stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE NW NW 1/73 JANUARY

P.J. VAN DER HOUWEN and H. FIOLET

EXPONENTIAL FITTED RUNGE-KUTTA FORMULAS
OF FOURTH ORDER

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

Contents

1. Introduction 1
2. Six-point formulas of fourth order 3
3. Stability 4
4. Step size control 11
5. An interpolation formula 13
6. The procedure EFFORK 16
7. Numerical examples 24
References 33

1. Introduction

In reference [4] a class of fourth order, m-point Runge-Kutta formulas is described of which the characteristic root contains m - 4 free parameters. These parameters can be used to adjust the stability properties of the formula to the differential equation under consideration. In practice, this implies that in many cases a more efficient integration formula is obtained than the standard fourth order Runge-Kutta method. The present paper gives a more detailed analysis of six-point stabilized fourth order formulas.

In particular, attention is paid to a technique called "exponential fitting" (see Liniger and Willoughby [7]). Since a six-point formula of fourth order has two free parameters it is possible to fit the characteristic root at two points (cf. section 3). The stability regions of exponentially fitted Runge-Kutta formulas were studied in references [5,6]; it was pointed out that vector differential equations of type

$$\frac{du}{dt} = f(t,u) \,, \tag{1.1}$$

of which the Jacobian matrix

$$J = \left(\frac{\partial f_i}{\partial u_j}\right) \tag{1.2}$$

has eigenvalues with widely separated negative real parts (stiff differential equations), can be efficiently integrated by exponentially fitted Runge-Kutta methods. Furthermore, it was shown that the efficiency increases as the number of free parameters is larger. Therefore, we also investigated six-point formulas containing 4 free parameters. It is proved (section 3) that these formulas also are fourth order exact, but the error constant is considerably larger; effectively, the four-parameter forms are only second order correct.

The step size strategy used in our formulas is based on the assumption that an exponentially fitted formula integrates a linear system accurately (cf. [7]). This suggests to choose the integration steps in such a way that the differential equation is sufficiently linear over the successive inte-

gration steps. For that purpose a reference formula was derived which is identical to the actual integration formula in case of linear equations. For non-linear equations the reference formula is only second order accurate. The difference of the results produced by these formulas is taken as an estimate of the non-linearity. By monitoring this estimate an indication of a suitable step is obtained. In addition, we automatically have a (conservative) estimate of the local error, provided that the system is non-linear. The price to be paid for the step size control just described is an additional function evaluation in the reference formula.

Finally, an interpolation formula of third order is derived which can be used when integration steps are chosen, which are larger than the spacing of the reference points prescribed by the user of the integration formula.

In section 6 an ALGOL 60 version of our integration formula is presented; in section 7 a number of numerical examples is given.

2. Six-point formulas of fourth order

Consider the six-point Runge-Kutta formula defined by

$$
\begin{aligned}
u_{k+1}^{(0)} &= u_k, \\
u_{k+1}^{(1)} &= u_k + \tfrac{1}{2}\tau_k f_{k+1}^{(0)}, \\
u_{k+1}^{(2)} &= u_k + \tfrac{1}{2}\tau_k f_{k+1}^{(1)}, \\
u_{k+1}^{(3)} &= u_k + \lambda_{3,1}\tau_k f_{k+1}^{(1)} + \lambda_{3,2}\tau_k f_{k+1}^{(2)}, \\
u_{k+1}^{(4)} &= u_k + \lambda_{4,1}\tau_k f_{k+1}^{(1)} + \lambda_{4,3}\tau_k f_{k+1}^{(3)}, \\
u_{k+1}^{(5)} &= u_k + \tau_k f_{k+1}^{(4)},
\end{aligned}
$$

$$
\begin{aligned}
u_{k+1} &= u_k + \tfrac{1}{6}\tau_k [f_{k+1}^{(0)} + 2f_{k+1}^{(1)} + 2f_{k+1}^{(2)} + f_{k+1}^{(5)}], \\
f_{k+1}^{(j)} &= f(t_{k+1}^{(j)}, u_{k+1}^{(j)}), \\
t_{k+1}^{(0)} &= t_k,\ t_{k+1}^{(1)} = t_{k+1}^{(2)} = t_k + \tfrac{1}{2}\tau_k, \\
t_{k+1}^{(3)} &= t_k + (\lambda_{3,1} + \lambda_{3,2})\tau_k, \\
t_{k+1}^{(4)} &= t_k + (\lambda_{4,1} + \lambda_{4,3})\tau_k, \\
t_{k+1}^{(5)} &= t_k + \tau_k.
\end{aligned}
\tag{2.1}
$$

This formula is second order exact irrespective the values of the parameters $\lambda_{j,l}$. It can be proved (cf. reference [4]) that it is fourth order exact as $\tau_k \to 0$ when the parameters $\lambda_{3,1}$, $\lambda_{3,2}$, $\lambda_{4,1}$, $\lambda_{4,2}$ satisfy the conditions

$$
\begin{aligned}
&\lambda_{4,1} + \lambda_{4,3} = \tfrac{1}{2} + O(\tau_k^2) \text{ as } \tau_k \to 0, \\
&\lambda_{4,1} + 2\lambda_{4,3}(\lambda_{3,1} + \lambda_{3,2}) = \tfrac{1}{2} + O(\tau_k) \text{ as } \tau_k \to 0
\end{aligned}
\tag{2.2}
$$

For future reference we represent scheme (2.1) in the form of the generating matrix (cf. Butcher [1]).

$$(2.1') \qquad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \lambda_{3,1} & \lambda_{3,2} & 0 & 0 & 0 \\ 0 & \lambda_{4,1} & 0 & \lambda_{4,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{6} \end{pmatrix}$$

3. Stability

The characteristic root of scheme (2.1) is given by (cf. [4])

$$(3.1) \qquad R(z) = 1 + z + \frac{1}{2} z^2 + \beta_3 z^3 + \beta_4 z^4 + \beta_5 z^5 + \beta_6 z^6,$$

where

$$(3.2) \qquad \begin{aligned} \beta_3 &= \frac{1}{12} + \frac{1}{6}(\lambda_{4,1} + \lambda_{4,3}), \\ \beta_4 &= \frac{1}{12}[\lambda_{4,1} + 2\lambda_{4,3}(\lambda_{3,1} + \lambda_{3,2})], \\ \beta_5 &= \frac{1}{12}\lambda_{4,3}(\lambda_{3,1} + \lambda_{3,2}), \\ \beta_6 &= \frac{1}{24}\lambda_{3,2}\lambda_{4,3}. \end{aligned}$$

We shall require that the parameters β_j, $j = 3,\ldots,6$ are such that the function $R(z)$ is exponentially fitted at two points $z_1 = \tau_k \delta_1$ and $z_2 = \tau_k \delta_2$, i.e.

$$(3.3) \qquad \begin{aligned} R(z_1) &= e^{z_1}, \quad R'(z_1) = e^{z_1}, \\ R(z_2) &= e^{z_2}, \quad R'(z_2) = e^{z_2}, \end{aligned}$$

or equivalently,

$$(3.3')\qquad \begin{aligned} \beta_3 + \beta_4 z + \beta_5 z^2 + \beta_6 z^3 &= F(z) \\ \beta_4 + 2\beta_5 z + 3\beta_6 z^2 &= F'(z) \end{aligned} \qquad \text{at } z = z_1 , z_2 .$$

where

$$F(z) = \frac{e^z - (1+z+\frac{1}{2}z^2)}{z^3}.$$

When the parameters β_j are solved from conditions (3.3') we can determine the parameters $\lambda_{j,1}$ from (3.2) by expressing the $\lambda_{j,1}$ in terms of β_j, that is

$$(3.2')\qquad \begin{aligned} \lambda_{4,1} &= 12(\beta_4 - 2\beta_5), \\ \lambda_{4,3} &= 6\beta_3 - \frac{1}{2} - \lambda_{4,1}, \\ \lambda_{3,2} &= 24\,\frac{\beta_6}{\lambda_{4,3}}, \\ \lambda_{3,1} &= 12\,\frac{\beta_5 - 2\beta_6}{\lambda_{4,3}} . \end{aligned}$$

The final step then is to show that the parameters $\lambda_{j,1}$ satisfy the consistency conditions (2.2).

