Automatic Control and Adaptive Time–Stepping

Gustaf Söderlind[†]

Numerical Analysis

Centre for Mathematical Sciences

Lund University, Box 118

SE-221 00 Lund, Sweden.

e-mail: Gustaf.Soderlind@na.lu.se

June 13, 2001

Abstract.

Adaptive time-stepping is central to the efficient solution of initial value problems in ODEs and DAEs. The error committed in the discretization method primarily depends on the time-step size h, which is varied along the solution in order to minimize the computational effort subject to a prescribed accuracy requirement. This paper reviews the recent advances in developing local adaptivity algorithms based on well established techniques from linear feedback control theory, which is introduced in a numerical context. Replacing earlier heuristics, this systematic approach results in a more consistent and robust performance. The dynamic behaviour of the discretization method together with the controller is analyzed. We also review some basic techniques for the coordination of nonlinear equation solvers with the primary stepsize controller in implicit time-stepping methods.

1 Introduction

The algorithmic content of ODE/DAE software for initial value problems is not dominated by the discretization method as such but by a considerable amount of control structures, support algorithms and logic. Nevertheless, it appears that only the discretization methods have received a thorough mathematical attention while control logic and structures have been largely heuristic. These algorithmic parts are, however, amenable to a rigorous analysis and can be constructed in a systematic way. Our objective is to introduce the numerical analyst interested in ODEs to some basic notions and techniques from control theory that have proved efficient in the construction of adaptive ODE/DAE software. Although lesser known to the numerical analyst, this methodology rests on the familiar classical theories of linear difference equations, difference operators and stability, and is therefore more readily accessible than might be expected. The present

[†]Supported by Swedish Research Council for Engineering Sciences TFR grant 222/98-74.

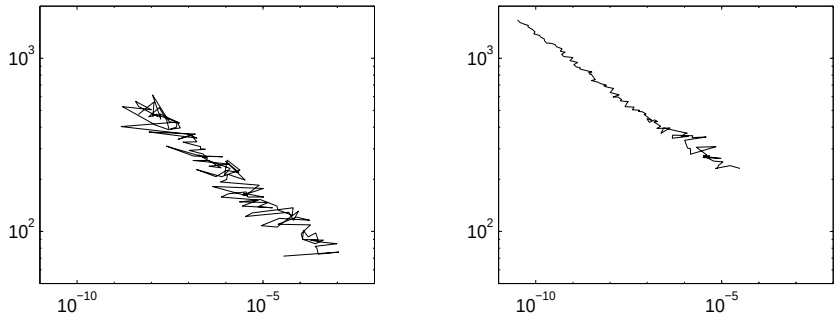

Figure 1.1: *Computational stability.* The same ODE was solved with two different codes: an implicit multistep method with elementary control (left) and an implicit one-step method with control-based adaptivity (right). The work/precision diagrams show # of f evaluations vs. achieved accuracy when TOL is gradually changed through the same values for both codes. The right graph shows a higher computational stability.

paper should serve as a starting point for gaining the necessary insight into this area and for accessing the state of the art in adaptive time-stepping as of 2000 through the referenced literature.

We shall consider the problem of numerically solving an ODE

$$\dot{y} = f(y); \qquad y(0) = y_0, \quad t \geq 0\,, \tag{1.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$. The qualitative behaviour of the solution $y(t)$ may vary considerably depending on the properties of f; f may be linear or nonlinear, some problems are sensitive to perturbations, some are stiff, some have smooth solutions while others have intervals where $y(t)$ changes rapidly. Sometimes an accuracy of a few digits is sufficient but occasionally high precision results are required. To handle the great variety of situations many different discretization methods are needed, but each automatic integration procedure must nevertheless be able to adapt properly to a wide range of operating conditions with which it might be confronted.

The *work/precision efficiency* of an adaptive time-stepping method depends on the discretization as well as on problem properties and size. The integration procedure should attempt to compute a numerical solution $\{y_n\}$ to (1.1) with minimum effort, subject to a prescribed *error tolerance* TOL. As TOL $\to 0$, the global error $\|y_n - y(t_n)\|$ should decrease in a predictable way—a property called *tolerance convergence*. At the same time, computational efforts increase, usually in a logarithmic proportionality between accuracy and effort. There are further requirements for high-quality software, e.g. *computational stability*: a small change of TOL or any other parameter in the computational setup should only produce a small change in the computed results; a non-smooth or irregular behaviour is inferior to a regular one, even if both should satisfy the basic accuracy requirement (see Fig. 1.1). Tolerance convergence is then further refined

to *tolerance proportionality*, which requires that

$$c \cdot \text{TOL} \leq \|y_n - y(t_n)\| \leq C \cdot \text{TOL}, \tag{1.2}$$

where the two constants C and c must be sufficiently close; a moderately high demand is to require that they differ by less than half an order of magnitude, i.e., $C/c = \sqrt{10}$. (On an absolute scale, the values of these constants vary with the problem's global error accumulation.) If the software is capable of delivering such quality (see the right graph of Fig. 1.1), a global error estimate becomes very cheap: it suffices to extrapolate on different values of TOL.

Performance is therefore not only subject to an accuracy requirement—trying to make a code achieve "ultimate performance" on a few test problems used for tuning, is usually of little value as it often trades robustness and coherence for a marginal gain in a single aspect of performance, e.g. total number of steps or f evaluations. Other aspects, such as the quality of the numerical solution in terms of better smoothness achieved by smoother stepsize sequences, improved computational stability, and a regular, tight tolerance proportionality, are often overlooked but do belong in a serious evaluation of software performance and quality. The use of control theory in the design of adaptive techniques helps achieve these goals. Although the primary goal is quality, the new algorithms typically do not increase the computational efforts; occasionally a higher quality may even be obtained at a reduced expense.

Minimizing work subject to a bound on the global error requires a global computational strategy. But the nature of time-stepping is inherently sequential or local; given the "state" $y(t)$, the method is a procedure for computing an approximation to $y(t+h)$ a time-step $h > 0$ ahead. The size of h is used to trade accuracy for efficiency and vice versa, and is therefore the principal *internal* means of controlling the error and making the computational procedure adaptive; this will be linked to the *external* error control variable TOL, through the well established practice of *controlling the local error*. This is inexpensive and far simpler than controlling the global error, and, by using differential inequalities, it can be shown that if the local error per unit time of integration is kept below TOL, then the global error at time t is bounded by $C(t) \cdot \text{TOL}$. Thus a local error tolerance indirectly affects the global error.

The purpose of this paper is to review the recent advances in using the theory of automatic control for the design of adaptive numerical time-stepping. In Section 2 we study the elementary controller and its shortcomings, and review some of the historical development. In Section 3 we introduce some methodology and basic notions from control theory. The framework is that of linear difference equations, and basic stability and smoothness issues are studied. The structure of the PI controller is also introduced. Section 4 then gives a detailed control theoretic account of the PI control design process and the parameterization of the controller. The framework is now linear difference operators and z transforms; impulse, step and frequency responses are discussed. The relation to the Lax equivalence theorem from numerical analysis is demonstrated. In Section 5, we proceed to stiff computations where the PI control may be replaced by a

predictive control. As stiff computations use implicit methods, they also rely on iterative methods of Newton type. The convergence rate of such iterations is affected by the stepsize, and stepsize changes may force matrix refactorizations. In Section 6 we describe a local strategy for coordinating matrix strategies with the controller without interfering with the predictive controller. This reduces unnecessary matrix handling and convergence failures. These techniques are important steps towards eliminating heuristics in numerical ODE software. They are available in the literature in concise algorithmic descriptions to facilitate their practical implementation.

2 Elementary control

We shall assume that an s-stage Runge–Kutta method (A, b) is used to solve (1.1). Using standard notation, [8], we then have

$$Y = \mathbb{1} \otimes y_n + (A \otimes I_d)h\dot{Y} \tag{2.1}$$

$$\dot{Y}_i = f(Y_i)\,; \qquad i = 1:s \tag{2.2}$$

$$y_{n+1} = y_n + (b^{\mathrm{T}} \otimes I)h\dot{Y}, \tag{2.3}$$

where y_n approximates $y(t_n)$. Further, h is the stepsize, Y is the sd–dimensional stage vector, and $\dot{Y}$ is the corresponding stage derivative. The method may be either explicit or implicit, and we assume that the method supports a reference method defined by a different quadrature formula

$$\hat{y}_{n+1} = y_n + (\hat{b}^{\mathrm{T}} \otimes I)h\dot{Y}, \tag{2.4}$$

providing the *local error estimate* $\hat{l}_{n+1} = y_{n+1} - \hat{y}_{n+1} = ((b^{\mathrm{T}} - \hat{b}^{\mathrm{T}}) \otimes I_d)h\dot{Y}$. Let $\hat{y}(t;\tau,\eta)$ denote a solution to (1.1) with initial condition $\hat{y}(\tau) = \eta$. Then the *local error* in a step of y_{n+1} is $l_{n+1} = y_{n+1} - \hat{y}(t_{n+1}; t_n, y_n)$. By expanding the local error in an asymptotic series one shows that

$$l_{n+1} = \Phi(t_n, y_n)h^{p+1} + O(h^{p+2}), \tag{2.5}$$

where Φ is the principal error function and p is the order of the method. Similarly, one derives a corresponding expression for the local error estimate,

$$\hat{l}_{n+1} = \hat{\Phi}_n h^{\hat{p}+1} + O(h^{\hat{p}+2}), \tag{2.6}$$

where the order $\hat{p} \le p$, depending on design objectives for the method. For simplicity we shall make no distinction between $\hat{p}$ and p but wish to emphasize that in the local error control algorithms under discussion, the "order" always refers to that of the error estimate, i.e. $\hat{p}$.

It is of interest to control either the local error per step (EPS) or the local error per unit step (EPUS). In the former case we let $\hat{r}_{n+1} = \|\hat{l}_{n+1}\|$ and in the latter $\hat{r}_{n+1} = \|\hat{l}_{n+1}/h_n\|$. A step is accepted if $\hat{r}_{n+1} \le \mathrm{TOL}$, and efficiency suggests choosing the stepsize adaptively, as large as possible, subject to this accuracy requirement. To reduce the risk of having to reject a step, it is common to aim

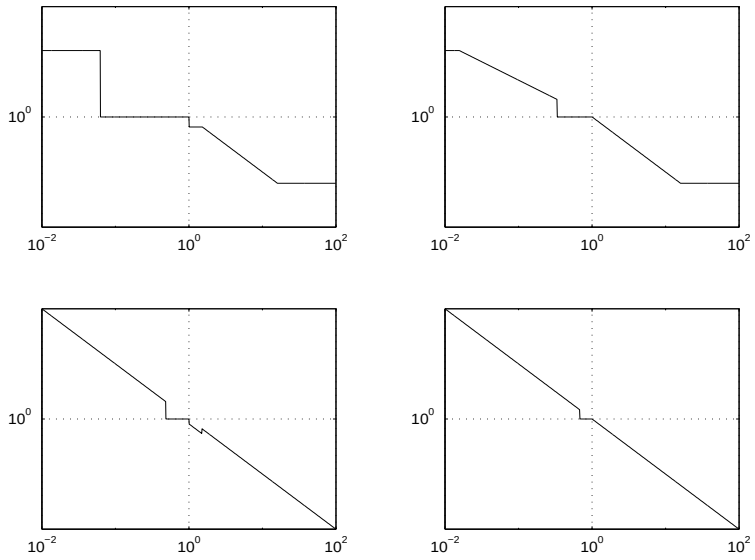

Figure 2.1: *Stepsize strategies.* Stepsize change ratios $\log(h_{n+1}/h_n)$ as functions of the error excess $\log(\hat{r}_{n+1}/\varepsilon)$ in four codes: DASSL (top left), RADAU5 (top right), LSODE (bottom right) and DASP3 (bottom left), a lesser known DAE solver developed by the author in 1976–80. The graphs show the essential features of the strategies. The overall negative "slopes" reflect a negative feedback, which has a stabilizing effect. All strategies are nonlinear, discontinuous and unsymmetric, making it virtually impossible to analyze their dynamics.

for a slightly smaller error, $\varepsilon = \theta \cdot \text{TOL}$, where $\theta < 1$ is a safety factor. The *elementary local error control* algorithm [2, p. 156] is

$$h_{n+1} = \left(\frac{\varepsilon}{\hat{r}_{n+1}}\right)^{1/k} h_n, \tag{2.7}$$

were $k = p + 1$ (EPS) or $k = p$ (EPUS). In practice, this elementary control is implemented with limiters and exceptions, see Fig. 2.1.

