IV.8 Implementation of Implicit Runge-Kutta Methods

These have not been used to any great extent ...
(S.P. Nørsett 1976)

However, the implementation difficulties of these methods have precluded their general use; ...
(J.M. Varah 1979)

Although Runge-Kutta methods present an attractive alternative, especially for stiff problems, ... it is generally believed that they will never be competitive with multistep methods.
(K. Burrage, J.C. Butcher & F.H. Chipman 1980)

Runge-Kutta methods for stiff problems, we are just beginning to explore them ...
(L. Shampine in Aiken 1985)

If the dimension of the differential equation $y' = f(x,y)$ is n, then the s-stage fully implicit Runge-Kutta method (3.1) involves a $n \cdot s$-dimensional nonlinear system for the unknowns $g_1, \dots, g_s$. An efficient solution of this system is the main problem in the implementation of an implicit Runge-Kutta method.

Among the methods discussed in Sect. IV.5, the processes Radau IIA of Ehle, which are L-stable and of high order, seem to be particularly promising. Most of the questions arising (starting values and stopping criteria for the simplified Newton iterations, efficient solution of the linear systems, and the selection of the step sizes) are discussed here for the particular Ehle method with $s = 3$ and $p = 5$. This then constitutes a description of the code RADAU5 of the appendix. An adaptation of the described techniques to other fully implicit Runge-Kutta methods is more or less straight-forward, if the Runge-Kutta matrix has at least one real eigenvalue. We also describe briefly our implementation of the diagonal implicit method SDIRK4 (Formula (6.16)).

Reformulation of the Nonlinear System

In order to reduce the influence of round-off errors we prefer to work with the smaller quantities

$$z_i = g_i - y_0. \tag{8.1}$$

Then (3.1a) becomes

$$z_i = h \sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, y_0 + z_j) \qquad i = 1, \dots, s. \tag{8.2}$$

Whenever the solution $z_1, \dots, z_s$ of the system (8.2a) is known, then (3.1b) is an explicit formula for y_1. A direct application of this requires s additional function evaluations. These can be avoided, if the matrix $A = (a_{ij})$ of the Runge-Kutta coefficients is nonsingular. Indeed, (8.2a) can be written as

$$\begin{pmatrix} z_1 \\ \vdots \\ z_s \end{pmatrix} = A \begin{pmatrix} hf(x_0 + c_1 h, y_0 + z_1) \\ \vdots \\ hf(x_0 + c_s h, y_0 + z_s) \end{pmatrix},$$

so that (3.1b) is seen to be equivalent to

$$y_1 = y_0 + \sum_{i=1}^{s} d_i z_i \tag{8.2b}$$

where

$$(d_1, \dots, d_s) = (b_1, \dots, b_s) A^{-1}. \tag{8.3}$$

For the 3-stage Radau IIA method (Table 5.6) the vector d is simply $(0, 0, 1)$, since $b_i = a_{si}$ for all i. (stiffly accurate)

Another advantage of Formula (8.2b) is the following: the quantities $z_1, \dots, z_s$ are computed iteratively and are therefore affected by iteration errors. The evaluation of $f(x_0 + c_i h, y_0 + z_i)$ in Eq. (3.1b) would then, due to the large Lipschitz constant of f, amplify these errors, which then "can be disastrously inaccurate for a stiff problem" (L.F. Shampine 1980).

Simplified Newton Iterations

For a general nonlinear differential equation the system (8.2a) has to be solved iteratively. In the stone-age of stiff computation (i.e., before 1967) people were usually thinking of simple fixed-point iteration. But this transforms the algorithm into an explicit method and destroys the good stability properties. The paper of Liniger & Willoughby (1970) then showed the advantages of using Newton's method for this purpose. Newton's method applied to system (8.2a) needs for each iteration the solution of a linear system with matrix

$$\begin{pmatrix} I - ha_{11} \frac{\partial f}{\partial y}(x_0 + c_1 h, y_0 + z_1) & \dots & -ha_{1s} \frac{\partial f}{\partial y}(x_0 + c_s h, y_0 + z_s) \\ \vdots & & \vdots \\ -ha_{s1} \frac{\partial f}{\partial y}(x_0 + c_1 h, y_0 + z_1) & \dots & I - ha_{ss} \frac{\partial f}{\partial y}(x_0 + c_s h, y_0 + z_s) \end{pmatrix}.$$

In order to simplify this, we replace all Jacobians $\frac{\partial f}{\partial y}(x_0 + c_i h, y_0 + z_i)$ by an approximation

$$J \approx \frac{\partial f}{\partial y}(x_0, y_0).$$

Then the simplified Newton iterations for (8.2a) become

$$(I - hA \otimes J)\Delta Z^k = -Z^k + h(A \otimes I)F(Z^k)$$
$$Z^{k+1} = Z^k + \Delta Z^k. \tag{8.4}$$

Here $Z^k = (z_1^k, \ldots, z_s^k)^T$ is the k-th approximation to the solution, and $\Delta Z^k = (\Delta z_1^k, \ldots, \Delta z_s^k)^T$ are the increments. $F(Z^k)$ is an abbreviation for

$$F(Z^k) = \big(f(x_0 + c_1 h, y_0 + z_1^k), \ldots, f(x_0 + c_s h, y_0 + z_s^k)\big)^T .$$

Each iteration requires s evaluations of f and the solution of a $n \cdot s$-dimensional linear system. The matrix $(I - hA \otimes J)$ is the same for all iterations. Its LU-decomposition is done only once and is usually very costly.

