CWI Quarterly Volume 11 (1) 1998, pp. 55 – 74

# The Automatic Control of Numerical Integration

Gustaf Söderlind
*Department of Computer Science,*
*Lund University,*
*Box 118, S-221 00 Lund, Sweden*
*e-mail:* Gustaf.Soderlind@dna.lth.se

This paper reviews the recent advances in developing automatic control algorithms for the numerical integration of ordinary differential equations (ODEs) and differential-algebraic equations (DAEs). By varying the stepsize, the error committed in a single step of the discretization method can be affected. Modern time-stepping methods provide an estimate of this error, and by comparing the estimate to a specified accuracy requirement a control algorithm selects the next stepsize. To construct efficient controllers it is necessary to analyze the dynamic behaviour of the discretization method together with the controller. Based on feedback control theory, this systematic approach has replaced earlier heuristics and resulted in a more consistent and robust performance. Other quantities affected by the stepsize are convergence rates of fixed-point and Newton iterations, and we therefore also review new techniques for the coordination of nonlinear equation solvers with the primary stepsize controller. Taken together, the recent development provides principles and guidelines for constructing ODE/DAE software where heuristics and tuning parameters have largely been eliminated.

## 1. Introduction

Software for solving initial value problems for ODEs and DAEs has existed for the past 15–30 years. Yet there has been a quite rapid recent development in the analysis and design of the algorithmic structures needed to turn a numerical time-stepping method into an *adaptive* integration procedure.

We shall consider the problem of numerically solving an ODE

$$\dot{y} = f(y); \qquad y(0) = y_0, \quad t \geq 0\,, \tag{1}$$

where $f : \mathbf{R}^d \to \mathbf{R}^d$. 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 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.

The *work/precision efficiency* of a 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) with minimum effort, subject to a prescribed *error tolerance* `tol`. As $\texttt{tol} \to 0$, the global error $\|y_n - y(t_n)\|$ should decrease in a regular way—a property called *tolerance proportionality*. At the same time, computational efforts increase, usually in a logarithmic proportionality between accuracy and effort.

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 means of controlling the error and making the computational procedure adaptive. Thus it is a well established practice to *control the local error*, an approach which is inexpensive and far simpler than controlling the global error. By using differential inequalities, however, 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 $C(t) \cdot \texttt{tol}$. Thus a local error tolerance indirectly affects the global error.

We shall assume that an $s$-stage Runge–Kutta method characterized by the matrix-vector pair $(A, b)$ is used to solve (1). Using standard notation, [8], the method can be written

$$\begin{aligned} Y &= \mathbb{1} \otimes y_n + (A \otimes I_d) h\dot{Y} && (2)\\ \dot{Y}_i &= f(Y_i)\,; \qquad i = 1:s && (3)\\ y_{n+1} &= y_n + (b^T \otimes I) h\dot{Y}, && (4) \end{aligned}$$

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. For the purposes of this study, it is of secondary importance if the method is explicit or implicit, but we shall assume that the method also supports a reference method defined by a different quadrature formula

$$\hat{y}_{n+1} = y_n + (\hat{b}^T \otimes I) h\dot{Y}. \tag{5}$$

By means of this reference method, a *local error estimate* is obtained by

$$\hat{l}_{n+1} = y_{n+1} - \hat{y}_{n+1} = ((b^T - \hat{b}^T) \otimes I_d) h\dot{Y}. \tag{6}$$

Let $\hat{y}(t; \tau, \eta)$ denote a solution to (1) with initial condition $\hat{y}(\tau) = \eta$. Then the *local error* in a step from $y_n$ to $y_{n+1}$ may be written

$$l_{n+1} = y_{n+1} - \hat{y}(t_{n+1}; t_n, y_n). \tag{7}$$

By expanding the local error in an asymptotic series one shows that

$$l_{n+1} = \Phi(t_n, y_n)h^{p+1} + O(h^{p+2}), \tag{8}$$

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

$$\hat{l}_{n+1} = \hat{\Phi}_n h^{\hat{p}+1} + O(h^{\hat{p}+2}), \tag{9}$$

where the order $\hat{p} \leq p$, 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.

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 $\hat{r}_{n+1} = \|\hat{l}_{n+1}/h_n\|$ (we now use $h_n$ instead of $h$). A step is accepted if $\hat{r}_{n+1} \leq \texttt{tol}$, 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 for a slightly smaller error, $\varepsilon = \theta \cdot \texttt{tol}$, where $\theta < 1$ is a safety factor. The *elementary local error control* algorithm [2, p. 156] is

$$h_{n+1} = \left(\frac{\varepsilon}{\hat{r}_{n+1}}\right)^{1/k} h_n, \tag{10}$$

were $k = p+1$ (EPS) or $k = p$ (EPUS). The heuristic derivation of the control law (10) assumes the following behaviour of the integration process:

PROCESS ASSUMPTIONS

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

If these were correct, and there is a deviation between $\varepsilon$ and $\hat{r}_{n+1}$, then (10) will eliminate this deviation in a single step and make the error equal $\varepsilon$:

$$h_{n+1} = \left(\frac{\varepsilon}{\hat{\varphi}_n h_n^k}\right)^{1/k} h_n \quad \Rightarrow \quad \hat{\varphi}_n h_{n+1}^k = \varepsilon. \tag{11}$$

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;
(ii) in stiff ODEs one uses "large" stepsizes far outside the asymptotic regime for modes which have decayed;

(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. 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 quite 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 (10). As long as stability is at hand, the solution $y_n$ remains smooth and the error small, and (10) 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 (10) 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, cf. [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 (10) 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 (10) 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, which implies the opposite approach; instead of constructing methods that match the control law (10), the controller should be designed to match the method. Moreover, because (10) 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 *proportional integral (PI) controller* (see also Section 2), 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 $\theta$; 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].