The heuristic derivation of the control law (2.7) assumes the following behaviour of the integration process:

Process assumptions

1. *Asymptotics:* $\hat{r}_{n+1} = \hat{\varphi}_n h_n^k$ where $\hat{\varphi}_n = \|\hat{\Phi}_n\|$
2. *Slow variation:* $\hat{\varphi}_n \approx \hat{\varphi}_{n-1}$

If these were correct, and there is a deviation between ε and $\hat{r}_{n+1}$, then (2.7)

will eliminate this deviation in a single step and make the error equal ε:

$$h_{n+1} = \left(\frac{\varepsilon}{\hat{\varphi}_n h_n^k}\right)^{1/k} h_n \quad \Rightarrow \quad \hat{\varphi}_n h_{n+1}^k = \varepsilon. \tag{2.8}$$

Hence if $\hat{\varphi}_n$ is constant the new stepsize exactly meets the accuracy requirement.

In practice it may happen that one or both process assumptions are false. The first states that h_n is small enough for the error to exhibit its theoretical asymptotic behaviour. Some reasons why this may be false are (*i*) in an explicit method the stability region is bounded and if the stepsize is limited by numerical stability it is no longer in the asymptotic regime [4, p. 534]; (*ii*) in stiff ODEs one uses "large" stepsizes far outside the asymptotic regime for modes which have decayed [8, pp. 113–114]; (*iii*) for stiff ODEs and DAEs, some methods suffer from order reduction, also invalidating the classical asymptotic model for the order. The second assumption asserts that f and/or y have a negligible variation during a time-step of size h and is rarely if ever correct, [5, p. 503]. It also depends on the smoothness of the norm $\|\cdot\|$, and whether the norm is "aligned" with the behaviour of the solutions.

In spite of its shortcomings, the elementary error control has been very successful in practical computations. One apparent success is its ability to prevent numerical instability. In computations with an explicit method, stepsizes may eventually grow until h_n must be limited by numerical stability. This is at large managed by (2.7). As long as stability is at hand, the solution y_n remains smooth and the error small, and (2.7) will attempt to increase the stepsize. But if h_n increases to cause instability, no matter how slight, then y_n quickly becomes less regular. As a result $\hat{r}_n$ increases, forcing (2.7) to reduce h_n. This process repeats itself and keeps instability at bay through a continual adjustment of h_n.

But this suggests that h_n will oscillate around a maximum stable stepsize. Such oscillations are indeed observed in practice, see [3], [8, p. 25], and may even have visible effects on the smoothness of the numerical solution [4, p. 534], [8, p. 31]. This led Hall [9, 10] to study a new question of stability: is the method *in tandem* with the control (2.7) stable when numerical stability limits the stepsize? It was found that for the linear test equation $\dot{y} = \lambda y$ stable stepsize equilibria exist on parts of the boundary of the stability region, [8, p. 26]. For some popular methods, however, the performance is less satisfactory. Hall and Higham [12] then took the approach of constructing new methods that together with (2.7) had improved "step control stability" and thus overcame stepsize oscillations.

Around the same time, however, Gustafsson *et al.* [3] studied the problem from a control theoretic point of view. This starts with the same question of considering the stability of method and controller in tandem, but the approach is the opposite: instead of constructing methods that match the control law (2.7), the controller is designed to match the method. Moreover, because (2.7) is as elementary to feedback control theory as the explicit Euler method is to numerical ODEs, it should be possible to employ a more advanced digital control design parameterized for each given method. A first experimental study [3]

confirmed that a proven standard technique, the *PI controller*, worked well. It was subsequently thoroughly analyzed, tested and parameterized [4]. This resulted in smoother stepsize sequences and numerical solutions [4, p. 547], [8, p. 31], fewer rejected steps, an improved ability to prevent numerical instability and a possibility to run closer to TOL, i.e. to use a larger θ; in all, a more robust performance at no extra computational expense. This technique has since become the modern standard for nonstiff computations and is promoted in various guises in research literature [8, pp. 28–31] and general numerical analysis textbooks alike [1, pp. 173–179].

3 Difference equations and feedback control

Let us now consider (2.7), known in control theory as a discrete-time integral controller (*I controller*). Limiters and additional logic (see Fig. 2.1) will be disregarded. Taking logarithms in (2.7) yields the linear difference equation

$$\log h_{n+1} = \log h_n + \frac{1}{k}\,(\log \varepsilon - \log \hat{r}_{n+1}). \tag{3.1}$$

The term $\log \varepsilon - \log \hat{r}_{n+1}$ is called the *control error*, and the factor $k_{\mathrm{I}} = 1/k$ is referred to as the *integral gain*. The controller acts to make the error estimate $\log \hat{r}_{n+1}$ equal the *setpoint* $\log \varepsilon$ so that the control error vanishes; only then will the *control* $\log h_n$ cease to change.

Solving the difference equation (3.1) we obtain

$$\log h_n = \log h_0 + \frac{1}{k} \sum_{m=1}^{n} (\log \varepsilon - \log \hat{r}_m). \tag{3.2}$$

This is a discrete-time analogue of an integral, hence the term *integral control*; the stepsize is obtained as a weighted sum of all past control errors.

We next turn to the *process*, which consists of the numerical method and the differential equation to be solved; it will here be modelled by the asymptotic relation $\hat{r}_{n+1} = \hat{\varphi}_n h_n^k$, or

$$\log \hat{r}_{n+1} = k \log h_n + \log \hat{\varphi}_n. \tag{3.3}$$

The *closed loop dynamics*, representing the interaction of controller and controlled process, is then found by eliminating $\log \hat{r}_{n+1}$ from process and controller. Thus we insert the process model (3.3) and its dependence on the control $\log h_n$, according to the asymptotic assumption, into the control law (3.1) to obtain

$$\log h_{n+1} = \frac{1}{k}\,(\log \varepsilon - \log \hat{\varphi}_n). \tag{3.4}$$

This is a first-order difference equation for $\log h_n$ with characteristic equation $q = 0$, i.e., the roots are at the origin. Such closed loop dynamics is referred to as *deadbeat control*, and is in this case the result of choosing the integral gain $k_{\mathrm{I}} = 1/k$ in the controller. We also note that $\log h_{n+1}$ is proportional to $\log \hat{\varphi}_n$,

which implies that the stepsize sequence generated by the deadbeat control has *the same smoothness* as the principal error function $\{\hat{\varphi}_n\}$.

From the point of view of control theory, however, the integral gain k_{I} is not determined by the asymptotic model of the process. It is a free *design parameter* used to achieve a good overall closed loop dynamic behaviour for process and controller *together*. We therefore replace the factor $1/k$ in (3.1) by an integral gain k_{I} to be determined later, and insert the asymptotic model for $\hat{r}_{n+1}$. We then obtain the closed loop dynamics

$$\log h_{n+1} = (1 - kk_{\mathrm{I}}) \log h_n + k_{\mathrm{I}}(\log \varepsilon - \log \hat{\varphi}_n). \tag{3.5}$$

Thus the root of the characteristic equation is now $q = 1 - kk_{\mathrm{I}}$. Stability evidently requires $kk_{\mathrm{I}} \in [0, 2]$, but the choice $kk_{\mathrm{I}} = 1$ is by no means necessary. Instead, the value of kk_{I} determines how quickly the control responds to changes in the *external disturbance* $\log \hat{\varphi}_n$, the influence of which is to be *rejected* or compensated by the controller. In the deadbeat controller, $\log h_{n+1}$ depends exclusively on $\log \hat{\varphi}_n$, implying—as we already noted—that the error will be immediately rejected. But in general this leads to a rather "nervous" control, which is often not to advantage. Taking $kk_{\mathrm{I}} \in (0, 1)$ leads to a slower, smoother control, where the control variable $\log h_n$ depends in part on its history and in part on $\log \hat{\varphi}_n$. By contrast, $kk_{\mathrm{I}} \in (1, 2)$ makes the controller overreact, and on subsequent steps it must even compensate its own actions, resulting in damped oscillations. The choice of kk_{I} is therefore a tradeoff between response time and sensitivity; for an I controller to work well one must in effect choose $kk_{\mathrm{I}} \in (0, 1)$. In summary we have:

$kk_{\mathrm{I}} \in [0, 2]$	$\Leftrightarrow$	*stability*
$kk_{\mathrm{I}} \in (1, 2)$	$\Leftrightarrow$	fast, *oscillatory* control
$kk_{\mathrm{I}} = 1$	$\Leftrightarrow$	*deadbeat* control
$kk_{\mathrm{I}} \in (0, 1)$	$\Leftrightarrow$	slow, *smooth* control

How much smoother is the stepsize sequence if the integral gain is reduced? The solution of the difference equation (3.5) is given by the *convolution*

$$\log h_n = (1 - kk_{\mathrm{I}})^n \log h_0 + k_{\mathrm{I}} \sum_{m=1}^{n} (1 - kk_{\mathrm{I}})^{n-m} (\log \varepsilon - \log \hat{\varphi}_{m-1}). \tag{3.6}$$

Known as "exponential forgetting," (3.6) *mollifies the variations in* $\log \hat{\varphi}_n$ *when* $kk_{\mathrm{I}} \in (0, 1)$, in exactly the same way as a continuous convolution with a smooth function yields one more derivative. Damping $(-1)^n$ oscillations, a low-gain I controller therefore yields *smoother stepsize sequences* than the deadbeat control (3.2), for which $\log h_{n+1}$ and $\log \hat{\varphi}_n$ had the same smoothness, see (3.4).

The general I controller can be written as

$$h_{n+1} = \left(\frac{\varepsilon}{\hat{r}_{n+1}} \right)^{k_{\mathrm{I}}} h_n. \tag{3.7}$$

When we compare this to (2.7), we wish to emphasize that the change of integral gain from $1/k$ to k_{I} is not an arbitrary violation of the theoretical asymptotics of

the method but *a deliberate change of controller dynamics* to achieve smoother stepsize sequences *for the very same asymptotic error model* as was assumed for the elementary control algorithm (2.7). The control analysis above rests on the asymptotic assumption, but not on the assumption of slow variation.

In control theory it is common to use *PI and PID control* structures to increase robustness and in other ways improve control performance. PI stands for *proportional–integral*, and in PID one adds D for *–derivative*. In most cases, a PI controller is satisfactory. Its idea is to construct the control $\log h_n$ as follows:

$$\log h_n = \log h_0 + k_{\mathrm{I}} \sum_{m=1}^{n} (\log \varepsilon - \log \hat{r}_m) + k_{\mathrm{P}} (\log \varepsilon - \log \hat{r}_n). \tag{3.8}$$

Thus $\log h_n$ consists of two terms: it adds a term *proportional* to the control error to the previous *integral* term in (3.2). The *proportional gain* k_{P} and integral gain k_{I} are to be chosen to obtain suitable closed loop dynamics.