In order to solve equations (3.3') we introduce the abbreviations

$$S(z^j) = z_2^j + z_1^j,\ S(F) = F(z_2) + F(z_1),\ldots$$

$$D(z^j) = z_2^j - z_1^j,\ D(F) = F(z_2) - F(z_1),\ldots$$

Equations (3.3') can then be written as

$$\beta_4 D(z) + \beta_5 D(z^2) + \beta_6 D(z^3) = D(F),$$

$$2\beta_4 + 2\beta_5 S(z) + 3\beta_6 S(z^2) = S(F'),$$

$$2\beta_5 D(z) + 3\beta_6 D(z^2) = D(F').$$

A simple calculation leads to the following expressions for the parameters β_j :

$$\beta_6 = \frac{D(z)\, S(F') - 2D(F)}{3D(z)\, S(z^2) - 2D(z^3)},$$

$$\beta_5 = \frac{D(F') - 3\beta_6 D(z^2)}{2D(z)} \tag{3.4}$$

$$\beta_4 = \frac{1}{2} S(F') - \beta_5 S(z) - \frac{3}{2} \beta_6 S(z^2),$$

$$\beta_3 = \frac{1}{2} S(F) - \frac{1}{2} \beta_4 S(z) - \frac{1}{2} \beta_5 S(z^2) - \frac{1}{2} \beta_6 S(z^3).$$

For $z_2 \to 0$ and $z_1 \to 0$ we deduce from these expressions

$$\beta_6 = \frac{1}{720} + O(z_2+z_1),$$

$$\beta_5 = \frac{1}{120} + O((z_2+z_1)^2),$$

$$\beta_4 = \frac{1}{24} + O((z_2+z_1)^3),$$

$$\beta_3 = \frac{1}{6} + O((z_2+z_1)^4).$$

This means that the parameter $\lambda_{j,1}$ behave as

$$\lambda_{4,1} = \frac{3}{10} + O((z_1+z_2)^2),$$

$$\lambda_{4,3} = \frac{1}{5} + O((z_1+z_2)^2),$$

$$\lambda_{3,2} = \frac{1}{6} + O((z_1+z_2)),$$

$$\lambda_{3,1} = \frac{1}{3} + O((z_1+z_2)).$$

Substitution into the consistency conditions shows that we have fourth order accuracy when $\tau_k \to 0$. It should be noted, however, that effectively ($\tau_k \neq 0$) method (2.1), (3.2), (3.3) is only second order exact. Methods which are also effectively fourth order accurate can be obtained by putting

$$\beta_3 = \frac{1}{6}, \quad \beta_4 = \frac{1}{24}$$

for all values of the step size τ_k and by fitting only once at z_1 and z_2. Formulas (3.2') and (3.4) then reduce respectively to

(3.2'')
$$\lambda_{4,1} = \frac{1}{2} - 24\beta_5, \quad \lambda_{4,3} = 24\beta_5,$$
$$\lambda_{3,2} = \frac{\beta_6}{\beta_5}, \quad \lambda_{3,1} = \frac{1}{2} - \frac{\beta_6}{\beta_5}$$

and

(3.4')
$$\beta_6 = \frac{D(F)}{D(z)},$$
$$\beta_5 = \frac{1}{2}\,\frac{D(z)\,S(F) - S(z)\,D(F)}{D(z)}$$

Having derived the coefficients β_j in terms of the fit-points z_1 and z_2, we arrive at the problem to determine the stability regions of the integration method. This problem was considered in [5,6]. It was found that for large values of $|z_1|$ and $|z_2|$ the stability region (defined by the set of points $S = \{z \mid |R(z)| < 1\}$) consists of three subregions situated at the origin and the points z_1 and z_2. For small values of

$|z_1|$ and $|z_2|$ these subregions melt together and become approximately the stability region of the polynomial

$$R(z) = \sum_{j=0}^{6} \frac{1}{j!} z^j.$$

The cases of interest, however, are the larger values of $|z_1|$ and $|z_2|$. We then have (cf. [5]) for the left hand subregions the disks

$$(3.5) \qquad \begin{aligned} &|z-z_j| < \sqrt{2}\ \left|\frac{z_j}{z_2-z_1}\right|, && z_1 \neq z_2 \\ &|z-z_j| < \sqrt[4]{2}\ \sqrt{|z_j|}, && z_1 = z_2 \end{aligned} \qquad , j = 1,2$$

in case of (3.3) and the disks

$$(3.5') \qquad \begin{aligned} &|z-z_j| < 24\ |z_j|^{-3}\ \left|\frac{z_j}{z_2-z_1}\right|, && z_1 \neq z_2 \\ &|z-z_j| < \sqrt{24}\ |z_j|^{-1}, && z_1 = z_2 \end{aligned} \qquad , j = 1,2$$

in case of (3.4'). From this we can easily derive an upper bound for the stepsize τ. For example in case of (3.3) we find

$$\begin{aligned} &\tau < \sqrt{2}\ \frac{\delta_j}{\rho_j |\delta_2-\delta_1|}, && \delta_1 \neq \delta_2 \\ &\tau < \sqrt{2}\ \frac{\delta_j}{\rho_j^2}, && \delta_1 = \delta_2 \end{aligned} \qquad , j = 1,2,$$

where δ_j is the center and ρ_j the radius of the cluster near δ_j. The right hand subregions resemble respectively the stability regions of the polynomials

$$(3.6) \qquad 1 + z + \frac{1}{2} z^2$$

and

$$(3.6') \qquad 1 + z + \frac{1}{2} z^2 + \frac{1}{6} z^3 + \frac{1}{24} z^4.$$

It can be proved that the following right hand stability conditions hold:

$$(3.7) \qquad \tau < \frac{2}{\delta_0+\rho_0} \quad , p = 2,$$

$$\tau < \frac{2.63}{\delta_0+\rho_0} \quad , p = 4,$$

where the eigenvalues close to the origin are supposed to be in the negative interval $[-\delta_0-\rho_0, 0]$ when $p = 2$ and in the disk $|-\delta_0+\delta| \leq \rho_0$ when $p = 4$.

In an actual computation it is important that the parameters $\lambda_{j,l}$, and therefore the coefficients β_j, are calculated with high accuracy, in particular when $|z_1|$ and $|z_2|$ have large values. Hence, we shall derive asymptotic expressions for the coefficients β_j which holds for $|z_1| \to \infty$ and $|z_2| \to \infty$. Let us write $R(z)$ in the approximate form

$$R(z) = \frac{1}{z_1^2 z_2^2} (z-z_1)^2 (z-z_2)^2 (1+az+bz^2),$$

where a and b are determined by the condition $R'(0) = R^{IV}(0) = 1$ (cf. (3.1)).