Starting Values for the Newton Iteration. A natural and simple choice for the starting values in the iteration (8.4) (or equivalently (8.13) below), since the exact solution of (8.2a) satisfies $z_i = \mathcal{O}(h)$, would be

$$z_i^0 = 0, \qquad i = 1, \ldots, s. \tag{8.5}$$

However, better choices are possible in general. If the implicit Runge-Kutta method satisfies the condition $C(\eta)$ (see Sections IV.5 and II.7) for some $\eta \le s$, then

$$z_i = y(x_0 + c_i h) - y_0 + \mathcal{O}(h^{\eta+1}). \tag{8.6}$$

Suppose now that $c_i \neq 0$ $(i = 1, \ldots, s)$ and consider the interpolation polynomial of degree s, defined by

$$q(0) = 0, \qquad q(c_i) = z_i \qquad i = 1, \ldots, s.$$

Since the interpolation error is of size $\mathcal{O}(h^{s+1})$ we obtain together with (8.6)

$$y(x_0 + th) - y_0 - q(t) = \mathcal{O}(h^{\eta+1})$$

(cf. Theorem 7.10 of Chapter II for collocation methods). We use the values of $q(t)$ also beyond the interval $[0, 1]$ and take

$$z_i^0 = q(1 + w c_i) + y_0 - y_1, \quad i = 1, \ldots, s, \quad w = h_{new}/h_{old} \tag{8.5'}$$

as starting values for the Newton iteration in the subsequent step. Numerical experiments with the 3-stage Radau IIA method have shown that (8.5') usually leads to a faster convergence than (8.5).

Stopping Criterion. This question is closely related to an estimation of the iteration error. Since convergence is linear, we have

$$\|\Delta Z^{k+1}\| \le \Theta \|\Delta Z^k\|, \qquad \text{hopefully with} \qquad \Theta < 1. \tag{8.7}$$

Applying the triangle inequality to

$$Z^{k+1} - Z^* = (Z^{k+1} - Z^{k+2}) + (Z^{k+2} - Z^{k+3}) + \ldots$$

(where Z^* is the exact solution of (8.2a)) yields the estimate

$$\|Z^{k+1} - Z^*\| \le \frac{\Theta}{1-\Theta} \|\Delta Z^k\|. \tag{8.8}$$

The convergence rate Θ can be estimated by the computed quantities

$$\Theta_k = \|\Delta Z^k\| / \|\Delta Z^{k-1}\|, \qquad k \ge 1. \tag{8.9}$$

It is clear that the iteration error should not be larger than the local discretization error, which is usually kept close to Tol. We therefore stop the iteration when

$$\eta_k \|\Delta Z^k\| \le \kappa \cdot Tol \quad \text{with} \quad \eta_k = \frac{\Theta_k}{1 - \Theta_k} \tag{8.10}$$

and accept Z^{k+1} as approximation to Z^*. This strategy can only be applied after at least two iterations. In order to be able to stop the computations after the first iteration already (which is especially advantageous for linear systems) we take for $k = 0$ the quantity

$$\eta_0 = (\max(\eta_{old}, Uround))^{0.8}$$

where η_{old} is the last η_k of the preceding step. It remains to make a good choice for the parameter κ in (8.10). To this end we applied the code RADAU5 for many different values of κ between 10 and 10^{-4} and with some different tolerances Tol to several differential equations. The observation was that the code works most efficiently for values of κ around 10^{-1} or 10^{-2}.

It is our experience that the code becomes more efficient when we allow a relatively high number of iterations (e.g., $k_{max} = 7$ or 10). During these k_{max} iterations, the computations are interrupted and restarted with a smaller stepsize (for example with $h := h/2$) if one of the following situations occurs

a) there is a k with $\Theta_k \ge 1$ (the iteration "diverges");

b) for some k,

$$\frac{\Theta_k^{k_{max}-k}}{1 - \Theta_k} \|\Delta Z^k\| > \kappa \cdot Tol. \tag{8.11}$$

The left-hand expression in (8.11) is a rough estimate of the iteration error to be expected after $k_{max} - 1$ iterations. The norm, used in all these formulas, should be the same as the one used for the local error estimator.

