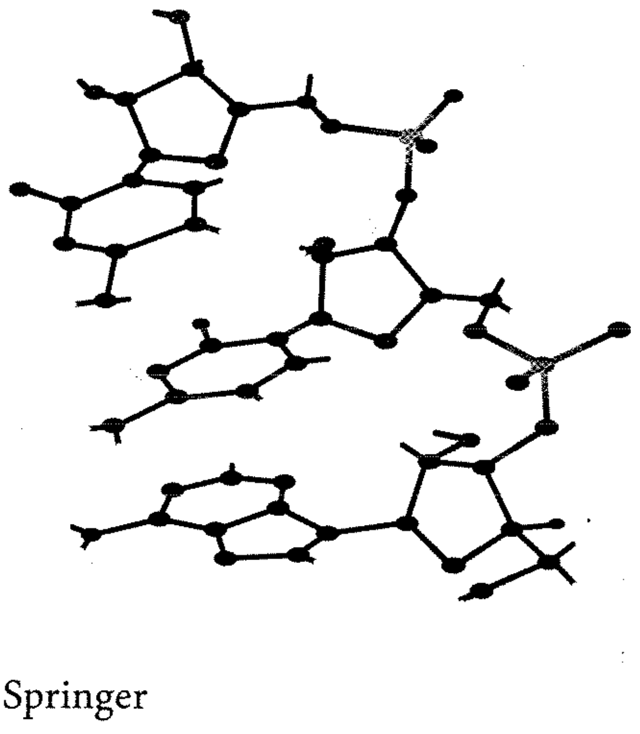
Peter Deuflhard
Folkmar Bornemann

TEXTS IN APPLIED MATHEMATICS 42

Scientific Computing with Ordinary Differential Equations



Springer

Contents

5 Adaptive Control of One-Step Methods

In the previous chapter we met a number of one-step methods that allowed us to approximate the solution $x \in C^1([t_0, T], \mathbf{R}^d)$ of the initial value problem

$$x' = f(t, x), \qquad x(t_0) = x_0,$$

by a mesh function x_Δ, for any *given* mesh $\Delta = \{t_0, t_1, \ldots, t_n\}$, $t_0 < t_1 < \cdots < t_n = T$. For methods of order p we obtained a characterization of the discretization error in the form

$$\max_{t \in \Delta} |x(t) - x_\Delta(t)| \leq C\tau_\Delta^p,$$

where τ_Δ denotes the maximal step size of the mesh Δ. But this information alone does not permit us to specify a priori a suitable mesh Δ. In fact, without further information about the solution, we can hardly come up with a better idea than an equidistant mesh

$$\Delta_n = \left\{ t_0 + j\frac{T - t_0}{n} : j = 0, 1, \ldots, n \right\},$$

with some $\tau = \tau_\Delta$. However, it cannot possibly be expected that such equidistant meshes will do justice to the multitude of different problems. Accordingly, in this chapter we address the question of how to construct efficiently a *problem-adapted* mesh.

In the construction of meshes Δ the following data ought to be taken into account:

1. The accuracy of the approximation x_Δ.

2. The number of mesh points n_Δ.

3. The cost of generating the approximation x_Δ.

Usually, in approximation theory one considers only items 1 and 2 and searches for *optimal* meshes that with given accuracy minimize the number of nodes. On the other hand, the user of algorithms is interested in items 1 and 3 and wants to expend *minimal effort* in generating an approximation with a desired accuracy. Of course, when the effort (including that of the mesh generation!) grows with the number of nodes, the two viewpoints agree with each other. However, in the numerical solution the construction of a problem-adapted mesh Δ will also be a cost factor which a priori is unknown but has to be taken into account. Thus a good heuristic approach is desired, which, of course, will then produce, in general, only *suboptimal* meshes.

Example 5.1 Consider the three-looped periodic satellite orbit of Section 1.1 with period $T = 11.12\ldots$. If the classical Runge-Kutta method of order 4 (Table 4.2) is used with an absolute error tolerance 2.5×10^{-7} for the space variable (corresponding to a position error of 100 m), then for an equidistant mesh Δ_1, about

$$n_1 = 117000$$

nodes and hence $4n_1 = 468000$ f evaluations are required! But here the generation of the equidistant mesh costs nearly nothing.

With the techniques discussed in the next sections it is possible to compute, with the same tolerance, a problem-adapted mesh Δ_2 that has only

$$n_2 = 1883$$

nodes. In this case, a total of 7669 f evaluations are needed, which indicates that the generation of the mesh causes an additional $7669 - 4n_2 = 137$ f evaluations. Compared with the equidistant mesh, we have here a cost savings by a factor of 61 (again measured in terms of f evaluations). This is a factor of about the same order seen in the change from one computer generation to the next!

Obviously, it pays to use problem-adapted meshes. For initial value problems the construction of such meshes reduces to the problem of determining *adaptively* the next *step size*

$$\tau_j = t_{j+1} - t_j.$$

These local step sizes should be as large as possible in order for the computational cost (in the sense of the mentioned heuristics) to be kept small. At the same time, they should be small enough to ensure a specified *quality* for the approximation $x_\Delta(t_{j+1})$. To balance these demands is precisely the problem of an efficient *step-size control*.

5.1 Local Accuracy Control

As indicated we have to decide how to ensure a desired quality for the approximation x_Δ. A first idea may be to use as a quality measure the *discretization error*

$$\epsilon_\Delta(t_{j+1}) = x(t_{j+1}) - x_\Delta(t_{j+1})$$

at the new mesh point t_{j+1}. This error depends on the already computed value $x_\Delta(t_j)$ in the following way

$$\begin{aligned}\epsilon_\Delta(t_{j+1}) = &\left(\Phi^{t_{j+1},t_j}x(t_j) - \Phi^{t_{j+1},t_j}x_\Delta(t_j)\right)\\ &+ \left(\Phi^{t_{j+1},t_j}x_\Delta(t_j) - \Psi^{t_{j+1},t_j}x_\Delta(t_j)\right).\end{aligned}$$

In other words, the error $\epsilon_\Delta(t_{j+1})$ consists of two parts, namely, the *local* discretization error (consistency error)

$$\epsilon_{j+1} = \Phi^{t_{j+1},t_j}x_\Delta(t_j) - \Psi^{t_{j+1},t_j}x_\Delta(t_j)$$

and the part

$$\bar\epsilon_{j+1} = \Phi^{t_{j+1},t_j}x(t_j) - \Phi^{t_{j+1},t_j}x_\Delta(t_j),$$

which represents the "prior" approximation error $\epsilon_\Delta(t_j)$ propagated by the evolution of the initial value problem. In linearized form this second part becomes

$$\bar\epsilon_{j+1} = W(t_{j+1},t_j)\epsilon_\Delta(t_j) + O(|\epsilon_\Delta(t_j)|^2), \tag{5.1}$$

where W is the propagation matrix introduced in Section 3.1.1. This contribution to the error can be corrected only by an improvement of the approximation $x_\Delta(t_j)$ itself, which evidently would mean a repetition of the entire computation up to that point. Because of this global aspect $\epsilon_\Delta(t)$ is also called the *global discretization error*. Accordingly, as a rule, we are restricted to control only the local error contribution ϵ_{j+1} and hence to pose

$$|\epsilon_{j+1}| \leq \text{TOL}$$

as the quality condition. In fact, generally we cannot even determine the quantity $|\epsilon_{j+1}|$ *exactly* and are reduced to using some *computable estimate* $|[\epsilon_{j+1}]| \approx |\epsilon_{j+1}|$. In other words, we are left with an implementable substitute condition of the form

$$|[\epsilon_{j+1}]| \leq \text{TOL}, \tag{5.2}$$

where the *local tolerance* TOL is prescribed by the user as a control quantity.

The error estimate $|[\epsilon_{j+1}]|$ characterizes the quality of the step from t_j to t_{j+1}. If this estimate *satisfies* condition (5.2), then we want to generate from the current information a prediction of the point t_{j+2}, i.e., of a *new* step size τ_{j+1}. Actually, this requires some information about the future that is not available. Thus we have to confine ourselves to compute from the current information some "optimal" *current* step size τ_j^*, which then becomes the *prediction* of the next step $\tau_{j+1} = \tau_j^*$.

On the other hand, if the estimate *violates* condition (5.2), then the current information is used to compute an "optimal" *current* step size τ_j^* and to *repeat* the step with the *corrected value* $\tau_j = \tau_j^*$.

