VARIABLE–ORDER STARTING ALGORITHMS FOR IMPLICIT RUNGE-KUTTA METHODS ON STIFF PROBLEMS

S. González–Pinto [a], J.I. Montijano [b] and S. Pérez–Rodríguez [c]

[a] *Dpto. de Análisis Matemático, Universidad de La Laguna, 38271 La Laguna-Tenerife, Spain. email: spinto@ull.es*

[b] *Dpto. de Matemática Aplicada, Universidad de Zaragoza, 50009 Zaragoza, Spain. email: monti@posta.unizar.es*

[c] *Dpto. de Análisis Matemático, Universidad de La Laguna, 38271 La Laguna-Tenerife, Spain. email: sperezr@ull.es*

Abstract

This paper deals with starting algorithms for Newton-type schemes for solving the stage equations of implicit s–stages Runge-Kutta methods applied to stiff problems. We present a family of starting algorithms with orders from 0 to $s+1$ and, with estimations of the error in these algorithms, we give a technique for selecting, at each step, the most convenient in the family. The proposed algorithms, that can be expressed in terms of divided differences, are based on the Lagrange interpolation of the stages of the last two integration steps. We also analyse the orders of the starting algorithms for the non-stiff case, for the Prothero and Robinson model and the stiff order. Finally, by means of some numerical experiments we show that this technique allows, in general, to improve greatly the performance and reliability of implicit Runge-Kutta methods on stiff problems.

AMS subject classifications: 65L05

Key words: Stiff Initial Value Problems, Implicit Runge-Kutta methods, Starting algorithms.

[1] This work was supported by project DGES PB97–1018

1 Introduction.

Starting algorithms for solving the stage equations of fully implicit Runge-Kutta methods (by means of some Newton type iteration) considered in standard codes, when applied to stiff problems,

$$y'(t) = f(t, y), \quad y(0) = y_0 \in \mathbb{R}^m, \quad t \in [0, T], \tag{1.1}$$

are usually based on the Lagrange interpolation of the internal stages of the preceding integration step. Thus, STRIDE [2,3] and RADAU5 [15, Chap.IV.8] use such a kind of algorithms. In the last work the authors use for their code RADAU5, which is based on the three-stage Radau IIA formula, the Lagrange interpolation of the internal stages of the preceding integration step and of y_n. That is, by considering the Runge-Kutta method (with coefficient matrix A non–singular),

$$Y_{i,n} = y_n + h_n \sum_{j=1}^{s} a_{ij} f(t_n + c_j h_n, Y_{j,n}) \quad (i = 1, \ldots, s), \tag{1.2}$$

and

$$y_{n+1} = \omega y_n + \sum_{j=1}^{s} w_j Y_{j,n}, \quad (w_j)_{j=1,s} = b^T A^{-1}, \quad \omega = 1 - \sum_{j=1}^{s} w_j, \tag{1.3}$$

the starting algorithm proposed in order to begin the Newton iteration for the new step $t_{n+1} \to t_{n+2} = t_{n+1} + h_{n+1}$, is given, for the RADAU5 case ($s = 3$) by,

$$Y^0_{i,n+1} := p_3(t_{n+1} + c_i h_{n+1}), \quad 1 \le i \le 3, \tag{1.4}$$

where $p_3(t)$ is the interpolation polynomial on the points

$$\{(t_n, y_n), (t_n + c_i h_n, Y_{i,n}),\ 1 \le i \le 3\}.$$

Then, the iterative scheme is stopped when the Euclidean norm of the increment of the stage solution satisfies

$$\|Y^r_{i,n+1} - Y^{r-1}_{i,n+1}\| \le c \text{ Tol}$$

where Tol is the error tolerance used for the integration and c is a fixed coefficient that for example in RADAU5 is taken, by default, as 0.03.

In general, starting algorithms based on the internal stages of the preceding integration step, perform satisfactorily on stiff problems, when the underlying Runge-Kutta method is a collocation method (or it has a high stage order) with good stability properties. However, starting algorithms like (1.4) can make sometimes that standard variable step-size codes fail to complete the integration of certain "delicate" stiff problems. As an example, let us consider

the reaction of Robertson [15, p. 144] integrated along the interval $[0, 10^{11}]$. For this problem, if the second component y_2 becomes negative at some point t_n, then the corresponding integral curve tends to infinity when $t \to \infty$ (see [15, p. 144]) whereas the exact solution for $t \to \infty$ is $(0, 0, 1)^T$. If we integrate it with RADAU5, with the standard "default" options, and with absolute and relative tolerances greater or equal to 10^{-3}, the integration is not completed, and the code returns an error code "IDID=-3" that means "STEP SIZE BECOMES TOO SMALL". It can be argued that this problem must be solved with a lower relative error tolerance. However, if we integrate the simple test problem $y' = -(y-1)^2$, $y(0) = 2$, $t \in [0, 10^{11}]$, the code again stops the integration with the same error code, but in this case this happens for error tolerances greater than 10^{-9} (no change is observed if we specify different absolute and relative error tolerances because the solution remains close to 1).

These two problems can be solved robustly with RADAU5 if, instead of the default values, some special parameters are specified to the code. Thus, if you specify for the coefficient c of the stopping control a much smaller value, the integration is completed. For example, the second problem can be integrated with error tolerance 10^{-6} if $c \leq 10^{-6}$. In this case the code takes 122 accepted steps and the iterative scheme fails to converge at 33 steps, requiring a total of 479 solutions or pairs of triangular linear systems. The relative efficiency decreases for higher error tolerances. For example for tolerance 10^{-4}, the code integrates the problem with $c \leq 10^{-7}$ and takes 82 accepted steps, 51 rejected steps, needing 405 solutions of pairs of systems.

An other way to integrate the problem consists in selecting, instead of the Lagrange initial values, the approximation y_{n+1} as starting value (we can keep $c = 0.03$). In this case the integration is completed much more efficiently and thus, for Tol= 10^{-6} the code takes 98 steps with no rejected steps, needing 203 linear system solutions, while for Tol= 10^{-4}, it takes 45 steps, no rejections and 76 triangular system solutions.

As we have seen in these experiences, the starting algorithm chosen can be critical in the performance of a code and algorithms with high order are not always preferable to other with lower order. In order to gain some additional information about the influence of the starting algorithm on the behaviour of the integration code, we have developed a very simple variable step-size code, based on the RadauIIA formula of order 5, not intended to compete with standard existing codes, but just to study the performance when different initial guesses are used. We have used local extrapolation as method for estimating the local error, halving the stepsize when a step is rejected. The stage equations are solved by a modified Newton scheme, which is stopped when the Euclidean norm of the increment is smaller or equal to Tol/100. If the increment at one iteration does not decrease by at least a factor of 0.9, or well the scheme does not converge in 10 iterations, we reject the step, halving the

stepsize. Also, the Jacobian matrix is evaluated, and the corresponding matrix factorized, every two steps (since we use extrapolation as error estimation, the always take two consecutive steps with the same stepsize).

In the code we have considered the possibility of using, for $i = 1, 2, 3$ the following starting algorithms:

(1) $\mathcal{L}_i^{(3)} = p_3(t_{n+1} + c_i h_{n+1})$, with $p_3(t)$ as in (1.4).
(2) $\mathcal{L}_i^{(2)} = p_2(t_{n+1} + c_i h_{n+1})$, being $p_2(t)$ the interpolation polynomial on the three points $\{(t_n + c_j h_n, Y_{j,n}),\ 1 \le j \le 3\}$.
(3) $\mathcal{L}_i^{(1)} = p_1(t_{n+1} + c_i h_{n+1})$, being $p_1(t)$ the interpolation polynomial on the two points $\{(t_n + c_j h_n, Y_{j,n}),\ 2 \le j \le 3\}$.
(4) $\mathcal{L}_i^{(0)} = Y_{3,n} = y_{n+1}$.

As we will see later in section 3, $\mathcal{L}_i^{(k)}$ has order k, that is,

$$\mathcal{L}_i^{(k)} - Y_{i,n+1} = \mathcal{O}(h_n^{k+1}) \quad (h_n \to 0^+), \quad 1 \le i \le 3, \quad 0 \le k \le 3.$$

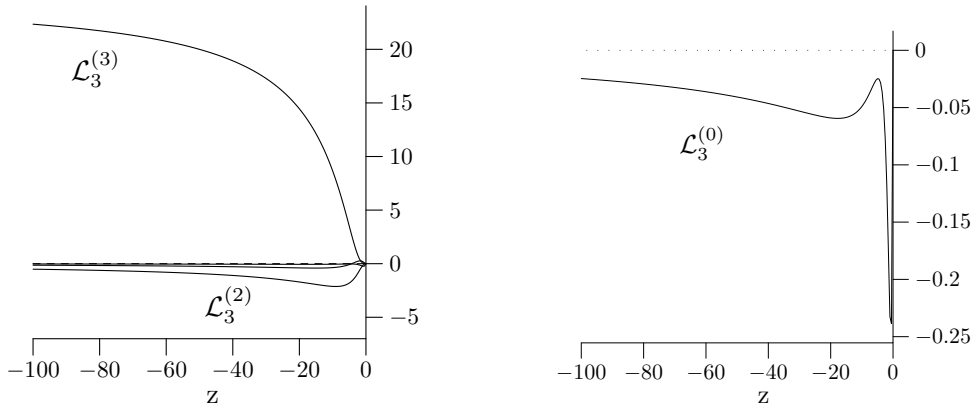
Table 1.1 displays the results obtained when the Robertson problem is integrated, for tolerances $TOL = 10^{-1}, 10^{-2}, \ldots, 10^{-8}$, when every starting algorithm is used ("*" means that the code was unable to complete the integration). In all cases we have taken as initial stepsize $h_0 = 10^{-3}$.