Forming a recursion from (3.8), we obtain the control

$$\log h_{n+1} = \log h_n + k_{\mathrm{I}} (\log \varepsilon - \log \hat{r}_{n+1}) + k_{\mathrm{P}} (\log \hat{r}_n - \log \hat{r}_{n+1}). \tag{3.9}$$

The *PI controller* can therefore be written

$$h_{n+1} = \left(\frac{\varepsilon}{\hat{r}_{n+1}} \right)^{k_{\mathrm{I}}} \left(\frac{\hat{r}_n}{\hat{r}_{n+1}} \right)^{k_{\mathrm{P}}} h_n, \tag{3.10}$$

and is obviously a relatively simple modification to the elementary local error control. The new proportional factor accounts for error trends; if $\hat{r}_n$ is increasing, the new factor is less than 1 (provided that $k_{\mathrm{P}} > 0$) and makes for a quicker stepsize reduction than the I controller would have produced. Conversely, a decreasing error leads to a faster stepsize increase.

The major change, however, is that the closed loop has second order dynamics. Inserting the asymptotic model into (3.9), the closed loop dynamics is

$$\begin{aligned} \log h_{n+1} \quad = \quad & (1 - k k_{\mathrm{I}} - k k_{\mathrm{P}}) \log h_n + k k_{\mathrm{P}} \log h_{n-1} + \\ & k_{\mathrm{I}} (\log \varepsilon - \log \hat{\varphi}_n) + k_{\mathrm{P}} (\log \hat{\varphi}_{n-1} - \log \hat{\varphi}_n), \end{aligned}$$

with the characteristic equation

$$q^2 - (1 - k k_{\mathrm{I}} - k k_{\mathrm{P}}) q - k k_{\mathrm{P}} = 0. \tag{3.11}$$

The dynamics is therefore determined by two roots, the locations of which are functions of the two design parameters $k k_{\mathrm{I}}$ and $k k_{\mathrm{P}}$. One should not, however, choose the values of $k k_{\mathrm{I}}$ and $k k_{\mathrm{P}}$ solely from this characteristic equation, as we have assumed that the asymptotic process model holds when we derived (3.11). It is necessary to study additional process models, e.g. the dynamics when numerical stability limits the stepsize, to find a good parameterization.

The PI controller (3.10) can also be written in the form

$$h_{n+1} = \left(\frac{\varepsilon}{\hat{r}_{n+1}} \right)^{k_{\mathrm{I}} + k_{\mathrm{P}}} \left(\frac{\varepsilon}{\hat{r}_n} \right)^{-k_{\mathrm{P}}} h_n, \tag{3.12}$$

which is appealing in particular because the control action is expressed in terms of present and past control errors only. It may also be convenient to introduce new parameters $\alpha = k_{\mathrm{I}} + k_{\mathrm{P}}$ and $\beta = k_{\mathrm{P}}$, [8, p. 30]. But this is not always advantageous as α and β become interdependent: k_{I} and k_{P} have different effects on the controller dynamics and it can be argued that they should be handled separately.

In the case of the pure asymptotic model and an almost constant external disturbance $\log \hat{\varphi}_n$, a controller with integral action is typically both necessary and sufficient [20]. In many but certainly not all nonstiff computations these assumptions are not unrealistic, explaining the success of the elementary controller (2.7). It is in the remaining cases that a more robust controller is needed, and our next task is to outline the control analysis necessary to design and parameterize the PI controller properly.

4 Time-stepping with PI control: First order adaptivity

The analysis and design of a linear control system has four steps:

1. Find the dynamics of the process to be controlled
2. Select a control structure
3. Combine to find the closed loop dynamics
4. Select appropriate control parameters

It is often steps 1 and 4 that present the more challenging tasks. The procedure is usually carried out in the frequency domain rather than in the time domain. This implies that process and controller dynamics are represented in terms of their *z-transforms*. We shall denote the transform variable, corresponding to the forward shift operator, by q, and let $G_{\mathrm{P}}(q)$ and $G_{\mathrm{C}}(q)$ denote the *transfer functions* of the process and controller, respectively. Further, transformed stepsize, error and disturbance sequences are denoted without subscript, see Fig. 4.1.

4.1 Process models

The first task is to *identify the process models*, which map $\log h$ to $\log \hat{r}$. When the asymptotic assumption holds, we have $\log \hat{r} = G_{\mathrm{P}}(q) \log h + q^{-1} \log \hat{\varphi}$, where the process is just a constant gain,

$$G_{\mathrm{P}}(q) = G_{\mathrm{P}}^0(q) = kq^{-1}, \tag{4.1}$$

implying that the process is *static*. The backward shift q^{-1} appears as a consequence of indexing conventions; when the method takes a step of size h_n it produces an error estimate $\hat{r}_{n+1} = \hat{\varphi}_n h_n^k$.

When numerical stability limits the stepsize, however, the process becomes *dynamic*. A dynamic process model was first obtained by Hall in [9, 10] for explicit Runge–Kutta methods applied to the linear test equation[1] $\dot{y} = \lambda y$.

[1] Although this might appear to be a too simple equation, this problem represents differential equations of the form $\dot{x} = f(x - \psi(t)) + \dot{\psi}(t)$, with $f(0) = 0$ and $\psi(t)$ slowly varying, as $x(t)$ approaches the quasi-stationary solution $\psi(t)$.

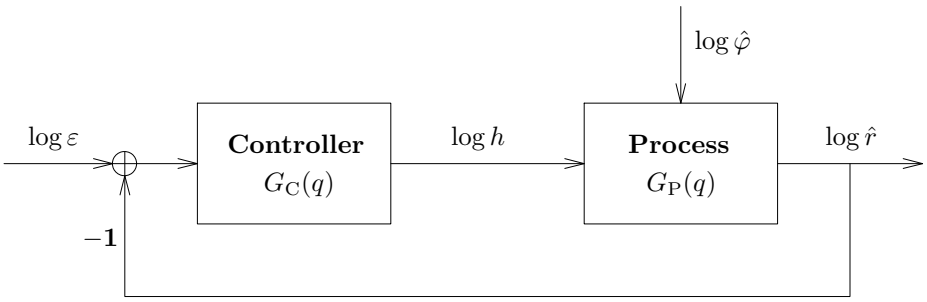

Figure 4.1: *Adaptive stepsize selection viewed as a feedback control system.* The process consists of the discretization method which takes a given stepsize $\log h$ as input and produces an error estimate output $\log \hat{r}$. The method is applied to an ODE whose influence is represented as an external disturbance $\log \hat{\varphi}$ accounting for problem properties. The error estimate $\log \hat{r}$ is fed back with reversed phase (the factor -1 on the negative feedback loop), then added to $\log \varepsilon$ to compare actual and desired error levels. Taking this difference (the control error) as its input, the controller selects the next stepsize $\log h$. In the analysis of a linear control system, the process and controller are represented by transfer functions (z transforms) mapping the inputs to outputs. The closed loop transfer function, i.e. the map from $\log \hat{\varphi}$ to $\log \hat{r}$ with the controller included, must be stable and have a good ability to reject the external disturbance; variations in $\log \hat{\varphi}$—even if substantial—must not force $\log \hat{r}$ far away from the setpoint $\log \varepsilon$.

Given the stepsize h_n, let $z_n = h_n \lambda$. The explicit Runge–Kutta method then yields $y_{n+1} = P(z_n) y_n$, where P is the stability polynomial of the method. For the error estimate one similarly has $\hat{l}_{n+1} = E(z_n) y_n$ where E is the error estimator polynomial. Let the stepsize h^* be such that $h^* \lambda = z^* \in \partial S$, the boundary of the stability region, i.e., $|P(z^*)| = 1$. We next write

$$\hat{l}_{n+1} = E(z_n) y_n = E(z_n) P(z_{n-1}) y_{n-1} = P(z_{n-1}) \frac{E(z_n)}{E(z_{n-1})} \hat{l}_n \tag{4.2}$$

and consider small stepsize variations $h_n = (1 + \epsilon_n) h^*$, i.e. $z_n = (1 + \epsilon_n) z^*$. Expanding the polynomials of the rational function $P(z_{n-1}) E(z_n) / E(z_{n-1})$ around z^*, retaining terms of first order in ϵ_n and ϵ_{n-1}, we obtain the approximation

$$\hat{l}_{n+1} \doteq P(z^*) \left(\frac{h_n}{h^*} \right)^{C_1} \left(\frac{h_{n-1}}{h^*} \right)^{C_2 - C_1} \hat{l}_n, \tag{4.3}$$

with

$$C_1(z^*) = \operatorname{Re} \left(z^* \frac{E'(z^*)}{E(z^*)} \right); \qquad C_2(z^*) = \operatorname{Re} \left(z^* \frac{P'(z^*)}{P(z^*)} \right). \tag{4.4}$$

These coefficients depend on the method parameters $(A, b, \hat{b})$ as such, but also vary along ∂S. In order to cover both EPS and EPUS, we shall take a different

approach from [11, 4] and normalize the coefficients by k, defining

$$\hat{c}_1(z^*) = C_1(z^*)/k\,; \qquad \hat{c}_2(z^*) = C_2(z^*)/k\,. \tag{4.5}$$

We now take the logarithm of $\hat{r}_n = |\hat{l}_n|$ (or in the EPUS case $|\hat{l}_n|/h_{n-1}$) and express (4.3) in terms of the forward shift q. Noting that $|P(z^*)| = 1$ we obtain

$$\log \hat{r} \doteq G_{\mathrm{P}}^{\partial S}(q) \cdot (\log h - \log h^*), \tag{4.6}$$

where the sought dynamic process model takes the form

$$G_{\mathrm{P}}^{\partial S}(q) = kq^{-1}\left(\hat{c}_1(z^*) - \frac{\delta_{pk}}{k} + \frac{\hat{c}_2(z^*)}{q-1}\right), \tag{4.7}$$

with $\delta_{pk} = 0$ for EPS and $\delta_{pk} = 1$ for EPUS, cf. [4, p. 538]. To verify this dynamic model, Gustafsson used *system identification*, [17]. The coefficients obtained when a real ODE solver was applied to quasi-stationary nonlinear problems were found to be well within 1% of theoretical values [4, pp. 541, 549–551], confirming that the dynamic model is highly accurate.

We thus have two different transfer functions representing the process, the *asymptotic model* $G_{\mathrm{P}}^{0}(q) = kq^{-1}$ valid near $z = 0$ and the *dynamic model* $G_{\mathrm{P}}^{\partial S}(q)$ valid near $z \in \partial S$. These models are *compatible*: since for all Runge–Kutta methods $\hat{c}_1(0) - \delta_{pk}/k = 1$ and $\hat{c}_2(0) = 0$, the dynamic process $G_{\mathrm{P}}^{\partial S}(q)$ reduces to the static process $G_{\mathrm{P}}^{0}(q)$ near $z = 0 \in \partial S$. For many methods $\hat{c}_1(z^*) \approx 1$, and $\hat{c}_2(z^*)$ often varies in $[0, 2]$ along ∂S, cf. [4, p. 552].

In spite of the compatibility of the models, it is important to note the following. While $G_{\mathrm{P}}^{0}(q)$ has 0th order dynamics, $G_{\mathrm{P}}^{\partial S}(q)$ has 1st order dynamics. As q is the forward shift operator, $q - 1$ is the forward difference operator; $1/(q-1)$ is therefore a *summation operator* corresponding to "integral action." In the static model (4.1) there is no such integral action, but it is present in the last term of the dynamic model (4.7), where it reflects an error accumulation process occurring when $h^*\lambda \in \partial S$. The models therefore exhibit quite different behaviours.

