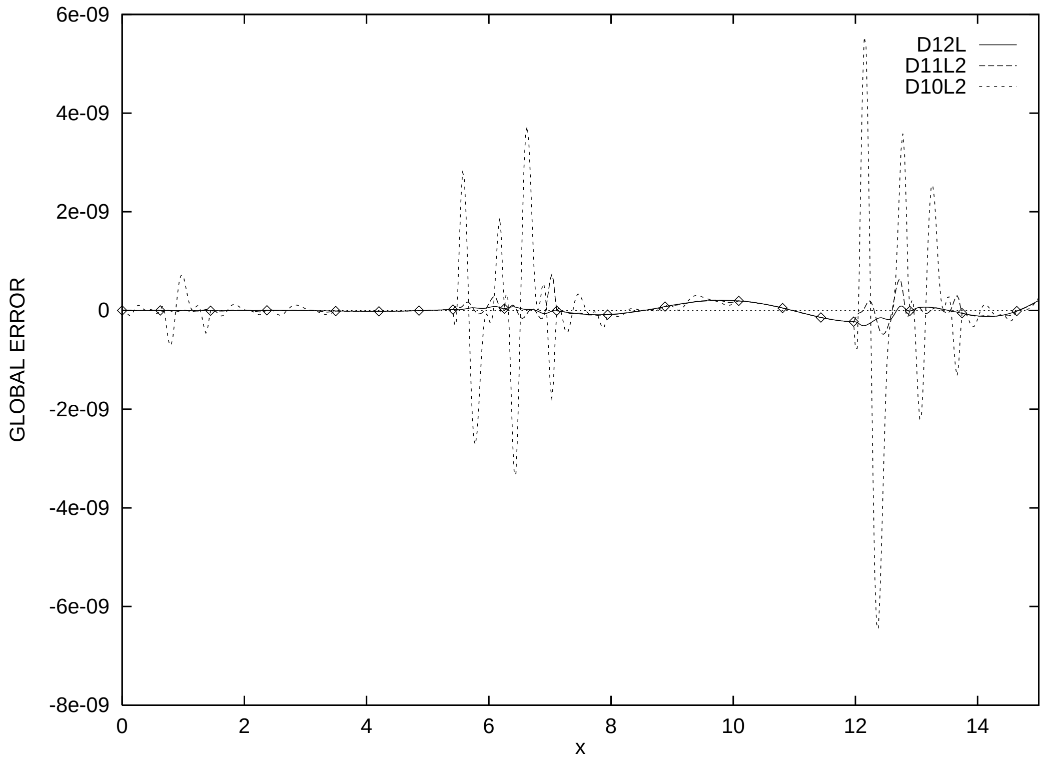


Figure 8.19: Global error on ${}^{1}y'^{*}$ for a D1 orbit using three RKN12(10)17M triples.

extra stages, and a measure of efficiency is ideally needed. In this case, one extra stage out of thirty does not seem to merit further analysis, and both methods have been presented without expressing any preference in Sections B.3.3 and B.3.4.

For illustration purposes, a continuous global error plot from the D1 orbit problem results, with tolerance 1×10^{-8}, is shown in Figures 8.18 and 8.19.

Chapter 9

Conclusion

9.1 Thesis summary

The main purpose of the work in Chapters 3, 5, 6, 7 and 8 has been the study of dense output formulae through derivation and testing. Primarily, the generalisation of common stages has been achieved through software applications in Fortran 90 and MACSYMA. The provision of continuous extensions has been shown possible with a unifying initial strategy for presenting the order conditions as a dual set of linear systems ($\overline{A}_i$ and $\overline{W}$). What has been termed 'pre-analysis' is the ability to eliminate fixed stages from these systems, then allowing a change of variables into a particularly succinct form. This is such that the requirements for deriving methods of a given order in the minimum number of extra stages is clear, and importantly, these initial stages can be automated first by use of the RK-AID algorithms and then by symbolic linear algebra. All method dependent parts of the solution process then follow, and many parallels have been found between the different methods. Not least, the rejection of singular cases has been characterised in such a way that the need for extra stages (above the theoretical minimum) has been obvious from simple properties of the principal over-determined sub-systems $\overline{D}_i$. A substantial benefit in generality of the models has proved to be successful optimisation with large numbers of degrees of freedom to 'fine tune' continuous processes with carefully chosen practical requirements. The correlation between theoretical local error characteristics and continuous global error in practice has been shown sensitive to free parameters.

This makes the optimisation stage valuable, and maintaining as many degrees of freedom in the coding of error norms must be considered as a necessary effort.

The RK-AID package has evolved into a tool eventually 'paying for itself' in time saved during development of the twelfth-order Nyström cases. Particularly, the simplification of linear dependencies in error terms and output of an independent set for numerical coding has been invaluable. The initial philosophy when working with existing formulae was for generalising all RK-AID algorithms beyond any obvious order-barrier. This enabled the huge numbers of error terms involved in optimisation of principal and secondary terms for high-order methods to be considered with no extra effort than for low-order cases.

9.2 Discussion

The provision of continuous extensions has been considered in as much generality as possible via the solution of continuous order conditions, but it must be recognised that this is equivalent to fitting simple polynomial interpolants to the discrete RK(N) function evaluations and extra intermediate results of correct asymptotic order. In this respect, properties of (and patterns in) the β and $\overline{D}_i$ matrices could have been exploited further. For the eighth-order RK and twelfth-order RKN cases, it is apparent that the polynomial nature (in c_i) of the $\boldsymbol{\beta}_i$ vectors after the change of variables relates in most cases to the underlying embedding of lower-order interpolants. In all cases except the RK8(6)12M, treatment of the coefficients of c_i in $\boldsymbol{\beta}_i^{(q)}$ corresponded to analysis of $p^* = q-1$ denses. For example, with reference to Figure 8.4, coefficients of c_i in $\boldsymbol{\beta}_i$ were the free 'ψ_i' relating to $\beta^{(11)}$, and inability to set all these to zero to a give a 'stage independent' solution is undoubtedly due to the non-existence of eleventh-order dense in 25 stages. While this result is therefore not at all surprising, it could presumably have been predicted by the algebra given a deeper understanding of underlying principals. Further to this, twelfth-order dense output in 29 stages required very clear sub-conditions enforcing eleventh-order output in 26 stages (8.8). With $s^* = 30$, one extra stage than the minimum, it was necessary to have eleventh-order dense using less than 28 common evaluations (8.12). In some sense, it therefore seems necessary to have at least three function evaluations of an eleventh-order result to achieve $p^* = 12$ — an independent analysis based upon interpolation theory may have more simply shown this to be necessary. Additionally, the appearance of singular cases that were easily characterised by a single theorem (7.3.1) relating to the continuous order p models *only* is a further indication of unknown order-barriers relating to the same interpolation theory. Similar general questions arise from the RK8(7)13M work, but not for the RK8(6)12M which has a $\boldsymbol{\beta}_i^{(8)}$ vector *cubic* in c_i. The only difference that can be seen (superficially) is the lack of a $p = q - 1$ embedding or use of $R_{0j} = 0,\ j = 2, \ldots, s$ (lower-order process) simplifying conditions in this case, which clearly affect linear properties of the $M^{(q)}$ (and $A_i^{(p)}$) matrices. Again, this could be better understood. The RK8(6)12M $p^* = 8$ case was, questionably, the hardest formula encountered. This was not due to the size of the equation systems or any conceptual difficulty, but to the huge degree polynomials that arose from the consistency conditions. Free 'β' parameters were allowed to become polynomial in the treatment of $\overline{D}_i$ systems, and this could have been stopped by use of the 'δ-dependency' technique from Chapter 8. Although not strictly necessary given the MACSYMA symbolic manipulator with a large enough working memory, the coding of error norms could have been much easier having made this slight simplification.

The problem of numerical rounding effects when implementing the new formulae has perhaps not been treated with enough care. For the sixth-order Nyström cases, it is not doubted whether a double-precision (16 decimal figure) implementation would be useful. For the remaining models, in most cases, a compromise was made to help improve numerical stability. One problem was the checking of polynomial weight coefficient sizes, which surprisingly could be many orders higher than necessary without increasing truncation error norms. In attempts to reduce them, however, truncation error norms became less controllable and compromises were inevitably made. The main concern is that if the formulae are inevitably to be implemented in a higher precision (quadruple, say), then no compromise would have been necessary and the methods could have been better. In short, it would have been much easier to develop these processes knowing in what circumstances they were to be implemented. There is no doubt that the $87\mathrm{D}^8_{20}\mathrm{L}$ and $86\mathrm{D}^8_{19}\mathrm{L}$ extensions (presented in Baker, et al. [66]) perform badly in double-precision arithmetic for tolerances less than 1×10^{-10}, effects that were improved slightly by the compromised $87\mathrm{D}^8_{20}\mathrm{L}_2$ and $86\mathrm{D}^8_{19}\mathrm{L}_2$ recommended here. If only double-precision calculations were available, a minimal amount of software arithmetic to precisely evaluate the b^*_i polynomials increases practicability by several 'tolerances' of accuracy. It seems reasonable that such a scheme (the `FLOAT` command of RK-AID) could be incorporated into a production code. Use of the RKN12(10)17M and its continuous extensions is a different matter, and can only be implemented effectively in quadruple-precision arithmetic, at least for the range of tolerances in which the triple is most useful (i.e., for high-accuracy work).

The use of embedded continuous extensions of varying orders has been left open-ended as of Chapter 8. For the RKN6(4)6FM, RK8(7)13M and RK8(6)12M, is was found necessary for embedded $p^* = q-1$ interpolants to be competitive with dedicated models. The RKN12(10)17M, being the most expensive formula with ≥ 29 stages for $p^* = 12$, will require large amounts of storage in a production code. It is therefore most relevant here for lower-order dense output to share function evaluations, but it was found difficult to derive good $p^* < q$ cases within the $p^* = q$ processes. This either suggests that better twelfth-order formulae are possible by ensuring optimal embedded interpolants, or that underlying interpolation theory has much less influence on such high-order cases, and good embedded formulae do not naturally exist. This could be clarified by additional analysis.

Finally, since the work of Chapter 7 (and paper [66]) was undertaken, dense output formulae for the RK8(7)13M have been found to be available (cited as a private communication with J. H. Verner in Shampine, et al. [67]). These results have yet to be published, and it is unknown to what extent they compare with those here. It is expected that the global error criterion adopted for optimisation of the formulae presented here will be more suited to applications where continuous global error is of primary importance. Given more time and the availability of his results, a comparison with Verner's formulae would be instructive, not least from the point of view of a commercial software implementation using the RK8(7)13M pair, as is the RKSUITE code of [67].

9.3 Recent developments

It has become apparent that treatment of free b^*'s is over-simplified throughout Chapters 6–8. Firstly, having satisfied continuity requirements using coefficients of the

		Degree		
		4	5	7
s^{y^*}	6	7.48×10^{-5}	–	–
	7	1.80×10^{-5}	1.58×10^{-5}	1.54×10^{-5}
	8	9.51×10^{-6}	4.84×10^{-6}	3.24×10^{-6}
	9	8.95×10^{-6}	3.82×10^{-6}	1.76×10^{-6}

Table 9.1: $N_l^{(7)}$ over varying s^{y^*} and 'b^*' polynomial degree for the $\mathrm{FD}_9^6\mathrm{L}$.

		Degree		
		4	5	7
s^{y^*}	7	3.70×10^{-5}	–	–
	8	1.24×10^{-5}	1.24×10^{-5}	1.21×10^{-5}
	9	4.24×10^{-6}	2.84×10^{-6}	1.31×10^{-6}

Table 9.2: $N_l^{(7)}$ over varying s^{y^*} and 'b^*' polynomial degree for the $\mathrm{FMD}_9^6\mathrm{L}$.

low powers of σ of each free b_i^*, all remaining coefficients appear only linearly in truncation error coefficients. While this was exploited in Chapters 7 and 8, general optimisation procedures using GLOBS were employed for the sixth-order y^* processes of the RKN6(4)6FD and RKN6(4)6FM. By exact minimisation of the integral of the *squared* two-norm of principal truncation error (linear deviation) quantities, better y^* formulae are possible which show improved performance in numerical tests (in an

SBD sense). It is now clear that specification of a free b_i^* allows an unlimited number of degrees of freedom to be introduced by choice of the arbitrary polynomial degree. This has been investigated with regard to the y^* processes of the RKN6(4)6FD and RKN6(4)6FM. In Chapter 6, phrases such as 'assuming fourth-degree polynomials' should be considered as a simplification. Exact minimisations have been done for the seventh-order linear deviation quantities using fifth- to seventh-degree b^*'s utilising varying numbers of the possible extra stages available from the y'^* formula. The sixth-degree optimisations always produced the fifth-degree formulae, but otherwise, increasing the interpolant degree was found to benefit linear deviation quantities in almost all cases (see Tables 9.1 and 9.2). While the fifth- and seventh-degree b^*'s could be seen as beneficial at stringent tolerances ($< 1{\times}10^{-6}$), differences were marginal. The new fourth- and fifth-degree y^* formulae for both RKN6(4)6FD and RKN6(4)6FM cases are given in Figures 9.3–9.6. With one of these modifications, the $\mathrm{FD}_9^6\mathrm{L}$ and $\mathrm{FMD}_9^6\mathrm{L}$ (along with RKN12(10)17M work) are to be included in a paper on RKN triples similar to Baker, et al. [66]. It is expected that similar generalisations could be considered for the seventh-order dense formulae of the RK8(7)13M (in which b_{13}^* is free), tenth-order dense formulae for the RKN12(10)17M (b_{17}^* and $b_{17}'^*$ free) and the twelfth-order y^* cases ($b_{s^*-2}^*$ to $b_{s^*}^*$ free).

9.4 Ideas for future work

It would be of great interest to use RK-AID as the starting point for development of new high-order continuous RK(N) formulae. The techniques of Chapters 5–8 could equally well be applied to 'stand-alone' dense processes which could then be constrained to contain good discrete processes. This is a very different methodology than has been used in this work, but one which is begged from success in forming continuous extensions.

Development of increasingly complex methods suggests improvement to the reduction routines of RK-AID. Specifically, many techniques for development of RK(N) models exist that can not be handled by the RK-AID routines. These include degener-

$$T = \frac{1950437}{3938220}\sigma^4 - \frac{140753}{100980}\sigma^3 + \frac{20731}{19448}\sigma^2 + \frac{14965}{393822}\sigma$$
$$S = \frac{1041202}{8448466455}\sigma^4 + \frac{53148899}{11264621940}\sigma^3 - \frac{2068921}{166883288}\sigma^2 + \frac{57119995}{6758773164}\sigma$$

b_i^*
$\frac{124897}{277992}\sigma^4-\frac{149515}{92664}\sigma^3+\frac{68663}{30888}\sigma^2-\frac{411601}{277992}\sigma+\frac{1}{2}$
0
$T + S\sqrt{8581}$
$T - S\sqrt{8581}$
$\frac{86000}{590733}\sigma^4-\frac{138800}{196911}\sigma^3+\frac{23000}{21879}\sigma^2-\frac{284000}{590733}\sigma$
$\frac{721}{1320}\sigma^4-\frac{567}{440}\sigma^3+\frac{87}{88}\sigma^2-\frac{65}{264}\sigma$
$-\frac{371293}{135432}\sigma^4+\frac{371293}{45144}\sigma^3-\frac{371293}{45144}\sigma^2+\frac{371293}{135432}\sigma$
$\frac{945}{484}\sigma^4-\frac{2835}{484}\sigma^3+\frac{2835}{484}\sigma^2-\frac{945}{484}\sigma$
$-\frac{2000033}{1489752}\sigma^4+\frac{2000033}{496584}\sigma^3-\frac{2000033}{496584}\sigma^2+\frac{2000033}{1489752}\sigma$

Table 9.3: A new degree-four y^* formula for the $\mathrm{FD}_9^6\mathrm{L}$.

$$T = \frac{24496297}{3938220}\sigma^5 - \frac{413761}{19448}\sigma^4 + \frac{69829313}{2625480}\sigma^3 + \frac{11408543}{787644}\sigma^2 - \frac{8265199}{2625480}\sigma$$
$$S = \frac{2391606923}{33793865820}\sigma^5 - \frac{41315849}{166883288}\sigma^4 + \frac{7281118567}{22529243880}\sigma^3 - \frac{639797381}{3379386582}\sigma^2 + \frac{75969407}{1733018760}\sigma$$

b_i^*
$\frac{118763}{694980}\sigma^5-\frac{221}{1485}\sigma^4-\frac{195643}{231660}\sigma^3+\frac{124801}{69498}\sigma^2-\frac{323207}{231660}\sigma+\frac{1}{2}$
0
$T + S\sqrt{8581}$
$T - S\sqrt{8581}$
$-\frac{1302800}{590733}\sigma^5+\frac{516200}{65637}\sigma^4-\frac{161000}{15147}\sigma^3+\frac{3878000}{590733}\sigma^2-\frac{311800}{196911}\sigma$
$\frac{79}{264}\sigma^5-\frac{441}{880}\sigma^4+\frac{51}{880}\sigma^3+\frac{127}{528}\sigma^2-\frac{17}{176}\sigma$
$-\frac{2599051}{270864}\sigma^5+\frac{1856465}{60192}\sigma^4-\frac{6311981}{180576}\sigma^3+\frac{8539739}{541728}\sigma^2-\frac{371293}{180576}\sigma$
$-\frac{32751}{9680}\sigma^5+\frac{267057}{19360}\sigma^4-\frac{408159}{19360}\sigma^3+\frac{55431}{3872}\sigma^2-\frac{70551}{19360}\sigma$
$\frac{8468327}{3724380}\sigma^5-\frac{3848803}{413820}\sigma^4+\frac{17702573}{1241460}\sigma^3-\frac{7234213}{744876}\sigma^2+\frac{1539041}{620730}\sigma$

Table 9.4: A new degree-five y^* formula for the $\mathrm{FD}_9^6\mathrm{L}$.

b_i^*
$\frac{932627}{1902096}\sigma^4-\frac{159365}{90576}\sigma^3+\frac{89771}{37296}\sigma^2-\frac{2981243}{1902096}\sigma+\frac{1}{2}$
$\frac{3175}{19314}\sigma^4-\frac{2435}{6438}\sigma^3+\frac{7025}{38628}\sigma^2+\frac{25}{333}\sigma$
$\frac{34925}{25308}\sigma^4-\frac{10285}{2812}\sigma^3+\frac{138325}{50616}\sigma^2-\frac{1100}{6327}\sigma$
$\frac{34925}{9324}\sigma^4-\frac{38995}{3108}\sigma^3+\frac{260425}{18648}\sigma^2-\frac{11275}{2331}\sigma$
$-\frac{9921875}{4158948}\sigma^4+\frac{2740625}{346579}\sigma^3-\frac{531250}{61161}\sigma^2+\frac{140625}{47804}\sigma$
$\frac{5501}{9768}\sigma^4-\frac{4687}{3256}\sigma^3+\frac{12433}{9768}\sigma^2-\frac{1291}{3256}\sigma$
$\frac{143145755}{271735992}\sigma^4-\frac{143145755}{90578664}\sigma^3+\frac{143145755}{90578664}\sigma^2-\frac{143145755}{271735992}\sigma$
$-\frac{773822065}{382526424}\sigma^4+\frac{773822065}{127508808}\sigma^3-\frac{773822065}{127508808}\sigma^2+\frac{773822065}{382526424}\sigma$
$-\frac{115856201}{47062224}\sigma^4+\frac{115856201}{15687408}\sigma^3-\frac{115856201}{15687408}\sigma^2+\frac{115856201}{47062224}\sigma$

Table 9.5: A new degree-four y^* formula for the FMD$_9^6$L.

b_i^*
$-\frac{2759833}{19020960}\sigma^5+\frac{37971371}{38041920}\sigma^4-\frac{1799447}{745920}\sigma^3+\frac{21073117}{7608384}\sigma^2-\frac{8912099}{5434560}\sigma+\frac{1}{2}$
$\frac{3698815}{1757574}\sigma^5-\frac{3616265}{502164}\sigma^4+\frac{10653275}{1171716}\sigma^3-\frac{1487900}{292929}\sigma^2+\frac{1320905}{1171716}\sigma$
$\frac{25501795}{2303028}\sigma^5-\frac{8197915}{219336}\sigma^4+\frac{23629925}{511784}\sigma^3-\frac{14365175}{575757}\sigma^2+\frac{2744555}{511784}\sigma$
$-\frac{4868545}{848484}\sigma^5+\frac{5776595}{242424}\sigma^4-\frac{21702725}{565656}\sigma^3+\frac{2001725}{70707}\sigma^2-\frac{4358915}{565656}\sigma$
$\frac{79309375}{31538689}\sigma^5-\frac{604840625}{54066324}\sigma^4+\frac{71328125}{3710434}\sigma^3-\frac{2833328125}{189232134}\sigma^2+\frac{529728125}{126154756}\sigma$
$-\frac{37633777}{73333260}\sigma^5+\frac{24716711}{10476180}\sigma^4-\frac{15273053}{4074070}\sigma^3+\frac{18742519}{7333326}\sigma^2-\frac{11973359}{18333315}\sigma$
$-\frac{5043797935727}{680019319980}\sigma^5+\frac{643268393819}{24286404285}\sigma^4-\frac{3961959577739}{113336553330}\sigma^3+\frac{684208079749}{34000965999}\sigma^2-\frac{2880121219751}{680019319980}\sigma$
$\frac{500583364063}{174049522920}\sigma^5-\frac{85882890359}{7104062160}\sigma^4+\frac{2205928171339}{116033015280}\sigma^3-\frac{923090211553}{69619809168}\sigma^2+\frac{1204761443213}{348099045840}\sigma$
$-\frac{149338643089}{31406190816}\sigma^5+\frac{381746182295}{26919592128}\sigma^4-\frac{880159558997}{62812381632}\sigma^3+\frac{282804986641}{62812381632}\sigma^2+\frac{15872299537}{188437144896}\sigma$

Table 9.6: A new degree-five y^* formula for the FMD$_9^6$L.

ate cases where nodal values are used for simplification purposes. It would therefore be ideal for RK-AID to accept linear combinations of parameters to be zero. For example, combinations of the form

$$c_i - c_j = 0,\; b_i + b_j = 0,\; R_{ki} + R_{kj} = 0 \text{ and } Q_{ik} - Q_{jk} = 0,$$

for some i and j, can be used effectively for elimination of RK order conditions (for example, see Hairer [28]). This would not be difficult to achieve as an addition to the 'reduction mode', and would vastly increase the scope of the software for development of new (high-order) models.

Appendix A

RK-AID reference manual (v7.6)

A.1 Introduction

RK-AID is a general purpose Fortran 90 package written by the author to aid development and analysis of RK and special RKN processes, the technical details of which are described in Chapter 3. The main features may be summarised as follows:

1. Automation of common stages in the development of RK(N) processes including generation of order conditions and truncation error coefficients under a wide range of simplifying schemes.

2. Ability to deal with both RK and RKN methods in an entirely consistent manner.

3. Detailed analysis of explicit RK(N) triples including real negative stability limits, (continuous) truncation error norms and continuity conditions, implicit and semi-implicit *single* processes being considered as special cases.

4. Concurrent output into LaTeX[1] to accompany standard output, allowing convenient relay of more detailed information than is possible neatly in ASCII, and also minimising the chance of propagating typing errors when presenting the output (this thesis is written in LaTeX and so many results contained within will be direct from RK-AID).

[1]LaTeX(Lamport [68]) is one of several popular macro-extensions to TeX, the mathematical typesetting language of Knuth [69].

5. Output of MACSYMA[2] batch-files, thus making the transition between early development and the main body of algebraic work easier and smoother. Much use will be made of MACSYMA throughout this work.

6. Incorporated testing routines containing a range of problems for comparing finished processes (again, both RK and RKN methods are handled).

7. Implementation of sophisticated software based arithmetic to replace or augment floating-point hardware calculations.

A.1.1 Installation and execution

The RK-AID source code is split over six files requiring separate compilation and subsequent 'linking' into a single executable. The files are as follows, and should also be compiled in this order:

rats.f90, *data.f90*, *tn.f90*, *rkm.f90*, *subs.f90* and *main.f90.*

This source code is available via anonymous ftp at *ftp.tees.ac.uk* in the directory

/pub/t.baker/f90/rkaid

as is a LaTeX 'style file' necessary for the compilation of all ".tex" files produced by RK-AID. On first calling the executable, the following prompt will be seen:

```
 >RK-AID{5} v7.6 [5/97] ("?" for command line help, "><" to exit)


rkaid:
```

The number in braces (`{5}`) is the compiled multi-integer size — see Section A.5. This is followed by the version number (`v7.6`) and date of last modification (`[5/97]`).

[2]MACSYMA is a commercial symbolic manipulator similar to other widely used algebra packages such as GNU Calc, DERIVE, MAPLE and MATHEMATICA.

```
-------------------------------------------------------
   "{text}" to add commands, delimited by pipes (|),
   ".", ".." & "..." for listing contents of buffer
   "\", "\\" & "\\\" for deleting contents of buffer
     ["blank", "blank/commented" and "all" lines],
   "!{#}" to comment or un-comment line {#},
   "{#}. {text}" to replace line {#} with {text},
   "\{text}\" to replace the closest matching line,
   "{#}. \{text1}\{text2}\" to substitute string
          {text1} for {text2} in line {#},
   "'{exp1} & {exp2} & ... " to evaluate the {exp#},

  "< {file}" to read, and "> {file}" to write a file
          ( {file} defaults to "rkaid.dat" ).
-------------------------------------------------------
   "<<" to re-start (clear) RK-AID or "><" to exit,
         "?" to re-display THIS message,

           "batch" to enter batch mode,
    "echo" to echo the command line in batch mode,

   <RETURN> to start each run ("go" in batch-files).
-------------------------------------------------------
```

Figure A.1: The RK-AID front-end command summary.

A.2 The front-end text buffer

The RK-AID prompt (`rkaid:`) allows entry into a text buffer for the input of commands and data. This buffer then acts as a top level data-file which may be edited, run or saved to disk using commands at the front-end level. A set of these commands can be seen by entering the symbol "?" at the prompt (followed by a carriage return). This summary page is shown in Figure A.1. Commands and data entered into this buffer have no effect until the data-file is run, and its state is maintained between runs. As an example, consider the case of generating a set of truncation error coefficients for an RK process. The relevant command in its simplest form is `ORDER n:m`, where `n` and `m` are the range of orders required for analysis. Thus to enter the command for generation of terms up to fourth-order,

```
order :4
```

should be entered at the prompt. To run the command, give a blank line. Following execution, the standard output should appear as in Figure A.2. To exit the program at this point enter "><" at the prompt. However, as the command `order :4` is still in the buffer, if the same terms are required employing Butcher 'Q' parameters (see Section 3.2.3), give the additional command `use Q's`. To see the new contents of the buffer, give the front-end list command "." followed by a carriage return. This will output:

```
1. order :4
2. use Q's
```

where lines in the buffer are numbered for use of simple editing features, removing the need for a text editor when running simple tasks (see Figure A.1). Giving a blank line will now run the new data-file and yield the standard output in Figure A.3. As well as generation of truncation error coefficients in this way, complete RK and RKN processes may be entered via their method coefficients and then checked for order properties or numerically tested using integration routines. The general form of command entry will now be described, and is followed by methods for entering process coefficients, in Section A.3.2.

```
 >RK-AID{5} v7.6 [5/97] ("?" for command line help, "><" to exit)

rkaid: order :4
rkaid:

 [running]


      1.  Tau(1,1) = b(i) - 1
      2.  Tau(2,1) = b(i)c(i) - 1/2!
      3.  Tau(3,1) = (1/2)*b(i)c(i)^2 - 1/3!
      4.  Tau(3,2) = b(i)a(i,j)c(j) - 1/3!
      5.  Tau(4,1) = (1/6)*b(i)c(i)^3 - 1/4!
      6.  Tau(4,2) = b(i)c(i)a(i,j)c(j) - 3/4!
      7.  Tau(4,3) = (1/2)*b(i)a(i,j)c(j)^2 - 1/4!
      8.  Tau(4,4) = b(i)a(i,j)a(j,k)c(k) - 1/4!


rkaid:
```

Figure A.2: RK-AID standard output following an `order :4` command.

```
 [running]


      1.  Tau(1,1) = b(i) - 1
      2.  Tau(2,1) = b(i)c(i) - 1/2!
      3.  Tau(3,1) = (1/2)*b(i)c(i)^2 - 1/3!
      4.  Tau(3,2) = b(i)Q(i,1) + Tau(3,1)
      5.  Tau(4,1) = (1/6)*b(i)c(i)^3 - 1/4!
      6.  Tau(4,2) = b(i)c(i)Q(i,1) + 3*Tau(4,1)
      7.  Tau(4,3) = (1/2)*b(i)Q(i,2) + Tau(4,1)
      8.  Tau(4,4) = b(i)a(i,j)Q(j,1) + Tau(4,3)


rkaid:
```

Figure A.3: Standard output from `order :4` and `use Q's` commands.

A.3 Command and data entry

A.3.1 Function keywords

Owing to the relatively small number of commands, one or more keywords for implementing a certain feature are simply entered at any point within the data-file structure (a 'structure' in the sense that each data-file may be linked onto the end of another by incorporating an '`INCLUDE` *file*', or '`READ` *file*' on any line by itself). Most keywords may be abbreviated to their syntactic root (i.e. `PARAM`, `PARAMETER`, `PARAMETERISE` and `PARAMETRIC` would be read the same) and are case-independent, allowing for data-files to be written sparsely or freely embellished with comments. Comment lines may also be explicitly included following a '!' character, causing the remainder of the line to be ignored by the parser. A complete list of commands and functions is given in Section A.6.

A.3.2 Coefficient entry syntax

Coefficient information may be entered in a variety of ways. The primary one is that of $variable = value$, where subscripts are indicated as in Fortran or MACSYMA (for example) by use of round or square brackets, i.e.,

$$\mathtt{a}[2,1] = 1/20, \quad \underline{\mathtt{or}} \quad \mathtt{a}(2,1) = 3/20 \tag{A.1}$$

Multiple assignments of this form may be included on the same line by separating them with either the "`&`" character or the word "`and`", for example,

$$\mathtt{b}(2) = 0 \ \mathtt{and} \ \mathtt{b}(3) = 0$$

This can also be achieved more simply by use of a (Fortran 90/MATLAB style) vector notation, as in $\mathtt{b}(2:3) = 0$. Lists of subscript values may also be entered within brackets if separated by semicolons (`;`), for example,

$$\mathtt{R}(1,1;3:\mathtt{s}-2;\mathtt{s})=0$$

The parser implemented allows for symbolic entry including use of simple variables of one to twenty *letters* (including the under-score character '`_`'), e.g.,

$$one\_over\_twenty = 1/20$$

$$a[2,1] = one\_over\_twenty \ \underline{\text{or}}\ a(2,1) = 3 * one\_over\_twenty$$

is equivalent to (A.1). Without use of parenthesis, which may be recursively nested, the expressions may not be entered in a *strictly* symbolic manner due to the lack of any robust operator priority scheme. Raising to a power and implicit multiplication always have highest priority, but beyond that, a simple left-to-right scheme operates. The parser accepts the symbols $+$, $-$, $/$, $*$, and ˆ, with the square-root function being implemented as either `root`$(\cdot)$, `sqr`$(\cdot)$ or `sqrt`$(\cdot)$. Raising to a power (ˆ) is only allowed for positive integer exponents. Variables must be restricted to names not used for system functions, i.e., `root`, `sqr`, `sqrt`, and the floating-point exponent operators `e`, `d`, and `b`. It should be noted that all *other* variable names are case-dependent for more freedom of choice. If all coefficients for a particular method are entered in this manner, relevant process information is determined automatically, i.e., s, s^*, and the maximum polynomial degree of the continuous weights are calculated internally, but RK or RKN type must still be specified explicitly. Parameters given in this way are assumed to be exact, if not, the keyword `APPROXimate` is best included. Data can also be entered in floating-point decimal and exponential forms, and may be freely mixed with the fractional representation above. The coefficients, if entered exactly, may be used in either exact or floating-point forms — such arithmetic features are described in Section A.5.

Different processes embedded within the method (main formula, step size control formula and dense process) are identified by way of the embellishment that appears on the plain symbol. Primarily, this is done (within commands and in ASCII output) by enclosing an identifier within square brackets following the process dependent parameters (b's, R's and τ's), or in isolation when specifying the required process in commands. These are summarised in column 2 of Table A.1.

Coefficient set	Identifiers	Variable form
b	no symbol, `[y]` or `[p]`	`b`
$\hat{b}$	`[h]` or `[q]`	`bh`
b^*	`[s]`	`bs`
b'	`[y']`, `[p']`, `[p]'`, `[']` or `'`	`bd`
$\hat{b}'$	`[h']`, `[h]'`, `[q']` or `[q]'`	`bdh`
b'^*	`[s']` or `[s]'`	`bds`
all	`[all]`	—

Table A.1: RK-AID process identifiers.

For example, $\mathtt{R[s'](0,1)=0}$, $\mathtt{R[s]'[0,1]=0}$ and $\mathtt{RDS(0,1)=0}$ are equivalent. In the last form here, the "d" on the variable identifies the derivative symbol, "s" for star, etc. This enables input and output of variable names which are easily ported to, and from, languages such as Fortran and MACSYMA (see the `PORT` command in Section A.6.8). The last row of Table A.1 is particularly useful for imposing conditions on all processes at once — as in, for example, $\hat{b}_2 = b_2 = b_2^* = 0$ (for an RK triple), where the single command $\mathtt{b[all](2)=0}$ will achieve the same result.

Polynomial entry

The RK(N) b_i^* (and $b_i'^*$) polynomials in σ are entered in one of two ways. The first is not recommended due to its lack of generality, and is mentioned mainly because it *can* be done but also to provide insight into the way RK-AID handles such polynomials. During numerical evaluation of τ^*'s (and τ'^*'s), the displayed value is that calculated at one value of σ only. This value is determined by the variable `SIGMA`[3] which is

[3]`SIGMA` has a default value of 0.47 if not otherwise specified.

case-independent on entry, as mentioned previously. It is *sufficient* only that the polynomials are specified at that point. Therefore, the following value could be given to b_1^*:

```
b[s](1)=10*SIGMA^3+9*SIGMA^2-8*SIGMA^1+1
```

which would be evaluated at the value of σ held as the line was parsed. More generally, the polynomial for any continuous weight may be given by the addition of a second coordinate onto the parameter name, e.g.,

$$\begin{aligned} \texttt{b[s](1,3)} &= 10 \\ \texttt{b[s](1,2)} &= 9 \\ \texttt{b[s](1,1)} &= -8 \\ \texttt{b[s](1,0)} &= 1 \end{aligned}$$

is equivalent to the above. Prior to calculation, all polynomials given in this form are evaluated at $\sigma = \texttt{SIGMA}$ for use in the truncation error coefficient calculation process. In this case, of course, the general polynomials are available for calculation at any given σ value. This is the form necessary for all other dense analysis facilities.

A.3.3 Assumed data entry

Other techniques for entering method coefficients depend on the data being present in a pre-determined, or *assumed*, form such that no identification is required for each value. The explicit assumed form comes in two subtly different variations — the general schemes, i.e., those for an RKN triple, are shown in Figure A.4. There are two ways of arranging these items of data in the file, as follows:

1. *Common-denominator* form. Here, data is written in floating-point or decimal form, preceding each vector of quantities by their common-denominator. Note that the b_i^*'s and $b_i'^*$'s are polynomials in σ and are thus entered via the coefficients of each power of σ, from σ^0 through to σ^d, where d is the *maximum* degree of all continuous weights, and each common-denominator relates to a

single polynomial. The file in this case should contain the keyword `ASSUMEd` (plus `BLOCK` if necessary). Note that these data files do not necessarily carry approximate information since a floating-point coefficient given in common-denominator form can be exact, and addition of the keyword `APPROXimate` should also be done when necessary.

$$
\begin{array}{l}
a_{2,1} \\
a_{3,i},\ i = 1, 2 \\
\vdots \\
a_{s,i},\ i = 1, \ldots, s-1 \\
\hline
\hat{b}_i,\ i = 1, \ldots, s \\
\hat{b}'_i,\ i = 1, \ldots, s \\
b_i\ i = 1, \ldots, s \\
b'_i,\ i = 1, \ldots, s \\
\hline
a_{s+1,i},\ i = 1, \ldots, s \\
\vdots \\
a_{s^*,i},\ i = 1, \ldots, s^*-1 \\
\hline
b^*_i,\ i = 1, \ldots, s^* \\
b'^*_i,\ i = 1, \ldots, s^*
\end{array}
\qquad\qquad
\begin{array}{l}
a_{2,1} \\
a_{3,i},\ i = 1, 2 \\
\vdots \\
a_{s,i},\ i = 1, \ldots, s-1 \\
a_{s+1,i},\ i = 1, \ldots, s \\
\vdots \\
a_{s^*,i},\ i = 1, \ldots, s^*-1 \\
\hline
\hat{b}_i,\ i = 1, \ldots, s^* \\
\hat{b}'_i,\ i = 1, \ldots, s^* \\
b_i,\ i = 1, \ldots, s^* \\
b'_i,\ i = 1, \ldots, s^* \\
\hline
b^*_i,\ i = 1, \ldots, s^* \\
b'^*_i,\ i = 1, \ldots, s^*
\end{array}
$$

Figure A.4: Assumed and assumed *block* form coefficient tables, respectively.

2. *Long* form. Here, rational data is represented, not coincidentally, in a form similar to MACSYMA standard output, whereby the numerator and denominator are presented, one per 3 or more lines, separated by a row of minus signs, i.e.,

```
                3
    a[2, 1] = --
                20
```

or

```
              7651276
    b[2] = - -------
              9898713
```

The horizontal position of the fraction is unimportant, and the coefficient name is optional since the form is assumed. In this case, the file should contain the keywords `ASSUMEd` and `LONG` (plus `BLOCK` if necessary).

As implicit methods are considered in isolation only, i.e., as stand-alone discrete processes, the form of assumed input in this case is simply an $s \times s$ block of a_{ij} coefficients, followed by the vector of b_i's (and then b'_i's for RKN). In all cases, additional keywords should include the class of method, i.e. `RKN` or `NYSTROM` for a Nyström process (RK is default). Before the coefficient data must also be included a line containing the following data separated by spaces or commas:

- s, the number of stages in the discrete process (single formula or pair),

and for dense methods,

- $s^* - s$, the number of extra stages used by the continuous extension, and
- the maximum degree of the b_i^* and $b_i'^*$ polynomials.

It is also good practice to precede the coefficient information by a line including the word `CUT`, as for speed the parser will then not look any further in the file for commands.

Example A.3.1 (The Merson RK$_5^{4(3)}$ pair [30]) This is a typical data-file for an RK pair, the header line containing the required keywords `EXPLICIT` and `RK` (though these are both default values), and `PAIR` denoting the presence of an embedded process. The name of the method is detected as M435, being preceded by the characters "RK", the expanded title "Merson 4(3)5" will be ignored and is present only as information.