If only one Newton iteration was necessary to satisfy (8.10) or if the last Θ_k was very small, say $\le 10^{-3}$, then we don't recompute the Jacobian in the next step. As a consequence, the Jacobian is computed only once for linear problems with constant coefficients (as long as no step rejection occurs).

The Linear System

An essential gain of numerical work for the solution of the linear system (8.4) is obtained by the following method, introduced independently by Butcher (1976) and Bickart (1977), which exploits with much profit the special structure of the matrix $I - hA \otimes J$ in (8.4).

The idea is to premultiply (8.4) by $(hA)^{-1} \otimes I$ (we suppose here that A is invertible) and to transform A^{-1} to a simple matrix (diagonal, block diagonal, triangular or Jordan canonical form)

$$T^{-1} A^{-1} T = \Lambda. \tag{8.12}$$

With the transformed variables $W^k = (T^{-1} \otimes I)Z^k$, the iteration (8.4) becomes equivalent to

$$(h^{-1}\Lambda \otimes I - I \otimes J)\Delta W^k = -h^{-1}(\Lambda \otimes I)W^k + (T^{-1} \otimes I)F((T \otimes I)W^k) \qquad (8.13)$$
$$W^{k+1} = W^k + \Delta W^k.$$

We also replace Z^k and ΔZ^k by W^k and ΔW^k in the formulas (8.7)–(8.11) (and thereby again save some work).

For the sequel, we suppose that the matrix A^{-1} has one real eigenvalue $\widehat{\gamma}$ and one complex conjugate eigenvalue pair $\widehat{\alpha} \pm i\widehat{\beta}$. This is a typical situation for 3-stage implicit Runge-Kutta methods such as Radau IIA. With $\gamma = h^{-1}\widehat{\gamma}, \alpha = h^{-1}\widehat{\alpha}, \beta = h^{-1}\widehat{\beta}$ the matrix in (8.13) becomes

$$\begin{pmatrix} \gamma I - J & 0 & 0 \\ 0 & \alpha I - J & -\beta I \\ 0 & \beta I & \alpha I - J \end{pmatrix} \qquad (8.14)$$

so that (8.13) splits into two linear systems of dimension n and $2n$, respectively. Several ideas are possible to exploit the special structure of the $2n \times 2n$-submatrix. The easiest and numerically most stable way has turned out to be the following: transform the real subsystem of dimension $2n$ into an n-dimensional, complex system

$$((\alpha + i\beta)I - J)(u + iv) = a + ib \qquad (8.14')$$

and apply simple Gaussian elimination. For machines without complex arithmetic, one just has to modify the linear algebra routines. Then a complex multiplication consists of 4 real multiplications and the amount of work for the solution of (8.14') becomes approximately $4n^3/3$ operations. Thus the total work for system (8.14) is about $5n^3/3$ operations. Compared to $(3n)^3/3$, which would be the number of operations necessary for decomposing the untransformed matrix $I - hA \otimes J$ in (8.4), we gain a factor of about 5 in arithmetical operations. Observe that the transformations, such as $Z^k = (T \otimes I)W^k$, need only $\mathcal{O}(n)$ additions and multiplications. The gain is still more drastic for methods with more than 3 stages.

Transformation to Hessenberg Form. For large systems with a full Jacobian J a further gain is possible by transforming J to Hessenberg form

$$S^{-1}JS = H = \begin{pmatrix} * & \dots & * & * \\ * & & & * \\ & \ddots & & \vdots \\ & & * & * \end{pmatrix}. \qquad (8.15)$$

This procedure was originally proposed for multistep methods by Enright (197[illegible] and extended to the Runge-Kutta case by Varah (1979). With the code ELMHES taken from LINPACK, this is performed with $2n^3/3$ operations. Because the multiplication of S with a vector needs only $n^2/2$ operations (observe that S is triangular) the solution of (8.13) is found in $\mathcal{O}(n^2)$ operations, if the Hessenberg matrix H is known. This transformation is especially advantageous, if the Jacobian J is not changed during several steps.

Step Size Selection

One possibility to select the step sizes is Richardson extrapolation (cf. Sect. II.4). We describe here the use of an embedded pair of methods which is easier to program and which makes the code more flexible. The following formulas are for the special case of the 3-stage Radau IIA methods; the same ideas are applicable to all implicit Runge-Kutta methods, whose Runge-Kutta matrix has at least one real eigenvalue.

