On the implementation of high order implicit Runge–Kutta methods.

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

[a] *Dpto. de Análisis Matemático, Universidad de La Laguna, 38271 La Laguna-Tenerife, Spain.*

[b] *Dpto. de Matemática Aplicada, Universidad de Zaragoza, 50009 Zaragoza, Spain.*

Abstract

In this paper we develop single–Newton iterative schemes for the solution of the stage equations of some implicit Runge–Kutta methods such as Gauss, Radau IIA and Lobatto IIIA with four implicit stages. We also compare the implementation cost of these schemes with Simplified Newton iteration and we present some numerical experiments on some well known stiff test problems to show that the proposed iterations are reliable and efficient.

1 Introduction

We consider the numerical solution of stiff initial value problems for a system of ordinary differential equations

$$y'(t) = f(t, y(t)), \quad t \in [t_0, t_f], \qquad y(t_0) = y_0 \in I\!R^m, \tag{1.1}$$

where the derivative function $f(t, y)$ is supposed to be sufficiently smooth so that (1.1) has a unique solution $y(t)$.

Implicit Runge–Kutta methods of s stages require, to advance from (t_n, y_n) a step of size h, the solution of an implicit set of ms equations of the form

$$Y = e \otimes y_n + h(A \otimes I)F(Y), \tag{1.2}$$

[1] Partially supported by project 201 55/98 (University of La Laguna).

together with the advancing formula

$$y_{n+1} = y_n + h(b^T \otimes I)F(Y), \tag{1.3}$$

where $F(Y)$ is the ms vector

$$F(Y) = \left(f(t_n + c_1 h, Y_1)^T, \ldots f(t_n + c_s h, Y_s)^T\right)^T,$$

$e = (1, \ldots, 1)^T \in I\!R^s$, $\otimes$ stands for the standard Kronecker product and $A \in I\!R^{s\times s}$ and $b = (b_1, \ldots, b_s)^T$ are coefficients of the Runge–Kutta method (as usual, we consider $c = Ae$). For notation purposes we denote throughout the paper by I the identity matrix, and its dimension should be read from the context, so when it appears on the left side of a Kronecker product its dimension will be s (the number of stages of the Runge-Kutta method) and when it appears on the right side its dimension will be m (the dimension of the initial value problem).

Some implicit methods such as Radau IIA, Gauss or Lobatto IIIA possess excellent stability and convergence properties for stiff problems (see [7], [3], [11]). However, the computational cost involved in the solution of (1.2) has limited seriously their practical use. Between the codes based on fully implicit Runge–Kutta methods we can mention RADAU5 (see Hairer & Wanner [11], ch. IV.8, pp. 118–127) that uses the 3 stages Radau IIA formula of order 5.

When an implicit Runge–Kutta method is applied to stiff differential systems, the stage equations (1.2) are usually solved by Newton type iterations. A possible approach is to use the Simplified Newton scheme given by,

$$(I - h(A \otimes J))(Y^{(\nu+1)} - Y^{(\nu)}) = D(Y^\nu), \quad \nu = 0, 1, \ldots, \tag{1.4}$$

where the residual mapping $D(Y)$ is defined by,

$$D(Y) := -Y + e \otimes y_n + h(A \otimes I)F(Y)$$

and J is an approximation to the Jacobian matrix $\partial f/\partial y(t, y)$ at some intermediate point, usually the previous point (t_n, y_n). The main drawback of (1.4) is that it needs, at each integration step (or every few integration steps) the LU factorization of the $ms \times ms$ matrix $(I \otimes I - h(A \otimes J))$, whose cost as a function of the number of stages s varies as s^3. In order to reduce the cost in the solution of (1.4), Hairer and Wanner, in their code RADAU5, employ a similarity transformation proposed by Butcher in [2] and Bickart in [1] such that the LU factorization of the $ms \times ms$ matrix ($s = 3$ in this particular case) is replaced by the factorization of two matrices of dimension m, one

of them with complex values, and then the LU cost reduces by a factor of 5 approximately.

Another kind of iterative scheme for solving (1.2) which has received considerable attention lately (see [5], [6],[9],[10]) is the so called Single–Newton iteration, given by

$$\begin{aligned} Y^{(\nu+1)} &= Y^{(\nu)} + (S \otimes I)E^{(\nu)}, \quad \nu = 0, 1, \ldots, \\ (I \otimes I - \tau I \otimes hJ)E^{(\nu)} &= (B \otimes I)D(Y^{(\nu)}) + (L \otimes I)E^{(\nu)} \end{aligned} \tag{1.5}$$

where $\tau > 0$ and L, B and S are suitable constant matrices. Since L is chosen to be strictly lower triangular, only one LU factorization of the $m \times m$ matrix $(I - h\tau J)$ is needed.

Efficient Single–Newton schemes for Gauss, Radau IIA and Lobatto IIIA methods with 2 and 3 implicit stages have been developed in [5], [9], [10] and also a Single–Newton scheme for the four-stage Gauss RK method has been obtained in [5]. These schemes, compared to the Simplified Newton, present the advantage that they need only one LU factorization of size m. However, as has been shown in [4], the Simplified Newton iterative scheme has better convergence properties than Single–Newton schemes when the step-size tends to zero.

The aim of this paper is to gain insight of the behaviour of Single–Newton schemes as a practical alternative to Simplified Newton iteration when applied to fully implicit Runge-Kutta methods such as Gauss, Radau IA-IIA, Lobatto IIIA-B-C. Note that the iteration (1.5) is algebraically equivalent to

$$(I \otimes I - h(T \otimes J))(Y^{(\nu+1)} - Y^{(\nu)}) = (P \otimes I)D(Y^{\nu}), \quad \nu = 0, 1, \ldots, \tag{1.6}$$

where $T = \tau S(I - L)^{-1}S^{-1}$ and $P = \tau^{-1}TSB$. Hence, the particular choice $B = (I - L)S^{-1}$, which has several advantages as indicated in [9], gives the Single-Newton scheme,

$$(I \otimes I - h(T \otimes J))(Y^{(\nu+1)} - Y^{(\nu)}) = D(Y^{\nu}), \quad \nu = 0, 1, \ldots. \tag{1.7}$$

These equations can be seen as an approximation to (1.4) where the matrix A has been replaced by a matrix T, with a unique eigenvalue $\tau > 0$. It is clear that when the number of stages increases, the "distance" between A and T also increases, since the matrix A for the implicit Runge-Kutta methods under consideration possesses a multi-point spectrum with pairs of conjugate complex eigenvalues (all distinct between them), and at most one real eigenvalue.

Consequently, the speed of convergence of Single–Newton schemes can reduce when the number of stages is increased.