1. **Free form.** Note that the c_i coefficients have been entered explicitly, though optional, as $c_i,\ i = 1, \ldots, s$, are assumed to satisfy the RK row-sum conditions (3.4) and so can be calculated rationally if necessary. Also, use of a vector notation is allowed, where the lower limit may be omitted, as may the upper

limit, assuming the number of stages is established *somewhere*. Entries are delimited with a semicolon (;).

```
RKM435 - Merson 4(3)5 explicit RK pair

c(1:5) = 0 ; 1/3 ; 1/3 ; 1/2 ; 1

a(2, 1) = 1/3
a(3, :2) = 1/6 ; 1/6
a(4, :3) = 1/8 ; 0 ; 3/8
a(5, :4) = 1/2 ; 0 ; -3/2 ; 2

b(:) = 1/10 ; 0 ; 3/10 ; 2/5 ; 1/5
b[h](:) = 1/6 ; 0 ; 0 ; 2/3 ; 1/6
```

2. **Assumed form.** Here, note that the keyword `APPROXimate` is not required in the header as the common-denominator form contains exact information.

```
RKM435 explicit RK pair (assumed form coefficients)

------------------------cut------------------------

5 !no. of stages

3
  1 !a(2,1) * 3
6
  1 1 !a(3, 1:2) * 6
8
  1 0 3 !a(4,1:3) * 8
2
  1 0 -3 4 !a(5,1:4) * 2
6
  1 0 0 4 1 !b[h](1:5) * 6
10
  1 0 3 4 2 !b(1:5) * 10
```

□

Example A.3.2 (The RK5(4)7FM with fourth-order dense [14]) Here, note the ability to mix numerical types (in the ***c*** vector), and the use of "`&`" as a continuation character.

```
RK547FMD4 explicit RK triple

a(2, :) = 1/5
a(3, :) = 3/40 ; 9/40
a(4, :) = 44/45 ; -56/15 ; 32/9
a(5, :) = 19372/6561 ; -25360/2187 ; 64448/6561 ; -212/729
a(6, :) = 9017/3168 ; -355/33 ; 46732/5247 ; 49/176 ; &
  & -5103/18656
a(7, :) = 35/384 ; 0 ; 500/1113 ; 125/192 ; -2187/6784 ; 11/84

c(:) = 0 ; 0.2 ; 0.3 ; 0.8 ; 8/9 ; 1 ; 1

b[h](:7) = 35/384 ; 0 ; 500/1113 ; 125/192 ; -2187/6784 ; 11/84 ; 0

b(1:7)   = 5179/57600 ; 0 ; 7571/16695 ; 393/640 ; &
  & -92097/339200 ; 187/2100 ; 1/40

b[s][1, 0:4] = 1 ; -4034104133/1410260304 ; &
  & 105330401/33982176 ; -13107642775/11282082432 ; &
  & 6542295/470086768
b[s][2] = 0
b[s][3, 0:4] =  0 ; 132343189600/32700410799 ; &
  & -833316000/131326951 ;  91412856700/32700410799 ; &
  & -523383600/10900136933
b[s][4, 0:4] =  0 ; -115792950/29380423 ; &
  & 185270875/16991088 ; -12653452475/1880347072 ; &
  & 98134425/235043384
b[s][5, 0:4] = 0 ; 70805911779/24914598704 ; &
  & -4531260609/600351776 ; 988140236175/199316789632 ; &
  & -14307999165/24914598704
b[s][6, 0:4] = 0 ; -331320693/205662961 ; 31361737/7433601 ; &
  & -2426908385/822651844 ; 97305120/205662961
b[s][7, 0:4] = 0 ; 44764047/29380423 ; -1532549/353981 ; &
  & 90730570/29380423 ; -8293050/29380423
```

□

Example A.3.3 (The RKN6(4)6FD with fifth-order dense [15]) Here, the c_i coefficients have not been given and so will be calculated rationally from the RKN row-sum conditions (3.5). Note the use of a variable to store the surd part ($\sqrt{8581}$), and common-denominators where appropriate.

```
RKN646FD5 explicit RKN triple.

T = root(8581)

a(2, :)   = (26131 - 209T)/810000
a(3, :)   = (26131 - 209T)/607500; (26131 - 209T)/303750
a(4, 1)   = (980403512254 + 7781688431T)/11694469921875
a(4, 2)   = -(1262884486208 + 15385481287T)/11694469921875
a(4, 3)   = (7166233891441 + 78694563299T)/46777879687500
a(5, 1:2) = -9(329260 + 3181T)/27040000; 27(35129 + 3331T)/13520000
a(5, 3)   = -27(554358343 + 31040327T)/464060480000
a(5, 4)   = 153(8555257 - 67973T)/2745920000
a(6, 1:3) = 329/4212; 0; (84119543 + 366727T)/409622616
a(6, 4:)  = (84119543 - 366727T)/409622616; 200/17901

b[h](1:3)  = 329/4212; 0; (84119543 + 366727T)/409622616
b[h](4:6)  = (84119543 - 366727T)/409622616; 200/17901; 0
b[h]'(1:3) = 329/4212; 0; (389225579 + 96856T)/1024056540
b[h]'(4:6) = (389225579 - 96856T)/1024056540 ; 2000/17901; 1/20

b(1:3) = (2701 + 23T)/4563; -(9829 + 131T)/9126 ; 5(1798 + 17T)/9126
b(4:6) = 0

b'(1:3) = 115/2106; 0; (84119543 + 366727T)/256014135
b'(4:6) = (84119543 - 366727T)/256014135; 6950/17901; -1/10

den = 4212
b[s][1, 0:4] = 1/2 ; -5244/den ; 6386/den ; -3819/den ; 900/den

b[s][2] = 0

den = 22529243880
b[s][3, 0:4] = 0 ; 18(81356461 + 25954829T) / den ; (22190560391 - &
  & 1109665151T) / den ; -6(4929647204 - 156109769T) / den ; &
  & 1800(5860823 - 152228T) / den
b[s][4, 0:4] = 0 ; 18(81356461 - 25954829T) / den ; (22190560391 + &
  & 1109665151T) / den ;  -6(4929647204 + 156109769T) / den ; &
  & 1800(5860823 + 152228T) / den

den = 17901
b[s][5, 0:4] = 0 ; 39000/den ; -124000/den ; 130200/den ; -45000/den

den = 220
b[s][6, 0:4] = 0 ; -234/den ; 757/den ; -823/den ; 300/den

den = 4212
b[s’][1, 0:4] = 1 ; -15732/den ; 25544/den ; -19095/den ; 5400/den

b[s’][2] = 0

den = 11264621940
b[s’][3, 0:4] = 0 ; 27(81356461 + 25954829T) / den ; &
  & 2(22190560391 - 1109665151T) / den ; -15(4929647204 - &
  & 156109769T) / den ; 5400(5860823 - 152228T) / den
b[s’][4, 0:4] = 0 ; 27(81356461 - 25954829T) / den ; &
  & 2(22190560391 + 1109665151T) / den ; -15(4929647204 + &
  & 156109769T) / den ; 5400(5860823 + 152228T) / den

den = 17901
b[s’][5, 0:4] = 0 ; 117000/den ; -496000/den ; 651000/den ; &
  & -270000/den

den = 220
b[s’][6, 0:4] = 0 ; -702/den ; 3028/den ; -4115/den ; 1800/den
```

$\square$

A.4 Output files

Along with standard output, RK-AID makes use of several other formats to convey results and data. The primary alternative medium is LaTeX output, a typesetting language based on ASCII text files which must be compiled into a device independent (DVI) file for viewing or printing. A LaTeX compiler is therefore required to make use of this facility. The text file for compilation is written to the current working directory as a ".tex" file, and contains principal results in a more mathematically precise form than could conveniently be given in ASCII to the standard output. Certain features of the software only output to the LaTeX form, in conjunction with the `LATEX` command (see Section A.6.9).

The MACSYMA language is similarly catered for through ".m" files whenever symbolic functions are emulated, such as in the reduction environment. Such files are MACSYMA 'batchable' and may conveniently form the starting point for algebraic solution of expressions determined by the truncation error term generation facility.

A general purpose ".out" file is always produced, containing calculated coefficient information that may be of use, along with warning and error messages. This file is also contained for reference in the LaTeX output (in a 'verbatim' form) following a `LATEX` command.

Other output files are specific to certain commands, the names of which will be detailed in the standard output in most cases. These include plots of continuous stability regions, global error data from the test routines, and assumed form coefficient files. Further output forms are described with the relevant commands (Section A.6).

One further output file is actually removed on successful exit from the program, and contains a mix of standard output and ".out" file information that may be of interest in the event of a crash due to syntax error or a 'control break'. This file will be named "*rkaid.tmp*" or "*rkaid_#.tmp*" where the hash (#) will be the next in the sequence of integers $1, 2 \dots$ such that the file does not already exist. This care to keep any output from a failed run is intended for reference when running many tasks from the operating system in an automatic or 'batched' manner, where partially successful output may be required. This build up of temporary files may need to be deleted from time-to-time to free disk space.

A.5 Arithmetic

The default arithmetic for general use is a floating-point form, the specific nature of which is machine dependent. RK-AID uses a Fortran 90 'double-precision' type which on most platforms gives approximately 16 digits of accuracy. Truncation error coefficients for RK(N) methods can involve large integer factors which must be determined by any algorithm attempting to generate them. Such integers (see Section 3.2) contain factorial components which grow in size with order, motivating an arithmetic system that allows analysis to *arbitrary* order.

A.5.1 The multi-integer type

Internally, a 'multi-integer' system is employed — the number of integer digits handled is therefore limited only by the memory of the system running the software, and theoretically RK and RKN τ's can be generated indefinitely. It should be noted that the *maximum* integer size of any one executable version of RK-AID is not dynamic and must be set at compile time. Thus, a working version may have a multi-integer type consisting of, say, 5 standard integers (of 8 digits each), which is enough for most applications.

Each multi-integer is equipped with a (single integer) exponent, and as such a floating-point arithmetic of arbitrary precision is possible for performing order checks. A multi-integer type consisting of 5 standard integers may therefore allow calculation to 40 significant digits — see the `FLOAT` command in Section A.6.15. Such software based arithmetic is *much* slower than that of a mathematics co-processor found on most machine platforms and should therefore only be used where necessary. Hereafter, 'floating-point' should be taken to mean either hardware (double-precision) or software (multi-precision) arithmetic. Some tasks are, however, only performed using machine arithmetic, amongst these are stability limit checks and method testing, though the latter is aided by availability of a high accuracy polynomial representation for the dense output.

A scheme has also been implemented on top of this multi-integer (multi-precision) system which allows for storage and manipulation of rational/surd types of the form

$$\frac{a + b\sqrt{S}}{c}. \tag{A.2}$$

It is therefore possible to check RK(N) order conditions and error terms exactly when the method coefficients are all of type (A.2) *for a single S only.* Further to this, it can be useful to work with coefficients exactly or with a high precision when passing them into another environment such as MACSYMA which, whenever relevant, is done automatically via the various output files in conjunction with the `PORT` command.

Example A.5.1 (An exact order check for the RKN6(4)6FD$_6^5$) The following commands yield truncation error coefficients evaluated and displayed alongside their symbolic representation for the RKN6(4)6FD$_6^5$:

```
check 646fd5.rat
[p'] order :6
exact arithmetic, exact output
```

Assuming that the data-file of Example A.3.3 is available as *"646fd5.rat"*, RK-AID will produce the following[4]:

$$\begin{array}{cll l}
 & 1'. & \tau_1'^{(1)} = \sum_i b_i' - 1 = 0 & \\ \hline
 & 2'. & \tau_1'^{(2)} = \sum_i b_i' c_i - \frac{1}{2!} = 0 & \\ \hline
 & 3'. & \tau_1'^{(3)} = \frac{1}{2} \sum_i b_i' c_i^2 - \frac{1}{3!} = 0 & \\
* & 4'. & \tau_2'^{(3)} = \tau_1'^{(3)} = 0 & (3\ /\ 4) \\ \hline
 & 5'. & \tau_1'^{(4)} = \frac{1}{6} \sum_i b_i' c_i^3 - \frac{1}{4!} = 0 & \\
* & 6'. & \tau_2'^{(4)} = 3\tau_1'^{(4)} = 0 & \\
 & 7'. & \tau_3'^{(4)} = \sum_{ij} b_i' a_{ij} c_j - \frac{1}{4!} = 0 & (5\ /\ 7) \\ \hline
 & 8'. & \tau_1'^{(5)} = \frac{1}{24} \sum_i b_i' c_i^4 - \frac{1}{5!} = -\frac{11}{72000} & \\
* & 9'. & \tau_2'^{(5)} = 6\tau_1'^{(5)} = -\frac{11}{12000} &
\end{array}$$

[4]More precisely, along with standard output, a stand-alone ".tex" file is produced which contains the LaTeX commands for the table as shown. These commands had to be lifted out for inclusion here.

$*\quad 10'.\quad \tau_3^{\prime(5)} = 3\tau_1^{\prime(5)} = -\dfrac{11}{24000}$

$\quad 11'.\quad \tau_4^{\prime(5)} = \sum_{ij} b'_i c_i a_{ij} c_j - \dfrac{4}{5!} = -\dfrac{11}{18000}$

$\quad 12'.\quad \tau_5^{\prime(5)} = \frac{1}{2}\sum_{ij} b'_i a_{ij} c_j^2 - \dfrac{1}{5!} = -\dfrac{11}{72000}$

$*\quad 13'.\quad \tau_6^{\prime(5)} = \tau_5^{\prime(5)} = -\dfrac{11}{72000}$

(8 / 13)

$\quad 14'.\quad \tau_1^{\prime(6)} = \frac{1}{120}\sum_{i} b'_i c_i^5 - \dfrac{1}{6!} = -\dfrac{14003}{162000000}$

$*\quad 15'.\quad \tau_2^{\prime(6)} = 10\tau_1^{\prime(6)} = -\dfrac{14003}{16200000}$

$*\quad 16'.\quad \tau_3^{\prime(6)} = 15\tau_1^{\prime(6)} = -\dfrac{14003}{10800000}$

$\quad 17'.\quad \tau_4^{\prime(6)} = \frac{1}{2}\sum_{ij} b'_i c_i^2 a_{ij} c_j - \dfrac{10}{6!} = -\dfrac{14003}{16200000}$

$*\quad 18'.\quad \tau_5^{\prime(6)} = \tau_4^{\prime(6)} = -\dfrac{14003}{16200000}$

$\quad 19'.\quad \tau_6^{\prime(6)} = \frac{1}{2}\sum_{ij} b'_i c_i a_{ij} c_j^2 - \dfrac{5}{6!} = -\dfrac{14003}{32400000}$

$\quad 20'.\quad \tau_7^{\prime(6)} = \frac{1}{6}\sum_{ij} b'_i a_{ij} c_j^3 - \dfrac{1}{6!} = \dfrac{842239 + 2029\sqrt{8581}}{3888000000}$

$*\quad 21'.\quad \tau_8^{\prime(6)} = \tau_6^{\prime(6)} = -\dfrac{14003}{32400000}$

$*\quad 22'.\quad \tau_9^{\prime(6)} = 3\tau_7^{\prime(6)} = \dfrac{842239 + 2029\sqrt{8581}}{1296000000}$

$\quad 23'.\quad \tau_{10}^{\prime(6)} = \sum_{ijk} b'_i a_{ij} a_{jk} c_k - \dfrac{1}{6!} = \dfrac{78608 - 637\sqrt{8581}}{108000000}$

Total independent / Total : (13 / 23)

The set of coefficients evaluated here is that of the embedding on the y' formula (signified by `[p']` on the `ORDER` command), and it can be seen that τ''s are zero up to fourth-order and yield rational expressions from the fifth-order onwards. Also notice the information given after each order regarding the dependency of the τ''s. Standard output from the same run is given in Figure A.5, where the addition of order information collected from the run can be seen, and would have been included into the LaTeX output if the `LATEX` command had been given. It should be clear that, although only one set of coefficients is output (that on the `ORDER` command), all sets have been considered and the collated results given at the end. Other information presented only to the standard output is continuity and parameter information before the order check. This bar shows the result of a general $C^0/C^1/C^2$ continuity check

```
>C2 process - |C| = 1.00E-01, |A| = 6.86E-01,
                               ... |B*| = 7.27E+00/|B'*| = 3.64E+01

      1'.  Tau'(1,1) = b'(i) - 1 = 0
      2'.  Tau'(2,1) = b'(i)c(i) - 1/2! = 0

 ...deleted...

     23'.  Tau'(6,10) = b'(i)a(i,j)a(j,k)c(k) - 1/6! = [78608-
                    ... 637*root(8581)]/108000000

****************************************************************

6'th order formula
------------------
4'th order embedded process
---------------------------
5'th order dense
----------------
```

Figure A.5: Standard output from the RKN6(4)6FD$_6^5$ order check.

for a dense process plus norms of the a_{ij} and all 'b' parameters and the minimum separation of the c_i's, denoted by `|c|`. The particular norm used is specified by the `NORM` command ('maximum' by default). Average sizes of any b_i^* or $b_i'^*$ polynomials is taken over the set of the coefficients of the powers of σ, not the continuous quantities. Note that continuity checks are only performed using machine arithmetic, i.e., no use is made of the multi-integer type.

□

Example A.5.2 (A Butcher tableau in LaTeX for the Merson RK$_5^{4(3)}$ pair)
Using the ability of RK-AID to store rational forms, a method tableau may be output

with

```
read mer435
LaTeX
exact output
```

With the data-file of Example A.3.1 in "*mer435*", the following is put into the LaTeX output: In this way, whenever coefficients are available exactly, detailed LaTeX output

$$
\begin{array}{c|ccccc}
c & & & a & & \\
\hline
0 & & & & & \\
\frac{1}{3} & \frac{1}{3} & & & & \\
\frac{1}{3} & \frac{1}{6} & \frac{1}{6} & & & \\
\frac{1}{2} & \frac{1}{8} & 0 & \frac{3}{8} & & \\
1 & \frac{1}{2} & 0 & -\frac{3}{2} & 2 & \\
\hline
b & \frac{1}{6} & 0 & 0 & \frac{2}{3} & \frac{1}{6} \\
\widehat{b} & \frac{1}{10} & 0 & \frac{3}{10} & \frac{2}{5} & \frac{1}{5}
\end{array}
$$

is enabled, and output is exact, a tableau of parameters will be written into the ".tex" file. Note that external weights are appended to the bottom of the tableau, with extra stages for a dense process being given as a separate table if present. Following a command `BLOCK TABLeau`, extra a_{ij} are included in the main block and all b's (including dense polynomials) are appended to the right.

□

A.5.2 Expression output and evaluation

In addition to the *input* of coefficient data via the various arithmetic schemes, the state of a parameter or variable may be queried at any position in a data-file by enclosing its name inside single quotation marks, i.e., `'{exp}'`. Multiple displays of this form may be done by enclosing a list of expressions within the quotes, separated by "`&`" or "`and`". For example, the commands

```
exact output
v = 3
a[2, 1] = 1/v & a[3, 1] = -(1/v)/3 & a[3, 2] = (v+1)/3v
'v & a[2, 1] & a[3, 1] & a[3, 2]'
```

yield the following:

```
[v = 3]
[a[2, 1] = 1/3]
[a[3, 1] = -1/9]
[a[3, 2] = 4/9]
```

Possible confusion that may arise from use of the RK-AID expression parser can be eased by checking the values it produces in this way. Note that the output form is affected by the `FLOAT OUTput` and `EXACT OUTput` commands (see Section A.6.15). While not apparent from the example above, the `'{exp}'` command can itself contain calculations or combinations of parameters, i.e., `'a[3, 1] + a[3, 2]'` will give the necessary value of c_2 (for RK methods) in the form:

```
[a[3, 1] + a[3, 2] = 1/3]
```

Note that any quantities calculated from those entered (for example, the c_i from the a_{ij}) are determined *after* the command parsing is complete and so cannot be queried in this way.

Interactive use

A feature has been implemented that allows the arithmetic features of RK-AID to be accessed interactively from the front-end command line rather than during normal parsing. At the `rkaid:` prompt, multiple expressions may be entered in the form `'{exp1} & {exp2} & ...`, i.e., not closing the quotation as above. These commands are evaluated *immediately*, and so are very different from the normal data-file forms. This interaction allows true access to the coefficient information as it was at the *end of the previous run*, using 'float' or 'exact' forms as stored. For example, any c_i calculated from row-sum conditions in the previous run may now be checked — the query `'c(2)` would return `1/3` after running the data-file above. Also, if an order check has been performed, external weight polynomials will have been evaluated at a single value of σ, and their values can be displayed as in `'bs[#]`, for `#` a given stage. The value of method parameters can be changed from the front-end by making declarations of the usual form, i.e., `'{variable}={value}`, and will take effect until the coefficients are re-read. Variable declarations will only last until any data-file is next run.

These features allow the RK-AID front-end to be used as a rational or multi-precision calculator, including use of variables to store intermediate results. For this purpose, the additional commands `'FLOAT_OUTput`, `'EXACT_OUTput` and `'MODE` are available for immediate effect, with '`_OUTput`' optional in each case (noting the need for the under-score character '`_`'). The `'MODE` command displays the current arithmetic and output mode. To perform the calculation

$$(\sqrt{x} + x)/x^3$$

in extended-precision arithmetic for $x = 1.34$, say, can be achieved with

```
'float_output & x = 1.34 & num = sqrt(x) + x & num / x^3
```

giving the following output from the last expression:

```
    [num / x^3 = 1.03801984048861667964293356030973115821B0]
```

Note that the number of decimal figures depends on the compiled multi-integer size, and cannot be changed otherwise. The last calculation or value displayed can be referenced by the variable "`%`", and the above calculation can be similarly achieved with

```
'x = 1.34 & sqrt(x) + x & % / x^3
```

Of course, the expression could have been parenthesised and entered in a single step as `(sqrt(x) + x)/x^3`. Results can be converted from one arithmetic form to another by changing the mode as required. For example,

```
 'exact_output & sqrt(2) / 3
```

will display the result

```
    [sqrt(2) / 3 = root(2)/3]
```

noting that $\sqrt{2}$ has been kept in surd form (this can only be done for a *single* such root). The commands

```
 'float_output & %
```

will then yield:

```
    [% = 4.714045207910316829338962414032326928538B-1]
```

A.6 Summary of commands

Throughout, recognised command sections are *shown* in capitals wherever they appear (remembering that commands are, in fact, case-independent), in each case being followed by a description of function, usage and specifiers which, where optional, are denoted by an asterisk (*). As some RK-AID functions may be invoked by more than one (equivalent) command keyword, options are given as appropriate.

A.6.1 The `TITLE` command

The title command is used for naming a run of RK-AID such that all output files are specific to that run. In the absence of such a command, the name 'rkive' is used.

- *Automatic titling.* If the command `TITLE` is used in isolation, the title of the run will be taken from the method name, if one is present, where a method name is specified by the presence of a word of text starting with either "RK" or "RKN". In most cases, a word of text is any followed by either a space or a comma. The naming of methods in this way is recommended, and should be done whenever possible such that RK-AID then knows when a method tableau is to change. When performing several tasks on a given method from within batch-files or as successive runs on the command line, the presence of a method name will remove the need to re-read the coefficient data.

- *Specific titling**. The `TITLE` command may also be followed by any name you choose, i.e., `TITLE example` names the run 'example'.

A.6.2 The `PREFIX` and `POSTFIX` commands

These commands allow a prefix or postfix to be added to the current `TITLE`. For example, `TITLE, PREFIX new#` will name a default run as 'new#rkive', and `TITLE, POSTFIX #old` will name the run 'rkive#old'. This is especially useful when runs are titled automatically from method names (see above), and a new run is to be done without overwriting old files.

A.6.3 The `CUT` command

This is used for speed when reading assumed form data-files, and signifies that no commands are to be found thereafter.

A.6.4 The `ORDER` command

The `ORDER` command is fundamental for specifying which processes, and through what range of orders, a method or model is to be worked on.

- *Process type/order range.* The `ORDER` command must include specification of the process it refers to (as in Table A.1) and be followed by the range of orders to be considered in the form `n:m` where either `n`, or `n` and the colon (`:`) may be omitted in an obvious manner, e.g., `[all] ORDER :8` or `[s'] ORDER 10`.

  If the process to be considered is the lower-order '[p]' formula, the specifier need not be given as this is default, noting the simple case of generating a list of RK truncation error coefficients up to fourth-order with the command `order :4`, from Section A.2.

- *Specific τ's to consider (for one order only)**. For a single order, for the process specified, a range of τ's may be given after the order number, e.g.,

  ```
  Nystrom, [y'] ORDER 9, TAUs 68:72
  ```

- *Specifying the principal order of error terms**. In a situation where consideration should be taken away from *equations* and given to the error terms (τ's) themselves, the order at which this should occur is denoted as in

  ```
  [s] ORDER :9, ERROR/PRINCipal terms/coefficients 8:
  ```

  The keywords `ERROR` and `PRINCipal` are equivalent here, only one of which is required, and emphasising again that any command sections not in capitals are optional and are present for readability only.

- *Specification of a set of reduction data files**. When RK-AID is in reduction mode it writes a set of files to the current directory containing information regarding the independent set of error coefficients it processes — see the `REDUCe` command (Section A.6.19). A set of ".rd" files can be specified following a `USING` command, e.g., `[p'] ORDER :8, USING rkive`, in the default case. This causes all terms missing from the data-lists to be skipped. For high-order checks, this vastly speeds up calculation of truncation error norms, where clearly only an independent set of τ's is required.

A.6.5 The `USE` command

The `USE` command must start the line, and precedes information regarding which parameter forms are to be used in the model. Multiple coefficient sets may be included per single `USE` statement by separating them with either the "`&`" character or the word "`and`". Common forms are:

1. `USE Q's`

2. `USE Q(1:2)'s`

3. `USE Q's & R[h]'s`

4. `USE R[p](0)'s`

5. `USE R(0)'s`

6. `USE R[all](0)'s`

Note the two essentially different forms here, i.e., (1) and (3) refer to *all* parameters of that type, whereas the remainder are specific to ranges or individual classes. The last form relates to all of the 'R_0' class of parameters within the model, i.e., $R_{0i}, \hat{R}_{0i}$ and R^*_{0i}'s for an RK *or* RKN triple. The `'s` on each of these examples is optional.

A.6.6 Process specification commands

These consist of the following six keyword alternatives:

- the `RKN/NYSTrom` command,
- the `EXPLICIT` command,
- the `SEMI` command,
- the `IMPLICIT` command,
- the `TRIPLE/DENSE` command, and
- the `PAIR/EMBEDded` command.

These specify the special form of the model, for example:

1. `EXPLICIT RK TRIPLE`

2. `SEMI-IMPLICIT RKN`

3. `EMBEDded NYSTrom method`

Type (3) here is explicit by default, and by necessity, since only single implicit methods can be analysed, the embedding in this case would not be allowed for any other than an explicit method.

A.6.7 The `BACKUP` command

By default, RK-AID does not overwrite ".tex" or ".m" files without making a backup (of the same name, with a hash symbol (#) appended to the start). By giving a command `BACKUP OFF` or `NO BACKUP`, the files will be lost without prompting.

A.6.8 The `PORT` command

This keyword causes method coefficients to be written into the multi-purpose ".out" file, MACSYMA batch ".m" file (in reduction mode) and also to the standard output, where relevant. The form used is 'tailored' as appropriate, intended to facilitate porting from one representation to another, and is affected by the `FLOAT OUTput` and `EXACT OUTput` commands.

A.6.9 Typesetting commands

Various possibilities exist for modifying the output of RK-AID, mostly relating to ".tex" files. These commands are as follows:

- `LARGE`

  Produces all LaTeX output with bigger text.

- `CENTRE`

  Left-right centred LaTeX tables.

- `LANDscape`

  Modifies LaTeX page margins to produce output intended for printing in landscape mode.

- `LATEX`

  A ".tex" file of LaTeX commands is always produced by RK-AID giving a typeset version of the standard output. The `LATEX` command details this output and provides extra information, producing something more like a reference document.

- `WIDTH`

  Sets the width of the display (in characters) intended to view the standard output. This defaults to 78, convenient for the width of most standard terminals.

A.6.10 The `ASSUMEd` command

The `ASSUMEd` command signals either the input or output of a coefficient set in an assumed form (see Section A.3.2).

- *Block form**. The assumed block form is an alternative to the default assumed form, represented by the addition of the keyword `BLOCK`.

- *Output in assumed form**. If a coefficient set has been read in by the general non-assumed form method, addition of the keyword `OUTput` will cause a file to be written to the current directory containing the equivalent assumed (or assumed block) common-denominator form data-file, i.e., `OUTput ASSUMEd BLOCK form`. No output of assumed 'long form' data-files is possible. Note that the output here is not truly a common-denominator representation of the method coefficients, but uses 1 for every denominator.

A.6.11 The `INCLUDE/READ` command

Any RK-AID data-file may link onto another by use of this command (in one of the two forms), noting that the front-end is effectively a data-file for these purposes. For example, a file containing a command `INCLUDE file.dat` will cause the file "*file.dat*" in the current directory to be read following the *end* of the current file, i.e., not at the point it appears — the order that commands are read may be relevant, for example when variables are being used for parameter declarations. It should be noted that if the data-file in question is an assumed form coefficient file, it must be the *last* in the data-file structure.

A.6.12 The `EQUATion MODE` command

This causes order conditions to be output instead of the default truncation error coefficient form, the difference being in the removal of constants from the beginning of each condition.

- τ *inclusion**. By default, the specific τ from which an equation is generated is not shown. However, by addition of the keyword `TAU` to the `EQUATion MODE` command, this information is also given. Since the case where τ's are given equated to an order condition is not correct to the definition (or unique), a dagger (†) is shown on the τ symbol to make this distinction. For example, dense RK order conditions in the fifth-order (with τ's) are generated as follows:

```
[s] order 5
equation mode, with taus
```

1*. $^\dagger\tau_1^{(5)*} = \sum_{i} b_i^* c_i^4/\sigma^4 - \frac{1}{5}$

2*. $^\dagger\tau_2^{(5)*} = \sum_{ij} b_i^* c_i^2 a_{ij} c_j/\sigma^4 - \frac{1}{10}$

3*. $^\dagger\tau_3^{(5)*} = \sum_{ijk} b_i^* a_{ij} c_j a_{ik} c_k/\sigma^4 - \frac{1}{20}$

4*. $^\dagger\tau_4^{(5)*} = \sum_{ij} b_i^* c_i a_{ij} c_j^2/\sigma^4 - \frac{1}{15}$

5*. $^\dagger\tau_5^{(5)*} = \sum_{ij} b_i^* a_{ij} c_j^3/\sigma^4 - \frac{1}{20}$

6*. $^\dagger\tau_6^{(5)*} = \sum_{ijk} b_i^* c_i a_{ij} a_{jk} c_k/\sigma^4 - \frac{1}{30}$

7*. $^\dagger\tau_7^{(5)*} = \sum_{ijk} b_i^* a_{ij} c_j a_{jk} c_k/\sigma^4 - \frac{1}{40}$

8*. $^\dagger\tau_8^{(5)*} = \sum_{ijk} b_i^* a_{ij} a_{jk} c_k^2/\sigma^4 - \frac{1}{60}$

9*. $^\dagger\tau_9^{(5)*} = \sum_{ijkl} b_i^* a_{ij} a_{jk} a_{kl} c_l/\sigma^4 - \frac{1}{120}$

A.6.13 The `OPTions/ALTernatives` command

Where 'Q' and 'R' parameter forms are being considered simultaneously, it is often convenient to have several possible algebraic options displayed concurrently. By default, these options will only be displayed when the 'Q' and 'R' forms *overlap* on the tree. By giving the `OPTions` command, where relevant, a form will be given with 'Q' parameters only, alongside the 'QR form'.

- *Concurrent display of the standard form**. The classical '*abc*' form will also be given following the command `show ALL OPTions`. As an example, for the set of fourth-order RK $\hat{\tau}$'s:

  ```
  [h] order :4
  use Q's & R[h]'s
  show all alternatives
  ```

  gives

  $\hat{1}$. $\hat{\tau}_1^{(1)} = \sum_i \hat{b}_i - 1$

  $\hat{2}$. $\hat{\tau}_1^{(2)} = \sum_i \hat{b}_i c_i - \frac{1}{2!}$

  $\hat{3}$. $\hat{\tau}_1^{(3)} = \frac{1}{2}\sum_i \hat{b}_i c_i^2 - \frac{1}{3!}$

  $\hat{4}$. $\hat{\tau}_2^{(3)} = \sum_{ij} \hat{b}_i a_{ij} c_j - \frac{1}{3!} = \sum_i \hat{b}_i Q_{i1} + \hat{\tau}_1^{(3)}$ or $\sum_i \hat{R}_{0i} c_i + \hat{\tau}_1^{(2)} - 2\hat{\tau}_1^{(3)}$

  $\hat{5}$. $\hat{\tau}_1^{(4)} = \frac{1}{6}\sum_i \hat{b}_i c_i^3 - \frac{1}{4!}$

  $\hat{6}$. $\hat{\tau}_2^{(4)} = \sum_{ij} \hat{b}_i c_i a_{ij} c_j - \frac{3}{4!} = \sum_i \hat{b}_i c_i Q_{i1} + 3\hat{\tau}_1^{(4)}$ or $\sum_i \hat{R}_{1i} c_i + \frac{1}{2}\hat{\tau}_1^{(2)} - 3\hat{\tau}_1^{(4)}$

  $\hat{7}$. $\hat{\tau}_3^{(4)} = \frac{1}{2}\sum_{ij} \hat{b}_i a_{ij} c_j^2 - \frac{1}{4!} = \frac{1}{2}\sum_i \hat{b}_i Q_{i2} + \hat{\tau}_1^{(4)}$ or $\frac{1}{2}\sum_i \hat{R}_{0i} c_i^2 + \hat{\tau}_1^{(3)} - 3\hat{\tau}_1^{(4)}$

  $\hat{8}$. $\hat{\tau}_4^{(4)} = \sum_{ijk} \hat{b}_i a_{ij} a_{jk} c_k - \frac{1}{4!} = \sum_{ij} \hat{b}_i a_{ij} Q_{j1} + \hat{\tau}_3^{(4)}$ or $\sum_i \hat{R}_{0i} Q_{i1} + \hat{\tau}_2^{(3)} - \hat{\tau}_2^{(4)} \ldots$[5]

  Note that for the truncation error coefficients here, as opposed to in equation mode, use of 'Q' and 'R' parameters results in the need to show relevant linear

[5]Remaining combinations have been ommitted.

combinations of $\hat{\tau}$'s, and that this is done with each of the multiple options.

A.6.14 The `TREE` command

This displays the relevant tree alongside each truncation error coefficient in both the standard output and LaTeX. The tree is given in a form whereby the nodes are represented in a left-centre-right manner by their height from the root, which has height zero (Scoins [21]).

A.6.15 The `CHECK` command

In the presence of a set of RK or RKN method parameters, the `CHECK` command causes truncation error coefficients (or equations) to be numerically evaluated and the result displayed.

Equation mode checking

In this case, each equation is evaluated as it appears to the output, and the result equated to its symbolic expression.

τ checking

Out of equation mode, RK-AID displays the τ's themselves, and so this is what would ideally be evaluated. However, when 'Q' or 'R' parameters are present in the output, there are two possible modes of operation:

1. *Raw τ checking.* By default, RK-AID will evaluate the *raw* form of the coefficient, i.e., the '*abc*' form, independent of whether or not 'Q' or 'R' parameters are being displayed. Checking the fifth-order τ's for the Merson RK$_5^{4(3)}$ from Example A.3.1 with

```
read mer435
use Q's & R's
order 5
check
```

gives

1. $\tau_1^{(5)} = \frac{1}{24}\sum_i b_i c_i^4 - \dfrac{1}{5!} = 3.4722\times10^{-04}$
2. $\tau_2^{(5)} = \frac{1}{2}\sum_i b_i c_i^2 Q_{i1} + 6\tau_1^{(5)}$ or $\frac{1}{2}\sum_i R_{2i}c_i + \dfrac{1}{6}\tau_1^{(2)} - 4\tau_1^{(5)} = 2.0833\times10^{-03}$
3. $\tau_3^{(5)} = \frac{1}{2}\sum_i b_i Q_{i1}^2 + \tau_2^{(5)} - 3\tau_1^{(5)} = 1.0417\times10^{-03}$
4. $\tau_4^{(5)} = \frac{1}{2}\sum_i b_i c_i Q_{i2} + 4\tau_1^{(5)}$ or $\frac{1}{2}\sum_i R_{1i}c_i^2 + \dfrac{1}{2}\tau_1^{(3)} - 6\tau_1^{(5)} = 1.3889\times10^{-03}$
5. $\tau_5^{(5)} = \frac{1}{6}\sum_i b_i Q_{i3} + \tau_1^{(5)}$ or $\frac{1}{6}\sum_i R_{0i}c_i^3 + \tau_1^{(4)} - 4\tau_1^{(5)} = -1.3889\times10^{-03}$
6. $\tau_6^{(5)} = \sum_i R_{1i}Q_{i1} + \dfrac{1}{2}\tau_2^{(3)} - \tau_2^{(5)} + \tau_4^{(5)} - \dfrac{1}{2}\tau_1^{(3)} + 6\tau_1^{(5)} = 1.3889\times10^{-03}$
7. $\tau_7^{(5)} = \sum_i R_{0i}c_i Q_{i1} + \tau_2^{(4)} - 2\tau_2^{(5)} + 3\tau_5^{(5)} - 3\tau_1^{(4)} + 12\tau_1^{(5)} = -4.1667\times10^{-03}$
8. $\tau_8^{(5)} = \frac{1}{2}\sum_i R_{0i}Q_{i2} + \tau_3^{(4)} - \tau_4^{(5)} + \tau_5^{(5)} - \tau_1^{(4)} + 4\tau_1^{(5)} = -1.3889\times10^{-03}$
9. $\tau_9^{(5)} = \sum_{ij} R_{0i}a_{ij}Q_{j1} + \tau_4^{(4)} - \tau_6^{(5)} + \tau_8^{(5)} - \tau_3^{(4)} + \tau_4^{(5)} = -1.3889\times10^{-03}$

2. *Simplified τ checking.* By use of the `SIMPlify` command, the expressions are calculated using any 'Q' or 'R' parameters present, and so, without the facility for storage of the linear dependencies required to make the τ correct, only the summation part is evaluated, and the linear dependencies moved to the left hand side of the expression. This can be seen by changing the `CHECK` command above to `CHECK, and SIMPLify`[6], giving:

   1. $\tau_1^{(5)} = \frac{1}{24}\sum_i b_i c_i^4 - \dfrac{1}{5!} = 3.4722\times10^{-04}$
   2. $\tau_2^{(5)} - \frac{1}{6}\tau_1^{(2)} + 4\tau_1^{(5)} = \frac{1}{2}\sum_i R_{2i}c_i = 3.4722\times10^{-03}$
   3. $\tau_3^{(5)} - \tau_2^{(5)} + 3\tau_1^{(5)} = \frac{1}{2}\sum_i b_i Q_{i1}^2 = 0.0$
   4. $\tau_4^{(5)} - \frac{1}{2}\tau_1^{(3)} + 6\tau_1^{(5)} = \frac{1}{2}\sum_i R_{1i}c_i^2 = 3.4722\times10^{-03}$
   5. $\tau_5^{(5)} - \tau_1^{(4)} + 4\tau_1^{(5)} = \frac{1}{6}\sum_i R_{0i}c_i^3 = 0.0$
   6. $\tau_6^{(5)} - \frac{1}{2}\tau_2^{(3)} + \tau_2^{(5)} - \tau_4^{(5)} + \frac{1}{2}\tau_1^{(3)} - 6\tau_1^{(5)} = \sum_i R_{1i}Q_{i1} = 0.0$

[6]Noting the comma ending the `CHECK` command, necessary here as `CHECK` can take an argument.

7. $\tau_7^{(5)}-\tau_2^{(4)}+2\tau_2^{(5)}-3\tau_5^{(5)}+3\tau_1^{(4)}-12\tau_1^{(5)} = \sum_i R_{0i}c_iQ_{i1} = 0.0$
8. $\tau_8^{(5)}-\tau_3^{(4)}+\tau_4^{(5)}-\tau_5^{(5)}+\tau_1^{(4)}-4\tau_1^{(5)} = \frac{1}{2}\sum_i R_{0i}Q_{i2} = 0.0$
9. $\tau_9^{(5)}-\tau_4^{(4)}+\tau_6^{(5)}-\tau_8^{(5)}+\tau_3^{(4)}-\tau_4^{(5)} = \sum_{ij} R_{0i}a_{ij}Q_{j1} = 0.0$

It should be noted that, unlike most other equation modes, in the event of there being more than one way to write a term due to use of 'Q' *and* 'R' parameters, only one form is shown. This can be seen here with reference to the previous output, for $\tau_2^{(5)}$, $\tau_4^{(5)}$ and $\tau_5^{(5)}$.

- *Specific process checking**. Given a command of the form `CHECK method`, the file "*method*" will be read in via the `READ` command, and a `CHECK` command executed.

- *Extended-precision floating-point checking**. The `FLOAT` command switches from machine to software arithmetic and extends the precision of floating-point calculations to use all the available digits of the implemented multi-integer type. Since each integer can safely contain an even number of digits no greater than 8 on most systems, a version of RK-AID with a multi-integer size of 5 (the default number) can store 40 decimal digits. For the purpose of calculations, a digit is reserved for addition overflow and so the smallest number that can be added to 1 so as not to give 1 (the arithmetic *epsilon*) is $1.0{\times}10^{-38}$ in this case. By default, when multi-precision arithmetic is being used, decimal output is still done to only a small number of significant figures for brevity. If the keyword `OUTput` is included with the `FLOAT` command, i.e., `FLOAT OUTput`, the full number of decimal places will be shown. Such a multi-precision number is designated by the exponent indicator `B`, for 'big-float', i.e., `3.33333333333333333333B-1`. This is consistent with the system used by MACSYMA.

