STABILIZED STARTING ALGORITHMS FOR COLLOCATION RUNGE-KUTTA METHODS [1]

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

[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 propose a technique to stabilize some starting algorithms often used in the Newton-type iterations appearing when collocation Runge-Kutta methods are applied to solve stiff initial value problems. By following the ideas given in [4], we analyze the order (classical and stiff) of the new starting algorithms and pay special attention to their error amplifying functions. From the computational point of view, the new algorithms require the solution of an additional linear system per integration step, but as shown in the numerical experiments, this extra cost is compensated in most of problems, by their better stability properties.

AMS subject classifications: 65L05

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

1 Introduction.

We consider the numerical solution of stiff initial value problems

$$y'(t) = f(t, y(t)), \quad y(0) = u_0 \in \mathbb{R}^m, \quad t \in [0, T], \tag{1.1}$$

where $f : [0, T] \times \mathbb{R}^m \to \mathbb{R}^m$ is assumed to be sufficiently smooth in a tubular neighborhood of the unique solution $y(t),\ t \in [0, T]$ of (1.1).

[1] Partially supported by project DGES PB97-1018.

For the solution of (1.1) we consider implicit Runge–Kutta methods in which the time stepping from the current point (t_0, y_0) to $(t_1 = t_0 + h, y_1)$ is given by

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i f(t_0 + c_i h, X_i), \tag{1.2}$$

and the internal stages X_i are calculated from the system

$$X_i = y_0 + h \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, X_j) \quad (i = 1, \ldots, s) \tag{1.3}$$

that will be supposed to have a unique solution.

We will denote by $c = (c_1, \ldots, c_s)^T$ the node vector, by $b = (b_1, \ldots, b_s)^T$ the weight vector and by $A = (a_{ij}) \in \mathbb{R}^{s \times s}$ the coefficient matrix of the Runge–Kutta method. As usual, we will assume that $b^T e = 1$ and $Ae = c$, where $e = (1, \ldots, 1)^T \in \mathbb{R}^s$.

Once the algebraic system (1.3) has been solved, typically by some modified Newton iteration, we want to compute good approximations Y_i^0 to the internal stages Y_i of the next step $(t_1 = t_0 + h, y_1) \to (t_2 = t_1 + rh, y_2)$ to start the iterations for the new system,

$$Y_i = y_1 + \bar{h} \sum_{j=1}^{s} a_{ij} f(t_1 + c_j \bar{h}, Y_j) \quad (i = 1, \ldots, s), \tag{1.4}$$

where $r = \bar{h}/h$ is supposed to be of moderate size.

It is remarkable that in the numerical solution of stiff systems, little attention has been addressed to starting algorithms. In [4] the properties of several starting methods, including the most relevant ones such as those based on Lagrange interpolation (see e.g. [5, Chap. IV.8], [9]) were analyzed. However we will show that the choice of the starting algorithm may substantially change the performance of a RK code, even if the underlying formula such as Radau IIA method possesses excellent properties of stability and order.

The main motivation of this paper comes from the fact that some starting algorithms, such as Lagrange interpolation of the internal stages X_i $(i = 1, \ldots, s)$ of the preceding step and of y_0, may present stability problems in some cases, since they can amplify excessively the global error and round-off errors accumulated at the current point t_0. To avoid this circumstance we propose a technique to stabilize the starting algorithms, which is based on the consideration of the Prothero and Robinson model [8]. In any case, we will

consider in this paper starting algorithms based only on information from the previous step.

The rest of the paper is organized as follows. In section 2 we outline the effect of the stability properties in starting algorithms commonly used in practice when the solution of stiff systems is considered. In section 3 we propose several alternatives to stabilizing the most promising starting algorithms studied in [4] and analyze their convergence and stability properties. In section 4 we present some numerical experiments in order to verify the theory presented in previous sections.

2 The effect of stability on starting algorithms based on Lagrange interpolation

In order to begin our discussion we will consider collocation RK methods of s stages with all collocation knots non–zero, although most of the work can be adapted to other methods such as LobattoIIIA, Radau IA, Lobatto IIIC, etc. The starting algorithms obtained from the Lagrange interpolation of the stages and of y_0 are given by

$$Y_i^0 = l_0(1+rc_i)y_0 + \sum_{j=1}^{s} l_j(1+rc_i)X_j, \quad 1 \le i \le s. \tag{2.1}$$

where

$$l_j(t) = \frac{\Pi(t)}{(t-c_j)\Pi'(c_j)},\ j = 0,\dots,s, \text{ and } \Pi(t) = (t-c_0)\cdots(t-c_s), \tag{2.2}$$

being $c_0 = 0$. This starting algorithms are typically used in practical codes and, as shown in [4], they have classical order s (this means order s for nonstiff problems), and also order s for stiff problems.

On the other hand, if we apply these starting algorithms Y_i^0 to the Prothero and Robinson test problem [8]

$$y' = \lambda(y-\phi(t)) + \phi'(t), \quad y(0) = \phi(0), \tag{2.3}$$

where $\mathrm{Re}(\lambda) \le 0$, then (see [4]) for each $1 \le i \le s$,

$$Y_i^0 = Y_i^0(z) = R_i^0(z)(y_0 - \phi(t_0)) + \phi(t_0) + \sum_{j\ge 1} \frac{\phi^{(j)}(t_0)}{j!} v_{ij}^0(z)h^j. \tag{2.4}$$

Here $z = \lambda h$, the error amplifying functions are given by

$$R_i^0(z) = l_0(1 + rc_i) + \alpha_i^T (I - zA)^{-1} e, \quad 1 \le i \le s, \tag{2.5}$$

with

$$\alpha_i^T = (l_1(1 + rc_i), \ldots, l_s(1 + rc_i)), \quad 1 \le i \le s, \tag{2.6}$$

and the coefficients $v_{ij}^0(z)$ by

$$v_{ij}^0(z) = \alpha_i^T A(I - zA)^{-1}(-zc^j + jc^{j-1}), \quad 1 \le i \le s,\ j \ge 1. \tag{2.7}$$

(here and in the following, the powers of a vector will be understood as taking powers in its components, and by e_i we will denote the $i-$vector of the canonical base of $\mathbb{R}^s$).