4.2 Controller

The second task is to *select a controller* $G_{\mathrm{C}}(q)$, a linear operator mapping the control error $\log \varepsilon - \log \hat{r}$ to the control $\log h$, expressed as

$$\log h = G_{\mathrm{C}}(q) \cdot (\log \varepsilon - \log \hat{r}). \tag{4.8}$$

The controller must be able to adequately manage both processes. The elementary controller (2.7) will not do, as it is often unable to yield stable dynamics for $G_{\mathrm{P}}^{\partial S}(q)$, so we select a PI controller. Identifying (4.8) with (3.9) we readily find the general operator expression for the *PI controller*,

$$G_{\mathrm{C}}^{\mathrm{PI}}(q) = k_{\mathrm{I}}\,\frac{q}{q-1} + k_{\mathrm{P}}. \tag{4.9}$$

In this structure, we recognize the summation operator $q/(q-1)$, which represents the controller's integral action. Similarly, the constant term k_{P} represents the proportional part. The pure I controller is naturally obtained as the special case $k_{\mathrm{P}} = 0$.

4.3 Closed loop dynamics

We next proceed to derive the *closed loop dynamics*. It is obtained by combining process and controller, leading to the equation system

$$\log \hat{r} = G_{\mathrm{P}}(q) \log h + q^{-1} \log \hat{\varphi} \tag{4.10}$$

$$\log h = G_{\mathrm{C}}(q) \cdot (\log \varepsilon - \log \hat{r}), \tag{4.11}$$

in which the system inputs $\log \hat{\varphi}$ and $\log \varepsilon$ represent the data, see Fig. 4.1. We now solve for either $\log \hat{r}$ or $\log h$. The solutions are

$$\log \hat{r} = G_{\varepsilon}(q) \log \varepsilon + G_{\hat{\varphi}}(q) \log \hat{\varphi} \tag{4.12}$$

$$\log h = H_{\varepsilon}(q) \log \varepsilon + H_{\hat{\varphi}}(q) \log \hat{\varphi}, \tag{4.13}$$

where the four general *closed loop transfer maps*, valid for any choice of process model and controller, are given by

$$G_{\varepsilon}(q) = \frac{G_{\mathrm{P}}(q)G_{\mathrm{C}}(q)}{1 + G_{\mathrm{P}}(q)G_{\mathrm{C}}(q)}\,; \qquad G_{\hat{\varphi}}(q) = \frac{q^{-1}}{1 + G_{\mathrm{P}}(q)G_{\mathrm{C}}(q)}\,; \tag{4.14}$$

$$H_{\varepsilon}(q) = \frac{G_{\mathrm{C}}(q)}{1 + G_{\mathrm{C}}(q)G_{\mathrm{P}}(q)}\,; \qquad H_{\hat{\varphi}}(q) = -\,\frac{q^{-1}G_{\mathrm{C}}(q)}{1 + G_{\mathrm{C}}(q)G_{\mathrm{P}}(q)}\,. \tag{4.15}$$

The maps $G_{\hat{\varphi}}(q) : \log \hat{\varphi} \mapsto \log \hat{r}$ and $H_{\hat{\varphi}}(q) : \log \hat{\varphi} \mapsto \log h$ are usually the most interesting and will be examined here. The other two are useful e.g. in system identification when an unknown process is to be identified.

In the special case of *a PI controller and the asymptotic process*, we insert (4.9) and (4.1) into $G_{\hat{\varphi}}(q)$. We then obtain an explicit expression for the transfer function from external disturbance $\log \hat{\varphi}$ to error estimate $\log \hat{r}$:

$$G^0_{\hat{\varphi}}(q) = \frac{q - 1}{q^2 - (1 - kk_{\mathrm{I}} - kk_{\mathrm{P}})q - kk_{\mathrm{P}}}\,. \tag{4.16}$$

Several important observations can now be made. First, concerning *stability*, the closed loop dynamics is governed by the poles of the transfer function; by looking at the denominator of (4.16) we recognize the same characteristic equation (3.11) as before. For stability these poles must be located inside the unit circle. As for *consistency*, we note that $G^0_{\hat{\varphi}}(1) = 0$: the numerator of $G^0_{\hat{\varphi}}(q)$ contains the forward difference operator $q - 1$. Appearing as consequence of the controller's integral action, this difference operator will remove any constant disturbance $\log \hat{\varphi}$ in (4.12) at a rate determined by the location of the poles. Finally, we find that $G^0_{\varepsilon}(1) = 1$. This is also a *consistency* condition, implying that $\log \hat{r}$ will, in a stable system, eventually approach the setpoint $\log \varepsilon$, see (4.12).

The same conclusions also follow by rewriting (4.12) as the difference equation

$$\big(q^2 - (1 - kk_{\mathrm{I}} - kk_{\mathrm{P}})q - kk_{\mathrm{P}}\big) \log \hat{r}_n = kk_{\mathrm{I}} \log \varepsilon + (q - 1) \log \hat{\varphi}_n. \tag{4.17}$$

Its homogeneous solutions tend to 0 if and only if the roots of the characteristic equation are inside the unit circle; further, if $\log \hat{\varphi}_n = \text{const.}$, the single remaining forcing term $kk_{\mathrm{I}} \log \varepsilon$ leads to the constant particular solution $kk_{\mathrm{I}} \log \hat{r}_n = kk_{\mathrm{I}} \log \varepsilon$, that is, $\hat{r}_n = \varepsilon$.

The observations above, well-known to the control theorist, follow exactly the same pattern as the Lax equivalence theorem in numerical analysis: *consistency and stability imply convergence*. To be precise:

Consistency:	$G^0_{\hat\varphi}(1) = 0$ and $G^0_{\varepsilon}(1) = 1$ (order 1 conditions);
Stability:	All poles of $G^0_{\hat\varphi}(q)$ inside unit circle;
Convergence:	$\log \hat r \to \log \varepsilon$.

Instead of using the term consistency, we shall speak of *1st order adaptivity*, corresponding to the first order difference operator in $G^0_{\hat\varphi}(q)$, which removes any *constant disturbance* (polynomials of degree 0) and makes $\log \hat r = \log \varepsilon$; the PI controller makes the error (via h) adapt[2] to TOL. But the controller fails, by lagging, to follow a linear trend (polynomials of degree 1) correctly. This would require *2nd order adaptivity* and hence a controller with *double integral action*. Such controllers will be examined in Section 5.

We also need to consider the closed loop dynamics on ∂S. Combining (4.6) and (4.8) we obtain

$$\log \hat r = G^{\partial S}_{\varepsilon}(q) \log \varepsilon + G^{\partial S}_{h^*}(q) \log h^*, \tag{4.18}$$

reflecting that the stepsize–error relation is different on ∂S. We now need to find the poles of the transfer functions

$$G^{\partial S}_{\varepsilon}(q) = \frac{G^{\rm PI}_{\rm C}(q)\, G^{\partial S}_{\rm P}(q)}{1 + G^{\rm PI}_{\rm C}(q)\, G^{\partial S}_{\rm P}(q)}\,; \qquad G^{\partial S}_{h^*}(q) = -\,\frac{G^{\partial S}_{\rm P}(q)}{1 + G^{\rm PI}_{\rm C}(q)\, G^{\partial S}_{\rm P}(q)}\,. \tag{4.19}$$

In the EPS case the characteristic equation is $q^3 + a_2 q^2 + a_1 q + a_0 = 0$, where

$$\begin{aligned} a_2 &= -2 - (kk_{\rm I} + kk_{\rm P})\hat c_1 \\ a_1 &= 1 - (kk_{\rm I} + 2kk_{\rm P})\hat c_1 + (kk_{\rm I} + kk_{\rm P})\hat c_2 \\ a_0 &= -kk_{\rm P}(\hat c_2 - \hat c_1). \end{aligned}$$

Similar expressions can be found also for the EPUS case. Note that because of the *dynamic process* model, the closed loop dynamics is now of *third* order. The poles are still determined by the parameters $kk_{\rm I}$ and $kk_{\rm P}$ just as in the asymptotic case, but with third order dynamics and only two free parameters one can expect to be limited in the design process.

4.4 Control parameters

It is a considerable challenge to choose the PI control parameters. There is a gradual change in process models from $G^0_{\rm P}(q)$ to $G^{\partial S}_{\rm P}(q)$ as z leaves the asymptotic domain and approaches ∂S [4, p. 541]. In addition $\hat c_1$ and $\hat c_2$ are method dependent and vary on ∂S.

[2] This does *not* hold for the strategies shown in Fig. 2.1, which all allow bounded deviations $0 < \log \varepsilon - \log \hat r_n < C$ between error and setpoint without changing h. As nonzero control errors are accepted without control action, $\log \hat r \not\to \log \varepsilon$; hence the order of adaptivity is 0.

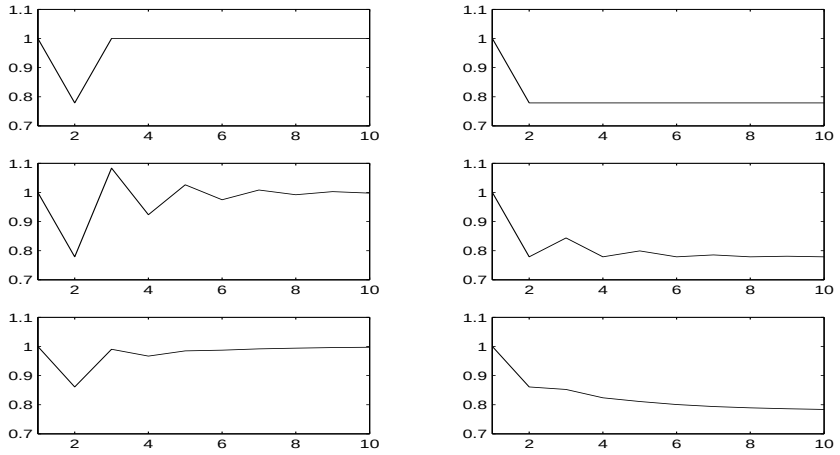

Figure 4.2: *Impulse and step responses* are used to investigate closed loop transient behaviour. The system (with asymptotic process) is originally in equilibrium when $\log\hat{\varphi}$ excites the closed loop by an impulse (left) or a step jump (right). The graphs show the PI controller's stepsize output during the next ten steps. From top to bottom: deadbeat (PI1.0) control; PI.68.32 ([8, p. 30]); PI.4.2 control. The deadbeat controller reacts with a phase-reversed impulse (top left) and step (top right), respectively, to counteract the external excitations. The PI.68.32 is prone to overshoot and its poles $q = \pm 0.57$ produce an oscillatory stepsize sequence. The PI.4.2 is less oscillatory and decays smoothly.

The choice of k_{I} and k_{P} is based on theoretical considerations as well as computational performance. A systematic investigation of the range of the coefficients $\hat{c}_1$ and $\hat{c}_2$ for a large number of methods reveals how large the closed loop stability region in the $(\hat{c}_1, \hat{c}_2)$–plane needs to be. Extensive practical testing led Gustafsson [4] to suggest the *PI.3.4* controller, defined by $(k_{\mathrm{I}}, k_{\mathrm{P}}) = (0.3, 0.4)$, to be considered as a good starting point for fine-tuning the controller for individual methods. This results in poles located at $q_+ = 0.8$ and $q_- = -0.5$ for (4.16), regardless of method.