- *Exact process checking**. The `EXACT` command switches from the default floating-point arithmetic to an exact rational/surd arithmetic. An advantage of working with exact arithmetic (when the `APPROXimate` command has not been given) is that RK-AID recognises a *true* zero, and as a result can be more intelligent in its analysis[7]. In the above case for the Merson $\mathrm{RK}_5^{4(3)}$ pair, all but the quadrature equation in the fifth-order evaluate to zero in at least one form, so now with exact checking:

```
read mer435
use Q's & R's
order 5
check, and simplify
exact arithmetic
```

gives

$$
\begin{array}{rll}
 & 1. & \tau_1^{(5)} = \frac{1}{24}\sum_i b_i c_i^4 - \dfrac{1}{5!} = 3.4722\times 10^{-04} \\
* & 2. & \tau_2^{(5)} = 6\tau_1^{(5)} \\
* & 3. & \tau_3^{(5)} = \tau_2^{(5)} - 3\tau_1^{(5)} \\
* & 4. & \tau_4^{(5)} = 4\tau_1^{(5)} \\
* & 5. & \tau_5^{(5)} = \tau_1^{(4)} - 4\tau_1^{(5)} \\
* & 6. & \tau_6^{(5)} = \tau_4^{(5)} \\
* & 7. & \tau_7^{(5)} = 3\tau_5^{(5)} \\
* & 8. & \tau_8^{(5)} = \tau_5^{(5)} \\
* & 9. & \tau_9^{(5)} = \tau_8^{(5)}
\end{array}
$$

Total independent / Total : (1 / 9)

- *Exact checking by accepted round-off**. A threshold may be imposed on floating-point (double- or multi-precision) arithmetic enabling a zero to be recognised, using the `THRESHold` command. In this mode, exact checking can be emulated to achieve the same result as above, using a data-file of, say,

```
read mer435
```

[7]If an approximate coefficient set is checked using the `EXACT` mode, no precision will be lost in the checking process, but a zero will not be recognised as such.

```
use Q's & R's
order 5
check, and simplify, threshold 1E-14
```

- *Exact process checking with exact output**. By default, when exact arithmetic is used, the output is in floating-point form on the assumption that all that is required is an exact zero, not the actual rationals. Rational/surd output may be obtained by including the keyword `OUTput` onto the `EXACT` command.

A.6.16 The `INDEPendent/REMAINing` command

In equation mode, dependent terms are usually marked with an asterisk ($*$). By inclusion of the command `INDEPendent terms only` or `REMAINing equations only`, dependent equations are not displayed.

A.6.17 The `CONTinuous` command

In equation mode, where dependent equations are being omitted from the output (see above), this command causes numbering to be done sequentially for the *entire run*, and is intended for production of a complete model of a method pair or triple. This should not be confused with the τ numbering scheme, which is fixed in the natural sequence of generation. Where the order of principal error terms is known (from the `ORDER` command), these truncation error terms are not numbered, and the next equation output continues the numbering scheme.

A.6.18 The `STABility ANALysis` command

This command causes estimated real negative stability limits of each process in the current model (see Sections 2.3.2 and 2.7.1) to be written to the standard output (no LaTeX output is done). Plots of continuous stability limits over σ (using a number of points as specified by the `POINTS` command — 32 by default) are also written into files of the current directory. Such calculations are only done with machine arithmetic, i.e., even after giving a `FLOAT` or `EXACT` command.

A.6.19 The `REDUCe` command

The `REDUCtion` mode is the primary environment for presentation of algebraically simplified truncation error terms and order conditions in a form suited to models under development. The simplification is done principally by recognising patterns of zeros in the recursive summation process, and so it is required that an exact arithmetic is in operation or at least emulated with the `THRESHold` command. A basic reduced model can be demonstrated using an independent set of explicit fourth-order RK equations as an example:

```
[h] order :4
reduction mode
equation mode
use Q's
```

gives

$$
\begin{array}{ll}
\hat{1}. & \displaystyle\sum_{i=1}^{s} \hat{b}_i - 1 \\
\hline
\hat{2}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i c_i - \frac{1}{2} \\
\hline
\hat{3}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i c_i^2 - \frac{1}{3} \\
\hat{4}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i Q_{i1} \\
\hline
\hat{5}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i c_i^3 - \frac{1}{4} \\
\hat{6}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i c_i Q_{i1} \\
\hat{7}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i Q_{i2} \\
\hat{8}. & \displaystyle\sum_{i=3}^{s} \hat{b}_i \sum_{j=2}^{i-1} a_{ij} Q_{j1}
\end{array}
$$

where, as a result of the reduction mode, summations have been split, the limits on the summations have been adjusted to take account of the explicit nature of the model (explicit RK is default), and the upper limit on the i summation has been left free, as s. Note that, by the explicit nature, it is not necessary to specifically impose $Q_{1k} = 0,\ k = 1, 2$. This is the 'free stage' mode in which conditions may be imposed, and the results seen algebraically in the output. For example, adding the simplifying conditions $\mathtt{Q(3:s,1) = 0}$ to the command file above will yield

$$\begin{array}{ll}
\hat{1}. & \displaystyle\sum_{i=1}^{s} \hat{b}_i - 1 \\
\hline
\hat{2}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i c_i - \frac{1}{2} \\
\hline
\hat{3}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i c_i^2 - \frac{1}{3} \\
\hat{4}. & \hat{b}_2 Q_{2,1} \\
\hline
\hat{5}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i c_i^3 - \frac{1}{4} \\
\hat{6}. & \hat{b}_2 c_2 Q_{2,1} \\
\hat{7}. & \displaystyle\sum_{i=2}^{s} \hat{b}_i Q_{i2} \\
\hat{8}. & \displaystyle\sum_{i=3}^{s} \hat{b}_i a_{i2}
\end{array}$$

It is not necessarily recommended, but the variable '`s`' for the number of stages can be replaced by any other name, as from its first appearance in the relevant place of a coefficient definition, this alternative name will be used. Extra stages for a dense process may be set up with a command of the form `star = s + n` or `s = star - n`, where `star` can be replaced by *any* otherwise unused variable name, and `n` should be replaced with the number of extra stages required. Note that s^* is assumed to be equal to s otherwise, and the two are interchangeable.

Reduction mode always writes a set of data-files to the current directory that contain information regarding the independent set of truncation error terms processed. These will be named by the current `TITLE`, and prefixed by the process type, for example, "*tds_rkive.rd*" and "*tds_rkive.tab*" for the default output files of a y' dense process. The ".rd" (reduction data) files may be read into RK-AID with a `USING` keyword on the `ORDER` command (Section A.6.4), whereby terms not contained in the data are missed from the order check, speeding up calculation of error norms. These files also contain normed multiples of each independent term, so norms of the complete set can be calculated with the correct weighting. The ".tab" files are not used by RK-AID itself, but always come in a pair with a ".rd" file and are for information only. They contain a data set for construction of the error coefficients that make up the independent set, including the 'tree' itself. These files have been used for the calculation of error norms outside RK-AID in optimisation processes.

The reduction mode may also be applied to 'complete' methods, in which case 'Q' and 'R' parameters will be calculated when needed to give the reduced form. For example, applying reduction mode to the RK5(4)7FMD$^4_7$ of Example A.3.2 with

```
include 547fmd4
[h] order :6, error terms 6:
use Q's & R[h]'s
reduce, equation mode
exact arithmetic
```

gives the independent equation set:

$$\begin{array}{cll}
 & \hat{1}. & \sum_{i=1}^{6} \hat{b}_i - 1 \\
\hline
 & \hat{2}. & \sum_{i=3}^{6} \hat{b}_i c_i - \frac{1}{2} \\
\hline
 & \hat{3}. & \sum_{i=3}^{6} \hat{b}_i c_i^2 - \frac{1}{3} \\
\hline
 & \hat{5}. & \sum_{i=3}^{6} \hat{b}_i c_i^3 - \frac{1}{4} \\
* & \hat{6}. & \sum_{i=3}^{5} \hat{R}_{1i} c_i \\
\hline
 & \hat{9}. & \sum_{i=3}^{6} \hat{b}_i c_i^4 - \frac{1}{5} \\
* & \hat{10}. & \sum_{i=2}^{5} \hat{R}_{2i} c_i \\
* & \hat{12}. & \sum_{i=3}^{5} \hat{R}_{1i} c_i^2 \\
* & \hat{13}. & \sum_{i=3}^{6} \hat{b}_i Q_{i3}
\end{array}$$

$\hat{18}.$ $\hat{\tau}_1^{(6)} = \frac{1}{120}\sum_{i=3}^{6}\hat{b}_i c_i^5 - \frac{1}{6!}$

$\ast$ $\hat{19}.$ $\hat{\tau}_2^{(6)} = 10\hat{\tau}_1^{(6)}$

$\ast$ $\hat{20}.$ $\hat{\tau}_3^{(6)} = 3\hat{\tau}_2^{(6)} - 15\hat{\tau}_1^{(6)}$

$\ast$ $\hat{21}.$ $\hat{\tau}_4^{(6)} = 10\hat{\tau}_1^{(6)}$

$\ast$ $\hat{22}.$ $\hat{\tau}_5^{(6)} = \hat{\tau}_2^{(6)} + \hat{\tau}_4^{(6)} - 10\hat{\tau}_1^{(6)}$

$\hat{23}.$ $\hat{\tau}_6^{(6)} = \frac{1}{6}\sum_{i=3}^{5}\hat{R}_{1i} c_i^3 + \frac{1}{2}\hat{\tau}_1^{(4)} - 10\hat{\tau}_1^{(6)}$

$\ast$ $\hat{24}.$ $\hat{\tau}_7^{(6)} = \hat{\tau}_1^{(5)} - 5\hat{\tau}_1^{(6)}$

$\hat{25}.$ $\hat{\tau}_8^{(6)} = \frac{1}{2}\hat{R}_{2,2}Q_{2,1} + \frac{1}{6}\hat{\tau}_2^{(3)} - \hat{\tau}_2^{(6)} + \hat{\tau}_4^{(6)} - \frac{1}{6}\hat{\tau}_1^{(3)} + 10\hat{\tau}_1^{(6)}$

$\ast$ $\hat{26}.$ $\hat{\tau}_9^{(6)} = \hat{\tau}_5^{(6)} + \hat{\tau}_8^{(6)} - \hat{\tau}_4^{(6)}$

$\ast$ $\hat{27}.$ $\hat{\tau}_{10}^{(6)} = \frac{1}{2}\hat{\tau}_2^{(4)} - 3\hat{\tau}_2^{(6)} + 3\hat{\tau}_6^{(6)} - \frac{3}{2}\hat{\tau}_1^{(4)} + 30\hat{\tau}_1^{(6)}$

$\ast$ $\hat{28}.$ $\hat{\tau}_{11}^{(6)} = \hat{\tau}_2^{(5)} - 3\hat{\tau}_2^{(6)} + 6\hat{\tau}_7^{(6)} - 6\hat{\tau}_1^{(5)} + 30\hat{\tau}_1^{(6)}$

$\ast$ $\hat{29}.$ $\hat{\tau}_{12}^{(6)} = \hat{\tau}_3^{(5)} - \hat{\tau}_3^{(6)} + \hat{\tau}_{11}^{(6)} - \hat{\tau}_2^{(5)} + 3\hat{\tau}_2^{(6)} - 3\hat{\tau}_7^{(6)} + 3\hat{\tau}_1^{(5)} - 15\hat{\tau}_1^{(6)}$

$\ast$ $\hat{30}.$ $\hat{\tau}_{13}^{(6)} = \frac{1}{2}\hat{\tau}_3^{(4)} - \hat{\tau}_4^{(6)} + \hat{\tau}_6^{(6)} - \frac{1}{2}\hat{\tau}_1^{(4)} + 10\hat{\tau}_1^{(6)}$

$\ast$ $\hat{31}.$ $\hat{\tau}_{14}^{(6)} = \hat{\tau}_4^{(5)} - 2\hat{\tau}_4^{(6)} + 4\hat{\tau}_7^{(6)} - 4\hat{\tau}_1^{(5)} + 20\hat{\tau}_1^{(6)}$

$\ast$ $\hat{32}.$ $\hat{\tau}_{15}^{(6)} = \hat{\tau}_5^{(5)} - \hat{\tau}_6^{(6)} + \hat{\tau}_7^{(6)} - \hat{\tau}_1^{(5)} + 5\hat{\tau}_1^{(6)}$

$\hat{33}.$ $\hat{\tau}_{16}^{(6)} = Q_{2,1}\sum_{i=3}^{5}\hat{R}_{1i}a_{i2} + \frac{1}{2}\hat{\tau}_4^{(4)} - \hat{\tau}_8^{(6)} + \hat{\tau}_{13}^{(6)} - \frac{1}{2}\hat{\tau}_3^{(4)} + \hat{\tau}_4^{(6)}$

$\ast$ $\hat{34}.$ $\hat{\tau}_{17}^{(6)} = \hat{\tau}_6^{(5)} - 2\hat{\tau}_8^{(6)} + \hat{\tau}_{14}^{(6)} - \hat{\tau}_4^{(5)} + 2\hat{\tau}_4^{(6)}$

$\ast$ $\hat{35}.$ $\hat{\tau}_{18}^{(6)} = \hat{\tau}_7^{(5)} - \hat{\tau}_{10}^{(6)} + 3\hat{\tau}_{15}^{(6)} - 3\hat{\tau}_5^{(5)} + 3\hat{\tau}_6^{(6)}$

$\ast$ $\hat{36}.$ $\hat{\tau}_{19}^{(6)} = \hat{\tau}_8^{(5)} - \hat{\tau}_{13}^{(6)} + \hat{\tau}_{15}^{(6)} - \hat{\tau}_5^{(5)} + \hat{\tau}_6^{(6)}$

$\ast$ $\hat{37}.$ $\hat{\tau}_{20}^{(6)} = \hat{\tau}_9^{(5)} - \hat{\tau}_{16}^{(6)} + \hat{\tau}_{19}^{(6)} - \hat{\tau}_8^{(5)} + \hat{\tau}_{13}^{(6)}$

Figure A.6: Reduction mode applied to principal error terms of the RK5(4)7FM.

and principal error terms in Figure A.6 (assuming the RK5(4)7FMD$_7^4$ data-file is available in "*547fmd4*").

- *Reduction with storage**. Ordinarily, terms derived from a single tree are considered in isolation with no storage facility to relate one to another. By use of the `FULL REDUCtion` command, the values of all independent terms, or the linear combination of τ's arising from a dependent term, are stored dynamically. This is a very powerful feature for the simplification of principal truncation error terms arrising from a reduced model. To illustrate this, by inclusion of the keyword `FULL` onto the `REDUCtion` command of the previous example, the sixth-order truncation error terms in Figure A.6 become:

$\hat{18}$. $\hat{\tau}_1^{(6)} = \frac{1}{120}\sum_{i=3}^{6}\hat{b}_i c_i^5 - \frac{1}{6!}$

* $\hat{19}$. $\hat{\tau}_2^{(6)} = 10\hat{\tau}_1^{(6)}$

* $\hat{20}$. $\hat{\tau}_3^{(6)} = 15\hat{\tau}_1^{(6)}$

* $\hat{21}$. $\hat{\tau}_4^{(6)} = 10\hat{\tau}_1^{(6)}$

* $\hat{22}$. $\hat{\tau}_5^{(6)} = 10\hat{\tau}_1^{(6)}$

$\hat{23}$. $\hat{\tau}_6^{(6)} = \frac{1}{6}\sum_{i=3}^{5}\hat{R}_{1i}c_i^3 - 10\hat{\tau}_1^{(6)}$

* $\hat{24}$. $\hat{\tau}_7^{(6)} = -5\hat{\tau}_1^{(6)}$

$\hat{25}$. $\hat{\tau}_8^{(6)} = \frac{1}{2}\hat{R}_{2,2}Q_{2,1} + 10\hat{\tau}_1^{(6)}$

* $\hat{26}$. $\hat{\tau}_9^{(6)} = \hat{\tau}_8^{(6)}$

* $\hat{27}$. $\hat{\tau}_{10}^{(6)} = 3\hat{\tau}_6^{(6)}$

* $\hat{28}$. $\hat{\tau}_{11}^{(6)} = -30\hat{\tau}_1^{(6)}$

* $\hat{29}$. $\hat{\tau}_{12}^{(6)} = -15\hat{\tau}_1^{(6)}$

* $\hat{30}$. $\hat{\tau}_{13}^{(6)} = \hat{\tau}_6^{(6)}$

* $\hat{31}$. $\hat{\tau}_{14}^{(6)} = -20\hat{\tau}_1^{(6)}$

* $\hat{32}$. $\hat{\tau}_{15}^{(6)} = -\hat{\tau}_6^{(6)}$

$\hat{33}$. $\hat{\tau}_{16}^{(6)} = Q_{2,1}\sum_{i=3}^{5}\hat{R}_{1i}a_{i2} + \hat{\tau}_6^{(6)} - \hat{\tau}_8^{(6)} + 10\hat{\tau}_1^{(6)}$

* $\hat{34}$. $\hat{\tau}_{17}^{(6)} = -2\hat{\tau}_8^{(6)}$

* $\hat{35}$. $\hat{\tau}_{18}^{(6)} = -3\hat{\tau}_{6}^{(6)}$

* $\hat{36}$. $\hat{\tau}_{19}^{(6)} = -\hat{\tau}_{6}^{(6)}$

* $\hat{37}$. $\hat{\tau}_{20}^{(6)} = -\hat{\tau}_{16}^{(6)}$

Here, note that the relevant linear combinations have been tidied up, and importantly, that *equations* output in this mode are assumed to be zero in the process.

A.6.20 The `STAGEs` command

A specific number of stages can be associated with a 'free stage' model using a command of the form `n STAGEs`, where `n` is the total number of stages in the model (i.e., s^* for a triple, or s otherwise). For example, explicit RK truncation error coefficients up to fourth-order in 3 stages are as follows:

```
order :4
reduce, in 3 stages
```

1. $\tau_1^{(1)} = \sum_{i=1}^{3} b_i - 1$

---

2. $\tau_1^{(2)} = \sum_{i=2}^{3} b_i c_i - \frac{1}{2!}$

---

3. $\tau_1^{(3)} = \frac{1}{2}\sum_{i=2}^{3} b_i c_i^2 - \frac{1}{3!}$
4. $\tau_2^{(3)} = b_3 a_{3,2} c_2 - \frac{1}{3!}$

---

5. $\tau_1^{(4)} = \frac{1}{6}\sum_{i=2}^{3} b_i c_i^3 - \frac{1}{4!}$
6. $\tau_2^{(4)} = b_3 c_3 a_{3,2} c_2 - \frac{3}{4!}$
7. $\tau_3^{(4)} = \frac{1}{2} b_3 a_{3,2} c_2^2 - \frac{1}{4!}$
8. $\tau_4^{(4)} = -\frac{1}{4!}$

Note that when such a command is in force, method stages must still be referenced with respect to s, as internally the description of the process is as a free stage model. For example, in the above case, if the condition $Q_{31} = 0$ were to be imposed the command would be $\texttt{Q}(\texttt{s},\texttt{1}) = \texttt{0}$.

A.6.21 The `PARAMetric/SYMbolic/ALGebraic` command

The `PARAMetric` command specifies a process from the current model to output in a parameterised form, for example, `PARAMeterise [h']` for the higher-order y' process of an RKN method. An example parameterised model can be generated by the following commands:

```
Nystrom
parameterise [h'] order :8
b[h'](2) = 0 & Q(3:s,1) = 0
```

producing a reduced set of explicit eighth-order RKN equations for $\widehat{y}'$ as follows:

1. $\sum_{i=1}^{s} \hat{b}'_i - 1$
2. $\sum_{i=3}^{s} \hat{b}'_i c_i^k - \frac{1}{k+1}, \quad k = 1:7$
3. $\sum_{i=3}^{s} \hat{b}'_i c_i^k Q_{i2}, \quad k = 0:3$
4. $\sum_{i=3}^{s} \hat{b}'_i c_i^k Q_{i3}, \quad k = 0:2$
5. $\sum_{i=3}^{s} \hat{b}'_i c_i^k a_{i2}, \quad k = 0:2$
6. $\sum_{i=3}^{s} \hat{b}'_i c_i^k Q_{i4}, \quad k = 0:1$
7. $\sum_{i=3}^{s} \hat{b}'_i c_i^l \sum_{j=2}^{i-1} a_{ij} c_j^m Q_{j2}, \quad l + m < 2$
8. $\sum_{i=3}^{s} \hat{b}'_i Q_{i5}$
9. $\sum_{i=3}^{s} \hat{b}'_i \sum_{j=2}^{i-1} a_{ij} Q_{j3}$
10. $\sum_{i=4}^{s} \hat{b}'_i \sum_{j=3}^{i-1} a_{ij} a_{j2}$

The parameterisation mode (also entered by inclusion of the keywords `SYMbolic` or `ALGebraic`) runs on top of the reduction mode, whereby the `CONTinuous numbering` and `INDEPendent equations` commands are also in effect. Here, equations that are parameterisable by c_i's or Q_{ik}'s are assumed to obey the same reduction rules, i.e., any equations solved by the current model are assumed also to be solved when any node is generalised by the addition of extra 'c' or 'Q' parameters. This is not necessarily true algebraically, but can be useful in practice. Note that the `SPLIT` mode is incompatible with the parameterisation schemes, but the *manual* assumed algebraic reduction scheme is valid in certain (not particularly useful) circumstances — namely for simplification in the *last* order of any model, where no parameterisation is involved.

- *'R form' parameterisation**. 'Q' parameters are always used in parameterisation mode, whether they are being used for simplification purposes or not. In the case where 'R' parameters are also being employed (which is optional), there are two modes of operation. A form of parameterisation making more use of 'R' coefficients instead of the default 'Q form' is signified by inclusion of the keywords `R FORM` onto the `PARAMetric` command. The difference is concerned with the inter-dependence of Q_{ik} and R_{kj} parameters. For example, here is the set of explicit fourth-order RK equations (again) with Q's and R's:

```
order :4
equation mode
use Q's & R's
reduce
```

1. $\displaystyle\sum_{i=1}^{s} b_i - 1$

2. $\displaystyle\sum_{i=2}^{s} b_i c_i - \frac{1}{2}$

3. $\displaystyle\sum_{i=2}^{s} b_i c_i^2 - \frac{1}{3}$

4. $\sum_{i=2}^{s} b_i Q_{i1}$ or $\sum_{i=2}^{s} R_{0i} c_i$

---

5. $\sum_{i=2}^{s} b_i c_i^3 - \frac{1}{4}$
6. $\sum_{i=2}^{s} b_i c_i Q_{i1}$ or $\sum_{i=2}^{s} R_{1i} c_i$
7. $\sum_{i=2}^{s} b_i Q_{i2}$ or $\sum_{i=2}^{s} R_{0i} c_i^2$
8. $\sum_{i=2}^{s} R_{0i} Q_{i1}$

noting the options put onto equations with a choice of representation. This may be parameterised in two ways, first the default 'Q form':

```
parameterise, order :4
use R's
```

1. $\sum_{i=1}^{s} b_i - 1$
2. $\sum_{i=2}^{s} b_i c_i^k - \frac{1}{k+1}, \quad k = 1:3$
3. $\sum_{i=2}^{s} b_i c_i^k Q_{i1}, \quad k = 0, \quad 1$
4. $\sum_{i=2}^{s} b_i Q_{i2}$
5. $\sum_{i=2}^{s} R_{0i} Q_{i1},$

or the 'R form':

```
parameterise, R form, order :4
use R's
```

1. $\sum_{i=1}^{s} b_i - 1$
2. $\sum_{i=2}^{s} b_i c_i^k - \frac{1}{k+1}, \quad k = 1:3$

3. $\sum_{i=2}^{s} R_{1i} c_i$
4. $\sum_{i=2}^{s} R_{0i} c_i^k, \quad k = 1, \quad 2$
5. $\sum_{i=2}^{s} R_{0i} Q_{i1}$

these two models being equivalent to the initial, non-parameterised, form.

A.6.22 The SPLIT command

The SPLIT command specifies a process from the current model to output in a form simplified by assumptions of linear independence on 'split' equation summations (see Section 3.3.4). Command syntax is to include the required process on the same line as the SPLIT keyword, e.g., `SPLIT [s]` would relate to an RK or RKN dense process on y (the '[p]' formula is default). The application of this complex feature is best understood by examples, suitable cases can be found in Section 3.3.4 and in Chapters 7 and 8.

A.6.23 The STORE/STORAGE command

In conjunction with the REDUCe or SPLIT command, this keyword causes the evaluated values of all independent error terms, or the linear combination of τ's arising from a dependent term, to be stored dynamically. Note that `REDUCtion with STORAGE` is equivalent to `FULL REDUCtion` (see Section A.6.19).

A.6.24 The BATCH command

Batch mode is entered when the keyword BATCH is read into the standard input of the front-end, and as such is not a true RK-AID command, but is for 'tagging' onto the beginning of a data-file that is intended for directing into the standard input from the operating system. The normal way to start a run is to give the front-end a blank line, which is fine for interactive use, but not so convenient for 'programming' a sequence of tasks. Thus, a run is started from batch mode by giving the command `GO` instead, and RK-AID exits automatically at the end of the file (with no need for the "><" symbol). Also note that commands entered directly to the standard input may

```
Batch Title: batch
==================
 n = 1
                    Total independent / Total :  (1 / 1)
 n = 2
                    Total independent / Total :  (2 / 2)
 n = 3
                    Total independent / Total :  (3 / 4)
 n = 4
                    Total independent / Total :  (6 / 8)
 n = 5
                   Total independent / Total :  (13 / 17)
 n = 6
                   Total independent / Total :  (29 / 37)

Batch Title: batch_2
====================
 n = 1
                    Total independent / Total :  (1 / 1)
 n = 2
                    Total independent / Total :  (2 / 2)
 n = 3
                    Total independent / Total :  (3 / 4)
 n = 4
                    Total independent / Total :  (4 / 8)
 n = 5
                    Total independent / Total :  (7 / 17)
 n = 6
                   Total independent / Total :  (14 / 37)
```

Figure A.7: RK-AID standard output from an example batch run.

be delimited by pipe symbols (`|`) for extra flexibility, and the batch-files may now contain blank lines for readability. Batch mode will only output 'principal' results. For example, the batch-file

```
 batch
```

```
order :6
Q(3:s, 1) = 0 | b(2) = 0 | go
R(0)'s = 0 | go
```

gives the standard output in Figure A.7. This shows that by applying the simplification

$$b_2 = 0 \text{ and } Q_{i1} = 0, \quad i = 3, \ldots, s,$$

to the set of explicit (by default) sixth-order RK order conditions, 8 equations are eliminated (as indicated by the final `Total independent / Total` line of the first run), and another 15 equations are eliminated by *also* making the condition

$$R_{0i} = 0, \quad i = 1, \ldots, s.$$

In this way, it is possible to experiment with different simplifying conditions applied from a batch-file of conditions. Note that batch mode, in the absence of a `TITLE` command, will name successive runs as 'batch', 'batch_2', 'batch_3' . . . , such that the ".tex" file does not already exist.

A.6.25 The `POINTS` command

Any form of continuous output by RK-AID is discretised by sampling at regular intervals over σ. This is set arbitrarily at 32 points (not including $\sigma = 0$), but may be changed with a command of the form `POINTS: #` or `# POINTS`, where `#` should be replaced by any positive integer.

A.6.26 The `NORM` command

The `NORM` command activates the calculation of truncation error norms via the `CHECK` command when no reduction, simplification or equation mode is being used. A variety of norms are also employed in the testing package for measurement of global error, and these are also selected by the `NORM` command. By default, an infinity norm[8]

[8]Also know as a *maximum* or *Chebychev* norm.

($\|\cdot\|_\infty$) is used, although one-norms ($\|\cdot\|_1$) or two-norms ($\|\cdot\|_2$) can be specified as an argument, i.e., `1 NORM` or `ONE NORM`.

- *Reduced 'one' and 'two' norms**. A *reduced* norm is a norm over a set of quantities taking into account known dependencies between them. It is defined as a normal one/two norm of the set, but is taken only over the *independent* set, with each independent term used in its *maximum* occurring multiple. As such, the reduced two-norm (`REDUCEd 2 NORM`) is intended as a 'smooth' alternative of the infinity norm which is ordinarily non-differentiable when taken over a parametric set of algebraic terms (as in the process of minimising a truncation error norm over a set of free parameters). The reduced one-norm was included as a natural generalisation.

Continuous norms

Provision is made for the following continuous measures $g(x)$ of *single* continuous quantities $f(x),\ a \le x \le b$ where, by default, $g(x) \equiv f(x)$.

1. *Power deviation**. Deviation of $f(x)$ from the simple polynomial x^k that passes through $f(b)$ at $x = b$, i.e.,

$$g(x) = f(x) - \left(\frac{x^k}{b^k}\right) f(b),\ a \le x \le b.$$

   The command takes the form

```
DEViation from ^#
```

   where `#` should be replaced by the value of k above.

2. *Linear deviation**. The $k = 1$ form of the power deviation above, included as a special case by `LINEar DEViation`.

3. *Box deviation**. The 'box deviation' (`BOX DEViation`) is implemented in the testing package as a specific gauge of continuous global error, and is defined as

$$g(x) = \left\{ \begin{array}{ll} f_{min} - f(x), & \text{where } f(x) < f_{min} \\ f(x) - f_{max}, & \text{where } f(x) > f_{max} \\ 0, & \text{otherwise} \end{array} \right\}, \; a \leq x \leq b, \qquad \text{(A.3)}$$

   where $f_{min} = \min\{f(a), f(b)\}$ and $f_{max} = \max\{f(a), f(b)\}$.

These continuous functions $g(x)$ may then be averaged to a single, discrete value by choice of one of the following, first defining g_n as a sample of $g(x)$ by

$$g_n = g(a + nh), \; n = 0, \ldots, m,$$

where $h = (b - a)/m$:

1. *Integration**. This is the default form of continuous average, which is done by trapezium rule on the discrete set, and is therefore defined by

$$I = \frac{h}{2}(g_0 + 2\sum_{n=1}^{m-1} g_n + g_m).$$

2. *Monotonicity**. The monotonicity of $\{g_0, g_1, \ldots, g_m\}$ may be gauged by

$$M = \sum_n |g_{n+1} - g_n|, \qquad \text{(A.4)}$$

   which is bounded below by $|g(b)|$, a value of $M = |g(b)|$ therefore indicating a perfectly monotonic behaviour over the set of discrete points, and the size of M above $|g(b)|$ representing its degree of monotonicity. Note that, except in the monotonic case, M is dependent on m, the number of samples taken and is thus an unreliable measure unless m is fixed[9]. For this function, and in fact for

[9]In fact, for a given *continuous* $g(x)$, M can grow without bound as m increases in a way likened to *fractal coastlines.*

every other form of discretisation in RK-AID, m is set arbitrarily at 32 points, though this may be changed with the `POINTS` command.

It should be remembered that these measures are of *single* quantities only. Double averages need separate consideration, and are described later.

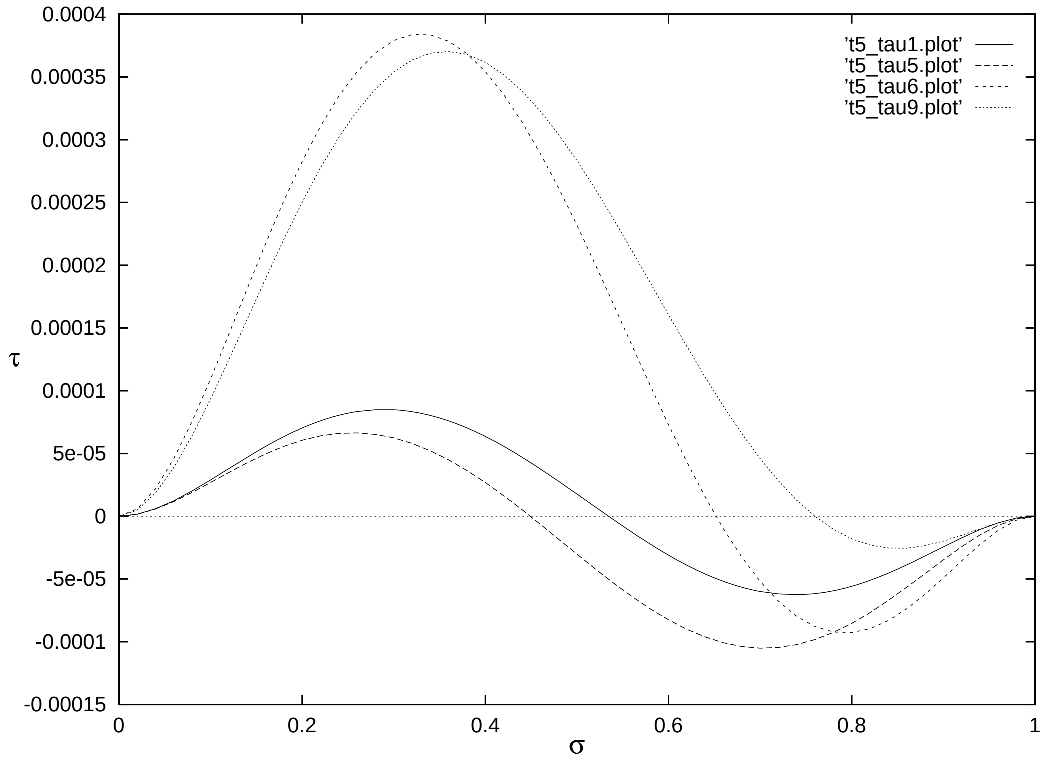


Figure A.8: Principal continuous truncation error coefficients for the RK5(4)7FMD$^4_7$.

- *The relative one-norm**. A relative one-norm over the set $\{{}^1x, {}^2x, \ldots, {}^nx\}$ is defined here as the quantity

$$\sum_{i=1}^{n} {}^ix/n,$$

as opposed to the normal *absolute* definition

$$\sum_{i=1}^{n} |{}^ix|/n.$$

This was included almost solely to accommodate the case where a single continuous truncation error term is required to be plotted in isolation, so as not to lose the sign of its behaviour over σ, and as such is probably not useful as any sort of averaging process. For example, the set of independent fifth-order continuous truncation error coefficients for the RK5(4)7FM with fourth-order dense (from Example A.3.2) may be generated by running the batch-file:

```
batch
check 547fmd4
relative 1 norm
[s] order 5, tau 1 | title tau1 | go
[s] order 5, tau 5 | title tau5 | go
[s] order 5, tau 6 | title tau6 | go
[s] order 5, tau 9 | title tau9 | go
```

This produces the individual ".plot" files shown in Figure A.8.

- *Scaling of continuous norms**. The `SCALE` command will scale the y-values of all continuous truncation error plots by the argument given, i.e., `2 NORM, SCALE 1E3` would increase the y-axis scale of all plots by three orders of magnitude. This was included in the event that software available to view the plot might only work on a limited range of y values.

Double averages

The averaging of a *set* of continuous terms in conjunction with the standard discrete norms presents a problem in that the *order* in which the average should be taken is not obvious. Control of this ordering is allowed with use of the `INSIDE` and `OUTside` commands.

- *A normed continuous average**. The `INSIDE` command causes an average over the continuous quantity to be done *first*, i.e., before the specified norm is taken. Thus, the commands

  ```
  linear deviation, inside
  2 norm
  ```

  will cause the linear deviation of all continuous quantities to be integrated (by default), and then a two-norm taken of those values.

- *The power deviation of a normed quantity**. The `OUTside` command as an argument applied to the `DEViation` command causes a power deviation to be taken on the averaged (normed) continuous quantities, as opposed to the other way round (which is default). Note that in the testing package, the `OUTside` command is ignored.

- *An alternative integrated two-norm**. The quantity

  $$\int_a^b \|\boldsymbol{g}(x)\|_2 \, dx, \tag{A.5}$$

  where $\boldsymbol{g}(x)$ is a vector of k continuous functions ${}^i g(x)$, $i = 1, \ldots, k$, is the integral of the two-norm of the continuous set. As a specific alternative,

  $$\sqrt{\int_a^b \sum_{i=1}^{k} \left[{}^i g(x)\right]^2 \, dx} \tag{A.6}$$

  is related to (A.5) (by application of the Cauchy-Schwartz inequality) but is more convenient to work with due to the 'smooth' integrand, hence it's inclusion here. Quantity (A.6) is used by including the keyword `OUTside` onto the two-norm command:

  ```
  2 norm, root outside
  ```

Another possible ambiguity is the use of an infinity norm on a continuous set. An `INSIDE` maximum norm is simply the continuous term with the largest integral (or M quantity in the case of the monotonicity gauge), the default average in this case is however *defined* to be the trapezoidal integral (or M value) of $\{g_0, g_1, \ldots, g_n\}$, formed from the k sampled quantities $\{{}^0 f, {}^1 f, \ldots, {}^k f\}$, where

$$g_n = \max_i \left\{{}^i f_n\right\}, \ i = 1, \ldots, k, \ n = 0, \ldots, m,$$

and is thus a *piecewise* maximum norm over the set.

A.6.27 The `TEST` command

The `TEST` command causes the current method to be run through the testing package, reporting statistics of the integration(s) to the standard output (no LaTeX output is done). Local extrapolation mode is assumed. The problem set is comprised of a, now standard, selection of DETEST problems (Hull et al. [13]), plus some others to include the RKN case or to test specific properties. The usual information is given regarding numbers of function evaluations, steps, and rejections (‘`F/S/R`’), plus discrete global error statistics (‘`GE`’) using a base 10 logarithm of the specified norm ($\|\cdot\|_\infty$ by default), taken over all components and steps (and the y and y' solution for a Nyström method). When a fixed number of points are available in each step through dense output, continuous global error statistics are also given using the specified averaging techniques (‘`D`’/‘`D'`’). Such norms are taken over the steps of the integration and problem components in a natural way, also being scaled by the discrete global error in each component for a higher degree of problem independence. Note that when dense output is being done, the discrete global error statistic relates to the error at the discrete points *and* in the interpolant. No exact or multi-precision calculations are allowed in these routines, all numerical integration being done in machine dependent ‘double-precision’. However, dense output using high-degree simple polynomials is vastly improved by the availability of b_i^* and $b_i'^*$ polynomial coefficients to a higher accuracy. As such, if the method parameters are read in after giving a `FLOAT` command, any extra accuracy will be used in the formation of the interpolating polynomials to great advantage. This is recommended for high accuracy work.

The routines here are used extensively in the main work, so no specific examples will be given. The step size control process is described in Sections 2.4.1 and 2.6.2.

- *Testing a specific method**. An argument may be given to the `TEST` command specifying a method file, i.e., `TEST method`, `READ`s the method file “*method*” before activating the testing routines.