Further, the internal stages Y_i of the underlying RK method (A, b) satisfy

$$\begin{aligned} Y_i = Y_i(z) &= R_i(z)(y_0 - \phi(t_0)) \\ &\quad + \phi(t_0) + \sum_{j \ge 1} \frac{\phi^{(j)}(t_0)}{j!} v_{ij}(z) h^j, \quad 1 \le i \le s, \end{aligned} \tag{2.8}$$

where,

$$\begin{gathered} R_i(z) = R(z)\varphi_i(rz), \qquad \text{with} \\ R(z) = 1 + zb^T(I - zA)^{-1}e, \quad \varphi_i(z) = e_i^T(I - zA)^{-1}e. \end{gathered} \tag{2.9}$$

Moreover, the coefficients $v_{ij}(z)$, for each $1 \le i \le s$, satisfy (see [4])

$$\begin{gathered} v_{ij}(z) = (1 + rc_i)^j, \quad (1 \le j \le s), \\ v_{ij}(\infty) = (1 + rc_i)^j,\ \forall\, j > s, \\ \sup_{\operatorname{Re} z \le 0} |v_{ij}(z)| < \infty,\ \ \forall\, j > s, \end{gathered} \tag{2.10}$$

provided that the method (A, b) is ASI-stable, i.e., the matrix $I - zA$ is non singular for all $\operatorname{Re} z \le 0$ and

$$\sup_{\operatorname{Re} z \le 0} |(I - zA)^{-1}|_2 < \infty.$$

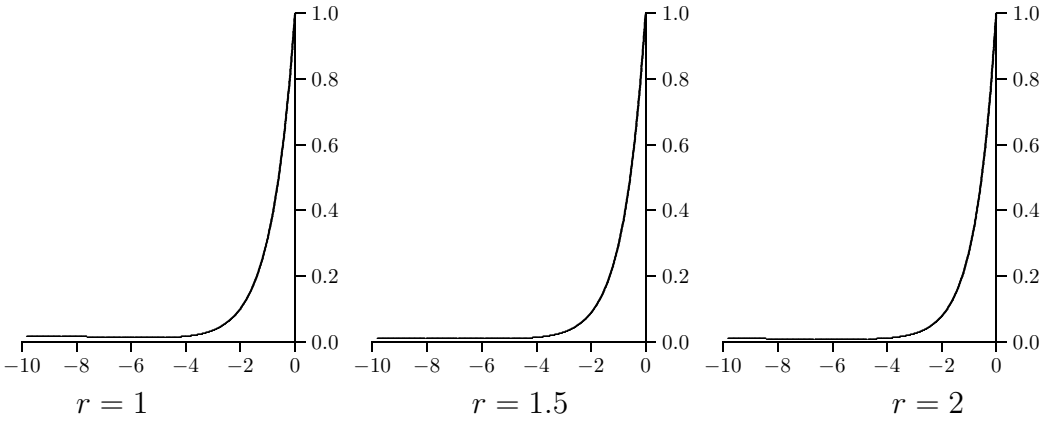


Fig. 1. Amplifying function $R_3(x)$ of the stage Y_3.

Since the approximation Y_i^0 has stiff order s, the error satisfies

$$Y_i(z) - Y_i^0(z) = (R_i(z) - R_i^0(z))(y_0 - \phi(t_0)) + \mathcal{O}(h^{s+1}), \quad 1 \leq i \leq s,$$

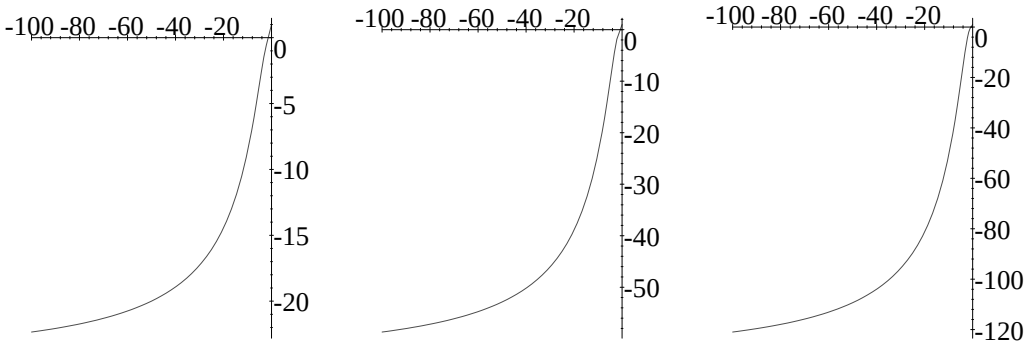
where the term $\mathcal{O}(h^{s+1})$ is bounded independently of the stiffness of the problem (i.e. independently of $z = \lambda h$). From this expression we see that the global error at the point t_0, $y_0 - \phi(t_0)$, can affect the error of the starting algorithm, if the error amplifying factor $R_i(z) - R_i^0(z)$ is large. In order to obtain an idea of the magnitude of these error amplification factors, we have plotted in figures 1 and 2 the functions $R_3(z)$ and $R_3^0(z)$ respectively for the three-stage Radau IIA method, with z varying on the nonpositive real semiaxis and for values of stepsize ratios $r = 1, 3/2, 2$. We have only considered the amplifying functions associated to the last stage (Y_3) because this is the one exhibiting the highest absolute value of the error amplifying factor. As we can see, $R_3(0) = 1$ and the function $R_3(z)$ goes to zero quickly when z becomes negative. That is the reason why we have displayed the functions separately instead of plotting the error $R_3(z) - R_3^0(z)$.

Particularly important for very stiff problems is the point $z = \infty$. Since $R_i(\infty) = 0$ and $R_i^0(\infty) = l_0(1 + rc_i)$, the error at this point can be expressed in the form

$$Y_i(\infty) - Y_i^0(\infty) = l_0(1 + rc_i)(y_0 - \phi(t_0)) + \mathcal{O}(h^{s+1}), \quad 1 \leq i \leq s$$

and the error amplifying factor is $l_0(1 + rc_i)$. For the three-stage Radau IIA method, for which $c_1 = (4 - \sqrt{6})/10$, $c_2 = (4 + \sqrt{6})/10$, $c_3 = 1$ and

$$l_0(1 + rc_3) = -\frac{(1 - c_1 + r)(1 - c_2 + r)r}{c_1 c_2 c_3},$$



$r = 1$ $r = 1.5$ $r = 2$

Fig. 2. Amplifying function for Y_3^0.

we get for the values $r = 1, 3/2$ and 2 that,

$$l_0(1 + c_3) = -25, \qquad l_0(1 + 1.5c_3) = -65.25 \qquad l_0(1 + 2c_3) = -134.$$

The error amplification can cause certain difficulties in the convergence of the Newton-type iterations when integrating certain classes of stiff problems. In practice this fact generates many step rejections in the integration due to problems in the convergence of the iterative scheme or even it can make the code fail to complete the integration. We will see more about this in the section devoted to numerical experiments.

Let us consider now as starting algorithms the Lagrange interpolation polynomial based just on the internal stages X_i given by

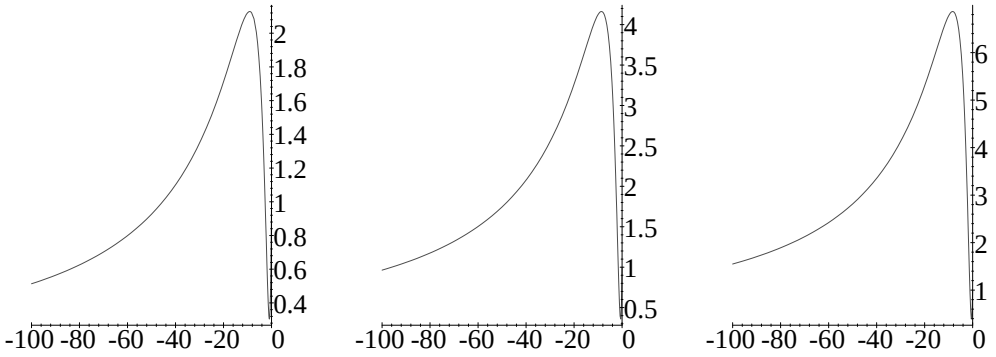
$$\hat{Y}_i^0 = \sum_{j=1}^{s} \hat{l}_j(1 + rc_i)X_j, \quad 1 \le i \le s, \tag{2.11}$$

where

$$\hat{l}_j(t) = \frac{\hat{\Pi}(t)}{(t - c_j)\hat{\Pi}'(c_j)}, \quad j = 1, \ldots, s, \tag{2.12}$$

with $\hat{\Pi}(t) = (t - c_1) \cdots (t - c_s)$. For the Prothero and Robinson test (2.3) (see [4]) we get

$$\hat{Y}_i^0 = \hat{R}_i^0(z)(y_0 - \phi(t_0)) + \phi(t_0) + \sum_{j \ge 1} \frac{\phi^{(j)}(t_0)}{j!} \hat{v}_{ij}^0(z)h^j, \quad 1 \le i \le s, \tag{2.13}$$