The purpose of this paper is to review the recent advances in the automatic

control of numerical integration. In Section 2 we study the elementary controller (10) for nonstiff computations while introducing some methodology and basic notions from control theory. We explain the structure of the PI controller and the effect of its parameters. Section 3 then gives a more detailed account of the PI control design process and the parametrization of the controller. In Section 4 we proceed to review the development in stiff computations where the PI control is replaced by a predictive control, [5]. 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. It is therefore important to coordinate matrix strategies with the controller. In Section 5 we describe a local strategy, [6], which by monitoring and predicting convergence rates manages a good performance without interfering with the predictive controller. This reduces unnecessary matrix handling and convergence failures. We also discuss high order predictors for the starting values of the iteration, [13], which are combined with an attempt to optimize global performance based on problem characteristics. 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. PI-controlled time-stepping: Basic notions

Let us consider (10), known in control theory as a discrete-time integral controller (*I controller*). Taking logarithms in (10) yields the linear difference equation

$$\log h_{n+1} = \log h_n + \frac{1}{k}\,(\log \varepsilon - \log \hat{r}_{n+1}). \tag{12}$$

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 (12) we obtain

$$\log h_n = \log h_0 + \frac{1}{k} \sum_{m=1}^{n} (\log \varepsilon - \log \hat{r}_m). \tag{13}$$

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

We shall first examine the *closed loop dynamics* for controller and controlled process. To this end we insert the asymptotic model $\hat{r}_{n+1} = \hat{\varphi}_n h_n^k$ (i.e. the process and its dependence on the control $\log h_n$, according to the asymptotic assumption) into the control law (12) to obtain

$$\log h_{n+1} = \frac{1}{k}\,(\log \varepsilon - \log \hat{\varphi}_n). \tag{14}$$

This first-order difference equation for $\log h_n$ has the characteristic equation $q = 0$, i.e. the roots are at the origin. Known as *deadbeat control*, this is the result of choosing the integral gain $k_\mathrm{I} = 1/k$ in the controller.

From the point of view of control theory, however, the integral gain $k_\mathrm{I}$ is a free parameter; it is not determined by the asymptotic model of the process but is a design parameter used to achieve a good overall dynamic behaviour for process and controller *together*. Replacing the factor $1/k$ in (12) by an arbitrary gain $k_\mathrm{I}$ and inserting the asymptotic model for $\hat{r}_{n+1}$ we obtain the closed loop dynamics

$$\log h_{n+1} = (1 - kk_\mathrm{I}) \log h_n + k_\mathrm{I}(\log \varepsilon - \log \hat{\varphi}_n). \tag{15}$$

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 $kk_\mathrm{I}$ determines how quickly the control responds to variations in the *external disturbance* $\log \hat{\varphi}_n$, whose fluctuations are to be *rejected* or compensated by the controller. The system's *unit step response* is the behaviour resulting when the control error increases from 0 to 1. 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 smoother control[1], 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 $kk_\mathrm{I}$ is therefore a tradeoff between response time and sensitivity; for an I controller to work well one must in effect choose $kk_\mathrm{I} \in (0, 1)$.

The solution of the difference equation (15) is given by the convolution

$$\log h_n = (1 - kk_\mathrm{I})^n \log h_0 + k_\mathrm{I} \sum_{m=1}^{n} (1 - kk_\mathrm{I})^{n-m} (\log \varepsilon - \log \hat{\varphi}_{m-1}). \tag{16}$$

This mollifies the fluctuations in $\log \hat{\varphi}_n$ (recall that $\hat{\varphi}_n$ is composed of high order derivatives of $f$) and therefore yields smoother stepsize sequences when $kk_\mathrm{I} \in (0, 1)$ than those obatined with the deadbeat control (13).

The general I controller can be written as

$$h_{n+1} = \left( \frac{\varepsilon}{\hat{r}_{n+1}} \right)^{k_\mathrm{I}} h_n. \tag{17}$$

When we compare this to (10), it is of importance to note that the change of integral gain from $1/k$ to $k_\mathrm{I}$ 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*

[1] Note that one cannot choose $kk_\mathrm{I} = 0$ as (15) then yields a constant stepsize.

as was assumed for the elementary control algorithm (10). 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 proportional integral derivative (PID) control structures to increase robustness. In most cases a PI controller is satisfactory. Its idea is to construct the control $\log h_n$ as follows:

$$\log h_n = \log h_0 + k_{\mathrm{I}} \sum_{m=1}^{n} (\log \varepsilon - \log \hat{r}_m) + k_{\mathrm{P}} (\log \varepsilon - \log \hat{r}_n). \tag{18}$$