A.6.28 The `PROBlem` command

The `PROBlem` command is used for selecting a problem with which to test the current method. Problems are referenced by number with a command of the form `PROBlem n`, n being chosen from the following set of 1–13 (10–13 only for RKN):

1. A class of first-order linear problems

$$y' = \lambda y, \; y(0) = 1.$$

2. DETEST problem A2

$$y' = -\frac{1}{2}y^3, \; y(0) = 1.$$

3. DETEST problem A3

$$y' = y\cos(x), \; y(0) = 1.$$

4. DETEST problem A4

$$y' = \frac{1}{4}(1 - \frac{y}{20})y, \; y(0) = 1.$$

5. Numerically unstable problems

$$y' = y + \cos(x) - \sin(x), \; y(0) = a.$$

6. DETEST problem E4

$$y'' = \frac{2}{5}\left[\frac{2}{25} - (y')^2\right], \; y(0) = 30, \; y'(0) = 0.$$

7. DETEST problem E5

$$y'' = \frac{\sqrt{(1 + y'^2)}}{25 - x}, \; y(0) = 0, \; y'(0) = 0.$$

8. First-order system

$$\boldsymbol{y}' = \begin{pmatrix} y_2 - x^2/5 \\ 2x/5 - y_1 \end{pmatrix}, \ \boldsymbol{y}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

9. ‘Runge’ problem (Baker, et al. [66])

$$y' = -2xy^2, \ y(0) = 1.$$

10. DETEST class D orbit problems

$$\boldsymbol{y}'' = -\boldsymbol{y}/r^3, \ \boldsymbol{y}(0) = \begin{pmatrix} 1-e \\ 0 \end{pmatrix}, \ \boldsymbol{y}'(0) = \begin{pmatrix} 0 \\ \sqrt{(1+e)/(1-e)} \end{pmatrix}$$

where

$$r = \sqrt{y_1^2 + y_2^2}.$$

11. ‘Fehlberg’ problem (Fehlberg [52])

$$\boldsymbol{y}'' = \begin{pmatrix} -4x^2 y_1 - 2y_2/r \\ 2y_1/r - 4x^2 y_2 \end{pmatrix}, \ \boldsymbol{y}\left(\sqrt{\frac{\pi}{2}}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ \boldsymbol{y}'\left(\sqrt{\frac{\pi}{2}}\right) = \begin{pmatrix} -\sqrt{2\pi} \\ 0 \end{pmatrix}$$

where

$$r = \sqrt{y_1^2 + y_2^2}.$$

12. Periodic second-order linear system

$$\boldsymbol{y}'' = -\boldsymbol{y}, \ \boldsymbol{y}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ \boldsymbol{y}'(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

13. Second-order linear problems

$$y'' = b^2 y, \ y(0) = 1, \ y'(0) = -b.$$

Problems $1, 5, 10$ and 13 above contain *classes* of problems, identified by the presence of an arbitrary parameter. This parameter is chosen by use of the keyword `PARAMeter` appended onto the `PROBlem` command[10] as in, for example, `PROBlem 10, PARAMeter 0.9` which corresponds to the DETEST orbit problem D5. Another way of specifying 'e' for problem 10 is with the `ECCENTricity` command, which is interchangeable with the `PARAMeter` command above. By default $a = 0$, $b = 1$, $e = 0.9$ and $\lambda = -1$.

A.6.29 Integration control commands

The following are example commands for specifying details of the integration(s) performed by the test package, with brief notes:

- `DISPlay every 10 steps`

  This is the interval for display of solution points — the default is for none.

- `TOLerance 1E-6`

  Defaults to $1.0{\times}10^{-3}$.

- `ABSolute error 1`

  Sets the component of absolute error used in the step size control process (default 1).

- `RELative error 0`

  Sets the component of relative error used in the step size control process (default 1). Note that in conjunction with the absolute error default, mixed error control is standard.

- `INITial/STARTing STEP 1E-4`

  By default, the initial step is chosen automatically in the case where a variable step formula is being used, or set to .5 for a fixed step implementation. An

[10]The `PARAMeter` command has another function, the meaning here being dependent on it appearing in the same line as the `PROBlem` command.

'automatic' choice of starting step is done simply by taking an initial step of *half* the integration range and not beginning the integration proper until a step is accepted, i.e., not counting rejections. The first step is then as large as can be allowed by the local error estimating procedure.

- `END of integration 20`

  Default integration end points for each problem are as follows:

Problem	1	2	3	4	5	6	7	8	9	10	11	12	13
x_{end}	20	20	20	20	20	20	20	20	5	20	10	20	5

- `error per UNIT STEP`

  Default is error per step (variable step implementation only).

- `CONSTant/FIXED STEP`

  The default is for variable step where possible, i.e., when an embedding is available.

- `5'th ORDER`

  This is for specifying the order of the error estimating (embedded) formula for the purposes of step size control. Note that the argument appearing *before* the `ORDER` command separates this case from that in Section A.6.4.

- `WATTS`

  This modifies the step size control with an addition due to Watts [70].

- `TUNE 10`

  In general, a command of the form `TUNE x` will scale the error estimate at each step by `x`, and is equivalent to modifying the lower-order external weights as follows:

$$b_i \rightarrow xb_i + (1-x)\widehat{b}_i$$

for an RK process, and also

$$b'_i \to x b'_i + (1-x)\widehat{b}'_i$$

for an RKN (assuming higher-order mode).

A.6.30 Multiple testing — the `NUMBer` and `LUMP` commands

The `PROBlem` command may, in fact, take a *list* of problem numbers (separated by any non-numeric characters), for example, `PROBlems 1, 2 & 3`, or `ALL PROBlems`, the latter selecting all relevant problems (i.e., only 10-13 for RKN). Where more than a single problem is tested and includes the 'orbit' problem 10, if no specific eccentricity is chosen (with the `ECCENTricity` command), a range of eccentricities will be selected automatically — $e = 0.1, 0.5$ and 0.9. Control over specifics of the integration is lost without the ability to reference particulars for each problem *separately.* In this case, *all* defaults apply unless otherwise given, in which case the same specification will apply to each problem.

Also, multiple values of tolerance (or step size, if constant) can be applied to each problem using the `NUMBer` or `NO.` command, which takes one of the forms:

```
NUMBer/NO. of TOLerances: 5
```

or

```
NUMBer/NO. of STEP sizes: 5
```

Tolerances decrease in powers of ten (step sizes in powers of two) from the value given on the `TOLerance` or `INITial STEP` command.

- *Decrease in half powers**. By inclusion of the keyword `HALF`, *half* powers of ten will be used for the decrease (half powers of two for step sizes).

- *Lumping**. The `LUMP` command can be used for averaging over either the tolerance (or step size) runs, *and/or* the set of problems specified, for example,

```
LUMP PROBlems 1, 2, 3, 4, 6, 7
STARTing TOLerance 1E-2
NUMBer of HALF TOLerances: 14
```

will report a *single* set of statistics representing the performance of a method on the DETEST problems A1, A2, A3, A4, E4 and E5 for fourteen half tolerances from $1{\times}10^{-2}$ to $1{\times}10^{-10}$.

For problem averages over a range of tolerances, as in the example above, results are output for each *problem* separately in individual files. The output files will be named by the current `TITLE`, and appended by "_#.eff" for # an integer. Such integers do not relate to the problem number itself, but increase consecutively from 1 each time. The `LUMP`'ed (averaged) results are sent to the standard output as expected, and to a file ending "_a.eff".

A.6.31 The `DENSE OUTput command`

If a dense formula is available, the command `DENSE OUTput` produces dense output at 32 regular, intermediate points throughout each step of the integration interval (see the `POINTS` command).

- *Specification of dense output range**. The `RANGE` and `AT` modifiers can be used to restrict the discrete steps in which interpolation is done by the dense formula. A range can be given, as in `DENSE RANGE 15 18`, whereby interpolation is done for any steps containing an x value in the closed interval $[15, 18]$. Also, if interpolation is required in one step only, a command of the form `DENSE OUTput AT 10` will activate dense output in the step containing $x = 10$. Again, 32 equally spaced intermediate points are done per step (unless otherwise specified).

- *Specification of a constant dense output interval**. A constant step for the dense output can be achieved throughout the range specified by the `DENSE RANGE` command using, for example, `DENSE STEP .1`, giving a dense point every .1, equally spaced over *all* the steps of the range. Dense output global error statistics are not calculated in this mode.

A.6.32 The `EXTRAPolation` command

This command is specific to the Nyström case and exploits the possibility of automatically generating a $\boldsymbol{y}^*$ interpolant one-order higher than the $\boldsymbol{y}'^*$ interpolant from the relation

$$\sigma^2 b_i^* = \int_0^1 \sigma b_i'^*, \; i = 1, \ldots, s^*.$$

Providing the $b_i'^*$ polynomial coefficients are available, this is done following an `EXTRAPolate` command, and also causes the $\boldsymbol{y}^*$ formula at $\sigma = 1$ to propagate the y solution whenever all extra function evaluations for the dense output have been done. In the light of new $p^* = q$ cases, this command is designed for use when the automatically generated $\boldsymbol{y}^*$ interpolant has a higher (local) order than the discrete $\widehat{\boldsymbol{y}}$ approximation and is therefore not continuous with it. In order that the resulting process is C^2 continuous, a new reusable stage $(s^* + 1)$ is added (the standard reusable stage which is, by necessity, present in all such formulae is lost, i.e., no longer reusable under this scheme), and the previous reusable stage modified — see Section 5.5 for details. By default, step size control is done with the standard $\boldsymbol{y}/\widehat{\boldsymbol{y}}$ and $\boldsymbol{y}'/\widehat{\boldsymbol{y}}'$ approximations, though the extrapolated y solution can be used instead by giving an `EXTRAPolated ERROR control` command. This causes s^* stages to be used by the discrete (with no FSAL), the reusable stage $s^* + 1$ only being needed when dense output is required.

Appendix B

Coefficient data

This chapter contains complete triples and additional coefficients for the continuous extension formulae contained in the main work. All data is given in various RK-AID formats which, where approximate, is specified to 40 (+) decimal figures. These data-files, and others for specification of all the methods from Chapters 6–8, are available via anonymous ftp at *ftp.tees.ac.uk* in the directory */pub/t.baker/files*. Where available in exact rational form, method coefficients are either given as RK-AID readable ".rat" files, or as MACSYMA batch files in directory */pub/t.baker/macsyma*.

B.1 Sixth-order RKN triples

Note that the discrete coefficients for the RKN6(4)6FD (with fifth-order dense) are presented in Example A.3.3, and so will not be repeated.

B.1.1 RKN6(4)6FD continuous extension FD_9^6L

```
T = sqrt(8581)

A[7,1] = -(251926886702T-406766121077443)/23751520386750000
A[7,2:3] = 0 ; 1/80
A[7,4] = -(344132127234932T-15831312622302763)/7402557187203750000
A[7,5] = (251926886702T-25704471264943)/2018879232873750
A[7,6] = -(251926886702T-28265721639943)/3721748208750000
A[8,1] = -(64121T-796692161)/31984875000
A[8,2:4] = 0 ; 0 ; -1/260
A[8,5] = (146747T-12857777)/4637806875
A[8,6] = -(3877T-549232)/227812500
A[8,7] = -(1500551T-4144866791)/118918125000
```

```
A[9,1] = -(676666627T-267674028232)/3063075750000
A[9,2:5] = 0 ; 0 ; 1/16 ; 0
A[9,6] = -(2681607271T-218788241311)/40841010000000
A[9,7] = (403183845701T+309223239559)/245046060000000
A[9,8] = -(2466376829T-382708639289)/1815156000000

BDS[1,0:5] = 1 ; -28091/5148 ; 6218/429 ; -917393/46332 ; &
  & 155371/11583 ; -12733/3564
BDS[2,0:5] = 0
BDS[3,1] = (96856T+389225579)/62581233
BDS[3,2] = -(1840264T+7395286001)/187743699
BDS[3,3] = (52399096T+210571038239)/2252924388
BDS[3,4] = -(64021816T+257278107719)/2816155485
BDS[3,5] = (677992T+2724579053)/86650938
BDS[4,0] = 0
BDS[4,1] = -(96856T-389225579)/62581233
BDS[4,2] = (1840264T-7395286001)/187743699
BDS[4,3] = -(52399096T-210571038239)/2252924388
BDS[4,4] = (64021816T-257278107719)/2816155485
BDS[4,5] = -(677992T-2724579053)/86650938
BDS[5,0:5] = 0 ; 40000/21879 ; -760000/65637 ; 5410000/196911 ; &
  & -5288000/196911 ; 140000/15147
BDS[6,0:5] = 0 ; 153/220 ; -207/44 ; 136/11 ; -3101/220 ; 637/110
BDS[7,0:5] = 0 ; 371293/25080 ; -2599051/45144 ; 1856465/22572 ; &
  & -11510083/225720 ; 2599051/225720
BDS[8,0:5] = 0 ; -18225/968 ; 99387/968 ; -24462/121 ; &
  & 166131/968 ; -51597/968
BDS[9,0:5] = 0 ; -50421/9196 ; 2201717/62073 ; -10806901/124146 ; &
  & 5563117/62073 ; -8084167/248292

BS[1,0:4] = 1/2 ; -916304408807/649884905397 ; &
  & 871256439355/433256603598 ; -1214417104049/866513207196 ; &
  & 246052739993/649884905397
BS[2,0:4] = 0
BS[3,0] = 0
BS[3,1] = (177986753200237567T+10586750993125874603)/6952239091937199060
BS[3,2] = -(295923315156387089T+15704550931400810551)/4634826061291466040
BS[3,3] = (130165782786913459T+7092397402949328506)/2317413030645733020
BS[3,4] = -(59187463560021089T+3439707222841672801)/3476119545968599530
BS[4,0] = 0
BS[4,1] = -(177986753200237567T-10586750993125874603)/6952239091937199060
BS[4,2] = (295923315156387089T-15704550931400810551)/4634826061291466040
BS[4,3] = -(130165782786913459T-7092397402949328506)/2317413030645733020
BS[4,4] = (59187463560021089T-3439707222841672801)/3476119545968599530
BS[5,0:4] = 0 ; 640442269000/11048043391749 ; &
  & -2080516450000/3682681130583 ; 3356010724600/3682681130583 ; &
  & -4343490163000/11048043391749
```

```
BS[6,0:4] = 0 ; -40080359288/101833817085 ; &
  & 194266042847/135778422780 ; -234999569681/135778422780 ; &
  & 141261008827/203667634170
BS[7,0:4] = 0 ; 1/5 ; -3/5 ; 3/5 ; -1/5
BS[8,0:4] = 0 ; -5/2 ; 15/2 ; -15/2 ; 5/2
BS[9,0:4] = 0 ; 1 ; -3 ; 3 ; -1
```

B.1.2 RKN6(4)6FM triples

The discrete coefficients for this method will be reproduced here from Dormand, et al. [15] for completeness.

Discrete coefficients

```
RKN646FM explicit RKN pair

[p] 4'th order

A[2,1] = 1/200
A[3,:] = -1/2200 ; 1/22
A[4,:] = 637/6600 ; -7/110 ; 7/33
A[5,:] = 225437/1968750 ; -30073/281250 ; 65569/281250 ; -9367/984375
A[6,:] = 151/2142 ; 5/116 ; 385/1368 ; 55/168 ; -6250/28101
BH[:] = 151/2142 ; 5/116 ; 385/1368 ; 55/168 ; -6250/28101 ; 0
BDH[:] = 151/2142 ; 25/522 ; 275/684 ; 275/252 ; -78125/112404 ; 1/12
B[:] = 1349/157500 ; 7873/50000 ; 192199/900000 ; 521683/2100000 ; &
  & -16/125 ; 0
BD[:] = 1349/157500 ; 7873/45000 ; 27457/90000 ; 521683/630000 ; -2/5 ; &
  & 1/12
```

FMD$_9^6$L continuous extension

```
A[7,1:2] = 25246896799/1803636513000 ; 10296975773/515324718000
A[7,3:4] = 8481183557/1030649436000 ; -312860602417/7214546052000
A[7,5:6] = 1/24 ; 93555/57258302
A[8,1] = 2842946529255253/44302873111646625
A[8,2] = 88318462621577/4219321248728250
A[8,3] = 1413081186447779/25315927492369500
A[8,4] = 5339147534539267/59070497482195500
A[8,5:7] = -1/21 ; -1994619121/803680237853 ; 3/19
A[9,1] = -90751072830192120623/61427030807976593917500
A[9,2] = 617935150143634404029/17550580230850455405000
A[9,3] = 461265044112972439229/3900128940188990090000
A[9,4] = 22144951532391526058159/245708123231906375670000
```

```
A[9,5:8] = -1/20 ; 14470664919179433/1560051576075596036 ; -5/36 ; -1/23

BDS[1,0:5] = 1 ; -19696739/3170160 ; 11741363/634032 ; -737419/26418 ; &
  & 65137151/3170160 ; -52111/8880
BDS[2,0:5] = 0 ; 8400/13949 ; -543500/125541 ; 317875/27898 ; &
  & -166300/13949 ; 540175/125541
BDS[3,0:5] = 0 ; 46200/9139 ; -2989250/82251 ; 3496625/36556 ; &
  & -914650/9139 ; 5941925/164502
BDS[4,0:5] = 0 ; 6600/481 ; -2989250/30303 ; 3496625/13468 ; &
  & -914650/3367 ; 5941925/60606
BDS[5,0:5] = 0 ; -39375000/4505527 ; 849218750/13516581 ; &
  & -2980078125/18022108 ; 779531250/4505527 ; -99296875/1590186
BDS[6,0:5] = 0 ; 16408/26455 ; -7249226/1571427 ; 13480039/1047618 ; &
  & -77922311/5238090 ; 191243557/31428540
BDS[7,0:5] = 0 ; -12825859648/735951645 ; 174294271288/1619093619 ; &
  & -502241195993/2158791492 ; 3433465450279/16190936190 ; &
  & -2254860561911/32381872380
BDS[8,0:5] = 0 ; -1492269707/345336355 ; 235240487903/7459265268 ; &
  & -105848919493/1243210878 ; 1155662788153/12432108780 ; &
  & -326639522707/9324081585
BDS[9,0:5] = 0 ; 5676953849/339893840 ; -1546564427149/20189694096 ; &
  & 110642671955/841237254 ; -3366665344859/33649490160 ; &
  & 2869642242569/100948470480

BS[1,0:1] = 1/2 ; -85487371803667153/43981479189572274
BS[1,2] = 17280262450172623/4886831021063586
BS[1,3] = -3024838689681329/1047178075942197
BS[1,4] = 19058981772435062/21990739594786137
BS[2,0:1] = 0 ; 199374422154175/198484733348521
BS[2,2] = -18623609603922575/7145450400546756
BS[2,3] = 956695992676835/396969466697042
BS[2,4] = -2733201955031525/3572725200273378
BS[3,0:1] = 0 ; 1423166113190525/260083443698062
BS[3,2] = -132997929245980975/9363003973130232
BS[3,3] = 20723314715215915/1560500662188372
BS[3,4] = -6656813862907175/1560500662188372
BS[4,0:1] = 0 ; -116738947820275/95820216099286
BS[4,2] = 10725623548353725/3449527779574296
BS[4,3] = -46285303382065/27377204599796
BS[4,4] = 219118244528975/1724763889787148
BS[5,0:1] = 0 ; 33872467133296875/128221137743144566
BS[5,2] = -502762202931390625/769326826458867396
BS[5,3] = -32038049229715625/256442275486289132
BS[5,4] = 112267009536015625/384663413229433698
BS[6,0:2] = 0 ; -210773985245/944041537924 ; 1066488125843/1416062306886
BS[6,3:4] = -217083085054/236010384481 ; 1104342724697/2832124613772
BS[7,0:1] = 0 ; -5067347446941263/739184524194492
```

```
BS[7,2] = 5067347446941263/246394841398164
BS[7,3] = -5067347446941263/246394841398164
BS[7,4] = 5067347446941263/739184524194492
BS[8,0:4] = 0 ; 1 ; -3 ; 3 ; -1
BS[9,0:4] = 0 ; 5/2 ; -15/2 ; 15/2 ; -5/2
```

FMD$^5_7$M continuous extension

```
A[7,1:3] = 337579/512400000 ; 623027/87840000 ; 244943/58560000
A[7,4:6] = -18556759/1229760000 ; 1/61 ; 51/400000

BDS[1,0:4] = 1 ; -331/42 ; 23930/1071 ; -8815/357 ; 3320/357
BDS[2,0:4] = 0 ; -25/66 ; 9025/2871 ; -4700/957 ; 700/319
BDS[3,0:4] = 0 ; -25/12 ; 5875/342 ; -2975/114 ; 650/57
BDS[4,0:4] = 0 ; 25/84 ; -425/126 ; 475/42 ; -50/7
BDS[5,0:4] = 0 ; 0 ; 15625/28101 ; -171875/37468 ; 31250/9367
BDS[6,0:4] = 0 ; -1/18 ; 5/9 ; -55/36 ; 10/9
BDS[7,0:4] = 0 ; 1000/99 ; -4000/99 ; 5000/99 ; -2000/99

BS[1,0:4] = 1/2 ; -331/126 ; 11965/2142 ; -1763/357 ; 1660/1071
BS[2,0:4] = 0 ; -25/198 ; 9025/11484 ; -940/957 ; 350/957
BS[3,0:4] = 0 ; -25/36 ; 5875/1368 ; -595/114 ; 325/171
BS[4,0:4] = 0 ; 25/252 ; -425/504 ; 95/42 ; -25/21
BS[5,0:4] = 0 ; 0 ; 15625/112404 ; -34375/37468 ; 15625/28101
BS[6,0:4] = 0 ; -1/54 ; 5/36 ; -11/36 ; 5/27
BS[7,0:4] = 0 ; 1000/297 ; -1000/99 ; 1000/99 ; -1000/297
```

B.1.3 The RKN6(4)6FU (RKN$^{6(4)}_{6f}$D$^{6(5)}_{8(6)}$L) triple

```
RKN646FU6 explicit RKN triple (approximate assumed form)

[p] 4'th order

----------------------------cut----------------------------

6 2 5

1d0 !A[2,1]
     6.91013053794178190471712172715299068863156391225 66d-2
1d0 !A[3,1:2]
     9.213507383922375872956162302870654251508751883008 8d-2
     1.842701476784475174591232460574130850301750376601 8d-1
1d0 !A[4,1:3]
     1.273966453085563083668012565335201575517259519625 9d-1
     2.911923321338429905526885863623317886896593187716 4d-1
     7.279808303346074763817214659058294717241482969290 9d-2
```

```
1d0 !A[5,1:4]
     2.5590432572787905642481791062373652249519796184855d-2
     2.0388105330247167398611534777091027526164926372327d-2
    -7.9902714514959494333381644125787531191854828836062d-3
     2.174558988572632070812387268738297956585669948297d-3
1d0 !A[6,1:5]
     8.5425953604808178656948433994338908170492012046204d-2
     0.0d0
     1.109075950311341191039384154492822721842074908550d-1
    -2.625910339157872786916129323919763826627811887252d-3
     3.0629236170321557502602927988029858347192830898605d-1
1d0 !BH[1:6]
     8.5425953604808178656948433994338908170492012046205d-2
     0.0d0
     1.1090759503113411910393841544928227218420749085501d-1
    -2.6259103391578727869161293239197638266278118872546d-3
     3.0629236170321557502602927988029858347192830897654d-1
     0.0d0
1d0 !BDH[1:6]
     8.5425953604808178656948433994338908170492012046205d-2
     0.0d0
     4.3240890990835784966063957717020395755664318413651d-1
    -3.0356103221713545353911298343432656695541919289235d-1
     4.2743507004773074167232410612174141662297303210261d-1
     3.5829109865623868354920086614804228460531096459291d-1
1d0 !B[1:6]
     1.2900583151222484794569108676713400549738641130959d-1
     2.9366574346507173755830398174721508236523823812194d-1
     7.7328425022703414496004931485650912137375350568476d-2
     0.0d0
     0.0d0
     0.0d0
1d0 !BD[1:6]
     8.2699108573693942138685700692459092947519937201279d-2
     0.0d0
     4.0621297250264061673473780063777665088494981419136d-1
     7.5041503796352617795125922233979500317036468850303d-2
     4.3604641512731282333145057643578475585049377975706d-1
     0.0d0
1d0 !A[7,1:6]
     7.8360037337749128915192712150132048691971556918303d-2
     0.0d0
     8.2006913737774670203310771519037258626434843927033d-2
    -1.6212614133028434711830702963635110374579168677631d-1
     2.5887977931406743742292254024005438897189561005761d-1
     1.4787941094069311057688100572712740745548967587337d-1
1d0 !A[8,1:7]
```

```
     3.4093898355119702498426361359298291834519170025691d-2
     0.0d0
    -2.7051311557765458520707179125627416210838607785129d-2
     2.1479256700501033619425765816085199064601017498496d-2
     5.4964114425833128478486187416418892526647255693343d-2
    -2.9542521633605409639204154928305953086581941853202d-2
     2.6056563709917003563573019462130985871653106420801d-2
1d0 !BS[1]
     5.0d-1
    -1.3555402427430100409398372525137925518572178794666d0
     1.9208802642771119093889960974847667372765737588619d0
    -1.3480100323011528026737382674564612781290338190929d0
     3.6809596437185911288152785647982600088016995174385d-1
     0.0d0
1d0 !BS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[3]
     0.0d0
     1.4502051108423062500460731239781973327308481066754d-2
    -5.4193677740004664924550384783113162500202900105762d-1
     1.2734040617532709759526164174684070055707343199085d0
    -6.3506174043051327010363488542777508171180630906264d-1
     0.0d0
1d0 !BS[4]
     0.0d0
     5.9831826559504663346995472525023318192239828754834d-1
    -3.1653829078683072451804119843786316838951607649332d-1
    -8.7407264502929023455616557138416522499251038808979d-1
     5.8966675988191645281733591524787544763300036514752d-1
     0.0d0
1d0 !BS[5]
     0.0d0
     4.1011320066944982745768464758905308097702670082659d0
    -9.8461859447739149067017207600758207603467415505864d0
     8.7992949696358976981567536656400991755912788981755d0
    -2.7479486698532654910058501015745106415428760468689d0
     0.0d0
1d0 !BS[6]
     0.0d0
    -9.1688614267809468881944243395933339307705102472215d-1
     1.4592029347530906487123229711377887562045982511987d0
    -5.2603854012813591451551950654561961778335439282454d-1
    -1.6278251946860045377361030632835745344192833652032d-2
     0.0d0
1d0 !BS[7]
     0.0d0
     5.8780861869729505510646403762033857176844975138970-1
    -1.7634258560918851653193921128610157153053492541691d0
     1.7634258560918851653193921128610157153053492541691d0
    -5.8780861869729505510646403762033857176844975138970-1
     0.0d0
1d0 !BS[8]
     0.0d0
    -3.0293345566741582958944462835277585918541546240819d0
     9.0880036700224748876833388505832757755624638722458d0
    -9.0880036700224748876833388505832757755624638722458d0
     3.0293345566741582958944462835277585918541546240819d0
     0.0d0
1d0 !BDS[1]
     1.0d0
    -4.7336993914610234019213530260037656087983344369062d0
     1.1157825760226233729802558514597733925086820587777d1
    -1.3859509850985401600078326575808020801763292450585d1
     8.6928957087650446903760803158039357124824129577411d0
    -2.1720862729400452395220107945955443188371146459808d0
1d0 !BDS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BDS[3]
     0.0d0
     1.0268073253583066757476069889042326364615748806971d1
    -5.5852877123879518480601032622467692858894000830163d1
     1.1679699479258827501658909461675174220015148512284d2
    -1.0451319776842011452331553345924848753674410387826d2
     3.3733415756036649079512041153092315788427513962747d1
1d0 !BDS[4]
     0.0d0
    -7.8833947487309775928469100312406746810702895658679d0
     4.0834937233245365378336473771369433349292730843098d1
    -8.2594159020938708013769719569583470443961858644236d1
     7.2395719143762417542682980049415380162594167865298d1
    -2.3056663639555232767941937203394994953810169691184d1
1d0 !BDS[5]
     0.0d0
     1.9790202221193374842579599944497419624746224720134 7d1
    -7.84184044425363781291754824527402988888928015748 45d1
```

Alternative b^*/b'^* coefficients for the embedded $\mathrm{FUD}_6^5\mathrm{L}$:

```
     0.0d0
     2.3342370338036895763160287999824182171226236347725d1
    -6.8548694508112378113589041022159710136301030257022d1
     6.7358083572296257154514834252337681742919003792439d1
    -2.2154385312559932676872997359326073541670837695029d1
1d0 !BS[5]
     0.0d0
     1.8570866095485287305305041303671974063262375492421d0
    -3.1140497533360062745626937235058205500146531735148d0
     2.0671587781979890660177266290700989652591905211039d0
    -5.0390327270729594695950775605117723809884658784507d-1
1d0 !BS[6]
     0.0d0
    -2.1899001383651217931411562060393731161450755750725d1
     6.4405548657672460376488681850440982061325712429208d1
    -6.3472384263047505642291878385848812922904468570834d1
     2.0965836989026263197214758595801562023029511892351d1
1d0 !BDS[1]
     1.0d0
    -3.4410267404135909874594327760150781972001200154062d0
     5.1811451058466910961156684638338691156201605434729d0
    -3.6120802224285683365682964296508090987875009804963d0
     9.5738781060027640656900917582635708853795246447574d-1
1d0 !BDS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BDS[3]
     0.0d0
    -6.4603399513890286993782574333778670511050344233178d0
     2.3847637309257004950536543117064325384339723461842d1
    -2.6152210214805134554635116048144029827581127733048d1
     9.1973217668455161531374699416277754519030818786604d0
1d0 !BDS[4]
     0.0d0
     7.0027111014110687289480863999472546513678709043173d1
    -2.7419477803244951245435616408863884054520412102809d2
     3.3679041786148128577257417126168840871459501896219d2
    -1.3292631187535959606123798415595644125002502617017d2
1d0 !BDS[5]
     0.0d0
     5.5712598286455861915915123911015922189787126477261d0
    -1.2456199013344025098250774894023282200058612694059d1
     1.0335793890989945330088633145350494826295952605519d1
    -3.0234196362437756817570465363070634285930795270704d0
```

```
1d0 !BDS[6]
     0.0d0
    -6.56970041509536537942346861811811934843522672521766d1
     2.57622194630689841505954727401763928245302849716 83d2
```

B.1.4 The RKN6(5)8F8 $\mathrm{RKN}_{8f}^{6(5)}\mathrm{D}_8^6$ triple

```
RKN658F8 explicit RKN triple

[p] 5'th order

A[2,1] = 328696969041/258794900180000
A[3,1] = 80093569429/15662221923528600
A[3,2] = 135556287221783/15662221923528600
A[4,1:2] = 128020206459/46322635116800 ; 340568676167361/34224934141963712
A[4,3] = 3747970172582631/993919437772998400
A[5,1:2] = 3206715476/977118084495 ; 227045784111574/14438644091140941
A[5,3:4] = 46271236698551/7764995607601550 ; 1058/243675
A[6,1] = -1023973641590441409/13442696156629128448
A[6,2] = 57523796106726869443125/29485559544516906009677 2
A[6,3] = 766760770966250025/10371435218631824944
A[6,4:5] = -6533183344905/22293176333888 ; 20444989542705/89172705335552
A[7,1] = 1919653678482450810691270558 3/10706836738655702195987464879 2
A[7,2] = -2095425711403361822361028164163937 5/75150893782062280573078974 4 &
  & 00304316
A[7,3] = -1477824327381408531738645575/1398626753012512270783121588 8
A[7,4] = 1015256501254417946602 0/126829042441595232045 93
A[7,5:6] = -30888787801944535468 65/75157951076500878249 44 ; 23/188
A[8,1:4] = 58159/856980 ; 0 ; 0 ; 14435451400/173705461119
A[8,5:6] = 14004534725/84843419544 ; 54339975640/460514503683
A[8,7] = 906975972631/13748267625480

C[2:8] = 573321/11375300 ; 127/965 ; 69/380 ; 23/95 ; 63/124 ; 65/83 ; 1

BDS[1,0:3] = 1 ; -1307557/188370 ; 451140827/19496295 ; -334855733/8665020
BDS[1,4:5] = 67246016/2166255 ; -37154120/3899259
BDS[2:3] = 0
BDS[4,0:1] = 0 ; 2371842200000/86992590029
BDS[4,2] = -8224370647760000/54022398408009
BDS[4,3] = 1906025424020000/6002488712001
BDS[4,4] = -1721565431632000/6002488712001
BDS[4,5] = 5096802181600000/54022398408009
BDS[5,0:2] = 0 ; -2850771875/126114336 ; 18703751370625/127265129316
BDS[5,3:4] = -670583728680625/2036242069056 ; 9846289124125/31816282329
BDS[5,5] = -19909383521875/190897693974
```

```
BDS[6,0:1] = 0 ; 4846342681280/1680681137373
BDS[6,2] = -4503895893209728/196639693072641
BDS[6,3] = 1448064289845664/21848854785849
BDS[6,4:5] = -85020791340928/1149939725571 ; 288948295212800/10349457530139
BDS[7,0:1] = 0 ; -14586267501029/19858608792360
BDS[7,2] = 827430938427937/134045609348430
BDS[7,3] = -2100022245115679/1072364874787440
BDS[7,4:5] = 89636808872108/3527516035485 ; -23200949387270/2116509621291
BDS[8,0:3] = 0 ; 240695/1821216 ; -6976889/6146604 ; 367418159/98345664
BDS[8,4:5] = -7964363/1536651 ; 23221325/9219906

BH[1:4] = 58159/856980 ; 0 ; 0 ; 14435451400/173705461119
BH[5:6] = 14004534725/84843419544 ; 54339975640/460514503683
BH[7:8] = 906975972631/13748267625480 ; 0
BDH[1:4] = 58159/856980 ; 0 ; 0 ; 5485471532000/54022398408009
BDH[5:6] = 1330430798875/6108726207168 ; 6738156979360/28091384724663
BDH[7:8] = 75279005728373/247468817258640 ; 20281099/295036992

B[1] = 632102589256978140242270793 49/196352672766502187831303352105
B[2] = -96625216897797773280570574658950 0/290145725824430267763877 96 &
    & 43262939
B[3] = -32369398882367252182010400 5/2564941521572334654775343647
B[4] = 32885341625233613342 2/1958667567230857206213
B[5] = 1445434472739758701 51/4352594593846349347 14
B[6:8] = 34/465 ; 16/285 ; 13/1613

BD[1:4] = 13956946/187464375 ; 0 ; 0 ; 6310444616/173705461119
BD[5:6] = 8198130583/28281139848 ; 88877836162504/402950190722625
BD[7:8] = 89298668334983/286422242197500 ; 67/1000

BS[1,0:2] = 1/2 ; -33873443/19496295 ; 112725233/38992590
BS[1,3:4] = -57975877/25995060 ; 12567704/19496295
BS[2:3] = 0
BS[4,0:1] = 0 ; 182665499654000/54022398408009
BS[4,2] = -508082475841000/54022398408009
BS[4,3:4] = 163206511166800/18007466136003 ; -513547048000/173705461119
BS[5,0:2] = 0 ; -1872583306375/1527181551792 ; 30943743185125/6108726207168
BS[5,3:4] = -11565652830475/2036242069056 ; 21270616375/10605427443
BS[6,0:1] = 0 ; -143084093349872/196639693072641
BS[6,2] = 541464329746216/196639693072641
BS[6,3:4] = -187208153569552/65546564357547 ; 3056514654272/3223601525781
BS[7,0:1] = 0 ; 168193760831951/402136828045290
BS[7,2] = -5746299968376259/3217094624362320
BS[7,3] = 2583572289296923/1072364874787440
BS[7,4] = -21789823670656/22340934891405
BS[8,0:2] = 0 ; -7987891/73759248 ; 141967693/295036992
BS[8,3:4] = -69453931/98345664 ; 1/3
```

B.2 Eighth-order RK triples

B.2.1 The RK8(7)13M with $87D^{8(7)}_{20(17)}L_2$ extension

```
RK8713MD8L2 explicit RK triple
                         (approximate assumed form)

[p] 7'th order

----------------------cut-----------------------

13 7 7

1d0 !A[2,1]
     5.5555555555555555555555555555555555555555d-2
1d0 !A[3,1:2]
     2.0833333333333333333333333333333333333333d-2
     6.25d-2
1d0 !A[4,1:3]
     3.125d-2
     0.0d0
     9.375d-2
1d0 !A[5,1:4]
     3.125d-1
     0.0d0
    -1.171875d0
     1.171875d0
1d0 !A[6,1:5]
     3.75d-2
     0.0d0
     0.0d0
     1.875d-1
     1.5d-1
1d0 !A[7,1:6]
     4.7910137111111111111111111111111111111111d-2
     0.0d0
     0.0d0
     1.1224871277777777777777777777777777777778d-1
    -2.5505673777777777777777777777777777777778d-2
     1.2846823888888888888888888888888888888889d-2
1d0 !A[8,1:7]
     1.6917989787292281181431107136038236065524d-2
     0.0d0
     0.0d0
     3.8784827848604316952654574415937335337074d-1
     3.5977369851500327896700889634772368008164d-2
     1.9697021421566606015671525607214988812822d-1
    -1.7271385234050183876139299700233384557264d-1
1d0 !A[9,1:8]
     6.9095753359192300648564548984547678561048d-2
     0.0d0
     0.0d0
    -6.3424797672885415188280787497173854654264d-1
    -1.6119757522460408036687692398181712344224d-1
     1.3865030945882525541986695013301580192774d-1
     9.4092861403575626972423968413025834347124d-1
     2.1163632648194398185537211713190210476354d-1
1d0 !A[10,1:9]
     1.8355699683904538548980602353688030844984d-1
     0.0d0
     0.0d0
    -2.4687680843155924527443157599741074577784d0
    -2.9128688781630045638800257280395198005434d-1
    -2.6473020233117375688439799465946143259634d-2
     2.8478387641928004491645182542167737702324d0
     2.8138733146984979253940364182671178209814d-1
     1.2374489986331465762703021266363972031224d-1
1d0 !A[11,1:10]
    -1.2154248173958880591605105250296629948884d0
     0.0d0
     0.0d0
     1.6672608665945772432280413288564107748594d1
     9.1574182841681796059571865045074263315944d-1
    -6.0566058043574709475545055430916340040824d0
    -1.6003573594156178111841706410078823030684d1
     1.4849303086297662557545391898026632082724d1
    -1.3371575735289849318293041396181595790894d1
     5.1341826481796379331732536116586028987134d0
1d0 !A[12,11]
     2.5886091643826428381573093223175776672964d-1
     0.0d0
     0.0d0
    -4.7744857854892051123101175097060427468294d0
    -4.3509301377703250944070041181031778193244d-1
    -3.0494833320722415095605128663120316139834d0
     5.5779200399360991174236766344649418586244d0
     6.1558315898610400973386891266889544811984d0
    -5.0621045867369383700774064339103916449904d0
     2.1939261731806790612749142904658060197884d0
     1.3462799865933494153572623788732366139564d-1
1d0 !A[13,12]
     8.2242759962650747796316820477266659095724d-1
     0.0d0
     0.0d0
    -1.1658673257277664283976553035458414775474d1
    -7.5762211669093619588111615408824496536644d-1
     7.1397358815958152797826928276505467531424d-1
     1.2075774986890056739566170448600679670964d1
    -2.1276591139204026563908208589693986354284d0
     1.9901662070489554183280716983443141521764d0
    -2.3428647154404029266029469185680153145124d-1
     1.7589857770794226507310510589014481831464d-1
     0.0d0
1d0 !BH[1:13]
     4.1747491141530246222085928468507115134194d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
    -5.5452328611239308961521894654716718893594d-2
     2.3931280720118009704674735424875696966034d-1
     7.0351066940344302305804641088970215136644d-1
    -7.5975961381446092988448767708505840765544d-1
     6.6056303092228634146137859483782063994044d-1
     1.5818748251012333552961483860068544397294d-1
    -2.3810953875286280447186355530569719352514d-1
     2.5d-1
1d0 !B[1:13]
     2.9553213676353496981964883112032246577334d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
    -8.2860627648779703976680561268871918473544d-1
     3.1124090005111832792991375162685705128974d-1
     2.4673451905998869819646857040687614585624d0
    -2.5469416518419087391273800754157089617814d0
     1.4435485836767752403018749506901042685114d0
     7.9415595881127287271301954162228677131474d-2
     4.4444444444444444444444444444444444444444d-2
     0.0d0
1d0 !A[14,1:13]
     4.1747491141530246222085928468507115134194d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
    -5.5452328611239308961521894654716718893594d-2
     2.3931280720118009704674735424875696966034d-1
     7.0351066940344302305804641088970215136644d-1
     -7.5975961381446092988448767708505840765554d-1
      6.6056303092228634146137859483782063994 04d-1
     0.0d0
1d0 !BS[6]
     0.0d0
     6.921868564605534596908047221008052296999d0
    -2.240648476389454707562905684311883126568d1
    -4.276599473961013701500968972044340466849d1
     3.131894864954899512740218160298218159249d2
    -5.657486959709070376193554236006099085285d2
     4.330505052167343828016889837258360967515d2
    -1.222961371310293862715861987071485372296d2
1d0 !BS[7]
     0.0d0
     3.036670372838668430239767655492398887752d1
    -2.279752229315990269237915254114961123007d2
     7.577516321217108556772813824587613302773d2
    -1.368974096091228904714540774770152370761d3
     1.394740091318518028528699624824422334132d3
    -7.539740281049082960419569086115061930927d2
     1.683042327663218392689572723092957798379d2
1d0 !BS[8]
     0.0d0
     2.419137364143642543118511074543486588196d1
    -3.727875194933697047794033780265121485469d2
     1.907770607922395229387024781782902321739d3
    -4.599719951983438432835996536403426918779d3
     5.744806768889312471912252136424882267701d3
    -3.597118672544433047370243193824023901369d3
     8.935609042375005012782391257116332155238d2
1d0 !BS[9]
     0.0d0
    -2.445115441796236123245649013661202438481d1
     3.677438830420754542932941374966117158027d2
    -1.930965373657927822575268144629653223811d3
     4.837951988982097419506005441498885678263d3
    -6.277065855263968391552314033379312682426d3
     4.066046672600237023376876672700514325343d3
    -1.040019920898365782746022071227518847194d3
1d0 !BS[10]
     0.0d0
     7.286977850457439417265042170970045168102d0
    -1.174667130779988514218375978420566536091d2
     6.285610656350245732200197753832287414868d2
    -1.602521824642011284210765194830960348382d3
     2.11847712895339129261815327505746129645d3
    -1.394736843986219114146801302211790951545d3
     3.610607722982782308654273808679856910713d2
1d0 !BS[11]
     0.0d0
    -1.938165845101156177286737414512839308397d0
     2.354263929620951269790599803782712460402d1
    -1.174348775544638824938711359367580786939d2
     3.00316632613417835239942654827041563 6d2
    -4.143088717924980819906208816119495646571d2
     2.925886544122385484953256954965536496719d2
    -8.260782364729265243586597855960116977253d1
1d0 !BS[12]
     0.0d0
    -1.237792228326919516727132040692673370309d0
     9.82022965914896851971009481710705367586d0
    -3.275938483069280702702778063130084960944d1
     4.883891247215796997972011797647476956963d1
    -2.084652647508862349802035673335805848694d1
    -1.736541637933166276897110822132959900291d1
     1.331186824338021150684430127779366003058d1
1d0 !BS[13]
     0.0d0
    -9.349931366753203306658157598335828004 98d-2
     6.894004452455872800501995967192988370465d0
    -5.033471430022876238565175096638955036479d1
     1.581260245441192767347718146968041887677d2
    -2.526338733539150291803543657013042254638d2
     2.003195038961131058948898007976892939117d2
    -6.202744592487693183109091321800933694123d1
1d0 !BS[14]
     0.0d0
     2.07934856013526078278397877219994047161d0
    -2.583323955683169586103072153746574253462d1
     1.288664759956064223189285000799014451106d2
    -3.251283570646629010235372834891383641557d2
     4.386507959678804415125064335296169934671d2
    -3.016923183008509546216652818572956050661d2
     8.305729439872342689201437450218133270708d1
1d0 !BS[15]
     0.0d0
    -3.613973332443397545222670855755811265415d1
     3.21114523022708179937237561607355644445d2
    -1.170221726770297695872224480806944834824d3
     2.235164810319892805335938584174506322515d3
    -2.360905586393680070046786802301087811786d3
     1.308469433741935460601828218615006449021d3
    -2.974817205961247045037663727312776567171d2
1d0 !BS[16]
     0.0d0
    -5.631008699065792331344287024182554978272d0
     4.528559660367774090417568186365907156083d1
    -1.36890228834664309617230375863646892457d2
     1.971970110788099779843504380946080107166d2
    -1.342703871002797703364209962321033918066d2
     3.186872547879289465589949446177029335677d1
     2.440291472729258740570044699895463607633d0
1d0 !BS[17]
     0.0d0
     4.099489719571383121511755584098471476481d0
    -1.509594477122158777160470853459457736729d1
     2.417434164637991788489327596724296203027d1
    -4.281112448221642784091596333725701585559d1
     8.163904631466787093856989061585013007471d1
    -8.065930022952648976501198592051900272617d1
     2.865349180234533343255773562517903236759d1
1d0 !BS[18]
     0.0d0
    -1.703990470879788876882143373224549403006d1
     1.792237173015007942184007043199830731132d2
    -8.278549848574775816839403155871342291503d2
     1.981251897145862416807952188840490920743d3
    -2.549778466194613989762231853926770338072d3
     1.673342022126834417357293484474162429132d3
    -4.39144280813308168168652774388486361735d2
1d0 !BS[19]
     0.0d0
    -6.86389975704773943555252934389741122745d0
     7.207325609099060757881124598581944300743d1
    -3.292086347569929005106154260771879988093d2
     7.820462843960555740372784743409962047543d2
    -1.003369465822644459236294234853659764271d3
     6.582517355724271971224736971273701618443d2
    -1.729292757227882795561012271794406352981d2
1d0 !BS[20]
     0.0d0
     2.765843642637547071510858792543972773624d1
    -2.876591817242628875670761377987233397136d2
     1.309011836742197639914724756593997307199d3
    -3.104522021416613412955655243000619990107d3
     3.980478392700170080140839579812313024799d3
    -2.611092778056228867528883295557776975921d3
     6.861253153283619772809417520253702460079d2
```