Our aim is to develop efficient Single–Newton schemes for Runge–Kutta methods with four practical implicit stages and compare them, from a practical point of view, with Simplified Newton. In particular we are interested in this kind of scheme for the eighth order Gauss and the seventh order Radau IIA formula, both having 4 stages, and also for the eight order Lobatto IIIA method. In this case the method has five stages but since the first of them is explicit, the iterative scheme can be developed with the same techniques as for the other two methods. Recall that high order methods are usually more efficient than lower order methods when they are required to integrate problems with small error tolerances. Thus it is essential to minimize the computational cost of iterative schemes for solving the nonlinear stage equations.

There is not much practical experience with high order implicit Runge–Kutta methods that let us know when the high order of convergence will compensate the high computational cost they require compared to methods of lower order. We can mention that Hairer and Wanner [12] recently have presented a variable order code based on Radau IIA methods of orders 5, 9 and 13. Here, we also intend to compare the behaviour of Radau IIA methods of orders 5 and 7, using Single–Newton schemes to solve the stage equations (1.2).

Finally, the paper is organized as follows: In Section 2 we study the implementation cost of Simplified Newton and Single–Newton schemes, showing that Single–Newton is in general the cheapest. In Section 3, following the ideas in [9] we obtain efficient Single–Newton schemes for the 4–stage Gauss of order 8, the 4–stage Radau IIA of order 7 and the 5–stage Lobatto IIIA of order 8. In Section 4 some numerical experiments comparing the efficiency of the new iterative schemes and the simplified Newton iteration are presented.

2 Implementation cost of Simplified–Newton and Single–Newton schemes

Consider a four–stage implicit Runge–Kutta method with a nonsingular matrix A and suppose that (1.2) is solved by means of a Simplified Newton scheme (1.4). In order to reduce the computational cost, we will consider a technique similar to the one used by Hairer & Wanner in their code RADAU5 [11]. Hence, if Q is a regular real matrix that transforms A^{-1} to a simple block matrix $\Lambda = Q^{-1}A^{-1}Q$, then pre-multiplying (1.4) by $A^{-1} \otimes I$ and denoting

$W^{(\nu)} = (Q^{-1} \otimes I)(Y^{(\nu)} - e \otimes y_n)$ we have

$$\begin{aligned} [\Lambda \otimes I - h(I \otimes J)]\Delta W^{(\nu)} &= R^{(\nu)}, \quad \nu = 0, 1, \ldots \\ R^{(\nu)} &= -(\Lambda \otimes I)W^{(\nu)} + (Q^{-1} \otimes I)F\Big((Q \otimes I)W^{(\nu)} + e \otimes y_n\Big) \\ W^{(\nu+1)} &= W^{(\nu)} + \Delta W^{(\nu)}. \end{aligned} \tag{2.8}$$

For Gauss and Radau IIA methods, matrix A has two pairs of complex conjugate eigenvalues and Λ has the form $\Lambda = \mathrm{diag}(\Lambda_1, \Lambda_2)$ with

$$\Lambda_j = \begin{pmatrix} \alpha_j & -\beta_j \\ \beta_j & \alpha_j \end{pmatrix} \qquad j = 1, 2.$$

For the Lobatto IIIA method with five stages, the matrix A has one eigenvalue zero and two pairs of complex conjugate eigenvalues that are the same as the corresponding ones of the four stages Gauss method (let us note that both methods have the m/m Padé aproximant as linear stability function whose denominator is $\det(I - zA)$). Even though the matrix A is singular, the process for this method follows a similar approach in which instead of the matrix A, its submatrix $\bar{A}$ obtained by removing the first row and column is used (see comments in section 3.3).

Now, by putting $R^{(\nu)} = (R_1^{(\nu)}, R_2^{(\nu)}, R_3^{(\nu)}, R_4^{(\nu)})^T$ and $\Delta W^{(\nu)} = (\Delta W_1^{(\nu)}, \Delta W_2^{(\nu)}, \Delta W_3^{(\nu)}, \Delta W_4^{(\nu)})^T$, equation (2.8) can split into two complex linear systems of dimension m

$$\begin{aligned} [(\alpha_1 + i\beta_1)I - hJ](\Delta W_1^{(\nu)} + i\Delta W_2^{(\nu)}) &= R_1^{(\nu)} + iR_2^{(\nu)} \\ [(\alpha_2 + i\beta_2)I - hJ]\,(\Delta W_3^{(\nu)} + i\Delta W_4^{(\nu)}) &= R_3^{(\nu)} + iR_4^{(\nu)}. \end{aligned} \tag{2.9}$$

Let us focus first on the computational effort to factorize the two complex matrices $(\alpha_j + i\beta_j)I - hJ$, $j = 1, 2$. We will only consider floating point sums, products and divisions and we will assume that all of them have equivalent computational cost. Moreover, we will not consider the effect that complex arithmetic has on the error propagation (for a recent study of errors and floating point arithmetic see for example [13]), and we will not consider the cost of evaluating the Jacobian matrix J and the derivative function f.

Taking into account that a complex multiplication consists of 4 real multiplications and 2 real sums, a complex sum consists of 2 real sums and a complex division consists of 8 real products or divisions and 3 real sums, the total work for the two LU factorizations will be about $16m^3/3$ floating point operations $((16m + 25)m(m - 1)/3$ exactly).

Table 1
Flops per iter. step in Simplified Newton and Single-Newton schemes.

Iterative scheme	$s = 3$	$s = 4$	$s = 5$	$s = 6$
Simplified Newton	$10m^2 + 48m$	$16m^2 + 86m$	$18m^2 + 123m$	$24m^2 + 177m$
Single Newton	$6m^2 + 35m$	$8m^2 + 63m$	$10m^2 + 99m$	$12m^2 + 143m$

Let us study now the number of operations involved in each iteration step. First, to solve the two complex systems (2.9) of dimension m with the matrix previously factorized, since each system require to be solved m divisions, $m(m-1)$ products and $m(m-1)$ sums, the total number of floating point operations (flops) will be $16m^2 + 6m$.

On the other hand, at each iteration step, the residual $R^{(\nu)}$ in (2.8) must be computed which, since the matrix Q is a 4×4 full matrix, requires $76m$ real operations. Then, in total, each iteration step needs $16m^2+86m$ floating point operations (let us observe that $4m$ sums are needed to update W^{ν}).

The final computational cost will be affected by each part of the calculations in a different way depending on the dimension m of the differential system and the number of iterations needed to get convergence in the iterative scheme. Thus, if m is large, the cost of the LU factorizations will be the dominant term due to the m^3 factor. Assuming for example an average number of iterations per integration step of 4 and a value of $m = 100$, the average number of operations involved in the iterations (per integration step) will be $4(16\times100^2+ 86\times100) = 674400$, approximately $1/8$ of the cost of the two LU factorizations. On the other hand, if we compare the influence of the evaluation of the residual and the solution of the triangular systems at each iteration step, for $m = 100$, the cost of the evaluation of the residual is 7600, which is negligible compared with the cost of the LU factorizations, and about $1/21$ of the cost of the system solution 160600. For a value of $m = 10$, the cost of the LU factorizations will be about 5550 flops, the cost of the systems solutions is 1660 and the cost of the evaluation of the residual 760. Assuming 4 iterations per integration step, both factors (the LU factorizations and the iteration cost) have a similar influence in the total cost of the method. Finally, for very small values of m, it is clear that the computational cost will be affected mainly by the calculations in the iterations.

