Stepsize Control in ODE-Solvers

Analysis and Synthesis

Gustafsson, Kjell

1988

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Gustafsson, K. (1988). *Stepsize Control in ODE-Solvers: Analysis and Synthesis*. [Licentiate Thesis, Department of Automatic Control]. Department of Automatic Control, Lund Institute of Technology (LTH).

Total number of authors:
1

Stepsize Control in ODE-Solvers – Analysis and Synthesis

Kjell Gustafsson

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Licentiate Thesis

Date of issue
June 1988

Document Number
CODEN: LUTFD2/(TFRT-3199)/1-98/(1988)

Author(s)
Kjell Gustafsson

Supervisor
Per Hagander, Gustaf Söderlind, Karl Johan Åström

Sponsoring organisation
The Swedish National Board for Technical Development (STU)

Title and subtitle
Stepsize Control in ODE-Solvers – Analysis and Synthesis

Abstract

The problem of automatic stepsize control in numerical solution of differential equations is approached using control theory. Although being an automatic control problem this point of view is seldom pursued.

Standard stepsize control is derived assuming a static relation between the stepsize and local truncation error in the integration method. In many cases such a model is not sufficient, and instead a dynamic model has to be used. Here a dynamic model for explicit Runge-Kutta methods is derived. The model is verified using numerical tests and system identification.

The derived model is used to analyze standard stepsize control. The analysis gives insight and leads to a good understanding of the properties of the control system. In particular, with the model, it is straightforward to explain the problems of oscillating stepsize sequences often encountered when using explicit Runge-Kutta methods to solve stiff differential equations.

Moreover, the acquired understanding is used to construct a new stepsize control algorithm which gives superior performance at little extra expense. The new controller also overcomes the stepsize oscillation problems.

Key words
stepsize control, numerical integration, ordinary differential equation

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title	*ISBN*

Language English	*Number of pages* 98	*Recipient's notes*
Security classification		

Contents

Preface

"It's one small step for man, one giant leap for mankind."

Neil Armstrong*

As according to Neil Armstrong, the size of a step is related to its environment. This is particularly evident in numerical integration of ordinary differential equations, where the stepsize is used to control the quality of the produced solution. A large stepsize results in a large error, while a small stepsize leads to inefficiency due to the many steps needed to produce the solution. What is regarded as small and large depends on the accuracy requirements and the differential equation being integrated. The art is to at all times choose exactly the right stepsize; sufficiently small for accuracy, but not too small for efficiency.

In this thesis the problem of choosing stepsize is approached as an automatic control problem. Normally, an estimate of the error in the produced solution is fed back to influence the choice of stepsize. Control theory provides tools to analyze this feedback loop, and also suggests improvements of the standard control algorithms used today.

This thesis consists of two parts. The first part is an article with the title *"A PI Stepsize Control for the Numerical Solution of Ordinary Differential Equations,"* coauthored with Michael Lundh and Gustaf Söderlind. The article, which will appear in BIT during 1988, formulates the choice of stepsize as a control problem. The standard stepsize used today is recognized as a pure integrating controller, and from general considerations a new controller is proposed. The new controller is tested for a number of different differential equations, and it is in most cases found to be superior to the standard controller.

In the second part of the thesis, entitled *"Analysis and Synthesis of Stepsize Control in Numerical Solution of Ordinary Differential Equations,"* a dynamic model for the relation between stepsize and error is derived for explicit Runge-Kutta methods. The model is verified using

* Journey to the Moon, Pickwick International Inc. Ltd, London, 1969

system identification techniques and numerical tests. Using the model facilitates an analysis of the standard stepsize controller as well as the controller proposed in the first part of the thesis. The analysis verifies the properties observed in the first part, and it also suggests further improvements for the new controller.

Acknowledgements

I would like to express my gratitude to Per Hagander and Gustaf Söderlind. Gustaf Söderlind suggested the problem of stepsize control and has been a tremendous source of information about numerical integration in general. Throughout this work Per Hagander has supplied many useful ideas, and I appreciate him always taking time for our discussions.

Michael Lundh coauthored the BIT-article. Many of the original ideas were born during discussions with him, and I wish to express my gratitude for his contributions.

Furthermore, I am grateful to Karl Johan Åström, who has been a consistent source of enthusiasm.

Per Hagander, Sven Erik Mattsson and Gustaf Söderlind have read several versions of the manuscript. Their suggestions and comments have greatly improved the quality of the thesis.

My colleagues at the Department of Automatic Control form a friendly and inspiring community. I am grateful for their encouragement and help.

Finally, I thank Eva, my beloved wife, for her patience and encouragement.

The second part of the thesis has been supported by the Swedish National Board for Technical Development (STU) under contract 88-01611P.

K.G.

A PI Stepsize Control for the Numerical Solution of Ordinary Differential Equations

Kjell Gustafsson, Michael Lundh
Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund, Sweden

Gustaf Söderlind
Department of Computer Sciences
Lund University
P.O. Box 118
S-221 00 Lund, Sweden

Abstract

A control-theoretic approach is used to design a new automatic stepsize control algorithm for the numerical integration of ODE's. The new control algorithm is more robust at little extra expense. Its improved performance is particularly evident when the stepsize is limited by numerical stability. Comparative numerical tests are presented.

1. Introduction

In the numerical integration of ordinary differential equations, automatic stepsize control is probably the most important means to make an integration method efficient. The objective of stepsize control is the following optimization problem: Given a method and an initial value problem, "minimize" the computational effort to construct an approximate solution in accordance with a user-specified "accuracy" requirement.

As for the accuracy, a standard approach is to adjust the stepsize to keep an estimate of the local truncation error per unit step bounded. This strategy is motivated by the fact that the global error can be bounded in terms of the local truncation error per unit step.

In order to minimize the work, one usually maximizes each individual step, without regard to global strategies (an interesting exception is the work in [Lindberg 1977]). More precisely, one tries to minimize the work

$$W = \alpha N + \beta M, \tag{1}$$

where N is the total number of steps and M is the number of stepsize changes. The parameters α and β represent the costs of taking one step and changing the stepsize, respectively. For some methods (e.g. explicit Runge-Kutta methods) changing the stepsize does not invoke extra computations, i.e. $\beta = 0$. In connection with implicit methods intended for stiff problems, on the other hand, a stepsize change may require additional matrix factorizations, thus causing the second term in (1) to be significant.

Despite the importance of stepsize control, it seems that no effort has been devoted to using control theory for the design of such algorithms. In this paper, we analyze a typical stepsize control algorithm from a control theory point of view. We propose a new control algorithm, based on a discrete PI (proportional integral) controller. The choice is motivated by the performance requirements: the controller must work properly for problems with a great diversity in dynamical behavior.

It is evident that such requirements are hard to meet, since the controller parameters must be tuned for a variety of test problems. It is of particular importance that the stepsize sequences are smooth. Our comparative tests indicate that the new controller generally produces stepsize sequences with a better regularity than the standard controller. As a result, the error estimates show a smoother behavior. The latter property might be of particular interest in connection with multistep methods, since their numerical performance may depend in a crucial way on the stepsize sequence. Indeed, if the stepsize sequence is smooth enough, there is no need to try to minimize the number of stepsize changes in (1). Ideally, one would like the stepsize sequence smoothness to resemble the smoothness of the solution itself.

In this investigation we have limited ourselves to tuning the controller parameters for an explicit Runge-Kutta method, although the algorithm can be used with any type of integration method (possibly after a change of parameters). For explicit methods, the typical controller sometimes oscillates violently if there is a conflict between accuracy and numerical stability. Since this will happen in any stiff problem, many of our test problems are stiff. The new controller overcomes the oscillatory behavior and thus has much improved stability characteristics. For nonstiff problems, its performance is similar to that of the standard controller. It is likely that our algorithm will be advantageous also in stiff integration methods. This will, however, require a separate analysis which has not been pursued in this paper.

2. Standard Stepsize Control Algorithms

We start by describing a typical stepsize control algorithm. Most integration methods today use an algorithm of this kind [Hairer et al. 1987], [Gear 1971].

A Typical Stepsize Control Algorithm

The user specifies the desired accuracy of the solution by giving an upper bound *tol* for the local error per unit step. For a method of order p the local error r depends on the stepsize h asymptotically as $r \sim h^{p+1}$. If we represent the error by

$$r = \phi h^{p+1}, \tag{2}$$

the coefficient vector ϕ is $O(1)$ as $h \to 0$. In addition, ϕ depends on the solution of the differential equation; in this respect it may be regarded as a function of time.

The error is often measured with the norm

$$\|r\| = \max_i \left| \frac{r_i}{|y_i| + \eta_i} \right|, \tag{3}$$

where η_i is a scaling factor for the i:th component of y, resulting in a mixed absolute-relative error measure.