Embedded Formula. Since our method is of optimal order, it is impossible to embed it efficiently into one of still higher order. Therefore we search for a lower order method of the form

$$\widehat{y}_1 = y_0 + h\Big(\widehat{b}_0 f(x_0, y_0) + \sum_{i=1}^{3} \widehat{b}_i f(x_0 + c_i h, g_i)\Big) \qquad (8.16)$$

where g_1, g_2, g_3 are the values obtained from the Radau IIA method and $\widehat{b}_0 \neq 0$ (the choice $\widehat{b}_0 = \gamma_0 = \widehat{\gamma}^{-1}$, where $\widehat{\gamma}$ is the real eigenvalue of the matrix A^{-1}, again saves some multiplications). The difference

$$\widehat{y}_1 - y_1 = \gamma_0 h f(x_0, y_0) + \sum_{i=1}^{3} (\widehat{b}_i - b_i) h f(x_0 + c_i h, g_i),$$

which can also be written in the form

$$\widehat{y}_1 - y_1 = \gamma_0 h f(x_0, y_0) + e_1 z_1 + e_2 z_2 + e_3 z_3, \qquad (8.17)$$

then serves for error estimation. In order that $\widehat{y}_1 - y_1 = \mathcal{O}(h^4)$ the coefficients have to satisfy

$$(e_1, e_2, e_3) = \frac{\gamma_0}{3}(-13 - 7\sqrt{6}, -13 + 7\sqrt{6}, -1). \qquad (8.18)$$

Unfortunately, for $y' = \lambda y$ and $h\lambda \to \infty$ the difference (8.17) behaves like $\widehat{y}_1 - y_1 \approx \gamma_0 h\lambda y_0$, which is unbounded and therefore not suitable for stiff equations. We propose (an idea of Shampine) to use instead

$$err = (I - h\gamma_0 J)^{-1}(\widehat{y}_1 - y_1). \qquad (8.19)$$

The LU-decomposition of $((h\gamma_0)^{-1}I - J)$ is available anyway from the previous work, so that the computation of (8.19) is cheap. For $h \to 0$ we still have $err = \mathcal{O}(h^4)$, and for $h\lambda \to \infty$ (if $y' = \lambda y$ and $J = \lambda$) we obtain $err \to -1$.

This behaviour (for $h\lambda \to \infty$) is already much better than that for $\widehat{y}_1 - y_1$, but it is not good enough in order to avoid the "hump" phenomenon, described in Sect. IV.7. In the first step and after every rejected step for which $\|err\| > 1$, we therefore use instead of (8.19) the expression

$$\widetilde{err} = (I - h\gamma_0 J)^{-1}\big(\gamma_0 h f(x_0, y_0 + err) + e_1 z_1 + e_2 z_2 + e_3 z_3\big) \qquad (8.20)$$

for step size prediction. This requires one additional function evaluation, but satisfies $\widetilde{err} \to 0$ for $h\lambda \to \infty$, as does the error of the numerical solution.

Standard Step Size Controller. Since the expressions (8.19) and (8.20) behave like $\mathcal{O}(h^4)$ for $h \to 0$, the standard step size prediction leads to

$$h_{new} = fac \cdot h_{old} \cdot \|err\|^{-1/4}. \tag{8.21}$$

where

$$R = \|err\| = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{err_i}{sc_i}\right)^2},$$

and $sc_i = Atol_i + \max(|y_{0i}|, |y_{1i}|) \cdot Rtol_i$ as in (4.11) of Sect. II.4. Here, the safety factor fac is proposed to depend on $Newt$, the number of Newton iterations of the current step and on the maximal number of Newton iterations k_{max}, say, as:

$$fac = 0.9 \times (2k_{max} + 1)/(2k_{max} + Newt).$$

In order to save LU-decompositions of the matrix (8.14), we also include the following strategy: if no Jacobian is recomputed and if the step size h_{new}, defined by (8.21), satisfies

$$c_1 h_{old} \le h_{new} \le c_2 h_{old} \tag{8.22}$$

with, say $c_1 = 1.0$ and $c_2 = 1.2$, then we retain h_{old} for the following step.

Predictive Controller. The step size prediction by formula (8.21) has the disadvantage that step size reductions by more than the factor fac are not possible without step rejections (observe that $h_{new} < fac \cdot h_{old}$ implies $\|err\| > 1$). For stiff differential equations, however, a rapid decrease of the step size is often required (see for example the situation of Fig. 8.1, where the step size drops from 10^{-2} to 10^{-7} within a very small time interval). Denoting by err_{n+1} the error expression (8.19) (or (8.20)), computed in the nth step with step size h_n, step size predictions are typically derived from the asymptotic formula

$$\|err_{n+1}\| = C_n h_n^4. \tag{8.23}$$

The strategy (8.21) is based on the additional assumption $C_{n+1} \approx C_n$, which, as we have seen, is not always very realistic.