In the Table TOL means the error tolerance considered (we have used a mixed option with the same relative and absolute error tolerances), SA denotes the starting algorithm taken, GE the global error at the end-point, NSS is the number of successful steps, NR-SN is the number of rejected steps by a fail in the iteration scheme, NLU the number of LU factorisations, NSOL is the number of (couple of) triangular systems solved, NFN represents the number of evaluations of the derivative function and NITER stands for the average of iterations per integration step (the first entry corresponds to the first step, the second one to the second step and the third one corresponds to the double step, recall that we are using the extrapolation technique to estimate the local error). Since in all the cases the error estimator was smaller than the error tolerance, there were no rejections by this cause and we have not included this number.

Regarding the Lagrange starting algorithms above described, in Table 1.1 we can see that the code was able to conclude the integration for all tolerances only when the starting algorithm denoted by $\mathcal{L}_i^{(0)}$ was used. Further, $\mathcal{L}_i^{(1)}$ is more robust than $\mathcal{L}_i^{(2)}$, and this one is more robust than $\mathcal{L}_i^{(3)}$. Observe that although $\mathcal{L}_i^{(3)}$ permitted the code to accomplish the integration for $TOL = 10^{-3}$, the numerical solution obtained was completely wrong. We must mention

Table 1.1
Reaction of Robertson

TOL	SA	GE	NSS	NR-SN	NLU	NSOL	NFN	NITER
10^{-1}	$\mathcal{L}_i^{(0)}$	$0.34 \cdot 10^{-8}$	92	0	184	412	616	1.7,1.8,1.0
	$\mathcal{L}_i^{(1)}$	*	*	*	*	*	*	*
	$\mathcal{L}_i^{(2)}$	*	*	*	*	*	*	*
	$\mathcal{L}_i^{(3)}$	*	*	*	*	*	*	*
	V-code	$0.32 \cdot 10^{-8}$	114	9	234	430	643	1.3,1.3,1.0
10^{-2}	$\mathcal{L}_i^{(0)}$	$0.34 \cdot 10^{-8}$	92	0	184	514	769	2.1,2.3,1.2
	$\mathcal{L}_i^{(1)}$	$0.28 \cdot 10^{-8}$	104	2	210	520	778	1.8,1.9,1.2
	$\mathcal{L}_i^{(2)}$	*	*	*	*	*	*	*
	$\mathcal{L}_i^{(3)}$	*	*	*	*	*	*	*
	V-code	$0.32 \cdot 10^{-8}$	108	7	228	516	772	1.6,1.7,1.2
10^{-3}	$\mathcal{L}_i^{(0)}$	$0.34 \cdot 10^{-8}$	98	2	196	682	1019	2.6,2.8,1.6
	$\mathcal{L}_i^{(1)}$	$0.28 \cdot 10^{-8}$	98	1	196	620	926	2.3,2.4,1.6
	$\mathcal{L}_i^{(2)}$	$0.18 \cdot 10^{-8}$	98	2	196	586	875	2.1,2.2,1.6
	$\mathcal{L}_i^{(3)}$	$0.47 \cdot 10^{8}$	928	336	1932	5296	7940	2.0,1.2,1.6
	V-code	$0.32 \cdot 10^{-8}$	98	2	196	594	887	2.2,2.2,1.6
10^{-4}	$\mathcal{L}_i^{(0)}$	$0.32 \cdot 10^{-8}$	98	2	196	856	1283	3.0,3.5,2.1
	$\mathcal{L}_i^{(1)}$	$0.27 \cdot 10^{-8}$	98	1	196	792	1187	2.9,2.9,2.1
	$\mathcal{L}_i^{(2)}$	$0.17 \cdot 10^{-8}$	98	2	196	760	1139	2.7,2.8,2.1
	$\mathcal{L}_i^{(3)}$	$0.19 \cdot 10^{-8}$	104	3	210	748	1121	2.5,2.5,2.1
	V-code	$0.30 \cdot 10^{-8}$	98	2	196	740	1109	2.6,2.8,2.1
10^{-6}	$\mathcal{L}_i^{(0)}$	$0.12 \cdot 10^{-8}$	106	1	214	1308	1961	4.1,4.8,3.3
	$\mathcal{L}_i^{(1)}$	$0.78 \cdot 10^{-9}$	106	1	214	1246	1868	4.0,4.3,3.3
	$\mathcal{L}_i^{(2)}$	$0.61 \cdot 10^{-9}$	106	1	214	1186	1778	3.8,4.0,3.3
	$\mathcal{L}_i^{(3)}$	$0.99 \cdot 10^{-9}$	106	1	214	1138	1706	3.6,3.7,3.3
	V-code	$0.99 \cdot 10^{-9}$	106	1	214	1180	1769	3.7,4.0,3.3
10^{-8}	$\mathcal{L}_i^{(0)}$	$0.82 \cdot 10^{-11}$	142	3	286	2152	3228	5.0,5.8,4.2
	$\mathcal{L}_i^{(1)}$	$0.61 \cdot 10^{-11}$	142	2	286	2064	3096	4.9,5.3,4.2
	$\mathcal{L}_i^{(2)}$	$0.87 \cdot 10^{-11}$	142	3	286	1948	2922	4.4,4.9,4.2
	$\mathcal{L}_i^{(3)}$	$0.55 \cdot 10^{-11}$	142	3	286	1844	2766	4.1,4.5,4.2
	V-code	$0.65 \cdot 10^{-11}$	142	3	286	1892	2838	4.2,4.7,4.2



(a) (b)

Fig. 1.1. Amplifying factors $R_3(z) - R_3^0(z)$ of the starting algorithms $\mathcal{L}_3^{(0)}$, $\mathcal{L}_3^{(1)}$, $\mathcal{L}_3^{(2)}$ and $\mathcal{L}_3^{(3)}$.

that a similar behaviour is observed if we integrate the problem taking the smaller integration interval $[0, 40]$.

It can be surprising that starting algorithms perform worse when we increase their order. An explanation of such a behaviour might be given by considering the Prothero & Robinson test problem [19]

$$y' = \lambda(y - \phi(t)) + \phi'(t), \quad \mathrm{Re}(\lambda) \leq 0. \tag{1.5}$$

Since the error of a starting algorithm $Y_{i,n+1}^0$ of order p when this problem is integrated can be expressed in the form (see [12])

$$Y_{i,n+1} - Y_{i,n+1}^0 = \Big(R_i(h_n\lambda) - R_i^0(h_n\lambda)\Big)(y_n - \phi(t_n)) + \mathcal{O}(h_n^{p+1}), \tag{1.6}$$

being $R_i(z) = Y_{i,n+1}/y_n$ and $R_i^0(z) = Y_{i,n+1}^0/y_n$ the amplifying functions of the true solution $Y_{i,n+1}$ and the starting algorithm respectively, the error is affected by a term $\mathcal{O}(h_n^{p+1})$ independent of the stiffness, but also by a term that can be interpreted as a propagation of the previous error. Then, if a starting algorithm has a high order but it has poor stability properties, that is, its amplifying function is not close to the amplifying function of the true solution, the errors can be larger than expected. This phenomenon can be particularly present for large tolerances where the order theory, based on asymptotic arguments can lose the validity.

In Figure 1 (a) we have plotted the functions $R_3(z) - R_3^0(z)$ for the third stage of the starting algorithms $\mathcal{L}_i^{(0)}$ to $\mathcal{L}_i^{(3)}$, with z varying on the negative real axis and for the particular value of the stepsize ratio $r_n = 1$. In figure 1 (b) we have plotted the same functions but removing the function for $\mathcal{L}_i^{(3)}$, to see more clearly the relative values of the functions. As it can be seen, the amplifying factor increases as the order of the starting algorithm increases (for

example for $z = -50$ $\mathcal{L}_i^{(3)}$ amplifies the error by a factor of about 20 while $\mathcal{L}_i^{(0)}$ reduces the error by factor of about -0.05). Therefore, it is not surprising that for large tolerances, where the errors are allowed to be large, the stability can be determinant. Note that the internal stages of the Runge-Kutta method are not exactly calculated at each step, and the propagation of accumulated errors (iteration errors, round-off errors,etc) through the starting algorithm may seriously affect to the accuracy of the numerical solution obtained by the iteration process.

These problems are not the only examples where low order starting algorithms work much better than higher order ones . Thus, for problem E5 (see [15, p. 145]) and for the *Ring modulator* problem [17] similar results are encountered when integrating these problems with large tolerances.

We must remark that this stability analysis, based on a linear model does not provide a rigourous explanation of the behaviour of the code with these differential problems, that are nonlinear (further investigation in this line is being carried out). However, it can serve as a first approximation for a better treatment of the problem.

On the other hand, for low tolerances all the above starting algorithms work properly, and it can be appreciated in table 1.1 that the higher order ones perform quite better than those of a lower order. This also happens on most stiff problems proposed as tests in [9,15,17].

In [14] a stabilisation of the highest order starting algorithms considered in [12] was carried out, trying to obtain starting algorithms with order as high as possible but maintaining good stability properties. However, the algorithms developed there, even though are more efficient than $\mathcal{L}_i^{(3)}$ for the Radau IIA formula of order 5, are not able to make the integration code robust enough to integrate problems like that of Robertson for large error tolerances. Unfortunately, in order to get better stability properties (with minimum additional cost) we need to decrease the order, getting the better stability for the initial guess of order 0.

After these results and from previous investigations [18], we have concluded that, instead of using a predetermined fixed starting algorithm it would be more efficient using a family of starting algorithms with different orders and stability properties. The key point is to develop a technique than selects, at each step, the most adequate starting algorithm, that is, the one providing the smallest error.

In this paper we will focus first in developing a simple family of starting algorithms with increasing orders, but of course, with decreasing stability properties so that we can select at each step the one that, having enough stability,

has the highest order possible. We will consider orders from 0 to $s+1$ (in this last case we need to use the information of the two integration steps) for an s–stage Runge–Kutta method. Then, we will present a technique for the automatic selection of the most adequate first iterant.