By working out the right hand side we can easily find the coefficients β_j. Straightforward calculation yields

$$R(z) = 1 + \frac{az_1z_2 - 2(z_1+z_2)}{z_1z_2} z$$

$$+ \frac{(z_1+z_2)^2+2z_1z_2-2az_1z_2(z_1+z_2)+bz_1^2 z_2^2}{z_1^2 z_2^2} z^2$$

$$+\frac{a(z_1+z_2)^2 + 2az_1z_2 - 2z_1z_2(z_1z_2)b - 2(z_1+z_2)}{z_1^2\, z_2^2}\, z^3$$

$$+\frac{1-2a(z_1+z_2) + (z_1+z_2)^2 b + 2z_1z_2 b}{z_1^2\, z_2^2}\, z^4$$

$$+\frac{a-2(z_1+z_2)b}{z_1^2\, z_2^2}\, z^5$$

$$+\frac{b}{z_1^2\, z_2^2}\, z^6.$$

Identification with (3.1) leads to the following expressions for β_j, $j > 2$:

$$\beta_6 = \varepsilon_2^2\,[\tfrac{1}{2} - 2\varepsilon_1 + (3\varepsilon_1^2 - 2\varepsilon_2)],$$

$$\beta_5 = \varepsilon_1\,\varepsilon_2\,[1 - 4\varepsilon_1 + (6\varepsilon_1^2 - 2\varepsilon_2)] + \varepsilon_2^2(1-4\varepsilon_1)$$

$$\beta_4 = (\varepsilon_2+\tfrac{1}{2}\varepsilon_1^2) - 2\varepsilon_1(\varepsilon_1^2+\varepsilon_2) + 3\varepsilon_1^4 - 3\varepsilon_2^2$$

$$\beta_3 = \varepsilon_1 + (2\varepsilon_2-3\varepsilon_1^2) + 2\varepsilon_1(2\varepsilon_1^2-3\varepsilon_2).$$

Here, the parameters ε_1 and ε_2 are defined by

$$\varepsilon_1 = -\frac{z_1+z_2}{z_1z_2}\,,\quad \varepsilon_2 = \frac{1}{z_1z_2}\,.$$

For the parameters $\lambda_{j,1}$ we finally have

$$\lambda_{4,1} = 12[(\varepsilon_2+\tfrac{1}{2}\,\varepsilon_1^2) - 2\varepsilon_1(\varepsilon_1^2+ 2\varepsilon_2)$$

$$+ \varepsilon_1^2(3\varepsilon_1^2+8\varepsilon_2) - 12\varepsilon_1^3\varepsilon_2 + 12\varepsilon_1\varepsilon_2^2 - 5\varepsilon_2^2],$$

$$\lambda_{4,3} = -\frac{1}{2} + 6\varepsilon_1 - 24\varepsilon_1^2 + 12\varepsilon_1\varepsilon_2 +$$

$$- 48\,\varepsilon_1^2\,\varepsilon_2\,[2-3\varepsilon_1] - 36\varepsilon_1^4$$

$$+ 48\varepsilon_1^3 - 144\varepsilon_1\varepsilon_2^2 + 60\varepsilon_2^2\,,$$

$$\lambda_{3,2} = 24\,\frac{\beta_6}{\lambda_{4,3}}\,,$$

$$\lambda_{3,1} = 12\,\frac{\varepsilon_2}{\lambda_{4,3}}\,[\varepsilon_1 - 2\varepsilon_1\varepsilon_2 + \varepsilon_1^2(-4+6\varepsilon_1-6\varepsilon_2) + 4\varepsilon_2^2].$$

Similar expressions can be derived in case of formula (3.4'). Finally, we remark that in the case (3.4') the special fit-points

$$z_1 = -7.59521,\ z_2 = -9.70395$$

generate the stability polynomial

$$R(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4 + .005303430 * z^5 +$$
$$+ .0002404730 * z^6\,.$$

This polynomial has a real stability boundary

$$\beta_{real} = 9.97,$$

which is, in fact, the largest value obtainable by fourth order polynomials of degree 6(cf. reference [3]).

4. Step size control

The stability considerations given in the preceding section are local

considerations, that is they are based on a linear approximation of the differential equation in a neighbourhood of (t_k, u_k). Thus, the differential equation should be sufficiently linear over the integration step τ_k actually used. By choosing τ_k sufficiently small, this condition can always be satisfied. Hence, we are faced with the problem how small should τ_k be chosen. We need the following strategy: let $\tilde{u}_{k+1}$ be a reference solution which is identical to u_{k+1} as soon as the differential equation under consideration is linear; then τ_k should be such that

$$(4.1) \qquad ||u_{k+1}-\tilde{u}_{k+1}|| = \eta_k = \eta_a + \eta_r||u_k|| ,$$

where η_a and η_r are given absolute and relative tolerances, respectively. Furthermore, let $\tilde{u}_{k+1}$ be of order $\tilde{p}$. Then

$$(4.2) \qquad ||u_{k+1}-\tilde{u}_{k+1}|| = \phi(t_k,\tau_k)\ \tau_k^{\tilde{p}+1} ,$$

provided that $\tilde{p} \le p$. The error function $\phi(t,\tau)$ generally is a slowly varying function of t and τ, so that

$$(4.3) \qquad \phi(t_k,\tau_k) \cong \phi(t_{k-1},\tau_{k-1}) = \frac{||u_k-\tilde{u}_k||}{\tau_{k-1}^{p+1}} .$$

From (4.1) - (4.3) it then follows that

$$(4.4) \qquad \tau_k \cong \left[\frac{\eta_a+\eta_r||u_k||}{||u_k-\tilde{u}_k||}\right]^{\frac{1}{\tilde{p}+1}} \tau_{k-1}$$

Next we consider the construction of the reference solution $\tilde{u}_{k+1}$. We try to satisfy the requirements imposed on $\tilde{u}_{k+1}$ for the class of formulas generated by the parameter matrix

$$
(4.5)\qquad
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & \lambda_{3,1} & \lambda_{3,2} & 0 & 0 & 0 \\
0 & \lambda_{4,1} & 0 & \lambda_{4,3} & 0 & 0 \\
0 & 0 & 0 & 0 & \lambda_{5,4} & 0 \\
\frac{1}{3} - \frac{1}{6\lambda_{5,4}} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{6\lambda_{5,4}}
\end{pmatrix}
$$

Here, $\lambda_{3,1}$, $\lambda_{3,2}$, $\lambda_{4,1}$ and $\lambda_{4,3}$ are identical to the parameters used in the calculation of u_{k+1}. The parameter $\lambda_{5,4}$ is a free parameter $\neq 1$. Hence, an additional point is introduced for the computation of $\tilde{u}_{k+1}$. It is easily verified that (4.5) generates Runge-Kutta formulae of which the characteristic root is identical with R(z), defined by (3.1), (3.2). From this it is immediately clear that $\tilde{u}_{k+1} = u_{k+1}$ in case of linear differential equations. Furthermore, this implies that $\tilde{u}_{k+1}$ is second order correct (i.e. $\tilde{p} = 2$) in case of non-linear equations. In our experiments we have chosen

$$
(4.6)\qquad \lambda_{5,4} = \frac{1}{2} .
$$

The step size τ_k can now be predicted by formula (4.4) with $\tilde{p} = 2$. In actual computation, however, we used a rational approximation of (4.4), namely

$$
(4.4')\qquad \tau_k = \left[\frac{5}{3} - \frac{4}{3\left(1 + \frac{\eta_a + \eta_r ||u_k||}{||u_k - \tilde{u}_k||}\right)}\right] \tau_{k-1}
$$

For $||\ ||$ the Euclidean norm was chosen.