To take as long steps as possible without violating the prescribed *tol*, the stepsize should be chosen to fulfill

$$\|r\|/h = tol. \tag{4}$$

Motivated by these relations the stepsize for the next step (h_{n+1}) is chosen as

$$h_{n+1} = \theta h_n$$
$$\theta = \gamma \left(\frac{tol}{\|r_n\|/h_n} \right)^{1/p}, \tag{5}$$

where γ is a "safety factor" chosen ≤ 1. A typical choice is $\gamma = 0.9$. The purpose of the safety factor is to reduce the risk of rejecting the next step. If the error per unit step is too big in one step ($\|r\|/h > \rho \cdot tol$), then the step is rejected and recalculated with a new stepsize. A typical value of ρ is 1.2.

To prevent many small stepsize changes a dead-zone is often used. If θ is close to 1 no stepsize change is made. Here we introduce the dead-zone mainly for the sake of studying different control strategies. Although a dead-zone is not commonly used for Runge-Kutta methods (but rather for multistep methods), it may occasionally prevent stepsize oscillations. There is also a limit on how much the stepsize may increase in one step. Hence

$$\theta \leftarrow \begin{cases} 1, & \text{if } \theta_{lo} \leq \theta \leq \theta_{hi} \\ \theta_{max}, & \text{if } \theta > \theta_{max} \\ \theta, & \text{otherwise} \end{cases} \tag{6}$$

where '$\leftarrow$' means assignment. Typical values of the parameters are: $\theta_{lo} = 1.0$, $\theta_{hi} = 1.2$, and $\theta_{max} = 2.0$.

This standard stepsize control algorithm normally performs quite well. However, there are differential equations and integration methods for which its performance is unacceptable. The stepsize oscillates violently (cf. Section 4) and much computation time is spent recalculating rejected steps and changing the stepsize. This is especially true for non-stiff integration methods applied to stiff differential equations.

Analysis from a Control Theory Point of View

Considering the choice of stepsize as a standard automatic control problem, the problem may be viewed as in Figure 1. The plant G_p consists of the integration routine and the differential equation. It takes a stepsize h

as input and produces an error estimate r as output. Naturally, the numerical solution of the differential equation is also produced by the plant, but it is not used for stepsize control. The controller G_c is the stepsize control algorithm. It tries to select the stepsize such that the estimated local error per unit step comes as close as possible to the prescribed tolerance.

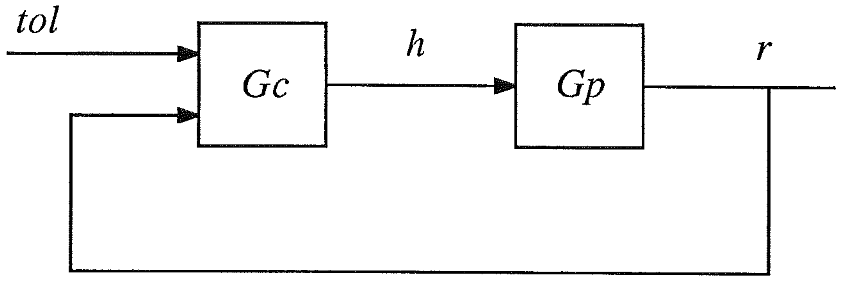

Figure 1. Stepsize control viewed as an automatic control problem.

The plant is nonlinear and time-varying. Its properties depend on the changing behavior of the solution of the ODE. One part of the nonlinearity is approximately known, and can be taken care of. From (2) we know that $\|r\|$ is asymptotically proportional to h^{p+1}. If the logarithm of h is regarded as plant input and the logarithm of $\|r\|$ as the output, this part of the nonlinearity will turn into an affine relation, i.e. $\log\|r\| = (p+1)\log h + \log\|\phi\|$.

The standard control strategy described above can be viewed as an *integrating controller* with the logarithm of h as the control variable. To see that, we start by expressing $\log(h_{n+1})$ as a function of $\log(h_n)$ using formula (5). Some manipulations give

$$\log h_{n+1} = \log h_n + \frac{1}{p}\left(\log(\gamma^p \cdot tol) - \log\left(\frac{\|r_n\|}{h_n}\right)\right). \tag{7}$$

Thus h_n will change until the deviation $\log(\gamma^p \cdot tol) - \log(\|r_n\|/h_n)$, known as the *control error*, is zero. Note that the use of a safety factor γ is equivalent to using a smaller tolerance $\gamma^p \cdot tol$. We recognize $\log(\gamma^p \cdot tol)$ as the *set point*, i.e. the controller aims at a local truncation error per unit step as close as possible to $\log(\gamma^p \cdot tol)$. The control h is thus obtained by "integrating" the control error signal.

When the dead-zone or the limitation is active it means invoking a different control signal than the calculated control. The states in the controller are updated to reflect this difference to prevent the controller from behaving improperly. In control engineering this special update, when the control signal is limited, is referred to as *anti-windup*, see e.g. [Franklin et al. 1986, Åström & Wittenmark 1984]. Thus the controller can be expressed with the following equations

$$\begin{aligned}
e_n &= \log(\gamma^p \cdot tol) - \log\left(\|r_n\|/h_n\right) && \text{(control error)} \\
I_{temp} &= I_{n-1} + p^{-1} e_n && \text{(integration)} \\
h_{temp} &= \exp(I_{temp}) && \\
h_{n+1} &= \begin{cases} h_n, & \text{if } \theta_{lo} h_n \le h_{temp} \le \theta_{hi} h_n \\ \theta_{max} h_n, & \text{if } h_{temp} > \theta_{max} h_n \\ h_{temp}, & \text{otherwise} \end{cases} && \text{(limitation)} \\
I_n &= I_{temp} + (\log h_{n+1} - \log h_{temp}) && \text{(anti-windup)}
\end{aligned} \tag{8}$$

The control error is multiplied by the factor $1/p$ before being integrated. This factor is referred to as the *integration gain.* The integration gain will determine how fast the controller responds to a non-zero control error. The performance of the closed loop system will also depend on the properties of the integration routine and the ODE. These properties will differ from problem to problem, thus making the behavior of the closed loop system vary considerably. It should be noted that such variations, which are represented by the term $\log \|\phi\|$ above, are not explicitly accounted for by this controller or by the new controller discussed in Section 3.

A good controller must work well for a large class of problems. However, the standard controller does not have an entirely satisfactory performance. Oscillations can clearly be seen when applying it to certain problems (cf. Section 4). One origin of the oscillations is the poor stabilizing capability of a pure integrating controller. This is further accentuated by a large integration gain. The observed oscillations suggest that the currently used value $(1/p)$ is too large.

Controller stability

In particular, when a problem integrated by an explicit method becomes stiff, the stepsize will be limited by the numerical stability requirement. In that situation, the standard stepsize control increases the stepsize until the numerical stability is lost. The estimated truncation errors in subsequent steps will then be large, forcing the stepsize to be reduced until stability is regained. This process repeats itself, causing the stepsize to oscillate. The dead-zone may sometimes prevent such oscillations.

For embedded explicit Runge-Kutta methods, the oscillation phenomenon has recently been studied in [Hall 1985, Hall 1986], and earlier in [Shampine 1975]. Shampine showed for the linear test equation $\dot{y} = \lambda y$, that when this type of instability occurs, the *average* stepsize $\bar{h}$ will place $\bar{h}\lambda$ on the boundary of the stability region. Hall investigated the stability of the standard control algorithm and gave stability test criteria for real [Hall 1985] and complex [Hall 1986] values of λ, respectively. More recently, these criteria have been used [Hall & Higham 1987, Higham & Hall 1987] to construct new embedded Runge-Kutta methods for which the standard stepsize control is stable when h becomes limited by the numerical stability requirement.

It may be argued that when numerical stability limits the stepsize, the asymptotic relevance[1] of the error estimate used for stepsize control is questionable. This is true unless the effect of numerical instability is negligible. However, as long as numerical stability is maintained, the error estimate may be considered relevant. Therefore, it is most important that the controller acts correctly to prevent numerical instability.

It is also important to realize that the asymptotic relation between the error r and the stepsize h is a rather weak argument for selecting the integration gain $1/p$ in the design of the controller. Both from the numerical and the control theoretic points of view, it is more important that the algorithm manages to maintain numerical stability and control the errors. For control purposes it is, strictly speaking, of no concern whether or not an asymptotic relation is used to achieve this end.

Therefore, our approach is different from that taken by Higham and Hall [1987]. Rather than constructing new numerical methods that go

[1] By asymptotic relevance we mean that the error estimate should represent the dominating part of the true error.

together well with the standard controller, we have opted for the design of a new controller that can be tuned to perform well for almost any method. In addition, we shall drop a few old techniques. First, we do not use a dead-zone. The stepsize sequences obtained with the new controller will in general be smoother. Second, we omit the safety factor γ. We believe that the user-specified tolerance *tol* should also be the set point of the control algorithm.