Alternative b^* coefficients for the $87D^7_{17}L_2$:

```
1d0 !BS[1]
     1.0d0
    -8.4848028011449550688022446477644498246 26d0
```

B.2.2 The RK8(6)12M with $86D^{8(7)}_{19(16)}L_2$ extension

```
RK8612MD8L2 explicit RK triple
                      (approximate assumed form)

[p] 6'th order

----------------------cut----------------------

12 7 7

1d0 !A[2,1]
     5.260015195876773187855875444880160900791d-2
1d0 !A[3,1:2]
     1.972505698453789945445953291830060337797d-2
     5.917517095361369836337859875490181013339d-2
1d0 !A[4,1:3]
     2.958758547680684918168929937745090506695d-2
     0.0d0
     8.876275643042054754506789813235271520085d-2
1d0 !A[5,1:4]
     2.413651341592666855023697986645101263135d-1
     0.0d0
    -8.845494793282860853448649627170606042173d-1
     9.24834003261792003115737966542746857636d-1
1d0 !A[6,1:5]
     3.703703703703703703703703703703703703704d-2
     0.0d0
     0.0d0
     1.708286087294738712796044821732026980738d-1
     1.254676875668224250166918141230935982225d-1
1d0 !A[7,1:6]
     3.7109375d-2
     0.0d0
     0.0d0
     1.702522110195440393149780602717143574367d-1
     6.021653898045596068502193972828564256326d-2
    -1.7578125d-2
1d0 !A[8,1:7]
     3.709200011850479271087793198363556544293d-2
     0.0d0
     0.0d0
     1.703839257122399938102140547044867110858d-1
     1.072620304463732846518091991677920552794d-1
    -1.531943774862440175279361582362177579432d-2
     8.273789163814022887584737660015136293978d-3
1d0 !A[9,1:8]
     6.24110958716075717114429577812045746175d-1
     0.0d0
     0.0d0
    -3.360892629446941294068571098245987753108d0
    -8.682193468417260068181898914525741593117d-1
     2.759209969944670830494156007968549546225d1
     2.015406755047789340861867889788636416711d1
    -4.348988418106995884773662551440329218107d1
1d0 !A[10,1:9]
     4.776625364382643658904339085273915228682d-1
     0.0d0
     0.0d0
    -2.488114619971667641926425864682396693132d0
    -5.902908268368429963714464757431352217614d-1
     2.123005144818119423472889498970487302677d1
     1.527923363288242358325969229374844220878d1
    -3.328821096898486291944532655869588229067d1
    -2.033120170850862613582229285929255284559d-2
1d0 !A[11,1:10]
    -9.371424300859873257170402165804155281433d-1
     0.0d0
     0.0d0
     5.186372428844063708300238532089536376143d0
     1.091437348996729578185002546539049230918d0
    -8.149787010746926125139972673569091879683d0
    -1.852006565999695986415661807011477507839d1
     2.273948709935050428189700567335972730811d1
     2.493605552679652389870893967618881717874d0
    -3.046764471898219500382366902200055003968d0
1d0 !A[12,1:11]
     2.273310147516538207923597684493877424478d0
     0.0d0
     0.0d0
    -1.053449546673725019840666898789285791355d1
    -2.000872058224862499096757184438171968477d0
    -1.795893186311879891727659505336182733784d1
     2.7948884529419960050849980883715913462d1
    -2.858998277135023694740655086735730541988d0
    -8.872856933530629544335492892584644715629d0
     1.236056717579430306472662015275737291514d1
     6.433927460157635303559704840460686758734d-1
1d0 !BH[1:12]
     5.429373411656876223805357663625380160813d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     4.450312892752408881441139505655634687893d0
     1.891517899314500383042815990435932981641d0
    -5.801203960010584781467211422697151945646d0
     3.111643669578198944089160623697586176069d-1
    -1.521609496625160785561788068053732214133d-1
     2.013654008040303483747765375007018995575d-1
     4.471061572777259051768855690424317875298d-2
1d0 !B[1:12]
     6.331868142253823210370308649690393639067d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     2.0d0
     1.135262836251133593499719928992383448074d0
    -2.715355126634071820420777390315663860222d0
     5.400729799378105855575261431813780737951d-2
     2.069143347541890194818812659711202364131d-1
     2.111413604846573262620319376328752532114d-1
     4.471061572777259051768855690424317875298d-2
1d0 !A[13,1:12]
     5.429373411656876223805357663625380160813d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     4.450312892752408881441139505655634687893d0
     1.891517899314500383042815990435932981641d0
    -5.801203960010584781467211422697151945646d0
     3.111643669578198944089160623697586176069d-1
    -1.521609496625160785561788068053732214133d-1
     2.013654008040303483747765375007018995575d-1
     4.471061572777259051768855690424317875298d-2
1d0 !A[14,1:13]
     4.03330195819761531508328593985012610567d-2
     0.0B0
     0.0B0
     1.401208525297287226738461430547893663357d-1
     2.090980513561860835217933390681938815397d-2
    -1.071505384322163055562630099330827728709B0
    -4.780634138838958589917771048135808105849d-1
     1.435379844265730056619887879698127976286B0
    -9.541176947942155464643504264500465717176d-2
     1.327216241985937616855964214927888887843d-1
     3.959365633189334535516871556627405195523d-3
     4.127038980448884299433363085382451867648d-4
    -5.061320934835304737647223381195407190761d-4
1d0 !A[15,1:14]
     1.601033543883667492565459261178492947475d-2
     0.0B0
     0.0B0
     5.76920863378842084917677730881464530362d-1
     1.4904463782966680395363706726908330376l1d-1
    -4.6755355048954871539027163538727178736 83B0
```

```
     1.49044637829666803953637067269083303761 1d-1
```

```
     1.490446378296668039536370672690833037611d-1
    -4.675535504895487153902716353872717873683B0
    -2.011266802109947808141531259403564329243B0
     6.273634723693586166116241520740532440848B0
    -3.502912579204025742085999826260182511934d-1
     4.96339178284634089324812676098234443589d-1
     1.37067676656239142113449076312461063121d-2
    -1.273520423920135286755442746765421829924d-3
     1.091196791233134166868489399720818681277d-3
    -2.499190792711267345381724075214622355406d-1
1d0 !A[16,1:15]
    -2.13017938571114639005055316222988408605d-1
     0.0B0
     0.0B0
     2.11258078191740168322637312499046385091 2B0
     7.760317060712801142951220774289030354668d-1
    -1.60305745783658141347245215726399600789 2d1
    -1.06122617904362949859234167583334681413 8d1
     2.43011907759140763920091933734726107380 7d1
    -4.963223138141681530009756655777195944722d-1
     7.717596880709187410919617270799752156281d-1
     1.906147223888269435442428763304333157278d-2
    -1.012563995691701573866898494476409150825d-2
     9.484106980958953146584436033210655155526d-3
    -4.031109015561899870909602390469240955322d-1
     2.50033020737022850350019812442213710 21d-1
1d0 !A[17,1:16]
    -4.078888665728777600229883098616199488167d-2
     0.0B0
     0.0B0
     6.966336680120998837451633194419774338385d-1
     1.996479228863924834395374020257075784742d-1
    -3.080634680289059622490927304223628086133B0
    -2.502715775176515140191997678141815451634B0
     5.247263609225062036520165130160098308442B0
    -1.259990264873402817428145473830632035577d-1
     2.034674187193701370754186188974090395598d-1
     5.875626514148771044355985341859740569043d-3
    -7.836234908834481223263895334932691544616d-3
     7.518668283109401536708077609782681243603d-3
    -7.642893250851160022205697347834594038951d-2
    -6.298471834747388644038841251685157536504d-3
    -5.541919149217213712966474839148797073676d-2
1d0 !A[18,1:17]
    -6.16761234359009761919639391531993541825d-1
     0.0B0
     0.0B0
     4.801037603034258015195489407005035359981B0
     1.435638518844034563811185856680724874808B0
    -2.651485961258290027813447826279614075367d1
    -2.16000372772423481789297664562262462175d1
     4.317932547224609182625385074191824207846d1
    -6.640514245667238251351810282688714539782d-1
     1.251759306183693319836288313622065056579B0
     5.326711345669278664624125506955885604752d-2
    -5.985415143345306332911738317011310395145d-2
     5.36744927393099194042485359832538373772d-2
    -9.514724400320895757241037293336152528218d-1
     7.312792466530634133241044295288551258015d-1
     7.621159690271623181635879181439980009415d-2
    -4.70611755297880847660935625749700119952d-1
1d0 !A[19,1:18]
     3.641575400615103731791070242747808513304d-2
     0.0B0
     0.0B0
     2.325247659011449967738592251444928430476d-1
     1.717014931208695161092028468004015008672d-1
    -1.193610302182198412200842157574150807524d-1
    -2.000535039847340865558034285023361144871d-1
     3.206972700056131309593983925946500480444d-1
    -7.669314912945063366461740883157683954069d-2
     1.268281787587124571306731095530905056048d-1
    -4.323621304395577387721542956686185072895d-5
    -4.199376119496727955453609361587680423732d-3
     4.427302833597985223570849166422374361054d-3
    -5.587269320988770008034178010322452012922d-2
     3.227772173291663017809632036447409262259d-2
```

```
    -2.689217499883437047052311152536342725045d-2
     9.78902953586497890295358649789029535865d-2
     8.13165537270087124878993223620522749274d-3
1d0 !BS[1]
     1.0B0
    -1.022786037512902589442119713543207996331d1
     5.220605189298039279840704984307973164442d1
    -1.49357484107852888975982424538286846695d2
     2.524981725133100863388417252310253598655d2
    -2.515708309499291153291650663968920788544d2
     1.368483334501442690181310174641393168632d2
    -3.134208868940714919357305089099714905872d1
1d0 !BS[2]
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
1d0 !BS[3]
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
1d0 !BS[4]
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
1d0 !BS[5]
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
     0.0B0
1d0 !BS[6]
     0.0B0
     7.394867889107582391764917218905147683947d1
    -1.398910667539143593518909467368219860835d3
     7.861375309145840706283359212787836699156d3
    -1.984402033249406453707305645747117629294d4
     2.518086104865990738008921422036669859626d4
    -1.568869856892970167895251449971917958966d4
     3.819894845158838308135698958720644605875d3
1d0 !BS[7]
     0.0B0
    -6.71905688476585398471909903063474425266d1
     5.835538964317821688394485292776020041357d2
    -1.932288993682393427599417512507128365097d3
     3.358012276472363064322203853517941041805d3
    -3.393521360402621583602367074747622019799d3
     1.942667940239284082586038462343496472903d3
    -4.893416723114412643156724515875057584404d2
1d0 !BS[8]
     0.0B0
    -2.41155893658623595206744073589599249642d1
     9.832885698732886829216484334809031311939d2
    -6.455136969953907761809077961659073825799d3
     1.724598216822612774021321507568564715686d4
    -2.232069958657113827773644053019580165997d4
     1.390584160342817044633361426533862383081d4
    -3.340961399596689055183752086714035860078d3
1d0 !BS[9]
     0.0B0
```

```
      1.49468839193451665862558370282390839367d1
```

Alternative b^* coefficients for the $86D^7_{16}L_2$:

```
1d0 !BS[1]
     1.0d0
    -9.4907551017256450793912218029930406238l6d0
     4.3192053035649657720870065710646842437 96d1
```

```
1d0 !BS[1]
     1.0d0
    -9.490755101725645079391221802993040623816d0
     4.319205303564965772087006571064684243796d1
    -1.038275016159632538809026080355486514794d2
     1.36706641029821360113237984248182354863d2
    -9.286515770690738273462592318088714530946d1
     2.533901409324183262304975663723589391326d1
1d0 !BS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[3]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[4]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[5]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[6]
     0.0d0
     1.639686681130215643848312560985096030282d0
    -2.096663436535879560299334483875382595346d2
     1.44156286516680027659124669007020651197d3
    -3.023901052848125082635173534588706599733d3
     2.584732641653816943567187206251610664359d3
    -7.899174841072819882557340864009017784034d2
1d0 !BS[7]
     0.0d0
    -4.822187183539036177449147431614505058846d1
     4.154537680548318004898008315234826247123d2
    -1.230306386828997250233088583902078486567d3
     1.813344020341129164096337447855658311457d3
    -1.343233967942440467898855386585234878249d3
     3.948559561101816157033399814147534122166d2
1d0 !BS[8]
     0.0d0
     2.745493141043992199451447864302847402108d1
    -1.476154138468880407851281138716761816053d1
    -8.223580283330864699728606839517276307188d2
     2.113170067377879404188060635363585570573d3
    -1.878102968990017885586330846497340210072d3
     5.687963359594632486736620164069242624119d2
1d0 !BS[9]
     0.0d0
    -2.690372127053307980585346438789920722867d0
     2.966450359947175603138330903548449933749d1
    -1.034903825476339226608397982005987786899d2
     1.645528286646952437724775212129719820078d2
    -1.216625128804044645241797393355477113583d2
     3.393709965788251525615296978884968804339d1
1d0 !BS[10]
     0.0d0
     3.052883427115193223005206569308431610214d0
    -3.120829024044490250918622281397146421807d1
     9.330610765570351413398819230447524174583d1
    -1.199194998703277979394013242880317297062d2
     6.842429395211108484866367842434382044631d1
    -1.380765587381960783562570900292967309946d1
1d0 !BS[11]
     0.0d0
     5.044693758219424243918020716499610329825d-1
    -6.335318767366073761684106041606562021118d0
     2.845127164504617845886924448264458449257d1
    -5.912067374678297335535434487107330083488d1
     5.711601855441020669750180586489420440851d1
    -2.041440166032525011534962496900818517851d1
1d0 !BS[12]
     0.0d0
     8.914373544052028487632503347562570441788d-2
    -1.521170151934398941856315382803851796144d0
     8.031694983264530175795177455995886345917d0
    -1.888429114171957265542522045584125227599d1
     1.962543357427495726013236480586182642798d1
    -7.296100383598263533004642899783991227431d0
1d0 !BS[13]
     0.0d0
    -2.008478281160838604650898499382599106485d-1
     2.953379154642584678825106336134310977665d0
    -1.475081456549023525040491651937563972917d1
     3.353947660630326200133353944330870243901d1
    -3.463578699412573766442730542333643004791d1
     1.309459362678621009513866601320731627106d1
1d0 !BS[14]
     0.0d0
     2.400094042484381861321089520010406328164d1
    -1.647834217205121086206474936050827050714d2
     4.631578312156442216582186881947985448959d2
    -6.643362053885865926479777390576116816155d2
     4.783279018880698617378349119506382324207d2
    -1.363670464194592007406392626828464539113d2
1d0 !BS[15]
     0.0d0
     7.418890832618261627896487608493611746379d0
    -1.059077013551791466880191123559416138277d2
     4.21762199581169886348573328193950135004d2
    -7.135524229479645032206217224646181964922d2
     5.483545984100446260081174717286843845516d2
    -1.58075564520689124075946452710568320982d2
1d0 !BS[16]
     0.0d0
    -3.557098995124475116810375279178991581207d0
     4.292008342911759170896019736836379716415d1
    -1.815388563564574753685947300927417172697d2
     3.384011119236780882825067576021758393178d2
    -2.86080493518831741711018238003686757577d2
     8.985525351761801220495638840506782994597d1
```

B.3 The RKN12(10)17M with continuous extensions

B.3.1 Discrete coefficients

```
RKN121017M explicit Runge-Kutta-Nystrom pair
                             (approximate assumed form)

[p] 10'th order

----------------------cut-----------------------

17

1d0 !A[2,1]
     2.0d-4
1d0 !A[3,1:2]
     2.6666666666666666666666666666666666666666d-4
     5.3333333333333333333333333333333333333333d-4
1d0 !A[4,1:3]
     2.9166666666666666666666666666666666666666d-3
    -4.1666666666666666666666666666666666666666d-3
     6.25d-3
1d0 !A[5,1:4]
     1.6460905349794238683127572016460905349794d-3
     0.0d0
     5.4869684499314128943758573388203017832647d-3
     1.7558299039780521262002743484224965706447d-3
1d0 !A[6,1:5]
     1.9456d-3
     0.0d0
     7.1517460317460317460317460317460317460317d-3
     2.9127111111111111111111111111111111111111d-3
     7.8994285714285714285714285714285714285714d-4
1d0 !A[7,1:6]
     5.6640625d-4
     0.0d0
     8.8097304894179894179894179894179894179894d-4
    -4.3692129629629629629629629629629629629629d-4
     3.3900669642857142857142857142857142857142d-4
    -9.9464699074074074074074074074074074074074d-5
1d0 !A[8,1:7]
     3.0833333333333333333333333333333333333333d-3
     0.0d0
     0.0d0
     1.7777777777777777777777777777777777777777d-3
     2.7d-3
     1.5782828282828282828282828282828282828282d-3
     1.0860606060606060606060606060606060606060d-2
1d0 !A[9,1:8]
     3.6518393748011297137511915033756259593590d-3
     0.0d0
     3.9651717140723430661755728980699942022920d-3
     3.1972582629306282235009342609103389657310d-3
     8.2214673068554353696870188340120364416460d-3
    -1.3130926959572379836201388486250712799210d-3
     9.7715869680648678156260949414678372951670d-3
     3.7557690692328337948793264107892384157230d-3
1d0 !A[10,1:9]
     3.7072410687185008101956553052115036112710d-3
     0.0d0
     5.0820458545552859807610816347850385721110d-3
     1.1747080021754120447356910494265790493370d-3
    -2.1147629915126991499622976636190748209470d-2
     6.0104636981078808122257352513561188889220d-2
     2.0105734768506188184674870877690624640630d-2
    -2.8350750122933580843036677436840988960640d-2
     1.4879568918581932755590558247912357963080d-2
1d0 !A[11,1:10]
     3.5125376560733441531130829305195711007610d-2
     0.0d0
    -8.6157491951384791034057607854521398212800d-3
    -5.7914480510079165216763225247049327324720d-3
     1.9455548237826158423943881041082212419070d0
    -3.4351238674565135963678716757446378529770d0
    -1.0930701107475221758389257200093615508160d-1
     2.3496383118995166394320161088023011620160d0
    -7.5600940868702297802719072977820067854320d-1
     1.0952897222156926424650201861821364422390d-1
1d0 !A[12,1:11]
     2.0527792537482496650972057167187460324330d-2
     0.0d0
    -7.2864467644801799177824794314921673967810d-3
    -2.1153556079618402406925956254916766402010d-3
     9.2758079687235222425676803323474583612570d-1
    -1.6522824844257366790730267332515582173780d0
    -2.1079563005686569819191436691244241911950d-2
     1.2065364326207871544770883253639878902070d0
    -4.1371447700106614132466246364551084998810d-1
     9.0798739828096537595679573951550006960250d-2
     5.3555526005339850491687065821469473516410d-3
1d0 !A[13,1:12]
    -1.4324078875545515045892109163157200039300d-1
     0.0d0
     1.2528703773091817277846448023061922578100d-2
     6.8260191639698271286811241173656368930330d-3
    -4.7995553955743872655021625429151289627790d0
     5.6986250439519414337916979415594045534640d0
     7.5534303695236452224944402871632878141260d-1
    -1.2755487858281083717540079654149043114480d-1
    -1.9605926051117384328913325542295306432570d0
     9.1856090566352624097623428534066007024930d-1
    -2.3880085505284431053482701340194805826750d-1
     1.5911081357234215513874017096284913124510d-1
1d0 !A[14,1:13]
     8.0450192055204894869723077813407970348730d-1
     0.0d0
    -1.6658527067011245177851626826127272551810d-2
    -2.1415834042629734811731437190946361358100d-2
     1.6827235928962465870200935356424849088770d1
    -1.1172835357176097926788298424133641790600d1
    -3.3771592972263237414885647552130792819410d0
    -1.5243326655360845646181768293869900735770d1
     1.7179835738215416562024768402623473945220d1
    -5.4377192398239946453541373855558181089360d0
     1.3878671618364655755125677883937873221750d0
    -5.9258277326528116534767702918064836060920d-1
     2.9603873171297352796159279455196341913010d-2
1d0 !A[15,1:14]
    -9.1329676669735808209625048264769474072330d-1
     0.0d0
     2.4112725757805178392448994610235986465070d-3
     1.7658122693861741982069883922560595692480d-2
    -1.4851649779720383824612855708792253227770d1
     2.1589708670045756003078216156126174607730d0
     3.9979155831178799011528275433674841167520d0
     2.8434151800232231898454251498750324492480d1
    -2.5259364354941598437884335223543584057780d1
     7.7338785423622373655340014113947067061050d0
    -1.8913028948478674610382580128979697245520d0
     1.0014845070224717803668595924825669301780d0
     4.6411995991090519051051824705183691337800d-3
     1.1218755022148957033975049906341685556650d-2
```

```
1d0 !A[16,1:15]
    -2.751962972055939382060652270387676365146d-1
     0.0d0
     3.661188877915492013422932855529192323348d-2
     9.789519688231562624650996716202314779068d-3
    -1.229306234588621030421472650901098551183d1
     1.420722645393790269429296659658848348008d1
     1.586647690678953683224819642722838612682d0
     2.457773532759594543903243469746394379786d0
    -8.935193694403271905522590863741025993646d0
     4.373672731613406948393270775123931046714d0
    -1.834718176544949163043444102640770929574d0
     1.159208528906149120780831983726127644833d0
    -1.72902531653839221518003422953246227118d-2
     1.932597790446076667276498753234313305803d-2
     5.204442937554993111849264015262159106749d-3
1d0 !A[17,1:16]
     1.307639184740405758799945629832338566267d0
     0.0d0
     1.736410918974584186708799912955289976316d-2
    -1.854445645426579502436211558797049585675d-2
     1.481152203286772689684783562232341208278d1
     9.383176308482470907879221771263864818455d0
    -5.228426199944542254147402455301063361317d0
    -4.895128052584765080400934827429129894568d1
     3.829709603433792256255838758364694015523d1
    -1.058738133697597970916190375053123568191d1
     2.433230437622627635851196187870571003321d0
    -1.045340604257544428486524565126094844671d0
     7.177320950867259451981848575083494459676d-2
     2.162210970808278269055053200267534001509d-3
     7.009595759602514236992827819881325015789d-3
     0.0d0
1d0 !BH[1:17]
     1.212786851718541497688903954954288801232d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     8.629746251568874443637922744111624697468d-2
     2.525469581187147194323434493163185776171d-1
    -1.974186799326823033583079548858569280766d-1
     2.031869190789725908092615610088713014794d-1
    -2.077580807771491661219335546909220230251d-2
     1.0967804874502013625011123782257836373d-1
     3.806513252646650573448787191046844353843d-2
     1.163406880432422964409277092150432335028d-2
     4.658029704024878686936152384548985676779d-3
     0.0d0
     0.0d0
1d0 !BDH[1:17]
     1.212786851718541497688903954954288801232d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     9.083943422704078361724129204328025997335d-2
     3.156836976483933992904293116453982220214d-1
    -2.632249065769097378110772731811425707689d-1
     3.047803786184588862138923415133069522192d-1
    -4.155161615542983322438671093818440460502d-2
     2.467756096762953065627502851008013183926d-1
     1.522605301058660229379514876418737741537d-1
     8.143848163026960750864939645053026345199d-2
     8.502571193890811280080183268810684284445d-2
    -9.155189630077962873141002513513545695226d-3
     2.5d-2
1d0 !BH[1:17]
     1.700870190700699175275446461887139853241d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     7.225933593083140694886000384628917149598d-2
     3.720261773267530453882105020665750223097d-1
    -4.018211450093035214393402338631191593696d-1
     3.354550683013516666965840348956183022491d-1
    -1.313065010753318084302818407834344757679d-1
     1.894319066160486527226598364552047093567d-1
     2.684080204002904790536916558056721439147d-2
     1.63056656059179238935180933101983663164d-2
     3.799988356696594561665973873229450485631d-3
     0.0d0
     0.0d0
1d0 !BD[1:17]
     1.700870190700699175275446461887139853241d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     7.606245887455937573564210931188333841682d-2
     4.650327216584413067352631275832187778872d-1
    -5.357615266790713619191203118174922124928d-1
     5.031826024520275000448760523434274533737d-1
    -2.626130021506636168605636815668689515359d-1
     4.262217898861094686259846320242105960526d-1
     1.073632081601161916214766623222688575659d-1
     1.141396592414254672546266531713885642148d-1
     6.936338665004867700906029200909217798516d-2
     2.0d-2
     0.0d0
```

B.3.2 Continuous extension $\mathbf{D}^{11(10)}_{26(23)}\mathbf{L}_2$

```
RKND11L2 explicit Runge-Kutta-Nystrom triple
                          (approximate assumed form)

[p] 10'th order

-----------------------cut------------------------

17 9 10

  !!!!!   Discrete coefficients removed   !!!!!

1d0 !A[18,1:17]
     1.21278685171854149768890395495428880 1232d-2
```

```
      1.5457007336882761405988508176598601687 37d-2
```

```
      1.545700733688276140598850817659860168737d-2
     -1.179615785757245267588444550680955758862d-2
      2.844276797408162275826904284910506686816d-3
     -1.235426798733333549152441266339679963968d-3
      0.0d0
     -9.137136995104077333796059220655872731846d-4
      0.0d0
      1.039686433917816069249585887269365731475d-3
      3.204089807336647102786865749023038661357d-4
      3.335528945294107415986909696485112475925d-6
      8.588313020033804586481730275622807270804d-5
      0.0d0
      0.0d0
1d0  !A[25,1:24]
      8.352298736061804943914474993961351864571d-3
      0.0d0
      0.0d0
      0.0d0
      0.0d0
     -8.306964067032100684448590903260626521143d-4
      8.378515425855513418068802250721957732553d-2
      1.68550877241619979501138590309829715652d-1
     -7.649202877617220452818969819309448764089d-2
      1.125774629840374205939994518847456076732d-1
      2.895405242733390170045992475427769312449d-2
      5.178985673405385006621189623901812337665d-2
      2.619208358209739313464680734760331994324d-2
      1.09959879645348417254395880236299870659d-3
      0.0d0
     -4.619414472380841188533261156472378801042d-4
      0.0d0
      5.660298654704075899273316667351217236166d-4
     -3.994536319106461666023271281104612730272d-3
      8.379984370917163081409187018495453329262d-6
     -1.666001205686356409474709001960393999453d-5
      0.0d0
      0.0d0
      0.0d0
1d0  !A[26,1:25]
      2.694097022706314385791371809025469441569d-3
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      1.491694418720265179749939016125773049458d-3
      4.197196992984104908827644289533356302846d-3
      7.543204438725726289111932035233287827925d-3
     -1.463149802165347984207692161113475645872d-2
      6.793694807236131802649541545857827404202d-3
     -8.531833921978476699835602015754278732141d-4
     -5.242679791061923367369968750376881123805d-4
      6.521684993396820442239875801694900702306d-4
     -2.769293823242899431209629156537940704293d-4
      0.0d0
     -7.630962254182173031270479856805664771482d-4
      0.0d0
      7.946643792102302504027937885727470536493d-4
      1.531640318070117593599912810109318692321d-3
      4.239322040496553312607131907283355289236d-7
     -8.566971870828983911481794015576518583276d-5
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BS[1]
      5.0d-1
     -7.16068656507275465779294910069600639605d0
      5.63680109241541861188530899381418933419 3d1
     -2.709337363662595859676091134271281577504d2
      8.403325456193291443380820309688552931485d2
     -1.734122802733853631751567648713415522541d3
      2.40471046867350876290516652322655257923d3
     -2.214558946925961888484763292934047657651d3
      1.298909778711630699963916690546152714039d3
     -4.391941985761900490575421416163559313826d2
      6.516169510723230200823370015149033884875d1
1d0  !BS[2]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BS[3]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BS[4]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BS[5]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BS[6]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BS[7]
      0.0d0
      1.232344901424957528712957174708652195237d1
     -1.427746826275557403642681045135139654063d2
      8.056816108411381114217801180139330990409d2
     -2.714816819502226431023358399350917708965d3
      5.875693109975788913690318420694048700816d3
     -8.388794689687265991251403484796604672809d3
      7.869235565512795245424037083766850019255d3
     -4.670187845495068771798027319208324643926d3
      1.590742994455423979545252792661832118127d3
     -2.370163950247632021870242997869483518388d2
1d0  !BS[8]
      0.0d0
      9.80464800344974675318406876940602179959d0
     -1.812617155881238589627015286844481542935d2
      1.346776281151967111396667998744719434392d3
     -5.255891895200705827692272192963331535343d3
```

```
      1.22905752982089176213477805269827713265d4
     -1.82211987354200899985951522082169153l952d4
      1.732097897074542516796309289102545187506d4
     -1.024986467108065363781075950009900288407d4
      3.441195056174988179907756994273489422707d3
     -5.00860690037055789588164706382823868653d2
1d0  !BS[9]
      0.0d0
      9.243114940271519456202954442983929669102d-1
     -1.644082077565157801847798643457364763185d1
      2.268359452278819066109665450357133182804d2
     -1.544993244909005104694542342787224096099d3
      5.44514088815153164603885332854533014416d3
     -1.106740015077080219225719308801944720163d4
      1.360419853732286296805191834933373719054d4
     -1.003132615597947998984396560012360204788d4
      4.097162678681131874991922717362936462313d3
     -7.142994071224293651284605263120543719384d2
1d0  !BS[10]
      0.0d0
     -1.556298312743532031364714163710007330492d0
      3.583700919226722571024369528430038201964d1
     -4.113359753253022842722576110149274581557d2
      2.391644583462330005021381136902336374398d3
     -7.653257520208414161975621516333641707438d3
      1.457354390312771784276885772383723028901d4
     -1.705278195631157805539723304783177605505d4
      1.207569261651866060021882543078988320754d4
     -4.761998975141368440861928464485096749598d3
      8.044157999175097734099066285764105959042d2
1d0  !BS[11]
      0.0d0
     -1.724072821752897652638315757661966147899d0
      2.372334020860455821968805189390986910336d1
     -1.498952500259112677457628535371513465598d2
      5.4652906397317315490714155121792738664d2
     -1.297809739310299954669675031501229773263d3
      2.118864857982601050656770422634468027169d3
     -2.381950172756909700320183459480334048359d3
      1.757522014611418871783352220263473582384d3
     -7.599897043425106631572514345521853354898d2
      1.44708886673509133061946655463314512321d2
1d0  !BS[12]
      0.0d0
      2.67247888604376305508327945680158362655d0
     -3.555589302422155424746084403824399303628d1
      1.904196762673626479203327098803523052563d2
     -5.42327818615691051325433712107304446689d2
      1.051480812627540335461854188415541712312d3
     -1.689390147052858014900772279847176699732d3
      2.231955644877565719465255461289439618148d3
     -2.034350116678842130488933153632351337251d3
      1.055483196815967770198567225952250431466d3
     -2.302781560541224650022427641314865957364d2
1d0  !BS[13]
      0.0d0
      2.089538684992690736754510605431206662319d0
     -4.029000533964882824318629970055981394646d1
      3.318609906899712191836349878476843379785d2
     -1.532581818017608789520332954979679437401d3
      4.378250362845512055326903305418799752093d3
     -8.014877044976004189819785684108339492911d3
      9.399512282789942964291823208357080550839d3
     -6.818397551428175479864916251713200096095d3
      2.782797309051300893696117607769685408571d3
     -4.883259991677560692812779416249919473458d2
1d0  !BS[14]
      0.0d0
      1.580397503294880875366743590300328098177d0
     -3.029422186043361411279169697210652000007d1
      2.531503480140423447785596827149617367114d2
     -1.187684608453947144799250682398684952199d3
      3.433119149122610657224041353106831389516d3
     -6.341321616241981875872839051529587799438d3
      7.499262354013222148221457376462704237623d3
     -5.490721110082885912909864726824787798442d3
      2.265589459804202660044346427551791419137d3
     -4.026685177493198192193813329305005366827d2
1d0  !BS[15]
      0.0d0
      1.660768271980838393657732574741675761269d0
     -3.255874181215354524129341986150636733646d1
      2.765557804307198950952088346596767433211d2
     -1.319576531469767748048109391306526053481d3
      3.885665862228902454911947543816267262534d3
     -7.321206512232014240850345867763829946308d3
      8.839769277798005148535158267150060739123d3
     -6.611521539211429388563553341955906956574d3
      2.787294599309865449804342337372336623811d3
     -5.060783052844048391583257585329291718644d2
1d0  !BS[16]
      0.0d0
     -1.089700423363680417711588450809229748744d-1
      2.305921686961455321324956358845220673422d0
     -2.014727197320806591369613808952803248245d1
      9.96241350486062477673831319157838908289d1
     -3.089674965048272611828184898270392284517d2
      6.185707407303364801025111167451032569252d2
     -7.949062403210232439036492030850969486235d2
      6.311577487127184886992052196012896057772d2
     -2.8121365126379213762362675292303231379d2
      5.368508392656440477513731814875547211784d1
1d0  !BS[17]
      0.0d0
      4.612036310475900077015487189239338027344d-1
     -9.319326830885617719723557722276850508882d0
      8.056983431546243430390078717938106526959d1
     -3.902892711442864133287256604543068661251d2
      1.16596687519807090694898862199042683203d3
     -2.22761120039507875960795396848474362605d3
      2.725600112910566417761641100097642875681d3
     -2.064366521255266629703113470898644372479d3
      8.806686979939325813416993812692736016734d2
     -1.616804044235625100044147816956765932942d2
1d0  !BS[18]
      0.0d0
     -6.291619439460042333485065325450426255604d-1
      1.247410249298570155027551411303549942797d1
     -1.070923297262107659712485639543062187623d2
      5.148539973932188558730904733406143079782d2
     -1.521962396582974202197279226662109054545d3
      2.871847955340885431550167394101876581723d3
     -3.468710164809142574538351217348562347941d3
      2.594670651742456912579536136993361623811d3
     -1.09439868623036616450016251151291818629d3
      1.989460323230928098873205074615528372245d2
1d0  !BS[19]
      0.0d0
     -3.702409973596982539941496757235298374163d0
      5.944915734736810051863306205239961648922d1
     -4.492541087813297728001409445649426065587d2
      1.927486719007364016088962462003810815029d3
     -5.082139355585835903383185265556309132925d3
      8.552220599400354252387164655388001843809d3
     -9.237252201624878835451898965093497688267d3
      6.210572416951519570249252670115487323265d3
     -2.369986970292087774997919924393289183149d3
      3.926061535511233299290737468055743106823d2
1d0  !BS[20]
      0.0d0
      9.228125338647721118060247148534501296402d-3
     -1.148264951819391100719720210083765134342d-1
      7.283547197029199955780557987876679253861d-1
     -2.836917927003141587232604521850993054431d0
      7.126889808600210703470544015463639134181d0
     -1.173778634343392523341021716266068050611d1
      1.259344673387003658370095544997933384327d1
     -8.481271082819884362232059784499140122134d0
      3.257481652270399334511872275707289512661d0
     -5.445991913433240454326342970672747206927d-1
1d0  !BS[21]
      0.0d0
      2.329649202276119436613556160295132047497d-3
     -1.647722788757466814512808370724449019431d-1
     2.0518782444685187973851456937324487623 95d0
```

```
     2.05187824446851879738514569373244876239 5d0
```

```
      2.941394401034924025955220630821806539877d1
     -7.25046862352495435850806114737792617174d2
      6.733881405759835556983339993723597171963d3
     -3.153535137120423496615363315777998921206d4
      8.603402708746242334943446368887939928554d4
     -1.457695898833607199887612176657353225562d5
      1.558888107367088265116678360192290668755d5
     -1.024986467108065363781075950009900288407d5
      3.785314561792486997898532693700838364978d4
     -6.010328280444669475057976476593886423836d3
1d0  !BDS[9]
      0.0d0
      2.77293448208145583686088633289517890073d0
     -6.576328310260631207391194573829459052742d1
      1.134179726139409533054832725178566591402d3
     -9.269959469454030628167254056723344576599d3
      3.811598621706072152227197329981731100912d4
     -8.85392012061664175380575447041555776131d4
      1.224377868359057667124672651440036347148d5
     -1.003132615597947998984396560012360204788d5
      4.506878946549245062491114989099230108544d4
     -8.571592885469152381541526315744652463261d3
1d0  !BDS[10]
      0.0d0
     -4.668894938230596094094142491130021991477d0
      1.433480367690689028409747811372015280785d2
     -2.056679876626511421361288055074637290778d3
      1.434986750077398003012828682141401824639d4
     -5.357280264145889913382935061433549195207d4
      1.16588351225021742742150861790697842312d5
     -1.534750376068042024985750974304859844954d5
      1.207569261651866060021882543078988320754d5
     -5.238198872655505284948121310933606424558d4
      9.652989599010117280918879542916927150851d3
1d0  !BDS[11]
      0.0d0
     -5.172218465258692957914947272985898443697d0
      9.489336083441823287875220757563947641345d1
     -7.494762501295563387288142676857567327991d2
      3.27917438383903892944284930730756431984d3
     -9.084668175172099682687725220508608412847d3
      1.695091886386080840525416338107574421735d4
     -2.143755155481218730288165113532300643523d4
      1.757522014611418871783352220263473582384d4
     -8.359886747767617294729765780074038690388d3
      1.736506640082109596743359865559774147852d3
1d0  !BDS[12]
      0.0d0
      8.017436658131289165249838370404750879651d0
     -1.422235720968862169898433761529759721451d2
      9.520983813368132396016635494017615262817d2
     -3.253966911694146307952602272643826680134d3
      7.360365688392782348232979318908791986184d3
     -1.351512117642286411920617823877741359786d4
      2.008760080389809147518729915160495656333d4
     -2.034350116678842130488933153632351337251d4
      1.161031516497564547218423948547475474612d4
     -2.763337872649469580026913169577839148836d3
1d0  !BDS[13]
      0.0d0
      6.268616054978072210263531816293619986957d0
     -1.611600213585953129727451988022392557858d2
      1.659304953449856095918174939238421689892d3
     -9.195490908105652737121997729878076624407d3
      3.064775253991858438728832313793159826465d4
     -6.411901635980803351855828547286671594329d4
      8.459561054510948667862640887521372495755d4
     -6.818397551428175479864916251713200096095d4
      3.061077039956430983065729368546653949428d4
     -5.85991199001307283137533529949990336815d3
1d0  !BDS[14]
      0.0d0
      4.741192509884642626100230770900984294531d0
     -1.211768874417344564511667878884260800002d2
      1.265751740070211723892798413574808683557d3
     -7.126107650723682868795504094392109713194d3
      2.403183404385827460056828947174781972661d4
     -5.07305729299358550069827124122367023955d4
      6.74933611861189999333993116388164338l386d4
     -5.490721110082885912909864726824787798442d4
      2.492148405784622926048781070306970561051d4
     -4.832022212991837830632575995166006440193d3
1d0  !BDS[15]
      0.0d0
      4.982304815942515180973197724225027283809d0
     -1.302349672486141809651736794460254693458d2
      1.382778902153599475476044173298383716605d3
     -7.917459188818606488288656347839156320891d3
      2.719966103560231718438363280671387083773d4
     -5.856965209785611392680276694211063957047d4
      7.955792350018204633681642440435054665211d4
     -6.611521539211429388563553341955906956574d4
      3.066024059240851994784776571109570286192d4
     -6.072939663412858069899909102395150062373d3
1d0  !BDS[16]
      0.0d0
     -3.269101270091041253134765352427689246234d-1
      9.22368674784582128529982543538088269369d0
     -1.007363598660403295684806904476401624122d2
      5.977448102916374866042987914947033449734d2
     -2.162772475533790828279729428789274599162d3
      4.948565925842691840820088933960826055402d3
     -7.154156162889209195132842827765872537612d3
      6.311577487127184886992052196012896057772d3
     -3.09335016390171351385989428215335545169d3
      6.442210071187728573016478177850656654141d2
1d0  !BDS[17]
      0.0d0
      1.383610893142770023104646156771801408203d0
     -3.727730732354247087889423088910740203552d1
      4.028491715773121715195039358969053263479d2
     -2.341735626865718479972353962725841196751d3
      8.161768126386496348642920353932987824212d3
     -1.78208896031606300768636317478779490084d4
      2.453040101619509775985476990087878588113d4
     -2.064366521255266629703113470898644372479d4
      9.687355677933258394758693193962009618407d3
     -1.940164853082750120052977380348119119531d3
1d0  !BDS[18]
      0.0d0
     -1.887485831838012700045519597635127876681d0
      4.989640997194280620110205645214199771188d1
     -5.354616486310538298562428197715310938115d2
      3.089123984359313135238542840043685847869d3
     -1.065373677608081941538095458663476338181d4
      2.297478364272708345240133915281501265378d4
     -3.121839148328228317084516095613706113147d4
      2.594670651742456912579536136993361623811d4
     -1.203838554853402780950178762664210004919d4
      2.387352387877113718647846089538634046694d3
1d0  !BDS[19]
      0.0d0
     -1.107229920790947619824490271705895122249d1
      2.377966293894724020745322482095984659569d2
     -2.246270543906648864000704722824713032793d3
      1.156492031404418409653377477202286489017d4
     -3.557497548910085132368229685889416393047d4
      6.841776479520283401909731724310401475047d4
     -8.31352698146239095190670906858414791944d4
      6.210572416951519570249252670115487323265d4
     -2.606985667321296552497711916832618101464d4
      4.711273842613479959148884961666891728187d3
1d0  !BDS[20]
      0.0d0
      2.76843760159431633541807414456035038892d-2
     -4.593059807277564402878880840335060537368d-1
      3.64177359851459997789027899393833962693d0
     -1.702150756201884952339562713110595832658d1
      4.988822866020147492429380810824547393926d1
     -9.390229074747140186728173730128544404889d1
      1.133410206048303292533085990498140045895d2
     -8.481271082819884362232059784499140122134d1
      3.583229817497439267963059503278018463927d1
     -6.535190296119888545191611564807296648313d0
```

```
1d0 !BDS[21]
     0.0d0
     6.988947606828358309840668480885396142492d-3
    -6.590891155029867258051233482897796077724d-1
     1.025939122234259398692572846866224381197d1
    -6.748236415931182934303159283870771995671d1
     2.284455391592348212999259327795319239944d2
    -4.329661948626798430181217662667152780042d2
     4.646558402955043643213825052152016736523d2
    -2.636518047729024325954547830257352041419d2
     6.126143110887229453098133009666776112858d1
     1.302621768361891848879282509034938155046d-1
1d0 !BDS[22]
     0.0d0
    -6.313116724429102552597850384444925145699d0
     1.605211478155548962409740501074122692823d2
    -1.275915418885290275227568833957093576676d3
     4.538772502848256373255296374510005147851d3
    -7.659816963794426283608130529035189089268d3
     3.863694914906481795292283535941328743232d3
     6.768880592280923854229220900175699486633d3
    -1.247446186194859998676862461758059555801d4
     7.958008889503549087827896879285053452728d3
    -1.873370686002020358688749909062175950232d3
1d0 !BDS[23]
     0.0d0
    -1.448249168382543434381473245486894300087d1
     3.579826316879655903842735540650122826185d2
    -3.534282564575364919070848313968638719499d3
     1.877170843986862026395145965627665489066d4
    -6.018502968103759163745058170377645381506d4
     1.217038566754990203701260113399684365507d5
    -1.559225538366081352716912574309351361804d5
     1.225753308556727689850580304732935302069d5
    -5.389706981386271841248610107507886508145d4
     1.014453978503926046552282823261032880851d4
1d0 !BDS[24]
     0.0d0
     1.204352458028106517908622663324490770248d1
    -3.323812976487108132173093115603016509139d2
     3.476448979119025090396534415601130488664d3
    -1.873865350347562121666511750196251720346d4
     5.838266432098340629399848569904918587225d4
    -1.109300145117368006771427867962796768224d5
     1.304131086152184855319139245472168084693d5
    -9.249636659136914993266384651228059784497d4
     3.620758060610072309450630062813394704855d4
    -5.994430141771638436305271394551223264726d3
1d0 !BDS[25]
     0.0d0
    -9.844640633992644351585642158555361601933d0
     2.551910427757759310862843193440421272919d2
    -2.689392889613188918320222104808136902070d3
     1.527723168505147946615460231049223399164d4
    -5.203794223034557941569211135779464415252d4
     1.110379943781390155888879686404987985157d5
    -1.493706566658182312758845030683346627040d5
     1.228580988884824671460057673085443026160d5
    -5.635706059985523375966140608972837481965d4
     1.103638103181748788177520568394499668917d4
1d0 !BDS[26]
     0.0d0
    -3.134353635056974914567385229548342930884d1
     7.531563344909775765554106726225455818001d2
    -6.506718244567657811944393367772052973679d3
     2.924159052428389831078609368064531902126d4
    -7.827164054565393158400566262196458511889d4
     1.319076696009679657100003891581064111792d5
    -1.415299531349614008667068170177745959928d5
     9.390796606439758303947689869035580798439d4
    -3.514422268201564645004461906977910421510d4
     5.673495619408781825028373727855737963167d3
```