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

Forming a recursion from (18), we obtain

$$\log h_{n+1} = \log h_n + k_{\mathrm{I}} (\log \varepsilon - \log \hat{r}_{n+1}) + k_{\mathrm{P}} (\log \hat{r}_n - \log \hat{r}_{n+1}). \tag{19}$$

The *PI controller* can therefore be written

$$h_{n+1} = \left( \frac{\varepsilon}{\hat{r}_{n+1}} \right)^{k_{\mathrm{I}}} \left( \frac{\hat{r}_n}{\hat{r}_{n+1}} \right)^{k_{\mathrm{P}}} h_n, \tag{20}$$

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 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 (19), the closed loop dynamics is described by

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

with the characteristic equation

$$q^2 - (1 - kk_{\mathrm{I}} - kk_{\mathrm{P}}) q - kk_{\mathrm{P}} = 0. \tag{21}$$

The dynamics is therefore determined by two roots, the locations of which are functions of the two design parameters $kk_{\mathrm{I}}$ and $kk_{\mathrm{P}}$. One cannot, however, choose the values of $kk_{\mathrm{I}}$ and $kk_{\mathrm{P}}$ solely from this characteristic equation, as we have assumed that the asymptotic process model holds when we derived (21). It is necessary to study additional process models, e.g. the dynamics when numerical stability limits the stepsize, to find a good parametrization. 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 [19]. In many but certainly not all nonstiff computations these assumptions are

not unrealistic, explaining the success of the elementary controller (10). 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.

## 3. Nonstiff computation: PI control

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

(i) finding the dynamics of the process to be controlled;
(ii) choosing a control structure;
(iii) finding the closed loop dynamics of controller and process together in order to select appropriate control parameters.

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, we denote transformed stepsize, error and disturbance sequences without a subscript, see Figure 1.

The first task is to *identify the process models*, which map $\log h$ to $\log \hat{r}$. When the asymptotic assumption holds, we have

$$\log \hat{r} = G_{\mathrm{P}}(q) \log h + q^{-1} \log \hat{\varphi}, \tag{22}$$

where the process is just a constant gain, $G_{\mathrm{P}}(q) = G_{\mathrm{P}}^0(q) = kq^{-1}$. 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 $\hat{r}_{n+1} = \hat{\varphi}_n h_n^k$.

If numerical stability limits the stepsize, however, the process is no longer static but dynamic. A dynamic model was first obtained in [9, 10] for the linear test equation[2] $\dot{y} = \lambda y$. The explicit Runge–Kutta method then yields $y_{n+1} = P(z_n) y_n$, where $P$ is the stability polynomial of the method and $z_n = h_n \lambda$. For the error estimate one similarly has $\hat{l}_{n+1} = E(z_n) y_n$ where $E$ is the error estimator polynomial. We now assume that $z^* = h^* \lambda$ is such that $|P(z^*)| = 1$, i.e. $z^* \in \partial S$, the boundary of the stability region. We next write

$$\hat{l}_{n+1} = E(z_n) y_n = E(z_n) P(z_{n-1}) y_{n-1} = P(z_{n-1}) \frac{E(z_n)}{E(z_{n-1})} \hat{l}_n \tag{23}$$

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

$$\hat{l}_{n+1} \doteq P(z^*) \left( \frac{h_n}{h^*} \right)^{C_1} \left( \frac{h_{n-1}}{h^*} \right)^{C_2 - C_1} \hat{l}_n, \tag{24}$$

[2] This problem models 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 $\psi(t)$.

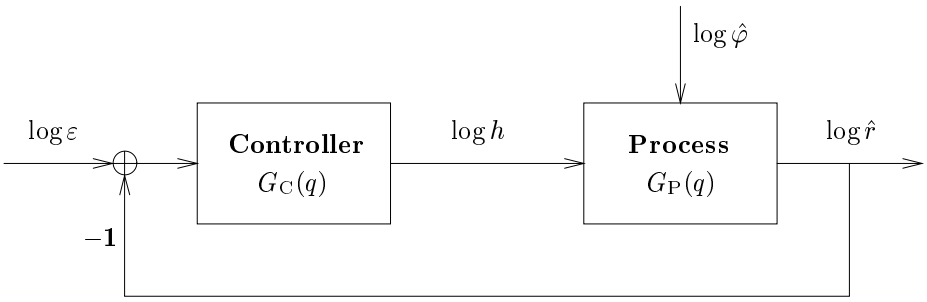


FIGURE 1. Adaptive stepsize selection viewed as a feedback control system. The process consists of the discretization method which takes a given stepsize $\log h$ as input and produces an error output which is estimated by $\log\hat{r}$. The method is applied to an ODE whose influence is represented as an external disturbance $\log\hat{\varphi}$ accounting for problem properties. The error $\log\hat{r}$ is fed back with reversed sign (the factor $-1$ on the feedback loop), then added to $\log\varepsilon$ to compare actual and desired error levels. Taking this difference 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 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 $\log\varepsilon$.