For the "optimal" current step size τ_j^*, we require that the local error $|\epsilon_{j+1}^*|$ resulting from it not fall too far below the desired tolerance (efficiency) nor exceed that tolerance by too much (dependability):

$$\mathrm{TOL} \approx |\epsilon_{j+1}^*|.$$

The local errors $|\epsilon_{j+1}|$, $|\epsilon_{j+1}^*|$ and the step sizes τ_j, τ_j^* are connected by the approximate relation

$$|[\epsilon_{j+1}]| \approx |\epsilon_{j+1}| = c(t_j)\tau_j^{p+1} + O(\tau_j^{p+2}) \approx c(t_j)\tau_j^{p+1} \qquad (5.3)$$

and correspondingly also by

$$\mathrm{TOL} \approx |\epsilon_{j+1}^*| \approx c(t_j)(\tau_j^*)^{p+1}. \qquad (5.4)$$

After dividing (5.4) by (5.3) and solving the result for τ_j^*, we obtain

$$\tau_j^* = \sqrt[p+1]{\frac{\rho \cdot \mathrm{TOL}}{|[\epsilon_{j+1}]|}}\,\tau_j, \qquad (5.5)$$

where for safety's sake we have "adorned" the desired accuracy TOL with a factor $\rho < 1$.

In the above estimation formula (5.5) it is possible, in principle, that the denominator becomes zero or, at least, is so small that exponent overflow occurs. Hence, in general, some additional *bound on the step increase* has to be introduced: Either the growth of the step is limited internally (i.e., inside the program) by some factor $q > 1$, or we force all steps to remain below a maximal step size $\tau_{\max}$ that is externally chosen by the user on the basis of some problem information.

The following *pseudocode* summarizes the algorithmic part of the preceding discussion. Note that in the first step we have to begin with an *externally* given step size prediction τ_0.

Algorithm 5.2 *Basic adaptive algorithm.*

```
$j := 0$
$\Delta := \{t_0\}$
$x_\Delta(t_0) := x_0$
*while* $(t_j < T)$
    $t := t_j + \tau_j$
    $x := \Psi^{t,t_j} x_\Delta(t_j)$
    compute the error estimate $|[\epsilon_j]|$
    $\tau := \min\left(q\tau_j, \tau_{\max}, \sqrt[p+1]{\frac{\rho\cdot\mathrm{TOL}}{|[\epsilon_j]|}}\tau_j\right)$
    *if* $(|[\epsilon_j]| \le \mathrm{TOL})$ *then*        (* step is accepted *)
        $t_{j+1} := t$
        $\Delta := \Delta \cup \{t_{j+1}\}$
        $x_\Delta(t_{j+1}) := x$
        $\tau_{j+1} := \min(\tau, T - t_{j+1})$
        $j := j + 1$
    *else*                                   (* step is rejected *)
        $\tau_j := \tau$
    *end if*
*end while*
```

Note that the **while** loop of the algorithm need never terminate. The behavior of good algorithms, described in Examples 2.12 (blowup), 2.13, and 2.14 (collapse), to "grind to a halt" at the endpoint t_+ of the existence interval of the solution, means exactly that for an input $T \ge t_+$ we expect an infinite loop with $t_j \to t_+$, $\tau_j \to 0$. On the other hand, we should note that, of course, an adaptive algorithm, as sketched here, may simply "overlook" a local singularity, although this is more likely to be the case for equidistant meshes.

Scaling. So far we have applied *norms* to the discretization errors without specifying them more closely. Yet the choice of norm plays a very decisive role in the operation of the overall algorithm. For adaptive algorithms it is recommended to use *smooth* norms—e.g., the Euclidean—for measuring the errors. (In fact, for instance, the maximum may cause "back-and-forth jumps" in the error components representing the maximum and thereby produce a "non-smooth" course of the step control). Moreover, it is strongly recommended that one pay detailed attention to the *scaling* of the components of the vectors under the norms. When such a scaling is prescribed from the outside, i.e., by the user, it is called *external* scaling. For safety's sake, any reasonably efficient integration program should also incorporate some *internal* scaling. Formally, any scaling is specified by some positive diagonal matrix

$$D_j = \mathrm{diag}\,(\sigma_1(t_j), \ldots, \sigma_d(t_j)).$$

Now instead of the above error norms $|[\epsilon_j]|$, the scaled error norms $|D_j^{-1}[\epsilon_j]|$ are used at all appropriate places in the basic adaptive algorithm. Typical scaling techniques include *relative* scaling $\sigma_k(t_j) = |(x_\Delta(t_j))_k|$ and *abso-*

lute scaling $\sigma_k(t_j) = \sigma_k^*$ with fixed values $\sigma_k^* > 0$. In addition, it is highly desired in programs that program-internal scalings be invariant under external rescalings. Since the relative error concept is well known to break down near zero (see, e.g., [58, Chapter 2]), it is usual to require the user to provide, besides the relative error tolerance TOL, an absolute lower bound $s_{\min} > 0$ such that

$$\sigma_k(t_j) = \max(s_{\min}, |(x_\Delta(t_j))_k|), \qquad k = 1, \ldots, d.$$

Simultaneous Order Control. In the theory of numerical quadrature the adaptive Romberg method (see, e.g., [58, Section 9.5.3]) provides an example for the simultaneous control of orders and step sizes. The techniques developed there carry over directly to the extrapolation methods for differential equations discussed in Section 4.3. Accordingly, it suffices to sketch here only the principal points.

For a given number of columns k, an extrapolation method computes successive discrete evolutions

$$\begin{tikzcd}
\Psi_{11} \arrow[rd] & & & \\
\Psi_{21} \arrow[r] & \Psi_{22} & & \\
\vdots & & \ddots & \\
\Psi_{k-1,1} \arrow[r] \arrow[rd] & \cdots \arrow[r] \arrow[rd] & \Psi_{k-1,k-1} \arrow[rd] & \\
\Psi_{k,1} \arrow[r] & \cdots \arrow[r] & \Psi_{k,k-1} \arrow[r] & \Psi_{k,k}
\end{tikzcd}$$

where in the case of the extrapolated *explicit midpoint rule* (see Section 4.3.3), $\Psi_{\nu\mu}$ has order 2μ. The details of computing $\Psi_{\nu\mu}$ for a given subdivision sequence

$$\mathcal{F} = \{2n_1, 2n_2, \ldots\}$$

are shown in Algorithm 4.45. The *cost* of this computation—measured by the number of f evaluations—corresponds to the stage count s_μ of the Runge-Kutta method belonging to the discrete evolution $\Psi_{\mu\mu}$. This leads us to the *cost sequence*

$$\mathcal{A} = \{s_1, s_2, \ldots\},$$

which depends on the sequence $\mathcal{F}$ and, for the explicit midpoint rule, is calculated by the relation (4.28). As in the case of the adaptive Romberg quadrature, the basic idea of order control is a *minimization of the cost per step*. For an explanation of this minimization principle, recall that for any diagonal, discrete evolution $\Psi_{\mu\mu}$, $\mu = 2, \ldots, k$, a corresponding step-size prediction τ_μ^* is given by the formula (5.5). We now determine the *optimal* column index k^* and the corresponding size τ^* of the next step such that

$$\frac{s_{k^*}}{\tau^*} = \min_{2 \le \mu \le k} \frac{s_\mu}{\tau_\mu^*}$$

holds. For further algorithmic details, such as *order windows* or the selection of certain parameters by means of Shannon's information theory, we refer to [58] or to the original articles [49, 51] on which the program DIFEX1 is based. A different approach to determining the parameters is described in [91] and is implemented in the program ODEX.

Remark 5.3 In comparison with the best explicit Runge-Kutta methods of fixed order, explicit extrapolation methods with the same fixed order are not particularly economical. However, due to the above-sketched additional facility for the dynamic adaptation of the order to a particular problem, they turn out to be rather competitive for numerous problems and wide ranges of accuracies. For special cases, such as second-order differential equations without first derivatives, they even represent, among the one-step methods, the methods of choice (see Section 4.3.4 or [51]).

5.2 Control-Theoretic Analysis

A deeper insight into the adaptive mechanism of step-size control can be gained by studying it from the viewpoint of control theory. There the problem is interpreted as a comparison of an *observed value* $\|[\epsilon_{j+1}]\|$ and a *set value* TOL. In response to the input signal $\tau_j = t_{j+1} - t_j$, the observed value $\|[\epsilon_{j+1}]\|$, i.e., the estimated error at the next integration point t_{j+1}, is the output signal of the entire system consisting of the *one-step method plus the problem* as specified by the discrete evolution Ψ. In control theory, the term *optimal control* refers to the problem of determining the input for which the system output agrees exactly with the set value. In general, this input is not accessible, since the system response is incompletely known and may also be subject to additional perturbations. In that case, one turns to an *adaptive control* (or feedback control) involving an *iterative* process that adjusts the system output to the set value. The process terminates when observed value and set value are in sufficient agreement. This reflects our *step-size control* process.