Alternative b^*/b'^* coefficients for the $\mathrm{D}^{10}_{23}\mathrm{L}_2$:

```
1d0 !BS[1]
     5.0d-1
    -6.194414399703707891667753795448556488833d0
     3.888068737332201642780515364372675777799d1
    -1.471332764886480057551845498752649167379d2
     3.606064254542895837632222363163012996773d2
    -5.865779742084336073892384232100234771692d2
     6.288605989824252421877981964372340984637d2
    -4.266841364746350138756692785515735797695d2
     1.658776867657162487483686235546183512107d2
    -2.812346913581557080045731548002043407625d1
1d0 !BS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[3]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[4]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[5]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[6]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[7]
     0.0d0
     8.426839682230416129859655012830583019407d0
    -7.238292018724187442057380500339023360419d1
     3.110731071300995263178933200574477111928d2
    -8.176623934659262412621359614898656271533d2
     1.391405651717058664549222989000247183183d3
    -1.540046213130229750847256160943916798244d3
```

```
     1.070159835721826967403198372167975071404d3
    -4.238061578658699647478780052696506426667d2
     7.291854786056794562210597569576386911634d1
1d0 !BS[8]
     0.0d0
    -7.591687248493701802011418102313758438O9d0
     1.277432896233382540015668314520966662443d2
    -7.041551215148900031161563735592827057405d2
     2.034715049222339251926562354923956941942d3
    -3.500336839191014820664098592505468339843d3
     3.715559195822903016484739054825726356 75d3
    -2.388788815234896905993612712327999079823d3
     8.525956862090271091731179702212714834266d2
    -1.294882107301934852906747714786712459406d2
1d0 !BS[9]
     0.0d0
    -1.593009047517364810816218152624812636367d0
     2.305820546724466170701375097423856213974d1
    -7.685725914838366476140354326917934504707d1
    -1.369206369665152606613406956871238229587d2
     1.20142885130384858027765852058733728694d3
    -2.656743116010926988022392255528154228651d3
     2.835877657288416949984654921996941613155d3
    -1.512066378499825274199339243537309220565d3
     3.236182669337256781826064546609881106958d2
1d0 !BS[10]
     0.0d0
     1.53860584658949934548382998541571333868d1
    -2.576395872812017927614628887499866227433d2
     1.54167667335719197785491111920165037977d3
    -4.668406163011315992709782409180521275324d3
     8.154053971125575699164822213481643353947d3
    -8.578530962621423748853745871359830002271d3
     5.35792976348943404573408036724376113466d3
    -1.82219282973096870384254483639566801082d3
     2.579262631258924945496932674658027806973d2
1d0 !BS[11]
     0.0d0
     3.776625273096118891407714933803179990984d0
    -6.392871879187006771077810812549675562184d1
     3.088201960029618512664034846863929010194d2
    -5.851844670629398555194964036437013314401d2
     1.770053457969642698988457294332532245682d2
     1.00938475885194493545796104163538140872d3
    -1.63461418705278956868254825031701917323d3
     1.023154920744437509989566865303860492714d3
    -2.384352495698829085079742672619430389238d2
1d0 !BS[12]
     0.0d0
     1.700040195252509269794672148550011782467d1
    -2.954731898497469399708768115264691855548d2
     1.968611866786279629994705986705272576154d3
    -6.952200215083936351831927552721607462513d3
     1.457409488286116543354920927483806561998d4
    -1.877602642722504704894001153795475023972d4
     1.460882440231151391219749814716392637649d4
    -6.303350383435997668175644554307231741115d3
     1.158628339731988960615350437555116516824d3
1d0 !BS[13]
     0.0d0
     5.295373250203029466611982388330024357403d0
    -9.162365755124775731208072650379061299657d1
     5.760366051686863159541142299365004063285d2
    -1.891791074265335989863827930221465524257d3
     3.686439181298420654487843356075270667962d3
    -4.445165942389837942338404569149243919536d3
     3.270621739060470135489296491948276116056d3
    -1.349497563045751677328487787738310586814d3
     2.397234036069196979506694411363438973439d2
1d0 !BS[14]
     0.0d0
    -1.352056625711308471328743935296391830867d0
     2.293269612261146577504615980823022067329d1
    -1.320826103304877974058593310065150017752d2
     3.771252458066940802177375524935827651776d2
    -6.010778703825499928442344864774889178686d2
     5.441802821244416062922575148674551143763d2
```

```
    -2.599904712746377104242771029238926887904d2
     4.895512707823446095907103463499749791663d1
     1.321291550209520131231495309848906444528d0
1d0 !BS[15]
     0.0d0
     2.736792354985211205343443311100999284416d-1
    -4.39964350839199827166029624781136262883d0
     1.933545213300468914147207690188720553235d1
    -2.382241671351280849009029479203678485883d1
    -4.99052309561718430244470205685876009111d1
     1.950476819494114595549270682892791806664d2
    -2.533838453745399132095992370770272279479d2
     1.529192227846299119439271790387686796111d2
    -3.606024152022399388637688372319764040656d1
1d0 !BS[16]
     0.0d0
    -1.566244949624318067934669032680678127527d0
     2.681737106248670654522355257722963230967d1
    -1.60541040805558636117797834491982556698d2
     4.887528777064235976915489037663832870683d2
    -8.588653349844424870225311751525920761523d2
     9.050050942269761110090798811116542414558d2
    -5.604548195815689979593665738851329431413d2
     1.856550700704311159204463183890018837694d2
    -2.480297274512309199866840328188079048399d1
1d0 !BS[17]
     0.0d0
     8.180296947247624935873606776035127103099d-1
    -1.369298821666974560069600610881169070959d1
     7.553398280172389042061189872013438945513d1
    -1.989364064417770172417333430003520645447d2
     2.716452158216899491923915540825830387421d2
    -1.703048614779595511532874355588901700973d2
     1.795532993422082207837960694824230472827d0
     5.150247051770655521578864897849761365564d1
    -1.836097569286092553450063848558885968431d1
1d0 !BS[18]
     0.0d0
     7.160639702202198782937631844473712156342d-1
    -1.267332776108300869292840669144477445492d1
     8.471135153842415115485018295536729483187d1
    -3.010549685563792279006821490032625284584d2
     6.391686345955590501187982704511551515366d2
    -8.401970745502397727532803296143325006304d2
     6.720489203813720551903912831939963242924d2
    -3.001896393770904581976762707009724280804d2
     5.747003975921699120223365622504608974745d1
1d0 !BS[19]
     0.0d0
     1.69821995693519537413435146752568072481d0
    -3.182800811030436999535957499167520555457d1
     1.364584457110796121326936626708286729851d2
    -1.321138578340698489535413810066155945532d2
    -4.58984803503193160765806369880857649905d2
     1.461473784388219610006147193991359057885d3
    -1.735412627948789258451805258950745167119d3
     9.733720287662129288943815576604706881418d2
    -2.146631814260907082408441809602904826048d2
1d0 !BS[20]
     0.0d0
     5.165351836605761260664106523686834324171d-3
    -5.046732638046993263879901032698121024 87d-2
     3.395221399080240231200367331831439046433d-1
    -1.507135345260158637825863357663511329241d0
     3.979817434893588935184618854366098389277d0
    -6.188780984645408578198697727281339108548d0
     5.581143701120825425062646868738755766893d0
    -2.704721831056688367472299363779820749754d0
     5.454568595836813715076928962399675026552d-1
1d0 !BS[21]
     0.0d0
    -2.167531377568441530684573811994656690376d-1
     3.807210666345361386480584713367475416085d0
    -2.492051532159352701937130014401590912279d1
     8.576539479774612878988162109644940299709d1
    -1.743703673793070097682557157146193072654d2
     2.170408031184131517495891040280392695007d2
    -1.626287593690167320776551989077893030436d2
```

```
      1.3147941962080362520977748332447533738884d5
     -6.3033503834359976681756445543072317411115d4
      1.2744911737051878566768854813106281685066d4
1d0  !BDS[13]
      0.0d0
      1.58861197506090883998359471649900730722d1
     -3.6649463020499102924832290601516245198663d2
      2.8801830258434315797705711496825020316642d3
     -1.1350746445592015939182967581328793145554d4
      2.5805074269088944581414903492526894675573d4
     -3.5561327539118703538707236553193951356629d4
      2.94355956515442312194036684275344850445d4
     -1.3494975630457516773284877877383105868814d4
      2.6369574396761166774573638524997828707782d3
1d0  !BDS[14]
      0.0d0
     -4.0561698771339254139862318058891754926602d0
      9.1730784490445863100184639232920882693318d1
     -6.6041305165243898702929665503257500887763d2
      2.2627514748401644813064253149614965910666d3
     -4.20754509267784994990964140534242242508d3
      4.3534422569955328503380601189396409150011d3
     -2.3399142414717393938184939263150341991113d3
      4.8955127078234460959071034634997497916663d2
      1.4534207052304721443546448408337970889881d1
1d0  !BDS[15]
      0.0d0
      8.2103770649556336160303299333029978532488d-1
     -1.7598574033567993086641184991245450513332d1
      9.6677260665023445707360384509436027661175d1
     -1.4293450028107685094054176875222070915299d2
     -3.4933661669320290117112914398011320637777d2
      1.5603814555952916764394165463142334453331d3
     -2.2804546083708592188863931336932450515331d3
      1.5291922278462991194392717903876867961111d3
     -3.9666265672246393275014572095517404447222d2
1d0  !BDS[16]
      0.0d0
     -4.6987348488729542038040070980420343825583d0
      1.0726948424994682618089421030891852923887d2
     -8.0270520402779318058898917245991278349902d2
      2.9325172662385415861492934225982997224409d3
     -6.0120573448910974091577182260681445330666d3
      7.2400407538158088880726390488932339316646d3
     -5.0440933762341209816342991649661964882272d3
      1.8565507007043111592044631838900188376694d3
     -2.7283270019635401198535243610068869532239d2
1d0  !BDS[17]
      0.0d0
      2.4540890841742874807620820328105381309299d0
     -5.4771952866678982402784024435246762838839d1
      3.7766991400861945210305949360067194727756d2
     -1.1936184386506621034504000580021123872688d3
      1.9015165107518296443467408785780812711195d3
     -1.3624388918236764092262994844711213607779d3
      1.6159796940798739870541646253418074255444d1
      5.1502470517706555215788648978497613655664d2
     -2.0197073262147018087950702334147745652775d2
1d0  !BDS[18]
      0.0d0
      2.1481919106606596348812895533421136469902d0
     -5.0693311044332034771713626765779097819668d1
      4.2355675769212075577425091477683647415993d2
     -1.80632981133827536740409289401957517075d3
      4.4741804421689133508315878931580860607756d3
     -6.7215765964019181820262426369146600050043d3
      6.0484402834323484967135215487459669186632d3
     -3.0018963937709045819767627070097242808804d3
      6.32170437351386903224570218475506987222d2
1d0  !BDS[19]
      0.0d0
      5.0946598708055861224030544025770421744332d0
     -1.2731203244121747998143829996670082221882d2
      6.8229222855539806066346831335414336492259d2
     -7.9268314700441909372124828603969356731196d2
     -3.2128936245223521253606445891660035493335d3
      1.1691790275105756880049177551930872463308d4
     -1.5618713651539103326066247330556706504407d4
      9.7337202876621292889438155766047068814418d3
     -2.3612949956869977906492859905631953086653d3
1d0  !BDS[20]
      0.0d0
      1.5496055509817283781992319571060502972251d-2
     -2.0186930552187973055519604130792484099948d-1
      1.6976106995401201156001836659157195232216d0
     -9.0428120715609518269551801459810679754448d0
      2.7858722044255122546292331980562688724944d1
     -4.9510247877163268625589581818250712868839d1
      5.0230293310087428825563821818648801902003d1
     -2.7047218310566883674722993637798207497754d1
      6.0000254554204950865846218586396425292007d0
1d0  !BDS[21]
      0.0d0
     -6.5025941327053245920537214359839700711129d-1
      1.5228842665381445545922338853469901664334d1
     -1.2460257660796763509685650072007954561339d2
      5.1459236878647677273928972657869641798825d2
     -1.2205925716551490683777900100023351508857d3
      1.7363264249473052139967128322243141560005d3
     -1.4636588343211505886988967901701037273392d3
      6.7396247278489789665569210027809863837765d2
     -1.3060586718652350430486832489846229315776d2
1d0  !BDS[22]
      0.0d0
     -6.47423788210741064606965337804683430817d1
      1.4771081634765029991721995105312655337722d3
     -1.1392442590839490066551053451266662363397d4
      4.3438940028550187592192652608003501986634d4
     -9.4116523929720478852085943785106486929901d4
      1.2183534905241057758554398601315674108808d5
     -9.3495698401494557775032936137335275026887d4
      3.9289582653158372497600922739733626788807d4
     -6.9715725967200398743791309639362427260023d3
1d0  !BDS[23]
      0.0d0
     -3.99044954519990237542462573463348373245d1
      9.2470402959865523613147804862999090834465d2
     -7.4920943049595001038739618798454588662286d3
      3.0736868422006669071907935285041748996683d4
     -7.2967197989729961928781609417953502863392d4
      1.0485786037899402304440863640804542268827d5
     -9.0112233592235768081584443067700069108803d4
      4.2634209482558375545887133155302632177721d4
     -8.5422119307804937603409222741744290895585d3
```

B.3.3 Continuous extension $\mathbf{D}_{29}^{12}\mathbf{LM}$

```
RKND12LM explicit Runge-Kutta-Nystrom triple
                          (approximate assumed form)

[p] 10'th order

-----------------------cut------------------------

17 12 11

  !!!!!   Discrete coefficients removed   !!!!!

1d0 !A[18,1:17]
      1.21278685171854149768890395495428880123 2d-2
```

```
1d0 !A[18,1:17]
      1.2127868517185414976889039549542888801232d-2
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      8.6297462515688744436379227441116246974 68d-2
```

```
    -4.2493550315493535129202880934825875O8354d-1
     6.426397974895866541948869566413596077118d-2
    -2.793443590382263872228973679823522371725d-2
     5.770587697190620336522752278781278604203d-3
     2.758090856067979828266414641822697616795d-3
    -4.18964567821703999d-9
    -3.025216703766429235304858652844902427969d-3
     7.843337773637672103264112915695801683704d-5
    -3.20323082097975094640522666860607696925d-2
     4.24753105852294589981577075012249579525d-3
     9.32674130959239790140884946232728939111d-4
    -1.942495107586715595288656487115047056998d-3
1d0 !A[25,1:24]
    -6.180039431011914147221951117324578452895d-2
     0.0d0
     4.72451471127345865216873885374659495259d-3
     3.023227888492508425997566078922414685802d-3
    -2.260296588204758577649613256996012639461d0
     2.342497770973836208228631033160786637338d0
     4.562654151367882825054300994885762938262d-1
     9.119239015216511306757722682869205272959d-1
    -1.595081072632575155212612271141625223512d0
     6.831124225675848694409245105923335964085d-1
    -1.451041356082513053683188944078859362937d-1
     1.292822189688766151744832406490069373812d-1
     4.191767328271343748719451848220955509013d-3
     3.209769505304155335348779755766886983615d-4
    -3.080447172443453080892761709361476289813d-4
     5.212158115095087493294643764612209620178d-5
     0.0d0
     4.285407141744189273112046329042606488865d-5
    -1.063093759235891575231740340171277445257d-3
    -1.284600469459047298232188848303106867675d-1
     1.656733187612957027599659114613724526718d-2
    -2.445393055664507821155774140806333743855d-2
    -1.931224003172288384163698760679420644802d-2
    -3.19495038280906678d-3
1d0 !A[26,1:25]
    -8.864867963819832785173041260543380550911d-2
     0.0d0
     5.860768733487394109549906889947748663029d-4
     2.279115539327650550731314400061413101774d-3
    -1.780799429951433747233641420402315660089d0
     7.319045194889397121438525492631529062499d-1
     4.773168760429536920938099590315187743248d-1
     2.159880703478402292357117812179185232851d0
    -2.188374777608872733098535241071706015394d0
     7.550245240337726443184496359599907981758d-1
    -1.486774116068280950140292873359174507155d-1
     1.202343705789328904979401542264330378159d-1
     3.786358802019835525234329022890532679664d-2
     3.755753838047655168466229339331749066075d-3
     2.234840379827494325773598271172631570151d-4
    -1.514836591520388596130421762967203536345d-3
    -3.82405299333007749d-9
     1.420873253938196073825972354681423891184d-3
     1.129253868708882560542161538903855889465d-3
     2.321687301886334286949748715854568456401d-1
     1.516417179554158897034906973208717120735d-2
     8.228757678175567195951703763052382214032d-3
     2.904925919921182297257937108373336449632d-2
     3.83916326502290878d-2
    -1.51069904665320862d-2
1d0 !A[27,1:26]
     1.011372468416803536191858142480073396314d-2
     0.0d0
    -6.97491178132676255470994985728096787164d-4
    -4.799803768560151799918952200504754414921d-6
     2.811245741490942765030180291178116869061d-2
    -9.520715207186878855162791775059920261189d-2
     6.421237987652670719495126512533370448008d-2
     1.023436398288024258657769552097370139488d-1
    -7.67353994661099280130243893600797746322d-2
     6.963761656466230688285815379714646529314d-2
     8.285129014895624885537995266360581579861d-3
     1.039682689610388993236849228774358166967d-2
     7.079480803010296767626049365393720665033d-3
```

```
    -1.138178411482309940371032231091538908804d-3
    -1.05690103515750872187967612801192674793d-3
    -2.35161458943361389878876376824812584355d-4
    -6.48881892547108701d-10
     4.010580875068877107485771081087721396842d-4
    -2.225210958377309128279169124405097642485d-4
     4.402029735938633752040675531038004055577d-2
     2.872667015086228145735340605150965316511d-2
    -5.811981789898903089070223689478401248917d-3
     1.429371186846005700708052522160969988899d-2
     2.83185007720329236d-2
    -6.27348423633716096d-3
     3.10471184313300787d-3
1d0 !A[28,1:27]
     4.541428643689302548529675301737094763657d-3
     0.0d0
     6.102781294890493316008397597057076479917d-4
    -7.214462112156221812785153309070682497051d-4
     1.738404023499353422903313586041088481061d-1
    -3.371365674738386524575727220479525777921d-1
     1.378771268184707172335213648217091710791d-2
     2.048930676722808282091207785063916152661d-1
    -3.907875543156977516150742818398254637614d-2
     5.541731748037433305310028656097592959871d-3
     2.11275174461586053246863474417995813254d-3
    -2.443923285433490686893907629633121576498d-3
     3.761774733732104753833626124186954101466d-4
    -3.147498704506869942646192971589779757661d-4
    -2.333180956957701296234451483682730228842d-4
    -7.356659832596039850980914688278109250955d-5
     2.22159332352890566d-10
     1.255241033842680158898078375654853863561d-4
     6.981985335042423290232433500794117785179d-4
     7.292163014407424698329336893817496855351d-2
    -1.876877861532336710225942138428683026579d-2
     5.389409159029107798327190699932893926385d-3
     2.490246591620423717102291036916829312278d-3
    -6.71752111384813552d-3
    -1.95757148422278336d-4
     5.06719588838740122d-4
     2.01906769029932003d-6
1d0 !A[29,1:28]
     2.457421541739763093430145942258357174823d-2
     0.0d0
     2.358248536617319680768889588657715370841d-3
    -4.245090342284993751200438959780571265854d-6
    -2.117140114520939669339936024989232952886d-1
     8.956813145644702910372966660127736989781d-1
     1.521562336223165947201354131684526889443d-2
    -8.460046888220142024448446164745618967862d-1
     3.969133916218172149918338969762794699121d-1
     7.978465730792510299495505929813308343914d-2
    -4.752806321634756813057336257029390748778d-2
     1.257502283990385187308151691123130642577d-1
     2.988979381025679428605949803693841963284d-2
     1.025076281602443464192064678589437605848d-2
     4.307841107797709692357831756230775532529d-3
    -2.106822236583315611716538606207468205245d-4
     3.49920365205021554d-9
    -3.91241204166168017253886692979506942545d-4
     5.379786548891833430816670994730599120493d-4
     2.052544583208764098824825598724544203341d-2
    -3.869645487211653352770559348697564603855d-2
     1.106691017754533253122619893562824158909d-2
    -1.614890905273510622949371727605506942345d-2
    -2.61555751165794999d-2
     3.31459125036690124d-3
    -4.3761192704223845d-3
    -1.25548785740757811d-3
     3.9309413359695948d-2
1d0 !BS[1]
     5.0d-1
    -1.370517672861708167882492409972426963075d1
     1.500796194595806193411811608833235459324d2
    -8.839344204844318027344051346171086497161d2
     3.170912733135655820847435469515327596812d3
    -7.336979824616248786161804395547324141071d3
```

```
     1.11970098289684220842655383726618212403d4
    -1.121220707994091524435822823829587892204d4
     7.092616395443334430518827878651557706323d3
    -2.571844157844682020215158620832372904248d3
     4.075642104764191655904153207199283402258d2
     0.0d0
1d0 !BS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[3]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[4]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[5]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[6]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[7]
     0.0d0
    -1.016799251551491758399230704243955166307d-1
     4.036953716345690117180366768167741477803d1
    -3.983280076105247222872790361000329397676d2
     1.831791786416903074651026077697505295226d3
    -4.924756276589732839250586189805449252027d3
     8.326308125373705291990795011676636217609d3
    -8.994673425038669979436033435224504281192d3
     6.039755487816909773169406636553248579172d3
    -2.300492932157009614278416002979433761925d3
     3.802136820126329521895595727982182398897d2
     0.0d0
1d0 !BS[8]
     0.0d0
     7.278146303421068778324265590912576549379d0
    -2.460548063595252786541003683196594549555d2
     1.920520731771808419167007099905947402668d3
    -7.021629636655237277100028300893122890606d3
     1.434871212982388664555590624326421851637d4
    -1.709220613222874442806866582745777631133d4
     1.125494105954328478850133970235070628872d4
    -3.064481611170419728272334462671308630176d3
    -4.188133650195169025713157338631705419744d2
     3.119860309491614073832997255425693633114d2
     0.0d0
1d0 !BS[9]
     0.0d0
     1.360742893738307874926567793956372512293d0
     7.718825458307788110215190170321657690172d1
    -8.62617751833648269282072637399229146607d2
     3.728463844289185861024982353960247855493d3
    -8.641045929917227064130580427249187197954d3
     1.175688406835691113936392289617241678618d4
    -9.441000315456126210685932956329980422178d3
     4.159778928164037397465519677137439222647d3
    -7.831808586616657784454585376523405488342d2
     3.971598901784053409182853908574644904425d0
     0.0d0
1d0 !BS[10]
     0.0d0
    -3.137144646570387901791094682874810373966d0
    -3.343107630378151294328074750197970690455d1
     4.377714566066372130064791284603562045049d2
    -1.578369329601433004933105779666009437461d3
     2.369760288933237859103380605732185729522d3
    -6.289244889521538783464707934540852316204d2
    -2.774986773305095043247307639742309688867d3
     3.942308815854768879768865405861108524621d3
    -2.189128644389974066507683890014008413274d3
     4.583400827234429145917240665686257011539d2
     0.0d0
1d0 !BS[11]
     0.0d0
     2.923249471801556079766318684469445536634d0
    -2.629582388760744973985055364766829781948d1
     1.213190418379013625869623823233637233764d2
    -3.862388661558512798019164444058233372676d2
     8.675712101990375767414003810240672905529d2
    -1.320600339348901823932799576184065155376d3
     1.307171301829209842391847284359971664271d3
    -7.976089053300206507769200470663128552509d2
     2.702987034164977711400422583064592971016d2
    -3.856034784014461960514419674993086732724d1
     0.0d0
1d0 !BS[12]
     0.0d0
     2.237928847211700487718929303569657844991d-1
    -4.085768653251332555623531610263734269921d1
     4.554132818258094370680653018407242459161d2
    -2.221362190193450059727826005696148106148d3
     6.044856475007373157398351730369091738794d3
    -9.985323165727029504465345053367205244198d3
     1.026057006868487778482069360907159310014d4
    -6.421099128635292109726391269974276610365d3
     2.243695224740117415090196272665611307169d3
    -3.360069940058689448140310504992874760359d2
     0.0d0
1d0 !BS[13]
     0.0d0
    -2.810420932854235598112099544321887756353d0
    -2.867416414237734572657640661929862726709d1
     4.522052067686852283940000059934905394062d2
    -2.227239961456628088057679252809706658217d3
     5.809522501247033768534771882369461360203d3
    -9.101871602493408048980158368628176811074d3
     8.881430820809542395006701101682823574434d3
    -5.311108597746584626893349548573745084689d3
     1.790793580756895969842941381496795818312d3
    -2.622092976777785500168042074954117549078d2
     0.0d0
1d0 !BS[14]
     0.0d0
    -1.428425791791166061550737967031552141681d0
    -1.66473925652259402165930295754758687154d1
     2.624081369993537937359494469591843153846d2
    -1.348747573395107425289764567672706642336d3
     3.731201931230362212401393991695086093522d3
    -6.280360326014556752869777944090165029883d3
     6.656515694503005908047196849676752620454d3
    -4.361339370211156882290986298046075951735d3
     1.620284929186829622934386484430771780974d3
    -2.618759698729090461606101026394182611993d2
     0.0d0
1d0 !BS[15]
     0.0d0
    -1.87798099530906751756181980568988397726d0
    -3.171387481893718251699192914969501249189d1
     4.565641922959560182652094558075844582028d2
    -2.337934758148522159346809609803383151529d3
     6.543350073609224643448036633942257491618d3
    -1.121292126023344155730362308215388183167d4
     1.213837798747372193130713137421773791045d4
    -8.134965845209501684086747898966916776079d3
     3.091975301858709872676236545796762927867d3
    -5.108491778021967900461927337323915834081d2
     0.0d0
1d0 !BS[16]
     0.0d0
     3.481816805542508280399759247464568168116d-2
     5.563645832083295307047266034481606016092d0
    -6.443587358847035087834293412886862016727d1
     3.212740136898439249888284284477507359806d2
    -9.085921390894086299192674336667077754062d2
     1.592963365941501978646708861327181578648d3
    -1.771843829888482431708830564806543572031d3
     1.220780994658343502065025241846675418081d3
    -4.763296099695106780173680636389427780617d2
     8.058461424604396443339520099249876125849d1
     0.0d0
1d0 !BS[17]
     0.0d0
    -5.869589614724729014877711399652900021449d-1
    -1.113925187365054466226344948155930808433d1
     1.557670919632051736315790325454967871173d2
    -7.926496743173436786743729640752990381274d2
     2.212898621941627806057425194129999394261d3
    -3.787055252123957402726497690053492427603d3
     4.095736076726904284995866907489567669913d3
    -2.742472178245175690353495399251278277843d3
     1.041364072901411576374936958343305078692d3
    -1.718625480115490517416908185067745883235d2
     0.0d0
1d0 !BS[18]
     0.0d0
     1.341310675588159552161047497906726822383d-1
     7.855565607751146897594930685555986492061d0
    -9.883833479859772610183650656870528672549d1
     5.148652947598080431349221766359396044648d2
    -1.51255937263391526968491275445452684988d3
     2.753464158409006356836177030009008313891d3
    -3.182022096079372517870826102488326446071d3
     2.279216118047514223038839830171883069594d3
    -9.248008477620626662597276027094503729261d2
     1.626853833823095940545528939688313084781d2
     0.0d0
1d0 !BS[19]
     0.0d0
    -5.111104346277360096693800314820986684076d-2
     1.87259310752138935618646687678727073661 2d0
```

```
    -1.688068031157989964474015801614691226128d1
     7.497512630626466431198229695913203017654d1
    -1.960318689125053647213224116596231616854d2
     3.221735019820489743405070000061632095845d2
    -3.375703207305805147036768639662437777913d2
     2.192634219277702208026102679733803721414d2
    -8.055728615500281204910785440229451314128d1
     1.280662382952611590852819423199369210706d1
     0.0d0
1d0 !BS[20]
     0.0d0
    -4.792409579596613487556293822965084767775d0
     1.742780385741226399912062328328855362342d2
    -1.434581514937301339018186198214480832728d3
     5.761059723166900067628948548146630285747d3
    -1.355520317972014786587122705986647462647d4
     1.993852311338488921725383824314308098667d4
    -1.855103724892345987866301481188265094603d4
     1.057706249463239426192421957406480442098d4
    -3.354470154250129058706643505356421230524d3
     4.491611376523285689484152709555914908945d2
     0.0d0
1d0 !BS[21]
     0.0d0
    -1.517834773269600032239384348197841169116d0
     6.422665738473048937324410153575073756034d0
     1.20204244683510043626722073593034687253d2
    -1.094005309562222615007762699493569852709d3
     4.028995197985435035058920650118839989751d3
    -8.24451832019953898792628577409004928106d3
     1.013558185424223496745348066272955451925d4
    -7.468989705143732785521122545651928193681d3
     3.04921247007214883854342380859654629103d3
    -5.313852630430379451324612016078053924293d2
     0.0d0
1d0 !BS[22]
     0.0d0
     1.466305606392288949569584806984621686054d1
    -2.116072455408381072512395536300595685781d2
     1.386358152214774014892001153790966055374d3
    -5.261963971107120240369072061504543702884d3
     1.261017760787978418188454139871066162767d4
    -1.971487546180925179179516480275586598982d4
     2.009782061733296976897093675250741024996d4
    -1.289291393300244810099352176400857859412d4
     4.729147042734775343351263576867442685809d3
    -7.568058647665679581854405480472789802834d2
     0.0d0
1d0 !BS[23]
     0.0d0
    -5.504961670189401067972353810563410756234d0
     2.543970592132382077188365849787088525012d1
    -1.317513437877883323595088046940917795137d1
    -5.538594071619037899375923115874947357932d1
    -9.605980915259896907814458772180703723539d1
     6.687031229876284759294128395907253219391d2
    -1.09395110173548626421614785701655226 86d3
     8.235664458336402675490316672021080874094d2
    -2.892244400386326507256547165579607686571d2
     3.559211294928393306730146144433784218096d1
     0.0d0
1d0 !BS[24]
     0.0d0
    -1.658899757925491143586178066291796686226d0
     1.525145380086908911326928072397846508608d1
     4.251967004363009519254110340480127581644d1
    -7.762763183637200887664098780421375682844d2
     3.283295416817722387202184807374456387451d3
    -7.16683987530920854857544703006669871167d3
     9.165042507662443853011803047990922477547d3
    -6.943131049908751621154227488070794646721d3
     2.894955681269680515200702816787349698703d3
    -5.131585862547401900808304820355855812422d2
     0.0d0
1d0 !BS[25]
     0.0d0
     3.51464159030875666480626641234411538798 4d0
     -2.3866603083741211286603127569404558887962d1
      1.1510506156873278067277212008068493322774d2
     -6.1079904594094399287872928929242317070067d2
      2.3477976662250936104233872623077879591 99d3
     -5.4378055944771517484619766854998510735 77d3
      7.4984912189367031586681794677928771207 43d3
     -6.0600995669337249143219641146665010353 3d3
      2.6553194188621646672428997744487191375 82d3
     -4.8765719674744110672277167401423342669 73d2
      0.0d0
1d0  !BS[26]
      0.0d0
      6.9200825764438305997525883939463698516 08d-1
      5.2163908787553923782677810524411646151 86d1
     -6.4082481506266368549776759809765949226 24d2
      3.2929211311111571031985672470246120874 47d3
     -9.5355302839735202926690654145163825661 06d3
      1.7066911830308151340209648035163305841 69d4
     -1.9335178556638786215820875434228134825 5d4
      1.3537369934124526724295915078648330089 47d4
     -5.3544925255452263571306990318974672196 24d3
      9.1596736863116307657162404853958980174 05d2
      0.0d0
1d0  !BS[27]
      0.0d0
     -3.5834164742550752004088423561136552217d-2
      6.5484812715604092696778169067247554101 96d1
     -7.3939893915314490291466205624001525437 75d2
      3.6412233398418012142400967946964804093 2d3
     -1.0116062936641391100117796063854845693 41d4
      1.7277734114869884097582509841594209117 77d4
     -1.8566622483139445505612249836324276255 8d4
      1.2270835025095784911688460807914414598 07d4
     -4.5675278362667569461253660851833143118 69d3
      7.3437073684240668931423251675366097275 27d2
      0.0d0
1d0  !BS[28]
      0.0d0
      4.4019862322932203638411284133700325537 53d0
      3.9005183037741508315619125920249553733 8d1
     -5.7331544140383316984657042026823229628 65d2
      2.3625591195277571106257360266090020549 25d3
     -4.7545021026686220028886895911031823521 98d3
      5.0777740672794509232938589227181438006 99d3
     -2.5174095328789112039188617197768815282 94d3
     -1.8773931208930157663350188096855975668 17d1
      5.3738473754761872372335141930898011601 64d2
     -1.5712408546456495200493470372459340548 05d2
      0.0d0
1d0  !BS[29]
      0.0d0
      1.9822660374903975153659577078438886630 33d0
      9.3129407790385417690104000121766312242 92d0
     -1.9982535501702887879747474458574601844 62d2
      1.0125564633684934042947106648209950742 64d3
     -2.6208153969845006993163044515926029254 74d3
      3.9948525210557445937392955737386206838 94d3
     -3.7131764439895676729331912997876342615 03d3
      2.0544297611467143597676889590196225431 38d3
     -6.1356850528668076508778165196156677419 72d2
      7.4251748890296719048680592628291158437 02d1
      0.0d0
1d0  !BDS[1]
      1.0d0
     -4.5520133719521405744144709151076952354d1
      7.2687741239700433773220444856890200668 15d2
     -6.0329574707054993645663255651013762885 89d3
      3.0151845208869025621470042275164170454 47d4
     -9.7242476456464329777105930156145770315 19d4
      2.0927705263154243729983608609017513301 22d5
     -3.0357068121952382062358647493031644465 36d5
      2.9319014488418712764988664799743871549 99d5
     -1.8072102992547094087551376639970131126 66d5
      6.4327139121154770268240277331878274583 6d4
     -1.0061381924397735945233639493769666537 61d4
1d0  !BDS[2]
      0.0d0
```

```
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BDS[3]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BDS[4]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BDS[5]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BDS[6]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0  !BDS[7]
      0.0d0
     -5.2955520201143029774266034964814438728 98d0
      2.4844102840909689886177338391008896171 85d2
     -3.0287321568908355390197070924561629564 2d3
      1.8822678159860835843976554691316842059 04d4
     -7.0233521699047131067962303157956926201 66d4
      1.6806745767927715452390417157397960681 45d5
     -2.6442153905111397795387183681171999704 65d5
      2.7236950042150951553299995924692907512 04d5
     -1.7692589200109712561282175473830978158 14d5
      6.5782467213849459064127426436265791509 58d4
     -1.0675473203302650346433239687170011955 08d4
1d0  !BDS[8]
```

```
         0.0d0
         4.2735686877558720037608253938964196758 57d1
        -2.4981016286523521050491351625099086518 32d3
```

```
    -4.3120622158623854326981810285342297206826d2
1d0 !BDS[20]
     0.0d0
    -2.7927697318383816777839292124790485789514d1
     1.6022004772673637671040010918961116728544d3
    -1.8769536857923778806101951720792389346694d4
     1.0420085541928277355109155385003052138534d5
    -3.3606416292105936757584861764715540766594d5
     6.8321408744114926603514599338931791727034d5
    -9.0306803529311418014202946802569810728024d5
     7.7397616769267196537188238698420166116524d5
    -4.1377427899571041959319748335736970003224d5
     1.2481651601265991877324453671157807021224d5
    -1.6105885277905157564513111983883886895114d4
1d0 !BDS[21]
     0.0d0
     2.9542483422561590822449026781250384625954d0
    -1.6581515403316518493634278676216447320214d2
     3.2165307333405093062321894827742453904224d3
    -2.632143779041165560203620274434701277124d4
     1.1611643607251968576876396014083105700764d5
    -3.094576044083756834823287486780335006524d5
     5.2357277621281032132272824273594503251414d5
    -5.6756628608685074894188912719330468226314d5
     3.8262490610785676124568668708054653009684d5
    -1.4627204063060881831955844157746462609714d5
     2.4249580695410537728255538637136996209184d4
1d0 !BDS[22]
     0.0d0
     5.266874656878104023940644513477085295124d1
    -1.0743279214667900581580603421843412237264d3
     9.8065913060900850909244761624277422106074d3
    -5.1635781217939214784985956511120208562634d4
     1.7225680191214553149604577281489506963354d5
    -3.7977037847583947696022128469969432657234d5
     5.6105513462313784546177451904794411041574d5
    -5.4976275570715147034442305921568818995094d5
     3.4288612880342739105304891760827013010384d5
    -1.232534461672604342065605822454200322224d5
     1.9439364098287752212315850935435275312954d4
1d0 !BDS[23]
     0.0d0
     8.4939860315008793011557503280108752635794d-1
    -5.2036823283817267859699126578535430209624d1
     1.0250646835994073353800630834172009807854d3
    -8.9994967751802838705633146354361057388944d3
     4.5799864182170525521720227904089688688824d4
    -1.4336021084226065380142409423428722016554d5
     2.816673607617421976755912635917546473814d5
    -3.4753248088184204368744859140625231284914d5
     2.6133289516350039463575562334651234128914d5
    -1.094801242291737500923925889247944245384d5
     1.959831536212487346331099482654191929444d4
1d0 !BDS[24]
     0.0d0
     1.0525320301298459366979837099373866677254d0
    -5.4826040923032672651068836919363604001524d1
     1.8958293810334050527724443827528213370794d3
    -1.8902985883167814657042213932248039403474d4
     9.1998499727220241069090252688438454003264d4
    -2.6020220857151430131503489510816208283754d5
     4.5839068684971868561471590754024109720754d5
    -5.1183843387708637337023570180511303506334d5
     3.5303270034891524353696625723141487114344d5
    -1.3745562678928259148984749763938673206194d5
     2.3135312323056408385329817495272071892284d4
1d0 !BDS[25]
     0.0d0
    -4.1526391584319303999610360095628026445434d-1
     2.5236527612775441746146098718330028632864d1
    -5.1402002451792592785575653439209977949474d2
     4.7723495909553880210482302051460279654344d3
    -2.5365994855391958408021272250280250938534d4
     8.3090280817351136053866490780738475054754d4
    -1.7215321886910822426218591354819037443684d5
     2.2526572696254713324330513244996992664034d5
    -1.8003206710487834676991297892206345011794d5
     8.0107130794243405267642481271626943442594d4
    -1.5195008574897539466592563447672571578654d4
1d0 !BDS[26]
     0.0d0
     1.3931687721001349387147947103552780294064d0
    -8.1966687174070722839691370328582715463044d1
     1.4702185848961411206368547789533728858614d3
    -1.2761979153368796505268893964027969829734d4
     6.2254161524065595880739851239685100181774d4
    -1.8508109514601776452730595730160481354914d5
     3.4860474575250799262057009420218532000434d5
    -4.1831253786291567249140429956135176112724d5
     3.100647459413528948007694127781576855444d5
    -1.2943448082703741352955813070866540368854d5
     2.3276794704918993218722045112286697016084d4
1d0 !BDS[27]
     0.0d0
     2.4024046882856222523372883733182430415214d0
    -1.4092863252501291189114451057456339368334d2
     2.5493154091506445061156808503517781160634d3
    -2.2068694998129638062036338765680969560574d4
     1.0671154438459163981796134931509640994014d5
    -3.1299188110908075195273503797540400295294d5
     5.7915470681909078707655459174800990676364d5
    -6.8024902651048858438323285029577131407524d5
     4.9211335620452174167644724951960410487544d5
    -2.0007764555654238728368136888910188934764d5
     3.4996851584723275894245531715097221391744d4
1d0 !BDS[28]
     0.0d0
     3.9103949947342018982915541302405318091464d0
    -2.2266645275799082248964972319686665269514d2
     4.1233191677668567249326201125404877675854d3
    -3.5622980996201644928648873596069555387564d4
     1.649407375280238106886882409335546222414d5
    -4.5292260933611782945403236096627785679974d5
     7.779730563251132233904714543802192565824d5
    -8.4760802584354317146384845973675487530554d5
     5.705823675652346107881717336807793201174d5
    -2.1692505363169508152414646020247710996464d5
     3.5677945279182482399003463563552336870724d4
1d0 !BDS[29]
     0.0d0
     3.4569354718458702756721067795528066159044d-1
    -2.0038968445136047493552988907438650634464d1
     3.3343689095503570940483433581787494788234d2
    -2.7573546512169778433085476219007750951824d3
     1.2561444307531572054241015245050586938464d4
    -3.4025468645670220684973803670784838728234d4
     5.7026810317012040057116486923976022902944d4
    -5.9455637568818761938797620189815132017544d4
     3.7282629717580544352267539690692165383174d4
    -1.2736029402178354232097776164072533182824d4
     1.7898623097030739866138572292661122212894d3
```