5. An interpolation formula

Suppose that the solution of a differential equation is required in

given reference points ξ_v, $v = 1,2,\ldots,N$. When formula (2.1) is used this problem requires at least $6N$ function evaluations. However, if the intervals $\xi_v - \xi_{v-1}$ are small it may happen that the accuracy of the results is much larger than required, so that relatively much computing time is spend to the problem. For instance, when it turns out that an integration step as large as $\xi_N - \xi_0 = \xi_N - t_0$ yields sufficiently accurate results at $t = \xi_N$, one may ask whether it is possible to interpolate at the points $t = \xi_v$, $v < N$. We have tried to find an interpolation formula generated by the parameter matrix

$$(5.1) \qquad M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \lambda_{3,1} & \lambda_{3,2} & 0 & 0 & 0 \\ 0 & \lambda_{4,1} & 0 & \lambda_{4,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \theta_0 & \theta_1 & \theta_2 & 0 & 0 & \theta_5 \end{pmatrix},$$

where θ_0, θ_1, θ_2 and θ_5 are parameters which will be chosen such that the result of applying M with step τ_k is a reasonable approximation to the solution at a point $t = t_k + \tau$, $0 < \tau < \tau_k$. This approximation will be denoted by $u_k + \frac{\tau}{\tau_k}$. Clearly, the parameters θ_j will appear to be functions of τ. When we succeed we have obtained an interpolation formula which does not require additional function evaluations.

Our starting point in finding a reasonable accurate value for $u_k + \frac{\tau}{\tau_k}$, is the observation that applying M with step τ_k is equivalent with applying $\frac{\tau_k}{\tau}$ M with step τ. This implies that we simply have to satisfy as many consistency conditions of $\frac{\tau_k}{\tau}$ M with step τ as possible. In doing so we find the following set of conditons:

$$
(5.2)\quad
\begin{aligned}
&p \geq 1 && \frac{\tau_k}{\tau}\left[\theta_0 + \theta_1 + \theta_2 + \theta_5\right] = 1 ,\\
&p \geq 2 && \frac{\tau_k^2}{\tau^2}\left[\frac{1}{2}\theta_1 + \frac{1}{2}\theta_2 + \theta_5\right] = \frac{1}{2} ,\\
& && \frac{\tau_k^3}{\tau^3}\left[\frac{1}{4}\theta_1 + \frac{1}{4}\theta_2 + \theta_5\right] = \frac{1}{3}\\
&p \geq 3 && \hspace{10em} ,\\
& && \frac{\tau_k^3}{\tau^3}\left[\frac{1}{4}\theta_2 + (\lambda_{4,1} + \lambda_{4,3})\,\theta_5\right] = \frac{1}{6}
\end{aligned}
$$

Since only 4 parameters θ_j are available at most third order accuracy can be obtained for $u_{k + \frac{\tau}{\tau_k}}$ by solving these four consistency conditions.

However, when equations (5.2) are actually solved, it turns out that $u_{k + \frac{\tau}{\tau_k}} \neq u_{k+1}$ for $\tau = \tau_k$, unless the parameter β_3 defined by (3.2) equals 1/6. This is easily explained by observing that u_{k+1} is fourth order exact as $\tau_k \to 0$; for $\tau_k \neq 0$ it is effectively second order exact, unless $\beta_3 = 1/6$ and $\beta_4 = 1/24$. In order to make $u_{k + \frac{\tau}{\tau_k}}$ equal to u_{k+1} as $\tau \to \tau_k$ we replace in the last equation of (5.2) the righthand side by β_3. The corresponding interpolation function $u_{k + \frac{\tau}{\tau_k}}$ then is also effectively second order correct for $\tau_k \neq 0$ and third order correct as $\tau_k \to 0$.
A simple calculation yields

$$
\begin{aligned}
\theta_5 &= -\frac{1}{2\tau_k^2}\tau^2 + \frac{2}{3\tau_k^3}\tau^3 ,\\
\theta_2 &= \frac{4\beta_3}{\tau_k^3}\tau^3 - 4(\lambda_{4,1} + \lambda_{4,3})\,\theta_5\\
&= \frac{1}{3\tau_k^3}\tau^3 + \left(\frac{2}{3\tau_k^3}\tau^3 - 4\theta_5\right)(\lambda_{4,1} + \lambda_{4,3}) .
\end{aligned}
$$

$$\theta_1 = \frac{4\tau^3}{3\tau_k^3} - \theta_2 - 4\theta_5,$$

$$\theta_0 = \frac{\tau}{\tau_k} - \theta_1 - \theta_2 - \theta_5.$$

Substitution into (5.1) leads to a third degree polynomial for $u_k + \frac{\tau}{\tau_k}$:

$$(5.3) \qquad u_k + \frac{\tau}{\tau_k} = u_k + f_{k+1}^{(0)}\tau + \tau_k^{-1}\left[-\frac{3}{2} f_{k+1}^{(0)} + 2(1-\lambda_{4,1}-\lambda_{4,3}) f_{k+1}^{(1)} + \right.$$

$$\left. + 2(\lambda_{4,1}+\lambda_{4,3}) f_{k+1}^{(2)} - \frac{1}{2} f_{k+1}^{(5)}\right]\tau^2 +$$

$$+ \tau_k^{-2}\left[\frac{2}{3} f_{k+1}^{(0)} + (2\lambda_{4,1}+2\lambda_{4,3}-\frac{5}{3}) f_{k+1}^{(1)} + \right.$$

$$\left. + (\frac{1}{3} - 2\lambda_{4,1}-2\lambda_{4,3}) f_{k+1}^{(2)} + \frac{2}{3} f_{k+1}^{(5)}\right]\tau^3.$$

In the ALGOL 60 implementation of scheme (2.1) the coefficients of this polynomial are automatically computed and stored in an array I, where the j-th row of I contains the j-th component of the coefficients u_k, $f_{k+1}^{(0)}, \ldots$. Thus in terms of I formula (5.3) reads

$$(5.3') \qquad u_k + \frac{\tau}{\tau_k} = I \begin{pmatrix} \tau^0 \\ \tau^1 \\ \tau^2 \\ \tau^3 \end{pmatrix}$$

6. The procedure EFFORK

In this section we describe an ALGOL 60 version of the integration process discussed in the preceding sections.