5.2.1 *Excursion to PID Controllers*

We turn now to a mathematical description of our control problem. Let the system be defined by the mapping

$$F : \mathbf{R} \to \mathbf{R}$$

and choose $F = 0$ as the set value. This choice has the advantage that the observed value always equals the deviation from the set value. Thus, we are looking for an input ξ_* such that

$$F(\xi_*) = 0.$$

With this we are in a mathematically familiar field and can apply, in principle, any iterative methods for the determination of zeros of F. Instead, we will look at approaches of *linear* control theory. For this we begin with a good linear *model* F_0 of F such that

$$F(\xi) \approx F_0(\xi) = \alpha(\xi - \xi_0)$$

for ξ in some domain of interest. As in backward error analysis (see, e.g., [58, Chapter 2]), we consider the original system F as the system F_0 with a perturbed input:

$$F(\xi) = F_0(\xi + \delta).$$

Of course, in general, the unknown perturbation δ will depend on ξ, and this perturbation must now be "controlled away" iteratively. In the unperturbed case the correct input would be

$$\xi = \xi_0.$$

This value is changed iteratively to

$$\xi_k = \xi_0 - \delta_k$$

such that, if possible, $\delta_k \approx \delta$ is an approximate compensation of the perturbation δ. In the case of the fairly general type of *PID controller* one chooses

$$\delta_k = \beta_P F(\xi_{k-1}) + \beta_I \sum_{j=0}^{k-1} F(\xi_j) + \beta_D \nabla F(\xi_{k-1}),$$

where

$$\nabla F(\xi_k) = F(\xi_k) - F(\xi_{k-1})$$

denotes the backward difference operator. Moreover, it is agreed to use $F(\xi_j) = 0$ for $j < 0$, whence $\delta_0 = 0$ holds consistently.

Hence, a PID controller reacts with a correction that consists of a sum of three (control) terms, namely:

- the P term, proportional to the latest deviation from the set value ("P" for proportional);
- the I term, proportional to the sum of all prior deviations from the set value (in the case of time-continuous control an integral, thus "I" for integral);
- the D term, proportional to the last change of the differences ("D" for difference).

With this it is hoped that the perturbation δ has been sampled well enough to allow us to compensate for it.

As an iterative method for solving $F(\xi_*) = 0$, the PID controller looks rather strange, at first. Accordingly, we introduce the difference $\xi_{k+1} - \xi_k = \nabla\xi_{k+1}$ and then rewrite the iteration in the form

$$\xi_{k+1} = \xi_k - \nabla\xi_{k+1} = \xi_k - \beta_I F(\xi_k) - \beta_P \nabla F(\xi_k) - \beta_D \nabla^2 F(\xi_k). \quad (5.6)$$

It is not easy to assess its convergence for nonlinear F. Therefore, we restrict ourselves at first to *linear* systems

$$F(\xi) = \alpha(\xi - \xi_*),$$

or, equivalently, to constant-input perturbations $\delta = \xi_0 - \xi_*$. Then the difference

$$\eta_k = \xi_k - \xi_*,$$

satisfies the homogeneous linear difference equation of third order

$$\eta_{k+1} = \eta_k - \beta_I \alpha \eta_k - \beta_P \alpha \nabla \eta_k - \beta_D \alpha \nabla^2 \eta_k. \quad (5.7)$$

Hence the question about the convergence $\eta_k \to 0$ reduces to the question about the *asymptotic stability* of the difference equation (5.7). The answer has already been given by Theorem 3.40: We have convergence $\eta_k \to 0$ if and only if all roots λ of the characteristic equation

$$\lambda^3 = \lambda^2 - \beta_I \alpha \lambda^2 - \beta_P \alpha \nabla \lambda^2 - \beta_D \alpha \nabla^2 \lambda^2 \quad (5.8)$$

satisfy the condition $|\lambda| < 1$ where $\nabla\lambda^2 = \lambda^2 - \lambda$ and $\nabla^2\lambda^2 = \lambda^2 - 2\lambda + 1$. In this case the PID controller is called *stable for* α.

Now we can also go a step further and consider the nonlinear case, since all essential preparations have already been provided in Section 3.3.

Theorem 5.4 *Let $F : \mathbf{R} \to \mathbf{R}$ be continuously differentiable and satisfy*

$$F(\xi_*) = 0, \qquad \alpha_* = F'(\xi_*) \neq 0.$$

Moreover, let the parameters $(\beta_P, \beta_I, \beta_D)$ be chosen such that the PID controller is stable for α_*. Then with the choice $F(\xi_{-2}) = F(\xi_{-1}) = 0$ the iteration

$$\xi_{k+1} = \xi_k - \beta_I F(\xi_k) - \beta_P \nabla F(\xi_k) - \beta_D \nabla^2 F(\xi_k), \qquad k = 0, 1, 2, \ldots,$$

of the PID controller converges to ξ_* for any ξ_0 sufficiently close to ξ_*.

Proof. The trick that led us in Example 3.39 from Theorem 3.33 to Theorem 3.40 for linear models also allows for a proof of the stated result for nonlinear models by means of Theorem 3.38. We leave the details to the reader but point out the paradigm once more: The linear stability result $\xi_k \to \xi_*$, which holds for all ξ_0, applies in the nonlinear case for all starting values ξ_0 in some neighborhood of the fixed point ξ_* of the iteration. By setting $F(\xi_{-2}) = F(\xi_{-1}) = 0$ we ensure precisely that ξ_* is a fixed point of the PID iteration. □

Thus a PID controller can be applied in the case of "small" nonlinear deviations, provided that it is suitably dimensioned. The choice of the parameters proceeds along the following lines:

- The linear model F_0 is assumed to be good in the sense that
$$\alpha_* \approx \alpha. \tag{5.9}$$
- Then the parameters $(\beta_P, \beta_I, \beta_D)$ are chosen such that the PID controller is stable for α. If the parameters are not pushed to the border of feasibility, then for a good approximation (5.9) the PID controller will also be stable for α_*.

An indication for the selection of the parameters is given by the following result.

Lemma 5.5 *For any β_I satisfying*
$$|1 - \alpha\beta_I| < 1, \tag{5.10}$$
there exists a $\beta_0 = \beta_0(\beta_I, \alpha) > 0$ such that with
$$|\beta_P|, |\beta_D| \le \beta_0$$
the PID controller is stable for α_*.

Proof. For $\beta_P = \beta_D = 0$ the characteristic equation (5.8) becomes
$$\lambda^3 = (1 - \beta_I\alpha)\lambda^2.$$
It has the roots $\lambda_0 = 0$ and $\lambda_1 = (1 - \beta_I\alpha)$, and hence (5.10) is exactly the stability condition. Since the roots of a polynomial depend continuously on the coefficients, it follows that for fixed β_I, the parameters $|\beta_P|$ and $|\beta_D|$ can be moved slightly away from zero without any root reaching the unit circle. □

5.2.2 *Step-size Selection as Controller*

We now apply the above stability criteria for controllers to the step-size control algorithm. For this we introduce first a termination criterion for the process of matching observed values to set values: With a safety factor $\rho < 1$, the user parameter TOL is transformed into the *set value* $\rho \cdot \mathrm{TOL}$ and then, as before, we terminate with $|[\epsilon_{k+1}]| \le \mathrm{TOL}$. In other words, we attempt to achieve somewhat more with the controller but are satisfied with the original acceptance condition. The index k in the error estimates $|[\epsilon_{k+1}]|$ and the step sizes τ_k represents here the iteration index of the controller iteration and not the index j of the corresponding time interval $[t_j, t_j + \tau_k]$.