with

$$C_1(z^*) = \mathrm{Re}\left(z^*\,\frac{E'(z^*)}{E(z^*)}\right); \qquad C_2(z^*) = \mathrm{Re}\left(z^*\,\frac{P'(z^*)}{P(z^*)}\right). \tag{25}$$

These coefficients depend on the method parameters $(A, b, \hat{b})$ as such, but also vary along $\partial S$. We shall take a different approach from [11, 4] and normalize the coefficients by $k$, defining

$$\hat{c}_1(z^*) = C_1(z^*)/k; \qquad \hat{c}_2(z^*) = C_2(z^*)/k. \tag{26}$$

We now take the logarithm of $\hat{r}_n = |\hat{l}_n|$ (or in the EPUS case $|\hat{l}_n|/h_n$) and express (24) in terms of the forward shift $q$. Noting that $|P(z^*)| = 1$ we obtain

$$\log\hat{r} \doteq G_{\mathrm{P}}^{\partial S}(q)(\log h - \log h^*), \tag{27}$$

where the sought dynamic process model takes the form

$$G_{\mathrm{P}}^{\partial S}(q) = kq^{-1}\left(\hat{c}_1(z^*) - \frac{\delta_{pk}}{k} + \frac{\hat{c}_2(z^*)}{q-1}\right), \tag{28}$$

with $\delta_{pk} = 0$ for EPS and $\delta_{pk} = 1$ for EPUS, see [4, p. 538]. To verify this dynamic model, Gustafsson used *system identification*, [16]. 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 $\hat{c}_1(0) - \delta_{pk}/k = 1$ and $\hat{c}_2(0) = 0$, $G_{\mathrm{P}}^{\partial S}(q)$ reduces to $G_{\mathrm{P}}^{0}(q)$ 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 $\partial S$, cf. [4, p. 552].

The next task is to *design a controller*, which must be able to adequately manage both processes. The elementary controller (10) will not do, as it is often unable to yield stable dynamics for $G_{\mathrm{P}}^{\partial S}(q)$.

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

$$\log h = G_{\mathrm{C}}(q)(\log\varepsilon - \log\hat{r}). \tag{29}$$

Identifying (29) with (19) we readily find the expression for the *PI controller*,

$$G_{\mathrm{C}}^{\mathrm{PI}}(q) = k_{\mathrm{I}}\,\frac{q}{q-1} + k_{\mathrm{P}}. \tag{30}$$

As $q$ is the forward shift operator, $q - 1$ is a difference operator; $1/(q-1)$ is therefore the summation operator representing the controller's integral action. Moreover, the pure I controller is obtained as the special case $k_{\mathrm{P}} = 0$.

The *closed loop dynamics* is obtained by combining process and controller and eliminating $\log h$. Thus, in the asymptotic regime, we insert (29) into (22) to obtain

$$\log\hat{r} = G_{\varepsilon}^{0}(q)\log\varepsilon + G_{\hat{\varphi}}^{0}(q)\log\hat{\varphi}, \tag{31}$$

where

$$G_{\varepsilon}^{0}(q) = \frac{G_{\mathrm{C}}(q)\,G_{\mathrm{P}}^{0}(q)}{1 + G_{\mathrm{C}}(q)\,G_{\mathrm{P}}^{0}(q)}\,; \qquad G_{\hat{\varphi}}^{0}(q) = \frac{q^{-1}}{1 + G_{\mathrm{C}}(q)\,G_{\mathrm{P}}^{0}(q)}\,. \tag{32}$$

To consider PI control we insert (30) and the process model $G_{\mathrm{P}}^{0}(q) = kq^{-1}$ into (32). The transfer function from external disturbance $\log\hat{\varphi}$ to error estimate $\log\hat{r}$ can then be written explicitly as

$$G_{\hat{\varphi}}^{0}(q) = \frac{q-1}{q^2 - (1 - kk_{\mathrm{I}} - kk_{\mathrm{P}})q - kk_{\mathrm{P}}}\,. \tag{33}$$

The closed loop dynamics is governed by the poles of the transfer functions (32). Thus, by looking at the denominator of (33) we recognize the same characteristic equation (21) as before. Moreover, the numerator of $G_{\hat{\varphi}}^{0}(q)$ contains the difference operator $q - 1$. Appearing as consequence of the controller's integral action, this operator will remove any constant disturbance $\log\hat{\varphi}$ in (31) at a rate determined by the location of the poles. Last, we find that $G_{\varepsilon}^{0}(1) = 1$,

implying that $\log \hat{r}$ will, in a stable system, eventually approach the setpoint $\log \varepsilon$; this criterion is akin to the notion of consistency in numerical analysis.