B.3.4 Continuous extension $\mathbf{D}_{30}^{12}\mathbf{L}$

```
RKND12L explicit Runge-Kutta-Nystrom triple
                        (approximate assumed form)

[p] 10'th order

-----------------------cut------------------------

17 13 11

  !!!!!   Discrete coefficients removed   !!!!!

1d0 !A[18,1:17]
     1.21278685171854149768890395495428880l232d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     8.62974625156887444363792274411162469746 8d-2
     2.52546958118714719432343449316318577617 1d-1
    -1.97418679932682303358307954885856928076 6d-1
     2.03186919078972590809261561008871301479 4d-1
    -2.07758080777149166121933554690922023025 1d-2
     1.09678048745020136250111237822578363 73d-1
     3.80651325264665057344878719104684435384 3d-2
     1.16340688043242296440927709215043233502 8d-2
     4.65802970402487868693615238454898567677 9d-3
     0.0d0
     0.0d0
1d0 !A[19,1:18]
     4.51378151784984494184551196216673866231 1d-3
     0.0d0
    -6.83248711970302151093738964546570210830 4d-4
     1.12680339535537571187187562673405241970 1d-3
     4.82836179628307882512485742511191608151 2d-2
    -1.29265765076657397694188615286719463374 6d-1
    -2.05999579860592275082501022067474486178 9d-3
     2.08995176534755283282975098935090163572 2d-1
    -1.92547849412798072943317318431390027851 4d-1
     9.31612192673515279853477786318139581841 8d-2
    -6.56189560849718248486026946352833895597 9d-2
     4.58415522343447889782543213522748342731 2d-2
    -5.71965022449433395812212330084061371232 8d-3
     2.03549700828772775450992711275723737545 2d-3
    -1.59203848022311084975117987212825805941 5d-4
    -5.05795204830082069289841562905877950414 5d-4
     5.099482243212713d-4
    -9.46317827463607056386274823826319748732 8d-5
1d0 !A[20,1:19]
     9.51660448297866138114711266539547456821 5d-3
     0.0d0
     2.27927586821289551910377559379128603513 7d-4
     1.08036645975902535105760359889327129722 4d-3
     5.58538455866674760632054682583727639556 1d-2
    -1.41512110794828087759432389599568376271 5d-1
    -1.27967334719409714021033672519953097 32d-2
     3.19445836343752377194287129994332830176 3d-1
    -3.68581464886723625374924172817763914539d-1
     2.03870360565652167701251281841355392317d-1
    -1.55375831141819176750521539872629878840 9d-1
     1.10159734204245094596354626168654538027 1d-1
    -1.47949939308409518859335476227073017371 2d-2
     6.04927336298431040036953802543185957049 4d-3
    -1.11510358940696723550121571148611882813 8d-3
     2.30539836343939682036422940926394037145d-4
    -9.697139001265911d-5
    -1.23234453891867426718041128840504109595 6d-4
     1.85084411914885391240360184113864039464 1d-3
1d0 !A[21,1:20]
    -5.89240034170502859210611374322426454357 3d-3
     0.0d0
     1.76716093097729001194947415018142424894 2d-3
     5.78447266125253936961438160375026402026 8d-3
    -5.71985507303852113174539092208835381872 8d-1
     1.32120665033611444676638825292558184464 4d0
     6.30893440577640661960166351354819182 65d-2
    -1.41464261770434629825461968510060801640 2d0
     9.33823310423283687450571034908588182108 1d-1
    -3.53943523238254390132539061782226739834 5d-1
     2.21720286453815141117228465371173044848 3d-1
    -1.53623399484249638517318945234661214140 1d-1
     1.94514268459839385549502495861026248239 3d-2
    -7.28247414586037030211876887786529545879 5d-3
     8.25573332260114259486040578908564747234 5d-4
     7.55310497599050695503238073372709351111 3d-4
    -1.059855138289816d-3
     5.24620285310450344184926924946596387689 3d-4
    -1.79437328919408410010151639388951123150 2d-2
     2.42535442413777120836413162822885112248 1d-3
1d0 !A[22,1:21]
    -8.78959390148359449949099641850054817942 8d-2
     0.0d0
     4.70725189867540211471451886097548833655 7d-2
     3.17040007829973841725256718671913928457 8d-2
    -6.21556582179509019688048662047667983387d0
     1.26432767499917611284364623876337852216 9d1
     4.51580290372897096360362629740204638032 4d-1
    -1.10595487708310425629378935907797533317 8d1
     5.69025802395929015958716156253603464934 4d0
    -1.72282432134668759842901431162179837063 8d0
     9.41545818810652639603806343725897373556 1d-1
    -6.46519147888864524356192145467638023237 9d-1
     8.34404522427194527342160580097725369161 7d-2
    -3.34611191795070769069557343383142395648 6d-2
     5.53704362561237343761522398459257394583 7d-3
    -1.48970536830007031889806237160058904161 7d-3
    -4.100514370612258d-4
     2.16030451484785773612015125008589721 08d-3
    -1.27383877110646376155733138271603454448 3d-1
     4.61333534958188704426241059774691997163d-2
     3.23901971888684593122044244177604957754 61d-2
1d0 !A[23,1:22]
     7.06972787555513215184879270676170157702 9d-2
     0.0d0
    -3.18099315208168853323900281841184361931d-3
    -1.33064905039177652519475168736061183803 7d-2
     4.26268673123153438049147356466209757867 4d0
    -7.32282162201977602077937259605125886532d0
    -3.13850518621011994009680790311359231858 3d-1
     4.66457631377976590288547871156992566854 1d0
    -1.37200526107757865063993720479632717166 3d0
     1.13175830314944880824492734466843062978d-1
     1.40537390798670952932012045065531209161 9d-2
    -8.21841949219331359677325423905475562913 1d-3
     3.06044069260912160738807342134910854940 4d-4
    -7.06264163792725932664093696293680048053 2d-5
     1.99675120047863001649392584798355150482 3d-5
     3.66629229215020775242056821636445677866 9d-4
    -1.040627735417264d-4
    -2.67668819939439555728068177362192413875 7d-4
    -4.35225985545823553865614965917496131296 5d-2
     4.79146045505004161368530617495104095284 7d-2
     2.12398732767118514557261727601280228796d-2
     1.76135813688487618131336208267808868268 4d-3
1d0 !A[24,1:23]
     3.13612545181528157774318817667174531517 2d-4
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     3.16208453147077332629987130630991111255 1d-4
    -3.06003801414389947029510396477034438009d-4
     2.01516013570782530766948253162030438455 8d-3
    -1.38814075314682284001254171598888422611 6d-3
     1.64963450349643421811127818818643167841 4d-3
    -1.3042332606777483404631472378728003739530d-3
     3.182928689741827935505599337389908164187d-4
    -2.295067788255238943144537704758876828503d-4
     1.091633561912333597013845350174978273029d-4
    -2.0506200707138006568053739535301730972 7d-4
     1.755523345823009d-4
     6.588638459999865152689092992919339061807d-7
     2.993989125107163794204232856142949674038d-4
    -1.005612963646591701661914639145852231628d-3
    -2.10199269813929015934663722561639932 71d-4
     1.204816881007965248998615638156046429278d-5
    -3.722134875404463455190039106097152007389d-5
1d0 !A[25,1:24]
     7.586206594123550075808238152647764288833d-3
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     3.912216339740297625629360928766014909908d-2
     8.926877913000929392089349530437847308448d-2
    -2.790571062452951676412247075223729660145d-2
     3.201784698238587975179892717004834086384d-2
    -2.763731089317843375350356719748552442669d-3
     4.117057881119717002173075846928810008251d-3
    -7.172368362631151556599712238418212098606d-4
     4.497496465487303150084255026094247833846d-4
    -1.954958068531693530494894832215405733041d-4
     3.194750955519716025958075962584609682034d-4
    -2.230731348752259d-4
    -4.711613141872529800648523112417042624724d-5
     2.324856655011289367167896446807636165074d-2
    -4.746849364151676163063716003370173322796d-2
    -1.582614247791182307744921087611277113492d-2
    -6.008749698243057067952591536837600655484d-4
    -3.69028293869575310944132367651613906486d-3
    -1.017073544774749d-2
1d0 !A[26,1:25]
    -5.46574079712703715732034853590975935819d-4
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
    -4.850716485328962236017559761792302752381d-3
    -1.179811235724965364928965659272013344264d-2
    -2.600573824811757589641452400255084291029d-2
     2.007582747737000932106312961612872542498d-2
     2.763516055467244032534453734432517755227d-2
    -1.182484956497704079996263789485584527134d-2
     1.21584374656859133526500655310698873765d-3
    -7.057205435854722766839013061117304546015d-4
     2.982382044805420530728398662402883662091d-4
    -4.396579058438079701271841340533318293559d-4
     4.111890566798012d-4
    -4.609920278944869563374444557319383652 63d-5
    -1.286872030873067279890730738251125269923d-2
     3.279851047916650878287708560903051591307d-2
     1.255928953240011847641477838602911383576d-2
     1.910068194983980986218158364058961732736d-3
    -3.198676776100750492111368005356036883351d-3
     1.138810217106281d-2
    -2.957438170191387d-2
1d0 !A[27,1:26]
     6.338663295275007975330437097181031478947d-3
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     6.027955779775273672945353117967619893226d-2
     1.381157168063327395028416846554982568675d-1
    -1.682615886830736734985632507547664920018d-1
     1.413979356570646785321857313766668240333d-1
    -3.281541975701784304033317235473338150269d-2
     7.625061460065439250514347920450139185369d-2
     1.404466891587110009907686342968216441831d-2
     1.239156610186734236104520780293740341044d-4
    -1.187253522639726896824600197959875157705d-4
    -8.967067668188708306270770986641338780918d-4
     5.107427693890004d-4
     4.337990601857671070092865044123209953859d-4
    -4.008729774774358567812883387930547802921d-2
     9.514455186526438241896576055763673706088d-2
     3.175191219155944866465307803317729532279d-2
    -1.03818221627486141477235562886446567008d-3
     1.182295151694194296658382794479334360042d-2
     1.724142213022374d-2
     3.403885832586257d-3
     2.614790567448408d-3
1d0 !A[28,1:27]
     1.17569497498156201470274711050251489354d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     7.798713524865281530549085132874031743263d-2
     1.869288194882097450736349526575920349426d-1
    -7.182975468233221366964185175917131016014d-2
     5.607954787659797515102553870999569244308d-2
    -7.86879665538575113987802297751631860939d-2
     6.326809608292224113852746476708441084621 5d-2
    -3.131239895545657244813095018100382301458d-3
    -5.454459621477421459073020652143535856165d-4
    -7.45786536655103551314836362588944016 98d-4
     5.733548726083891006183676361156775800811d-4
    -1.781284341097197d-4
    -1.96926424625066782741794333646569663552d-4
     1.129192346859785638387171228471319886398d-2
    -5.597142442671427473000763550848787664802d-2
    -1.590891846996134704411188132914073030996d-2
    -3.820048104476802876869342503804488858626d-3
     1.364131984917830888736896952679166297824d-2
    -1.877824427945785d-2
     7.349682711888039d-2
     4.466524984374358d-3
     2.844318233318722d-3
1d0 !A[29,1:28]
     1.076369990144876509425940537792058922 24d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     8.322745240607143496686456357033570853226d-2
     2.364441374969475791262482753501149669659d-1
    -2.288692982064068390646493515176781747087d-1
     2.266885518723383040348690838492328125158d-1
     3.955734978841540519074967214383350863546d-2
     6.961958210537126428349659591884582831772d-2
     3.23884514564188864511217567958443361711d-2
     6.881044672051692337528878969773171833711d-3
     9.909624335170792870504488083962834805541d-4
    -9.108670732047420189018582650258782243993d-5
    -7.027700583182549d-4
     6.257314724225233407161157116572209466337d-4
    -5.92636548955296055342907904368370506279d-3
     2.799697944246655899119443530366183873563d-2
     1.33857868419946971703128258115920640718d-2
     3.989201654081478762981042338652649670401d-3
    -9.545108310664937249035176472662981205321d-3
     3.154554511964843d-4
    -5.74660266317259d-2
    -1.059479903443691d-2
    -3.150631325871766d-3
    -1.496409140626227d-3
1d0 !A[30,1:29]
     1.018723089073203892949990789904371832553d-2
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     7.970389138148538061673894428046476744164d-2
     2.273833196442368513472066981656186329592d-1
    -2.072380142917854166122821293986520820567d-1
     2.071885073251880244453605147325465419813d-1
    -7.076734183260816720841123944480014128472d-3
     9.971205583740909051169617960664487699513d-2
     3.630281867384239843240472450000374500351d-2
     9.560264813645950387133777623371568668753d-3
     3.486454062573464185658072282208198613941d-3
    -3.033130171835239911135051575913455860685d-4
    -3.681630896829066d-6
     1.027448472418344532088731217752406243342d-4
    -1.190271529716130000510008140559007011553d-2
     3.627031537895124528749711662767851049421d-2
     1.046596034209157030302179598761115271088d-2
     7.311000942134160057400275402703774331344d-4
    -6.401918943530997358560970758788193648101d-4
     7.661747364005177d-3
    -1.502586123212124d-2
    -2.890721589874046d-3
     2.737052930774714d-4
     2.879733924146138d-5
     8.433700585298561d-5
1d0 !BS[1]
     5.0d-1
    -8.70748956197562793616329497204702754229d0
     7.717812196298565104099048587705659498845d1
    -4.089938521200644865817417561921917009063d2
     1.394527850858742386827155729165980240602d3
    -3.161578455667831003983567606979906894092d3
     4.808597329393688889356299156406058847367d3
    -4.844032301603280499438727812290383012537d3
     3.097899858802206192201179422391520195009d3
    -1.138334155434589855821984533120116348946d3
     1.829552212386355397515370987535786489443d2
     0.0d0
1d0 !BS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[3]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[4]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[5]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[6]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BS[7]
     0.0d0
     7.357054784200412432455176351892110464257d0
    -8.282986373022712713800460843457438755249d1
     3.823463427982595517811455664826455912862d2
    -7.926934408664454443959341888081727144036d2
     3.089844115035849984894694419950857835594d2
     1.920906825851649329923719963748172678256d3
    -4.27768968115548105350610741709539818194d3
     4.13799099851861099542501804713193840393d3
    -1.994355921984470809204795150728468524499d3
     3.900695717428348349374695485843203571455d2
     0.0d0
1d0 !BS[8]
     0.0d0
     1.008419998815230385505670587489194684301d1
    -1.741558033928230210773736985395373448801d2
     1.278727027013331643371875733172965700329d3
    -4.98260660173040985500430707187593289662d3
     1.16239592316772841183186463422620341351d4
    -1.712722312828287399183366003124091717641d4
     1.610342316773919957131237044329481039519d4
    -9.374164717629748136391636790714842519841d3
     3.076910215125100178167659891766011023778d3
    -4.347010435490940959991991805501669449239d2
     0.0d0
1d0 !BS[9]
     0.0d0
    -2.688886511290868904295855053909436836025d0
     4.462931982141830412278761032281453438783d1
    -2.629761203043428880614367971061015771034d2
     5.887679624960360457852113828976482915588d2
    -8.465639835960613213693859824898055101114d1
    -2.091344533090359619037813112018136837773d3
     4.343307336050685500298105175750822612414d3
    -4.10726693655962976257517765900597378044d3
     1.946556677058512212064647707150970636012d3
    -3.745258392813554738584481626440397481372d2
     0.0d0
1d0 !BS[10]
     0.0d0
    -8.808253270757753820543792154255991278909d0
     1.024120953027244156944846522447979433879d2
    -6.279659774648270307752557223245306391138d2
     2.620317813970753943097343026098073243976d3
    -7.254835473719793564613999604535033950414d3
     1.296763906967885716516809470145921864351d4
    -1.471167413897137312253098966935848954382d4
     1.021967725500437516388368632398335343687d4
    -3.964745065116472569901538365549154531244d3
     6.581858615055923263895277116970302594279d2
     0.0d0
1d0 !BS[11]
```

```
      0.0d0
     -1.9530769592610922371220536325386078274 84d1
```

```
      0.0d0
     -1.953076959261092237122053632538607827484d1
      2.32182758916010734148942414901088574477d2
     -1.259533935458791249165299622532589072815d3
      3.974756787452268740573935062935351457942d3
     -7.744477378552453207272585197758987008257d3
      9.295891052854118818931950017013982173649d3
     -6.492209356620751417214652166831497421463d3
      2.175992691242931727502989687997346640699d3
     -5.023071085525686303008640048010472162808d1
     -1.128619151935440770205854522746736365307d2
      0.0d0
1d0  !BS[12]
      0.0d0
      5.689690794888545931471717660974732596195d0
     -7.797560846088168268068639049284875894524d1
      4.664296339075455539170904159556970055357d2
     -1.522636040728707541877162781931168688207d3
      2.827649378841452484808661076128353083721d3
     -2.801840298364393680856313661529064603288d3
      9.492262500706462482827727671173632775023d2
      7.143999382314865302480500870015163913827d2
     -7.628885088968526377284411804207089129874d2
      2.02055242653561200090808061747709051054d2
      0.0d0
1d0  !BS[13]
      0.0d0
     -1.568040802523745748879896515223462196351d1
      1.832764137112986110414729661147385075844d2
     -9.901240512735189590094271783155419296041d2
      3.148632854276783507598661062472347963842d3
     -6.364118519196390634825149553844617078349d3
      8.352879744154078448022468412210238418481d3
     -7.019901236795149982976319542880926535989d3
      3.580315237924800757825376928818479428709d3
     -9.775650292476205560986143748098545229073d2
      1.023230596034827324160647332592808386404d2
      0.0d0
1d0  !BS[14]
      0.0d0
     -1.167676098875428951991518367303661196368d1
      1.379699179933499097263209779806799306607d2
     -7.508511099817489932416524763076303710106d2
      2.386540246604678847387887446819612809661d3
     -4.777131355564928016227571682389728170148d3
      6.143658543326756651939168693422646379619d3
     -4.990220750258633081178591653166368855573d3
      2.406871874035695025286744131316699061428d3
     -5.939529272376965162753977185740029007603d2
      4.880395614008478633265155734205023241032d1
      0.0d0
1d0  !BS[15]
      0.0d0
     -1.482858440871034799019014058704335663558d1
      1.760534377055773878502804117871741626559d2
     -9.608245836663342994446157498156103614125d2
      3.050363064611586040564639575010300243215d3
     -6.071365572522317676955804322757278426155d3
      7.722968278578798685969982196566171737057d3
     -6.158895647626180095739741271059196755494d3
      2.878108214534697941416233479317153389421d3
     -6.65980411221354682448767116305616 86976d2
      4.440646204394107165666987399633078609309d1
      0.0d0
1d0  !BS[16]
      0.0d0
      1.768971863006039727133755263866909429768d0
     -2.104705085210589221331059037611179591704d1
      1.149966180006183141881928759827019645244d2
     -3.647982750384601249082818967625182292961d2
      7.239769326979476664961816196931202505925d2
     -9.159262816304252703647061794675451557079d2
      7.237947828011710951912403577159339637167d2
     -3.328156941598426432876362454131310831916d2
      7.434564241399015823696335241584096522178d1
     -4.295646095899343065777049052157789372537d0
      0.0d0
1d0  !BS[17]
      0.0d0
     -4.829364086099193235849792309156519883026d0
      5.746124021208477691879230214151150650891d1
     -3.139857244964333910602741142320695557971d2
      9.96181330677198174479831553730961806688d2
     -1.977377981441295667711925871439253526576d3
      2.50224310346723369735963581227238679193d3
     -1.978025105552724187552893171970765412748d3
      9.100441456960528534528145127468568573886d2
     -2.035340389063303808106242105856643340414d2
      1.182239443031331816049297964519238653115d1
      0.0d0
1d0  !BS[18]
      0.0d0
     -1.77599497063078824473386983748920027 36d0
      3.834898221902898079434108066297097237553d1
     -3.366549075035943384565180426437887723515d2
      1.604443167652476869908434708102854023502d3
     -4.62504087623768776957605622126304716146d3
      8.435841572620538285826563180664731085575d3
     -9.80573301230499355112758738313312715906d3
      7.046328223166853607713172607604644428518d3
     -2.854501875118696459876660449030563625948d3
      4.987447204767051630390443888728154091233d2
      0.0d0
1d0  !BS[19]
      0.0d0
     -7.319585732189914912037318133083072809209d-1
      3.996077066819299547527407615162077161536d1
     -4.633224685736771235184426001693120942417d2
      2.464728720508200566836551336625135639364d3
     -7.396741260077507376868753927302507117991d3
      1.357679784545874805557682109713876397694d4
     -1.559935164196316678721572364314541769821d4
      1.097097041626277670858846838684097725839d4
     -4.325584569303544970264849437747042323841d3
      7.332741455931969228818584434210898952567d2
      0.0d0
1d0  !BS[20]
      0.0d0
      1.124359826071302262484911029354504589905d-1
     -7.406458161634546486677617226417174070029d0
      8.83979043559783311732001807153673137944d1
     -4.755453056712363291940971357070430018928d2
      1.435333390181262891613369176328331561802d3
     -2.643481671136232696020581995222284807774d3
      3.043863487347393014261547910396086071039d3
     -2.143877610292501186038562679920928888345d3
      8.461507371666215121173297950206861388862d2
     -1.435469097722581216517761254867326638979d2
      0.0d0
1d0  !BS[21]
      0.0d0
      3.926147802539428370892398719272827026971d-3
      1.33003945760838992921416212343450259118d1
     -1.811125023567812251113884402977107780376d2
      1.019908331576102957837527147034002202323d3
     -3.14822187504459422928601232946818291053d3
      5.873573278857454242851298779889048096708d3
     -6.81868113904570477689703221209722820 11d3
      4.829007680837951926618359771465563884996d3
     -1.913256649643678567958671180437992431284d3
      3.254785540953632332254059502794358381868d2
      0.0d0
1d0  !BS[22]
      0.0d0
     -3.753261003339108352293208756359267302277d-2
     -1.560139891003397814799660971948863295062d0
      2.527329855516594981665901374828877495176d1
     -1.495205653987170819523715188400601762 89d2
      4.720680553007309247379242955816312716272d2
     -8.918505973768313738921288545073201264921d2
      1.043435154026478094476919436227209774487d3
     -7.427781285941867865002785812003254019426d2
      2.953421232370787067164124872061485017452d2
     -5.037166724868164450481368515606016211965d1
     0.0d0
1d0 !BS[23]
     0.0d0
     6.240351648668761838220792721554806761371d0
    -7.456516521975477690728966899358879541284d1
     4.107970355103472782374224686321980865813d2
    -1.332078975736311721378511845146428935796d3
     2.704790439309051905213313431383093098725d3
    -3.461382044103809567326706568677309822916d3
     2.70599714772490685641845678900153216279d3
    -1.179984714513940912508297239006881091242d3
     2.208067284885987383249272367108436033108d2
    -6.208031077565619115353966250131128012423d-1
     0.0d0
1d0 !BS[24]
     0.0d0
     6.957768605137426841867143707227493379703d0
    -9.055643560985842415618481658861485139351d1
     6.142243006900115257463128252506468095997d2
    -2.574806942986782647714928160760357165864d3
     6.929407255907051658447920125256098419268d3
    -1.211750324278972167183581321645795114224d4
     1.36693581138802995416656221275094772052d4
    -9.588633086370498993343877564178908391114d3
     3.803271643246973514532594412532184337969d3
    -6.517193745726119301835128762698027148042d2
     0.0d0
1d0 !BS[25]
     0.0d0
     1.353035896770151751257658089001945410627d1
    -1.545137306873793337564195708256509979737d2
     8.622957611410034777097286964457274931114d2
    -3.098286998538946101829729176527572338655d3
     7.517747730620459809387546646579329830003d3
    -1.218659132453387838330491132400228383774d4
     1.2859463120684317850668768994297000449d4
    -8.430933190521796135783679411039761300619d3
     3.109091942060226485291405424941381932196d3
    -4.918036691917091858952868607581906834222d2
     0.0d0
1d0 !BS[26]
     0.0d0
    -1.009303343312636668786645613900923902567d1
     1.555342660648080775773146103394804524857d2
    -9.012035922476214691179361762407168486102d2
     2.69795895727561093330061078895683450656d3
    -4.549026787328999336063550309185341190075d3
     4.146894673183367650733217391024504756404d3
    -1.366906203093383948001002619593552422694d3
    -8.26100724686250819194912197311543211864d2
     8.777098770235369233718839577116810488214d2
    -2.247674327579416459177589895623378520029d2
     0.0d0
1d0 !BS[27]
     0.0d0
     1.7865010383880678475136293067153439354d1
    -2.271494750334234823339456033142707150338d2
     1.349595304023079952590387564138200452825d3
    -4.740013821262620623788438634912698552821d3
     1.060961873559930675528596705107048556656d4
    -1.549319038226104121951898080060317337153d4
     1.464313733431605730621960604286421132294d4
    -8.590706203804208721730592816508124418106d3
     2.825994761934502453441712500902453433916d3
    -3.951512638955330986408515967042371581116d2
     0.0d0
1d0 !BS[28]
     0.0d0
     1.100760540835460610501339430158985024439d1
    -1.161339498653059946963481491254088866946d2
     5.494536693195690879770026308635019413823d2
    -1.48859319443588312268223565251074725321d3
     2.49510037069541916759938809685990494445d3
    -2.63121643632178629861098083895348598666d3
     1.68409091987726502961645509843749460903d3
    -5.7013018630017094874361571000873510386d2
     5.27381364041677879649091395283379759604d1
     1.36830652183706854704119906075479093574d1
     0.0d0
1d0 !BS[29]
     0.0d0
     8.3856540468431612520815211915088147221d0
    -7.594956106359218249409937544057235233229d1
     2.435699514782374128947371429601079477343d2
    -6.98171554013289863752133712877674957201d1
    -1.69744175777586794681695836506177298316d3
     5.35609273441146968919003542546704576643d3
    -8.04030908251206274620683141449990162327d3
     6.71578093289722600124843365791111417152d3
    -3.00175789287734174987954140907499279681d3
     5.61446176796417347187356187835230550886d2
     0.0d0
1d0 !BS[30]
     0.0d0
     1.038600741120287514867208557210570244399d1
    -1.54464477185573881928003459428734053539d2
     1.071441978654587374140233561829744619347d3
    -4.35572977016458943309657738477863498046d3
     1.12033777591557201819404862870971690228d4
    -1.87424341119454058382466582446034964834d4
     2.03345324829844651411743348345103109806d4
    -1.37959962737228903853122601502180083577d4
     5.31776927168459794906952562097774324684d3
    -8.88882866872113982889753150958199696933d2
     0.0d0
1d0 !BDS[1]
     1.0d0
    -2.928802336416062207284797696678639985855d1
     3.90624440738555419446284318815926641441d2
    -2.943836845906944535371434477278400947467d3
     1.383633117847306654857158806014741303015d4
    -4.266385274517636475029160136821579380027d4
     8.850504634875820273636358108407845802208d4
    -1.242674778583728403349252716434415137474d5
     1.16427397616936574402527893410280677695d5
    -6.970438531880273224022375722491824155932d4
     2.411715626691518154615862654765741282363d4
    -3.668702932330020984768083841119602215053d3
1d0 !BDS[2]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BDS[3]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
1d0 !BDS[4]
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
     0.0d0
```

```
      0.0d0
      0.0d0
      0.0d0
1d0 !BDS[5]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0 !BDS[6]
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
      0.0d0
1d0 !BDS[7]
      0.0d0
     -6.210777654173512756957792507756106584847d0
      2.30161489198019104515773267636799023787d2
     -3.11549681519668101702283145955668458958d3
      2.149808914657865014540131541916698385563d4
     -8.606009546214736934979030073639630493016d4
      2.145050670587327435702398136493941037305d5
     -3.441514871375579827002100770726227134527d5
      3.562638884679694620097436835502326601636d5
     -2.302590287735699942745472615618898831324d5
      8.459667311517721636295098608911542945482d4
     -1.350146947209566329774052611128059073679d4
1d0 !BDS[8]
      0.0d0
      1.460248442271043291883978469521098941929d1
     -4.906894796931013859288195400231996862301d2
      5.445815270970202874482097735425222965383d3
     -2.90832693525822968638412678160182891 39d4
      8.996849919254509956772924318442485425496d4
     -1.754709343804378636735788734407132091016d5
      2.23382388782271670725356742448569163 47d5
     -1.857000568180860227374270659856409930924d5
      9.728194403583377316552343476054224741139d4
     -2.915926402188039614267917726769857713671d4
      3.811279970333872430844136565749214463047d3
1d0 !BDS[9]
      0.0d0
      3.536587817384770486880597053741611922716d0
     -1.521587994003287561021870380563936024434d2
      2.558169037163962565507223468571114451652d3
     -2.115163270149489544451496995163681915585d4
      9.529663568612467697293414894189751663827d4
     -2.564782621736478196540044187822033702061d5
      4.33261565694848945115457063376466679744d5
     -4.649076640040831670304325631829771414929d5
      3.083445580658381749208608835362934686599d5
     -1.15473908916628837009090984412309709546d5
      1.869889829855532663916111236962773175515d4
1d0 !BDS[10]
      0.0d0
     -8.630255250030792215052244073845518657446d0
      3.398670157379099718070873788139631071541d2
     -5.070237433353022845060148179897343012778d3
      3.819027676704968987832060870332280647011d4
     -1.619723109306090337992591157115429111164d5
      4.18427601902465672159735402326848617622d5
     -6.862066447009861815119111594494120010168d5
      7.19903708124270828311685739316464439636 5d5
     -4.690199928968823221777460651901937313723d5
      1.731124151691016207597154601190865722078d5
     -2.769574798116651149618654317707505369968d4
1d0 !BDS[11]
      0.0d0
      1.139393102791684021894224190065682965576d1
     -4.004873582907174561131451637900313143447d2
      4.923039671536676022168205338212545158522d3
     -3.086403890519935093242085629776915938631d4
      1.16761896274160554028026777711114955567d5
     -2.851730301611638133048672830106973596401d5
      4.591815086833407311709635816902765255163d5
     -4.835024781082529932207574467106635774128d5
      3.199236162650463512380759537254512010694d5
     -1.206181565187954332471276024999644974114d5
      1.975669467497392343199964858921780283973d4
1d0 !BDS[12]
      0.0d0
     -1.236082374653669059183836943404592138107d1
      4.529539728516296462533827926480818959012d2
     -6.030483160809866053183871791167777514243d3
      4.139598506692728701537327017097862226252d4
     -1.683879121665276751521795417360276156126d5
      4.323733923872493105504830392621053801057d5
     -7.188174501135542314680566376653967506466d5
      7.717639255641460927191033966423084476289d5
     -5.165007960038143803014896203324926780661d5
      1.959620804394316414297324302017066050222d5
     -3.219908838654359540013744642494316835289d4
1d0 !BDS[13]
      0.0d0
     -3.817803335721419751864419826987221420118d0
      1.480284568299251028900642429804861292656d2
     -2.163721074081366290138197308487935426891d3
      1.618231020740904476165774846559973513086d4
     -6.945515663473409015722559123144607843125d4
      1.83529939931345114197757194575149515819d5
     -3.091906706339338663792045863789643484177d5
      3.334636235066359266230949327112223130436d5
     -2.231113806448248526549160578474616876964d5
      8.44171596461931365652178982280452584601d4
     -1.381616269697314448335860308532262951544d4
1d0 !BDS[14]
      0.0d0
     -1.915996573034651263272744028125131501942d0
      7.49179786524500281551358928319904709099d1
     -1.109025803535957016578437592579194326037d3
      8.378316464449694525544158999211648937902d3
     -3.614520761157343856508293965706914498366d4
      9.569454495936166998254941421687313262836d4
     -1.612450737799178638642944026152404380253d5
      1.737848452055598572876025315732762429824d5
     -1.161493066883351916279580485988283105073d5
      4.389276034907220227638167574054160641206d4
     -7.17477363867875810544830656559295792748d3
1d0 !BDS[15]
      0.0d0
     -1.918058216378363386625592707344730755382d0
      7.539739231180957163476637872818228356214d1
     -1.12495474608758748270927561270093295275d3
      8.552471086849828843535757118374234120362d3
     -3.702368314599403953182000421129830285593d4
      9.817507459954543205904147506311982163076d4
     -1.655154953252298017868217329750012175428d5
      1.783885887578681632442615740105826646285d5
     -1.191943461781921560225701074103675222683d5
      4.502559798258876869878594785468251862134d4
     -7.356647339732100321838973821579412827115d3
1d0 !BDS[16]
      0.0d0
      2.004574885026416238132690717603861067088d-1
     -7.907488429808646212060029370770544890996d0
      1.185962659853506607670106345283143448406d2
     -9.054510090141300581640260239003798139998d2
      3.929473844982129127224930637276602215388d3
     -1.043379408122245429825369682987940824545d4
     1.7602841223742929061785808713989936802 95d4
```

```
     1.760284122374292906178580871398993680295d4
    -1.897823580092698817578939993063321743494d4
     1.268236795886564926836604796300904689244d4
    -4.790826179778757943460203691675447937253d3
     7.827256531179482841489021465810498212654d2
1d0 !BDS[17]
     0.0d0
    -5.531532664265836164038894379284774393016d-1
     2.179869797540918547738407207694699324133d1
    -3.264482295341340750033339513688315671069d2
     2.489101537275013800813991151932342093522d3
    -1.079240308632172519824804522413440042398d4
     2.863869934749213404591070000879347528208d4
    -4.829462765960035656507019544551584915514d4
     5.205082403896749455201255368321833095705d4
    -3.477459499409812260313762381051537134575d4
     1.313366861639842793164945544738919229178d4
    -2.145440115287714490788482042437906648266d3
1d0 !BDS[18]
     0.0d0
     2.741764565848349077831621300257226698426d0
    -1.030861325278360278324584964560204000007d2
     1.441156699711799914612520294647072583803d3
    -1.053486846241167517019794172896808452206d4
     4.596501974506273323442220313532600000979d4
    -1.272733971347866781236338059758617164403d5
     2.291851372943908471096186914163681866766d5
    -2.674019371795407375884810252994208279607d5
     1.948586708869121867343979259512781956211d5
    -8.05610290229291902959924906017124496067d4
     1.442159154155270186400854968349938681176d4
1d0 !BDS[19]
     0.0d0
    -1.483685848787950568854902504150173689443d1
     5.29548959086336017348800990984543705049d2
    -6.686471759143475426900651004293280979756d3
     4.270152453483806435969232745665997000662d4
    -1.601154282756679095708668641149992384058d5
     3.788549458236586351312772519842776021603d5
    -5.833584807048888330120267565276236310976d5
     5.844337982835183518664771649326069363197d5
    -3.678764498713487280594139027211375280244d5
     1.322667985297951381344234586928367729282d5
    -2.073494866135969993432228066427064487556d4
1d0 !BDS[20]
     0.0d0
     3.001101659958948748539415287401326077035d0
    -1.071136630063437621009998641283473338992d2
     1.352495308358371682219240117100042879611d3
    -8.637382116232329948098639272003862171533d3
     3.238709043263386974177733704112507382695d4
    -7.663227412486683553137057217847196954438d4
     1.179978973463125697892133195317269594776d5
    -1.182154054780268671588289684419006543589d5
     7.441161653395934459263227919320320522399d4
    -2.675405369334056710908193395049507561332d4
     4.194128352548828754890398408557226287818d3
1d0 !BDS[21]
     0.0d0
    -7.615313275773598558056023528000723027851d0
     2.718015556727614585372936571842341562019d2
    -3.43196487296065179022147331988791026603d3
     2.191740838881587857692413311487612772208d4
    -8.21824342127385647076025624420425470261d4
     1.944548504577491226860706741474610475756d5
    -2.99420364915940891168776521650561581525d5
     2.999722930912281150562549519494866137494d5
    -1.888199186396725055960633848598260679715d5
     6.788857005078695265577777646398095297115d4
    -1.064262558966444357234283103714286866284d4
1d0 !BDS[22]
     0.0d0
     1.294187475829822940147312581082380053234d0
    -4.619142463669908903034297770894147469228d1
     5.83246649905202505112410282749412439786d2
    -3.724762778924749788460366261633512487703d3
     1.396652681770732801188732572379664489025d4
    -3.304670772736071603816210593286201073407d4
     5.088511427564893785874330939961434745583d4
    -5.09789119312603901154769398766439322207d4
     3.208907697442640417330395621172540694798d4
    -1.15373503269041552654688598274372853 61d4
     1.808665283923007924611465945818788164266d3
1d0 !BDS[23]
     0.0d0
    -2.492707572221787156726840939803698767962d0
     8.609208049352538615981609465349600483734d1
    -1.02144385349578552117188443087045953549d3
     6.149185051924085238544920882785954549985d3
    -2.263986205508318858979832710140633883991d4
     5.469068938282963282253427185108159994922d4
    -8.81386034259370399214866261732740741419d4
     9.351058473199922993332558926710969674549d4
    -6.252905849382715577610572728364544698941d4
     2.384385572753392575998934763706670098148d4
    -3.948946438865007544834653902561325025538d3
1d0 !BDS[24]
     0.0d0
     4.412086574202246678166723641068370995505d1
    -8.709755027763351195682541368091549459709d2
     8.035482702347852227232402210941648413255d3
    -4.328915582193813150449803886565443129058d4
     1.473747293513880934487519208566187859723d5
    -3.291970868334006316673899229714001377666d5
     4.890275283115693065750512810783905200316d5
    -4.786744579811296737884096577021598487196d5
     2.966573414545908395342508359089474293695d5
    -1.055265641510443922371150640684660305652d5
     1.641903760465105006491283045318053579135d4
1d0 !BDS[25]
     0.0d0
     6.249126524211618011269159124424929234087d0
    -2.516704150606577005544771987728218527563d2
     3.884776923851723048422187719430959470335d3
    -3.024199559157785480899930881443738618822d4
     1.319034138916292189645156449043456042329d5
    -3.483414033599664245509789190618183052531d5
     5.810415151314026976465640821277703147426d5
    -6.175795534419133570417540015242983658501d5
     4.064408153028398051252226575516412597632d5
    -1.512042011349585183431436500151256051412d5
     2.434205356722915604269451515213992114709d4
1d0 !BDS[26]
     0.0d0
    -5.897592739009822717996245153986151818698d0
     1.354029638570534793939442073305919260471d2
    -9.997380354707414052843641819110821653188d1
    -7.157134038942422039178908175781131613184d3
     4.785875282733995955198094364562522493681d4
    -1.49556624986610765148340534346843038881d5
     2.721629530866335214851123138639527777221d5
    -3.04873468678430202866671892221998478753d5
     2.078184525990591757713009619016780373116d5
    -7.929762558669759614210417957114658325992d4
     1.301516321007735987175378336052769497892d4
1d0 !BDS[27]
     0.0d0
    -9.255338855367684516066664099130933943703d-1
     3.018392697373610472315283404492938330522d1
    -3.200807968116487664209486555143581354165d2
     1.805950931753580341636158174401053571875d3
    -7.639653005455935933972384420006829312564d3
     2.465706813630764425492471930667651027754d4
    -5.445160221088040196077576319866893979233d4
     7.646138405113297010065247549937069216105d4
    -6.466094383963742622546157264059909898279d4
     2.995331156890481480320165474542256170385d4
    -5.83469322840179595005588497871660778112 4d3
1d0 !BDS[28]
     0.0d0
     7.810250764483732296076028542479055014428d0
    -2.958786383909233035662248477740457571147d2
     4.173537203443198624787569917578296348446d3
    -3.037985349329982999644990287764978624171d4
```

```
      1.29200443277297416384073352555275565827d5
     -3.42708989398612788963449377810136717817 5d5
```