In section 2 some order results (non–stiff order, order for the Prothero & Robinson test equation and stiff order) for starting algorithms based on the information of the two last integration steps. In section 3, for s–stages collocation Runge–Kutta methods, we give a family of starting algorithms with orders ranging from 0 to $s+1$, based on Lagrange interpolation, that can be expressed in a simple way in terms of divided differences. The one of order $s+1$, with better stability properties than the one proposed in [12], is defined using the information in the last two integration steps and can also be computed using divided differences. Finally, in section 4, we present an strategy for the selection of the starting algorithm to be used at each particular step. It is based on the construction, for each first iterant, of an associated error estimator. In this section we also present some numerical experiments testing the performance of a code using the strategy presented, showing that it make the codes more robust and efficient.

2 Order results for two–step starting algorithms

In this section we introduce the order theory for starting algorithms based on the two last integration steps. Thus, we will assume that a first step of the Runge-Kutta method, with non–singular coefficient matrix, given by the equations (1.2)-(1.3) together with a second step, from $t_{n+1} \to t_{n+2} = t_{n+1} + h_{n+1}$, given by the equations

$$Y_{i,n+1} = y_{n+1} + h_{n+1} \sum_{j=1}^{s} a_{ij} f(t_{n+1} + c_j h_{n+1}, Y_{j,n+1}) \quad (i = 1, \ldots, s), \qquad (2.1)$$

and

$$y_{n+2} = \omega y_{n+1} + \sum_{j=1}^{s} w_j Y_{j,n+1}, \qquad (2.2)$$

have been carried out.

In order to start the iterative process for the new integration step, $t_{n+2} \to t_{n+3} = t_{n+2} + h_{n+2}$, we consider starting algorithms that can be written as,

$$Y^0_{i,n+2} = \sum_{j=1}^{s} u^{[n]}_{ij} Y_{j,n} + \sum_{j=1}^{s} u^{[n+1]}_{ij} Y_{j,n+1}, \quad 1 \le i \le s, \qquad (2.3)$$

where the scalars

$$u_{ij}^{[n+l]} := u_{ij}^{[n+l]}(r_n, r_{n+1}, A, b), \quad l = 0, 1, \quad 1 \leq i, j \leq s,$$

may depend on the coefficients of the Runge-Kutta method, and on r_n, r_{n+1} (here and henceforth $r_j = h_{j+1}/h_j$, $j = 0, 1, \ldots$). Moreover, $u_{ij}^{[n+l]}$ must be uniformly bounded when

$$0 < r_n,\ r_{n+1} \leq r^*, \quad r^* > 1. \tag{2.4}$$

Note that we have not used the values of the advancing solutions, y_n, y_{n+1}. This is motivated by the propagation of initial errors on the simple test

$$y' = \lambda y, \qquad y(t_n) = y_n, \qquad \mathrm{Re}(\lambda) \leq 0. \tag{2.5}$$

For this problem, the internal stages $Y_{i,n+2}$ at the step $t_{n+2} \to t_{n+3}$, are given by

$$Y_{i,n+2} = R(\lambda h_n) R(\lambda h_{n+1}) R_i(\lambda h_{n+2}) y_n, \quad (i = 1, \ldots, s), \tag{2.6}$$

where $R(z) = 1 + z b^T (I - zA)^{-1} \mathbb{1}_s$ is the stability function of the RK method and $R_i(z) = e_{i,s}^T (I - zA)^{-1} \mathbb{1}_s$, $(i = 1, \ldots, s)$. Here, as usual $\mathbb{1}_k = (1, \ldots, 1) \in \mathbb{R}^k$ and $e_{i,k}$ denotes the i-vector of the canonical base of $\mathbb{R}^k$.

On the other hand, the starting algorithm (2.3) applied to (2.5) gives,

$$Y_{i,n+2}^0 = \left(\sum_{j=1}^{s} u_{ij}^{[n]} R_j(\lambda h_n) + R(\lambda h_n) \sum_{j=1}^{s} u_{ij}^{[n+1]} R_j(\lambda h_{n+1}) \right) y_n, \quad 1 \leq i \leq s. \tag{2.7}$$

Hence, by assuming that the Runge-Kutta matrix A is nonsingular, we have

$$\lim_{Re(\lambda) \to -\infty} Y_{i,n+2} = 0 = \lim_{Re(\lambda) \to -\infty} Y_{i,n+2}^0, \quad (i = 1, \ldots, s) \tag{2.8}$$

independently on the values of h_n, h_{n+1}, t_n and y_n. This means that the starting algorithm cushions the very stiff components, as the underlying Runge-Kutta method does, and this behaviour can not be guaranteed, for non–stiffly accurate methods, if the starting algorithm uses the values y_n, y_{n+1}. This property can be immediately generalised for stiff linear systems.

In order to develop the order theory for the starting algorithms, we can consider the internal stages $Y_{i,n+2}$ as obtained from the application of a Runge-Kutta method with coefficient matrix $\hat{A}$, and whose Butcher tableau is given

by

$$\begin{array}{c|c} \hat{c} & \hat{A} \\ \hline Y_{i,n+2} & e_{2s+i,3s}^T \hat{A} \end{array} = \begin{array}{c|ccc} c & A & \mathbf{0} & \mathbf{0} \\ \mathbb{1}_s + r_n c & \mathbb{1}_s b^T & r_n A & \mathbf{0} \\ (1+r_n)\mathbb{1}_s + r_{n+1} c & \mathbb{1}_s b^T & r_n \mathbb{1}_s b^T & r_n r_{n+1} A \\ \hline Y_{i,n+2} & b^T & r_n b^T & r_n r_{n+1} e_{i,s}^T A \end{array} . \tag{2.9}$$

Henceforth, $h = h_n$ will denote the first step-size of the three consecutive steps to be considered $\{h_n, h_{n+1}, h_{n+2}\}$, and $\bar{h} = h_n + h_{n+1} + h_{n+2}$. Of course, we assume that $A\mathbb{1}_s = c$.

We will denote by $\tau \geq 1$ the stage order of the underlying Runge-Kutta (A, b), that is the largest integer such that

$$b^T c^{j-1} = 1/j \quad \text{and} \quad Ac^{j-1} = c^j/j, \qquad j = 1, \ldots, \tau, \tag{2.10}$$

where the powers of a vector are defined as usual taking powers on each component. It is not difficult to prove that (2.10) is equivalent to

$$\hat{A}\hat{c}^{j-1} = \hat{c}^j/j, \quad j = 1, \ldots, \tau. \tag{2.11}$$

Following the approach in [16], we will say that the starting algorithm (2.3) attains *non-stiff order* q if

$$\| Y_{i,n+2} - Y_{i,n+2}^0 \| \leq C h^{q+1}, \quad 1 \leq i \leq s, \tag{2.12}$$

for all $h \in (0, h^*]$, $0 < r_n$, $r_{n+1} \leq r^*$, $r^* > 1$ when the elementary differentials of the successive derivatives of f (up to order q) are uniformly bounded in some ball around the point (t_n, y_n).

Following the approach of [12] we will say that the starting algorithm possesses *stiff order* q if (2.12) holds, independently of the stiffness, provided that the differential system is dissipative (with respect to the norm associated to an inner product on $\mathbb{R}^m$) [7, Chap. I], and the successive derivatives (until order $q+1$) of the local exact solution $y(t; t_n, y_n)$ considered, are uniformly bounded on some interval $[t_n, t_n + \delta]$ with $\delta > 0$.

If the starting algorithm reaches stiff order q for the Prothero and Robinson test (1.5), then we will say that it possesses *P-R order* q. Moreover, since the P-R order when $\mathrm{Re}(\lambda) \to -\infty$ will be relevant for the very stiff components of linear differential systems, we will denote the order in this particular situation by means of P-R(∞).

Note that although these concepts of order might appear much as the same,

they are quite different. Thus, if we consider the 1-stage Runge-Kutta Gauss,

$$y_{n+1} = y_n + h_n f(t_n + h_n/2, Y_{1,n}),$$
$$Y_{1,n} = y_n + h_n/2\ f(t_n + h_n/2, Y_{1,n}), \quad n = 0, 1, \ldots$$

and we take the following starting algorithm (which does not belong to the class previously considered in (2.3), but it is a very simple starting algorithm)

$$Y^0_{1,n+2} = y_{n+2} + \frac{h_{n+2}}{2} f(t_{n+2}, y_{n+2}), \tag{2.13}$$

it is not difficult to prove that it has non-stiff order 1. However, for the Prothero & Robinson test with $\Phi(t) = t^2$ and $y_n = \Phi(t_n)$, we have that $\lim_{\lambda\to-\infty} |Y^0_{1,n+2}| = \infty$, if $r_n \neq 1$, whereas $Y_{1,n+2} = \phi(t_{n+2} + h_{n+2}/2) = (t_{n+2} + h_{n+2}/2)^2$. Therefore P-R($\infty$) is not defined (it cannot achieve even order 0).

In order to simplify the exposition we use throughout the rest of the paper the following notation. Given the vectors

$$Z_{i,n} \in \mathbb{R}^m, \quad 1 \leq i \leq s, \quad n = 0, 1 \ldots,$$

we will denote

$$Z_n := \begin{pmatrix} Z_{1,n} \\ \vdots \\ Z_{s,n} \end{pmatrix} \in \mathbb{R}^{sm}, \qquad f(Z_n) := \begin{pmatrix} f(t_n + c_1 h_n, Z_{1,n}) \\ \vdots \\ f(t_n + c_s h_n, Z_{s,n}) \end{pmatrix} \in \mathbb{R}^{sm},$$

$$\widetilde{Z}_n := \begin{pmatrix} Z_n \\ Z_{n+1} \\ Z_{n+2} \end{pmatrix} \in \mathbb{R}^{3sm}, \qquad f(\widetilde{Z}_n) := \begin{pmatrix} f(Z_n) \\ f(Z_{n+1}) \\ f(Z_{n+2}) \end{pmatrix} \in \mathbb{R}^{3sm}.$$

Although we are using $f(\ )$ to represent several kind of functions, it will be clear from the context which mapping must be applied.