$r = 1$ $r = 1.5$ $r = 2$

Fig. 3. Amplifying function for $\hat{Y}_3^0$.

where the error amplifying functions are given by,

$$\hat{R}_i^0(z) = \hat{\alpha}_i^T (I - zA)^{-1} e, \quad 1 \leq i \leq s, \tag{2.14}$$

with $\hat{\alpha}_i^T = (\hat{l}_1(1 + rc_i), \ldots, \hat{l}_s(1 + rc_i))$, and the coefficients $\hat{v}_{ij}^0(z)$ by,

$$\hat{v}_{ij}^0(z) = \hat{\alpha}_i^T A (I - zA)^{-1} (-zc^j + jc^{j-1}), \quad 1 \leq i \leq s,\ j \geq 1. \tag{2.15}$$

In figure 2 we have plotted as in previous figures the amplifying function $\hat{R}_3^0(z)$ corresponding to the algorithm $\hat{Y}_3^0$ for the Radau IIA method of order 5.

In this case the amplifying factor vanishes at the infinite point and

$$Y_i(\infty) - \hat{Y}_i^0(\infty) = \mathcal{O}(h^s), \quad 1 \leq i \leq s.$$

Note that a zero value of the amplifying function at the infinity point does not ensure small absolute values at every complex z in the left half plane, even for step-ratio $r = 1$.

After seeing figures 2 and 2, from a stability point of view we can expect this starting algorithm to be more adequate than the one given by (2.1). However, the first iterant (2.11) has order (classical and stiff) $s - 1$ and this makes it in many cases (if stability is not a crucial requirement) less efficient than the classical algorithm (2.1) (see [4]).

3 Developing stabilized starting algorithms

After the considerations in the previous section, we are interested in the development of starting algorithms with "small" amplifying functions and orders as high as possible. When the Runge-Kutta method considered is fully-implicit (as in the case of collocation methods) and the IVP (1.1) is stiff and nonlinear, the algebraic system (1.3) is usually solved by some modification of the Newton iteration, that involves in most cases the LU factorization of a real matrix $(I - h\beta J)$ where β is a given positive parameter and J is the jacobian matrix $\partial f/\partial y$ evaluated at (t_0, y_0) or well at some other previous point such that

$$J - \partial f/\partial y(t_0, y_0) = \mathcal{O}(h).$$

Our main idea here will be to use this already factorized matrix to get starting algorithms with appropriate amplifying functions.

If we denote by

$$\begin{aligned} P(\tau) &:= l_0(\tau)y_0 + \textstyle\sum_{j=1}^{s} l_j(\tau)X_j, \\ \hat{P}(\tau) &:= \textstyle\sum_{j=1}^{s} \hat{l}_j(\tau)X_j, \end{aligned} \tag{3.1}$$

where $\{l_j(\tau),\ (j = 0, \ldots, s)\}$ and $\{\hat{l}_j(\tau),\ (j = 1, \ldots, s)\}$ are respectively defined by (2.2) and (2.12), we have that the starting algorithms (2.1) and (2.11) can be written in the form

$$Y_i^0 = P(1 + rc_i), \quad \hat{Y}_i^0 = \hat{P}(1 + rc_i), \quad 1 \le i \le s. \tag{3.2}$$

Then, we define new starting algorithms by,

$$\tilde{Y}_i^0 = \hat{Y}_i^0 + (I - h\beta J)^{-1}(Y_i^0 - \hat{Y}_i^0), \quad 1 \le i \le s, \tag{3.3}$$

so that we expect it to behave as Y_i^0 for h sufficiently small and not very stiff problems, and as $\hat{Y}_i^0$ when the differential problem has a very stiff behavior.

These algorithms can be rewritten in such a way that they only involve the solution of an extra linear system (independently of the number of stages of the RK method) with $(I - h\beta J)$ (already factorized) as the coefficient matrix.

To this end, we use the following equality which is straightforward to prove,

$$l_j(\tau) - \hat{l}_j(\tau) = \frac{\hat{\Pi}(\tau)}{c_j \hat{\Pi}'(c_j)}, \quad 1 \leq j \leq s, \quad l_0(\tau) = \frac{(-1)^s}{c_1 \cdots c_s} \hat{\Pi}(\tau). \tag{3.4}$$

¿From here we get,

$$P(\tau) - \hat{P}(\tau) = \hat{\Pi}(\tau) \left(\frac{(-1)^s}{c_1 \cdots c_s} y_0 + \sum_{j=1}^{s} \frac{1}{c_j \hat{\Pi}'(c_j)} X_j \right),$$

hence the new starting algorithms can be written in the form

$$\begin{gathered} \tilde{Y}_i^0 = \hat{Y}_i^0 + \hat{\Pi}(1 + rc_i)(I - h\beta J)^{-1} V, \quad 1 \leq i \leq s, \quad \text{with} \\ V := (-1)^s (c_1 \cdots c_s)^{-1} y_0 + \textstyle\sum_{j=1}^{s} (c_j \hat{\Pi}'(c_j))^{-1} X_j. \end{gathered} \tag{3.5}$$

Of course, an alternative formula by using divided differences can be employed to write all the starting algorithms above given.

In order to make more readable the remainder of the paper we will give now some results that are immediate consequence of the previous work in [4].

Proposition 3.1 *Let us consider an s–stage collocation RK method with $c_i \neq c_j, \forall i \neq j$ and $c_i \neq 0$, $\forall i$. Then*

- *$Y_i - Y_i^0 = \mathcal{O}(h^{s+1})$ and $Y_i - \hat{Y}_i^0 = \mathcal{O}(h^s)$ for nonstiff problems.*
  This result is consequence of [4, th. 2.4.].
- *If the RK method is ASI–stable, $Y_i - Y_i^0 = \mathcal{O}(h^{s+1})$ and $Y_i - \hat{Y}_i^0 = \mathcal{O}(h^s)$ on the Prothero and Robinson model, $\hat{R}_i^0(\infty) = 0$ and $R_i^0(\infty)$ is bounded.*
  This result is consequence of [4, th. 3.4.]. Recall that in our case, ASI-stability implies AS-stability because the matrix A is nonsingular.
- *If the RK method is diagonally stable (i.e., there exists a positive definite diagonal matrix D such that $Q = DA + A^T D$ is positive definite), $Y_i - Y_i^0 = \mathcal{O}(h^{s+1})$ and $Y_i - \hat{Y}_i^0 = \mathcal{O}(h^s)$ for dissipative differential systems (for a detailed description of this class of problems, see for example [1, Chap. I]).*
  This result is a consequence of [4, th. 3.7.].