3. A New Stepsize Control Algorithm

The pure integrating controller often performs quite well. We therefore choose to generalize this structure and suggest the use of a standard discrete PI controller [Franklin et al. 1986]. P stands for *proportional* and I for *integral*. The output of such a controller is formed as a sum of two components. The first component is directly proportional to the control error and the second is proportional to the integral of the control error. In the controller $\log(tol)$ is regarded as set point, $\log(\|r\|)$ as plant output and $\log(h)$ as control signal.

A PID controller has also been tested. The output of such a controller contains a third component, proportional to the time *derivative* (D) of the control error. In simulations it was noted that the influence of the D-part on the controller's performance was insignificant. This is to be expected based on the assumptions of a static plant (2). In the following we therefore only present the PI controller.

The PI Controller

The plant is discrete-time. It takes a sequence of stepsizes $\{h_n\}_{n=1}^{N}$ as input and produces a sequence of errors $\{r_n\}_{n=1}^{N}$ as output. The discrete PI controller is derived from the corresponding continuous time equivalent, by replacing the integration with a summation. We get the following expressions:

$$\begin{aligned} e_n &= \log(tol) - \log\left(\|r_n\|/h_n\right) \\ P_n &= K_P \cdot e_n \\ I_n &= I_{n-1} + K_I \cdot e_n \\ h_{n+1} &= \exp(P_n + I_n) \end{aligned} \tag{9}$$

where K_P is the proportional gain and K_I is the integration gain.

Dead-zone on Stepsize Changes

The PI controller performs very well (see Section 4). It manages to control the error better than the old algorithm, but at the price of many small stepsize changes. It can be argued that a good controller should keep the number of stepsize changes down, since in certain methods (e.g. multistep methods) changing the stepsize may be an expensive operation. This is certainly true if the stepsize sequence is irregular and contains large changes. Irregular stepsize changes may cause instability in such methods, and large individual changes may imply expensive operations, such as refactorizations of the Jacobian. However, if the stepsize sequence is smooth and regular there is little or no need to prevent stepsize changes. Since this is normally the case with the new algorithm, we have chosen to omit the dead-zone.

Even if the dead-zone can be omitted, there is still a need to limit the stepsize increase. The same limitation factor as in the standard algorithm is used. Due to the limitation, anti-windup is incorporated into the integration part.

Rejected Steps

Occasionally, it will happen that the suggested stepsize gives rise to an unacceptable error ($\|r\|/h > \rho \cdot tol$). One cause may be sudden changes in the differential equations, affecting $\log \|\phi\|$ (and hence G_p), that call for a drastic decrease in stepsize. In such events the step will be rejected. The algorithm will keep on rejecting steps until a step giving an acceptable error is found. Although this is perfectly fine from the point of view of the controller, it is not an effective way to produce the solution of the differential equations.

Ideally one would like to have a controller with good stabilizing properties and with fast response to track the changing properties of G_p accurately, thus quickly resolving situations with rejected steps. Unfortunately these properties are conflicting. The new controller is designed to have better stabilizing properties than the old one. This will also make it a little slower when following transients. Normally this is advantageous since it produces smoother stepsize sequences, but in connection with rejected steps it may cause longer standstills.

The problem is resolved by using two parameter sets. The first set is chosen to optimize the stabilizing behavior of the algorithm. Parameter set two gives a faster response and is used when a step has been rejected. Typically, parameter set two is used in only a few percent of the calls to the stepsize control algorithm.

Complete Algorithm

Finally we state the complete control algorithm. The anti-windup and the limitation on stepsize increase have been included.

$$\begin{aligned} e_n &= \log(tol) - \log\left(\|r_n\|/h_n\right) \\ P_n &= K_P \cdot e_n \\ I_{temp} &= I_{n-1} + K_I \cdot e_n \\ h_{temp} &= \exp(P_n + I_{temp}) \\ h_{n+1} &= \begin{cases} \theta_{max} h_n, & \text{if } h_{temp} > \theta_{max} h_n \\ h_{temp}, & \text{otherwise} \end{cases} \\ I_n &= I_{temp} + (\log h_{n+1} - \log h_{temp}). \end{aligned} \tag{10}$$

The algorithm can be rewritten on a form resembling (5). Some manipulations, similar to the ones establishing (7), yield

$$\begin{aligned} h_{temp} &= \left(\frac{tol}{\|r_n\|/h_n}\right)^{K_I} \left(\frac{\|r_{n-1}\|/h_{n-1}}{\|r_n\|/h_n}\right)^{K_P} h_n \\ h_{n+1} &= \begin{cases} \theta_{max} h_n, & \text{if } h_{temp} > \theta_{max} h_n \\ h_{temp}, & \text{otherwise.} \end{cases} \end{aligned} \tag{11}$$

Thus, apart from giving a new value to K_I (i.e. $K_I < 1/p$), the new algorithm includes a correction factor in the relation (5). This factor will make the new stepsize depend not only on the current error per unit step $\|r_n\|/h_n$ but also on its most recent development.

Repeated simulations have suggested the following two parameter sets.

Set for normal case	Set for rejected case
$K_P = 0.13$	$K_P = 0$
$K_I = 1/15$	$K_I = 1/5$
$\theta_{max} = 2.0$	$\theta_{max} = 2.0$
$\rho = 1.2$	$\rho = 1.2$

4. Numerical Tests

The integration method used in this paper is DOPRI45 [Hairer et al. 1987], a fourth order Runge-Kutta method with an embedded fifth order error estimate. It was implemented as a PASCAL system in the simulation package Simnon [Elmqvist et al. 1986], which gives a convenient way to change parameters in the routine. There are also good plotting facilities included in the package.

Basic properties of the new controller

A first set of test runs solving Problem 6 (all problems can be found in the Appendix) shows how some of the differences between the new and the old algorithm affect the stepsize. All quantities are plotted as functions of time. The stepsize for a pure integrating controller with dead-zone (old strategy) is shown in Figure 2a. Figure 2b shows the effect of removing the dead-zone. In Figure 2c the integration gain K_I is changed from 1/4 to 1/15. (Note that in the original stepsize control, the integration gain was $1/p$, where p is the order of the method under consideration). Addition of the proportional term further improves the performance as seen in Figure 2d.

The next three test runs (Figure 3) show that it is possible to drive the estimated local error per unit step to the desired value *tol*. The tests also demonstrate the stability of the new controller. The problem is Problem 6, and three different tolerances have been used. In the error plots, the estimated local truncation error is normalized to *tol*.

For $tol = 10^{-2}$ and 10^{-5} we note that the stepsize reaches the same "steady-state" level. (The stepsize is not constant, but decreasing very

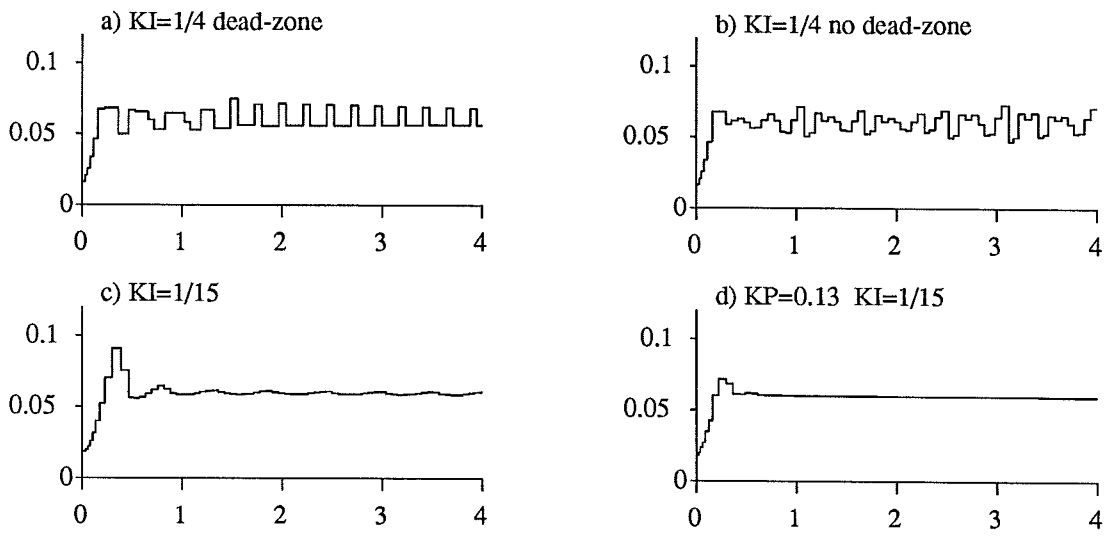

Figure 2. The effect on stepsize of some of the differences between the old and the new algorithm (Problem 6).