Before selecting controller parameters, we also need to consider the closed loop dynamics on $\partial S$. Combining (27) and (29) we obtain

$$\log \hat{r} = G_{\varepsilon}^{\partial S}(q) \log \varepsilon + G_{h^*}^{\partial S}(q) \log h^*, \tag{34}$$

where we need to find the poles of the transfer functions

$$G_{\varepsilon}^{\partial S}(q) = \frac{G_{\mathrm{C}}^{\mathrm{PI}}(q)\, G_{\mathrm{P}}^{\partial S}(q)}{1 + G_{\mathrm{C}}^{\mathrm{PI}}(q)\, G_{\mathrm{P}}^{\partial S}(q)}; \qquad G_{h^*}^{\partial S}(q) = -\frac{G_{\mathrm{P}}^{\partial S}(q)}{1 + G_{\mathrm{C}}^{\mathrm{PI}}(q)\, G_{\mathrm{P}}^{\partial S}(q)}. \tag{35}$$

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

$$\begin{aligned} a_2 &= -2 - (kk_{\mathrm{I}} + kk_{\mathrm{P}})\hat{c}_1 \\ a_1 &= 1 - (kk_{\mathrm{I}} + 2kk_{\mathrm{P}})\hat{c}_1 + (kk_{\mathrm{I}} + kk_{\mathrm{P}})\hat{c}_2 \\ a_0 &= -kk_{\mathrm{P}}(\hat{c}_2 - \hat{c}_1). \end{aligned}$$

Therefore the poles are determined by the parameters $kk_{\mathrm{I}}$ and $kk_{\mathrm{P}}$ both in the asymptotic case and on $\partial S$. It is nevertheless a considerable challenge to choose the PI control parameters. There is a gradual change in process models from $G_{\mathrm{P}}^{0}(q)$ to $G_{\mathrm{P}}^{\partial S}(q)$ as $z$ leaves the asymptotic domain and approaches $\partial S$ [4, p. 541]. In addition $\hat{c}_1$ and $\hat{c}_2$ are method dependent and vary on $\partial S$.

The choice of $kk_{\mathrm{I}}$ and $kk_{\mathrm{P}}$ 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 testings led GUSTAFSSON [4] to suggest $(kk_{\mathrm{I}}, kk_{\mathrm{P}}) = (0.3, 0.4)$ as a good choice, to be considered as a starting point for fine-tuning the controller for an individual method. This results in poles located at $q = 0.8$ and $q = -0.5$ for (33), 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[3]. 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 somewhat faster, better damped and smoother response near $z = 0$ is achieved by taking $(kk_{\mathrm{I}}, kk_{\mathrm{P}}) = (0.4, 0.2)$. The poles at $z = 0$ have then moved to $q \approx 0.69$ and $q \approx -0.29$. In most cases the parameter set of interest is $\{(kk_{\mathrm{I}}, kk_{\mathrm{P}}) \;:\; kk_{\mathrm{I}} + kk_{\mathrm{P}} \leq 0.8;\; kk_{\mathrm{I}} \geq 0.3;\; kk_{\mathrm{P}} \geq 0.1\}$, noting that individual methods have somewhat different characteristics and that the parameter choice is a tradeoff between different objectives.

Extensive additional tests with the safety factor $\theta$ showed that one can run as close as 90% of `tol` before stepsize rejections become frequent. For

[3] We note that [8, p. 30] maintain $kk_{\mathrm{I}} + kk_{\mathrm{P}} = 1$; finding $(kk_{\mathrm{I}}, kk_{\mathrm{P}}) = (0.36, 0.64)$ unsatisfactory for an 8th order method, they change to $(0.68, 0.32)$. Closed loop dynamics is less convincing in both cases, however. Governed by two roots $\pm\sqrt{kk_{\mathrm{P}}}$ of (21) near $z = 0$, it is quite oscillatory and a reduced integral gain might be preferred. Further we note that the control analysis in [1] only treats a static process model and therefore prematurely concludes that an I controller is satisfactory.

an increased margin of robustness at a negligible cost, a value of $\theta = 0.8$ is recommended, implying a setpoint of $\varepsilon = 0.8 \cdot \texttt{tol}$. Thus, a good overall performance can be expected from the *PI.3.4 controller*

$$h_{n+1} = \left(\frac{0.8 \cdot \texttt{tol}}{\hat{r}_{n+1}}\right)^{0.3/k} \left(\frac{\hat{r}_n}{\hat{r}_{n+1}}\right)^{0.4/k} h_n, \tag{36}$$

with (10)—the *PI1.0*—as a safety net in case of successive rejected steps, [4, p. 546]. In many codes it is common to use additional logic, e.g. a *deadzone* inhibiting stepsize increases unless they exceed say 20%. But such a deadzone is detrimental to controller performance as it is equivalent to disengaging the controller from the process, waiting for large control errors to build up. 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 *PI.4.2* lies in its capacity to exert a delicate stabilizing control on the process as is clearly demonstrated by the smoother stepsize sequences. We therefore advocate a continual control action.

## 4. Stiff computation: Predictive control

In stiff computations the PI control turns out to be less useful and shows few clear advantages over (10), depending on the fact that the process has other characteristics. The two main problems are with the process assumptions: *(i)* stepsizes "far oustide" 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 the PI control is effective as a countermeasure for fluctuations in $\log \hat{\varphi}_n$. In the stiff case, however, a better rejection of external disturbances can be achieved if the variation of $\log \hat{\varphi}_n$ is modelled and predicted. This is done by constructing an *observer* [19] for $\log \hat{\varphi}_n$. Using the observer estimate to select the stepsize, Gustafsson [5, 8, p. 124] arrived at the *predictive control*

$$\frac{h_{n+1}}{h_n} = \left(\frac{\varepsilon}{\hat{r}_{n+1}}\right)^{k_e/k} \left(\frac{\hat{r}_n}{\hat{r}_{n+1}}\right)^{k_r/k} \frac{h_n}{h_{n-1}}, \tag{37}$$

with a recommended observer gain of $(k_e, k_r)^T = (1, 1)^T$. We recognize this control structure as a PI controller for the *stepsize change* $\log h_{n+1} - \log h_n$. This may appear to be a minor modification but the dynamics is different, requiring an unbounded stability region $S$. It is therefore not suitable for nonstiff methods. Interestingly, similar controllers based on extrapolating error trends and stepsizes were first suggested and used in [17, 18], although without an analysis of controller dynamics and parametrization.