Proposition 3.2 *Let us consider an s–stage collocation RK method with $c_i \neq c_j, \forall i \neq j$ and $c_i \neq 0, \forall i$. Then, by considering the starting algorithms Z_i^0 given later by (3.8), we have that*

- *$Y_i - Z_i^0 = \mathcal{O}(h^{s+2})$ for nonstiff problems.*
  This result is consequence of [4, th. 2.5.].

- *If the RK method is ASI–stable, $Y_i - Z_i^0 = \mathcal{O}(h^{s+1})$ on the Prothero and Robinson model.*
  This result is consequence of [4, th. 3.5.].
- *If the RK method is diagonally stable, $Y_i - Z_i^0 = \mathcal{O}(h^{s+1})$ for dissipative differential systems.*
  This result is a consequence of [4, th. 3.8.].

Our main result concerning the order and the stability of the new algorithms is given in the following

Theorem 3.3 *Let us consider an s–stages collocation Runge-Kutta method with $c_i \neq c_j, \forall i \neq j$ and $c_i \neq 0$, $\forall i$. Then, for any constant $\beta > 0$,*

(a) *The starting algorithms (3.3) have classical order s.*

(b) *If the RK method is ASI-stable, they have stiff order $s-1$ on the Prothero and Robinson model and their error amplifying functions satisfy, $\tilde{R}_i^0(\infty) = 0, \;\; i = 1, \ldots, s$.*

(c) *They have stiff order $s-1$ for dissipative differential systems provided that the underlying RK method is diagonally stable.*

PROOF. (a) The starting algorithms can be written in the form,

$$\tilde{Y}_i^0 = -\beta h J (I - \beta h J)^{-1} \hat{Y}_i^0 + (I - h\beta J)^{-1} Y_i^0, \quad 1 \leq i \leq s,$$

and the internal stages of the RK method (A, b) as,

$$Y_i = -\beta h J (I - \beta h J)^{-1} Y_i + (I - h\beta J)^{-1} Y_i, \quad 1 \leq i \leq s.$$

¿From here we get,

$$\begin{aligned} Y_i - \tilde{Y}_i^0 = & -\beta h J (I - \beta h J)^{-1} (Y_i - \hat{Y}_i^0) \\ & + (I - h\beta J)^{-1} (Y_i - Y_i^0), \quad 1 \leq i \leq s. \end{aligned} \tag{3.6}$$

and taking into account that $\beta h J (I - \beta h J)^{-1} = \mathcal{O}(h)$ for the non-stiff case, from the proposition 3.1 we arrive at the desired result.

(b) For the Prothero and Robinson model (2.3) we have,

$$\tilde{Y}_i^0 = \tilde{R}_i^0(z)(y_0 - \phi(t_0)) + \phi(t_0) + \sum_{j \geq 1} \frac{\phi^{(j)}(t_0)}{j!} \tilde{v}_{ij}^0(z) h^j, \quad 1 \leq i \leq s,$$

where

$$\tilde{R}_i^0(z) = \hat{R}_i^0(z) + (1-\beta z)^{-1}(R_i^0(z) - \hat{R}_i^0(z)), \quad 1 \le i \le s, \tag{3.7}$$

and

$$\tilde{v}_{ij}^0(z) = \hat{v}_{ij}^0(z) + (1-\beta z)^{-1}(v_{ij}^0(z) - \hat{v}_{ij}^0(z)), \quad 1 \le i \le s,\ j \ge 1,$$

with $R_i^0(z)$, $v_{ij}^0(z)$, $\hat{R}_i^0(z)$, $\hat{v}_{ij}^0(z)$ defined respectively by (2.5), (2.7), (2.14) and (2.15).

¿From here, by considering again (3.6) the statement immediately follows after using the proposition 3.1.

(c) As a consequence of the von Neumann-theorem (see e.g. [5, Chap. IV.11]) we have

$$\begin{aligned} &| - \beta h J (I - \beta h J)^{-1}|_2 \le \sup_{\mathrm{Re}\, z \le 0} | - z(1-z)^{-1}| = 1, \\ &|(I - \beta h J)^{-1}|_2 \le \sup_{\mathrm{Re}\, z \le 0} |(1-z)^{-1}| = 1. \end{aligned}$$

¿From here and applying the proposition 3.1 it is clear that the starting algorithm $\tilde{Y}_i^0$ reaches the stiff order $s-1$. ∎

Remark 3.4 *Note that in (3.3) we are assuming that the starting algorithms Y_i^0 and $\hat{Y}_i^0$ are based on the true internal stages X_i of the previous step. However, in practice these values are computed by some iterative process and therefore additional errors can be introduced. In this paper we have not considered the effect of these errors on the starting algorithms.*

Now we will consider new stabilized starting algorithms based on some algorithms proposed in [4] (see theorem 2.5 and remark 2.2). Let us consider the algorithms given by

$$\begin{aligned} Z_i^0 &= y_1 + rh \textstyle\sum_{j=1}^s a_{ij} F_j \quad (i = 1, \dots, s), \\ F_i &= l_0(1 + rc_i) f(t_0, y_0) + \textstyle\sum_{j=1}^s l_j(1 + rc_i) f(t_0 + hc_j, X_j), \end{aligned} \tag{3.8}$$

with $l_j(t)$ defined by (2.2). They possess classical order $s+1$ and stiff order s for collocation Runge-Kutta methods (if zero is not a collocation knot), but their main inconvenience is that the error amplifying functions go to ∞ when

$z \to \infty$. Note that if the underlying Runge-Kutta method is not *stiffly accurate* then these algorithms will require the computation of one extra derivative function $f(t_0, y_0)$ per integration step.

On the other hand, it is not difficult to see that the algorithms (2.1) can be written as (see [4, th. 2.5])

$$\begin{aligned} Y_i^0 &= y_1 + rh \textstyle\sum_{j=1}^s a_{ij} \hat{F}_j \quad (i = 1, \ldots, s), \\ \hat{F}_i &= \textstyle\sum_{j=1}^s \hat{l}_j(1 + rc_i) f(t_0 + hc_j, X_j), \end{aligned} \tag{3.9}$$

with $\hat{l}_j(t)$ defined by (2.12).

We will now define a new family of starting algorithms depending on s parameters, θ_i $(i = 1, \ldots, s)$ (these parameters might depend on r, for particular choices, mainly due to stability reasons, but in any case they will be of moderate size):

$$\bar{Y}_i^0 = Y_i^0 + \theta_i (I - \beta h J)^{-1} (Z_i^0 - Y_i^0), \quad 1 \le i \le s. \tag{3.10}$$

Notice that by using (3.8), (3.9) and (3.4), the new starting algorithms can be rewritten in the following way,

$$\begin{aligned} \bar{Y}_i^0 &= Y_i^0 + r\theta_i \left(\textstyle\sum_{j=1}^s a_{ij} \hat{\Pi}(1 + rc_j) \right) (I - h\beta J)^{-1} W, \quad 1 \le i \le s, \\ W &:= h \left((-1)^s (c_1 \cdots c_s)^{-1} f(t_0, y_0) + \textstyle\sum_{j=1}^s (c_j \hat{\Pi}'(c_j))^{-1} f(t_0 + hc_j, X_j) \right). \end{aligned} \tag{3.11}$$

¿From here, it is clear that just the solution of an extra linear system (with respect to (3.8)) is needed when implementing the new starting algorithms.