As discussed above, the control-theoretic considerations require a suitable linear model of the system
$$\tau \mapsto |[\epsilon]|.$$
In the derivation of the basic adaptive algorithm the nonlinear model
$$|[\epsilon]| \approx c\tau^{p+1} \tag{5.11}$$
was used. In order to construct from this a linear model it may appear natural to consider a Taylor expansion, but that would have validity only for a very limited domain of τ. A more suitable model is obtained by applying logarithms,
$$\log \frac{|[\epsilon]|}{\rho \cdot \mathrm{TOL}} \approx (p+1) \log \frac{\tau}{\tau_{\mathrm{ref}}},$$
which at the same time transforms the set value to zero. Here τ_{ref} is defined via the unknown constant c. By means of the transformation
$$\xi = \log \frac{\tau}{\tau_{\mathrm{ref}}} \quad \mapsto \quad F(\xi) = \log \frac{|[\epsilon]|}{\rho \cdot \mathrm{TOL}}$$
this model F_0 is now linear and given by
$$F_0(\xi) = (p+1)\xi.$$
Then the iteration (5.6) of the PID controller has the form
$$\log \frac{\tau_{k+1}}{\tau_{\mathrm{ref}}} = \log \frac{\tau_k}{\tau_{\mathrm{ref}}} - \beta_I \log \frac{|[\epsilon_{k+1}]|}{\rho \cdot \mathrm{TOL}} - \beta_P \nabla \log \frac{|[\epsilon_{k+1}]|}{\rho \cdot \mathrm{TOL}} - \beta_D \nabla^2 \log \frac{|[\epsilon_{k+1}]|}{\rho \cdot \mathrm{TOL}},$$
or, after some rearrangements,
$$\tau_{k+1} = \left(\frac{\rho \cdot \mathrm{TOL}}{|[\epsilon_{k+1}]|}\right)^{\beta_I + \beta_P + \beta_D} \left(\frac{|[\epsilon_k]|}{\rho \cdot \mathrm{TOL}}\right)^{\beta_P + 2\beta_D} \left(\frac{\rho \cdot \mathrm{TOL}}{|[\epsilon_{k-1}]|}\right)^{\beta_D} \tau_k.$$
In line with our earlier conventions for PID controllers we set $|[\epsilon_0]| = |[\epsilon_{-1}]| = \rho \cdot \mathrm{TOL}$.

Now by Theorem 5.4 and Lemma 5.5 the choice of the constants is
$$|1 - (p+1)\beta_I| < 1, \qquad \beta_P, \beta_D \approx 0.$$
At this point, nothing speaks against the selection of a pure I controller, i.e., against setting $\beta_P = \beta_D = 0$. However, in Remark 6.12 we will encounter

a situation where indeed a PID controller is preferable. In the case of an I controller and with $\beta_I = 1/\gamma$ we obtain

$$\tau_{k+1} = \sqrt[\gamma]{\frac{\rho \cdot \mathrm{TOL}}{|[\epsilon_{k+1}]|}}\,\tau_k,$$

where the parameter has to satisfy

$$\gamma > \frac{p+1}{2}. \tag{5.12}$$

This is exactly the step-size formula of the basic adaptive Algorithm 5.2 except for the increased flexibility in the exponent of the root. In particular, Theorem 5.4 ensures that we obtain a convergent step-size control, that provided γ is not too large or too close to $(p+1)/2$, depending on the quality of the model (5.11). The choice $\gamma = p + 1$ derived in the previous section avoids the upper boundary but, from the viewpoint of this section, represents by no means a stringent requirement!

Example 5.6 We consider the behavior of the I controller for the step-size control in the case of the three-body problem of Example 5.1. For this we use a method with

$$p + 1 = 4$$

and a relatively small tolerance $\mathrm{TOL} = 4.0 \cdot 10^{-9}$, which corresponds to the accuracy used in Example 5.1. Figure 5.1 shows the numbers N_{Accept} and N_{Reject} of the accepted and rejected steps, respectively, in dependence on γ. The boundary

$$\gamma \approx \frac{p+1}{2} = 2$$

of the convergence domain is clearly visible. Moreover, we see that the canonical parameter choice $\gamma = p + 1$ need not be best possible.

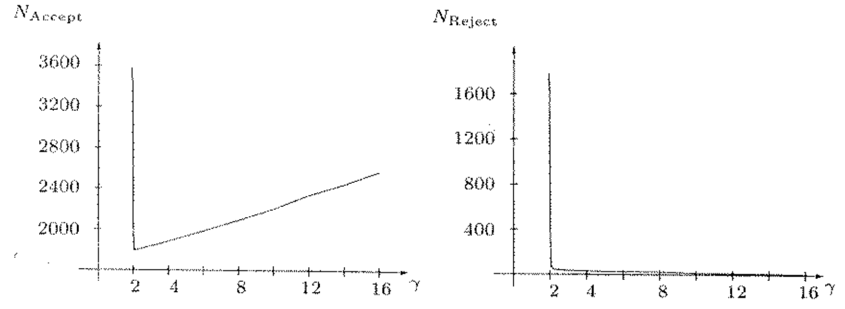


FIGURE 5.1. Step-size control in dependence on the root exponent γ.

As an upshot of the discussion in this section we focus explicitly on two observations:

- The root exponent $\gamma = p + 1$ in the step-size formula of the basic adaptive Algorithm 5.2 is *not a sensitive* choice; in fact, the algorithm implements an I controller that works reasonably well for a large range of parameters.
- The order p of a one-step method depends on the differentiability class of the right side f of the differential equation. Typically in a program, the step-size control is set for the order $p_{\max}$ of the method for $f \in C^\infty(\Omega, \mathbf{R}^d)$. On the other hand, by Theorem 4.24 we know that Runge-Kutta methods have for $f \in C^m(\Omega, \mathbf{R}^d)$ the order

  $$p = \min(p_{\max}, m).$$

  Thus the parameter choice

  $$\gamma = p_{\max} + 1 \geq p + 1$$

  satisfies the stability condition (5.12) *independently* of the differentiability class of the right side f.

Remark 5.7 The interpretation of step-size control in terms of I controllers goes back to the article [87] (see also [94]) by K. Gustafsson, M. Lundh, and G. Söderlind of the year 1988. In contrast to the presentation chosen here, these authors consider the controller in the time-continuous case and therefore obtain no conditions for the parameters β_P, β_I, β_D.

5.3 Error Estimation

As discussed in Section 5.1, the implementation of a step-size control requires a dependable *estimate of the local discretization error*. Such an estimate can be obtained by comparing two discrete evolutions $(\Psi, \hat{\Psi})$ that have been computed side by side. Let the local errors (consistency errors) be denoted by

$$\epsilon = \Phi^{t+\tau,t}x - \Psi^{t+\tau,t}x$$

and

$$\hat{\epsilon} = \Phi^{t+\tau,t}x - \hat{\Psi}^{t+\tau,t}x,$$

respectively, and assume that Ψ is the more accurate of these discrete evolutions and hence that

$$\theta = \frac{|\epsilon|}{|\hat{\epsilon}|} < 1. \tag{5.13}$$

The difference of the discrete evolutions

$$[\hat{\epsilon}] = \Psi^{t+\tau,t}x - \hat{\Psi}^{t+\tau,t}x = \hat{\epsilon} - \epsilon$$

is an *estimate of the less accurate local error* $\hat{\epsilon}$, since (5.13) implies that the corresponding relative error satisfies

$$\frac{|\hat{\epsilon} - [\epsilon]|}{|\hat{\epsilon}|} = \theta < 1 .$$

From this it follows, by the triangle inequality, that

$$\frac{1}{1+\theta}|[\hat{\epsilon}]| \leq |\hat{\epsilon}| \leq \frac{1}{1-\theta}|[\hat{\epsilon}]|, \qquad (5.14)$$

which is the type of inequality required as an error estimator for adaptive Romberg quadrature (see, e.g., [58, Section 9.5.2]). If, in addition, $\theta \to 0$ holds for $\tau \to 0$, then this error estimator is called *asymptotically exact*. Evidently, this will be the case when

$$|\hat{\epsilon}| = \hat{c}\tau^{\hat{p}+1} + O(\tau^{\hat{p}+2}), \quad \hat{c} \neq 0, \qquad \text{and} \qquad |\epsilon| = O(\tau^{\hat{p}+2}),$$

and hence when Ψ is not only more accurate but has a *genuinely* higher order than $\hat{\Psi}$.

This illuminates a *dilemma*: We estimate the local error for $\hat{\Psi}$, and hence the step-size predictions of the basic Algorithm 5.2 refer to $\hat{\Psi}$. On the other hand, the error estimate is useful exactly when the solution given by Ψ is more accurate. Thus the algorithm should really continue to compute with the more accurate solution

$$x_\Delta(t_{j+1}) = \Psi^{t_{j+1},t_j} x_\Delta(t_j), \qquad (5.15)$$

where $\tau_j = t_{j+1} - t_j$ is the step size determined by means of $\hat{\Psi}$. This way out of the dilemma can be justified as follows:

During the *step-size correction* ($|[\hat{\epsilon}]| > \mathrm{TOL}$) we work only with $\hat{\Psi}$, since the use of Ψ for the mesh function x_Δ is restricted solely to the initial value $x_\Delta(t_j)$ common to the corrector loop. Hence we are in the stable control loop analyzed in the previous section.