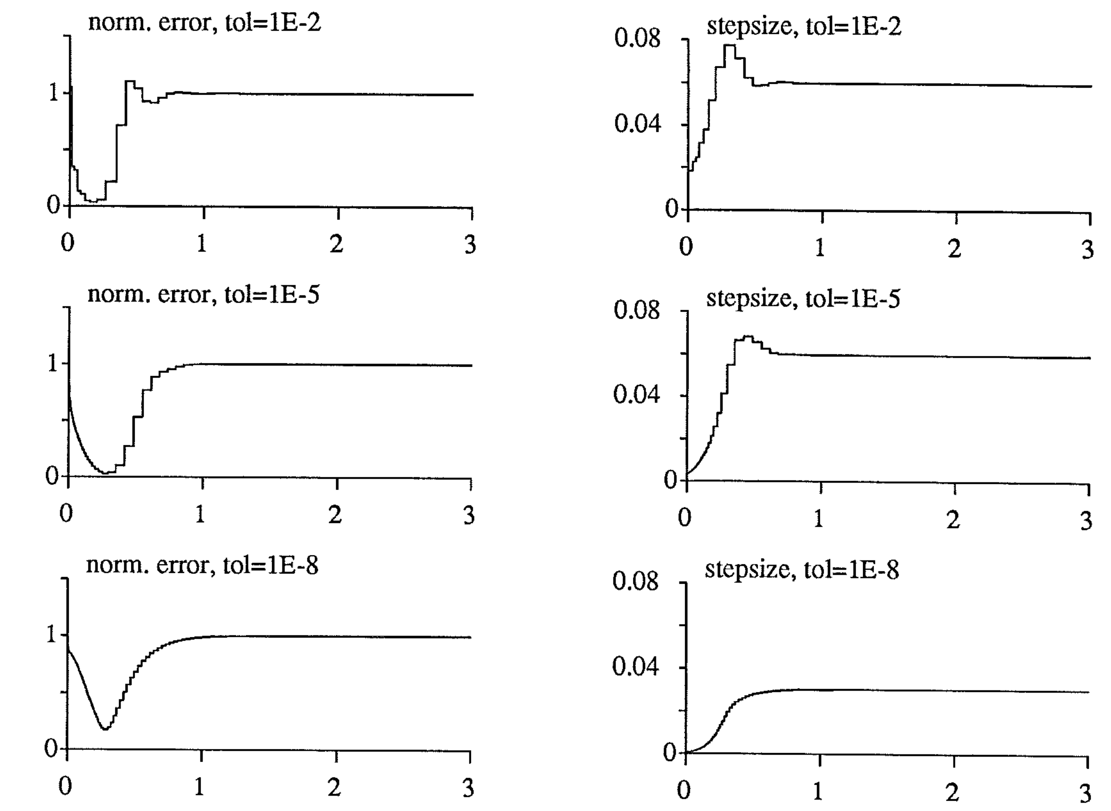

Figure 3. Error estimate and stepsize for different tolerances (Problem 6). New controller.

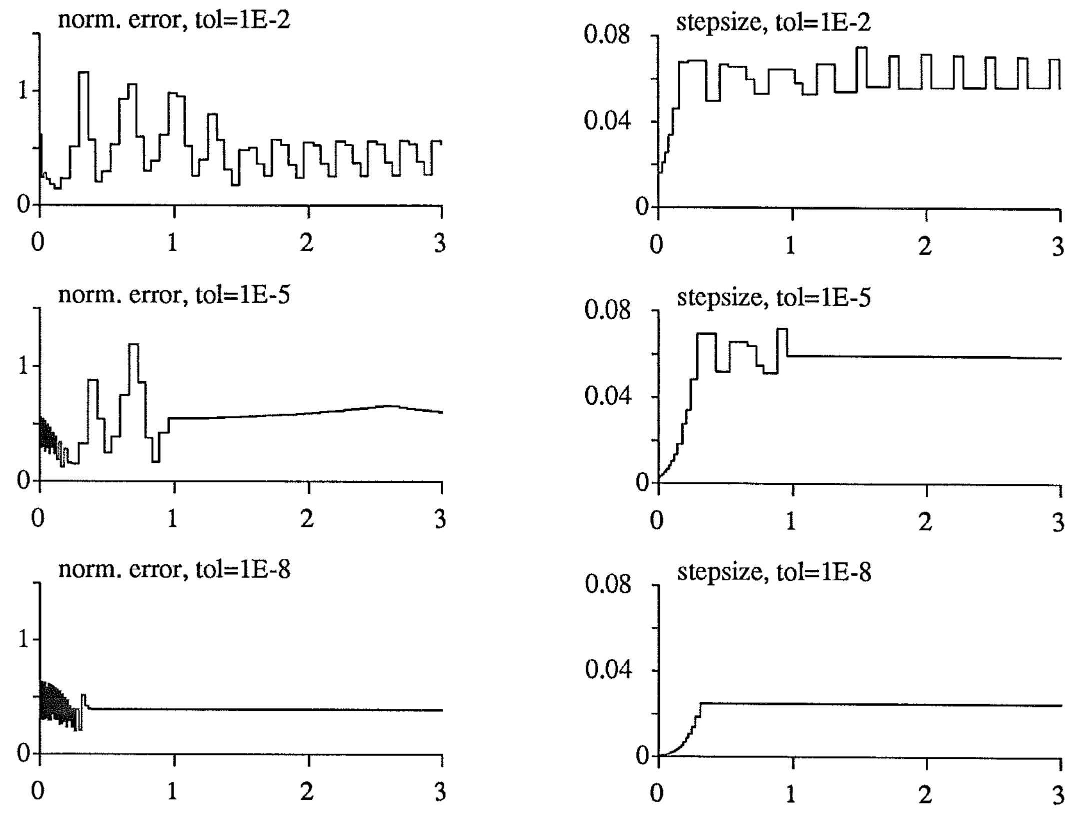

Figure 4. Error estimate and stepsize for different tolerances (Problem 6). Old controller.

slowly; this is due to a slow change in the nonlinear character of the problem). In fact, for any tolerance $tol \geq 10^{-6}$, we reach the same level. Since the stepsize is independent of the accuracy requirement in this interval, it is limited by the numerical stability requirement, but without any stepsize oscillations. Finally, for $tol = 10^{-8}$, the tolerance is tight enough to prevent the stepsize from reaching the stability limit. Hence, for the latter tolerance, the problem is no longer stiff.

It is interesting to note that, since dr/dh changes rapidly with h when the stepsize is limited by numerical stability (see also Figure 5), one would obtain a much improved accuracy in the numerical solution by taking steps only slightly shorter than the maximum stable stepsize. However, since the maximum stable stepsize is independent of tol, this cannot be achieved by using a smaller tol. In fact, we are not aware of any control technique that would achieve this desirable goal; any controller will (and should!) increase the stepsize if the control error is positive.

If the same problem is solved using the old controller (Figure 4), the stepsize oscillates at $tol = 10^{-2}$. For $tol = 10^{-5}$, the dead-zone manages to prevent oscillations for $t > 1$, but without the dead-zone the oscillations return. At $tol = 10^{-8}$ there are no oscillations, but the safety factor $\gamma = 0.9$ leads to a shorter steady-state stepsize than that used by the new controller. On the other hand, if the safety factor is dropped in the old controller, there is a significant increase in the number of rejected steps. Thus it seems as if the safety measures implemented in the old controller lead to inefficiency. These results are typical for all problems of a similar character.

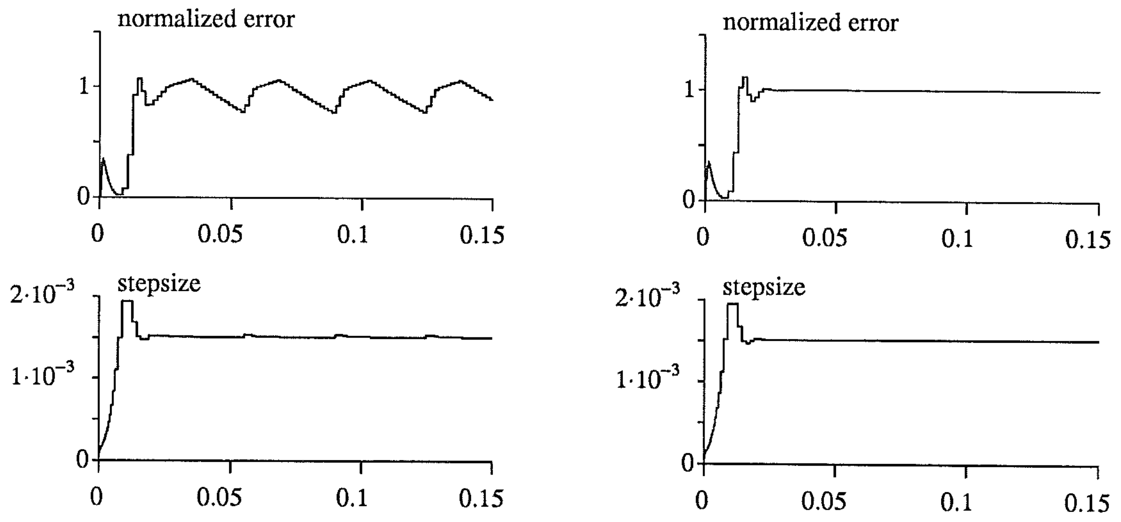

Figure 5. New controller with (left) and without (right) dead-zone.