According to our notation, the starting algorithm (2.3) can be written as

$$Y^0_{n+2} = (U^{[n]} \otimes I_m) Y_n + (U^{[n+1]} \otimes I_m) Y_{n+1}, \tag{2.14}$$

where the $(s \times s)$ matrices $U^{[n+l]}$ are defined by

$$U^{[n+l]} := (u^{[n+l]}_{ij}) \in \mathbb{R}^{s,s}, \quad l = 0, 1,$$

and

$$Y^0_{n+2} := \begin{pmatrix} Y^0_{1,n+2} \\ \vdots \\ Y^0_{s,n+2} \end{pmatrix}.$$

In order to deduce the order conditions we will use the exact local solution of the differential system, $x(t) = y(t; t_n, y_n)$, and we will denote

$$X_{i,n+j} := x(t_{n+j} + c_i h_{n+j}), \qquad 1 \le i \le s, \quad j = 0, 1, 2.$$

The internal stages of the RK method (A, b) on three consecutive steps ($h = h_n, h_{n+1}, h_{n+2}$) fulfill the algebraic system,

$$\widetilde{Y}_n = \mathbb{1}_{3s} \otimes y_n + h(\hat{A} \otimes I_m) f(\widetilde{Y}_n), \tag{2.15}$$

and the local exact solution (I_m denotes the identity matrix of order m) satisfies

$$\widetilde{X_n} = \mathbb{1}_{3s} \otimes y_n + h(\hat{A} \otimes I_m) f(\widetilde{X_n}) + \Delta, \tag{2.16}$$

where

$$\Delta = \sum_{k \ge \tau+1} \frac{h^k}{k!} (\hat{\gamma}_k \otimes I_m) x^{(k)}(t_n), \quad \hat{\gamma}_k := (\hat{c}^k - k\hat{A}\hat{c}^{k-1}). \tag{2.17}$$

Now, by subtracting (2.16) from (2.15) it follows that

$$\tilde{Y}_n - \tilde{X}_n = h(\hat{A} \otimes I_m) \mathbf{J}_n (\tilde{Y}_n - \tilde{X}_n) - \Delta, \tag{2.18}$$

where

$$\begin{aligned} &\mathbf{J}_n = \mathrm{diag}((J_{i,n})_{i=1,\dots,s}, (J_{i,n+1})_{i=1,\dots,s}, (J_{i,n+2})_{i=1,\dots,s}), \\ &J_{i,n+l} := \int_0^1 \frac{\partial f}{\partial y}(t_{n+l} + c_i h_{n+l}, X_{i,n+l} + \theta(Y_{i,n+l} - X_{i,n+l}))d\theta. \end{aligned} \tag{2.19}$$

By assuming that $(I_{3sm} - h(\hat{A} \otimes I_m)\mathbf{J}_n)$ is nonsingular, it follows from (2.18) that

$$\tilde{Y}_n = \tilde{X}_n - \Upsilon, \qquad \Upsilon = (I_{3sm} - h(\hat{A} \otimes I_m)\mathbf{J}_n)^{-1}\Delta. \tag{2.20}$$

And from here,

$$Y_{n+l} = X_{n+l} - (e^T_{l+1,3} \otimes I_{sm})\Upsilon, \quad l = 0, 1, 2. \tag{2.21}$$

By inserting this equation into (2.14), it follows after some simple calculations that

$$\begin{aligned} Y_{n+2} - Y^0_{n+2} = X_{n+2} - (U^{[n]} \otimes I_m) X_n - (U^{[n+1]} \otimes I_m) X_{n+1} \\ + ((U^{[n]}, U^{[n+1]}, -I_s) \otimes I_m)\Upsilon, \end{aligned} \tag{2.22}$$

where $(U^{[n]}, U^{[n+1]}, -I_s)$ denotes the $s \times 3s$ matrix whose first s columns are those of $U^{[n]}$, the following s columns are those of $U^{[n+1]}$ and the last s columns those of $-I_s$. Now, we can rewrite the last equation in the following compact form

$$Y_{n+2} - Y^0_{n+2} = ((U^{[n]}, U^{[n+1]}, -I_s) \otimes I_m)(-\tilde{X}_n + \Upsilon). \tag{2.23}$$

On the other hand, it is straightforward to see that

$$\tilde{X}_n = \sum_{l \geq 0} \frac{h^l}{l!} (\hat{c}^l \otimes I_m) x^{(l)}(t_n). \tag{2.24}$$

Equations (2.23) and (2.24) will let us deduce the order conditions.

2.1 Non-stiff order

For non-stiff problems we have that,

$$\begin{aligned} (I_{3sm} - h(\hat{A} \otimes I_m)\mathbf{J}_n)^{-1} &= I_{3sm} + h(\hat{A} \otimes I_m)\mathbf{J}_n + \mathcal{O}(h^2) \\ &= I_{3sm} + h(\hat{A} \otimes J_n) + \mathcal{O}(h^2), \end{aligned} \tag{2.25}$$

where $J_n = \frac{\partial f}{\partial y}(t_n, y_n)$. Hence, from (2.20) and (2.17) it follows that,

$$\begin{aligned} \Upsilon = \frac{h^{\tau+1}}{(\tau+1)!}(\hat{\gamma}_{\tau+1} \otimes I_m) x^{(\tau+1)}(t_n) + \frac{h^{\tau+2}}{(\tau+2)!}((\hat{\gamma}_{\tau+2} \otimes I_m) x^{(\tau+2)}(t_n) \\ +(\tau+2)(\hat{A}\hat{\gamma}_{\tau+1} \otimes I_m) J_n x^{(\tau+1)}(t_n)) + \mathcal{O}(h^{\tau+3}). \end{aligned} \tag{2.26}$$

From (2.17) it is clear that

$$-\hat{c}^k + \hat{\gamma}_k = -k\hat{A}\hat{c}^{k-1}, \quad k = 1, 2, \ldots$$

Thus from (2.26) and (2.24) we have

$$\begin{aligned} -\tilde{X}_n + \Upsilon = &-\sum_{l=0}^{\tau} \frac{h^l}{l!}(\hat{c}^l \otimes I_m) x^{(l)}(t_n) \\ &-\frac{h^{\tau+1}}{\tau!}(\hat{A}\hat{c}^\tau \otimes I_m) x^{(\tau+1)}(t_n) - \frac{h^{\tau+2}}{(\tau+1)!}(\hat{A}\hat{c}^{\tau+1} \otimes I_m) x^{(\tau+2)}(t_n) \\ &-\frac{h^{\tau+2}}{(\tau+1)!}(((\tau+1)\hat{A}^2\hat{c}^\tau - \hat{A}\hat{c}^{\tau+1}) \otimes I_m) J_n x^{(\tau+1)}(t_n) + \mathcal{O}(h^{\tau+3}). \end{aligned} \tag{2.27}$$

Now we are in position of giving the following theorem, which directly follows from (2.27) and (2.23).

Theorem 1 (a) *The starting algorithm reaches non-stiff order $q \leq \tau$ iff*

$$(U^{[n]}, U^{[n+1]}, -I_s)\hat{c}^l = 0, \quad l = 0, 1, \ldots, q.$$

In such a case the error satisfies

$$Y_{n+2} - Y^0_{n+2} = -\frac{h^{q+1}}{q!}((U^{[n]}, U^{[n+1]}, -I_s)\hat{A}\hat{c}^q \otimes I_m)x^{(q+1)}(t_n) + \mathcal{O}(h^{q+2}).$$

(b) *The starting algorithm reaches non-stiff order $\tau + 1$ iff it reaches non-stiff order τ and*

$$(U^{[n]}, U^{[n+1]}, -I_s)\hat{A}\hat{c}^\tau = 0.$$

In such a case the error satisfies

$$Y_{n+2} - Y^0_{n+2} = -\frac{h^{\tau+2}}{(\tau+1)!}((U^{[n]}, U^{[n+1]}, -I_s) \otimes I_m)((\hat{A}\hat{c}^{\tau+1} \otimes I_m)x^{(\tau+2)}(t_n)$$

$$+(((\tau+1)\hat{A}^2\hat{c}^\tau - \hat{A}\hat{c}^{\tau+1}) \otimes I_m)J_n x^{(\tau+1)}(t_n)) + \mathcal{O}(h^{\tau+3}).$$

(c) *The starting algorithm reaches non-stiff order $\tau + 2$ iff it reaches non-stiff order $\tau + 1$ and*

$$(U^{[n]}, U^{[n+1]}, -I_s)\hat{A}\hat{c}^{\tau+1} = 0, \qquad (U^{[n]}, U^{[n+1]}, -I_s)\hat{A}^2\hat{c}^\tau = 0.$$

2.2 The Prothero and Robinson order

Let us consider now the order of the starting algorithms on the Prothero and Robinson model (1.5) being λ any complex in the left half-plane. In this case, since $m = 1$ and

$$(I_{3sm} - h(\hat{A} \otimes I_m)\mathbf{J}_n)^{-1} = (I_{3s} - z\hat{A})^{-1}, \qquad z = \lambda h,$$

from (2.20) and (2.17) we get,

$$\Upsilon = (I_{3s} - z\hat{A})^{-1} \sum_{k \geq \tau+1} \frac{h^k}{k!}\hat{\gamma}_k x^{(k)}(t_n). \tag{2.28}$$