The heading of the procedure EFFORK (Exponentially Fitted Fourth Order Runge Kutta method) reads as follows:

```
procedure EFFORK (t, te, m0, m, u, derivative, output, k, phi, sigma0
                  sigma1, sigma2, ro0, ro1, ro2, p, eta, aeta, reta,
                  hmin, hmax, I, fillI);
integer mo, m, k, p;
real    t, te, phi, sigma0, sigma1, sigma2, ro0, ro1, ro2, eta, aeta,
        reta, hmin, hmax;
array   u, I;
procedure derivative, output;
boolean fillI;
```

The actual parameters corresponding to the formal parameters are:

t : <variable> ;
t is used as Jensen parameter;
entry: the initial value t_0 ;

te : <expression> ;
the end value of t;

m0,m : <expression> ;
indices of the first and last equation to be solved;

u : <array identifier> ;
a one-dimensional array u [m0 : m] ;
entry: the initial values of the solution u(t);

derivative: <procedure identifier> ;
a procedure to be declared by the user:
procedure derivative (t,v); real t; array v;
<body> ;
upon completion of a call of derivative array v should contain
the components of f(t,v);

output : <procedure identifier> ;
a procedure to be declared by the user: procedure output;
<body >; by this procedure one may order to print the values of
t, u[m0], ..., u[m], etc;

k : <variable> ;

counts the integration steps;
entry: it is required that k = 0;

phi : <expression> ;
the argument ϕ of the point in the complex plane where exponential fitting is desired;
phi = arg (z_1) = 2π - arg (z_2);

sigma0 : <expression> ;
the modulus of the center of the cluster near the origin;
sigma0 = $|\delta_0|$;

sigma1 , sigma2 : <expression> ;
moduli of the (complex) points where exponential fitting is desired;
sigma1 = $|\delta_1|$, sigma2 = $|\delta_2|$; if $\phi \neq \pi$ then it is required that sigma1 = sigma2 ;

ro0 , ro1 , ro2 : <expression> ;
radii of the clusters corresponding to sigma0 , sigma1 and sigma2 ;
ro0 = ρ_0, ro1 = ρ_1 and ro2 = ρ_2;

p : <expression> ;
determines the effective order of the scheme; the alternatives are p = 2 or p = 4 corresponding to (3.2') and (3.2"), respectively;

eta : <variable> ;
the tolerance η_k which is a function of aeta and reta (formula (4.1));

aeta, reta: <expression> ;
absolute and relative tolerance ;

hmin, hmax: <expression >;
minimal respectively maximal steplength by which the integration is performed;

I : <array identifier>; array I [m0 : m, 0 : 3]; in this array information is stored, to be used in the interpolation formula (formulas (5.3), (5.3');

```
fillI       : <Boolean expression> ;
              if fillI  = false then the statements concerning array I are
              skipped;
```