In Figure 5 we see the effect of incorporating a very small dead-zone ($\theta_{hi} = 1.02$, $\theta_{lo} = 0.996$) into the new controller. Its most significant effects occur when numerical stability limits the stepsize. The plots (Problem 3) show that even the most minute stepsize changes will cause a rapid growth or decay in the error estimate. The result is the ripple in the graph of the normalized error. We see no reason for using a dead-zone in the new controller, and consequently it has been omitted.

Comparative tests

A number of differential equations have been solved with the new stepsize control algorithm. Its performance has been compared with the old control algorithm. The results are shown in Figures 6 – 13, with each figure consisting of six small plots. The upper left shows the solution of the differential equation. In the upper right, two curves appear showing the cost for solving the differential equation. It is the number of integration routine calls for the old (solid line) and for the new method (dashed line). Note that this includes rejected steps in order to reflect the total work properly. The two plots in the middle show the estimated local error per unit step ($\|r_n\|/h_n$) for the old (left) and the new (right) method. The value is normalized to *tol*. The two lower plots compare the stepsize for the methods.

The first of these comparative simulations solves Problem 1. This is a system where both controllers have some difficulties. Even for constant steps, the slowly damped oscillations would lead to fluctuating errors to which the controllers respond.

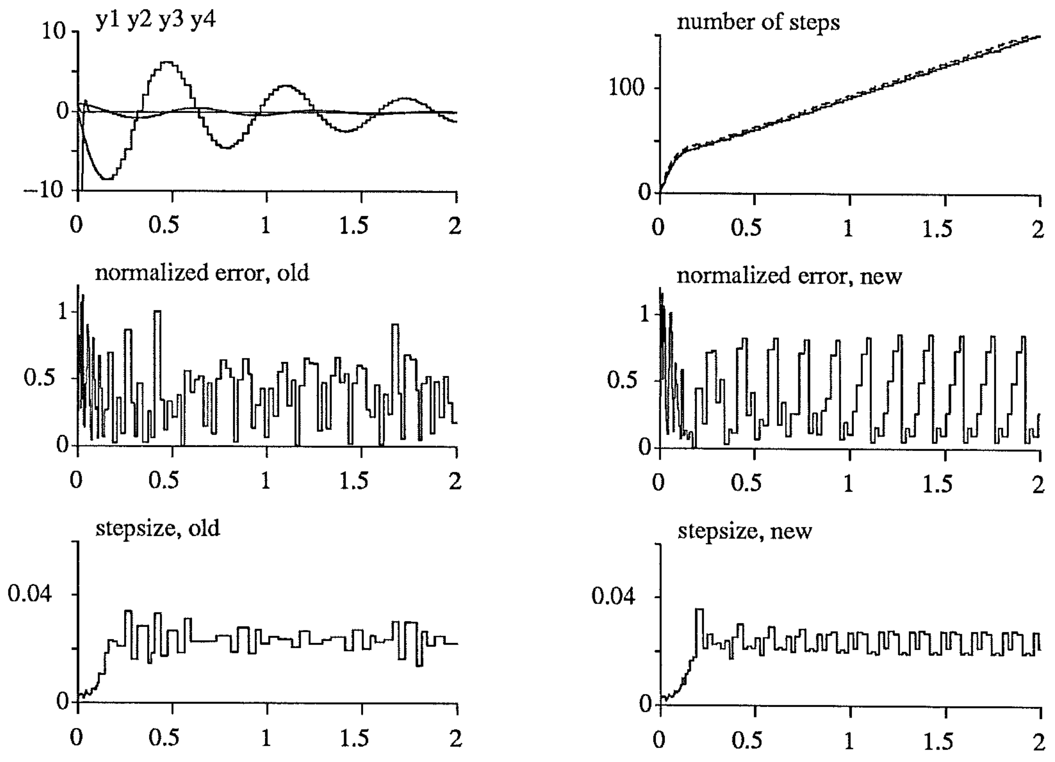

Figure 6. Solving Problem 1, $tol = 0.01$.

Next, Problem 2 is solved. The new controller quickly finds the maximum stable stepsize and stays there without oscillations. The total work is reduced by some 10-15%. A similar performance can be seen in Problem 3. In this case, the oscillations in the old controller show a remarkable periodicity, with an approximate period of 40 steps.

Problems 4 and 5 are van der Pol oscillators. The first is nonstiff and the second is moderately stiff, for the tolerances used. In the first case, the two controllers have a similar performance, with the old controller slightly

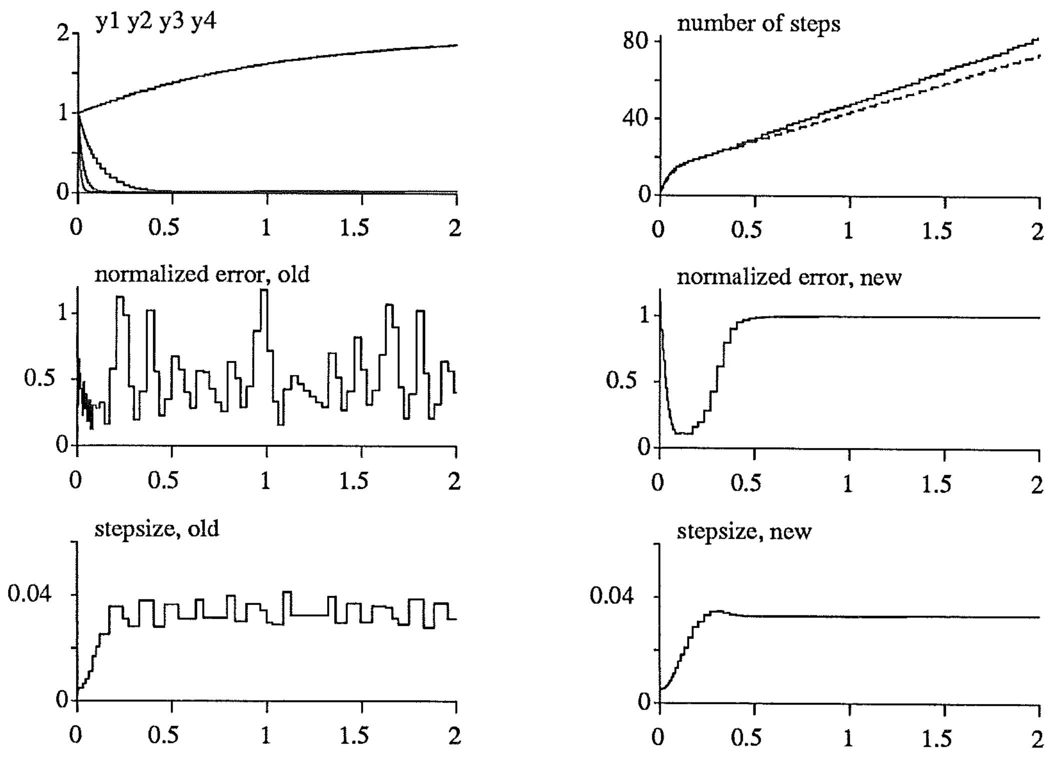

Figure 7. Solving Problem 2, $tol = 0.01$.