The *step-size prediction* ($|[\hat{\epsilon}]| \leq \mathrm{TOL}$) will be on the safe side if only the discrete evolution Ψ is so much more accurate than $\hat{\Psi}$ that

$$\theta \leq \tfrac{1}{2}.$$

In fact, by (5.14) and (5.13), this implies the error estimate

$$|\epsilon_j| \leq \frac{\theta}{1-\theta}|[\hat{\epsilon}_j]| \leq |[\hat{\epsilon}_j]| \qquad (5.16)$$

and a step-size prediction τ_j with

$$\tau_j = \sqrt[\hat{p}+1]{\frac{\rho \cdot \mathrm{TOL}}{|[\hat{\epsilon}_j]|}}\,\tau_{j-1} \leq \sqrt[\hat{p}+1]{\frac{\rho \cdot \mathrm{TOL}}{|\epsilon_j|}}\,\tau_{j-1} = \tau_j^* .$$

But by Section 5.2.1, even τ_j^* would already be a justifiable step-size prediction for the discrete evolution Ψ if only the root exponent $\gamma = \hat{p} + 1$ satisfied the condition (5.12), i.e.,

$$\hat{p} < p \leq 2\hat{p}.$$

This condition holds for all pairs $(\Psi, \hat{\Psi})$ discussed later.

Remark 5.8 Since we determine the mesh function x_Δ from the discrete evolution Ψ, the actually desired but unimplementable local error control would be

$$|\epsilon_j| \leq \mathrm{TOL} . \qquad (5.17)$$

The condition (5.16) indicates that this desired control is guaranteed by the implemented control

$$|[\hat{\epsilon}_j]| \leq \mathrm{TOL}$$

if only $\theta \leq \frac{1}{2}$, i.e., the discrete evolution Ψ gains at least *one bit* of information over the discrete evolution $\hat{\Psi}$, and this even though the implemented criterion originally refers to $\hat{\Psi}$. On the other hand, (5.16) shows that for very small θ the desired control (5.17) is *more than satisfied*. Hence one computes more accurately than required and therefore expends more effort than needed. Although we have $\theta = O(\tau)$ for step sizes $\tau \to 0$, we should take care, for cost reasons, that in the construction of the method the method-dependent parts of the constants do not become too small.

Remark 5.9 Historically, the continuation of the computation by means of (5.15) was slow in being accepted. Older programs continued with

$$x_\Delta(t_{j+1}) = \hat{\Psi}^{t_{j+1},t_j} x_\Delta(t_j),$$

since it was taken very literally that the step-size prediction $\tau_j = t_{j+1} - t_j$ referred to $\hat{\Psi}$. This applies, for instance, to the early popular methods of E. Fehlberg [70, 71] dating to the 1960s. With this (actually outdated) difference of approaches in mind, the following *terminology* is used: Runge-Kutta methods that continue the computation with order p are denoted by RK$p(\hat{p})$, while those using the lower-order $\hat{p}$ are identified by RK$\hat{p}(p)$. If it is clear that the higher order is selected, then the notation is just RK$p\hat{p}$.

Example 5.10 *Subdiagonal Error Estimation for Extrapolation Methods.* For the error estimation of the discrete evolution $\Psi_{\mu\mu}$ of the extrapolated explicit midpoint rule we choose a partner of lower order in the already computed tableau of discrete evolutions. In view of Remark 5.8, we should identify here the discrete evolution with the highest accuracy below that of $\Psi_{\mu\mu}$. The relevant highest order belongs to exactly two discrete evolutions, namely, the *subdiagonal* entry $\Psi_{\mu,\mu-1}$ and the *diagonal* entry $\Psi_{\mu-1,\mu-1}$.

When we compare their leading local error terms by means of Lemma 4.47, we obtain

$$\begin{aligned}\Phi^{t+\tau,t}x - \Psi^{t+\tau,t}_{\mu,\mu-1}x &= \frac{\eta_{\mu-1}(t)}{n_\mu^2\cdots n_2^2}\tau^{2\mu-1} + O(\tau^{2\mu})\\ &= \frac{n_1^2}{n_\mu^2}\frac{\eta_{\mu-1}(t)}{n_{\mu-1}^2\cdots n_1^2}\tau^{2\mu-1} + O(\tau^{2\mu})\\ &= \frac{n_1^2}{n_\mu^2}\left(\Phi^{t+\tau,t}x - \Psi^{t+\tau,t}_{\mu-1,\mu-1}x\right) + O(\tau^{2\mu}).\end{aligned}$$

Thus because of $n_1^2/n_\mu^2 < 1$ the choice will be the more accurate subdiagonal entry. With this we arrive, as for adaptive Romberg quadrature (see again [58]), at the *subdiagonal error estimator*

$$[\hat{\epsilon}_{\mu-1}] = \Psi^{t+\tau,t}_{\mu,\mu-1}x - \Psi^{t+\tau,t}_{\mu,\mu}x.$$

This estimator is especially easy to implement, since in a row-wise evaluation of the extrapolation tableau the required difference occurs in any case. It requires no further memory location and also avoids any potential cancellation of digits due to explicit subtraction.

Example 5.11 *Extrapolation with Runge-Kutta Methods of Fixed Order.* For a given Runge-Kutta method $(\hat{b}, \hat{\mathcal{A}})$ of order $\hat{p}$ the extrapolation technique allows for an easy construction of a discrete evolution of higher order. With the subdivision sequence $\mathcal{F} = \{1, 2\}$ the discrete evolution Ψ generated by extrapolation of order one is obtained as follows: Determine the coefficients $\alpha_* = \Psi^{t+\tau,t}x$ and α_0 of the polynomial $\chi(\tau) = \alpha_* + \alpha_0\tau^p$ specified by the interpolation conditions

$$\chi(\tau) = \hat{\Psi}^{t+\tau,t}x, \qquad \chi(\tau/2) = \hat{\Psi}^{t+\tau,t+\tau/2}\hat{\Psi}^{t+\tau/2,t}x.$$

It is easily verified that

$$\Psi^{t+\tau,t}x = \alpha_* = \chi(\tau/2) + \frac{\chi(\tau/2) - \chi(\tau)}{2^{\hat{p}} - 1}.$$

By Theorem 4.36 we know that for right sides $f \in C^{\hat{p}+1}(\Omega, \mathbf{R}^d)$ the discrete evolution Ψ has order $p = \hat{p} + 1$. Let $\hat{s}$ be the stage count of the Runge-Kutta method $(\hat{b}, \hat{\mathcal{A}})$. Then the method $(b, \mathcal{A})$ belonging to Ψ has the stage count

$$s = 3\hat{s} - 1.$$

This possibility for associating a step-size control with a given Runge-Kutta method is very easy to program. But it does not go easy on the number of f evaluations per step, as the given stage count s shows.

5.4 Embedded Runge-Kutta Methods

An approach for *optimizing* specifically the total number of f evaluations for the pair $(\Psi, \hat{\Psi})$ consists in constructing *embedded Runge-Kutta methods.* In that case, Ψ and $\hat{\Psi}$ are chosen to be of the form $(b, \mathcal{A})$ and $(\hat{b}, \mathcal{A})$, respectively; i.e., the discrete evolutions use the same coefficient matrix $\mathcal{A}$ and hence the same f evaluations. In the form of a Butcher array an embedded method is written as

$$\begin{array}{c|c} c & \mathcal{A} \\ \hline & b^T \\ \hline & \hat{b}^T \end{array}$$

Of course, the minimal stage count needed for the implementation of an embedded method of the type RK$p(p-1)$ is larger than the optimal stage count given in Table 4.5 for a separate method of order p.

Example 5.12 Consider the classical Runge-Kutta method $(b, \mathcal{A})$ of order 4 with the optimal stage count $s = 4$ given in Table 4.2. In order to generate from this an embedded method of type RK4(3) we need coefficients $\hat{b} = (\hat{b}_1, \hat{b}_2, \hat{b}_3, \hat{b}_4)^T$ such that the Runge-Kutta method $(\hat{b}, \mathcal{A})$ satisfies, by Theorem 4.18, the four conditions

$$\begin{aligned}\hat{b}_1 + \hat{b}_2 + \hat{b}_3 + \hat{b}_4 &= 1,\\ \tfrac{1}{2}\hat{b}_2 + \tfrac{1}{2}\hat{b}_3 + \hat{b}_4 &= \tfrac{1}{2},\\ \tfrac{1}{4}\hat{b}_2 + \tfrac{1}{4}\hat{b}_3 + \hat{b}_4 &= \tfrac{1}{3},\\ \tfrac{1}{4}\hat{b}_3 + \tfrac{1}{2}\hat{b}_4 &= \tfrac{1}{6},\end{aligned}$$

for order 3. Here we substituted into the equations (4.11) for stage 4 the coefficient matrix $\mathcal{A}$ of Table 4.2. This system of equations has the *unique* solution $\hat{b} = b = (\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6})^T$, which leads to $\Psi = \hat{\Psi}$ and hence does not constitute a meaningful embedded method; i.e., $s = 4$ is not enough.