For computational purposes it is better to rewrite (3.11) as a linear combination of the internal stages X_j $(j = 1, \ldots, s)$, y_0 and $hf(t_0, y_0)$, by using the formula (1.3). It is also convenient to write $hf(t_0, y_0)$ as a linear combination of the internal stages of the preceding step if the Runge-Kutta method under consideration is *stiffly accurate*. Thus, the starting algorithms (3.11) can be rewritten (in the case of collocation Runge-Kutta methods, where zero is

assumed not to be a collocation knot) in the following way,

$$\bar{Y}_i^0 = Y_i^0 + r\theta_i \left(\sum_{j=1}^s a_{ij}\hat{\Pi}(1+rc_j)\right)(I-h\beta J)^{-1}W, \quad 1 \le i \le s,$$

$$W = (-1)^s(c_1\cdots c_s)^{-1}hf(t_0,y_0) - \kappa y_0 + \sum_{j=1}^s \tau_j X_j, \tag{3.12}$$

$$\tau^T = ((c_1\hat{\Pi}'(c_1))^{-1},\ldots,(c_s\hat{\Pi}'(c_s))^{-1})^T A^{-1}, \quad \kappa = \tau^T A^{-1} e.$$

Moreover, for the next integration step, and in the case of *stiffly accurate* methods, we can compute $\bar{h}f(t_1,y_1)$ (recall that $\bar{h}=rh$) from,

$$\bar{h}f(t_1,y_1) = r\left(\sum_{j=1}^s u_j X_j - (e_s^T A^{-1}e)y_0\right), \quad u^T = (u_1,\ldots,u_s) = e_s^T A^{-1},$$

which avoids the undesirable evaluation of the derivative function (observe that the evaluations of derivative function can excessively amplify any error committed in the iteration process, i.e. round-off errors, convergence errors, etc).

The following result gives us the orders of the new family of starting algorithms as well as their error amplifying functions,

Theorem 3.5 *Let us consider a collocation Runge-Kutta method (with s distinct knots, all of them nonzero). Then for any constant $\beta > 0$,*

(a) *The family of starting algorithms (3.10) have classical order s. Moreover, for the particular choice $\theta_i = 1$, $(i = 1,\ldots,s)$, the resulting starting algorithms achieves classical order $s+1$.*

(b) *The whole family reaches order s on the Prothero and Robinson model, and the error amplifying functions satisfy*

$$\bar{R}_i^0(\infty) = l_0(1+rc_i) - \theta_i\beta^{-1}r\sum_{j=1}^s a_{ij}l_0(1+rc_j), \quad i=1,\ldots,s. \tag{3.13}$$

Moreover there not exist any particular choice of the parameters such that the resulting starting algorithms reach order $s+1$ on the left half-complex plane. On the other hand, for the particular choice of the parameters

$$\theta_i = \beta l_0(1+rc_i)(r\sum_{j=1}^s a_{ij}l_0(1+rc_j))^{-1}, \quad i=1,\ldots,s, \tag{3.14}$$

and by assuming that the denominators do not vanish, the amplifying func-

tions of the resulting starting algorithms satisfy,

$$\bar{R}_i^0(\infty) = 0, \quad i = 1, \ldots, s. \tag{3.15}$$

(c) *The whole family possesses stiff order s for dissipative problems, provided that the RK method is diagonally stable.*

PROOF. (a),(c) Since the internal stages of the Runge-Kutta method Y_i satisfy

$$Y_i = Y_i + \theta_i (I - h\beta J)^{-1}(Y_i - Y_i),$$

by subtracting (3.10) from the last equation it follows

$$Y_i - \bar{Y}_i^0 = (I - \theta_i (I - \beta h J)^{-1})(Y_i - Y_i^0) + \theta_i (I - h\beta J)^{-1}(Y_i - Z_i^0). \tag{3.16}$$

Now, we conclude the proof of statements (a)-(c) by using the proposition 3.2.

(b) If we consider the Prothero and Robinson model (2.3) for the starting algorithms (3.10) we get the following expressions,

$$\bar{Y}_i^0 = \bar{R}_i^0(z)(y_0 - \phi(t_0)) + \phi(t_0) + \sum_{j \geq 1} \frac{\phi^{(j)}(t_0)}{j!} \bar{v}_{ij}^0(z) h^j, \quad 1 \leq i \leq s,$$

where

$$\bar{R}_i^0(z) = R_i^0(z) + \theta_i (1 - \beta z)^{-1}(S_i^0(z) - R_i^0(z)), \quad 1 \leq i \leq s, \tag{3.17}$$

and the coefficients are given by,

$$\bar{v}_{ij}^0(z) = v_{ij}^0(z) + \theta_i (1 - \beta z)^{-1}(w_{ij}^0(z) - v_{ij}^0(z)), \quad 1 \leq i \leq s, \ j \geq 1,$$

with $S_i^0(z)$ and $w_{ij}^0(z)$ denoting respectively the error amplifying functions and the coefficients of the starting algorithms (3.8).

It is not difficult to see that

$$S_i^0(z) = R(z) + zr \sum_{j=1}^{s} a_{ij} \left(l_0(1 + rc_j) + \sum_{k=1}^{s} l_k (1 + rc_j) \varphi_k(z) \right), \tag{3.18}$$

where $R(z),\ \varphi_i(z)$ are defined by (2.9) and $l_j(\tau)$ by (2.2). Hence,

$$\lim_{z\to\infty}\frac{S_i^0(z)}{1-\beta z}=-r\beta^{-1}\sum_{j=1}^{s}a_{ij}l_0(1+rc_j),\quad i=1,\ldots,s$$

and from here, it follows (3.13), (3.14) and (3.15).

In order to prove that the order $s+1$ cannot be reached on the whole left half-complex plane, we proceed by contradiction. Let us assume that the order $s+1$ is reached at the particular points $z=0$ and $z=\infty$ simultaneously, for a fixed choice of the θ_i-parameters. This would imply that ($v_{i,j}(z)$ below, are the ones in (2.8)),

$$\begin{aligned}\bar{v}_{i,s+1}^0(0)&=v_{i,s+1}^0(0)+\theta_i(w_{i,s+1}^0(0)-v_{i,s+1}^0(0))\\&=v_{i,s+1}(0),\quad 1\le i\le s.\end{aligned}\tag{3.19}$$

On the other hand, since the starting algorithms (3.8) possess classical order $s+1$, then $w_{i,s+1}^0(0)=v_{i,s+1}(0)$ and since the starting algorithms (3.9) possess just classical order s, then $v_{i,s+1}^0(0)\neq v_{i,s+1}(0)$. Hence from (3.19) we get that $\theta_i=1,\ (i=1,\ldots,s)$.