In Table 1 we present the number of floating point operations required at each iteration step by Simplified Newton schemes applied to the 3, 4, 5 and 6 implicit stage Runge–Kutta methods.

Let us now consider a Single–Newton scheme (1.5) with $B = (I - L)S^{-1}$. It is clear that each LU matrix factorization will require a number of operations of the order $2m^3/3$ ($2m^2(m-1)/3$ exactly), independently of the number of stages s. For $s = 4$, this cost is about $1/8$ of the required by Simplified Newton

and this makes Single–Newton more efficient when the dimension m is large.

Putting $Z^{(\nu)} = Y^{(\nu)} - e \otimes y_n$ and $W^{(\nu)} = (S^{-1} \otimes I)Z^{(\nu)}$, the Single–Newton scheme can be rewritten in the form

$$\begin{aligned} G^{(\nu)} &= -W^{(\nu)} + h(S^{-1}A \otimes I)\ F\left(e \otimes y_n + (S \otimes I)W^{(\nu)}\right) \\ (I \otimes I - \tau I \otimes hJ)E^{(\nu)} &= G^{(\nu)} + (L \otimes I)(E^{(\nu)} - G^{(\nu)}) \\ W^{(\nu+1)} &= W^{(\nu)} + E^{(\nu)}, \quad \nu = 0, 1, \ldots \end{aligned} \tag{2.10}$$

which is more interesting in practice because we avoid a product by the matrix $S^{-1} \otimes I$, that is, $s(s-1)m$ flops per iteration can be saved (as we will see later, for the RK methods of interest the matrix S can be chosen upper triangular with "1" on the main diagonal).

¿From the equations (2.10) it follows that the number of floating point operations required at each iteration step by Single–Newton schemes applied to an implicit Runge–Kutta method of s stages is given by $2sm^2 + (4s^2 - 1)m$. The resulting values for $s = 3, 4, 5, 6$ are displayed in Table 1. We see that for the same number of stages, the cost per iteration step for Single–Newton is smaller than for Simplified Newton. Moreover, for large values of m the ratio of the cost approximates to 2 for $s = 4, 6$ and to $5/3, 9/5$ and $13/7$ for $s = 3, 5$ and 7 respectively. Nevertheless, a realistic comparison of the cost of the schemes depends also on the number of iterations that each scheme requires to get convergence.

The above considerations show that for problems with large dimensions, Single–Newton is expected to be more efficient than Simplified Newton if problems of convergence are not severe. For middle size problems, Single–Newton schemes may be more efficient if the number of iterations used is not much too large with respect to Simplified Newton. Finally, for small values of m, the efficiency of the schemes will depend on the number of iterations needed for convergence.

Remark 1 *Our previous study is based on the number of floating point operations independently of the computer. However in practice an important component in the CPU time taken by the codes comes from the way in which the compiler optimizes the code generated in the compilation process, getting benefit of the particular characteristics of the computer. Thus a greater number of floating point operations does not always imply a longer CPU time. An interesting example of such an optimization is found in computers based on 64 bits DEC Alpha processors where complex arithmetic operations can be done in a particularly optimized way and the codes using it improve considerably their efficiency.*

3 Efficient Single–Newton schemes for RK methods with 4 implicit stages

In this section we develop Single–Newton schemes (1.5) with $B = (I - L)S^{-1}$, for the four stages Gauss and Radau IIA methods and also for the five stages Lobatto IIIA. Thus, we will look for matrices S, L (lower triangular) and the parameter τ such that the resulting scheme is as efficient as possible.

When a Single–Newton is applied to the scalar linear test equation $y' = \lambda y$, the iteration errors satisfy

$$Y^{(\nu+1)} - Y = M(z)(Y^{(\nu)} - Y) = M(z)^{\nu}(Y^{(0)} - Y) \tag{3.11}$$

where $z = h\lambda$ and the amplifying matrix $M(z)$ is given by

$$M(z) = z(I - zT)^{-1}(A - T), \qquad T = \tau S(I - L)^{-1}S^{-1}. \tag{3.12}$$

In accordance with the ideas given in [9], we want to find a matrix T such that:

(P1) T has a only eigenvalue $\tau > 0$,

(P2) $\rho(M(\infty)) = 0$,

(P3) $b^T A^{-1} M(\infty) = 0$,

(P4) $\max\{\rho(M(z)) | z \in I\!R^-\}$ is minimum,

(P5) the "distance" between A and T is as small as possible.

Condition (P1) is required if matrix T must satisfy the relation expressed in (3.12). Condition (P3) guarantees that the amplifying functions of the iterations approximate to the amplifying function of the Runge–Kutta method and they are equal when $z \to \infty$ (see th. 2.1 in [9]). Conditions (P2) and (P4) are self-explanatory. The details of condition (P5) are made more precise later in this section.

In order to transform conditions (P1) and (P2) to a more useful form to obtain the matrix T, we will give first the following Lemmas (we will assume in the following that A and T are 4×4 matrices, A being nonsingular):

Lemma 2 *The matrix T fulfills (P1) if and only if*

$$\det(T) = \tau^4, \qquad tr(T) = 4\tau,$$
$$tr(T^2) = 4\tau^2, \qquad tr(T^{-1}) = 4/\tau \quad or \quad tr(T^3) = 4\tau^3.$$

The proof of this lemma follows at once from (3.12).

Lemma 3 *If A and T satisfy (P1) and (P2), then*

$$\tau = (\det A)^{(1/4)} \quad \textit{and} \quad \det(A - T) = 0.$$

PROOF. Since $M(\infty) = T^{-1}(T - A) = (I - T^{-1}A)$, (P2) implies that $\det(T - A) = 0$. Moreover, by using (P2) and the Schur's Decomposition of $M(\infty)$, it follows that there exists a orthogonal matrix U such that

$$I - T^{-1}A = U^{-1}\Delta U,$$

being Δ a strictly upper triangular matrix. From here, taking determinants and using (P1), $\det(A) = \tau^4$.