The total cost of the necessary increase of the stage count can be cleverly compensated by means of the so-called *Fehlberg trick.* E. Fehlberg [71] had the idea of saving *effectively* the f evaluation for the last stage k_s of an s-stage method by letting this stage be the same as the first stage k_1^* of the next step. Because of

$$k_s = f\Big(t + c_s\tau, x + \tau\sum_{j=1}^{s-1} a_{sj}k_j\Big)$$

and

$$k_1^* = f(t+\tau, \Psi^{t+\tau,t}x) = f\Big(t + \tau, x + \tau\sum_{j=1}^{s} b_jk_j\Big),$$

this is the case, for all right sides f, if the coefficients satisfy the conditions

$$c_s = 1, \qquad b_s = 0, \qquad a_{sj} = b_j \quad \text{for} \quad j = 1, \ldots, s-1. \tag{5.18}$$

Here the consistency condition $\sum_j b_j = 1$ ensures that the condition $c_s = \sum_j a_{sj}$ remains valid, which is critical for autonomization invariance. A look at the total effort in f evaluations for such a method shows that n steps of the s-stage method instead of $n \cdot s$ now cost only $n \cdot (s-1) + 1$ f evaluations. Accordingly, we speak of an *effectively* $(s-1)$*-stage* method.

Example 5.13 We continue with the construction of a method of type RK4(3) built upon the classical Runge-Kutta method of Table 4.2. With 5 stages we consider a Butcher array of the form

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{1}{2} & \frac{1}{2} & & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & & \\
1 & 0 & 0 & 1 & & \\
c_5 & a_{51} & a_{52} & a_{53} & a_{54} & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & 0 \\
\hline
 & \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \hat{b}_4 & \hat{b}_5
\end{array}$$

Then, by applying the Fehlberg trick (5.18) we obtain the coefficients

$$c_5 = 1, \qquad a_{51} = \tfrac{1}{6}, \qquad a_{52} = \tfrac{1}{3}, \qquad a_{53} = \tfrac{1}{3}, \qquad a_{54} = \tfrac{1}{6}.$$

Hence, it remains only to determine $\hat{b}$ from the equations

$$\begin{aligned}
\hat{b}_1 + \hat{b}_2 + \hat{b}_3 + \hat{b}_4 + \hat{b}_5 &= 1 \\
\tfrac{1}{2}\hat{b}_2 + \tfrac{1}{2}\hat{b}_3 + \hat{b}_4 + \hat{b}_5 &= \tfrac{1}{2} \\
\tfrac{1}{4}\hat{b}_2/4 + \tfrac{1}{4}\hat{b}_3 + \hat{b}_4 + \hat{b}_5 &= \tfrac{1}{3} \\
\tfrac{1}{4}\hat{b}_3 + \tfrac{1}{2}\hat{b}_4 + \tfrac{1}{2}\hat{b}_5 &= \tfrac{1}{6},
\end{aligned}$$

which are obtained again from Theorem 4.18 by substitution of the augmented matrix $\mathcal{A}$. Since the system is invariant under interchanges of $\hat{b}_4$ and $\hat{b}_5$ it follows that together with $(b^T, 0) = (1/6, 1/3, 1/3, 1/6, 0)$

$$\hat{b}^T = (\tfrac{1}{6}, \tfrac{1}{3}, \tfrac{1}{3}, 0, \tfrac{1}{6})$$

is also a solution. The error estimator of the effectively 4-stage method of type RK4(3) has the especially simple form

$$[\hat{\epsilon}] = \tfrac{1}{6}(k_4 - k_1^*),$$

where, as above, k_1^* denotes the first stage of the next step. This simple modification of the classical Runge-Kutta method of fourth order was used in the computation of Examples 5.1 and 5.6.

The most mature embedded methods of type RK$p(p-1)$ were constructed, from 1980/81 on, by J. R. Dormand and P. J. Prince [66, 67]. Among them is an effectively 6-stage method of type RK5(4) (Table 5.1) and a 13-stage method of type RK8(7) (Table 5.2). Note that the coefficients in Table 5.2 represent only rational approximations with a relative error of $5 \cdot 10^{-18}$, which, however, fully suffices for double precision computations (eps $= 2^{-52} = 2.22 \cdot 10^{-16}$).

TABLE 5.1. Dormand-Prince coefficient set of type RK5(4).

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Example 5.14 Originally, the restricted three-body problem in Example 5.1 was solved with the classical Runge-Kutta method of order 4. Application of the two Dormand and Prince methods to this problem results in the following step counts:

- Type RK5(4): $n_\Delta = 346$ with 2329 f evaluations,
- Type RK8(7): $n_\Delta = 101$ with 1757 f evaluations.

Here the local tolerance TOL was chosen so as to achieve a spatial accuracy of $2.5 \cdot 10^{-7}$. For comparison we repeat the data of Example 5.1:

- "The" classical Runge-Kutta method of order $p = 4$ on an *equidistant* mesh used $n_\Delta = 117000$ with 468000 f evaluations,
- The method of type RK4(3) from Example 5.13 used $n_\Delta = 1883$ with 7669 f evaluations.

We discuss briefly, using the notation of Section 4.2.3, the criteria underlying the construction of the methods of Dormand and Prince. For

TABLE 5.2: Dormand–Prince coefficient–set of type RK8(7)

0													
$\frac{1}{18}$	$\frac{1}{18}$												
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$											
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$										
$\frac{5}{16}$	$\frac{5}{16}$	0	$-\frac{75}{64}$	$\frac{75}{64}$									
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$								
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$-\frac{28693883}{1125000000}$	$\frac{23124283}{1800000000}$							
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$-\frac{800635310}{3783071387}$						
$\frac{490023248}{719169821}$	$\frac{39632708}{573591083}$	0	0	$-\frac{433636366}{683701615}$	$-\frac{421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$					
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$-\frac{37695042795}{15268766246}$	$-\frac{309121744}{1061227803}$	$-\frac{12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$				
$\frac{2011146811}{2990197 98}$	$-\frac{1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{1432422823}$	$-\frac{10304129995}{1701304382}$	$-\frac{48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$-\frac{45442868181}{3398467696}$	$\frac{3065993473}{597172653}$			
1	$\frac{185892177}{718116043}$	0	0	$-\frac{3185094517}{667107341}$	$-\frac{477755414}{1098053517}$	$-\frac{703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$-\frac{4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$		
1	$\frac{403863854}{491063109}$	0	0	$-\frac{5068492393}{434740067}$	$-\frac{411421997}{543043835}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$-\frac{13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$-\frac{160528059}{685178525}$	$\frac{248638103}{1413531060}$	0	
	$\frac{14005451}{335480064}$	0	0	0	0	$-\frac{59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{561292985}{797845732}$	$-\frac{1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$-\frac{528747749}{2220607170}$	$\frac{1}{4}$
	$\frac{13451932}{455176623}$	0	0	0	0	$-\frac{808719846}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$-\frac{3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011}{667516719}$	$\frac{2}{45}$	0

$f \in C^{p+1}(\Omega, \mathbf{R}^d)$ the local errors are given by

$$\epsilon = \Phi^{t+\tau,t}x - \Psi^{t+\tau,t}x = \tau^{p+1} \sum_{\#\beta=p+1} e_\beta^{(p+1)} f^{(\beta)} + O(\tau^{p+2})$$

and

$$\begin{aligned}\hat{\epsilon} =& \Phi^{t+\tau,t}x - \hat{\Psi}^{t+\tau,t}x \\ =& \tau^p \sum_{\#\beta=p} \hat{e}_\beta^{(p)} f^{(\beta)} + \tau^{p+1} \sum_{\#\beta=p+1} \hat{e}_\beta^{(p+1)} f^{(\beta)} + O(\tau^{p+2}).\end{aligned}$$