Now, considering $z=\infty$ we get,

$$\bar{v}_{i,s+1}^0(\infty)=v_{i,s+1}^0(\infty).$$

Moreover from (2.7) we have that,

$$v_{i,s+1}^0(\infty)=\alpha_i^T c^{s+1}\tag{3.20}$$

where α_i^T is defined into (2.6). Now, from (3.20) and taking into account that $v_{ij}(\infty)=(1+rc_i)^j$ for all $j\ge 1$ (see (2.10)), it immediately follows that,

$$v_{i,s+1}(\infty)-v_{i,s+1}^0(\infty)=(1+rc_i)^{s+1}-\alpha_i^T c^{s+1},\quad 1\le i\le s.$$

This yields a contradiction because the second side of the last equation is a polynomial on r of degree $s+1$ which cannot be identically null. ∎

Now, in order to compare the error amplifying functions of the starting algorithms above considered, we have chosen again the three-stages Radau IIA as underlying formula, and we have only considered the error amplifying functions associated to the last stage (Y_3), because this is the one exhibiting the highest absolute value for every starting algorithm.

In figures 3 and 3 we display the amplifying functions on $\mathbb{R}^-$ for the starting algorithms given respectively by (3.5) and, (3.12) for the particular choice $\theta_3 = 1$. Again, we have chosen values of stepsize ratios $r = 1, 3/2, 2$ and in the three cases we have taken as parameter β the value $60^{-1/3}r$ which is the corresponding value used in the single–Newton scheme developed in [3] for the Radau IIA method with three stages and which will be used later in our numerical experiments. We do not give any figure for the algorithm (3.12) with θ_i given in (3.14) because it has the same amplifying function as the algorithm (3.5). This fact is stated in the following theorem.

Theorem 3.6 *For any collocation Runge–Kutta method with nonzero knots, the amplifying functions for the starting algorithms (3.5), and (3.12) with θ_i given in (3.14), are identical.*

PROOF. From (2.5)-(2.6) and (2.14) we have that,

$$R_i^0(z) = \frac{p_i(z)}{\det(I - zA)}, \quad \hat{R}_i^0(z) = \frac{\hat{p}_i(z)}{\det(I - zA)}, \quad 1 \leq i \leq s,$$

where

$$\text{degree}(p_i(z)) \leq s, \quad \text{degree}(\hat{p}_i(z)) \leq s - 1.$$

Now, from (3.7) and theorem 3.3 (b) it follows that,

$$\tilde{R}_i^0(z) = \frac{\tilde{p}_i(z)}{(1 - \beta z)\det(I - zA)}, \ \text{degree}(\tilde{p}_i(z)) \leq s, \quad 1 \leq i \leq s. \quad (3.21)$$

Moreover, from (3.13)-(3.15) and (3.17) it follows that,

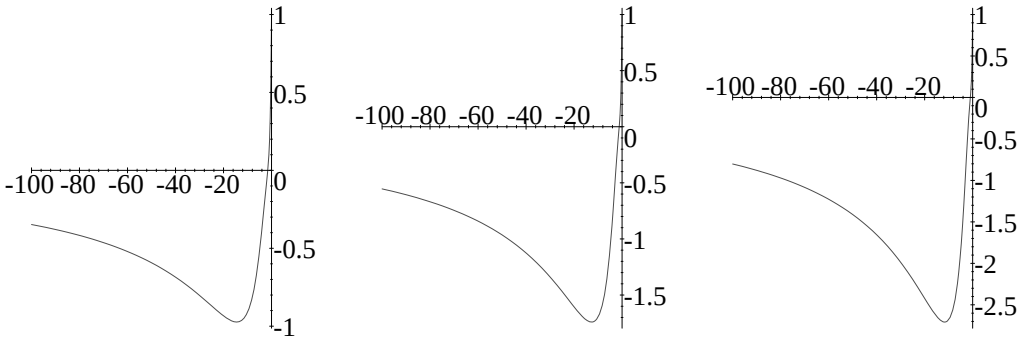
$$\bar{R}_i^0(z) = \frac{\bar{p}_i(z)}{(1 - \beta z)\det(I - zA)}, \ \text{degree}(\bar{p}_i(z)) \leq s, \quad 1 \leq i \leq s. \quad (3.22)$$

On the other hand, since the starting algorithms (3.5) and (3.12)-(3.14) reach classical order s, then by considering the linear test, $y' = \lambda y$, $(z = \lambda h)$, we have that

$$R_i(z) - \tilde{R}_i^0(z) = \mathcal{O}(z^{s+1}), \ (z \to 0), \quad 1 \leq i \leq s,$$

and

$$R_i(z) - \bar{R}_i^0(z) = \mathcal{O}(z^{s+1}), \ (z \to 0), \quad 1 \leq i \leq s.$$



$r = 1$ $r = 1.5$ $r = 2$

Fig. 4. Amplifying function of the starting algorithm (3.5).

Then,

$$\tilde{R}_i^0(z) - \bar{R}_i^0(z) = \mathcal{O}(z^{s+1}),\ (z \to 0), \quad 1 \le i \le s.$$

that implies by using (3.21)-(3.22) that

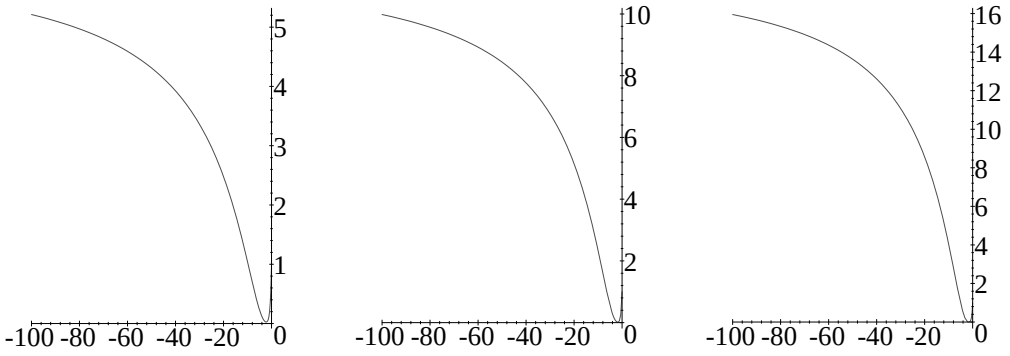
$$\tilde{p}_i(z) - \bar{p}_i(z) = \mathcal{O}(z^{s+1}),\ (z \to 0), \quad 1 \le i \le s.$$

¿From here we conclude $\tilde{p}_i(z) \equiv \bar{p}_i(z)$, which completes the proof. ∎

On the other hand, it is clear that the amplifying functions of the algorithms (3.5) and (3.12)–(3.14) are closer to zero than the amplifying function of the algorithm (3.12) with $\theta_3 = 1$, particularly when z goes to infinity. Thus, a better error propagation can be expected for those first iterants when the RK method is applied to highly stiff problems.

4 Numerical experiments

In this section our goal is to get some numerical evidence about the theoretical results presented in previous sections. First, we present some experiments that confirm numerically the order results obtained for the considered starting algorithms. Next, by means of some numerical examples we show the effect of the error amplifying factor on the error of the starting algorithms. Finally, we present some experiments comparing the efficiency of the integration codes depending on the first iterant selected.



$r = 1$ $r = 1.5$ $r = 2$

Fig. 5. Amplifying function of the starting algorithm (3.12) ($\theta_i = 1$).