Negative poles are less desirable as they may cause overshoot and risk step rejections, but they cannot be avoided in this design. It is however possible to "mask" the oscillations caused by q_- by choosing control parameters so that the ratio $q_+/|q_-|$ is fairly large[3]. Thus, if actual values of $\hat{c}_1$ and $\hat{c}_2$ permit a small reduction of the stability region in the $(\hat{c}_1, \hat{c}_2)$–plane, a faster, better damped and smoother response near $z = 0$ is achieved by the *PI.4.2* controller, with $(k_{\mathrm{I}}, k_{\mathrm{P}}) = (0.4, 0.2)$. The poles at $z = 0$ have then moved to $q \approx 0.69$ and $q \approx -0.29$, and the $q_+/|q_-|$ ratio has increased from 1.6 to 2.4. In most cases the parameter set of interest is $\{(k_{\mathrm{I}}, k_{\mathrm{P}}) : k_{\mathrm{I}} + k_{\mathrm{P}} \leq 0.7;\ k_{\mathrm{I}} \geq 0.3;\ k_{\mathrm{P}} \geq 0.2\}$,

[3] [8, p. 30] maintain $k_{\mathrm{I}} + k_{\mathrm{P}} = 1$, which leads to less convincing closed loop dynamics. The poles of (4.16) are $q = \pm\sqrt{k_{\mathrm{P}}}$ and the oscillatory mode cannot be masked, see Fig. 4.2.

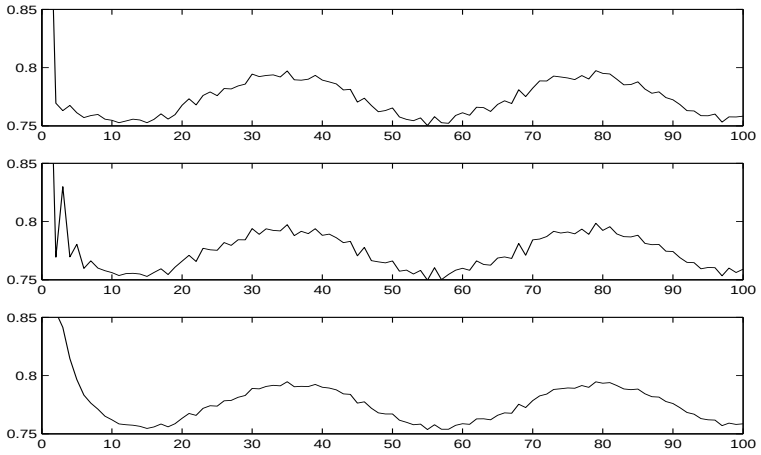

Figure 4.3: *Closed loop stepsize sequences.* The graphs show a simulation of a 100-step computation when the input $\log \hat{\varphi}$ consists of an initial jump followed by a smooth sinusoidal component onto which a randomized component, e.g. representing errors in a Newton iteration, is superimposed. From top to bottom: stepsize sequence output from PI1.0 deadbeat control; PI.68.32; PI.4.2 control. The controllers respond, pointwise, to exactly the same data sequence $\log \hat{\varphi}$. The smoothing effect of the PI.4.2 is evident.

noting that individual methods have somewhat different characteristics and that the parameter choice is a tradeoff between different objectives.

The character of the closed loop dynamics is investigated both in the time and frequency domain. In the time domain, *impulse and step responses* are used to study the transient behaviour for various control parameters, see Fig. 4.2. Likewise, time domain simulation is used to assess the controller's ability to follow and compensate a more complicated input signal $\log \hat{\varphi}$, see Fig. 4.3.

A more common technique for investigating the closed loop behaviour when subjected to external disturbances is to investigate the *spectral properties* in the frequency domain. The most elementary approach consists of considering (4.12) in its difference equation form (4.17), and studying the error output $\log \hat{r}_n$ in response to a "periodic" external disturbance input $\log \hat{\varphi}_n = \cos \omega n$ for frequencies $\omega \in [0, \pi]$. This interval covers all possible frequencies, as $\omega = 0$ corresponds to constant functions, and $\omega = \pi$ corresponds to the oscillation $(-1)^n$, which is the highest frequency that can be resolved according to the *sampling theorem*. As the closed loop difference equation is linear (even if a nonlinear ODE is being solved), *there is no intermodulation*, implying that input and output frequencies are identical, also when several different frequencies are superimposed; it therefore suffices to consider one frequency at a time. The particular solution of (4.17) is then $\log \hat{r}_n = A(\omega) \cos(\omega n + \psi(\omega))$, where the output amplitude $A(\omega)$ and phase $\psi(\omega)$ in general will depend on the frequency ω. The *frequency re-*

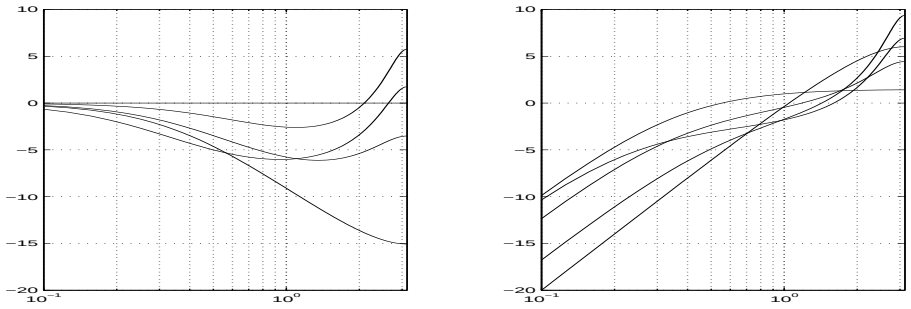

Figure 4.4: *Closed loop frequency response.* Frequency response is commonly measured in dB and plotted in log-log diagrams, here for frequencies $\omega \in [0.1, \pi]$. The left diagram shows *stepsize* frequency response $20 \log^{10} |k \cdot H^0_{\hat{\varphi}}(\mathrm{e}^{i\omega})|$. The scaling factor k is included to make the response depend only on $(kk_{\mathrm{I}}, kk_{\mathrm{P}})$. The right diagram shows the *error* frequency response $20 \log^{10} |G^0_{\hat{\varphi}}(\mathrm{e}^{i\omega})|$. From top to bottom at $\omega = \pi$ the graphs are, in both diagrams: PI.68.32; PI.3.4; deadbeat; PI.4.2; PI.3.0 control. The left diagram shows that only the PI.4.2 and PI.3.0 attenuate $(-1)^n$ stepsize oscillations (-3.5 dB and -15 dB, respectively, at $\omega = \pi$); by contrast the PI.68.32 has a significant amplification (almost $+6$ dB, or a factor of 2) of such oscillations. The purely integrating low-gain controller PI.3.0 is strongly averaging (smoothing) and may be suitable for stochastic differential equations, where $\log \hat{\varphi}$ has a significant high frequency content. The right diagram shows that $|G^0_{\hat{\varphi}}(\mathrm{e}^{i\omega})| = O(\omega/(kk_{\mathrm{I}}))$ as $\omega \to 0$, demonstrating the PI controller's first order adaptivity, see the order conditions in Section 4.3.

sponse is $|A(\omega)|$; it measures if a certain frequency's output amplitude has been amplified or attenuated. The same approach can also be applied to (4.13) and we then distinguish between the *error* and *stepsize* frequency responses.

Mathematically, however, it is more elegant (and less cumbersome) to follow the control theoretic practice. The frequency responses are then derived directly from the transfer maps $H^0_{\hat{\varphi}}(q)$ and $G^0_{\hat{\varphi}}(q)$, interpreted as z *transforms*, for $q = \mathrm{e}^{i\omega}$ with $\omega \in [0, \pi]$. The error frequency response is then $|G^0_{\hat{\varphi}}(\mathrm{e}^{i\omega})|$, and the stepsize frequency response is $|H^0_{\hat{\varphi}}(\mathrm{e}^{i\omega})|$, where

$$-k \cdot H^0_{\hat{\varphi}}(q) = \frac{(kk_{\mathrm{I}} + kk_{\mathrm{P}})q - kk_{\mathrm{P}}}{q^2 - (1 - kk_{\mathrm{I}} - kk_{\mathrm{P}})q - kk_{\mathrm{P}}}. \tag{4.20}$$

In Fig. 4.4, the stepsize and error frequency responses are investigated for various choices of PI control parameters. In particular, it is of importance that the transfer maps have an ability to attenuate $(-1)^n$ oscillations; this suggests choosing $kk_{\mathrm{I}} + kk_{\mathrm{P}}$ small.

Gustafsson [4] also carried out extensive additional tests with the safety factor θ, showing that one can run as close as 90% of TOL before stepsize rejections become frequent. For an increased margin of robustness at a negligible cost, a value of $\theta = 0.8$ is recommended, implying a setpoint of $\varepsilon = 0.8 \cdot \mathrm{TOL}$. Thus, a

good overall performance can be expected from the PI.3.4 controller

$$h_{n+1} = \left(\frac{0.8 \cdot \mathrm{TOL}}{\hat{r}_{n+1}}\right)^{0.3/k} \left(\frac{\hat{r}_n}{\hat{r}_{n+1}}\right)^{0.4/k} h_n, \tag{4.21}$$

with the elementary control (2.7)—the *PI1.0*—as a safety net in case of successive rejected steps, [4, p. 546]. The cause for successive rejected steps may often be that the asymptotic model has broken down and/or that the order k differs from its "theoretical" value, e.g. by *order reduction*. The control implementation should therefore include an estimator of the order k in case difficulties are encountered, see the pseudo-code in [4]. As the controller is parameterized in terms of kk_{I} and kk_{P}, an un-noticed change in k is in effect equivalent to a change of controllers that one would never have accepted: if k would drop from say 4 to 1 while the PI.3.4 is in use, then one is actually using a PI controller with $(kk_{\mathrm{I}}, kk_{\mathrm{P}}) = (0.075, 0.1)$ for the correct (unknown) value of k. Such parameter changes will degrade or even destroy closed loop dynamics. But before one concludes, from computational experience, that a particular control design should be discarded, one must *make sure that the process model is correct* as most difficulties are due to a poor process model.

In many codes it is common to use additional logic, e.g. a *deadzone* inhibiting stepsize increases unless they exceed say 20%, see Fig. 2.1. But such a deadzone is also detrimental to controller performance as it is equivalent to disengaging the controller from the process, waiting for large control errors to build up[4]. When the controller is eventually employed it is forced to use larger actions and might even find itself out of bounds. By contrast, the advantage of using a control design such as the PI.3.4 or the smoother PI.4.2 lies in its ability to exert a delicate stabilizing control on the process and regularize the stepsize sequence as is demonstrated in Fig. 4.3. We therefore advocate a continual control action.

5 Predictive control in stiff computation: Second order adaptivity

In stiff computations the PI control shows few clear advantages over (2.7). The main problems are still the process assumptions: (*i*) stepsizes "far ouside" the asymptotic regime for stiff, decayed solution components, and (*ii*) substantial changes in $\log \hat{\varphi}_n$. In the nonstiff case the asymptotic model is predominantly correct except near the stability boundary, and we have seen that the PI control is then effective as a countermeasure for variations in $\log \hat{\varphi}_n$. As long as the asymptotic assumption holds, h is essentially only following the changes in $\log \hat{\varphi}$. By modelling and predicting these variations, a second order controller (rather than the first order PI controller) could be employed. This will, however, typically require that changes in the process model such as (4.7) are avoided, and it is desirable that the stability region S is unbounded. A controller of second order adaptivity is therefore better suited to stiff computations, where it can offer improved external disturbance rejection.

[4]Should an auto-pilot be disengaged until serious action must be taken?