Next the complete ALGOL 60 text is presented:

```
procedure EFFORK(t,te,m0,m,u,derivative,output,k,
                         phi,sigma0,sigma1,sigma2,ro0,ro1,ro2,
                         p,eta,aeta,reta,hmin,hmax,I,fillI);
integer m0,m,k,p;
real t,te,sigma0,sigma1,sigma2,phi,ro0,ro1,ro2,
       eta,aeta,reta,hmin,hmax;
array u,I;
procedure derivative,output;
boolean fillI;
begin real tau,z1,z2,z01,z02,phi0,pi,d2md1,c,c1,tau0;
    integer i,j;
    boolean real,change,righthalfplane,first;
    array mu[0:5],labda[-2:5],beta[3:6],
          r,r1,u0,u1,s[m0:m],thetha[0:3],e[0:5,1:3];

    procedure forme;
    begin real t1,t2;
        e[0,1]:=1;t1:=1/tau;t2:=t1×t1;
        e[0,2]:=-1.5×t1;e[0,3]:=e[5,3]:=2×t2/3;e[1,2]:=2×t1×(1-mu[4]);
        e[1,3]:=t2×(2×mu[4]-5/3);e[2,2]:=2×t1×mu[4];
        e[2,3]:=t2×(1/3-2×mu[4]);e[5,2]:=-.5×t1
    end forme;

    procedure coefficient2;
    begin z01:=z1;z02:=z2;phi0:=phi;
        if righthalfplane then z1:=z2:=0;
        if abs(z1)>50∧abs(z2)>50 then
        begin real a,b,a2,b2,ab;
            b:=1/(z1×z2);a:=if real then (z1+z2)×b else -2×z1×cos(phi)×b;
            a2:=a×a;b2:=b×b;ab:=a×b;
            labda[-1]:=12×(b+a2×(.5-a×(2-3×a+12×b)+8×b)+
                       4×ab×(3×b-1)-5×b2);
            labda[4]:=-.5+6×a+12×ab+a2×(-24-96×b+144×ab-36×a2+48×a)+
                       b2×(60-144×a);
            labda[3]:=24×(b2×(.5-2×a+3×a2-2×b))/labda[4];
            labda[-2]:=12×b×(a-2×ab+a2×(-4+6×a-6×b)+4×b2)/labda[4];
            goto mu34
        end;
        if abs(z2-z1)>.1 then
        begin real array a[1:4,1:4],f[1:4];
            real z;integer j;
            procedure init(i,z);integer i;real z;
            begin real z2,z3;
                z:=-z;z2:=z×z;z3:=z2×z;
                if abs(z)<10-3 then
                begin f[i]:=1/6+z/24+z2/120;f[i+1]:=1/24+z/60+z2/240 end
                                else
```

```
            begin f[i]:=(exp(z)-(1+z+z2/2))/z3;f[i+1]:=f[i]-(3×f[i]-.5)/2 end;
            a[i,1]:=a[i+1,2]:=1;a[i+1,1]:=0;a[i,2]:=z;a[i+1,3]:=2×z;
            a[i,3]:=z2;a[i,4]:=z3;a[i+1,4]:=3×z2
        end;
        init(1,z1);init(3,z2);
        detsol(a,4,f);
        for j:=1 step 1 until 4 do beta[j+2]:=f[j]
    end else
    if real∨z1<10-3 then
    begin real z,z2,f1,f2,f3,f4;
        z:=-z1;z2:=z×z;
        if abs(z)<10-3 then
        begin f1:=1/6+z/24+z2/120;f2:=1/24+z/60+z2/240;
              f3:=1/60+z/120+z2/420;f4:=1/120+z/210+z2/672
        end else
        begin real expz;expz:=exp(z);
            f1:=(expz-(1+z+z2/2))/(z2×z);
            f2:=f1-(3×f1-.5)/z;
            f3:=f1+(1-6×f1)/z2+(.5-6×f2)/z;
            f4:=(expz-6×f1-18×z×f2-9×z2×f3)/(z2×z)
        end;
        beta[6]:=f4/6;
        beta[5]:=(f3-6×beta[6]×z)/2;
        beta[4]:=f2-3×beta[6]×z2-2×beta[5]×z;
        beta[3]:=f1-beta[6]×z2×z-beta[5]×z2-beta[4]×z
    end else
    begin real array a[1:4,1:4],f[1:4];
        real r,i,z2,expr,r2,i2,rt,it,rn,in,n;
        integer j;
        r:=z1×cos(phi);i:=z1×sin(phi);z2:=z1×z1;
        expr:=exp(r);r2:=r×r;i2:=i×i;
        rt:=expr×cos(i)-(1+r+r2/2-i2/2);
        it:=expr×sin(i)-(i+r×i);
        rn:=r×(r2-3×i2);in:=i×(3×r2-i2);
        n:=rn×rn+in×in;
        f[1]:=(rt×rn+it×in)/n;f[2]:=(it×rn-rt×in)/n;
        f[3]:=f[1]-3×(r×f[1]+i×f[2]-r/6)/z2;
        f[4]:=f[2]-3×(r×f[2]-i×f[1]+i/6)/z2;
        a[2,1]:=a[3,1]:=a[4,1]:=a[4,2]:=0;a[1,1]:=a[3,2]:=1;
        a[1,2]:=r;a[2,2]:=i;a[3,3]:=2×r;a[4,3]:=2×i;
        a[1,3]:=r2-i2;a[3,4]:=3×a[1,3];a[2,3]:=2×r×i;
        a[4,4]:=3×a[2,3];a[1,4]:=rn;a[2,4]:=in;
        detsol(a,4,f);
        for j:=1 step 1 until 4 do beta[j+2]:=f[j]
    end;
    labda[-1]:=12×(beta[4]-2×beta[5]);
    labda[4]:=6×beta[3]-.5-labda[-1];
    labda[3]:=24×beta[6]/labda[4];
    labda[-2]:=12×(beta[5]-2×beta[6])/labda[4];
mu34:mu[3]:=labda[3]+labda[-2];mu[4]:=labda[4]+labda[-1]
end coefficient2;
```

```
procedure coefficient4;
begin real g1,g2,a,b;
    z01:=z1;z02:=z2;phi0:=phi;
    a:=if real then -(z1+z2) else 2×z1×cos(phi);
    if righthalfplane then
    begin beta[5]:=1/120;beta[6]:=1/720 end else
    if abs(z1)<5_10-2 ∧ abs(z2)<5_10-2 then
    begin beta[5]:=1/120 - z1×z2/5040;beta[6]:=1/720+a/5040 end else
    if abs(z1)>_10 4 ∧ abs(z2)>_10 4 then
    begin beta[6]:=1/(24×z1×z2);beta[5]:=-a×beta[6] end else
    if real then
    begin z1:=-z1;z2:=-z2;
        if abs(z1-z2)<.1 then
        begin real z,z5,z6,expz;
            z:=z1;expz:=exp(z);z6:=1/z↑6;z5:= z6×z;
            beta[5]:=expz×z5×(6-z)-z5×(6+z×(5+z×(2+z×(.5+z/12))));
            beta[6]:=expz×z6×(z-5)+z6×(5+z×(4+z×(1.5+z×(1/3+z/24))))
        end else
        begin
            g1:=if abs(z1)<5_10-2 then 1/120+z1/720+z1×z1/5040 else
                if abs(z1)>_10 4 then -(z1+4)/(24×z1×z1) else
                1/(z1↑5)×(exp(z1)-(1+z1×(1+z1×(.5+z1×(1/6+z1/24)))));
            g2:=if abs(z2)<5_10-2 then 1/120+z2/720+z2×z2/5040 else
                if abs(z2)>_10 4 then -(z2+4)/(24×z2×z2) else
                1/(z2↑5)×(exp(z2)-(1+z2×(1+z2×(.5+z2×(1/6+z2/24)))));
            beta[5]:=(z2×g1-z1×g2)/(z2-z1);beta[6]:=(g2-g1)/(z2-z1)
        end
    end else
    begin real expre,expim,a2,a4,b2,b4,a2b2,ret,ren,imt,imn,d;
        a:=a/2;b:=d2md1×tau/2;
        expim:=exp(a);
        expre:=expim×cos(b);expim:=expim×sin(b);
        a2:= a × a; a4:=a2×a2;b2:=b×b;b4:=b2×b2;a2b2:=20×a2×b2;
        ren:=b2×(10×a4-a2b2+2×b4);
        imn:=a×b×(-2 ×a4+a2b2-10×b4);
        ret:=expre-1-a-(a2-b2-a×b2)/2-a2×a/6-(a4-.3×a2b2+b4)/24;
        imt:=expim-b×(1+a+a2/3+(a+1)×(a2-b2)/6);
        d:=1/(ren×ren+imn×imn);
        beta[6]:=-2×d×(ren×ret+imt×imn);
        beta[5]:=-a×beta[6]+2×b×d×(ren×imt-ret×imn)
    end;
    labda[3]:= beta[6]/beta[5];
    labda[ 4]:=24×beta[5];
    labda[-2]:=.5-labda[3];
    labda[-1]:=.5-labda[4]
end coefficient4;
```

```
procedure stepsize;
begin real d,d1,d2,s1,s2;
    real:=abs(phi-pi)<.01;
    righthalfplane:=(phi<pi×.5∨phi>pi×1.5);
    d1:=sigma1;d2:=sigma2;
    if hmin=hmax then begin first:=true;tau:=hmin end else
    if first then begin first:=false;tau:=tau0 end else
    begin real taustab,tau1;
        tau:=hmax;
        if real∧abs(d1-d2)<.1 then taustab:=if p=2 then
                              c×d1/(ro1×ro1) else c/sqrt(d1×ro1)
                              else
        begin d2md1:=if real then abs(d2-d1) else abs(2×d1×sin(phi));
              if p=2 then begin s1:=abs(c×d2/(ro1×d2md1));
                          s2:=abs(c×d1/(ro2×d2md1))
                     end else
            begin d:=d1×d2/d2md1;s1:=abs(c×(d/ro1)↑.25/d1);
                  s2:=abs(c×(d/ro2)↑.25/d2)
             end;
            taustab:=if s1<s2 then s1 else s2
        end;
        d:=abs(c1/(sigma0+ro0));
        if taustab>d then taustab:=d;
        if tau>taustab then tau:=taustab;
        for j:=m0 step 1 until m do u1[j]:=u1[j]-u[j];
        eta:=aeta+reta×sqrt(vecvec(m0,m,0,u,u));
        tau1:=tau0×(1/3+eta/(.75×(eta+sqrt(vecvec(m0,m,0,u1,u1)))));
        if tau1<tau then tau:=tau1;
        if righthalfplane then tau:=hmax;
        if tau<hmin then tau:=hmin;
    end;
    tau0:=tau;
    if t+tau>te then tau:=te-t;
    if tau<abs(t×10-12) then goto out;
    z1:=tau×d1;z2:=tau×d2;
    s1:=.1×ro1×tau;
    if real then begin s2:=.1×ro2×tau;
                       change:=k=0∨(abs(z01-z1)>s1∨abs(z02-z2)>s2)
                 end else
                 begin d:=s1×s1;
                       change:=k=0∨((z1-z01)×(z1-z01)+
                               z1×z01×(phi-phi0)×(phi-phi0)>d)
                 end
end stepsize;

procedure difference scheme;
begin real mt,lt,ltau;
    i:=-1;
    for j:=m0 step 1 until m do u0[j]:=u1[j]:=r[j]:=u[j];
    if fillI then for j:=m0 step 1 until m do I[j,0]:=u[j];
nextterm:
    mt:=mu[i+1]×tau;lt:=labda[i+1]×tau;
    if i=2∨i=3 then
    begin ltau:=labda[i-4]×tau;
          for j:=m0 step 1 until m do r[j]:=u0[j]+lt×r[j]+ltau×r1[j]
    end else
```

```
        if i>-1 then for j:=m0 step 1 until m do r[j]:=u0[j]+ lt×r[j];
        i:=i+1;
        if i=5∧hmax≠hmin then
        begin real lt1,mt1;lt1:=mt1:=tau/2;
            for j:=m0 step 1 until m do s[j]:=u0[j]+lt1×r[j];
            derivative(t+mt1,s)
        end;
        derivative(t+mt,r);
        if i=1 then for j:=m0 step 1 until m do r1[j]:=r[j];
        if fillI then
        begin integer k;
            if i=0 then for j:=m0 step 1 until m do
            for k:=1,2,3 do I[j,k]:=e[i,k]×r[j];
            if i=1∨i=2∨i=5 then
            for j:=m0 step 1 until m do
            for k:=2,3 do I[j,k]:=I[j,k]+e[i,k]×r[j]
        end;
        if i=0∨i=1∨i=2∨i=5 then
        begin real tht;
            tht:=if i=5 then tau×thetha[3] else tau×thetha[i];
            for j:=m0 step 1 until m do u[j]:=u[j]+ tht×r[j];
            if hmin≠hmax then
            begin if i=5 then
                begin tht:=tau/3;
                      for j:=m0 step 1 until m do u1[j]:=u1[j]+tht×s[j]
                end else
                if i=1∨i=2 then
                for j:=m0 step 1 until m do u1[j]:=u1[j]+tht×r[j]
            end
        end;
        if i<5 then goto nextterm;
        t:=t+tau
    end difference scheme;

    pi:=4×arctan(1);
    if p=2 then begin c:=sqrt(2);c1:=2 end
                        else begin c:=2↑(-.25);c1:=2.63 end;
    tau0:=hmin;first:=true;
    mu[0]:=0;mu[1]:=mu[2]:=mu[3]:=mu[4]:=.5;mu[5]:=1;
    thetha[0]:=thetha[3]:=1/6;thetha[1]:=thetha[2]:=1/3;
    labda[0]:=0;labda[1]:=labda[2]:=.5;labda[5]:=1;

    next level:
    stepsize;
    if change then
    begin if p=2 then coefficient2 else coefficient4 end;
    if fillI then forme;
    k:=k+1;
    difference scheme;
    output;
    if t<te then goto next level;
out:

end runge kutta orde 4;
```