In all the cases we have taken as Runge–Kutta method the three–stages, fifth–order Radau IIA and as first iterants we have considered the following four:

L – Lagrange interpolation, given by (2.1),
S1 – Starting algorithms given by (3.5),
S2 – Starting algorithms given by (3.12) with $\theta_i = 1$, $i = 1, 2, 3$,
S3 – Starting algorithms given by (3.12) with θ_i given in (3.14).

In order to verify the order of the starting algorithms, we have considered the following two differential problems:

Problem 1.- Prothero and Robinson problem (2.3) with $\lambda = -10^6$, $\phi(t) = e^{2t}$, $u_0 = \phi(0) = 1$.
Problem 2.- The scalar nonlinear problem (see [10, pp. 202])

$$y' = \lambda(y^3 - \phi(t)^3) + \phi'(t), \quad y(0) = u_0 = 2, \quad \lambda = -10^6, \phi(t) = 1 + e^t$$

Then, for values $h = h_0/2^k$, $k = 0, \ldots, 7$, with $h_0 = 0.4$, and several values of the stepsize ratio r, we have proceeded as follows:

- We advance one step from t_0 to $t_0 + h$ with the RK method, computing the internal stages X_i, $i = 1, 2, 3$ with 16 significative figures.
- Next, we advance a second step from $t_0 + h$ to $t_0 + h + rh$, computing the internal stages Y_i, $i = 1, 2, 3$ also with 16 significative figures.
- we compute the approximations V_i^0, $i = 1, 2, 3$ to the stages Y_i corresponding to each starting algorithm and compute their errors

$$\epsilon(h) := \max_{1 \le i \le 3} |Y_i - V_i^0|$$

We have taken initial values $y_0 = u_0$ and $J = \partial f/\partial y(t_0, y_0)$, that is, no initial error, so that the error of the starting algorithms will only be affected by the

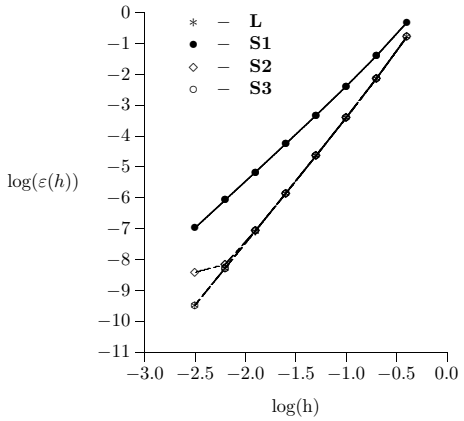


Fig. 6. Problem 1 ($y_0 = u_0$)

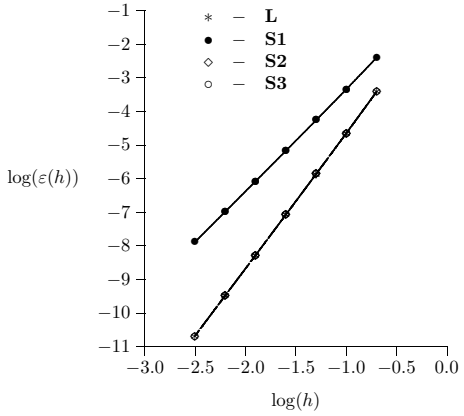


Fig. 7. Problem 2 ($y_0 = u_0$)

local error term.

In figures 4 and 4 we have plotted, for problems 1 and 2 respectively, the points $(\log(h), \log(\epsilon(h))$. We present only the results with stepsize ratio $r = 1$ because this case is representative of the behavior with other values of r. As we can see in the plots, the starting algorithms **S1** give an almost straight line with slope 3 while in the other three cases we get a line with slope 4. These results confirm the stiff order of the algorithms (2 for **S1** and 3 for the others) predicted in the theory.

Now, to check the influence of the stability properties of the algorithms, we have repeated the same process but introducing a perturbation in the initial value by taking $y_0 = \phi(0)(1 + 10^{-3})$ and $J = \partial f/\partial y(t_0, y_0)$, so that the initial error is $\phi(0)10^{-3}$. In figures 4 and 4 we have displayed again the points $(\log(h), \log(\epsilon(h))$. As it can be observed, the starting algorithms **S1**, whose error amplifying factor vanishes at $z = \infty$, gives again lines with slope 3 and the effect of the perturbation is only appreciated for small stepsizes, when the local error of the algorithm is small enough to be comparable to the term due to the propagation of the perturbation.

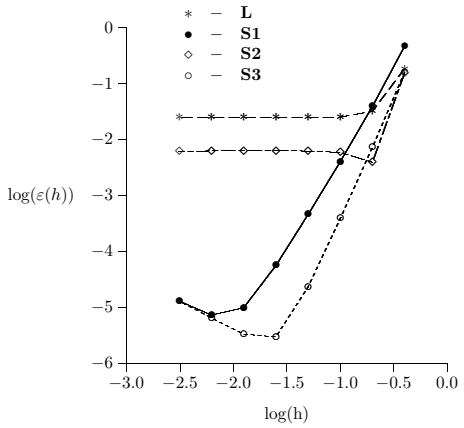


Fig. 8. Problem 1 ($y_0 = u_0 + u_0 * 10^{-3}$)

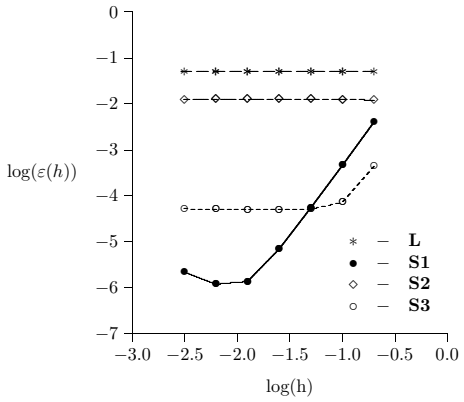


Fig. 9. Problem 2 ($y_0 = u_0 + u_0 * 10^{-3}$)

With respect to the algorithm **S3**, which has the same amplifying function as **S1**, the errors are not practically affected by the perturbation in the problem 1, which is linear, but they seem to be more affected than for **S1** in the case of problem 2. This phenomenon can be due to the nonlinearity of the problem.

Concerning the other two algorithms, that have bounded amplifying functions, the errors are clearly affected by the perturbation and they do not show a practical dependence on the stepsize h, because the dominant term in the expression of the error is the one due to the propagation of the initial error that does not depend on h. Moreover, the absolute value of the amplifying function of the algorithm **L** is greater than the corresponding one of the algorithm **S2** and the errors in this last case are smaller.

Finally, in order to compare the efficiency of the starting algorithms, we have developed a variable stepsize code based on the fifth–order Radau IIA method using local extrapolation technique for controlling the stepsize. To solve the stage equations associated to the method we have used the single–Newton iterative scheme proposed in [3]. With this code, and using the four starting algorithms, we have integrated a large variety of stiff problems. Here we present

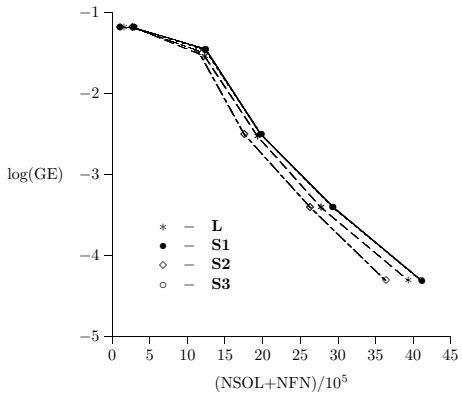


Fig. 10. Problem 3

the results obtained with the following two problems that we have selected because they let us see the possible relevance of the properties of the first iterant used in the code.