The prediction is done by constructing an *observer* [20] for $\log \hat{\varphi}_n$. Using the observer estimate to select the stepsize, Gustafsson [5, 8, p. 124] arrived at the *predictive controller*

$$\frac{h_{n+1}}{h_n} = \left(\frac{\varepsilon}{\hat{r}_{n+1}}\right)^{k_{\mathrm{E}}} \left(\frac{\hat{r}_n}{\hat{r}_{n+1}}\right)^{k_{\mathrm{R}}} \frac{h_n}{h_{n-1}}, \tag{5.1}$$

with a recommended *observer gain vector* $(kk_{\mathrm{E}}, kk_{\mathrm{R}})^{\mathrm{T}} = (1,1)^{\mathrm{T}}$. The control structure is clearly recognized as a *PI controller acting through the stepsize difference* $\log h_{n+1} - \log h_n$ to control the error. This may appear to be a minor modification but the dynamics is different, making it less suitable for nonstiff methods. Interestingly, similar controllers based on extrapolating error trends and stepsizes were first suggested and used in [18, 19], although without an analysis of controller dynamics and parameterization.

When an implicit Runge–Kutta method is used to solve DAEs or stiff ODEs, the behaviour of the local error estimate is of fundamental importance for successful control. Two central notions in control theory are *observability* and *controllability* [20]. Without going into details, in our context observability requires that we have an error estimate which adequately reflects the true stepsize–error relation, not only in the asymptotic regime but also outside, for decayed stiff solution components or algebraic variables. Controllability requires that the stepsize be an effective means to exert error control: by adjusting h_n we must be able to put the error at a prescribed level of ε. Neither of these requirements is trivial and only a brief account can be given here. To this end, we consider the linear test equation $\dot{y} = \lambda y$. The method yields $y_{n+1} = R(z_n)y_n$, where $R(z_n)$ is a rational function, typically a Padé approximation to e^{z_n}. For dissipative problems it is today common to select L-stable methods, e.g. the Radau IIa methods [8], for which the stability region S contains the left half-plane and in addition $R(\infty) = 0$. Given an embedded formula with rational function $\hat{R}(z)$, an error estimate is provided by $\hat{l}_{n+1} = (R(z_n) - \hat{R}(z_n))y_n$. Here $\hat{R}(z)$ must be bounded as $z \to \infty$. But it is also important that the estimated error for very stiff or algebraic solution components essentially vanishes[5]. Unless such conditions are met, the stepsize–error relation may suddenly break down, forcing stepsize reductions by several orders of magnitude to re-establish an asymptotic relation [8, pp. 113–114]. A simplified explanation is an unsuitable choice of R and $\hat{R}$ for which $(R(z_n) - \hat{R}(z_n))y_n$ is almost *constant* as a function of z_n when $|z_n|$ is large, i.e., *the error estimate is unaffected by moderate stepsize variations.* In such a situation any controller—no matter how sophisticated—is at a loss of *control authority* as it is unable to affect the process output through its control variable h. In order to avoid this breakdown one must seek method/error estimator combinations which offer controllability; other combinations can be discarded as inappropriate for efficient adaptive stiff computations. This is still an area of active research, however, as it must address method properties, error estimator construction (see e.g. [16]) both for stiff ODEs and DAEs, termination criteria for Newton iterations, and problem properties in DAEs.

[5] A sufficient condition is that both of $R(z)$ and $\hat{R}(z)$ are L-stable.

5.1 Process and closed loop

In the sequel we shall assume that the error estimator is appropriately constructed and free of the shortcomings mentioned above. This implies, with few exceptions, that the controller and error estimator operate in the asymptotic regime, i.e., the solution components or modes to be controlled satisfy $\hat{r}_{n+1} = \hat{\varphi}_n h_n^k$. Stiff or algebraic components, which are outside the asymptotic regime, typically give negligible contributions to the error estimate, cf. [8, p. 125]. Thus our *process model* is

$$\log \hat{r} = kq^{-1} \log h + q^{-1} \log \hat{\varphi}. \tag{5.2}$$

When this model holds, the error control problem is a matter of adapting the stepsize to variations in $\hat{\varphi}_n$. Investigations of real computational data show that variations in $\hat{\varphi}_n$ have a lot of structure [5, p. 503]. The simplest model for predicting $\log \hat{\varphi}_n$ is the *linear extrapolation*

$$\log \tilde{\varphi}_n = 2 \log \hat{\varphi}_{n-1} - \log \hat{\varphi}_{n-2} \tag{5.3}$$

where $\log \tilde{\varphi}_n$ denotes the predicted value of $\log \hat{\varphi}_n$. Note that the model is not compensated for stepsize variations by using divided differences. Practical testing of such models have not shown significant advantages over (5.3). Using the forward shift q we rewrite (5.3) in the form

$$\log \tilde{\varphi} = (2q - 1)q^{-2} \log \hat{\varphi}. \tag{5.4}$$

We shall choose h_n such that $\tilde{\varphi}_n h_n^k = \varepsilon$. This implies that h_n is immediately counteracting the predicted disturbance and will lead to a deadbeat control law; the stepsize is then given by

$$\log h = k^{-1}(\log \varepsilon - \log \tilde{\varphi}). \tag{5.5}$$

Inserting the estimate (5.4) into (5.5), noting that $\log \varepsilon$ is constant, we obtain the *closed loop dynamics*

$$\log h = \frac{2q - 1}{kq^2}(\log \varepsilon - \log \hat{\varphi}). \tag{5.6}$$

The double pole at $q = 0$ shows that we have indeed obtained a *deadbeat control* analogous to (3.4).

5.2 Controller

To find the controller that will produce the requested dynamics (5.6), we use the asymptotic process model (5.2) to eliminate $\log \hat{\varphi}$ from (5.6) and obtain, after rearranging terms,

$$\log h = \frac{1}{k} \frac{q}{q - 1} \left(\frac{q}{q - 1} + 1 \right) (\log \varepsilon - \log \hat{r}). \tag{5.7}$$

Thus $\log h = G_{\mathrm{C}}^{\mathrm{PC}}(q) \cdot (\log \varepsilon - \log \hat{r})$; consequently the *predictive controller* is

$$G_{\mathrm{C}}^{\mathrm{PC}}(q) = \frac{1}{k}\,\frac{q}{q-1}\left(\frac{q}{q-1}+1\right) = \frac{q}{q-1}\left(k_{\mathrm{E}}\,\frac{q}{q-1}+k_{\mathrm{R}}\right), \tag{5.8}$$

where $(kk_{\mathrm{E}}, kk_{\mathrm{R}}) = (1,1)$ in the derivation of the deadbeat control above. But other parameterizations may also be considered, as indicated by the parameters k_{E} and k_{R} in the general expression for the predictive controller. Two structural properties are to be noted: first, the usual PI control structure (4.9) is recognized as a part of the controller, with integral and proportional gains of k_{E} and k_{R}; second, the controller's transfer function contains a *double integral action*; this is known to be necessary to follow a linear trend without control error.

5.3 Closed loop and control parameters

In order to find the recursion formula (5.1) we rearrange (5.7), with general parameters $(k_{\mathrm{E}}, k_{\mathrm{R}})$, in the form

$$\frac{q-1}{q}\,\log h = \left(k_{\mathrm{E}}\,\frac{q}{q-1}+k_{\mathrm{R}}\right)(\log \varepsilon - \log \hat{r}), \tag{5.9}$$

Since the quantity on the left-hand side corresponds to $\log(h_n/h_{n-1})$, the formula (5.1) follows, and with $\theta = 0.8$ and $(kk_{\mathrm{E}}, kk_{\mathrm{R}}) = (1,1)$ we have obtained the *PC11 controller*,

$$h_{n+1} = \left(\frac{0.8\cdot \mathrm{TOL}}{\hat{r}_{n+1}}\right)^{1/k}\left(\frac{\hat{r}_n}{\hat{r}_{n+1}}\right)^{1/k}\frac{h_n}{h_{n-1}}\,h_n\,, \tag{5.10}$$

recommended by Gustafsson [5]. For a detailed discussion of a general observer for $\log \hat{\varphi}$ and the choice of gain parameters $(kk_{\mathrm{E}}, kk_{\mathrm{R}})$ we refer to [5, p. 512].

The purpose of changing $(kk_{\mathrm{E}}, kk_{\mathrm{R}})$ is to introduce dynamics in the estimation so that $\log \tilde{\varphi}_n$ depends on its own *history* as well as the present values of $\log \hat{\varphi}_n$ and the backward difference $(1-q^{-1})\log \hat{\varphi}_n$. By contrast, (5.6) shows that in the deadbeat PC11, $\log h_{n+1}$ is directly proportional to $\log \hat{\varphi}_n$ and $(1-q^{-1})\log \hat{\varphi}_n$, making it sensitive to fluctuations in $\log \hat{\varphi}$. In stiff computations irregularities are quite common as *the error estimate is also influenced by an irregular error contribution from truncated Newton iterations*. The controller should nevertheless be able to produce a fairly regular stepsize sequence, see Fig. 5.1.

For a lower gain than $(1,1)$, the controller becomes slower than the PC11, but the convolution operator mapping $\log \hat{\varphi}$ to $\log h$ again acts as a mollifier and smoother stepsize sequences are obtained. To investigate parameterizations of the PC controller in terms of their frequency responses, we seek the closed loop transfer maps $G_{\hat{\varphi}}^0(q) : \log \hat{\varphi} \mapsto \log \hat{r}$ and $H_{\hat{\varphi}}^0(q) : \log \hat{\varphi} \mapsto \log h$. They are obtained by inserting the general controller (5.8) and asymptotic process (4.1) into the general closed loop transfer maps (4.14) and (4.15), to find

$$G_{\hat{\varphi}}^0(q) = \frac{q^{-1}(q-1)^2}{q^2 - (2 - kk_{\mathrm{E}} - kk_{\mathrm{R}})q + (1 - kk_{\mathrm{R}})} \tag{5.11}$$

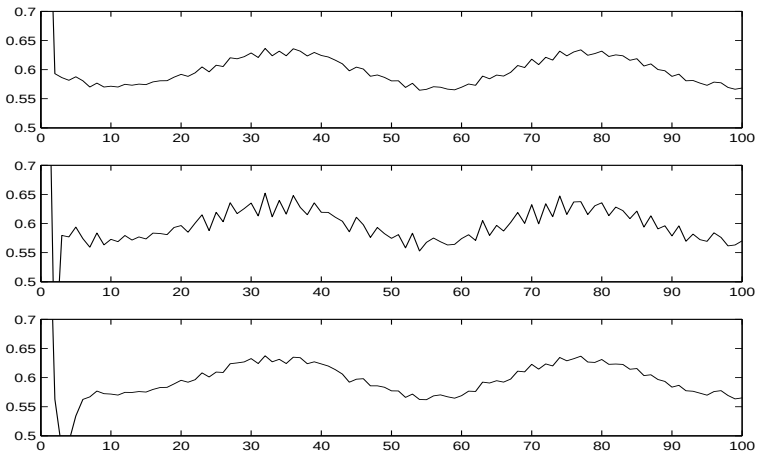

Figure 5.1: *Closed loop stepsize sequences.* The setup of this 100-step simulation is similar to that of Fig. 4.3. The graphs show the stepsize sequence output in response to a data sequence $\log\hat{\varphi}$, consisting of a regular behaviour onto which random fluctuations, modelling the effect of Newton iteration errors on the error estimate, have been superimposed. From top to bottom: classical 1st order PI1.0 deadbeat control as a reference standard; 2nd order PC11 deadbeat control; 2nd order PC.4.7 control. While the PC11 is sensitive to irregular input, the PC.4.7 achieves a smoothness on par with that of PI1.0. Both predictive controllers exhibit a significant overshoot at the initial transient.

and

$$-k\cdot H^0_{\hat{\varphi}}(q) = \frac{(kk_{\rm E}+kk_{\rm R})q - kk_{\rm R}}{q^2-(2-kk_{\rm E}-kk_{\rm R})q+(1-kk_{\rm R})}. \tag{5.12}$$

$G^0_{\hat{\varphi}}(q)$ now has a *double* difference operator in the numerator reflecting the controller's double integral action, see Fig. 5.2. The controller's order of adaptivity is therefore 2, enabling it to follow linear trends without error.