In order to analyse the order for this test equation, we must ensure (by definition of stiff order) that the successive derivatives of the exact local solution $x(t)$ are uniformly bounded independently of λ. Then we must consider the case $y_n = \phi(t_n)$, i.e.,

$$x(t) = \phi(t).$$

From (2.28) we have

$$\Upsilon = (I_{3s} - z\hat{A})^{-1} \sum_{k \geq \tau+1} \frac{h^k}{k!} \hat{\gamma}_k \phi^{(k)}(t_n). \tag{2.29}$$

and the expression (2.22) becomes

$$\begin{aligned} Y_{n+2} - Y_{n+2}^0 = (U^{[n]}, U^{[n+1]}, -I_s) \Bigg(& -\sum_{l \geq 0} \frac{h^l}{l!} \hat{c}^l \phi^{(l)}(t_n) \\ & + (I_{3s} - z\hat{A})^{-1} \sum_{k \geq \tau+1} \frac{h^k}{k!} \hat{\gamma}_k \phi^{(k)}(t_n) \Bigg). \end{aligned} \tag{2.30}$$

From here it is straightforward to prove the following theorem when

$$\sup_{Re(z) \leq 0} \| \, (I_{3s} - z\hat{A})^{-1} \, \|_2 < \infty.$$

Theorem 2 (a) *Let us assume that*

$$\sup_{Re(z) \leq 0} \| \, (I_s - zA)^{-1} \, \|_2 < \infty, \quad \textit{and} \quad \sup_{Re(z) \leq 0} \| \, zb^T (I_s - zA)^{-1} \, \|_2 < \infty.$$

Then the starting algorithm reaches P-R order $q \leq \tau$ iff

$$(U^{[n]}, U^{[n+1]}, -I_s)\hat{c}^l = 0, \quad l = 0, 1, \ldots, q.$$

(b) *If A is a nonsingular matrix, then the starting algorithm reaches P-R(∞) order $q > 0$ iff*

$$(U^{[n]}, U^{[n+1]}, -I_s)\hat{c}^l = 0, \quad l = 0, 1, \ldots, q.$$

In this case its error is given by

$$Y_{n+2} - Y_{n+2}^0 = -(U^{[n]}, U^{[n+1]}, -I_s) \sum_{l \geq q+1} \frac{h^l}{l!} \hat{c}^l \phi^{(l)}(t_n).$$

Note that for most implicit Runge–Kutta methods of practical interest the P–R order $\tau + 1$ can not be achieved because of the z–dependence of the $h^{\tau+1}$ term.

Let us consider dissipative problems [7, Chap.I], i.e., for some inner product $\langle \cdot, \cdot \rangle$ on $\mathbb{R}^m$ the derivative function f satisfies

$$\langle y - z, f(t, y) - f(t, z) \rangle \leq 0, \quad \forall t, y, z.$$

If the underlying Runge-Kutta (A, b) is diagonally stable (i.e., there exists a positive definite diagonal matrix D such that $DA + A^T D$ is positive definite) and

$$0 \leq c_j \leq 1, \quad 1 \leq j \leq s, \tag{2.31}$$

it is not difficult to prove that (use for instance [15, Chap. IV.14, theor. 14.3]), for $l = 0, 1, 2$,

$$\| Y_{i,n+l} - X_{i,n+l} \| \leq C_{\tau+1} h^{\tau+1} \max_{0 \leq \theta \leq 1} \| x^{(\tau+1)}(t_n + \theta \bar{h}) \|, \quad 1 \leq i \leq s, \tag{2.32}$$

where the constant $C_{\tau+1}$ depends only on the coefficients of the Runge-Kutta method and on r^*. If the knots do not fulfill (2.31) we can still prove (2.32), but with θ ranging on another interval.

Assuming that the coefficients $u_{ij}^{[n+l]}$ are uniformly bounded when $0 < r_n, r_{n+1} \leq r^*$, we have that

$$Y_{n+2} - Y_{n+2}^0 = X_{n+2} - (U^{[n]} \otimes I_m) X_n - (U^{[n+1]} \otimes I_m) X_{n+1} + \mathcal{O}(h)^{\tau+1}. \tag{2.33}$$

This implies that the conditions for stiff order $q \leq \tau$, are the same as for non-stiff order $q \leq \tau$, which are collected in theorem 1 (a). Moreover the stiff order $\tau + 1$ cannot be reached, since it cannot be even achieved on the Prothero and Robinson model.

One might wonder whether the stiff order of the starting algorithms is relevant for stiff problems, since in the analysis of the stiff order we have assumed that the exact local solution $x(t) = y(t; t_n, y_n)$ must possess uniformly bounded derivatives, and this is not the case when considering the whole class of dissipative problems and we advance in the integration of the problem with some Runge-Kutta method. However, the following theorem assures that the stiff order will be relevant at least for algebraically stable Runge-Kutta methods.

Theorem 3 *Let us assume that the initial value problem (1.1) is dissipative and its exact solution $y(t) = y(t; 0, y_0)$ possesses bounded derivatives until order $\tau + 1$, i.e.,*

$$\| y^{(l)}(t) \| \leq M_l, \quad l = 1, 2 \ldots, \tau + 1, \quad \forall t \in [0, T].$$

Let us also assume that the underlying Runge-Kutta is diagonally stable with stage order τ, and that the starting algorithm (2.3) possesses stiff order $q \leq \tau$.

Then,

$$\| Y_{i,n+2} - Y^0_{i,n+2} \| \leq K_1 \| \varepsilon_n \| + K_2 (h_n)^{q+1}, \quad 1 \leq i \leq s,$$

where the constants $K_j\,(j = 1, 2)$ only depend on the coefficients of the Runge-Kutta (A, b), on the constant M_{q+1} and on r^*. Here, $\varepsilon_n = y(t_n) - y_n$, represents the global error of the numerical solution given by the Runge-Kutta method after n consecutive steps.

PROOF. Let us denote by

$$Y_{i,n+l}\{y(t_n)\}, \quad 1 \leq i \leq s, \quad l = 0, 1, 2,$$

the internal Runge-Kutta stages when the Runge-Kutta (A, b) is applied from the point $(t_n, y(t_n))$ and after giving three consecutive steps of sizes h_n, h_{n+1} and h_{n+2} respectively, and let us also denote by

$$Y^0_{i,n+2}\{y(t_n)\}, \quad 1 \leq i \leq s,$$

the resulting starting algorithm. The error of the starting algorithm can be written in the form

$$\begin{aligned} Y_{i,n+2} - Y^0_{i,n+2} = &(Y_{i,n+2} - Y_{i,n+2}\{y(t_n)\}) \\ &+ (Y_{i,n+2}\{y(t_n)\} - Y^0_{i,n+2}\{y(t_n)\}) \\ &+ (Y^0_{i,n+2}\{y(t_n)\} - Y^0_{i,n+2}). \end{aligned} \tag{2.34}$$

Now, from the diagonal stability of the Runge-Kutta method (see e.g. [15, Chap. IV.14, theor. 14.3]) the first and the third addendum on the right–hand side of (2.34) are both bounded by $K_1 \| \varepsilon_n \|$ and the bound for the second addendum follows immediately from the stiff order q of the starting algorithm. □

Note that for algebraically stable Runge-Kutta methods the global error ε_n can be bounded in the form

$$\| \varepsilon_n \| \leq K_3 (\max_{0 \leq k \leq n-1} h_k)^\tau,$$

where K_3 is a constant that only depends on $M_{\tau+1}$ and on the Runge-Kutta coefficients.

3 Starting algorithms based on divided differences for collocation Runge-Kutta methods

In the rest of the paper we will assume that the nodes of the collocation Runge-Kutta method fulfill

$$0 < c_1 < c_2 < \ldots < c_s \leq 1. \tag{3.1}$$

Observe that in the case we are considering, we can replace the stage order τ by s in order to apply the preceding theory. Besides, (3.1) implies

$$0 < \hat{c}_1 < \hat{c}_2 < \ldots < \hat{c}_{3s}. \tag{3.2}$$

We will denote by $\mathcal{L}_n[\hat{c}_{j_1}, \ldots, \hat{c}_{j_q}](\theta)$, the Lagrange interpolating polynomial satisfying

$$\mathcal{L}_n[\hat{c}_{j_1}, \ldots, \hat{c}_{j_q}](\hat{c}_{j_l}) = \tilde{Y}_{j_l,n}, \quad 1 \leq l \leq q,$$

where $\tilde{Y}_{k,n}$ denotes the k-component of the vector $\tilde{Y}_n$.

Among the starting algorithms of type (2.3) we will mainly consider here those that can be written in the form

$$Y^0_{i,n+2} = \mathcal{L}_n[\hat{c}_{j_1}, \ldots, \hat{c}_{j_q}](\hat{c}_{2s+i}), \quad 1 \leq i \leq s. \tag{3.3}$$

because they can be easily implemented and they provide adequate order and stability properties.