Finally, an outline is given of the subprocedures occurring in procedure EFFORK.

procedure forme

This procedure is used for the construction of array I, concerning the interpolation formula.

procedure coefficient 2

If the parameter p has the actual value 2 this procedure is used for the calculation of $\lambda_{3,1}$, $\lambda_{3,2}$, $\lambda_{4,1}$, $\lambda_{4,3}$;

procedure coefficient 4

If p = 4 this procedure calculates $\lambda_{3,1}$, $\lambda_{3,2}$, $\lambda_{4,1}$, $\lambda_{4,3}$.

procedure stepsize

The determination of the step τ_k is based both on the stepsize control described in section 4 and the stability regions (3.5), (3.5'), (3.6), (3.6').

Also in stepsize the variations of the complex points $z_l = \tau * \delta_l$ is considered. The coefficients are newly computed when $|dz_l| > .1 * \tau * \rho_l$.

procedure difference scheme

By this procedure the values of u[j], representing the components of the numerical solution $u(t_k)$, are replaced by the components of an approximation to $u(t_k+\tau_k)$.

7. Numerical examples

In this section results are presented of procedure EFFORK when applied to a number of stiff differential equations.

Two coupled equations

Consider the following initial value problem (cf. Fowler and Warten [2])

$$(7.1) \qquad \dot{U} = DU + F,\ U(0) = U_0 ,$$

where $D = \begin{pmatrix} -500.5 & 499.5 \\ 499.5 & -500.5 \end{pmatrix}$, $F = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, $U_0 = \begin{pmatrix} -.1 \\ .1 \end{pmatrix}$.

The matrix D has the eigenvalues $\delta_1 = -1000$ and $\delta_r = -1$. The analytical solution of (7.1) is given by

$$U = 2(1-e^{-t}) \begin{pmatrix} 1 \\ 1 \end{pmatrix} + .1 * e^{-1000t} \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

A uniform steplength was used for the integration from t = 0 to t = 10. In table 7.1 we give $-^{10}\log \varepsilon$, where $\varepsilon = \max_{k=1,2} |u_k(t) - \tilde{u}_k(t)|$, for some values of the stepsize τ.

Parameters used

phi = π, sigma1 = sigma2 = 100, hmin = hmax = τ.

Since the integration was performed with a constant steplength, the choice of the parameters sigma0, ro0, ro1, ro2, eta, reta is irrelevant.

Table 7.1 Numerical results for problem (7.1)

effective order	t	stepsize τ					
		1	.5	.2	.1	.05	.02
2	1	.7	1.5	2.4	3.0	3.7	4.7
2	10	3.0	4.4	5.5	6.1	6.8	7.8
4	1	1.7	3.3	5.1	6.3	7.6	9.3
4	10	5.0	6.4	8.1	9.0	9.6	12.0

The single equation $\dot{U} = -e^t U + e^t \ln t + 1/t$

We consider the initial value problem:

$$(7.2) \quad \begin{cases} \dot{U} = -e^t U + e^t \ln t + 1/t, \\ U(.01) = \ln(.01). \end{cases}$$

This problem has the solution $\tilde{U}(t) = \ln(t)$. Since the Jacobian matrix behaves as $-\exp(t)$, the differential equation becomes increasingly stiff for $t > 3$. This suggests the use of a variable steplength. The fourth order exact scheme gives rise to the stability condition (compare (3.5') with $z_1 = z_2$)

$$(7.3) \qquad \tau_k \leq \frac{24^{1/4}}{(\delta * \rho)^{1/2}} ,$$

where

$\delta = e^t$ and

ρ = radius of the cluster.

Instead of $\rho = 0$ we took

$$(7.4) \qquad \rho = e^{t_k} (e^{\tau_k} - 1) \sim \tau_k e^{t_k} .$$

Substitution into (7.3) yields

$$(7.3') \qquad \tau_k \leq 24^{1/6} * e^{-2t_k/3} .$$

From this it follows that the parameters ro1 and ro2 should be chosen according to

$$(7.4') \qquad ro1 = ro2 = 24^{1/6} * e^{t_k/3} .$$

When the second order scheme is used, a similar calculation yields (compare (3.5) with $z_1 = z_2$)

$$ro1 = ro2 = 2^{1/6} * e^{2t_k/3} .$$

However, since both schemes give rise to almost the same results, in table 7.2 only the results obtained with the fourth order scheme are presented.

Parameters used

te = 6.5,
phi = π, sigma1 = sigma2 = e^t, ro1 = ro2 = $24^{1/6} * e^{t/3}$,
sigma0 = ro0 = 0,
p = 4,
aeta = η_a, reta = η_r, hmin = .01.

Table 7.2 The effective fourth order method applied to problem (7.2)

$\eta_a = \eta_r$	hmax	stepnumber k	$-^{10}\log\lvert u(6.5) - \tilde{u}(6.5)\rvert$
10^{-2}	.1	159	6.4
10^{-1}	.1	105	4.2
10^{-2}	.5	147	6.4
10^{-1}	.5	81	4.6

A more detailed description of the experiment with $\eta_a = \eta_r = 10^{-1}$ and hmax = .5 is shown in table 7.3. We have respectively given the stepnumber k, the value t_k of the integration variable, the number of correct digits of u(6.5), the maximal step τ_s allowed by stability, the maximal step τ_a allowed by (4.4'), the actual step $\tau = \min(\tau_s, \tau_a)$ and the eigenvalue δ.

Table 7.3 The effective fourth order method applied to problem (7.2) with $\eta_a = \eta_r = 10^{-1}$ and hmax = .5

k	t	$-^{10}\log \varepsilon$	τ_s	τ_a	τ	$\lvert\delta\rvert$
5	.174	2.9	1.58	.07	.07	1.19
10	1.588	2.3	.80	.45	.45	4.89
15	3.303	2.3	.22	.30	.22	27.2
20	4.062	2.6	.12	.17	.12	58.1
25	4.556	2.4	.086	.135	.086	95.2
30	4.925	1.9	.067	.067	.067	137.6
35	5.191	2.2	.055	.074	.055	179.7
40	5.412	2.5	.48	.057	.048	224.1
50	5.774	2.8	.037	.024	.024	321.9
60	6.068	2.2	.030	.035	.030	432.0
70	6.295	3.0	.026	.027	.026	541.6
80	6.495	3.1	.023	.013	.013	661.8
81	6.500	4.6	.022	.014	.005	665.1

From these results it is seen that initially the discrepancy of linearity controls the step size ([0,1.6]). For t > 3 the stiffness of the equation becomes an important factor; in the interval [3.3, 5.4] the steps are completely determined by stability conditions. However, when t increases the equation also becomes increasingly non-linear; for t > 5.4 both stiffness and non-linearity enter in the determination of a suitable step length.