Lemma 4 *If $\det(A - T) = 0$, then the eigenvalues of the matrix $M(z)$ are the roots of the equation*

$$\mu^4 - b_3\mu^3 + b_2\mu^2 - b_1\mu = 0 \tag{3.13}$$

with

$$\begin{aligned} b_1 &= q(p-2) - 2 + b_3 & q &= \det(I - zA)/(1 - \tau z)^4 \\ b_2 &= q(p-1) - 3 + 2b_3 & p &= trQ(z) \\ b_3 &= 4 - tr(Q(z)^{-1}) & Q(z) &= (I - zA)^{-1}(I - zT). \end{aligned}$$

PROOF. First, since $\det(A - T) = 0$, then $\det(M(z)) = 0$ and at least one of the eigenvalues of $M(z)$ vanishes.
Developing the function $\phi(\mu) = \det(\mu I - M(z))/\mu = (\mu - \lambda_1)(\mu - \lambda_2)(\mu - \lambda_3)$, (here λ_j, $j = 1, 3$, denote the non-null eigenvalues of $M(z)$) we get $\phi(\mu) = \mu^3 - b_3\mu^2 + b_2\mu - b_1$ with,

$$\begin{aligned} b_1 &= \phi'(1) - \phi(1) - 2 + b_3, \\ b_2 &= \phi'(1) - 3 + 2b_3, \\ b_3 &= \mathrm{tr}(M(z)) = \lambda_1 + \lambda_2 + \lambda_3. \end{aligned}$$

But since $\det(\mu I - M(z)) = \det\Big(\mu I - z(A + (\mu - 1)T)\Big)/(1 - z\tau)^4$, after some calculations we obtain

$$\begin{aligned} \phi(1) &= q \\ \phi'(1) &= \lim_{\epsilon \to 0} \frac{\phi(1+\epsilon) - \phi(1)}{\epsilon} = q(p-1). \end{aligned}$$

On the other hand, since $M(z) = I - Q(z)^{-1}$, it is clear that $b_3 = 4 - \mathrm{tr}(Q(z)^{-1})$.

Corollary 5 *If T satisfies (P1), then condition (P2) is equivalent to*

$$\tau = (\det(A))^{1/4}, \quad \det(A - T) = 0, \quad tr(T^{-1}A) = tr(A^{-1}T) = 4.$$

PROOF. By making $z \to \infty$, the eigenvalue equation for $M(\infty)$ becomes,

$$\mu^4 - b_3^*\mu^3 + b_2^*\mu^2 + b_1^*\mu - b_0^* = 0,$$

where $b_j^* = b_j(\infty)$, $(j = 1, 2, 3)$ (see (3.13)), and $b_0^* = \det(I - T^{-1}A) = \tau^{-4}\det(T - A)$. Now from the previous lemma,

$$b_0^* = 0 \Leftrightarrow \det(T - A) = 0, \;\; b_3^* = 0 \Leftrightarrow \mathrm{tr}(T^{-1}A) = 4, \quad \text{and}$$

$b_2^* = b_1^* = 0 \Leftrightarrow q(\infty) = 1$ and $p(\infty) = 4 \Leftrightarrow \tau = (\det(A))^{1/4}$ and $\mathrm{tr}(A^{-1}T) = 4$.

The next result allows us to express the condition (P4) in a more adequate form.

Theorem 6 *Let A be a given regular matrix and assume that T is a matrix such that (P1) and (P2) are fulfilled. Then the minimization condition (P4) holds if*

$$tr(AT) = \Gamma_1^*, \quad tr(A^{-2}T) = \Gamma_2^*, \quad tr(T^{-2}A) = \Gamma_3^*$$

where Γ_1^*, Γ_2^* and Γ_3^* are certain constants that only depend on the matrix A.

PROOF. By expanding $Q(z) = (I - zA)^{-1}(I - zT)$ in powers of z and z^{-1} respectively we get,

$$Q(z) = I + z(A - T) + z^2A(A - T) + \mathcal{O}(z^3), \quad (z \to 0),$$

$$Q(z) = A^{-1}T + z^{-1}A^{-2}(T - A) + \mathcal{O}(z^{-2}), \quad (z \to \infty).$$

Now, by using lemma 4 and corollary 5, it follows after some calculations, that the coefficients of the eigenvalue equation (3.13) are given by,

$$b_1 = \frac{\tau^4\xi_0 z^3}{(1 - \tau z)^4}, \quad b_2 = \frac{\xi_2 z^2 + \xi_3 z^3}{(1 - \tau z)^4}, \quad b_3 = \frac{\eta_1 z + \eta_2 z^2 + \eta_3 z^3}{(1 - \tau z)^4},$$

where

$$\xi_0 = 12\tau^{-1} - 3\mathrm{tr}(A^{-1}) + \Gamma_2 - \Gamma_3,$$

$$\xi_2 = (\mathrm{tr}^2(A) - \mathrm{tr}(A^2))/2 + 6\tau^2 - 4\tau\,\mathrm{tr}(A) + \Gamma_1, \quad \xi_3 = -4\tau^4\mathrm{tr}(A^{-1}) + 20\tau^3 + \tau^4\Gamma_2 - 2\tau^4\Gamma_3,$$

Table 2
Constants for the minimum spectral radius of $M(z)$ on the real semiaxis.

	Gauss	Radau IIA	Lobatto IIIA
Γ_1^*	0.06316638299274640	0.08296548473447771	0.06316638299274640
Γ_2^*	14.86450048645422	11.01111697854543	14.86450048645422
Γ_3^*	31.73222088578815	27.66997861432155	31.73222088578815
$\rho_{\max}^0$	0.0893204199714	0.104708968155	0.0893204199714
$\rho_{\max}^1$	0.320182072684	0.378417643002	0.320182072684
$\rho_{\max}^2$	0.147383853954	0.172953394381	0.147383853954

$$\eta_1 = \mathrm{tr}(A) - 4\tau, \quad \eta_2 = 12\tau^2 - 4\tau\mathrm{tr}(A) + \Gamma_1, \quad \eta_3 = 4\tau^3 - \tau^4\Gamma_3,$$

being

$$\Gamma_1 = \mathrm{tr}(AT), \quad \Gamma_2 = \mathrm{tr}(A^{-2}T), \quad \Gamma_3 = \mathrm{tr}(T^{-2}A).$$

Now, if we consider the function of three variables

$$g(\Gamma_1, \Gamma_2, \Gamma_3) := \max\{\rho(M(z)),\ z \in I\!R^-\},$$

since g is a nonnegative continuous function on $I\!R^3$ that satisfies $g(\Gamma_1, \Gamma_2, \Gamma_3) \to \infty$ when some Γ_j, $j = 1, 2, 3$ goes to infinity, it is clear that g possesses an absolute minimum that is reached at some values Γ_1^*, Γ_2^* and Γ_3^* that only depend on the matrix A.

For the Runge-Kutta methods of interest, the uniqueness of the constants Γ_j^* as well as the calculation of such constants has been carried out by using the Schur-Cohn criterion [15] (after an adequate change of variable) and the process is quite long and complicated. For the sake of brevity we will omit here such calculations and we refer to [16] for a detailed study. In Table 2 we present the values obtained for the constants Γ_j^* when considering the eighth order Gauss and Lobatto IIIA methods and the seventh order Radau IIA formula. In the table are also included the values

$$\begin{aligned}
\rho_{\max}^0 &= \max\{\rho(M(z)) \mid z \in I\!R^-\},\\
\rho_{\max}^1 &= \max\{\rho(M(iy)) \mid y \in I\!R\},\\
\rho_{\max}^2 &= \max\{\rho(M((1-i)y) \mid y \in I\!R^-\},
\end{aligned}$$

for each of the three methods.