Problem 3.- The ring modulator problem (see [6])
Problem 4.- Problem E5 from the stiff DETEST package [2], [5, pp. 145]

In tables 1 and 2 we collect the data corresponding to the integration of problems 3 and 4 respectively. At each column of them we have included the following data:

- RTOL: Parameter for controlling the error tolerance. In this sense a step is accepted in the code when the local error is smaller or equal to RTOL $*$ ($|y_n|+$ATOL), being ATOL=1 for problem 4 and ATOL=0.001 for problem 3.
- SA: Starting Algorithm.
- GE: Global error at the end of the integration interval.
- NR-IT: Number of steps rejected by a fail in the convergence of the iterative scheme.
- NLU: Number of LU matrix factorizations.
- NSOL: Number of linear systems solved.
- NITER: Average number of iterations required per step.

In figures 4 and 4 we present efficiency plots for problems 3 and 4 respectively. In each figure we have represented for the four starting algorithms the logarithm of the global error against NSOL+NFN as a measure of the cost involved in the integration.

Concerning the problem 3, we can see in table 1 as well as in figure 4 that it is integrated with similar global error independently of the starting algorithm taken. The most efficient one is **S2**, which has classical order 4. It is also interesting the fact that the average of iterations per step decreases as the classical order of the starting algorithm increases.

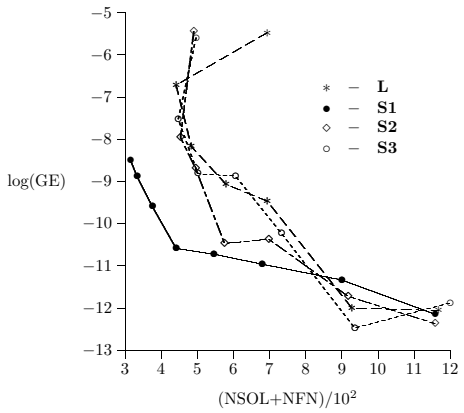


Fig. 11. Problem 4

With respect to problem 4, we must note that for RTOL=0.1 and RTOL=0.01 the code was able to conclude the integration only with **S1** as first iterant. The integration of this problem is very sensitive to the stability properties of the starting algorithm, mainly for large tolerances. This is also clear in figure 4 where we can observe that for large tolerances the algorithm **S1** gives the best efficiency. For small error tolerances, again the order plays a dominant role.

¿From our numerical experiments we can conclude that starting algorithms with good stability properties make the integration more robust and more efficient for some kind of problems when the numerical solution is not required to have very small errors. In general, when a small error is demanded, starting algorithms with good order properties can provide more efficient integrations. Further investigation should be done in order to get some kind of "variable selection of the starting algorithms" so that the code can select the most appropriate first iterant at each step depending on the evolution of the numerical integration.

References

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

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

[3] S. Gonzalez-Pinto, J. I. Montijano and L. Randez, *Iterative schemes for implicit Runge-Kutta methods*, Appl. Numer. Math., 17 (1995), pp. 363-382.

[4] S. Gonzalez-Pinto, J. I. Montijano and S. Pérez-Rodriguez, *On the starting algorithms for fully implicit Runge-Kutta methods*, to appear in BIT (2000).

RTOL	SA	GE	NR-IT	NLU	NSOL	NITER
10^{-2}	**L**	.6551E-01	600	3991	78321	4.11
	S1	.6522E-01	365	2278	51822	4.08
	S2	.6536E-01	350	2189	53734	4.54
	S3	.6514E-01	366	2316	55839	4.55
10^{-3}	**L**	.6514E-01	302	5663	148608	5.40
	S1	.6514E-01	271	5393	148238	5.33
	S2	.6516E-01	238	5134	138249	5.11
	S3	.6515E-01	256	5332	145851	5.29
10^{-4}	**L**	.2890E-01	413	23774	615045	5.20
	S1	.3558E-01	142	22728	632834	5.34
	S2	.3391E-01	121	22560	582861	4.67
	S3	.3449E-01	138	22699	630610	5.32
10^{-5}	**L**	.2950E-02	654	42420	968532	4.66
	S1	.3143E-02	92	40700	1009318	4.82
	S2	.3151E-02	87	40694	898916	3.94
	S3	.3135E-02	83	41022	1012237	4.79
10^{-7}	**L**	.4948E-04	1273	98845	1968534	4.09
	S1	.4859E-04	31	98493	2111732	4.13
	S2	.4885E-04	33	98682	1871762	3.36
	S3	.4905E-04	23	98641	2106752	4.10

Table 1
Problem 3 (Ring modulator)

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

[6] 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.sthtml, Test set for IVP solvers, 1996.

[7] H. Olsson and G. Söderlind, *Stage value predictors and efficient Newton iterations in implicit Runge–Kutta methods*, SIAM Journal on Scientific Computing, 20, 1 (1999), pp. 185–202.

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

RTOL	SA	GE	NACC	NR-IT	NLU	NSOL	NITER
10^{-1}	**L**	***	**	**	**	**	**
	S1	.3192E-08	32	0	32	174	1.00
	S2	***	**	**	**	**	**
	S3	***	**	**	**	**	**
10^{-2}	**L**	***	**	**	**	**	**
	S1	.3192E-08	32	0	32	174	1.00
	S2	***	**	**	**	**	**
	S3	***	**	**	**	**	**
10^{-3}	**L**	.3388E-05	56	4	58	348	1.38
	S1	.1312E-08	32	0	32	183	1.09
	S2	.3699E-05	38	1	39	265	1.42
	S3	.2566E-05	38	1	39	268	1.45
10^{-4}	**L**	.1944E-06	32	0	32	222	1.69
	S1	.2585E-09	32	0	32	204	1.22
	S2	.1121E-07	32	0	32	243	1.63
	S3	.3014E-07	32	0	32	240	1.59
10^{-5}	**L**	.6809E-08	32	0	32	243	1.91
	S1	.2601E-10	32	0	32	237	1.53
	S2	.2023E-08	32	0	32	264	1.81
	S3	.1542E-08	32	0	32	267	1.84
10^{-7}	**L**	.3444E-09	36	0	36	348	2.25
	S1	.1102E-10	36	0	36	358	2.06
	S2	.4328E-10	36	0	36	367	2.14
	S3	.5850E-10	36	0	36	385	2.31
10^{-9}	**L**	.8734E-12	46	0	46	585	2.93
	S1	.7169E-12	46	0	46	602	2.74
	S2	.4297E-12	46	0	46	602	2.74
	S3	.1334E-11	46	0	46	623	2.87

Table 2
Problem 4 (E5)

[9] J. Sand, *Starting methods for the iteration of IRK's*, Dept. of Computer Science, Univ. Copenhagen, Denmark, 1989.

[10] M. N. Spijker, *On the error commited by stopping the Newton iteration in implicit Runge–Kutta methods*, Annals. of Numer. Math., 1, (1994) pp. 199–212.