A third order differential equation

Consider the initial value problem

$$(7.5)\quad \begin{cases} \dddot{U} + (1-2r\cos\phi)\,\ddot{U} + r\,(r-2\cos\phi)\,\dot{U} + r^2 U = 0, \\ U(0) = 1,\ \dot{U}(0) = 0,\ \ddot{U}(0) = 0 \end{cases}$$

where r and ϕ are given parameters.
This problem can be written in the equivalent form

$$(7.5') \quad \begin{cases} \dot{\vec{U}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -r^2 - r(r-2\cos\phi) & 2r\cos\phi - 1 \end{pmatrix} \vec{U}, \\ \vec{U}(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \end{cases}$$

where $\vec{U}$ has the components U, $\dot{U}$ and $\ddot{U}$.

The eigenvalues of the Jacobian matrix of (7.3') are

$$-1, \; r\, e^{i\phi} \text{ and } r\, e^{-i\phi} .$$

When r = 1000 and $\phi = \frac{2}{3}\pi$, the analytical solution of this problem is given by

$$\vec{U} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} e^{-t} .$$

We use this problem to show some results of the interpolation formula (5.3).

Parameters used

te = τ,
phi = $\frac{2}{3}\pi$, sigma1 = sigma2 = 1000,
p = 4, hmin = hmax = τ,
fillI = true

Starting in t = 0 we applied formula (5.3) with $\tau_k = 1$ and $\tau_k = .5$, respectively. In table 7.4 we have given the interpolation point t and $-^{10}\log \varepsilon$, where $\varepsilon = \max_{k=1,2,3} |u_k(t) - \tilde{u}_k(t)|$.

Table 7.4 The interpolation formula applied to problem (7.5')

t	τ=1	τ=.5
.1	2.9	3.6
.2	2.3	3.1
.3	2.0	3.0
.4	1.8	3.0
.5	1.7	3.4
.6	1.6	
.7	1.6	
.8	1.7	
.9	2.2	
1.0	1.7	

A system of two non-linear equations

In nuclear reactor physics the following problem is of interest (cf. [7]).

$$(7.6)\quad \begin{cases} \dot{U}_1 = .2(U_2-U_1), \\ \dot{U}_2 = 10U_1 - (60+\frac{t}{8})\, U_2 + .124t, \\ U_1(0) = 0,\ U_2(0) = 0 \end{cases}$$

The eigenvalues of the Jacobian matrix of (7.6) are approximately -60 and -.17, changing slightly during the integration from t = 0 to t = 10. An analytical solution is not obtained and the results from EFFORK, p = 4, using a small steplength were taken as reference solution.

Since a uniform steplength was used, only the following parameters are of interest:

$$\text{phi} = \pi$$

$$\text{sigma1} = \text{sigma2} = \frac{1}{2}(60.2+\frac{t}{8}+\text{sqrt}((60.2+\frac{t}{8}) - \frac{4}{5} * (60+\frac{t}{8}) +8)$$

$$\text{hmax} = \text{hmin} = \tau.$$

In table 7.5 we listed $-^{10}\log|u_k(10)-\tilde{u}_k(10)|$, k = 1,2 for some values of the step size τ. Both second and fourth effective order schemes were used.

Table 7.5 Numerical results for problem (7.6)

	p = 4		p = 2	
τ	u_1	u_2	u_1	u_2
.1	8.4	6.4	5.7	6.6
.2	7.3	5.3	4.6	5.0
.3	7.1	4.6	4.1	4.8
.4	6.1	4.0	3.8	3.6
.5	4.4	4.9	3.5	4.4
.6	unstable		3.1	2.5
.7			2.9	2.7
.8			2.5	1.7
.9			unstable	

An examination of these results clearly shows that the p = 2-scheme has an extended region of stability, while the p = 4-scheme is more accurate.

Finally we give some results of the interpolation formula, applied to (7.6). Using the fourth order scheme, one integration step of length .5 was performed, starting in t = 0 and t = .5, respectively. The results at some interpolation points are listed in tables 7.6 and 7.7. Hereby, we denoted by u_i the reference solution and by $\tilde{u}_i$ the solution obtained with the interpolation formula.

Table 7.6 The interpolation formula applied to problem (7.6)

t	u_1	$\lvert u_1-\tilde{u}_1\rvert$	u_2	$\lvert u_2-\tilde{u}_2\rvert$
.1	$.15\ 10^{-5}$	$4\ 10^{-5}$	$.17\ 10^{-3}$	10^{-2}
.2	$.69\ 10^{-5}$	10^{-4}	$.38\ 10^{-3}$	$4\ 10^{-2}$
.3	$.16\ 10^{-4}$	$2\ 10^{-4}$	$.59\ 10^{-3}$	$6\ 10^{-2}$
.4	$.30\ 10^{-4}$	$2\ 10^{-4}$	$.80\ 10^{-3}$	$5\ 10^{-2}$
.5	$.47\ 10^{-4}$	$7\ 10^{-9}$	$.10\ 10^{-2}$	$3\ 10^{-5}$

Table 7.7 Results of the interpolation formula with initial value t = .5

t	u_1	$\lvert u_1-\tilde{u}_1\rvert$	$-^{10}\log(\lvert u_1-\tilde{u}_1\rvert/u_1)$	u_2	$\lvert u_2-\tilde{u}_2\rvert$	$-^{10}\log(\lvert u_2-\tilde{u}_2\rvert/u_2)$
.6	$.68\ 10^{-4}$	$3\ 10^{-8}$	3.4	$.12\ 10^{-2}$	$8\ 10^{-6}$	2.2
.7	$.93\ 10^{-4}$	$9\ 10^{-8}$	3.0	$.14\ 10^{-2}$	$2\ 10^{-5}$	1.8
.8	$.12\ 10^{-3}$	10^{-7}	3.0	$.16\ 10^{-2}$	$4\ 10^{-5}$	1.7
.9	$.15\ 10^{-3}$	10^{-7}	3.1	$.18\ 10^{-2}$	$3\ 10^{-5}$	1.8
1.0	$.19\ 10^{-3}$	$8\ 10^{-9}$	4.4	$.21\ 10^{-2}$	$2\ 10^{-6}$	3.0

From these tables it may be concluded that the interpolation formula yields poor approximations in the initial phase, but is quite satisfactory when the stiff components become negligible.

References

[1] Butcher, J.C., Implicit Runge Kutta processes, Math. Comp. 18,50 (1964).

[2] Fowler, M.E., R.M. Warten, A numerical integration technique for ordinary differential equations with widely seperated eigenvalues, I.B.M. Journal, 537-543, (1967).

[3] Houwen., P.J. van der, J. Kok, Numerical solution of a minimax problem, TW report 123/71, Mathematisch Centrum, Amsterdam, (1971).

[4] Houwen, P.J. van der, Stabilized Runge Kutta methods with limited storage requirements, TW report 124/71, Mathematisch Centrum, Amsterdam (1971).

[5] _____, A survey of stabilized Runge Kutta formulae, MC tract 37, Mathematisch Centrum, Amsterdam (1971).

[6] _____, Explicit Runge-Kutta formulas with increased stability boundaries, Numer. Math. 20, 149-164 (1972).

[7] W. Liniger, R. Willoughby, Efficient integration methods for stiff systems of ordinary differential equations, SIAM J. Numer. Anal., vol. 7, no. 1 (1970).