Here, the constant coefficients $e_\beta^{(p+1)}$, $\hat{e}_\beta^{(p)}$ and $\hat{e}_\beta^{(p+1)}$ depend only on the Runge-Kutta method. Over a large problem class, the local error ϵ of the discrete evolution Ψ now turns out, on average, to improve as the problem-independent *error constant*

$$A_{p+1} = \|e^{(p+1)}\|_2 = \left(\sum_{\#\beta=p+1} |e_\beta^{(p+1)}|^2 \right)^{1/2}$$

becomes smaller. Moreover, because of $\epsilon = \hat{\epsilon} - [\hat{\epsilon}]$, a small error constant also improves the quality of the error estimate. At the same time, for the local error $\hat{\epsilon}$ of the discrete evolution $\hat{\Psi}$ the factor of τ^p should markedly dominate that of τ^{p+1}, so that we can depend on an improvement factor

$$\theta = \frac{|\epsilon|}{|\hat{\epsilon}|} < 1.$$

Independently of the problem this is described by the quantity

$$B_p = \frac{\|\hat{e}^{(p+1)}\|_2}{\|\hat{e}^{(p)}\|_2},$$

which is to be as small as possible. Finally, the error estimate

$$[\hat{\epsilon}] = \hat{\epsilon} - \epsilon = \tau^p \sum_{\#\beta=p} \hat{e}_\beta^{(p)} f^{(\beta)} + \tau^{p+1} \sum_{\#\beta=p+1} \left(\hat{e}_\beta^{(p+1)} - e_\beta^{(p+1)}\right) f^{(\beta)} + O(\tau^{p+2})$$

should be as close as possible to the model

$$\|[\hat{\epsilon}]\| = c\tau^p, \tag{5.19}$$

which means that the quantity

$$C_p = \frac{\|\hat{e}^{(p+1)} - e^{(p+1)}\|_2}{\|\hat{e}^{(p)}\|_2}$$

is also required to be as small as possible. The criteria of Dormand and Prince [67] for the construction of embedded methods of type RK$p(p-1)$ are now as follows:

1. The quantities A_{p+1}, B_p, and C_p of the method should be as small as possible.

2. The coefficients $c_i = \sum_j a_{ij}$ should satisfy $c_i \in [0,1]$ and be as distinct as possible to avoid the danger of large b_i and a_{ij} with their attendant difficulties.

3. For the error coefficients of the discrete evolution $\hat{\Psi}$, the condition $\hat{e}_\beta^{(p)} \neq 0$ should hold for all rooted trees β of order p. Otherwise, there are special right sides f for which the error estimator is much too optimistic.

Of course, taken together with the conditions for the consistency order, these criteria do not form a well-defined optimization problem but can serve only as a guideline for a computer-supported search for suitable sets of coefficients. Typically, this leads to several candidate sets that cannot be further distinguished by these criteria. The selection and assessment of appropriate sets then requires much experience and intuition.

Remark 5.15 Formerly, the construction of methods of type RK$p-1(p)$ used, instead of the first criterion, an optimization of the error constant $\hat{A}_p = \|\hat{e}^{(p)}\|_2$. But this had a negative effect on the quantity C_p, which describes the quality of the model (5.19) for the error estimation. This fact explains, in part, the poor behavior of the step-size control of these earlier methods.

Example 5.16 We take a closer look at a specific comparison of the newer Dormand and Prince methods with the older Fehlberg methods.

- The method of type RK5(4) in Table 5.1 involves

$$A_6^{\mathrm{DP}} = 3.99 \cdot 10^{-4},$$

while the popular set of Fehlberg coefficients [70] of type RK5(4) has the larger error constant

$$A_6^{\mathrm{F}} = 3.36 \cdot 10^{-3} = 8.42 \cdot A_6^{\mathrm{DP}}.$$

- The method of type RK8(7) in Table 5.2 uses the quantities

$$A_9^{\mathrm{DP}} = 4.51 \cdot 10^{-6}, \qquad B_8^{\mathrm{DP}} = 2.24, \qquad C_8^{\mathrm{DP}} = 2.36,$$

in contrast to

$$A_9^{\mathrm{F}} = 1.09 \cdot 10^{-5}, \qquad B_8^{\mathrm{F}} = 4.29, \qquad C_8^{\mathrm{F}} = 4.47$$

for the Fehlberg coefficient set [70] of type RK8(7). This shows clearly the larger value of C_8. Besides this, the Fehlberg coefficients violate the third criterion, since

$$\hat{e}_\beta^{(p)} = 0 \qquad \text{for the rooted tree} \quad \beta = [\odot, \ldots, \odot], \quad \#\beta = p.$$

Since this rooted tree is "relevant" for quadrature problems (see Exercise 4.6), the method is expected to have an overly optimistic error estimator for these problems. In fact, the situation is more dramatic: *On principle*, the Fehlberg coefficient set delivers for quadrature problems the error estimate $[\hat{\epsilon}] = 0$. This blemish of the otherwise very successful Fehlberg method was ultimately the reason for the inclusion of the third item in the catalogue of criteria.

5.5 Local Versus Achieved Accuracy

In Section 5.1, based on algorithmic considerations, we replaced the control of the global discretization error by that of the local error contribution. In this section we look at the consequences of this approach by considering the actually achieved accuracy $\epsilon_\Delta(T)$ at the end $T = t_n$, $n = n_\Delta$ of the integration interval.

By using (5.1) we obtain for the actual discretization, in linear approximation, the recursive relation

$$\epsilon_\Delta(t_{j+1}) \doteq W(t_{j+1}, t_j)\epsilon_\Delta(t_j) + \epsilon_{j+1}.$$

Without input error $\epsilon_\Delta(t_0) = 0$ this gives the representation

$$\epsilon_\Delta(T) \doteq \sum_{j=1}^{n} W(T, t_j)\epsilon_j.$$

Here we have used that by Lemma 3.2 the propagation matrices $W(t,s)$ form an evolution. With the *pointwise* condition numbers $\kappa_j(t) = \|W(t,t_j)\|$, defined in Section 3.1.2 we find—working with the local error control (5.2) the approximate bound

$$|\epsilon_\Delta(T)| \dot{\leq} \mathrm{TOL} \sum_{j=1}^{n} \kappa_j(T). \tag{5.20}$$

For *strictly dissipative* systems (see Exercise 6.6), in which all local errors are damped out by the dynamics of the system itself, it turns out that in essence, only the last term $\kappa_n(T) = 1$ plays a role and hence that

$$|\epsilon_\Delta(T)| \approx \mathrm{TOL}.$$

Hence, for such systems a local accuracy control will suffice entirely. For nondissipative systems the situation is more complicated. In particular, there is not simply a proportionality between the global error $|\epsilon_\Delta(T)|$ and TOL, for as a rule, a reduction of TOL increases the number of terms in the formula (5.20). In order to obtain an analyzable analytical model for adaptive meshes it is frequently assumed that

$$\kappa_j(t_{j+1}) \le \kappa, \qquad \kappa \ge 1.$$

This model assumes that the step-size control condenses the mesh points exactly in those parts where the local error growth is largest and thins them out where only low error growth occurs. This assumption, therefore, reflects, at least to first order, the structure of our earlier-discussed step-size control. The evolution properties of W imply that

$$\kappa_j(T) \le \kappa_{n-1}(t_n) \cdot \kappa_{n-2}(t_{n-1}) \cdots \kappa_j(t_{j+1}) \le \kappa^{n-j},$$

which for (5.20) gives the upper bound

$$|\epsilon_\Delta(T)| \dot{\le} \frac{\kappa^n - 1}{\kappa - 1}\,\mathrm{TOL}.$$

In the limit $\kappa \searrow 1$ this reduces to the relation

$$|\epsilon_\Delta(T)| \approx n_\Delta \cdot \mathrm{TOL}.$$

This formula is often used by practitioners, without considering its derivation, as a rough guide for assessing, at least in some sense, the influence of the global growth of the local discretization errors. Again we cannot assume here a proportionality between the achieved global error and the prescribed local tolerance TOL: A reduction of TOL increases, in general, the number n_Δ of required integration steps. However, the functional dependence $n_\Delta(\mathrm{TOL})$ shows only a more or less *logarithmic* behavior for one-step methods of *fixed* order and turns out to be essentially *constant* for extrapolation methods with *simultaneous order and step-size control.* Thus, adaptive extrapolation methods frequently exhibit the desirable proportionality $|\epsilon_\Delta(T)| \propto \mathrm{TOL}$.