Remark 4 *If we define recursively, as usual, the divided differences for the internal stages of the composed Runge-Kutta method by*

$$[Y_{j_1}] := \tilde{Y}_{j_1,n},$$

$$[Y_{j_1}, \ldots, Y_{j_l}] := \frac{1}{\hat{c}_{j_l} - \hat{c}_{j_1}} \left([Y_{j_2}, \ldots, Y_{j_l}] - [Y_{j_1}, \ldots, Y_{j_{l-1}}] \right), \quad l = 2, \ldots, 3s, \tag{3.4}$$

we can rewrite the Lagrange interpolating polynomial above as

$$\mathcal{L}_n[\hat{c}_{j_1}, \ldots, \hat{c}_{j_q}](\theta) = [Y_{j_1}] + \sum_{l=2}^{q} [Y_{j_1}, \ldots, Y_{j_l}](\theta - \hat{c}_{j_1}) \cdots (\theta - \hat{c}_{j_{l-1}}). \tag{3.5}$$

On the other hand, if we define the corresponding divided differences for the local exact solution $x(t)$,

$$[X_{j_1}] := \tilde{X}_{j_1,n},$$

$$[X_{j_1}, \ldots, X_{j_l}] := \frac{1}{\hat{c}_{j_l} - \hat{c}_{j_1}} \left([X_{j_2}, \ldots, X_{j_l}] - [X_{j_1}, \ldots, X_{j_{l-1}}] \right), \quad l = 2, \ldots, 3s,$$

then from equation (2.32) we get

$$[X_{j_l}] - [Y_{j_l}] = \mathcal{O}(h^{s+1}),$$

which implies

$$[Y_{j_1}, \ldots, Y_{j_q}] = [X_{j_1} + \mathcal{O}(h^{s+1}), \ldots, X_{j_q} + \mathcal{O}(h^{s+1})] = [X_{j_1}, \ldots, X_{j_q}] + \mathcal{O}(h^{s+1}).$$

Moreover according to some well known expression for the $q-1$ divided difference in terms of the $q-1$ derivatives, we have that

$$[X_{j_1}, \ldots, X_{j_q}] = \frac{h^{q-1}}{(q-1)!} x^{(q-1)}(t_n + \theta_q h), \quad 0 < \theta_q < r_n r_{n+1}.$$

Consequently,

$$[Y_{j_1}, \ldots, Y_{j_q}] = \mathcal{O}(h^{q-1}) + \mathcal{O}(h^{s+1}), \quad q \geq 2. \tag{3.6}$$

The order of the starting algorithm (3.3) is stated in the following theorem.

Theorem 5 (a) *If $1 \leq q \leq s+1$ and $\{j_1, \ldots, j_q\} \subseteq \{1, \ldots, 2s\}$, then the starting algorithm given by (3.3) has stiff and non-stiff order $q-1$.*
(b) *If $1 \leq q \leq 2s$ and $\{j_1, \ldots, j_q\} \subseteq \{1, \ldots, 2s\}$, then the starting algorithm (3.3) reaches P-R(∞) order $q-1$.*

PROOF. (a) For the stiff and non-stiff cases we have

$$Y_{i,n+l} - X_{i,n+l} = \mathcal{O}(h^{s+1}), \quad 1 \leq i \leq s, \quad l = 0, 1, 2, \tag{3.7}$$

where the term $\mathcal{O}(h^{s+1})$ is not affected by the stiffness if we assume that the derivatives of $x(t)$ until order $s+1$ are uniformly bounded.

From (2.23) and (2.27) for the non-stiff case and from (2.32) for the stiff case, it follows that

$$Y_{n+2} - Y^0_{n+2} = X_{n+2} - (U^{[n]} \otimes I_m) X_n - (U^{[n+1]} \otimes I_m) X_{n+1} + \mathcal{O}(h^{s+1}). \tag{3.8}$$

Now, from the interpolation conditions

$$X_{n+2} - (U^{[n]} \otimes I_m) X_n - (U^{[n+1]} \otimes I_m) X_{n+1} = \mathcal{O}(h^q,)$$

and inserting this expression in (3.8) we get

$$Y_{n+2} - Y^0_{n+2} = \mathcal{O}(h^q) + \mathcal{O}(h^{s+1}) = \mathcal{O}(h^q),$$

which proves the first part of the theorem.

(b) By considering the Prothero and Robinson model (1.5) and making $Re(\lambda) \to -\infty$, it follows from (2.20)-(2.21) that

$$Y_{i,n+l} = \phi(t_{n+l} + c_i h_{n+l}), \quad 1 \le i \le s, \quad l = 0, 1, 2.$$

From here, by applying the well known polynomial interpolation error formula,

$$Y_{i,n+2} - \mathcal{L}_n[\hat{c}_{j_1}, \ldots, \hat{c}_{j_q}](\hat{c}_{2s+i}) = \frac{h^q}{q!} \Pi_{k=1}^q (\hat{c}_{2s+i} - \hat{c}_{j_k}) \cdot \phi^{(q)}(t_n + \theta_i \bar{h}), \quad 1 \le i \le s$$

for some $\theta_i \in (0, 1)$. □

From the order theory in section 2, it is clear that we can not obtain starting algorithms with P–R order higher than s, and therefore with stiff order $s+1$ or higher. By means of the Lagrange interpolation we can get P–R(∞) order $s+1$ or greater (for example $\mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_{s-1}](\hat{c}_{2s+i})$ has P–R(∞) order $s+1$). The goal now is to get starting algorithms with stiff order s and non stiff order greater or equal than $s+1$ (we will assume $s \ge 2$).

In [12] an algorithm with non–stiff order $s+1$ that uses the evaluation of the derivative function $f(t_{n+1}, y_{n+1})$ is studied. This algorithm has the disadvantage that its amplifying function is not bounded when z goes to infinity and the authors do not recommend its usage if the Runge-Kutta method is not stiffly accurate. Here we present another algorithm with non–stiff order $s+1$ for which the amplifying function is bounded, and that makes use of the internal stages $\tilde{Y}_{s-1,n}, \tilde{Y}_{s,n}, \ldots, \tilde{Y}_{2s,n}$.

From theorem 1, part (b), such an starting algorithm must satisfy

$$\begin{gathered} (U^{[n]}, U^{[n+1]}, -I_s)\hat{c}^j = 0, \quad j = 0, \ldots, s \\ (U^{[n]}, U^{[n+1]}, -I_s)\hat{A}\hat{c}^s = 0. \end{gathered} \tag{3.9}$$

and in order to assure its existence we must demand

$$\rho(r_n) := \det M \neq 0, \qquad M = \begin{pmatrix} 1 & \hat{c}_{s-1} & \cdots & (\hat{c}_{s-1})^s & \hat{v}_{s-1} \\ \vdots & & & & \vdots \\ 1 & \hat{c}_{2s} & \cdots & (\hat{c}_{2s})^s & \hat{v}_{2s} \end{pmatrix}, \tag{3.10}$$

where $\hat{v} = \hat{A}\hat{c}^s$. Observe that $\rho(r_n)$ is a polynomial on r_n with degree $s(s+3)/2$ at most. It would be convenient that the roots of ρ are out of the interval $(0, r^*]$.

The following theorem give us such an starting algorithm by means of the divided differences.

Theorem 6 *If $\rho(r_n) \neq 0$, then there exist constants $\delta_i = \delta_i(r_n, r_{n+1})$, $1 \leq i \leq s$ such that the starting algorithm*

$$Y^0_{i,n+2} = \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i}) + \delta_i[Y_{2s}, Y_{2s-1}, \ldots, Y_{s-1}], \quad 1 \leq i \leq s,$$

reaches non-stiff order $s+1$ and stiff order s. Moreover, such constants are unique.

PROOF. Since this algorithm only uses the stages $\tilde{Y}_{j,n} \equiv [Y_j]$ with $s-1 \leq j \leq 2s$, then $u^{[n]}_{ij} = 0$, $1 \leq i \leq s$, $1 \leq j \leq s-2$ and it can be written in terms of the vector $\tilde{Y}_n$ as,

$$Y^0_{i,n+2} = \sum_{j=s-1}^{s} u^{[n]}_{ij} \tilde{Y}_{j,n} + \sum_{j=1}^{s} u^{[n+1]}_{ij} \tilde{Y}_{j,n}, \quad 1 \leq i \leq s.$$

Now, using the divided differences defined by (3.4) we can put,

$$Y^0_{i,n+2} = \beta_{i0}[Y_{2s}] + \sum_{j=1}^{s+1} \beta_{ij}[Y_{2s}, \ldots, Y_{2s-j}], \quad \beta_{ij} = \beta_{ij}(r_n, r_{n+1}), \quad 1 \leq i \leq s. \tag{3.11}$$

On the other hand from (3.5) we have that

$$Y^0_{i,n+2} - \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i}) = (\tau_{i0} - \beta_{i0})[Y_{2s}] + \sum_{j=1}^{s+1} (\tau_{ij} - \beta_{ij})[Y_{2s}, \ldots, Y_{2s-j}]$$

where

$$\tau_{ij} := (\hat{c}_{2s+i} - \hat{c}_{2s}) \cdots (\hat{c}_{2s+i} - \hat{c}_{2s-j+1}), \quad 1 \leq i, j \leq s$$
$$\tau_{i,0} = 1, \quad \tau_{i,s+1} = 0, \quad 1 \leq i \leq s.$$

Bearing in mind that the order conditions imply

$$Y^0_{i,n+2} - \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i}) = \mathcal{O}(h^{s+1}),$$

and that from (3.6)

$$[Y_{2s}, \ldots, Y_{2s-j}] = \mathcal{O}(h^j) + \mathcal{O}(h^{s+1}), \quad j \geq 0,$$

then we conclude that

$$\beta_{ij} = \tau_{ij}, \quad 1 \leq j \leq s, \ 0 \leq i \leq s.$$

This completes the proof. □

Next, we give an alternative formula for computing δ_i in the preceding theorem.

Theorem 7 *Assuming $\rho(r_n) \neq 0$, the constants $\{\delta_i\}_{i=1}^{s}$ in theorem 6 can be calculated by*

$$\delta_i = \left(z_{2s+i} - \mathcal{L}[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i})\right) / [z_{2s}, \ldots, z_{s-1}], \quad 1 \leq i \leq s, \tag{3.12}$$

where $\{z_i\}_{i=1}^{3s}$ denotes the internal Runge-Kutta stages of the Runge-Kutta composed method (with coefficient matrix $\hat{A}$), when it is applied to the scalar problem

$$y' = t^s,\ t_n = 0,\ y_n = 0,\ h = 1, \tag{3.13}$$

$\mathcal{L}[\hat{c}_{2s}, \ldots, \hat{c}_s](t)$ is the corresponding Lagrange interpolating polynomial, and $[z_{2s}, \ldots, z_{2s-j}]$ represents the corresponding divided difference.