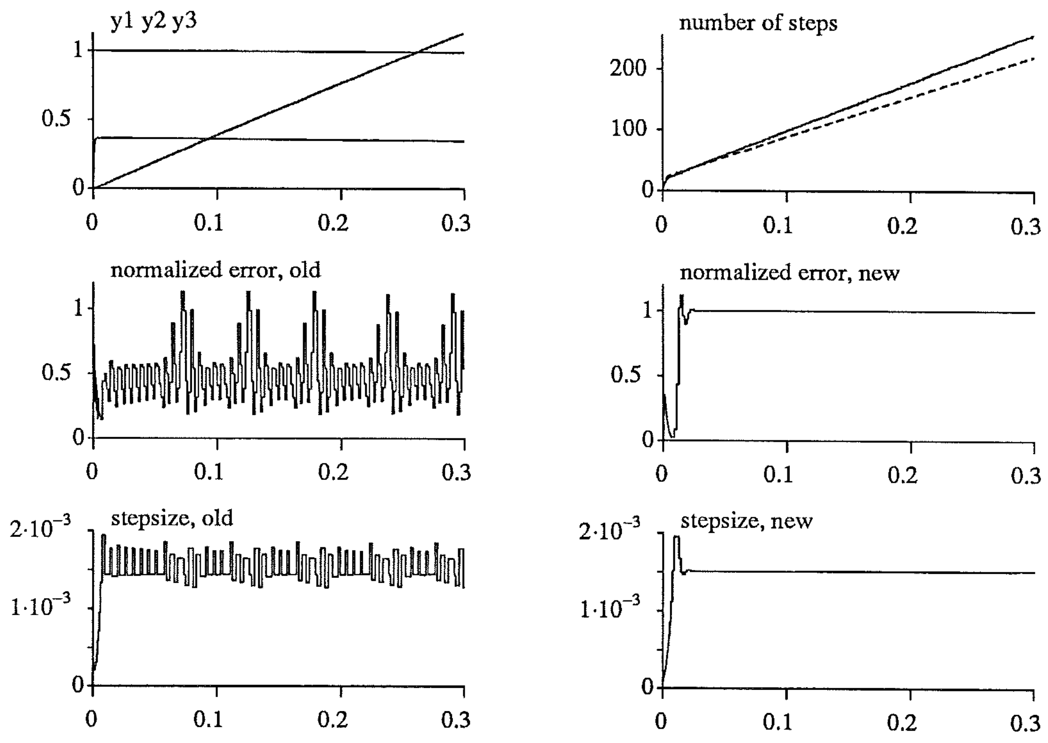

Figure 8. Solving Problem 3, $tol = 0.01$.

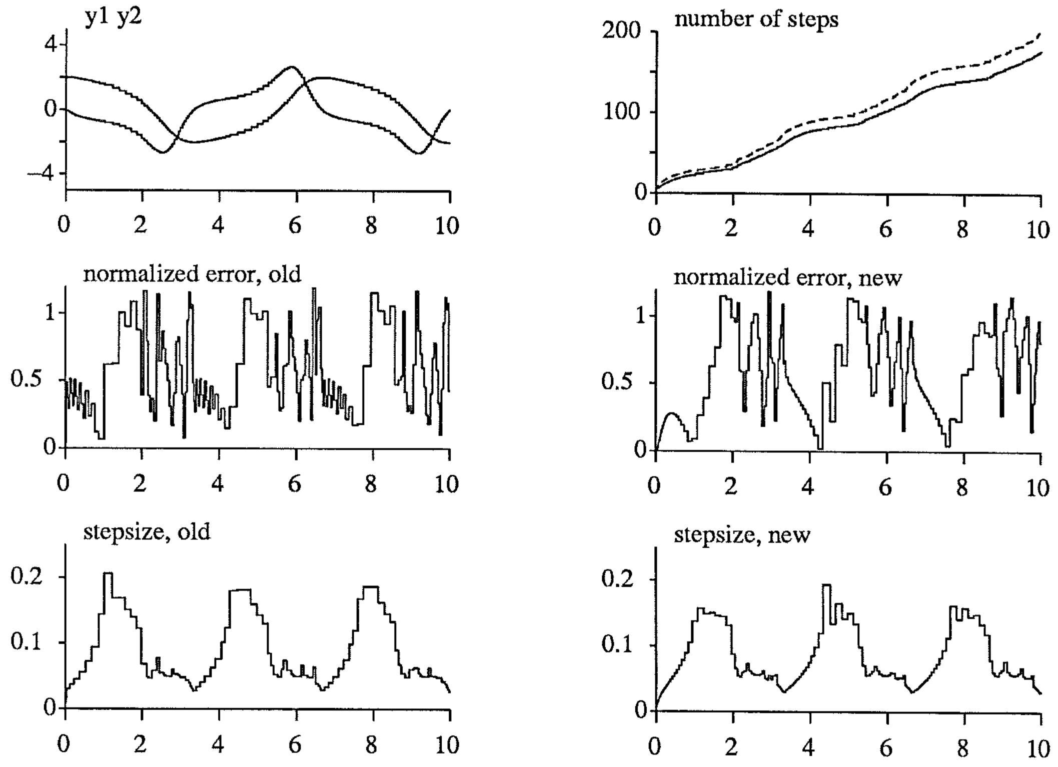

Figure 9. Solving Problem 4, $tol = 10^{-6}$.

more efficient. In the latter, the new controller has a much smoother behavior, while at the same time being marginally faster. Note that in the graph of the solution, the spikes have been clipped.

Problem 6, a chemical kinetics problem, is another typical example, showing the superior stabilizing effect of the PI controller. The efficiency is increased by 20% .

Problems 7 a and b are so-called "Brusselators", a type of nonlinear oscillating system arising in chemical kinetics. The first is non-stiff and the second moderately stiff. The behavior of the two controllers is very similar to that observed for the van der Pol oscillators. Thus, in the nonstiff system the controllers perform similarly. Note that the character of the solution changes so rapidly that neither control algorithm is able to obtain a smooth error graph. In the stiffer version of the system, on the other hand, the new controller achieves a smoother performance at the same cost.

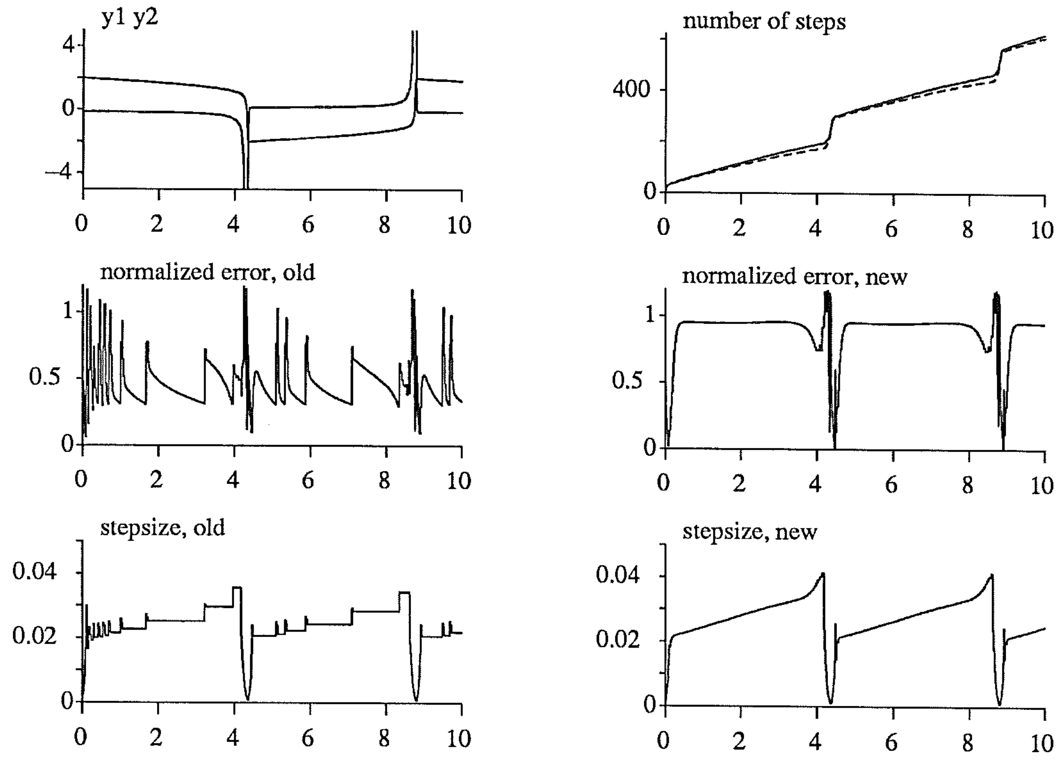

Figure 10. Solving Problem 5, $tol = 10^{-4}$.