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 notions of importance are *observability* and *controllability* [19]. Loosely speaking, in our context observability requires that we have an

error estimate which adequately reflects the true error. Controllability requires that the stepsize be an effective means to exert control; by adjusting $h_n$ we must be able to put the error at a prescribed level of $\varepsilon$. 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 $\mathrm{e}^{z_n}$. Today it is 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 $\hat{l}_n = (R(z_n) - \hat{R}(z_n))y_n$. 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[4]. 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.* In order to avoid this breakdown one must seek method/error estimator combinations which are controllable. This is still an area of active research, however, as it must address method properties, error estimator construction, cf. [15], termination criteria for Newton iterations and problem properties in DAEs.

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

$$\log \hat{r} = kq^{-1} \log h + q^{-1} \log \hat{\varphi}. \tag{38}$$

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*

$$\log \tilde{\varphi}_n = \log \hat{\varphi}_{n-1} + \nabla \log \hat{\varphi}_{n-1}, \tag{39}$$

where $\log \tilde{\varphi}_n$ denotes the predicted value of $\log \hat{\varphi}_n$ and $\nabla$ is the backward difference operator. Note that the model is not compensated by using divided differences. Such models have been tried but show no significant advantages over (39). Using the forward shift we rewrite (39) in the form

$$\log \tilde{\varphi} = (2q - 1)q^{-2} \log \hat{\varphi}. \tag{40}$$

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

We shall choose $h_n$ such that $\tilde{\varphi}_n h_n^k = \varepsilon$. The stepsize is therefore given by

$$\log h = k^{-1}(\log \varepsilon - \log \tilde{\varphi}). \tag{41}$$

Inserting the estimate (40) into (41), noting that $\log \varepsilon$ is constant, we obtain the *closed loop dynamics*

$$\log h = \frac{2q-1}{kq^2}(\log \varepsilon - \log \hat{\varphi}). \tag{42}$$

The double pole at $q = 0$ shows that we have obtained a *deadbeat control* analogous to (14). To find the controller, we use the asymptotic process model (38) to eliminate $\log \hat{\varphi}$ from (42) and obtain

$$\log h = \frac{2q-1}{kq}(\log \varepsilon - \log \hat{r}) + \frac{2q-1}{q^2} \log h. \tag{43}$$

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

$$G_{\mathrm{C}}^{\mathrm{PC}}(q) = \frac{1}{k} \frac{q\,(2q-1)}{(q-1)^2} = \frac{1}{k} \frac{q}{q-1} \left( \frac{q}{q-1} + 1 \right). \tag{44}$$

We note that the transfer function contains a *double integral action*; this is known to be necessary to follow a linear trend without control error.

In order to find the recursion formula (37) we rearrange (43) in the form

$$\frac{q-1}{q} \log h = \frac{1}{k} \left( \frac{q}{q-1} + 1 \right) (\log \varepsilon - \log \hat{r}), \tag{45}$$

where the usual PI control structure (30) is recognized in the right-hand side, with integral and proportional gains of $1/k$. Since the quantity on the left-hand side corresponds to $\log(h_n/h_{n-1})$, the formula (37) follows, and with $\theta = 0.8$ we have obtained the *PC11 controller*

$$h_{n+1} = \left( \frac{0.8 \cdot \mathtt{tol}}{\hat{r}_{n+1}} \right)^{1/k} \left( \frac{\hat{r}_n}{\hat{r}_{n+1}} \right)^{1/k} \frac{h_n^2}{h_{n-1}}. \tag{46}$$

For a detailed discussion of a general observer for $\log \hat{\varphi}$ with gain parameters $(k_e, k_r)$ we refer to [5]. Here it is sufficient to remark that the purpose of such observers 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 $\nabla \log \hat{\varphi}_n$. This control will be slower than the PC11 deadbeat design but is less sensitive to fluctuations in $\log \hat{\varphi}$; the convolution operator mapping $\log \hat{\varphi}$ to $\log h$ acts as a mollifier and smoother stepsize sequences are obtained. By contrast, (42) shows that in the PC11, $\log h_{n+1}$ is directly proportional to $\log \hat{\varphi}_n$ and its difference $\nabla \log \hat{\varphi}_n$. For stiff computations irregularities in $\log \hat{\varphi}_n$ are quite common as the error estimate is also influenced by an irregular error contribution from truncated Newton iterations. Therefore it may be worthwhile to try a different

parametrization of the PC controller. Because the dynamics of the PC closed loop is different from that obtained with the usual PI controller, the parameters discussed in the previous section are not relevant in this context. The choice of $(k_e, k_r)$ is discussed in [5, p. 512] where closed loop pole positioning as well as practical tests indicate the possibility of using slightly smaller values of the parameters, e.g. the *PC.6.9* controller with $(k_e, k_r) = (0.6, 0.9)$. This will provide somewhat smoother stepsize sequences without significant loss of efficiency and robustness.