Scaling. When we introduce the effect of scaling the components by some diagonal matrix $D_j = \operatorname{diag}(\sigma_1(t_j), \dots, \sigma_d(t_j))$, the local error control becomes

$$|D_j^{-1}[\epsilon_j]| \le \mathrm{TOL},$$

and we have

$$D_{j+1}^{-1}\epsilon_\Delta(t_{j+1}) \doteq \left(D_{j+1}^{-1} W(t_{j+1}, t_j) D_j\right) D_j^{-1}\epsilon_\Delta(t_j) + D_{j+1}^{-1}\epsilon_{j+1}.$$

Hence, as in the unscaled case, the introduction of the scaled pointwise condition

$$\hat\kappa_j(t_{j+1}) = \|D_{j+1}^{-1} W(t_{j+1}, t_j) D_j\| \le \hat\kappa, \qquad \hat\kappa \ge 1,$$

yields the estimate

$$|D_n^{-1}\epsilon_\Delta(T)| \dot{\le} \frac{\hat\kappa^n - 1}{\hat\kappa - 1}\,\mathrm{TOL}.$$

A suitable scaling can compensate for part of the error growth in such a way that despite larger steps, we obtain

$$\hat\kappa \approx 1,$$

and hence that we have the rough formula

$$|D_n^{-1}\epsilon_\Delta(T)| \approx n_\Delta \cdot \mathrm{TOL}.$$

Influence of Round-Off Errors. In the beginning of this chapter we noted that problem-adapted meshes generally have marked advantages over uniform meshes when it comes to the resulting *computational costs.* We now show that beyond this, there is a further important argument for keeping the number n_Δ of nodes of a mesh Δ as small as possible. Let $\hat x_\Delta$ be the mesh function as it was actually computed in finite-precision arithmetic. Standard round-off error theory (see, e.g., [58, Chapter 2]) shows that for a *numerically stable* implementation of a one-step method there is a moderate constant $\sigma \ge 1$ (the stability indicator of *one* step) such that

$$\max_{t\in\Delta} |x_\Delta(t) - \hat x_\Delta(t)| \le \sigma n_\Delta \kappa_\Delta \mathrm{eps} + O(\mathrm{eps}^2).$$

Here eps is the machine accuracy and κ_Δ the discrete condition number of the one-step method introduced in Section 4.1. Hence, the actual total error of a one-step method of order p can be bounded by

$$\max_{t\in\Delta} |x(t) - \hat x_\Delta(t)| \lesssim C\tau_\Delta^p + \sigma n_\Delta \kappa_\Delta \mathrm{eps}, \tag{5.21}$$

where for sufficiently small maximal step sizes τ_Δ, we have

$$\kappa_\Delta \approx \kappa[t_0, T].$$

We observe here two opposing effects: The reduction of the maximal step size τ_Δ reduces the contribution of the discretization error but increases that of the round-off error. Evidently, for fixed word length it is impossible to achieve arbitrary accuracies, an observation that *qualitatively* is certainly obvious. In the case of a *quasi-uniform* mesh, i.e., when

$$n_\Delta = O(\tau_\Delta^{-1}),$$

we find for the best achievable error $\epsilon_{\min}$ the rough formula

$$\epsilon_{\min} = O\left(\text{eps}^{p/(p+1)}\right).$$

But the constants hidden here are very difficult to estimate, and hence the formula provides only information about the expected improvement when the machine accuracy is *changed.*

Example 5.17 We return to the problem of Example 5.1 and apply the explicit Euler method on an equidistant mesh. From test runs with relatively large step sizes τ and from the discretization error relation

$$\max_{t\in\Delta} |x(t) - x_\Delta(t)| \approx C\tau,$$

we estimate that an error of 2.5×10^{-7} in the spatial variables requires about

$$n \approx 1.58 \times 10^{11}$$

nodes. Therefore, in the case of double precision with a typical mantissa of 52 bits we expect the round-off contribution of the error (5.21) to be about

$$\sigma\, n\, \kappa[t_0, T]\text{eps} \approx 3.5 \cdot 10^{-5} \cdot \sigma\, \kappa[t_0, T].$$

Clearly, since $\sigma\kappa[t_0, T] \geq 1$, the desired accuracy is not achievable with the explicit Euler method on equidistant meshes! In comparison, the data of Example 5.1 for the Runge-Kutta method of order 4 provide a round-off error contribution of size approximately

$$2.6 \cdot 10^{-11} \cdot \sigma_{\mathrm{RK4}}\kappa[t_0, T]$$

when equidistant meshes are used, and of size

$$4.2 \cdot 10^{-13} \cdot \sigma_{\mathrm{RK4}}\kappa[t_0, T].$$

when the meshes are adaptively constructed.

Thus, any overly restrictive error tolerance TOL close to the machine accuracy eps has to be used with much caution. Yet a lower bound for TOL cannot be found from the above estimates because of the critical role played by too many unknown constants in the problem and in the method. The larger the discrete condition κ_Δ, the more sensitive will be the reaction of a method to potentially unstable implementations of the right side f. However, for "sufficiently fine" meshes differences in the methods will level out, since then, in essence, only the condition of the problem is dominant. Unfortunately, for certain problems and methods, "sufficiently fine" may turn out to be "too fine" for any effective computation.

Exercises

Exercise 5.1 Consider a differential equation

$$x' = f(t, x). \tag{I}$$

By Section 3.2.1 the coordinate transformation $\hat{x} = Mx$ with $M \in \mathrm{GL}(d)$ produces the differential equation

$$\hat{x}' = \hat{f}(t, \hat{x}) = Mf(t, M^{-1}\hat{x}), \tag{II}$$

and the corresponding evolutions satisfy the relation

$$\hat{\Phi}^{t,s}\hat{x} = M\Phi^{t,s}M^{-1}\hat{x}.$$

In Exercise 4.1 we saw that for any Runge-Kutta method the discretization inherits this transformation behavior, i.e., the method associates with the differential equations (I) and (II) discrete evolutions Ψ and $\hat{\Psi}$, respectively, such that

$$\hat{\Psi}^{t+\tau,t}\hat{x} = M\Psi^{t+\tau,t}M^{-1}\hat{x}.$$

In particular, for a *fixed* mesh Δ it follows that M also transforms the mesh function

$$\hat{x}_\Delta = Mx_\Delta. \tag{III}$$

Provide reasons why for a Runge-Kutta method with step-size control the relation (III) is generally false, or even nonsensical.

However, the transformation behavior (III) can be "saved" for a restricted class of *scaling transformations*

$$M = \operatorname{diag}(s_1, \ldots, s_d) > 0,$$

provided that the algorithm uses the internal relative scaling discussed in Section 5.1. Give a precise formulation for this assertion and justify it.

Exercise 5.2 Use the extrapolation technique introduced in Example 5.11 to modify the program of Exercise 4.3 to a method of type RK5(4) with step-size control. Apply the program to the initial value problem (restricted three-body problem) solved in Exercise 4.4 with a constant step size. For the program utilize

- a safety factor $\rho = 0.9$,
- a step increase factor $q = 5$,
- the Euclidean norm,
- internal relative scaling with absolute minimal value $s_{\min} = 1.0$.

Perform computational runs with the tolerances TOL $= 10^{-1}, \ldots, 10^{-9}$ and tabulate the following results:

TOL, $|n_\Delta|$, # of step-size reductions, # of f evaluations, error at T.

What tolerance TOL is needed for an error of ± 1 km? Compare the cost with the results for constant step sizes in Exercise 4.4.

Exercise 5.3 Prove that there exists *no* four-stage Runge-Kutta method $(b, \mathcal{A})$ with consistency order $p = 4$ that has an embedding $(\hat{b}, \mathcal{A})$ with consistency order $\hat{p} = 3$ (consult Example 5.12).

Exercise 5.4 Program a method of type 5(4) with step-size control using the Dormand-Prince coefficients in Table 5.1. Proceed as in Exercise 5.2.

Exercise 5.5 Compute the procedural constant γ introduced in Lemma 4.26 for the diagonal and subdiagonal entries of the extrapolation tableaux of EULEX and DIFEX1 or ODEX. For this compare different subdivision sequences $\mathcal{F}$ up to order $p = 10\omega$. Use the results to explain why EULEX is not suited for high accuracy requirements.

Exercise 5.6 Modify the program of Exercise 4.3 to a method of the type RK4(3) with step-size control by embedding in it the Runge-Kutta method with consistency order $p = 3$ given in Example 5.13. For this:

- use the Fehlberg trick, and
- do not recompute the stage k_1 in a step reduction.

Choose the parameters (safety factor, norm, scaling, minimal value) of the step-size control as in Exercise 5.2 and apply the program in the same way as described there. Compare your results with those for the method of type RK5(4) discussed in Exercise 5.2.

Exercise 5.7 Consider a Runge-Kutta method of order p and combine it with extrapolation due to Richardson for step-size control. As a basis for *interpolation* of the values

$$x_0, f(t_0, x_0), \quad \hat{x}_1, f(t_0 + \tau, x_1), \quad \hat{x}_2, f(t_0 + 2\tau, \hat{x}_2)$$

by a quintic polynomial take the numerical solutions $x_0, x_1 = \Psi^{\tau/2} x_0$, $x_2 = \Psi^{\tau/2} x_1$ and the extrapolated values

$$\hat{x}_1 = x_1 + \frac{x_2 - w}{(2^p - 1)2}, \qquad \hat{x}_2 = x_2 + \frac{x_2 - w}{2^p - 1}, \qquad \text{where } w = \Psi^\tau x_0.$$

Show that the resulting method is of order $p^* = \min(5, p+1)$.