The poles determine the stability. As the dynamics of the PC closed loop is different from that of the usual PI controller, the parameters in the previous section are not relevant in this context. Pole positioning plays an important role. Because the product of the poles is $1-kk_{\rm R}$ (see (5.12)), stability requires that $kk_{\rm R}\in[0,2]$. In most cases, it appears preferable to take $kk_{\rm R}\lesssim 1$, and for smoothness one then typically needs to take $k_{\rm E}<k_{\rm R}$. The poles then become complex conjugate; some are listed below:

Controller	*Poles*	*Modulus*
PC11	0, 0	(deadbeat)
PC.6.9	$.25\pm.19i$	0.32
PC.5.8	$.35\pm.28i$	0.45
PC.4.7	$.45\pm.31i$	0.55
PC.3.6	$.55\pm.31i$	0.63

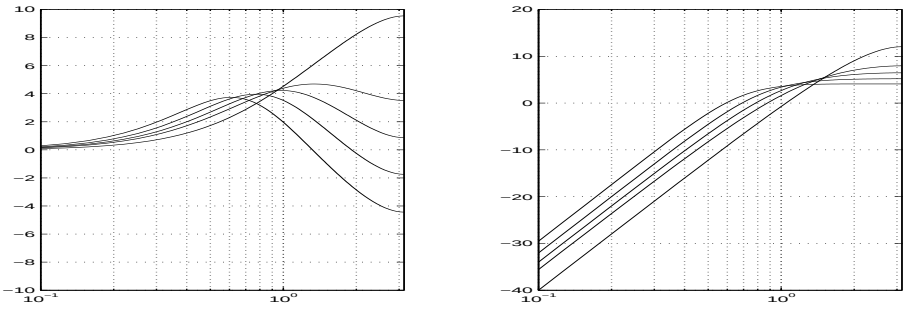

Figure 5.2: *Closed loop frequency response.* Stepsize frequency response $20\log^{10}|k\cdot H^0_{\hat\varphi}(\mathrm{e}^{i\omega})|$ (left) and error frequency response $20\log^{10}|G^0_{\hat\varphi}(\mathrm{e}^{i\omega})|$ (right) for frequencies $\omega\in[0.1,\pi]$. From top to bottom at $\omega=\pi$ the graphs are, in both diagrams: PC11 deadbeat; PC.6.9; PC.5.8; PC.4.7; PC.3.6 control. The PC11 has nearly +10 dB amplification of $(-1)^n$ stepsize oscillations (left diagram, $\omega=\pi$). Only the PC.4.7 and PC.3.6 controllers attenuate such oscillations; having poles of larger modulus, they are both slower and smoother. $G^0_{\hat\varphi}(q)$ contains a double (or second order) difference operator, see (5.11). As a consequence $|G^0_{\hat\varphi}(\mathrm{e}^{i\omega})|=O(\omega^2/(kk_{\mathrm{E}}))$ as $\omega\to 0$, causing the 40 dB/decade slopes characteristic of second order adaptivity, at the low frequency end of the error response graphs (right diagram). By contrast, a PI controller's single integral action only results in first order, 20 dB/decade slopes, see Fig. 4.4. A steeper slope implies stronger suppression of low-frequent variations in $\log\hat\varphi$. The error therefore stays closer to the setpoint $\varepsilon=\theta\cdot\mathrm{TOL}$.

Because of the complex conjugate poles one can expect to find a "resonance peak" in the stepsize frequency response. The left diagram in Fig. 5.2 shows that this is indeed the case.

A complete pseudocode of the PC11 controller (useful also for PC.4.7, say), including exception handling and estimation of k in case of order reduction is provided in [5, p. 511].

6 Stiff computation: Matrix strategies

In stiff ODE computations implicit methods are used. It is therefore necessary to employ iterative equation solvers on each step, usually of Newton type, to solve equations of the generic form

$$y_n=\gamma h_n f(y_n)+\psi, \tag{6.1}$$

where γ is a method-dependent constant and ψ is a known vector. The Newton iteration requires that we solve linear systems using the matrix $I-\gamma h_n J_n$, where J_n is an approximation to the Jacobian $f'(y_n)$. As the iteration matrix depends on h_n, it is usually argued that minor stepsize variations might force matrix refactorizations and should be avoided. As a remedy the elementary controller is often used in combination with a deadzone prohibiting small stepsize

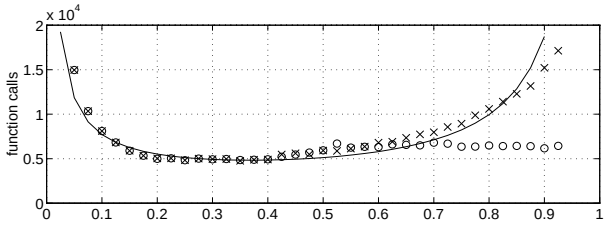

Figure 6.1: *Work per unit time as a function of convergence rate* α. Total work when a nonstiff problem was solved with an implicit method employing fixed-point iterations is plotted as a function of the convergence rate α. Solid line represents theoretical work, while $\times$ and $\circ$ denote data obtained in a real computation with different max iterations/step settings. As total work decreases only up to $\alpha < 0.2$, slower convergence rates never pay off. From [6].

increases, aiming to keep h_n piecewise constant. Stepsize decreases, on the other hand, are often accepted even when small, and may be controlled by a different strategy—as an example RADAU5 uses the elementary control (2.7) with a 20% deadzone for stepsize increases, and the predictive control (5.10) for decreases [8, p. 124], see Fig. 2.1. This "gain scheduling" makes the control unsymmetric, discontinuous and nonlinear, and yet it does not save factorizations during long sequences of shrinking steps which do occur in practice, see Fig. 6.2. Instead of treating stespize increments and decrements differently, a logic consistent with a 20% deadzone for increases should immediately effect a 20% decrease as soon as the error estimate exceeds ε, see e.g. the strategy in DASSL, Fig. 2.1. Such a "staircase" strategy, however, has more in common with step doubling/halving than with actual control. In addition its control actions are often large enough to cause transient effects, where process models no longer hold, resulting in an increased risk of step rejections. The lack of smooth control may also cause an erratic tolerance proportionality.

We advocate the use of a smooth, symmetric, linear control, allowed to work with minute adjustments of h_n on every step. The question is if we can allow a continual control action without an excessive number of matrix refactorizations. Because one normally uses modified Newton iteration, J_n is already in error, and small changes in h_n can be accomodated by viewing $h_n J_n$ as being approximate, as long as the convergence rate does not deteriorate.

We shall only discuss Newton iterations. For a discussion of nonstiff computations and fixed-point iterations for (6.1) we refer to [6]. The convergence rate α is then proportional to h_n. Stepsize increases therefore cause slower rates of convergence, which are acceptable as long as larger steps reduce total work per unit time of integration. But as work is, in theory, proportional to $-(\alpha \log \alpha)^{-1}$, [6, p. 27], the convergence rate should never exceed $\mathrm{e}^{-1} \approx 0.37$; beyond this point stepsize increases are counterproductive as total work per unit time of integration starts *growing*, see Fig. 6.1. As the minimum is very flat, one should in

practice restrain h_n further, to ensure that $\alpha < 0.2$. This method-independent stepsize limitation will therefore have to be coordinated with the controller, and becomes part of the complete controller specification.

As for Newton iterations, an analysis of efficiency in terms of convergence rates is extremely complicated and depends on problem size, complexity and degree of nonlinearity [13]. This model investigation indicates that $\alpha = 0.2$ is an acceptable rate for low precision computations but that rates as fast as $\alpha = 0.02$ may be preferred if the accuracy requirement is high. Since we normally want to keep the same Jacobian for several steps, as well as use the same Jacobian for all stage values in (2.2), a convergence rate of $\alpha = 0.1$ must generally be allowed or the strategy may break down when faced by a strongly nonlinear problem. In fact, using an upper bound of $\alpha = 0.2$ will only lose efficiency if this rate can be maintained for a considerable time using an old Jacobian [13]. A convergence rate of $\alpha = 0.2$ is therefore not only feasible but in many cases also acceptable.

Let us consider a modified Newton iteration and assume that the Jacobian J_m was computed at the point (t_m, y_m), when the stepsize h_m was used. Further, we assume that the iteration matrix $I - \gamma h_m J_m$ is still in use at time t_n, when actual values of stepsize and Jacobian have changed to $h_n J_n = h_m J_m + \delta(hJ)$. It is then straightforward to show that the iteration error e_j in the modified Newton method satisfies, up to first order terms,

$$e_{j+1} = (I - \gamma h_m J_m)^{-1} \gamma \, \delta(hJ) \, e_j. \tag{6.2}$$

We shall find an estimate of α in terms of the relative deviation in $h_n J_n$ from $h_m J_m$. Assuming that J_m^{-1} exists (this is no restriction), we rewrite (6.2) as

$$e_{j+1} = (I - \gamma h_m J_m)^{-1} \gamma h_m J_m (h_m J_m)^{-1} \delta(hJ) e_j. \tag{6.3}$$

Taking norms yields $\|e_{j+1}\| \leq \nu \cdot \|(h_m J_m)^{-1} \delta(hJ)\| \, \|e_j\|$, where for inner product norms $\nu = \|(I - \gamma h_m J_m)^{-1} \gamma h_m J_m\| \to 1$ in the presence of stiffness, i.e., as $\|h_m J_m\|$ grows large [6, p. 29]. In fact, if the logarithmic norm $\mu[\gamma h_m J_m] \leq 1/2$, then $\nu \leq 1$; a significant deviation $\nu \ll 1$ occurs only in the nonstiff case when $\|h_m J_m\| \ll 1$. We can therefore estimate the convergence rate by

$$\alpha \lesssim \|(h_m J_m)^{-1} \delta(hJ)\|. \tag{6.4}$$

For small variations around h_m and J_m we approximate $\delta(hJ) \approx J_m \delta h + h_m \delta J$, which together with (6.4) yields the bound

$$\alpha \lesssim \| \frac{\delta h}{h_m} I + J_m^{-1} \delta J \| \leq | \frac{\delta h}{h_m} | + \|J_m^{-1} \delta J\|. \tag{6.5}$$

Therefore the convergence rate is bounded by the sum of the relative changes in stepsize and Jacobian, respectively. This simple formula is useful for assessing the need for refactorizations and reevaluations of the Jacobian. Thus we see that if variations in J are negligible, then we can accept a 20% variation in stepsize without exceeding $\alpha = 0.2$; in other words, *a 20% deadzone in the stepsize strategy is unnecessary.* Conversely, if without stepsize change the estimated