A careful control-theoretic study of step size strategies has been undertaken by Gustafsson (1994). He came to the conclusion that a better model is to assume that $\log C_n$ is a linear function of n. This means that $\log C_{n+1} - \log C_n$ is constant or, equivalently,

$$C_{n+1}/C_n \approx C_n/C_{n-1}. \tag{8.24}$$

Inserting C_n and C_{n-1} from (8.23) and C_{n+1} from $1 = C_{n+1} h_{new}^4$ into (8.24) yields

$$h_{new} = fac \cdot h_n \left(\frac{1}{\|err_{n+1}\|}\right)^{1/4} \cdot \frac{h_n}{h_{n-1}} \left(\frac{\|err_n\|}{\|err_{n+1}\|}\right)^{1/4}. \tag{8.25}$$

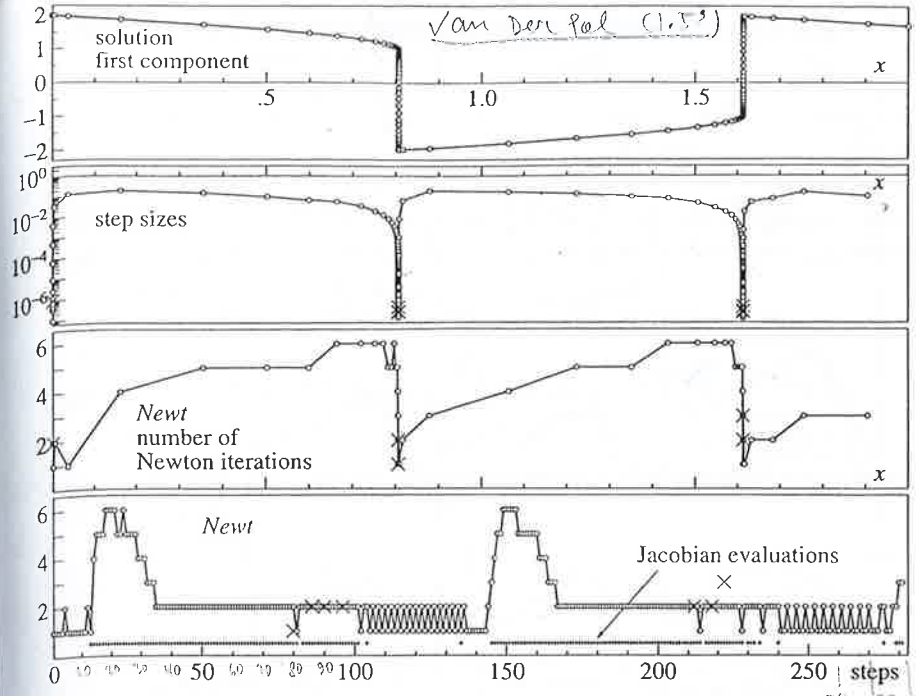

Fig. 8.1. Solution, step sizes and Newton iterations for RADAU5

In our code RADAU5 we take the minimum of the two step sizes (8.21) and (8.25). For the problem considered in Fig. 8.1, this new strategy reduces the number of rejected steps from 27 to 7.

Numerical Study of the Step-Control Mechanism. As a representative example we choose the van der Pol equation (1.5') with $\varepsilon = 10^{-6}$, initial values $y_1(0) = 2$, $y_2(0) = -0.6$ and integration interval $0 \le x \le 2$. Fig. 8.1 shows four pictures. The first one presents the solution $y_1(x)$ with all accepted integration steps for $Atol = Rtol = 10^{-4}$. Below this, the step sizes obtained by RADAU5 are plotted as function of x. The solid line represents the accepted steps. The rejected steps are indicated by $\times$'s. Observe the very small step sizes which are required in the rapid transients between the smooth parts of the solution. The lowest two pictures give the number of Newton iterations needed for solving the nonlinear system (8.2a), once as function of x, and once as function of the step-number. The last picture also indicates the steps where the Jacobian has been recomputed.

Another numerical experiment (Fig. 8.2) illustrates the quality of the error estimates. We applied the code RADAU5 with $Atol = Rtol = 10^{-4}$ and initial step size $h = 10^{-4}$ to the above problem and plotted at several chosen points of the numerical solution

a) the exact local error (marked by small circles)

b) the estimates (8.19) and (8.20) (marked by ✧ and ⌑ respectively)