After the above results, and taking into account that (P3) implies that $\det(A - T) = 0$, conditions (P1) to (P5) are equivalent to

(Q1) $\det(T) = \tau^4 = \det(A)$,

(Q2) $\mathrm{tr}(T) = 4\tau$,

(Q3) $\mathrm{tr}(T^2) = 4\tau^2$,

(Q4) $\mathrm{tr}(T^{-1}) = 4/\tau$ or $\mathrm{tr}(T^3) = 4\tau^3$,

(Q5) $\mathrm{tr}(A^{-1}T) = 4$,

(Q6) $\mathrm{tr}(T^{-1}A) = 4$,

(Q7) $\mathrm{tr}(AT) = \Gamma_1^*$,

(Q8) $\mathrm{tr}(A^{-2}T) = \Gamma_2^*$,

(Q9) $\mathrm{tr}(T^{-2}A) = \Gamma_3^*$,

(Q10) $b^T A^{-1}(I - T^{-1}A) = 0$,

(Q11) the "distance" between A and T is as small as possible.

We have to determine the 16 coefficients of the matrix T satisfying the 13 equations in conditions (Q1) to (Q10). It can be seen that there are three free parameters that must be used to fulfil condition (Q11). To this end, we have considered as "distance" the quantity

$$D_{AT} = \nu_1 \|I - A^{-1}T\|_F^2 + \nu_2 \|I - TA^{-1}\|_F^2 + \nu_3 \|e_4^T(A - T)\|_2^2,\ e_4^T = (0, 0, 0, 1),$$

where $\|\cdot\|_F$ is the Fröbenius norm and ν_i are weights that must be chosen so that the resultant scheme gives the best results for the stiff differential problems of the well known package DETEST [8], having at the same time good stability properties.

Due to the number of equations and parameters involved in the process, to accomplish this task we have proceeded numerically as follows:

- First, we obtain an initial matrix T_0 satisfying conditions (Q1) to (Q10).
- Then, we fix several sets of values ν_1, ν_2 and ν_3.
- For each set of values ν_i, starting from T_0, we obtain a matrix T that minimizes D_{AT} and fulfills the conditions (Q1) to (Q10).
- Among all matrices T obtained, we select one such that the corresponding iteration (1.7) has good stability properties and gives the best results on the stiff problems of DETEST package [8].

3.1 Eighth–order Gauss.

For this method, we have obtained the best Single–Newton scheme for $\nu_1 = 1$, $\nu_2 = \nu_3 = 0$. The values computed for this scheme are (with 16 significant

figures):

$$D_{AT} = 0.2186117957792797, \quad \|A - T\|_F = 0.1866324635411617.$$

with matrix $T = (t_{ij})$:

$$\begin{aligned}
t_{11} &= 0.07056898453975971 & t_{31} &= 0.1097496189565937 \\
t_{12} &= -0.01381201242940272 & t_{32} &= 0.3953973119562834 \\
t_{13} &= 0.01374509656255927 & t_{33} &= 0.2550102453783648 \\
t_{14} &= 0.001273397980705694 & t_{34} &= -0.03800926472551498 \\
t_{21} &= 0.1359096681314922 & t_{41} &= 0.1026795079784531 \\
t_{22} &= 0.2039916522067102 & t_{42} &= 0.3643735550837732 \\
t_{23} &= 0.01953742322502287 & t_{43} &= 0.4333395062278329 \\
t_{24} &= -0.007041562052392658 & t_{44} &= 0.09521710525921647 \ .
\end{aligned}$$

For this matrix, it is possible to obtain an upper triangular matrix

$$S = \begin{pmatrix}
1 & -0.6677448107835342 & 0.1296306965460327 & 0.01526277075698497 \\
0 & 1 & -0.2153491783691625 & 0.07296098377515141 \\
0 & 0 & 1 & 0.07575507029183779 \\
0 & 0 & 0 & 1
\end{pmatrix}$$

that transforms T to triangular form. With these matrices T and S, the matrix $L = I - \tau S^{-1} T^{-1} S$ is

$$L = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0.9627423789846739 & 0 & 0 & 0 \\
-1.194428300588649 & 1.918753137082504 & 0 & 0 \\
1.649572580382698 & -2.628995768624925 & 2.357166809194904 & 0
\end{pmatrix}.$$

The matrices L and S together with

$$\tau = (\det(A))^{1/4} = 0.1561969968460128$$

give the parameters of the Single–Newton scheme.

3.2 Seventh–order Radau IIA

In this case, the best Single–Newton scheme was obtained for $\nu_1 = \nu_2 = 1$ and $\nu_3 = 100$, giving the values

$$D_{AT} = 1.334357031843827, \quad \|A - T\|_F = 0.1198832704310612.$$

The corresponding matrices $T = (t_{ij})$, S and L are given by:

$$\begin{aligned}
t_{11} &= 0.1187824099582517 & t_{31} &= 0.2267733612906856 \\
t_{12} &= 0.01022763543870539 & t_{32} &= 0.4394654798955388 \\
t_{13} &= 0.02251934010521350 & t_{33} &= 0.2423196391476349 \\
t_{14} &= -0.002140831122870532 & t_{34} &= -0.01672793262894805 \\
t_{21} &= 0.2463531839329877 & t_{41} &= 0.2303363939912873 \\
t_{22} &= 0.2880948365341910 & t_{42} &= 0.4140965520644702 \\
t_{23} &= -0.02947965404901304 & t_{43} &= 0.3882107808506906 \\
t_{24} &= -0.002392091968997757 & t_{44} &= 0.09380543432526635\ ,
\end{aligned}$$

$$S = \begin{pmatrix}
1 & -0.3746257695117888 & 0.07689675270074446 & 0.04190406032755296 \\
0 & 1 & 0.05051271922734543 & -0.01257194014862304 \\
0 & 0 & 1 & 0.2253907333361419 \\
0 & 0 & 0 & 1
\end{pmatrix},$$

$$L = \begin{pmatrix}
0 & 0 & 0 & 0 \\
1.294297023384814 & 0 & 0 & 0 \\
-1.014023314466600 & 1.510766557167087 & 0 & 0 \\
1.286041959197947 & -1.706853680903114 & 2.297920385846297 & 0
\end{pmatrix}.$$

The matrices L and S together with

$$\tau = (\det(A))^{1/4} = 0.1857505799913360$$

determine the Single–Newton scheme.

3.3 Eight–order Lobatto IIIA

In the case of the Lobatto IIIA method, the first row of the coefficient matrix A vanishes. Then, denoting

$$A = \begin{pmatrix} 0 & 0^T \\ w & \bar{A} \end{pmatrix}$$

equation (1.2) can be rewritten

$$\bar{Y} = e \otimes y_n + h(w \otimes f(t_n, y_n)) + h(\bar{A} \otimes I)F(\bar{Y}),$$

where $\bar{Y} = (Y_2, Y_3, Y_4, Y_5)^T$. Thus, the system (1.2) reduces to a system of $4m$ equations that can be studied as in the previous cases, but taking the submatrix $\bar{A}$ instead of the matrix A.