PROOF. Observe that according to the previous theorem, the starting algorithm with non-stiff order $s+1$ for the problem (3.13) would be,

$$Y_{i,n+2}^0 = \mathcal{L}[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i}) + \delta_i [z_{2s}, z_{2s-1}, \ldots, z_{s-1}], \quad 1 \leq i \leq s.$$

Now, since $y^{(q)}(t_n) = 0,\ q \neq s+1$, from theorem 1 and using (2.22), (2.23), (2.20) and (2.17), we have for this particular problem that

$$Y_{i,n+2} = Y_{i,n+2}^0, \quad 1 \leq i \leq s.$$

Observe that in our notation, $Y_{i,n+2} = z_{2s+i}$. In this way, if $[z_{2s}, \ldots, z_{s-1}] \neq 0$, we achieve the formula (3.12).

In order to see that $\rho(r_n) \neq 0$ implies $[z_{2s}, \ldots, z_{s-1}] \neq 0$, we proceed by contradiction. Assume $[z_{2s}, \ldots, z_{s-1}] = 0$, then for arbitrary constants $\{\theta_i\}_{i=1}^{s}$ we have that

$$z_{2s+i} = \mathcal{L}[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i}) + \theta_i [z_{2s}, z_{2s-1}, \ldots, z_{s-1}], \quad 1 \leq i \leq s.$$

Now we consider the associated starting algorithm for arbitrary differential systems,

$$Y_{i,n+2}^0 = \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i}) + \theta_i [Y_{2s}, Y_{2s-1}, \ldots, Y_{s-1}], \quad 1 \leq i \leq s. \tag{3.14}$$

By using theorem 1 (a), we have that

$$Y_{i,n+1} - Y_{i,n+2}^0 = K_i y^{(s+1)}(t_n) h^{s+1} + \mathcal{O}(h^{s+2}), \quad 1 \leq i \leq s,$$

but the constants K_i, $i = 1, \ldots, s$ must vanish because this starting algorithm is exact for the particular problem (3.13). This would imply we have a family of starting algorithms of non-stiff order $s+1$, but this contradicts the theorem 6. □

4 Variable–order starting algorithm strategy and numerical experiments

In this section we propose a strategy for the integration code to choose automatically, at each step, the most convenient starting algorithm so that it has order as high as possible, but maintaining adequate stability properties. This technique works well independently of the iteration used to solve the implicit Runge-Kutta equations (typically, for stiff problems, a simplified Newton iteration [1,4], [15, Ch. IV.8] or some kind of Single-Newton iteration [5,6,10,11,13]). It is motivated by the theory previously presented and by numerical experiences carried out by the authors, and it is based on estimations of the error of the starting algorithms.

Let us assume that we have a family of starting algorithms, namely,

$$\{Y_{i,n+2}^{0,(l)}\}, \quad l = 0, 1, \dots p,$$

of consecutive orders, i.e.,

$$Y_{i,n+2}^{0,(l)} - Y_{i,n+2} = \mathcal{O}(h^{l+1}), \quad 1 \leq i \leq s, \quad l = 0, 1, \dots p.$$

Since

$$Y_{i,n+2}^{0,(l)} - Y_{i,n+2} = Y_{i,n+2}^{0,(l)} - Y_{i,n+2}^{0,(l+1)} + \mathcal{O}(h^{l+2}), \quad 1 \leq i \leq s, \quad l = 0, \dots, p-1.$$

we can estimate the error of $Y_{i,n+2}^{0,(l)}$ by the difference $\parallel Y_{i,n+2}^{0,(l)} - Y_{i,n+2}^{0,(l+1)} \parallel$.

In order to save some computations, instead of calculating the estimators for every subindex i, we can compute the error estimator only for the last stage ($i = s$). In this way we can define

$$E_s^l = \parallel Y_{s,n+2}^{0,(l)} - Y_{s,n+2}^{0,(l+1)} \parallel \simeq \parallel Y_{s,n+2}^{0,(l)} - Y_{s,n+2} \parallel, \quad l = 0, \dots, p-1. \tag{4.1}$$

Since $E_s^l = \mathcal{O}(h^{l+1})$, the quotient $E_s^l / E_s^{l-1} = \mathcal{O}(h)$ should be expected to be strictly smaller than 1 and therefore $\{E_s^l\}$ a strictly decreasing sequence . Then, the use of the order l algorithm is advised if it has an error E_s^l smaller than that of order $l-1$, E_s^{l-1} by a factor $\theta < 1$. In case of similar errors, the lower order algorithm is preferred because it has better stability properties.

On the other hand, if $E_s^l < \theta E_s^{l-1}$, but $E_s^{l+1} \geq \theta E_s^l$, we can assume that the asymptotic theory for the order is not valid for the size of the step we are using, or well the errors are affected by stability. Then, the "large" value of $E_s^{l+1} = \parallel Y_{s,n+2}^{0,(l+2)} - Y_{s,n+2}^{0,(l+1)} \parallel$ could be due to a large error of the starting algorithm of order $l+2$, being in this case E_s^{l+1} an invalid estimation of the error of $Y_{s,n+2}^{0,(l+1)}$. If $E_s^l << E_s^{l-1}$, we can assume that the error of $Y_{s,n+2}^{0,(l+1)}$ is

smaller than the error of $Y^{0,(l)}_{s,n+2}$ (otherwise E^l_s would not be much smaller than E^{l-1}_s).

Recall that for the Prothero & Robinson problem the error of the starting algorithms satisfies (1.6) and therefore

$$E^{l+1}_s = \Big(R^{0,(l+2)}_s(h_n\lambda) - R^{0,(l+1)}_s(h_n\lambda)\Big)(y_n - \phi(t_n)) + \mathcal{O}(h_n^{l+2})$$

If there is no stability problems, that is, if the amplifying functions are small enough, a large value of E^{l+1}_s will indicate that the error of $Y^{0,(l+1)}_{s,n+2}$ (the term $\mathcal{O}(h_n^{l+2})$) is large. However, a large value of E^{l+1}_s could also appear if the amplifying function $R^{0,(l+2)}_s(h_n\lambda)$ of the algorithm $Y^{0,(l+2)}_{s,n+2}$ is large (note that the error estimators are detecting the effect of the insufficient stability). Such a situation can be detected by comparing the values of E^{l+1}_s, E^l_s and E^{l-1}_s.

Taking into account the above considerations, we propose to choose the starting algorithm according to the following strategy:

For certain prefixed constants $\eta < \theta < 1$,

- If $E^1_s > \theta E^0_s$ then $Y^{0,(0)}_{i,n+2}$ must be used.
- If $E^j_s < \theta E^{j-1}_s$ for $j = 1, \ldots, l$ and moreover $E^{l+1}_s \geq \theta E^l_s$ (for $l \leq p-2$) or well $l = p-1$, then
  - If $E^l_s < \eta E^{l-1}_s$, $Y^{0,(l+1)}_{i,n+2}$ will be used.
  - Otherwise, $Y^{0,(l)}_{i,n+2}$ will be used.

The constants θ and η can be empirically adjusted. From our experiments (based on the fifth order RadauIIA method) we have observed that an adequate range of values for the constant θ can be the interval $[0.5, 0.7]$, and η can be chosen in $[0.05, 0.15]$.

For our numerical experiments, we have selected the family of starting algorithms based on Lagrange interpolation on the last consecutive internal stages, given by

$$Y^{0,(l)}_{i,n+2} = \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_{2s-l}](\hat{c}_{2s+i}), \quad 1 \leq i \leq s, \quad l = 0, 1, \ldots, s. \tag{4.2}$$

The algorithms in this family have orders (stiff and non–stiff) from 0 to s, and from (3.5) the error estimator can be computed easily by

$$\begin{aligned} E^l_s &= \| \, Y^{0,(l)}_{s,n+2} - Y^{0,(l+1)}_{s,n+2} \, \| \\ &= \| \, \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_{2s-l}](\hat{c}_{3s}) - \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_{2s-l-1}](\hat{c}_{3s}) \, \| \\ &= \| \, [Y_{2s}, \ldots, Y_{2s-l-1}](\hat{c}_{3s} - \hat{c}_{2s}) \cdots (\hat{c}_{3s} - \hat{c}_{2s-l}) \, \|, \qquad 0 \leq l \leq s-1. \end{aligned}$$

We have also employed the starting algorithm of nonstiff order $s+1$ that uses

the last $s+2$ stages of the method (we are using the last two steps in this case) given in theorem 6 by

$$Y^0_{i,n+2} = \mathcal{L}_n[\hat{c}_{2s}, \ldots, \hat{c}_s](\hat{c}_{2s+i}) + \delta_i[Y_{2s}, Y_{2s-1}, \ldots, Y_{s-1}], \quad 1 \le i \le s,$$

and therefore

$$E^s_s = \| \ \delta_s[Y_{2s}, Y_{2s-1}, \ldots, Y_{s-1}] \ \| \ .$$

The use of an algorithm with order (nonstiff) $s+2$, based on the last $s+4$ stages had also been considered. However, it did not give any substantial improvement to the code in practice and since it force to keep $s+4$ stages and it can not be expressed in a simple, divided differences, way, we finally decided not using it in our numerical experiments.

In order to check the behaviour of the technique we have incorporated it to our integration code (the specifications for this code have been given in the introduction) in a way that we can compare the performance when we use each of the starting algorithms and also the variable–order technique.