Appendix C

GLOBS optimisation code

Contained here is the general purpose Fortran 90 "*Search.f90*" module used for optimisation of the dense processes from Chapters 6–8. It contains a modified MDS algorithm for performing multiple local minima searches from a given starting point in the parameter space (see Section 5.7.1). This code is available via anonymous ftp at *ftp.tees.ac.uk* in */pub/t.baker/f90/globs.f90.*

```
MODULE Search

Double precision, allocatable :: MINS(:, :), SV(:)
Double precision :: pfh, tol2, pfw, eps
Integer :: gp, n
Logical :: epf

Contains

SUBROUTINE Globs(F, range, V0, nm, lG, fin, start, fmin, V, thresh, fuse, &
  & interactive, dizziness, name_in, prnt)

!----------------------------------------------------------------------!
!GLOBS(F, R(2), V, M, G(2), TOL(2), SS, FMIN, Z, T, USE, INT, D, NAME, P) !
!----------------------------------------------------------------------!
!'Minimises' the function F(X) from the starting vector V using a modified!
!MDS simplex algorithm to further the search for local minima* beyond     !
!single convergence using a modified function space (MFS).                !
!----------------------------------------------------------------------!
!The MFS is such that local minima* are 'plugged' with decaying           !
!exponential functions of height G(1) and 'effective width' G(2). If G(1) !
!is ZERO, an inverse-square law is used to avoid known minima* (compare   !
!with an electromagnetic repulsion). |SS| specifies the size of the       !
!initial simplex, which is regular (all sides of equal length) if SS is   !
!positive, or right-angled (based on the co-ordinate axis) otherwise - in !
!this case, if both nonzero, R(1) and R(2) specify a range of parameters  !
!to modify, all others remaining constant (i.e., the minimisation is done !
!in a parameter subspace). TOL(1) specifies the relative convergence point!
!in the MFS, and TOL(2) convergence in a scaled parameter space. M gives  !
!the maximum storage size for minima* found, and therefore the total      !
!number of stops. Stopping points are all logged to full precision in the !
!file "NAME_path", and to 3 decimal places in "NAME_mins" (for plotting)  !
!as: {iteration number, function value, vector of parameters}. If USE is  !
!set logical TRUE, and a "NAME_path" file exists, this information is     !
!included into the run, where the ‘best’ minima in the path is used as the!
!starting point. The file "NAME_best" holds the current ‘best’ point, so  !
!in the absence of a path file, this is used as the starting point when   !
!USE is TRUE. If the parameter INT (for INTeractive) is set TRUE, both of !
!these choices can be made from the standard input. If the string NAME is !
!passed in blank or null, the default of "GLOBS" will be used, unless in  !
!interactive mode, when it will be prompted from the standard input. The  !
!‘dizzy simplex’ parameter D controls the random walk - ZERO for none, ONE!
!is reasonable, but increase to taste. This ‘feature’ is intended to allow!
!concurrent searches over many CPU’s, where the random element forces each!
!execution to be different whilst hopefully not affecting the results.    !
!T is a relative threshold parameter below which no descent in the MFS    !
!will be allowed (useful for ’noisy’ functions - set negative for         !
!default). The integer parameter P specifies how often to display         !
!progress, ZERO for none - in this case, no output of any kind is done.   !
!-------------------------------------------------------------------------!
!Output arguments: Z = ’best’ parameters, FMIN = F(Z).                    !
!-------------------------------------------------------------------------!
!*A local minima is meant only as a point where the simplex has converged.!
!-------------------------------------------------------------------------!

Double precision :: V0(:), V(Size(V0)), tol1, VM(Size(V0), &
  & Size(V0) + 1), T(Size(V0), 2:Size(V0) + 1), FV(Size(V0) + 1), &
  & TFV(Size(V0) + 1), fmin_old, simp_size, f_span, fmin_last, &
  & thresh, lf, bf, start, fin(2), F, WV(Size(V0) + 1), f_perc, &
  & fmin, TWV(Size(V0) + 1), HV(Size(V0) + 1), THV(Size(V0) + 1), &
  & hmin, G(2), rf, h_span, alpha(2), wmin, dizziness, rn, &
  & KBV(Size(V0)), lG(2), LSV(Size(V0))
Logical :: replaced, con(2), lcon(2), tst, interactive, flag, fuse, &
  & stuck
Integer :: i, it, prnt, nm, gj, gl, gll, k, dummy, kv(1), range(2), &
  & nr, sr
Integer, parameter :: max_len = 70, top_len = 5000
Character :: zn*3, is1*5, is2*5, rs*10, name * 500
Character (len = top_len) :: rs2
Character(*) :: name_in

external F

eps = Epsilon(1.0D0) ! Machine DP epsilon.

con = .false. ! Convergence of modified function and step flags.
if (thresh <= 0.0D0) thresh = eps * 100 !Descent threshold.
tol1 = fin(1) ! Tolerance for convergence of function values.
```

```
if (tol1 <= 0.0D0) then ; con(1) = .true. ; tol1 = 0.0D0 ; endif
tol2 = fin(2) ! Tolerance for convergence test based on size of simplex.
if (tol2 <= 0.0D0) then ; con(2) = .true. ; tol2 = 0.0D0 ; endif

lcon = con ; n = Size(V0) ; gll = 0 ; gp = 0 ; gj = 1 ; gl = 0

Allocate(SV(n)) ; if (nm > 1) Allocate(MINS(nm, 0:n))

If (Len_trim(name_in) == 0 .and. interactive) then
  write(*, advance = "no", fmt = '(a)') "Run Name :"
  read(*, '(a)') name ; if (Len_trim(name) == 0) name = "globs"
else
  name = name_in
End If

if (Len_trim(name) == 0) name = "globs"

Inquire(file = Trim(name)//"_path", exist = tst)
open(10, file = Trim(name)//"_path")

If (tst .and. (fuse .or. interactive)) then
  Do while (gp < nm - 1)
    call Read10(rs2, flag) ; if (flag) exit
    read(rs2, *) dummy, MINS(gp + 1, :)
    If (gp == 0) then
      print*, " *** READING PATH DATA FROM "//Trim(name)// &
        & "_path ***" ; print*
    End If
    gp = gp + 1
  End Do
elseif (tst) then
  close(10, status = "delete") ; open(10, file = Trim(name)//"_path")
End If

close(10) ; Inquire(file = Trim(name)//"_best", exist = tst)

if (gp /= 0) gj = gp + 1

If ((gp /= 0 .or. tst) .and. (fuse .or. interactive)) then
  If (interactive) then
    write(*, advance = "no", fmt = "(a)") &
      & "  >> START FROM CURRENT BEST (yes ?) >> "
    read(*, '(a)') rs ; print*
  else
    rs = ""
  End If
  If (Index(rs, "n") == 0) then
    If (gp /= 0) then
      kv = Minloc(MINS(:gp, 0)) ; k = kv(1) ; V0 = MINS(k, 1:)
      bf = MINS(k, 0)
    else
```

```
      open(10, file = Trim(name)//"_best") ; read(10, *) rs2
      read(10, *) rs2 ; call Read10(rs2, flag) ; k = Index(rs2, "/)")
      rs2 = rs2(3:k - 1) ; close(10) ; read(rs2, *) V0 ; bf = Huge(1.0D0)
    End If
  else
    bf = Huge(1.0D0)
  End If
else
  bf = Huge(1.0D0)
End If

open(10, file = Trim(name)//"_mins") ; close(10, status = "delete")
open(10, file = Trim(name)//"_mins")

do i = 1, gp ; write(10, '(i3, 1000(" ", es10.3))') i, MINS(i, :) ; enddo

close(10) ; SV = 1.0D0 ; KBV = V0 ; call Scale(V0, LSV)

VM = 0.0D0 ; do i = 1, n ; VM(i, i + 1) = 1.0D0 ; enddo ; VM(:, 1) = V0
FV = 0.0D0 ; TFV = FV ; TWV = FV ; THV = FV ; FV(1) = FF(V0) ; HV = FV
G = lG ; fmin = FV(1) ; fmin_old = fmin ; hmin = fmin
fmin_last = fmin ; lf = bf ; it = 0 ; write(zn, '(i3)') n - 1

if (prnt > 0) write(*, '("Iter.    0:  F(V0) = ", es10.3)') fmin

If (G(1) /= 0.0D0) then
  if (tol1 /= 0.0D0) G(2) = 2 * (log(tol1 / G(1))) / Abs(G(2))
  epf = .true.
else
  G(1) = Abs(G(2)) ; G(2) = 0.0D0 ; epf = .false.
End If

pfh = G(1) ; pfw = G(2) ; stuck = .false.
sr = range(1) ; nr = range(2) - sr + 1

WV(1) = ISF(VM(:, 1)) ; HV(1) = FV(1) + WV(1)

if (Abs(start) <= tol2) start = Sign(tol2 * 2.0D0, start)

if (sr <= 0 .or. nr <= 0 .or. sr + nr > n + 1 .or. start > 0.0D0) then ; &
  & sr = 1 ; nr = n ; endif

If (start > 0.0D0) then
 !Regular simplex.
  alpha = start * Max(Maxval(Abs(V0)), 1.0D0) / (Real(nr, 2) * &
    & Sqrt(2.0D0)) * (/Sqrt(Real(nr + 1, 2)) - 1.0D0 + Real(nr, 2), &
    & Sqrt(Real(nr, 2) + 1.0D0) - 1.0D0/)
  do i = 2, nr + 1 ; VM(:, i) = VM(:, 1) + alpha(2) ; enddo
  Do i = 2, nr + 1
    VM(i - 1, i) = VM(i - 1, 1) + alpha(1) ; FV(i) = FF(VM(:, i))
    WV(i) = ISF(VM(:, i)) ; HV(i) = FV(i) + WV(i)
  End Do
else
 !Right-angled simplex based on co-ordinate axes.
  Do i = sr + 1, nr + sr
    VM(:, i) = VM(:, 1) + &
      & Abs(start) * Max(Maxval(Abs(VM(:, 1))), 1.0D0) * VM(:, i)
    FV(i) = FF(VM(:, i)) ; WV(i) = ISF(VM(:, i))
    HV(i) = FV(i) + WV(i)
  End Do
End If

Do  ! OUTER REPEAT LOOP.

  it = it + 1 ; call FNBV ; if (dizziness > 0.0D0) call Random_Seed

  If (lf - Minval(FV) > thresh) then
    If (pfh /= 0.0D0) then
      pfh = 0.0D0
      do i = 1, n + 1 ; WV(i) = ISF(VM(:, i)) ; HV(i) = FV(i) + WV(i) ; &
        & enddo
      call FNBV
    End If
    if (bf - fmin > thresh) call OB
  End If

  If (prnt > 0) then
    If (Mod(it, prnt) == 0) then
      simp_size = 0.0D0
      do i = sr + 1, nr + sr ; simp_size = Max(simp_size, &
        & Sqrt(Sum((VM(:, i) - V)**2))) ; enddo
      simp_size = simp_size / Max(1.0D0, Maxval(Abs(V)))
      f_span = Maxval(Abs(HV(sr + 1:nr + sr) - HV(1))) / &
        & Max(1.0D0, Abs(HV(1)))
      f_perc = 100.0D0 * (fmin_last - fmin) / (Abs(fmin_last) + eps)
      write(*, '("Iter. ", i4, ":  Simplex = ", es7.1, "/", &
        & es7.1, ", PF = ", es8.2, ", F = ", &
        & es10.3, " (", f4.1, "%)")') it, f_span, simp_size, &
        & wmin, fmin, f_perc
      fmin_last = fmin
    End If
  End If

  fmin_old = fmin

  Do  ! INNER REPEAT LOOP.

  !Stopping criterion 1 - function converged? i.e., relative
  !difference in function values over the simplex is < tol1
   h_span = Maxval(Abs(HV(sr + 1:nr + sr) - HV(1))) / &
     & Max(1.0D0, Abs(HV(1)))
   f_span = Maxval(Abs(FV(sr + 1:nr + sr) - FV(1))) / &
```

```
    & Max(1.0D0, Abs(FV(1)))
  if (h_span <= tol1 .or. simp_size < thresh) then ; con(1) = .true. ; &
    & elseif (.not. lcon(1)) then ; con(1) = .false. ; endif

 !Stopping criterion 2 - step converged?

  simp_size = 0.0D0
  do i = sr + 1, nr + sr ; simp_size = Max(simp_size, &
    & Sqrt(Sum((VM(:, i) - V)**2))) ; enddo
  simp_size = simp_size / Max(1.0D0, Maxval(Abs(V)))
  if (simp_size <= tol2 .or. h_span < thresh) then ; con(2) = .true. ; &
    & elseif (.not. lcon(2)) then ; con(2) = .false. ; endif

  rf = 1.0D0

  If (All(con)) then
    If (nm < 2) then
      if (prnt > 0) write(*, '(/, a)') 'Run complete.'
      V = V * SV ; return
    else
      con = lcon ; it = 0
      If (fmin < bf .or. Abs(lf - fmin) > thresh) then
        gp = gp + 1 ; MINS(gp, :) = (/fmin, V * SV/)
        If (n == 1) then
          write(rs2, '(i3, " ", es25.18, ", ", es25.18)') &
            & gj, fmin, V(1) * SV(1)
        else
          write(rs2, '(i3, " ", es25.18, ", ", es25.18, '//zn// &
            & '(" ", es25.18))') gj, fmin, V * SV
        End If
        open(10, file = Trim(name)//"_path", position = "append")
        call Write10(rs2) ; close(10)
        open(10, file = Trim(name)//"_mins", position = "append")
        write(10, '(i3, 1000(" ", es10.3))') gp, fmin, V * SV ; close(10)
        stuck = .false.
        write(is1, '(i4)') gll ; write(is2, '(i4)') gl
        rf = Max(1.0D0, &
          & (tol2 / Max(eps, simp_size) + tol1 / Max(eps, h_span)) / 2)
      End If
      If (fmin < bf) then
        if (prnt > 0) write(*, '(/, 13x, "**HIT**", /)') ; call OB
      elseif (lf - fmin > thresh .and. prnt > 0) then
        write(*, '(/, 10x, "**GOING DOWN (", a, ")**", /)') &
          & Trim(Adjustl(is2))
      elseif (Abs(fmin - lf) < thresh .and. prnt > 0) then
        if (.not. stuck) then ; &
          & write(*, '(/, 12x, "**STUCK**", /)') ; stuck = .true. ; endif
        rf = 2.0D0 / simp_size
      elseif (prnt > 0) then
        write(*, '(/, 11x, "**GOING UP (", a, ")**", /)') &
          & Trim(Adjustl(is1))
      End If
      if (lf - fmin > tol1 .or. Abs(fmin - lf) < tol1) then ; &
        & gll = 0 ; gl = gl + 1 ; else ; gll = gll + 1 ; endif
      If (fmin < bf) then
        gl = 0 ; KBV = V * SV ; bf = fmin ; call Scale(V, LSV)
        do i = 1, n + 1 ; VM(:, i) = VM(:, i) / LSV ; enddo
      End If
      If (gp == nm) then
        if (prnt > 0) write(*, '(/, a)') 'Run complete.'
        V = KBV ; fmin = bf ; return
      End If
      gj = gj + 1 ; lf = fmin
      pfh = G(1) * 2.0D0 ** gll ; if (epf) pfw = G(2) / 2.0D0 ** gll
      WV(1) = ISF(VM(:, 1)) ; HV(1) = FV(1) + WV(1)
      do i = sr + 1, nr + sr ; WV(i) = ISF(VM(:, i)) ; &
          & HV(i) = FV(i) + WV(i) ; enddo
      call FNBV
    End If
  End If

  If (dizziness > 0.0D0) then
    call Random_Number(rn) ; rn = rn - 5.0D-1
    rf = rf * exp(dizziness * rn)
  End If

  Do i = sr + 1, nr + sr   ! Reflection step with random scaling.
    T(:, i) = V + rf * (V - VM(:, i)) ; TFV(i) = FF(T(:, i))
    TWV(i) = ISF(T(:, i)) ; THV(i) = TFV(i) + TWV(i)
  End Do

  replaced = LT(Minval(THV(sr + 1:nr + sr)), hmin)

  If (replaced) then
    Do i = sr + 1, nr + sr  ! Expansion step.
      VM(:, i) = 2.0D0 * T(:, i) - V ; FV(i) = FF(VM(:, i))
      WV(i) = ISF(VM(:, i)) ; HV(i) = FV(i) + WV(i)
    End Do
    !Reject expansion?
    If (LT(Minval(THV(sr + 1:nr + sr)), Minval(HV(sr + 1:nr + sr)))) then
      VM(:, sr + 1:nr + sr) = T(:, sr + 1:nr + sr)
      FV(sr + 1:nr + sr) = TFV(sr + 1:nr + sr)
      WV(sr + 1:nr + sr) = TWV(sr + 1:nr + sr)
      HV(sr + 1:nr + sr) = THV(sr + 1:nr + sr)
    End If
  else  ! Contract
    Do i = sr + 1, nr + sr
      VM(:, i) = (1.5D0 * V) - (0.5D0 * T(:, i))
      FV(i) = FF(VM(:, i)) ; WV(i) = ISF(VM(:, i))
      HV(i) = FV(i) + WV(i)
    End Do
    replaced = LT(Minval(HV(sr + 1:nr + sr)), hmin)
```

```
   End If

   if (replaced) exit

  End Do ![INNER REPEAT LOOP].

End Do ![OUTER REPEAT LOOP].

Contains

SUBROUTINE OB

open(10, file = Trim(name)//"_best", status = "replace")
write(10, *) "RUN = ", gj, ", F(X) = ", fmin ; write(10, *) "X = "
If (n == 1) then
  write(rs2, '("(/", es25.18, "/)")') V(1) * SV(1)
else
  write(rs2, '("(/", es25.18, '//zn//'(", ", es25.18), "/)")') V * SV
End If
call Write10(rs2) ; close(10)

END SUBROUTINE OB

LOGICAL FUNCTION LT(a, b) ! `a < b' to within a tolerance (THRESH).

Real (kind = 2) :: a, b

LT = (b - a) / Max(1.0D0, a) > thresh

END FUNCTION LT

SUBROUTINE FNBV ! Find a new best vertex.

Double precision :: tf
Integer :: bv, bvv(1)

bvv = Minloc((/HV(1), HV(sr + 1:nr + sr)/)) ; bv = bvv(1)
if (bv > 1) bv = bv + sr - 1
hmin = Minval((/HV(1), HV(sr + 1:nr + sr)/)) ; wmin = WV(bv)
fmin = FV(bv) ; V = VM(:, bv) ; VM(:, bv) = VM(:, 1) ; VM(:, 1) = V
tf = FV(1) ; FV(1) = FV(bv) ; FV(bv) = tf ; tf = WV(1) ; WV(1) = WV(bv)
WV(bv) = tf ; tf = HV(1) ; HV(1) = HV(bv) ; HV(bv) = tf

END SUBROUTINE FNBV

SUBROUTINE Write10(string) ! Write to unit 10.

Integer :: len, llen, i
Character(*) :: string

len = Len_trim(string)
```

```
Do
  If (len <= max_len) then
    write(10, '(a)') string(:len) ; exit
  else
    llen = max_len ; i = Index(string(:llen), " ", back = .true.)
    if (i /= 0) llen = i
    write(10, '(a)') string(:llen)//" &"
    string = string(llen + 1:) ; len = len - llen
  End If
End Do

END SUBROUTINE Write10

SUBROUTINE Read10(string, flag) ! Read from unit 10.

Character (len = max_len + 10) :: lstr
Character (len = top_len) :: string
Integer :: len, k, tlen
Logical :: flag

string = "" ; flag = .false. ; tlen = 0

Do
  read(10, iostat = k, fmt = '(a)') lstr ; len = Len_trim(lstr)
  if (k /= 0) then ; flag = .true. ; return ; endif
  If (lstr(len:len) == "&") then
    string = string(:tlen)//lstr(: len - 2) ; tlen = tlen + len - 2
  else
    string = string(:tlen)//lstr ; exit
  End If
End Do

END SUBROUTINE Read10

FUNCTION FF(x) ! Function evaluation in scaled space.

Double precision :: x(n), FF

FF = F(x * SV)

END FUNCTION FF

FUNCTION ISF(x) ! Evaluate penalty function at x.

Double precision :: x(n), ISF, pf, PV(gp)
Integer :: i

If (gp /= 0 .and. .not. epf .and. tol1 == 0.0D0) then
  print* ; print*, "ERROR - cannot operate global repulsion scheme &
    &without a function tolerance" ; stop
```

```
End If

Do i = 1, gp
  pf = Sum((SV * x - MINS(i, 1:n))**2)
  if (epf) then ; PV(i) = exp(pfw * Sqrt(pf)) ; else ; &
    & PV(i) = 1.0D0 / Max(eps, 2.0D0 * pf / tol1) ; endif
End Do

ISF = pfh * Sum(PV)

END FUNCTION ISF

SUBROUTINE Scale(x, V) ! Pull powers of two from vector x (scaling in V).

Double precision :: x(n), lx, V(n)
Integer :: i

Do i = 1, n
  If (x(i) /= 0.0D0) then
    lx = 2.0D0**(Int(Log10(Abs(x(i)))/Log10(2.0D0)))
    SV(i) = SV(i) * lx ; x(i) = x(i) / lx ; V(i) = lx
  else
    V(i) = 1.0D0
  End If
End Do

END SUBROUTINE Scale

END SUBROUTINE Globs

END MODULE Search
```

References

[1] I. Euler. Institutionum calculi integralis. *Volumen Primum, Opera Omnia*, XII, 1768.

[2] C. Runge. Über die numerische auflösung von differentialgleichungen. *Math. Ann.*, 46:167–178, 1895.

[3] K. Heun. Neue methode zur approximativen integration der differentialgleichungen einer unabhängigen veränderlichen. *Zeitschr. für Math. u. Phys.*, 45:23–38, 1900.

[4] W. Kutta. Beitrag zur näherungsweisen integration totaler differentialgleichungen. *Zeitschr. für Math. u. Phys.*, 46:435–453, 1901.

[5] E. J. Nyström. *Acta. Soc. Sci. Fenn.*, 50(13), 1925.

[6] P. Henrici. *Discrete variable methods in ordinary differential equations.* John Wiley and Sons, London, 1962.

[7] C. W. Gear. *Numerical initial value problems in ordinary differential equations.* Prentice-Hall, London, 1971.

[8] J. C. Butcher. On Runge-Kutta processes of high order. *J. Austral. Maths. Soc.*, 4:179–194, 1964.

[9] J. C. Butcher. Coefficients for the study of Runge-Kutta integration processes. *J. Austral. Maths. Soc.*, 3:185–201, 1963.

[10] J. D. Lambert. *Computational methods in ordinary differential equations.* John Wiley and Sons, London, 1973.

[11] R. P. Harris. In *Proceedings of the forth Australian computer conference*, pages 429–433, Adelaide, 1969.

[12] P. J. Prince. *Runge-Kutta processes with global error estimation.* PhD thesis, CNAA, 1979.

[13] T. E. Hull, W. E. Enright, B. H. Fellen, and A. E. Sedgewick. Comparing numerical methods for ordinary differential equations. *SIAM J. Numer. Anal.*, 9:603–637, 1972.

[14] J. R. Dormand and P. J. Prince. Runge-Kutta triples. *Comput. Math. Applic.*, 12A(9):1007–1017, 1986.

[15] J. R. Dormand and P. J. Prince. Runge-Kutta-Nyström triples. *Comput. Math. Applic.*, 13(12):937–949, 1987.

[16] M. K. Horn. *Developments in high order Runge-Kutta-Nyström formulas*, Dissertation, The University of Texas at Austin, 1977.

[17] P. Albrecht. Numerical treatment of O.D.E's: The theory of A-methods. *Numer. Math.*, 47:59–87, 1985.

[18] P. Albrecht. The common basis of the theories of linear cyclic methods and Runge-Kutta methods. *Applied Numerical Mathematics*, 22:3–21, 1996.

[19] J. R. Dormand, M. E. A. El-Mikkawy, and P. J. Prince. Families of Runge-Kutta-Nyström formulae. *IMA J. Numer. Anal.*, 7:235–250, 1987.

[20] J. R. Dormand and P. J. Prince. New Runge-Kutta-Nyström algorithms for simulation in dynamical astronomy. *Celestial Mechanics*, 18:223–232, 1978.

[21] H. I. Scoins. In D. Michie, editor, *Machine intelligence 3*, pages 43–60. Edinburgh University Press, 1968.

[22] M. E. Hosea. A new recurrence relation for computing Runge-Kutta truncation error coefficients. *SIAM J. Numer. Anal.*, 32(6):1989–2001, December 1995.

[23] E. Hairer. A one-step method of order 10 for $y'' = f(x, y)$. *IMA J. Numer. Anal.*, 2:83–94, 1982.

[24] J. C. Butcher. On the attainable order of Runge-Kutta methods. *Math. Comput.*, 19:408–417, 1965.

[25] J. C. Butcher. The non-existence of ten stage eighth order explicit Runge-Kutta methods. *BIT*, 25:521–540, 1985.

[26] A. R. Curtis. An eighth order Runge-Kutta process with eleven function evaluations per step. *Numer. Math.*, 16:268–277, 1970.

[27] G. J. Cooper and J. H. Verner. Some explicit Runge-Kutta methods of high order. *SIAM J. Numer. Anal.*, 9(3):389–405, 1972.

[28] E. Hairer. A Runge-Kutta method of order 10. *J. Inst. Math, Applics.*, 21:47–59, 1978.

[29] J. C. Butcher. Implicit Runge-Kutta processes. *Math. Comput.*, 18:50–64, 1964.

[30] R. H. Merson. An operational method for the study of integration processes. In *Proceedings of a symposium on data processing*, Weapons research establishment, Salisbury, South Australia, 1957.

[31] D. Sarafyan. Error estimation for Runge-Kutta methods through pseudo-iterative formulas. Technical Report 14, Lousiana State Univ., New Orleans, May 1966.

[32] R. England. Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations. *The computer J.*, 12:166–170, 1969.

[33] E. Fehlberg. Classical fifth-, sixth-, seventh-, and eigth-order Runge-Kutta formulaes with step size control. Technical report, NASA TR R-287, 1968.

[34] E. Fehlberg. Technical report, NASA TR R-315, 1969.

[35] J. C. Butcher. *The numerical analysis of Ordinary Differential Equations.* Wiley, Toronto, 1987.

[36] J. H. Verner. Srategies for deriving explicit Runge-Kutta formula pairs. *Annuls of Numerical Mathematics*, 1994.

[37] E. Hairer, P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations 1 (second edition).* Springer-Verlag, Berlin, New York, 1992.

[38] I. Gladwell. Initial value routines in the NAG library. *ACM Trans. Math. Software*, 5:386–400, 1979.

[39] M. K. Horn. Fourth and fifth-order scaled Runge-Kutta algorithms for treating dense output. *SIAM J. Numer. Anal.*, 20:558–568, 1983.

[40] L. F. Shampine. Interpolation for Runge-Kutta methods. *SIAM J. Numer. Anal.*, 22:1014–1027, 1985.

[41] W. H. Enright, K. R. Jackson, S. P. Norsett, and P. G. Thomsen. Interpolants for Runge-Kutta formulas, Technical report. Technical report, no. 180/185, Dept. of Computer Science, Univ. of Toronto, Toronto, Ontario, 1985.

[42] I. Gladwell, L. F. Shampine, L. S. Baca, and R. W. Brankin. Practical aspects of interpolation in Runge-Kutta codes, Numerical Analysis Report No. 102, Dept. of Mathematics, Univ. of Manchester, Manchester, UK. 1985.

[43] P. Bogacki and L. F. Shampine. Interpolating high-order Runge-Kutta formulas. *Computers Math, Applic.*, 22(3):15–24, 1990.

[44] Calvo. M., J. I. Montijano, and L. Rández. A fifth order interpolant for the Dormand and Prince Runge-Kutta method. *J. Comput. Appl. Math.*, 29:91–100, 1990.

[45] J. H. Verner. Differentiable interpolants for high-order Runge-Kutta methods. *SIAM J. Numer. Anal.*, 30(5):1446–1466, 1993.

[46] B. Owren. *Continuous explicit Runge-Kutta methods with application to ordinary and delay differential equations.* PhD thesis, Institutt for Matematiske Fag. Trondheim, 1989.

[47] B. Owren and M. Zennaro. Order barriers for continuous explicit Runge-Kutta methods. *Math. Comput.*, 56:645–661, 1991.

[48] B. Owren and M. Zennaro. Derivation of efficient continous explicit Runge-Kutta methods. *SIAM J. Sci. Statist. Comput.*, 13:1488–1501, 1992.

[49] P. Albrecht. *ZAMM*, 35:100–110, 1955.

[50] R. H. Battin. *AIAA J.*, 13:1012–1021, 1976.

[51] E. Hairer. Méthodes de Nyström pour l'équation différentiale $y'' = f(x, y)$. *Numer. Math.*, 27:283–300, 1977.

[52] E. Fehlberg. Classical eighth and lower order Runge-Kutta-Nyström formulas with step size control for special second order differential equations. Technical report, NASA TR R-381, 1972.

[53] E. Filippi and J. Graf. New Runge-Kutta-Nyström formula-pairs of order 8(7), 9(8), 10(9) and 11(10) for differential equations of the form $y'' = f(x, y)$. *J. Comput. Appl. Math.*, 14:361–370, 1986.

[54] D. G. Bettis. *Celestial Mechanics*, 8:229–223, 1973.

[55] D. G. Bettis. In JSC, Internal note No. 76-FM-6. 1976.

[56] J. R. Dormand, M. E. A. El-Mikkawy, and P. J. Prince. High order embedded Runge-Kutta-Nyström formulae. *IMA J. Numer. Anal.*, 7:423–430, 1987.

[57] J. M. Fine. Low order Runge-Kutta-Nyström methods with interpolants. Technical Report 183/85, Computer Science Dept., Toronto University, Ontario, 1985.

[58] G. Storer. *Global error estimation for Runge-Kutta-Nyström processes.* PhD thesis, University of Teesside, 1990.

[59] Ch. Tsitouras and G. Papageorgiou. Interpolating Runge-Kutta-Nyström methods of high order. *Internat J. Comput. Math.*, 47:209–217, 1993.

[60] J. R. Dormand, R. R. Duckers, and P. J. Prince. Global error estimation with Runge-Kutta methods. *IMA J. Numer. Anal.*, 4:169–184, 1984.

[61] V. J. Torczon. *Multi-directional search: a direct search method for parallel machines.* PhD thesis, Rice University, Houston, Texas, 1989.

[62] V. J. Torczon. On the convergence of the multi-directional search algorithm. *SIAM J. Optimisation*, 1:123–145, 1991.

[63] V. J. Torczon. On the convergence of pattern search algorithms. Technical Report TR93-10, Rice University, Houston, Texas, June 1993.

[64] R. W. Brankin, J. R. Dormand, I. Gladwell, P. J. Prince, and W. L. Seward. A Runge-Kutta-Nyström code. *ACM ToMS*, pages 31–40, 1988.

[65] P. J. Prince. Private communication, 1995.

[66] T. S. Baker, J. R. Dormand, J. P. Gilmore, and P. J. Prince. Continuous approximation with embedded Runge-Kutta methods. *Applied Numerical Mathematics*, 22:51–62, 1996.

[67] L. F. Shampine and I. Gladwell. Software based on explicit RK formulas. *Applied Numerical Mathematics*, 22:293–308, 1996.

[68] L. Lamport. *LaTeX: a Document Preparation System.* Addison-Wesley, Reading, MA, 1986.

[69] D. E. Knuth. *The TeXbook.* Addison-Wesley, Reading, MA, 1986.

[70] H. A. Watts. Step size control in ODE solvers. *Trans. Soc. for Computer Simulation*, 1:15–25, 1984.