This time, the best Single–Newton scheme was obtained for $\nu_1 = 1$, and $\nu_2 = \nu_3 = 0$, for which

$$D_{\bar{A}T} = 0.07066885046321667, \quad \|\bar{A} - T\|_F = 0.09799263272854765.$$

The matrix $T = (t_{ij})$ for this method is

$$\begin{aligned}
t_{11} &= 0.1205065476893790 & t_{31} &= 0.2675367041374556 \\
t_{12} &= -0.001249676535040056 & t_{32} &= 0.4217726039803753 \\
t_{13} &= 0.003900830554640007 & t_{33} &= 0.1739257710023307 \\
t_{14} &= -0.0006329622087931463 & t_{34} &= 0.009132813977995455 \\
t_{21} &= 0.3079578502684815 & t_{41} &= 0.2775596403310148 \\
t_{22} &= 0.2327971369316140 & t_{42} &= 0.3986701534245386 \\
t_{23} &= -0.02614746695545937 & t_{43} &= 0.2894006793595838 \\
t_{24} &= 0.006158162143340951 & t_{44} &= 0.09755853176072735
\end{aligned}$$

and the upper triangular matrix S that transforms T to triangular form and the matrix L are given by,

$$S = \begin{pmatrix} 1 & -0.1345492788488319 & -0.7907579166890781E-3 & 0.01048164212642994 \\ 0 & 1 & 0.1654189391431284 & -0.03863351412430941 \\ 0 & 0 & 1 & 0.2457879968605093 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

and

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1.829166626367437 & 0 & 0 & 0 \\ -2.201612484488081 & 1.901230267943492 & 0 & 0 \\ 2.551217615151542 & -2.009365789995880 & 2.273595510125324 & 0 \end{pmatrix}.$$

The Single–Newton scheme is determined by the matrices L and S and by $\tau = (\det(\bar{A}))^{1/4} = 0.1561969968460128$.

4 Numerical experiments

In this section we present the results of some numerical experiments to show the behaviour of Single–Newton and Simplified–Newton schemes when they are used to solve the stage equations of Radau IIA methods of orders 5 and 7.

In order to compare the performance of the algorithms, we have implemented the following four variable stepsize codes:

- R5SN based on Radau IIA of order 5 + Single–Newton
- R5CSN based on Radau IIA of order 5 + Simplified–Newton
- R7SN based on Radau IIA of order 7 + Single–Newton
- R7CSN based on Radau IIA of order 7 + Simplified–Newton

These implementations, not intended to compete with standard codes, estimate the local error by the classical Richardson extrapolation technique. The stepsize changing policy has been based on the usual formula

$$h_{n+2} = \theta_n(\mathrm{Tol}/\|\mathrm{Est}\|_\infty)^{1/(p+1)} h_n, \qquad n = 0, 2, 4, \ldots$$

after two accepted steps of length h_n, but halving the stepsize when the local error estimate Est is bigger than the prescribed tolerance Tol (in order to save some calculations already done). In the above formula, p denotes the classical order of the Runge-Kutta considered and θ_n, is a parameter that ranges the interval $[0.6, 0.9]$, being typically 0.9 if there is not many step rejections (by convergence problems of the iteration or by the local error estimator) in the integration process.

To solve the implicit equations (1.2) and to advance from t_n to $t_{n+2} = t_n + 2h_n$ for $n = 0, 2, 4, \ldots$, we proceed as follows (assuming that there are no problem

of convergence for the iteration considered)

(1) We evaluate the Jacobian matrix $J_n = \partial f/\partial y(t_n, y_n)$ at every grid-point $(n = 0, 1, \ldots)$.

(2) We apply the iteration considered (single-Newton or Simplified-Newton) to get the internal stages $Y_{i,n}\,(i = 1, \ldots, s)$ of the first step $t_n \to t_{n+1} = t_n + h_n$. We use as initial guesses, $Y_{i,n}^{(0)}\,(i = 1, \ldots, s)$, those given by the interpolation of Lagrange of the internal stages of the preceding step $Y_{i,n-1}\,(i = 1, \ldots, s)$ and of y_{n-1} (for the initial step $n = 0$, the initial guesses are taken as y_0). Observe that we also need one LU factorization of a matrix of the form $(I - \tau h_n J_n)$ for the single-Newton iteration, and two LU factorizations (both complex or one real and another complex) for the simplified-Newton iteration. The convergence of the iteration process is reached when

$$\max_{1\le i\le s}\left(\|Y_{i,n}^{(l)} - Y_{i,n}^{(l-1)}\|_\infty\right) \le 0.01\ \text{Tol}, \qquad l = 1, \ldots, 10,$$

and then we take $Y_{i,n} := Y_{i,n}^{(l)}$. If the convergence condition has not been attained after ten iterations or if for some r the quotient

$$\tau_r = \max_{1\le i\le s}(\|Y_{i,n}^{(r)} - Y_{i,n}^{(r-1)}\|_\infty) / \max_{1\le i\le s}(\|Y_{i,n}^{(r-1)} - Y_{i,n}^{(r-2)}\|_\infty), \qquad r = 2, 3, \ldots, 10,$$

is greater than one, we consider that there is no convergence for the stepsize taken.

(3) We repeat the process in (2) for the second step, i.e. the step from $t_{n+1} \to t_{n+2} = t_n + 2h_n$. Here we use the LU factorizations already done in (2), and as initial guesses, $Y_{i,n+1}^{(0)}\ (i = 1, \ldots, s)$, those obtained from the Lagrange interpolation of the internal stages of the first step, $Y_{i,n}\ (i = 1, \ldots, s)$ and of y_n.

(4) We calculate a local error estimator by doing a new step from t_n to $t_{n+2} = t_n + 2h_n$, by using this time as step-size $h_n^* = 2h_n$. Now, new LU factorizations of matrices of the form $(I - \tau h_n^* J_{n+1})$ are needed. The initial guesses for the double step, are calculated from the previous ones, $Y_{i,n}, Y_{i,n+1},\ (i = 1, \ldots, s)$, by doing the Lagrange polynomial interpolation.

If there is not convergence of the iteration considered at some of the steps (2), (3) or (4) above, we restart the process again at step (2), but halving the step h_n in order to save some previous calculations.

Note that this code evaluates the Jacobian matrix at each gridpoint and does one LU factorization at each step. A tuning of the code reusing the Jacobian matrix and/or the LU factorized matrix when it is possible, such as is done in EPISODE or RADAU5 for example, could improve its efficiency. However, in order to test the performance of the iterative schemes, since the four codes work in the same conditions, the adopted simple strategy can be enough.

Here we present the results obtained with the following three problems taken from the related literature and chosen to have three different dimensions:

- The van der Pol oscillator (see e. g. [11] pp. 144), of dimension 2, integrated from 0 to 20.
- The Ring Modulator (see the CWI testset [14]) that has dimension 15.
- The CUSP problem (see e. g. [11] pp. 147), taking $N = 32$ so that the problem has dimension 96.