The PC controllers also need safety nets in case of rejected steps. A complete pseudocode of the PC11 controller, including exception handling and estimation of $k$ in case of order reduction is provided in [5, p. 511].

## 5. 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 form

$$y_n = \gamma h_n f(y_n) + \psi, \tag{47}$$

where $\gamma$ is a method-dependent constant and $\psi$ 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 has long been argued that minor stepsize variations might force matrix refactorizations and must be avoided. As a remedy the elementary controller is often used in combination with a deadzone prohibiting small stepsize increases, aiming to keep $h_n$ piecewise constant. Stepsize decreases, on the other hand, are readily accepted even when small, and may also be controlled by a different strategy—as an example Radau5 uses the elementary control (10) with a 20% deadzone for stepsize increases, and the predictive control (46) for decreases [8, p. 124]. This *gain scheduling* strategy makes an unsymmetric, discontinuous and nonlinear control, and yet it does not save factorizations during long sequences of shrinking steps which do occur in practice [6, p. 35, Fig. 8]. 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 $\varepsilon$. 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 (47) we refer to [6]. The convergence rate $\alpha$ 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 this puts an upper limit on the stepsize beyond which further stepsize increases are counterproductive; convergence slows to the point where total work per unit time of integration starts *growing* [6, p. 27]. In theory the convergence rate should never exceed $1/e \approx 0.37$, but in practice one should limit $h_n$ so that $\alpha \leq 0.2$. 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 (3), 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 most 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,

$$e_{j+1} = (I - \gamma h_m J_m)^{-1} \gamma \, \delta(hJ) \, e_j. \tag{48}$$

We shall find an estimate of $\alpha$ 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 (48) as

$$e_{j+1} = (I - \gamma h_m J_m)^{-1} \gamma h_m J_m (h_m J_m)^{-1} \delta(hJ) e_j. \tag{49}$$

By taking norms, we obtain $\|e_{j+1}\| \leq \nu \cdot \|(h_m J_m)^{-1} \delta(hJ)\| \, \|e_j\|$, where it can be shown for inner product norms that $\nu = \|(I - \gamma h_m J_m)^{-1} \gamma h_m J_m\| \to 1$ in the presence of stiffness, i.e. as $\|h_m J_m\|$ grows large [6, p. 29]. We can therefore estimate the convergence rate by

$$\alpha \lesssim \|(h_m J_m)^{-1} \delta(hJ)\|. \tag{50}$$

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 (50) yields the bound

$$\alpha \lesssim \| \frac{\delta h}{h_m} I + J_m^{-1} \delta J \| \leq | \frac{\delta h}{h_m} | + \|J_m^{-1} \delta J\|. \tag{51}$$

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 convergence rate grows and exceeds the acceptable level of $\alpha = 0.2$, then (51) demonstrates that this can only be due to a relative change in the Jacobian, calling for a reevaluation of $J$. Note that there is no need to compute $\|J_m^{-1}\delta J\|$. It is sufficient to monitor the convergence rate $\alpha$ 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 countermeasures. 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 [14, p. 160]. The purpose of these strategies is to establish a full coordination of stepsize control and iterative method. This can be achieved for various choices of the upper limit on acceptable convergence rate.

In [13] an attempt is made to develop a global matrix strategy. This requires a significant amount of problem information, however. Transformed into a local test, this strategy keeps the same Jacobian as long as $\alpha$ is small enough to satisfy

$$\frac{\log \alpha}{\sqrt{l+1}} \leq -\sqrt{\frac{\log(\|e_0\|/\varepsilon)}{\tau_f}}\,, \tag{52}$$

where $e_0$ is the initial iteration error, $l$ is the number of steps since the iteration matrix was formed, and $\tau_f$ is the time for computing and factorizing the Jacobian, relative to the time for a full iteration, including function evaluation. This strategy is therefore adaptive with respect to computational progress, problem properties and size, but it is more difficult to use in practice, not least because it typically puts a quite sharp upper bound on $\alpha$ which tends to interfere with the stespize controller. For certain problems in high precision computations, this adaptive strategy may reduce CPU times by 20%.

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 (4). Moreover, the terminal iteration errors must not make more than an insignificant contribution to the local error

(7) *and* its estimate (6). The iteration error is less regular than the truncation error of the dicsretization, and may—if too large—cause an irregular behaviour in $\log\hat{\varphi}$, i.e. the controller will be fed "noisy data." The first order predictor (39) is then prone to be erratic and it is no longer possible to achieve better control with the PC11. 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.6.9 or to resort to the PI1.0, which is based on the assumption of slow variation, equivalent to a predictor of order zero for $\log\hat{\varphi}$. In the coordination of controller and iteration strategy, these considerations need to be accounted for to obtain a coherent performance.