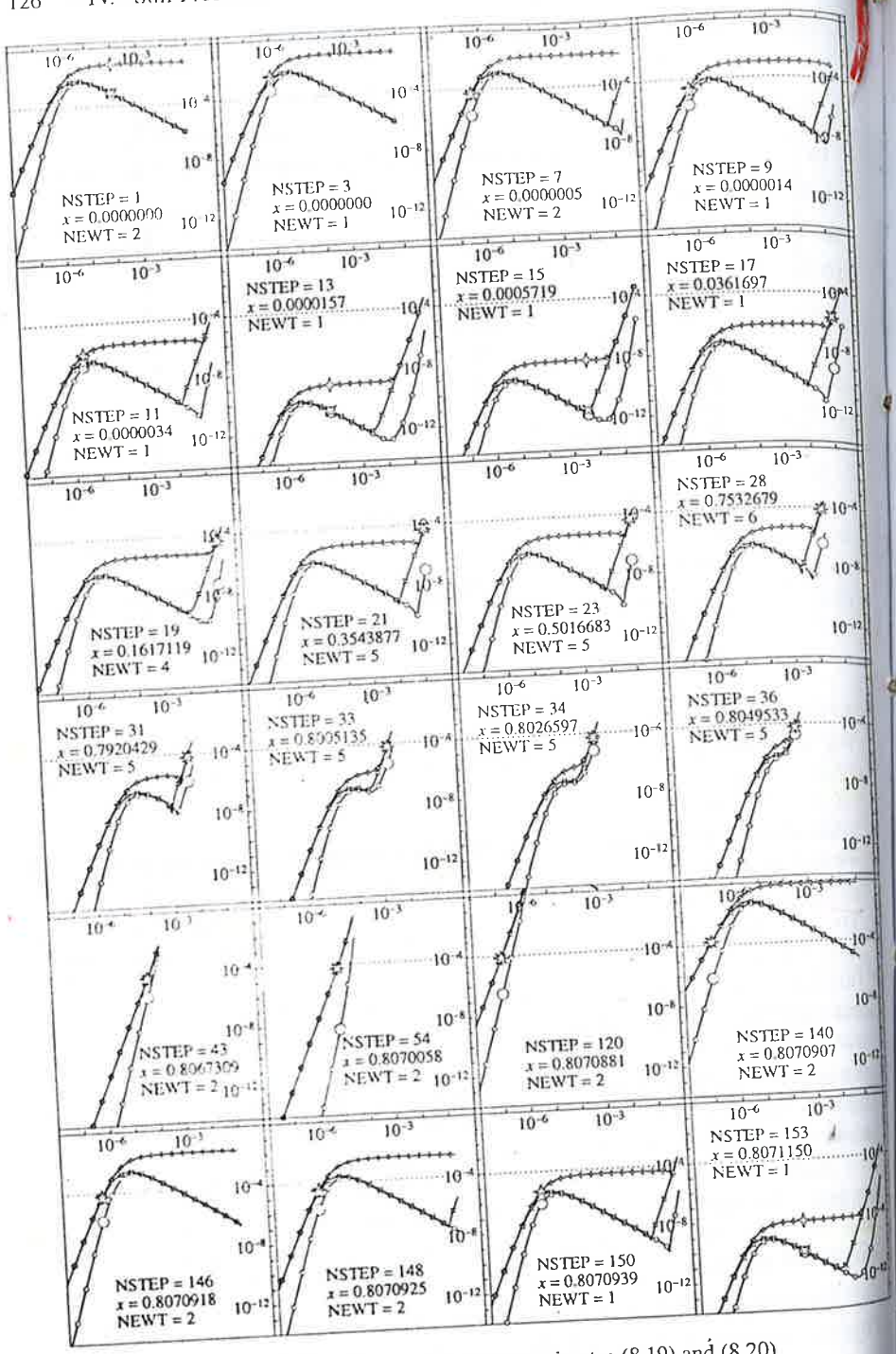

Fig. 8.2. Exact local error and the estimates (8.19) and (8.20)

as functions of h. The large symbols indicate the position of the actually use size. *Newt* is the number of required Newton iterations.

It is interesting to note that the local error behaves like $\mathcal{O}(h^6)$ (straight line of slope 6) only for $h \le \varepsilon$ and for large h. Between these regions, the local error *grows* like $\mathcal{O}(h^{-1})$ with decreasing h. This is the only region where the error estimate (8.20) is significantly better than (8.19). Therefore, we use the more expensive estimator (8.20) only in the first and after each rejected step. In any way, *both* error estimators are always above the actual local error, so that the code usually produces very precise results.

Implicit Differential Equations

Many applications (such as space discretizations of parabolic differential equations) often lead to systems of the form

$$My' = f(x,y), \qquad y(x_0) = y_0 \tag{8.26}$$

with a constant matrix M. For such problems we formally replace all f's by $M^{-1}f$ and multiply the resulting equations by M. Formulas (8.13) and (8.19) then have to be replaced by

$$\left(h^{-1}\Lambda \otimes M - I \otimes J\right) \Delta W^k = -h^{-1}\left(\Lambda \otimes M\right)W^k + (T^{-1} \otimes I)F((T \otimes I)W^k) \tag{8.13a}$$

$$err = \left((h\gamma_0)^{-1}M - J\right)^{-1}\left(f(x_0,y_0) + (h\gamma_0)^{-1}M(e_1 z_1 + e_2 z_2 + e_3 z_3)\right). \tag{8.19a}$$

Here the matrix J is again an approximation to $\partial f/\partial y$. These formulas may even be applied to certain problems (8.26) with singular M (for more details see Chapters VI and VII).