The computations have been carried out in a Pentium pro based computer with Linux as operating system.

The CPU time (measured in seconds) taken by the four integrators together with the logarithm of the endpoint global error GE is displayed in figures 4 to 4. In order to get a better understanding of the behaviour of the codes we give in Tables 3 to 5, for some error tolerances, the total number of iterations (NIT) required by the codes in the integration as well as the total number of LU factorizations (NLU), successful steps (NACC), the number of steps on which convergence of the iterative scheme is not achieved (NRIT) and the number of steps on which the local error test was not satisfied (NREST). Note that for R5SNC half of the LU factorizations correspond to $m \times m$ complex matrices while for R7SNC all of the LU factorizations correspond to $m \times m$ complex matrices.

In general, Single–Newton schemes employ more iterations to converge than Simplified–Newton schemes and codes using Single–Newton iterations integrate the problems with a few more steps than codes using Simplified–Newton, due to the fact that with these last schemes there are fewer steps in which the convergence of the iterative scheme is not achieved. On the other hand, seventh order method integrates the problems with fewer steps but with more iterations per step than the fifth order method (assuming that both use the same kind of iterative scheme).

Now we will comment briefly the behaviour of the codes with each one of the three problems. First, since the van der Pol is a two–dimensional problem, the computational cost is mainly affected by the cost of the iterations. As can be seen in figure 4 and in Table 3, even though Single–Newton schemes converge with more iterations than Simplified–Newton, since the cost of each iteration is more expensive in this last case, Single–Newton iterations are more efficient than simplified–Newton ones for this problem. In general with low dimensional problems we have not found great differences in efficiency between both classes of schemes. Concerning the behaviour of the methods of order 5 and 7, as expected the higher order method is more efficient for low tolerances while the method of order 5 is preferable for large tolerances.

Table 3
van der Pol problem

	Tol	NIT	NLU	NACC	NRIT	NREST	Log(GE)
R5SNC	10^{-5}	24050	7112	3418	312	144	-3.7171
	10^{-7}	39196	13406	6538	183	290	-5.3862
	10^{-9}	71524	26686	13190	25	306	-6.7644
R5SN	10^{-5}	24728	3594	3438	362	64	-3.5898
	10^{-7}	49543	6653	6492	195	246	-5.4003
	10^{-9}	96309	13327	13176	27	298	-6.7905
R7SNC	10^{-5}	20539	5120	2410	464	34	-4.3188
	10^{-7}	25996	6852	3324	287	102	-5.3982
	10^{-9}	39064	11434	5588	178	210	-7.8742
R7SN	10^{-5}	27775	2965	2724	592	2	-4.9363
	10^{-7}	37929	3857	3678	464	4	-5.9823
	10^{-9}	60473	6306	6158	295	194	-8.1881

Table 4
Ring modulator problem

	Tol	NIT	NLU	NACC	NRIT	NREST	Log(GE)
R5SNC	10^{-5}	154299	92988	44958	51	3060	-2.6725
	10^{-7}	349226	213530	102834	20	7858	-4.4781
	10^{-9}	714432	436514	211490	7	10710	-5.2241
R5SN	10^{-5}	331168	48259	46690	777	2448	-2.7365
	10^{-7}	695589	108887	104654	890	7106	-4.4843
	10^{-9}	1384071	219704	212666	1129	12118	-5.2742
R7SNC	10^{-5}	82350	41404	20322	114	736	-2.7866
	10^{-7}	162526	82512	40116	57	2274	-4.6380
	10^{-9}	301844	153424	74380	23	4664	-5.8876
R7SN	10^{-5}	251423	30715	28760	2697	16	-3.5881
	10^{-7}	410514	49696	47586	2334	789	-5.1354
	10^{-9}	677947	81573	78834	1569	3624	-6.0069

Table 5
CUSP problem

	Tol	NIT	NLU	NACC	NRIT	NREST	Log(GE)
R5SNC	10^{-5}	1770	544	250	55	6	-6.2808
	10^{-7}	2524	740	350	37	20	-7.7230
	10^{-9}	3715	1184	584	23	16	-9.5152
R5SN	10^{-5}	1833	269	248	55	0	-6.4022
	10^{-7}	2905	373	350	46	6	-7.6808
	10^{-9}	4610	608	596	23	20	-9.7380
R7SNC	10^{-5}	1376	464	212	55	0	-6.2861
	10^{-7}	2080	552	262	51	4	-7.7474
	10^{-9}	2969	764	368	36	22	-9.9585
R7SN	10^{-5}	1712	246	222	58	0	-6.4162
	10^{-7}	2642	306	290	61	0	-8.4424
	10^{-9}	3906	411	392	51	10	-9.9907

With the Ring Modulator problem, that has dimension $m = 15$, both the LU factorizations and the iterations contribute in a similar way in the final computational cost. In this problem, even though the total number of iterations given by the methods using Single–Newton iteration is considerably greater than for methods using Simplified–Newton schemes, the cost of the LU factorizations makes them less efficient when Simplified-Newton iteration is implemented.

We must notice at this point that in some computers such as those based on 64 bits DEC alpha processors, the Fortran compiler is able to optimize the complex operations in a particularly efficient way, improving therefore the performance of the methods using Simplified–Newton iteration. In figure 4 we present the efficiency plot obtained for the Ring Modulator problem in a DEC alpha 3000/500. As it can be seen, the method of order 7 using simplified–Newton (that has to factorize and solve two 15×15 complex systems) is now more efficient than the same method using single–Newton iteration.

With respect to the order, in the plot we can see that seventh order codes are more efficient than fifth order ones for all the values of the tolerances. This can be explained if we take into account that the seventh order method gives fewer steps and consequently it requires a smaller number of LU factorizations than fifth order method, and this compensates the greater number of iterations in the case of the method of order 7.

Finally, let us comment on the results obtained in integrating the CUSP problem. Since it has dimension $m = 96$, the computational cost is in practice determined by the LU factorizations. As it can be seen in figure 4, single–Newton schemes are clearly more efficient than simplified–Newton (even in 64 bits processors). Moreover, taking into account that single–Newton require at each step the LU factorization of an $m \times m$ matrix independently of the number of stages of the method, the seventh order method will be more efficient because it gives fewer steps than the fifth order method. On the contrary, in the case of simplified Newton iteration, the computational cost involved in the LU factorizations is not the same for the methods of order 5 and order 7. Thus, for this problem only for very low tolerances the four stage formula is more efficient than the 3 stage one.

In conclusion, Single–Newton schemes are a reliable and practical alternative to Simplified Newton iterations particularly for high dimensional problems. Moreover, this class of iterative schemes are very suitable for the solution of the stage equations of Runge–Kutta methods of high order, in particular when high precision is required or well the problem to be solved has high dimension.