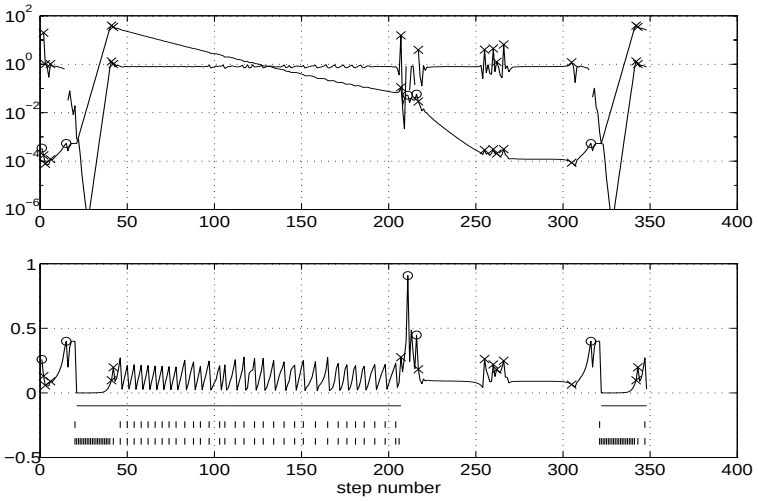

Figure 6.2: *Interaction of iteration and stepsize strategies.* The upper diagram shows stepsise h and error excess r/TOL when solving a stiff van der Pol problem; r/TOL is kept very close to $\theta = 0.8$. The lower diagram shows convergence rate α and how it is affected by the strategy described in Section 6. When α is likely to exceed the maximum, the strategy forces either a Jacobian reevaluation or an iteration matrix refactorization, without convergence failures. Jacobian evaluations are indicated by the upper tick marks, matrix refactorizations by the lower tick marks. Solid line indicates Newton iterations; during nonstiff phases, fixed-point iterations were used. During most of the stiff phase (steps 40–205), convergence rate slows down due to a changing Jacobian. By contrast, when the stepsize is increased after passing the turning point (steps 320–345), refactorizations are often necessary, but the same Jacobian can be kept. There is a total of 4% rejected steps (marked $\times$) and 1.5% convergence failures (all during nonstiff phase, marked $\circ$) in the computation. From [6].

convergence rate grows and exceeds the acceptable level of $\alpha = 0.2$, then (6.5) demonstrates that this can only be due to a relative change in the Jacobian, calling for a re-evaluation of J. Note that there is no need to compute $\|J_m^{-1}\delta J\|$. It is sufficient to monitor the convergence rate α and the accumulated stepsize change $|\delta h/h_m|$. Moreover, if a stepsize change is suggested by the controller, one can beforehand find out if the proposed change implies that a refactorization is necessary; it is not necessary to wait for a convergence failure before invoking counter-measures. This local strategy has been thoroughly tested in [6] where a full pseudocode specification can be found. Further improvements have been added by de Swart [15, p. 160]. The purpose of these strategies is to establish a full coordination of stepsize control and iterative method, see Fig. 6.2. This can be achieved for various choices of maximum acceptable convergence rate.

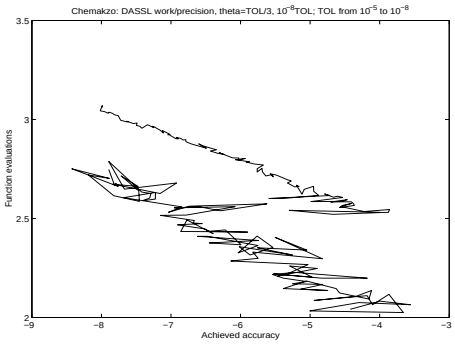

Figure 6.3: *Newton termination criteria and computational stability.* An ODE was solved using DASSL. Original termination criterion for the Newton iteration is TOL/3 and yields the lower work/precision graph. When the termination criterion is tightened to 10^{-8}TOL (in practice full precision), the upper graph is obtained. It displays a significantly improved computational stability. Work increase, in terms of f evaluations, is moderate (approximately a factor of 2).

The total work of the iteration depends to a large extent on starting values and termination criterion. *Starting values* are obtained from some predictor, and a more accurate predictor implies better efficiency. A high order predictor is a significant advantage but may be difficult to construct if an implicit Runge–Kutta method has low stage order. New second order predictors were developed in [13], indicating that overall performance increases of $10-20\%$ may be obtained, in particular for high precision computations.

Termination criteria can be formulated in different ways. One requirement is that the stage derivatives $\dot{Y}_i$ must be sufficiently accurate, as these are the values entering the quadrature formula (2.3). Moreover, the terminal iteration errors must not make more than an insignificant contribution to the local error *and* its estimate. The iteration error is less regular than the truncation error of the discretization, and may—if too large—cause an irregular behaviour in $\log\hat{\varphi}$, i.e. the controller will be fed "noisy data," see Fig. 6.3. The predictor (5.3) is then prone to be erratic and the PC11 is then no longer able to achieve better control. If the correct remedy of a sharper termination criterion is unacceptable, the remaining possibilities are either to give up deadbeat control and replace the PC11 by e.g. the PC.4.7 or to resort to the PI1.0, which is based on the assumption of slow variation, equivalent to a 0th order predictor for $\log\hat{\varphi}$. In the coordination of controller and iteration strategy, these considerations need to be accounted for to obtain a coherent performance.

7 Conclusions

The numerical integration of ODEs can be viewed as a time-discrete process which can be made adaptive by using techniques offered in the modern theory

of digital control. Section 4 showed that the computational process is an affine map, i.e. $\log \hat{r} = G_{\mathrm{P}}(q) \log h + q^{-1} \log \hat{\varphi}$, from $\log h$ to a corresponding error estimate $\log \hat{r}$. In the asymptotic regime it is static—$G_{\mathrm{P}}(q)$ is just a constant. But when $\log h$ becomes large the process representing an explicit method shifts to become dynamic. Both cases could be controlled by a standard technique, the PI controller. This is a linear map $\log h = G_{\mathrm{C}}(q)(\log \varepsilon - \log \hat{r})$ which finds a suitable stepsize from the deviation between accuracy requirement and estimated error. The recommended controllers, PI.3.4 and PI.4.2, are fully specified in the form of pseudocode in [4].

For implicit methods, Section 5 presented a more advanced technique based on predicting the evolution of the principal error function. The process $G_{\mathrm{P}}(q)$ was still just a constant, but the predictive controller had an increased complexity, enabling it to follow linear trends. A full specification of the recommended PC11 and PC.4.7 controller is found in [5]. Finally, in Section 6 continual control action was coordinated with matrix strategies for the Newton iterations; algorithm specifications are found in [15, p. 160], which improves the original in [6].

The new control algorithms are efficient, yield smoother stepsize sequences and fewer rejected steps, and are designed to be parts of a coherent strategy. Even if they are more complex and mainly directed towards qualitative improvements, the new algorithms may often decrease total work as well. These control algorithms are supported by a well established mathematical methodology, and should be considered as well-defined, fully specified structures accomplishing equally specific tasks. Although one may wish to use different controllers in different situations, the tuning of a controller relies in equal parts on control theoretic principles and a thorough knowledge of the process to be controlled; parameter choice must be consistent and is not arbitrary. If properly done, important quality aspects such as tighter tolerance proportionality and an improved computational stability may be achieved.

The recent development and present state of the art point to the possibility of eliminating heuristic elements from adaptive solution methods for ODEs and DAEs. This is an important goal as it may eventually bring ODE/DAE software to approach the level of quality and standardization found in modern linear algebra software.

Yet there are many open problems. The theory presented here is "single input, single ouput", which is simpler to treat from the control theory point of view. For multistep methods, it is known that the process dynamics depends on past stepsize sequences, making it a multiple input process. But while there is a considerable knowledge about stability limitations on stepsize ratios in multistep methods, it is generally not appreciated that a good controller can stabilize even an unstable process. For example, DASSL only doubles the stepsize—this is known to be beyond the permissible stepsize ratio for BDF methods—but it does not go unstable; even the elementary controller keeps instability at bay. But in order to control a multistep method well, one needs an accurate dynamic model for how stepsizes affect the process. Such models are not available today.

Another open problem is how to properly estimate the error in DAEs. Higher

index problems are sensitive to the smoothness of residuals, so there is a need to construct proper error estimators that reflect problem properties correctly. This is related to the problems of observability and controllability. Today, it is not uncommon to "filter" poor error estimators by multiplying the estimate by a sufficient number of powers of the iteration matrix, to quench undesirable components in the error estimate. That is more of an emergency remedy, and hopefully better techniques can be found. The control-based adaptivity surveyed in the present paper works because *process models and controllers are mathematically analyzed and experimentally verified*—much less can be expected from a heuristic approach.

Acknowledgements. The material of this paper was presented as part of a series of invited lectures at the ANODE01 meeting in Auckland, New Zealand. The author is grateful to Prof. John Butcher for the invitation to that meeting and for arranging an extended stay at the University of Auckland, where this paper was written. The author would also like to thank a large number of colleagues and collaborators in numerical analysis and automatic control who over many years have contributed directly as well as indirectly to shaping the techniques presented in this review. The research was in part funded by the Swedish Research Council for Engineering Sciences TFR contract 222/98-74.

REFERENCES

1. P. DEUFLHARD AND F. BORNEMANN. Numerische Mathematik II: Integration gewöhnlicher Differentialgleichungen. Walter de Gruyter, Berlin 1994.
2. C.W. GEAR. Numerical Initial Value Problems in Ordinary Differential Equations. Prentice–Hall, Englewood Cliffs 1971.
3. K. GUSTAFSSON, M. LUNDH AND G. SÖDERLIND. A PI stepsize control for the numerical solution of ordinary differential equations. *BIT* 28:270–287, 1988.
4. K. GUSTAFSSON. Control theoretic techniques for stepsize selection in explicit Runge–Kutta methods. *ACM TOMS* 17:533–554, 1991.
5. K. GUSTAFSSON. Control theoretic techniques for stepsize selection in implicit Runge–Kutta methods. *ACM TOMS* 20:496–517, 1994.
6. K. GUSTAFSSON AND G. SÖDERLIND. Control strategies for the iterative solution of nonlinear equations in ODE solvers. *SIAM J. Sci. Comp.* 18:23–40, 1997.
7. E. HAIRER, S.P. NØRSETT AND G. WANNER. Solving Ordinary Differential Equations I: Nonstiff Problems. Springer-Verlag, 2nd revised edition, Berlin 1993.
8. E. HAIRER AND G. WANNER. Solving Ordinary Differential Equations II:

Stiff and Differential-algebraic Problems. Springer-Verlag, 2nd revised edition, Berlin 1996.

9. G. Hall. Equilibrium states of Runge–Kutta schemes. *ACM TOMS* 11:289–301, 1985.
10. G. Hall. Equilibrium states of Runge–Kutta schemes, part II. *ACM TOMS* 12:183–192, 1986.
11. G. Hall and D. Higham. Analysis of stepsize selection schemes for Runge–Kutta codes. *IMA J. Num. Anal.* 8:305–310, 1988.
12. D. Higham and G. Hall. Embedded Runge–Kutta formulae with stable equilibrium states. *J. Comp. and Appl. Math.* 29:25–33, 1990.
13. H. Olsson and G. Söderlind. Stage value predictors and efficient Newton iterations in implicit Runge–Kutta methods. *SIAM J. Sci. Comp.* 19:, 1998.
14. H. Olsson and G. Söderlind. The approximate Runge–Kutta computational process. *BIT* 40:, 2000.
15. J.J.B. de Swart. Parallel software for implicit differential equations. Ph.D. thesis, CWI, Amsterdam 1997.
16. J.J.B. de Swart and G. Söderlind. On the construction of error estimators for implicit Runge–Kutta methods. *J. Comp. and Appl. Math.* 86:347–358, 1997
17. T. Söderström and P. Stoica. System Identification. Prentice–Hall, Englewood Cliffs 1989.
18. H.A. Watts. Step size control in ordinary differential equation solvers. *Trans. Soc. Comput. Sim.* 1:15–25, 1984
19. J.A. Zonneveld. Automatic numerical integration. Ph.D. thesis, Math. Centre Tracts 8, CWI, Amsterdam 1964.
20. K.J. Åström and B. Wittenmark. Computer Controlled Systems – Theory and Design. 2nd ed., Prentice–Hall, Englewood Cliffs 1990.