Solving the linear system (8.13a) is done by a decomposition of the matrix (see (8.14), (8.14')) of the form

$$\begin{pmatrix} \gamma M - J & 0 \\ 0 & (\alpha + i\beta)M - J \end{pmatrix}. \tag{8.27}$$

If M and J are banded or sparse, the matrices $\gamma M - J$ and $(\alpha + i\beta)M - J$ remain banded or sparse, respectively. The code RADAU5 of the appendix has options for banded structures.

An SDIRK-Code

We have also coded, using many of the above ideas, the SDIRK formula (6.16) together with the global solution (6.17). For this method also, it was again very important to replace the error estimator $y_1 - \widehat{y}_1$ by (8.19).

Here, in contrast to fully implicit Runge-Kutta methods, one can treat the stages one after the other. Such a serial computation has the advantage that the information of the already computed stages can be used for a good choice of the starting values for the Newton iterations in the subsequent stages. For example, suppose that

$$z_1 = \gamma h f(x_0 + \gamma h, y_0 + z_1)$$
$$z_2 = \gamma h f(x_0 + c_2 h, y_0 + z_2) + a_{21} h f(x_0 + \gamma h, y_0 + z_1)$$

are already available. Since for all i

$$z_i = c_i h f(x_0, y_0) + (\sum_j a_{ij} c_j) h^2 (f_x + f_y f)(x_0, y_0) + \mathcal{O}(h^3),$$

by solving

$$\begin{pmatrix} c_1 & c_2 \\ \sum_j a_{1j} c_j & \sum_j a_{2j} c_j \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} c_3 \\ \sum_j a_{3j} c_j \end{pmatrix}$$

one finds α_1, α_2 such that

$$\alpha_1 z_1 + \alpha_2 z_2 = z_3 + \mathcal{O}(h^3).$$

The expression $z_3^{(0)} = \alpha_1 z_1 + \alpha_2 z_2$ then serves as starting value for the computation of z_3. In the last stage one can take $\widehat{y}_1$, which is then available, for starting the Newton iterations for $g_s = y_1$. The computation of z_3, z_4, y_1, done in this way, needs few Newton iterations and a failure of convergence is usually already detected in the first stage.

However, when *parallel* processors are available, the exploitation of the triangular structure of the Runge-Kutta matrix may be less desirable. Whereas in the iteration (8.13) all s function evaluations and much of the linear algebra can be done in parallel, this is no longer possible for DIRK-methods, when $z_1, \dots, z_k$ is used in the computations of z_{k+1}.

SIRK-Methods

> The fact that singly-implicit methods have a coefficient matrix with a one-point spectrum is the key to reducing the operation count for these methods to the level which prevails in linear multistep methods.
>
> (J.C. Butcher, K. Burrage & F.H. Chipman 1980)

In order to avoid the difficulties (in writing a Runge-Kutta code) caused by the complex eigenvalues of the Runge-Kutta matrix A, one may look for methods with real eigenvalues, especially with a single s-fold real eigenvalue. Such methods were introduced by Nørsett (1976). Burrage (1978) provided them with error estimators, and codes in ALGOL and FORTRAN are presented in Butcher, Burrage & Chipman (1980). The basic methods for their code STRIDE are given by the following lemma.

Lemma 8.1. *For collocation methods (i.e., for Runge-Kutta methods satisfying condition $C(s)$ of Sect. IV.5), we have*

$$\det(I - zA) = (1 - \gamma z)^s, \tag{8.28}$$

if and only if

$$c_i = \gamma x_i, \qquad i = 1, \dots, s, \tag{8.29}$$

where $x_1, \dots, x_s$ are the zeros of the Laguerre polynomial $L_s(x)$ (c.f. Formula (6.11)).

Proof. The polynomial $\det(I - zA)$ is the denominator of the stability function (Formula (3.3)), so that by Theorem 3.10

$$M^{(s)}(0) + M^{(s-1)}(0) z + \dots + M(0) z^s = (1 - \gamma z)^s \tag{8.30}$$

with $M(x)$ given by (3.25). Computing $M^{(j)}(0)$ from (8.30) we obtain

$$\frac{1}{s!} \prod_{i=1}^{s} (x - c_i) = M(x) = \sum_{j=0}^{s} \binom{s}{j} (-\gamma)^{s-j} \frac{x^j}{j!} = (-\gamma)^s L_s(\frac{x}{\gamma})$$

which leads to (8.29). □

The stability function of the method of Lemma 8.1 has been studied in Sections IV.4 (multiple real-pole approximations) and IV.6. We have further seen (Proposition 3.8) that $R(\infty) = 0$ when $x_0 + h$ is a collocation point. This means that $c_q = 1$ or $\gamma = 1/x_q$ for $q \in \{1, \dots, s\}$ where $0 < x_1 < \dots < x_s$ are the zeros of $L_s(x)$. However, if we want A-stable methods, Theorem 4.25 restricts this point to be *in the middle* (more precisely: $q = s/2$ or $s/2 + 1$ for s even, $q = (s+1)/2$ for s odd). An apparently undesirable consequence of this is that many of the collocation points lie *outside* the integration interval (for example, for $s = 5$ and $q = 3$ we have $c_1 = 0.073$, $c_2 = 0.393$, $c_3 = 1$, $c_4 = 1.970$, $c_5 = 3.515$).