Acknowledgements

We want to thank Professor M. Calvo for his careful reading of the manuscript and his useful suggestions.

References

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

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

[3] J. C. Butcher, *The numerical analysis of ordinary differential equations*, John Wiley & Sons., Chichester, Wiley, 1987.

[4] M Calvo, S. González–Pinto, and J. I. Montijano, *On the iterative solution of the algebraic equations in fully implicit Runge–Kutta methods*, Submitted to Numerical algorithms, 1998.

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

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

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

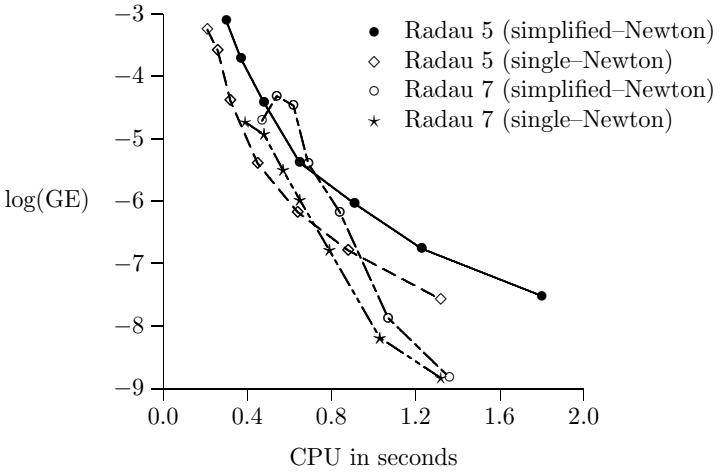

Fig. 1. van der Pol.

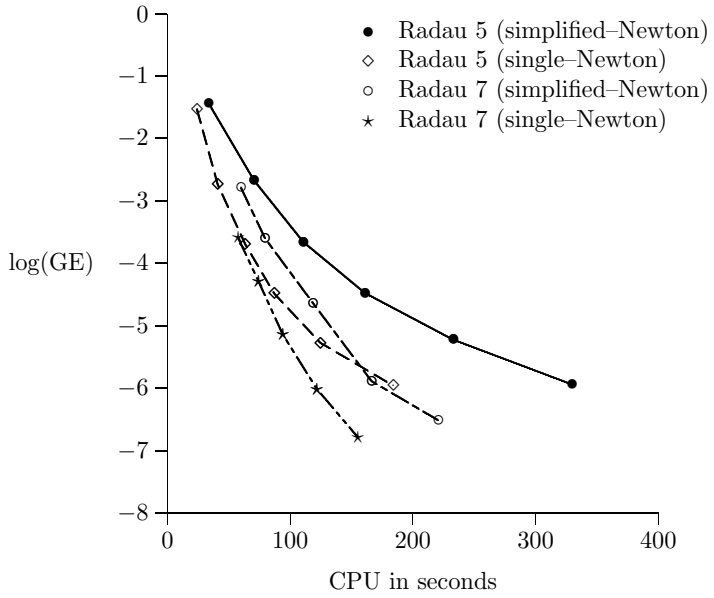

Fig. 2. Ring modulator.

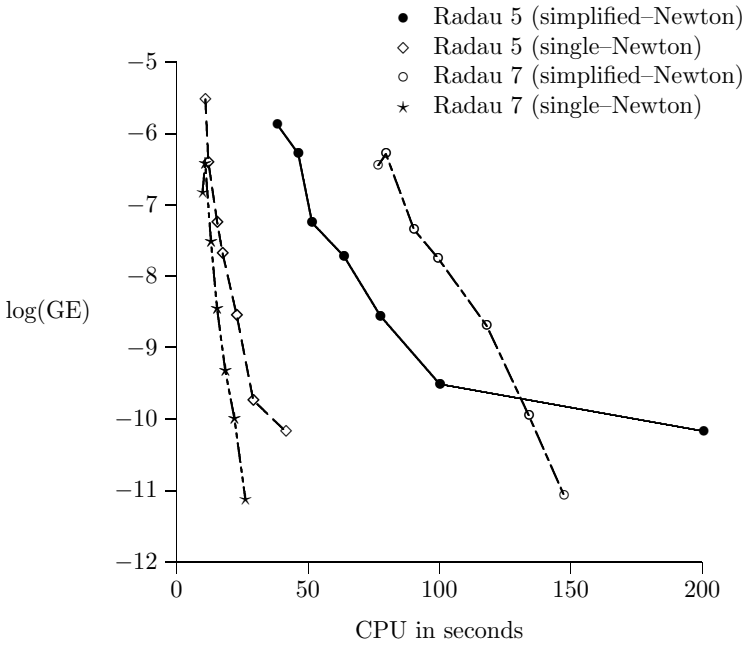

Fig. 3. CUSP problem.

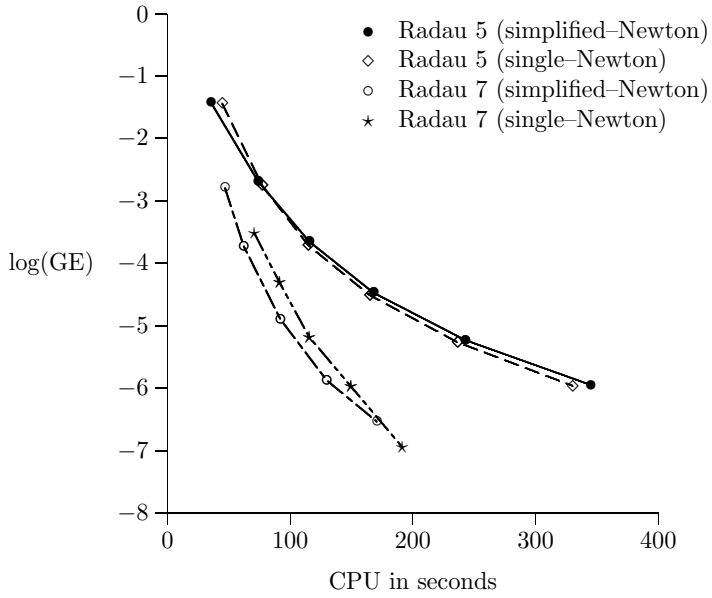

Fig. 4. Ring modulator on DEC Alpha.

[8] W. H. Enright and J. D. Pryce, *Two Fortran packages for assessing Initial Value Methods*, ACM Trans. Math. Software, No.13, vol. 1 (1987), pp. 1-27.

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

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

[11] E. Hairer and G. Wanner, *Solving ordinary differential equations II*, Springer Verlag, Berlin, 1996.

[12] E. Hairer and G. Wanner, *Stiff differential equations solved by Radau methods*, Talk presented at Coimbra, 1998.

[13] N. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[14] 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.

[15] M. Marden, *Geometry of polynomials*, American Mathematical Society, Providence, Rhode Island, 2nd edit. 1966.

[16] S. Pérez–Rodríguez , Tesina de Licenciatura, Universidad de La Laguna, 1996.