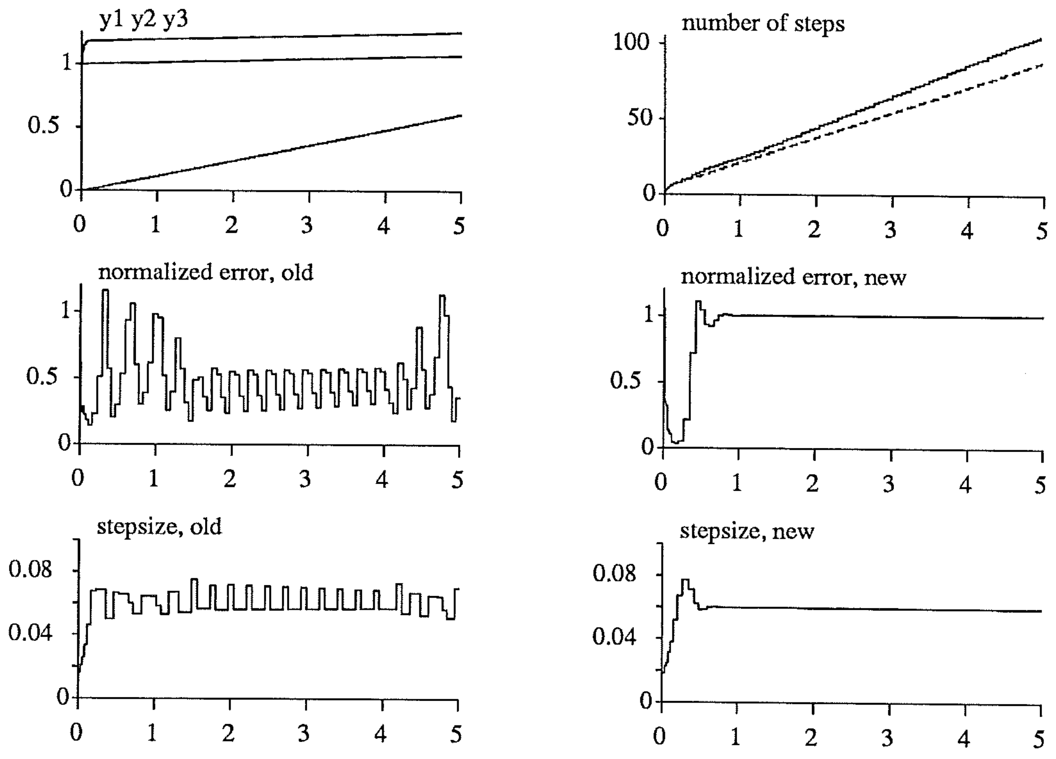

Figure 11. Solving Problem 6, $tol = 0.01$.

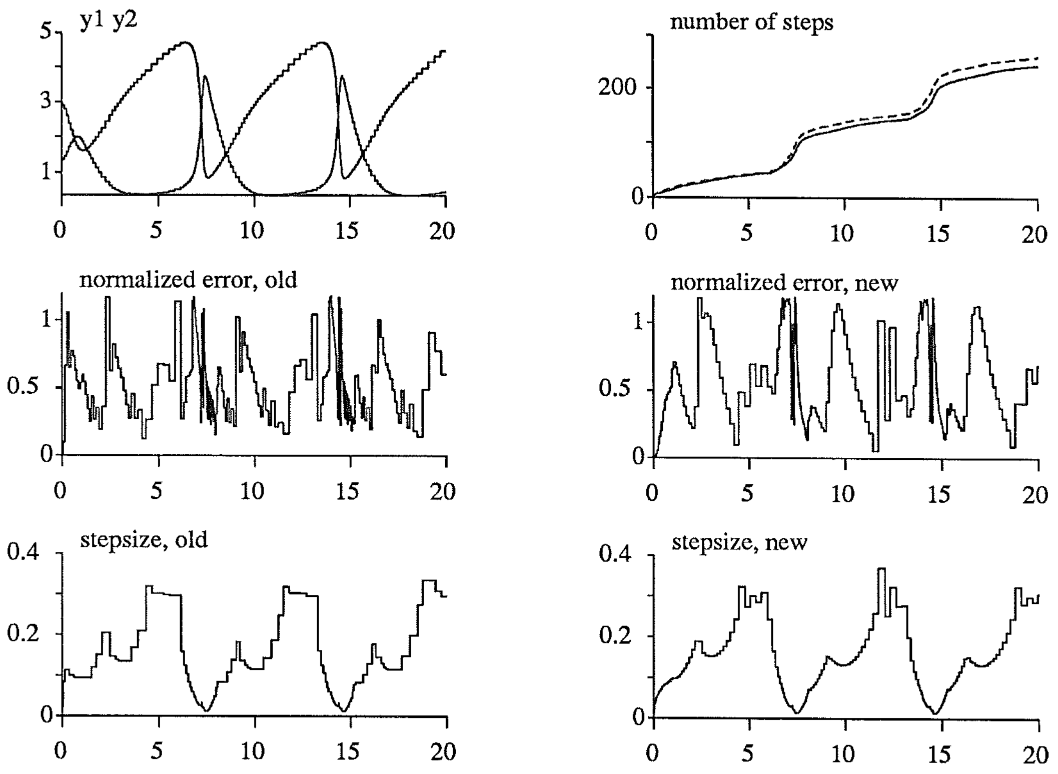

Figure 12. Solving Problem 7 a, $tol = 10^{-6}$.

5. Conclusions

By using standard control theory much insight and understanding of the stepsize control problem can be gained. The discussion of the standard control algorithm (Section 2) explains why it sometimes results in oscillations. A remedy is to use a PI control algorithm. In particular, the proportional part improves the stability of the controller, which yields better results and a more consistent performance.

This is particularly evident in problems where the stepsize becomes limited by numerical stability. One might argue that this is of little importance since problems of this type arise only rarely in nonstiff problems and never in stiff problems if a proper integration method is selected. However, we believe that the new algorithm significantly improves the robustness of the stepsize control at little or no extra expense. Moreover, this improvement may certainly be important for moderately stiff problems, in the transition from nonstiff to stiff and in connection with the implementation of type-insensitive codes intended for both classes of problems.

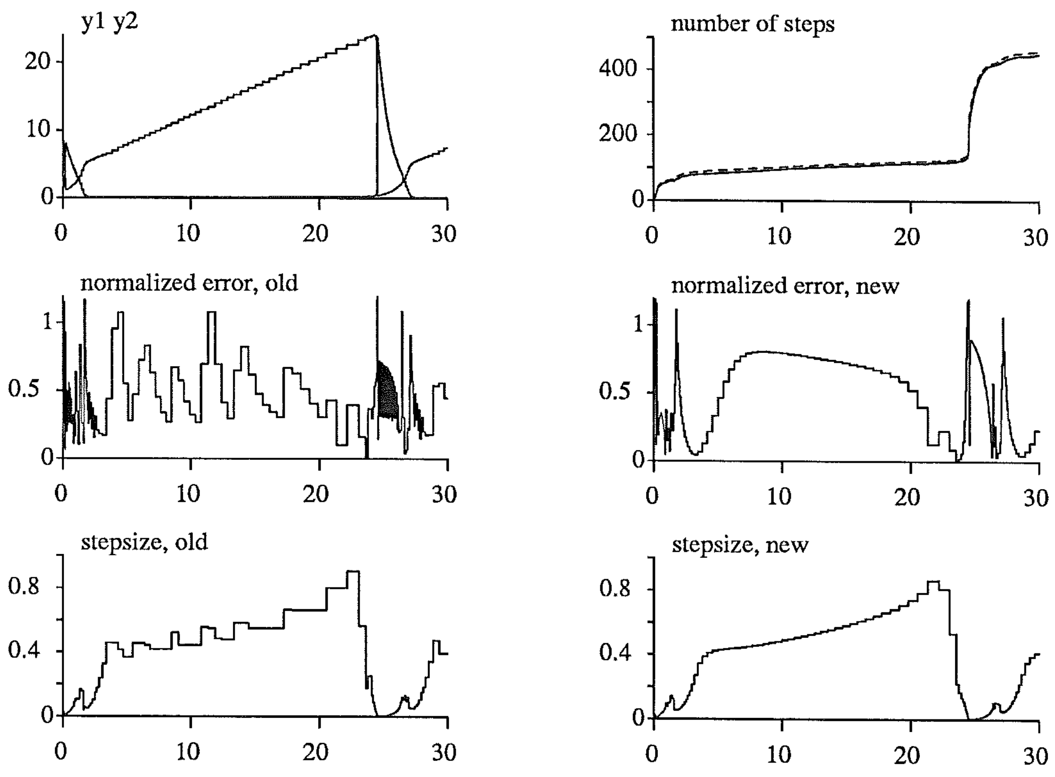

Figure 13. Solving Problem 7 b, $tol = 10^{-4}$.

Acknowledgements

The authors wish to thank Per Hagander and Ola Dahl for many valuable discussions, and Mats Andersson and Per-Olof Olsson for implementing the DOPRI45 routine.

References

ELMQVIST, H, K. J. ÅSTRÖM and T. SCHÖNTHAL (1986): *SIMNON, User's Guide for MS-DOS Computers*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ENRIGHT, W. H, T. E. HULL and B. LINDBERG (1975): "Comparing Numerical Methods for Stiff Systems of ODE's," *BIT*, 15, 28 – 33.