Since these methods with $\gamma = 1/x_q$ are of order $p = s$ only, it is easy to embed them into a method of higher order. Burrage (1978) added a further stage

$$g_{s+1} = y_0 + h \sum_{j=1}^{s+1} a_{s+1,j} f(x_0 + c_j h, g_j)$$

where c_{s+1} and $a_{s+1,s+1}$ are arbitrary and the other $a_{s+1,j}$ are determined so that the $(s+1)$-stage method satisfies $C(s)$ too. In order to avoid a new LU-decomposition we choose $a_{s+1,s+1} = \gamma$. The coefficient c_{s+1} is fixed arbitrarily

as $c_{s+1} = 0$. We then find a unique method

$$\widehat{y}_1 = y_0 + h\sum_{j=1}^{s+1} \widehat{b}_j f(x_0 + c_j h, g_j)$$

of order $s+1$ by computing the coefficients of the interpolatory quadrature rule. An explicit formula for the matrix T which transforms the Runge-Kutta matrix A to Jordan canonical form and A^{-1} to a very simple lower triangular matrix Λ is given in Exercise 1. It can be used for economically solving the linear system (8.13).

Exercises

1. (Butcher 1979). For the collocation method with $c_1, \ldots, c_s$ given by (8.29) prove that (e.g. for $s = 4$)

$$T^{-1}AT = \gamma \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ & & -1 & 1 \end{pmatrix}, \quad T^{-1}A^{-1}T = \frac{1}{\gamma}\begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

where the transformation T satisfies

$$T = \big(L_{j-1}(x_i)\big)_{i,j=1}^{s}, \qquad T^{-1} = \left(\frac{x_j L_{i-1}(x_j)}{s^2 L_{s-1}(x_j)^2}\right)_{i,j=1}^{s}$$

and $L_{j-1}(x)$ are the Laguerre polynomials.

Hint. Use the identities

$$L_n'(x) = L_{n-1}'(x) - L_{n-1}(x), \qquad L_n(x) = L_{n-1}(x) + \frac{x}{n}L_n'(x)$$

and the Christoffel-Darboux formula

$$\sum_{j=0}^{n} L_j(x)L_j(y) = \frac{n+1}{y-x}\Big(L_{n+1}(x)L_n(y) - L_{n+1}(y)L_n(x)\Big)$$

which, in the limit $y \to x$, becomes

$$\sum_{j=0}^{n} (L_j(x))^2 = (n+1)\Big(L_{n+1}(x)L_n'(x) - L_{n+1}'(x)L_n(x)\Big).$$

IV.9 Extrapolation Methods

> It seems that a suitable version of an IEM (implicit extrapolation method) which takes care of these difficulties may become a very strong competitor to any of the general discretization methods for stiff systems presently known.
>
> (the very last sentence of Stetter's book, 1973)

Extrapolation of explicit methods is an interesting approach to solving nonstiff differential equations (see Sect. II.9). Here we show to what extent the idea of extrapolation can also be used for stiff problems. We shall use the results of Sect. II.8 for the existence of asymptotic expansions and apply them to the study of those implicit and linearly implicit methods, which seem to be most suitable for the computation of stiff differential equations. Our theory here is restricted to classical $h \to 0$ order, the study of stability domains and A-stability.

A big difficulty, however, is the fact that the coefficients and remainders of the asymptotic expansion can explode with increasing stiffness and the h-interval, for which the expansion is meaningful, may tend to zero. Bounds on the remainder which hold uniformly for a class of arbitrarily stiff problems, will be discussed later in Sect. VI.5.

Extrapolation of Symmetric Methods

It is most natural to look first for symmetric one-step methods as the basic integration scheme. Promising candidates are the trapezoidal rule

$$y_{i+1} = y_i + \frac{h}{2}\Big(f(x_i, y_i) + f(x_{i+1}, y_{i+1})\Big), \tag{9.1}$$

and the implicit mid-point rule

$$y_{i+1} = y_i + hf\Big(x_i + \frac{h}{2}, \frac{1}{2}(y_{i+1} + y_i)\Big). \tag{9.2}$$

We take some step-number sequence $n_1 < n_2 < n_3 < \ldots$, set $h_j = H/n_j$ and define

$$T_{j1} = y_{h_j}(x_0 + H), \tag{9.3}$$

the numerical solution obtained by performing n_j steps with step size h_j. As described in Sect. II.9 we extrapolate these values according to

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{(n_j/n_{j-k})^2 - 1}. \tag{9.4}$$