## 6. Conclusions

The algorithmic content of ODE/DAE software 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. The objective of this review has been to introduce the numerical analyst interested in ODEs to some basic notions and techniques from control theory which have proved efficient in the construction of adaptive ODE/DAE software. This methodology is not unfamiliar to numerical analysis; digital control rests on the classical theories of linear difference equations, difference operators and stability. Bearing this in mind, the numerical analyst has a wide range of powerful concepts at his/her disposal for the automatic control of numerical integration. The present paper should serve as a starting point for accessing the state of the art in this field as of 1998 through the referenced literature.

The control theoretic approach views the adaptive numerical integration of ODEs as a process which maps a stepsize $\log h$ to a corresponding error $\log\hat{r}$. Section 3 showed that the process is an affine map, i.e. $\log\hat{r} = G_{\mathrm{P}}(q)\log h + \log\hat{\varphi}$. Moreover, in the asymptotic regime it is static—$G_{\mathrm{P}}(q)$ is just a constant. The process is therefore relatively simple to control. 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. But even if the process has a generic structure for all discretizations, it is a challenging task to find suitable controller parameters. The recommended controllers, PI.3.4 and PI.4.2, are fully specified in the form of pseudocode in [4].

For implicit methods we saw in Section 4 that a more advanced technique based on predicting the evolution of the principal error function could be used. 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 controller is found in [5]. Finally, in Section 5 we saw how a continual control action could be coordinated with matrix strategies for the Newton iterations; algorithm specifications are found in [14, p. 160], which improves the original in [6], and in [13].

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 the new algorithms are more complex and mainly directed towards qualitative improvements, they usually decrease total work as well.

Thus we have brought together a number of algorithms, chiefly based on modern control theory, for making numerical ODE/DAE integration methods adaptive. These control algorithms are supported by a well established mathematical methodology instead of heuristics, 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. A less systematic approach, such as trying to 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. Other performance aspects, such as the quality of the numerical solution in terms of better smoothness achieved by smoother stepsize sequences (cf. [4, p. 534, Fig. 1]) or a more stable and regular tolerance proportionality, are often overlooked but do belong in a serious evaluation of software performance and quality.

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.

ACKNOWLEDGEMENTS. The author would like to thank a large number of colleagues and collaborators in numerical analysis and automatic control who 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 222/91-405.

## REFERENCES

1. P. DEUFLHARD, F. BORNEMANN (1994). *Numerische Mathematik II: Integration gewöhnlicher Differentialgleichungen*. Walter de Gruyter, Berlin.
2. C.W. GEAR (19971). *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice–Hall, Englewood Cliffs.
3. K. GUSTAFSSON, M. LUNDH, G. SÖDERLIND (1988). A PI stepsize control for the numerical solution of ordinary differential equations. *BIT* **28**, 270–287.

4. K. Gustafsson (1991). Control theoretic techniques for stepsize selection in explicit Runge–Kutta methods. *ACM TOMS* **17**, 533–554.
5. K. Gustafsson (1994). Control theoretic techniqes for stepsize selection in implicit Runge–Kutta methods. *ACM TOMS* **20**, 496–517.
6. K. Gustafsson, G. Söderlind (1997). Control strategies for the iterative solution of nonlinear equations in ODE solvers. *SIAM J. Sci. Comp.* **18**, 23–40.
7. E. Hairer, S.P. Nørsett, G. Wanner (1993). *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer-Verlag, 2nd revised edition, Berlin.
8. E. Hairer, G. Wanner (1996). *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 2nd revised edition, Berlin.
9. G. Hall (1985). Equilibrium states of Runge–Kutta schemes. *ACM TOMS* **11**, 289–301.
10. G. Hall (1986). Equilibrium states of Runge–Kutta schemes, part II. *ACM TOMS* **12**, 183–192.
11. G. Hall, D. Higham (1988). Analysis of stepsize selection schemes for Runge–Kutta codes. *IMA J. Num. Anal.* **8**, 305–310.
12. D. Higham, G. Hall (1990). Embedded Runge–Kutta formulae with stable equilibrium states. *J. Comp. and Appl. Math.* **29**, 25–33.
13. H. Olsson, G. Söderlind (1998). Stage value predictors and efficient Newton iterations in implicit Runge–Kutta methods. To appear in *SIAM J. Sci. Comp.* **19**.
14. J.J.B. de Swart (1997). *Parallel software for implicit differential equations.* Ph.D. thesis, CWI, Amsterdam.
15. J.J.B. de Swart, G. Söderlind (1997). On the construction of error estimators for implicit Runge–Kutta methods. *J. Comp. and Appl. Math.* **86**, 347–358.
16. T. Söderström, P. Stoica (1989). System Identification. Prentice–Hall, Englewood Cliffs.
17. H.A. Watts (1984). Step size control in ordinary differential equation solvers. *Trans. Soc. Comput. Sim.* **1**, 15–25.
18. J.A. Zonneveld (1964). *Automatic numerical integration.* Ph.D. thesis, Math. Centre Tracts 8, CWI, Amsterdam.
19. K.J. Åström, B. Wittenmark (1990). *Computer Controlled Systems – Theory and Design.* 2nd ed., Prentice–Hall, Englewood Cliffs.