In this case we are using the RadauIIA formula with 3 stages and the coefficients δ_i are given by

$$\delta_1 = \frac{4-\sqrt{6}}{10000} \, \frac{(r_n r_{n+1})^2 q(r_n)}{p(r_n)} \, q_1(r_n, r_n r_{n+1}),$$

$$\delta_2 = \frac{-4-\sqrt{6}}{10000} \, \frac{(r_n r_{n+1})^2 q(r_n)}{p(r_n)} \, q_2(r_n, r_n r_{n+1}),$$

$$\delta_3 = \frac{-1}{20} \, \frac{(r_n r_{n+1})^2 q(r_n)}{p(r_n)} \, q_3(r_n, r_n r_{n+1}),$$

with

$$q_1(r,u) = (-52+3\sqrt{6})u^2 + (-88+32\sqrt{6})ru + (-60+15\sqrt{6})r^2,$$
$$q_2(r,u) = (52+3\sqrt{6})u^2 + (88+32\sqrt{6})\,ru + (60+15\sqrt{6})\,r^2,$$
$$q_3(r,u) = 5\,u^2 + 8\,ru + 3\,r^2,$$
$$q(r) = \left((-4+\sqrt{6})r - 6 + \sqrt{6}\right)\left((4+\sqrt{6})r + 6 - \sqrt{6}\right)\left(10\,r + 6 - \sqrt{6}\right),$$
$$p(r) = 100\,r^3 + (270-45\sqrt{6})r^2 + (252-72\sqrt{6})r + 78 - 33\sqrt{6}.$$

Moreover, for this method

$$\rho(r_n) = \det M = K\, r_n^6\, p(r_n), \qquad K = \frac{9}{5\cdot 10^7}(\sqrt{6}-1),$$

and therefore, for $r_n > 0$, $\rho(r_n) = 0$ if and only if $p(r_n) = 0$ and this only happens for the value $r_n = 0.03483...$ Then, the order 4 algorithm will not be used in the code if $r_n < 0.1$.

The constants θ and η were adjusted experimentally, taking finally $\theta = 0.6$ and $\eta = 0.1$.

With this code we have integrated a number of differential problems with error tolerances ranging from 10^{-1} to 10^{-9} using the variable order technique (we will refer it as **V-code**) and also using using the Lagrange interpolation $\mathcal{L}_i^{(3)}$,that is, the algorithm $\mathcal{L}_n[\hat{c}_6, \ldots, \hat{c}_3](x)$ as standard codes do (we will refer it as **L-code**). Here we will comment the results obtained with the codes on some representative stiff problems.

While the L-code could integrate the Robertson problem only with error tolerances $\leq 10^{-4}$, the V-code was able to integrate it for every tolerance. Moreover, the efficiency of the V-code and the L-code were similar for low tolerances (the particular results for this problem can be seen in table 1.1). A similar situation appears with the integration of the scalar test equation $y' = -(y-1)^2$, $y(0) > 1$, on a large integration interval as $[0, 10^{11}]$. The L-code could not integrate it for tolerances 10^{-1} to 10^{-6}, while the V-code behaved efficiently at every tolerance.

It is interesting to comment the behaviour of the codes when integrating the problem E5 of the stiff DETEST package (see e.g. [8] or[15, p. 145]) where we have taken $[0, 10^{11}]$ as integration interval as suggested by Alexander. The L-code was not able to complete the integration of this problem at any of the error tolerances, while the V-code performed an efficient integration for all the tolerances. This is a practical evidence of the relevance that the starting algorithms can have in the integration with implicit methods.

Finally we present the results obtained integrating the van der Pol problem (see e.g. [15, p. 144]) along the integration interval $[0, 2]$, for which both codes behaved properly. In figure fig.4.1 we present an efficiency plot contrasting the logarithm of the global error against the number of function evaluations. From the figure, it is clear that the efficiency of the two codes is similar. The V-code was slightly more efficient than the L-code for low tolerances, where the starting algorithm of order 4 was often used in the V-code, and also for large tolerances, due probably to the good stability properties of the low order starting algorithms. In table 4.1 we present, for each tolerance, the number of times that the V-code employed each starting algorithm along the integration of this problem. Here NSS means the number of successful steps. As it can be seen, for large tolerances the code uses the low order algorithms and, as we reduce the tolerance, the code increases the use of the higher order algorithms. For the smallest tolerance, the code uses the 4th order algorithm in most steps.

Table 4.1
Selection of each starting algorithm with van der Pol problem

TOL	NSS	$\mathcal{L}_i^{(0)}$	$\mathcal{L}_i^{(1)}$	$\mathcal{L}_i^{(2)}$	$\mathcal{L}_i^{(3)}$	order 4
10^{-1}	222	59	55	22	74	78
10^{-2}	230	55	49	18	92	78
10^{-3}	268	55	32	19	145	88
10^{-4}	312	62	27	17	160	121
10^{-5}	392	38	10	21	237	174
10^{-6}	502	34	4	16	277	258
10^{-7}	638	32	2	20	192	467
10^{-8}	888	25	3	22	127	778
10^{-9}	1284	19	3	23	104	1189

5 Conclusions

A variable–order technique for the selection of the most convenient starting algorithm for beginning Newton–type iteration in implicit Runge–Kutta methods have been described. By means of some numerical experiments, we have shown that in the case of the 3–stages RadauIIA method the new strategy has proved to be a very adequate tool to make the integration code more robust and efficient. This technique, that can be applied to any implicit Runge–Kutta method and not only for stiff problems, is based on the construction of a family of starting algorithms with consecutive orders. We have also proposed a particular family, based on the information of the two last integration steps, that can be easily implemented by means of divided differences. The order of the algorithms has been studied giving some order results for general two steps starting algorithms. In this paper it has not been our objective the search of the "best" family of starting algorithms but some research in this line is being carried out by the authors.

References

[1] Bickart, T. A., *An efficient solution process for implicit Runge–Kutta methods*, SIAM J. Numer Anal., 14 (1977) 1022–1027.

[2] K. Burrage, J.C. Butcher & F.H. Chipmann, *STRIDE: Stable Runge-Kutta integrator for differential equations*, Report Series No. 150, Dept. Mathematics, University of Auckland (1979).

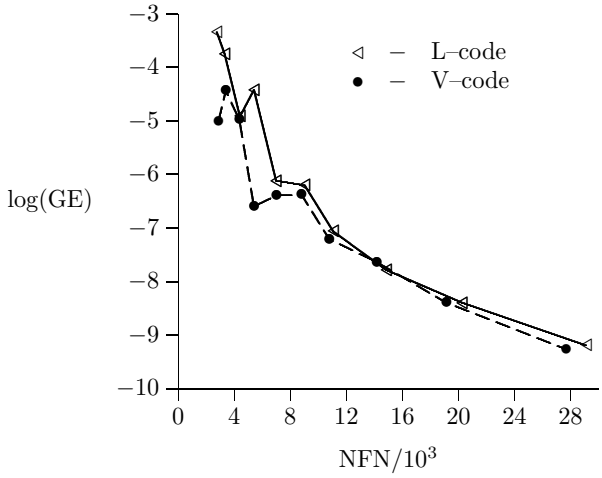


Fig. 4.1 Van der Pol problem

[3] K. Burrage, J.C. Butcher and F.H. Chipman, *An implementation of singly–implicit Runge–Kutta methods*, BIT, 20 (1980) 326–340.

[4] J.C. Butcher, *On the implementation of implicit Runge–Kutta methods*, BIT, 16 (1976) 237–240.

[5] G.J. Cooper and J.C. Butcher, *An iteration scheme for implicit Runge-Kutta methods,* IMA J. of Numer. Anal., 3 (1983) 127–140.

[6] G.J. Cooper and R. Vignesvaran, *A scheme for the implementation of implicit Runge-Kutta methods*, Computing, 45 (1990) 321–332.

[7] K. Dekker and J.G. Verwer, *Stability of Runge–Kutta methods for stiff nonlinear differential equations* (North Holland, Amsterdam, 1984).

[8] W.H. Enright, T.E. Hull and B. Lindberg, *Comparing numerical methods for stiff systems of ODEs*, BIT, 15 (1975) 10–48.

[9] W.H. Enright and J.D. Pryce, *Two FORTRAN packages for assessing initial value methods,* ACM Transactions on Mathematical Software, 13 (1) (1987) 1–27.

[10] S. González-Pinto, J.I. Montijano & L. Rández, *Iterative schemes for three–stage implicit Runge–Kutta methods*, Appl. Numer. Math., 17 (1995) 363–382.

[11] S. González Pinto, S. Pérez Rodríguez & J.I. Montijano, *On the numerical solution of stiff IVPs by Lobatto IIIA Runge–Kutta methods*, J. Comp. and Appl. Math., 82 (1997) 129–148.

[12] S. González Pinto, J.I. Montijano & S. Pérez-Rodríguez, *On the starting algorithms for fully implicit Runge-Kutta methods*, BIT, 40, 4 (2000) 685–714.

[13] S. González-Pinto, J.I. Montijano & S. Pérez-Rodríguez, *On the implementation of high order implicit Runge-Kutta methods*, Computers Math. Applic., 41, 7/8, (2001) 1009–1024.

[14] S. González Pinto, J.I. Montijano & S. Pérez-Rodríguez, *Stabilized starting algorithms for collocation Runge-Kutta methods*, Comp. Math. Applic., in press.

[15] Hairer, E., Wanner, G., *Solving Ordinary Differential Equations II* Springer–Verlag, (1996).

[16] M.P. Laburta, *Starting algorithms for IRK methods*, J. Comput. Appl. Math., 83 (1997) 269–288.

[17] W.M. Lioen, J.J.B. de Swart and W.A. van der Veen, *Test set for IVP solvers*, http://www.cwi.nl/cwi/projects/IVPtestset.shtml, Test set for IVP solvers, 1996.

[18] S.Pérez-Rodríguez, *Integración de problemas stiff a través de métodos Runger-Kutta*, Ph. D. Thesis (2000), Universidad de La Laguna.

[19] A. Prothero, A. Robinson, *On the stability and accuracy of one–step methods for solving stiff systems of ordinary differential equations*, Math. of Comp., 28 (1974) 145–162.