FRANKLIN, G. F, J. D. POWELL and A. EMAMI-NAEINI (1986): *Feedback Control Systems*, Addison-Wesley, pp. 99 – 103.

GEAR, C. W. (1971): *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall.

HAIRER, E, S. P. NØRSETT, G. WANNER (1987): *Solving Ordinary Differential Equations, I*, Springer.

HALL, G. (1985): "Equilibrium States of Runge-Kutta Schemes: Part I," *ACM Transactions on Mathematical Software*, **11**, 3, 289 – 301.

HALL, G. (1986): "Equilibrium States of Runge-Kutta Schemes: Part II," *ACM Transactions on Mathematical Software*, **12**, 3, 183 – 192.

HALL, G. and D. J. HIGHAM (1987): "Analysis of Stepsize Selection for Runge-Kutta Codes," NA report No. 137, University of Manchester.

HIGHAM, D. J. and HALL, G. (1987): "Embedded Runge-Kutta Formulae with Stable Equilibrium States," NA report No. 140, University of Manchester.

LINDBERG, B. (1977): "Characterization of Optimal Stepsize Sequences for Methods for Stiff Differential Equations," *SINUM*, **14**, 859 – 887.

SHAMPINE L. F. (1975): "Stiffness and Nonstiff Differential Equation Solvers," in L. Collatz (Ed.): *Numerische Behandlung von Differential Gleichungen*, Information Series of Numerical Mathematics 27, Birkhauser Verlag, Basel, pp. 287–301.

ÅSTRÖM, K. J. and B. WITTENMARK (1984): *Computer Controlled Systems, Theory and Design*, Prentice-Hall, Englewood Cliffs, New Jersey, pp. 369 – 373.

Appendix

Problem 1 Problem B1 in [Enright et al. 1975].

$$\begin{aligned} \dot{y}_1 &= -y_1 + y_2 & y_1(0) &= 1.0 \\ \dot{y}_2 &= -100y_1 - y_2 & y_2(0) &= 0.0 \\ \dot{y}_3 &= -100y_3 + y_4 & y_3(0) &= 1.0 \\ \dot{y}_4 &= -10000y_3 - 100y_4 & y_4(0) &= 0.0 \end{aligned}$$

Problem 1 Problem C2 in [Enright et al. 1975] with $\beta = 0.1$.

$$\begin{aligned} \dot{y}_1 &= -y_1 + 2 & y_1(0) &= 1.0 \\ \dot{y}_2 &= -10y_2 + \beta y_1^2 & y_2(0) &= 1.0 \\ \dot{y}_3 &= -40y_3 + 4\beta \cdot (y_1^2 + y_2^2) & y_3(0) &= 1.0 \\ \dot{y}_4 &= -100y_4 + 10\beta \cdot (y_1^2 + y_2^2 + y_3^2) & y_4(0) &= 1.0 \end{aligned}$$

Problem 3 Problem D2 in [Enright et al. 1975].

$$\begin{aligned} \dot{y}_1 &= -0.04y_1 + 0.01y_2y_3 & y_1(0) &= 1.0 \\ \dot{y}_2 &= 400y_1 - 100y_2y_3 - 3000y_2^2 & y_2(0) &= 0.0 \\ \dot{y}_3 &= 30y_2^2 & y_3(0) &= 0.0 \end{aligned}$$

Problem 4 Problem E2 in [Enright et al. 1975].

$$\begin{aligned} \dot{y}_1 &= y_2 & y_1(0) &= 2.0 \\ \dot{y}_2 &= (1 - y_1^2)y_2 - y_1 & y_2(0) &= 0.0 \end{aligned}$$

Problem 5 Problem E2 in [Enright et al. 1975] (sligthly changed).

$$\begin{aligned} \dot{y}_1 &= y_2 & y_1(0) &= 2.0 \\ \dot{y}_2 &= 50(1 - y_1^2)y_2 - 10y_1 & y_2(0) &= 0.0 \end{aligned}$$

Problem 7 Brusselator, *a.* $\beta = 3.0$, *b.* $\beta = 8.533$.

$$\begin{aligned} \dot{y}_1 &= 1.0 + y_1^2y_2 - (\beta + 1.0)y_1 & y_1(0) &= 1.3 \\ \dot{y}_2 &= \beta y_1 - y_1^2y_2 & y_2(0) &= \beta \end{aligned}$$

Analysis and Synthesis of Stepsize Control in Numerical Solution of Ordinary Differential Equations

Kjell Gustafsson

Department of Automatic Control
Lund Institute of Technology, P.O. Box 118
S-223 57 LUND, Sweden

Abstract

The problem of automatic stepsize control in numerical solution of differential equations is approached using control theory. Although being an automatic control problem this point of view is seldom pursued.

Standard stepsize control is derived assuming a static relation between the stepsize and local truncation error in the integration method. In many cases such a model is not sufficient, and instead a dynamic model has to be used. Here a dynamic model for explicit Runge-Kutta methods is derived. The model is verified using numerical tests and system identification.

The derived model is used to analyze standard stepsize control. The analysis gives insight and leads to a good understanding of the properties of the control system. In particular, with the model, it is straightforward to explain the problems of oscillating stepsize sequences often encountered when using explicit Runge-Kutta methods to solve stiff differential equations.

Moreover, the acquired understanding is used to construct a new stepsize control algorithm which gives superior performance at little extra expense. The new controller also overcomes the stepsize oscillation problems.

1. Introduction

The choice of stepsize in numerical integration of ordinary differential equations is a trade off between efficiency and accuracy. For efficiency, the stepsize should be large in order to reach the end of the integration interval as quickly as possible. On the other hand, accuracy requirements impose limits on the largeness of the stepsize.

To meet the user-specified accuracy requirements, the strategy is normally to adjust the stepsize to keep an estimate of the local truncation error per unit step bounded. This leads to a standard stepsize control algorithm: at each step a new stepsize is chosen trying to make the next error estimate equal a tolerance related to the user-specified accuracy requirement.

Standard Stepsize Control Anomalies

The standard stepsize control algorithm normally performs quite well. However, for two types of differential equations and integration methods, its performance is unacceptable. The first is nonstiff integration methods applied to stiff differential equations. In stationarity this often results in a stepsize sequence that oscillates violently, wasting computer time recalculating rejected steps and changing the stepsize. Moreover, the nonsmooth stepsize sequence may excite modes that the error estimator fails to recover properly, leading to an erroneous solution. Figure 1 shows an example from a simulation of a small control system (problem 3 in Appendix D). The oscillatory component in the signal has no correspondence in the physical world, but is caused by an oscillating stepsize sequence. By improving the stepsize control algorithm the artifact can be removed, yielding a correct simulation result.

The second case, in which the standard stepsize controller does not perform well, is differential equations with drastic changes in behavior. Here, one often finds long sequences of alternating accepted and rejected steps, resulting in much computing time spent uselessly. A simulation of the Brusselator [Hairer et al. 1987] (problem 5 in Appendix D) is shown in Figure 2. Just before and during the large state transitions at $t = 24.5$, there are many rejected steps, which partly can be attributed to poor stepsize control.

Earlier studies of the problem of oscillating stepsize sequences have

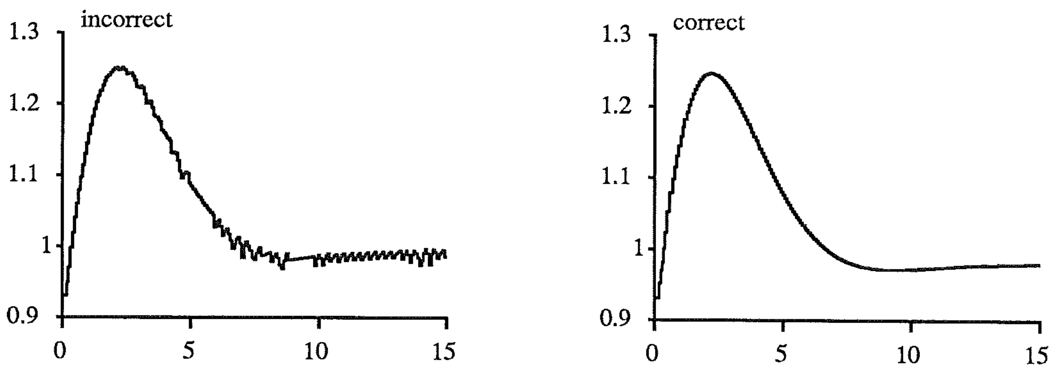

Figure 1. A signal drawn from a control system simulation. The oscillatory component, in the signal to the left, is caused by an irregular stepsize sequence. The correct signal, to the right, is obtained by improving the stepsize control algorithm.

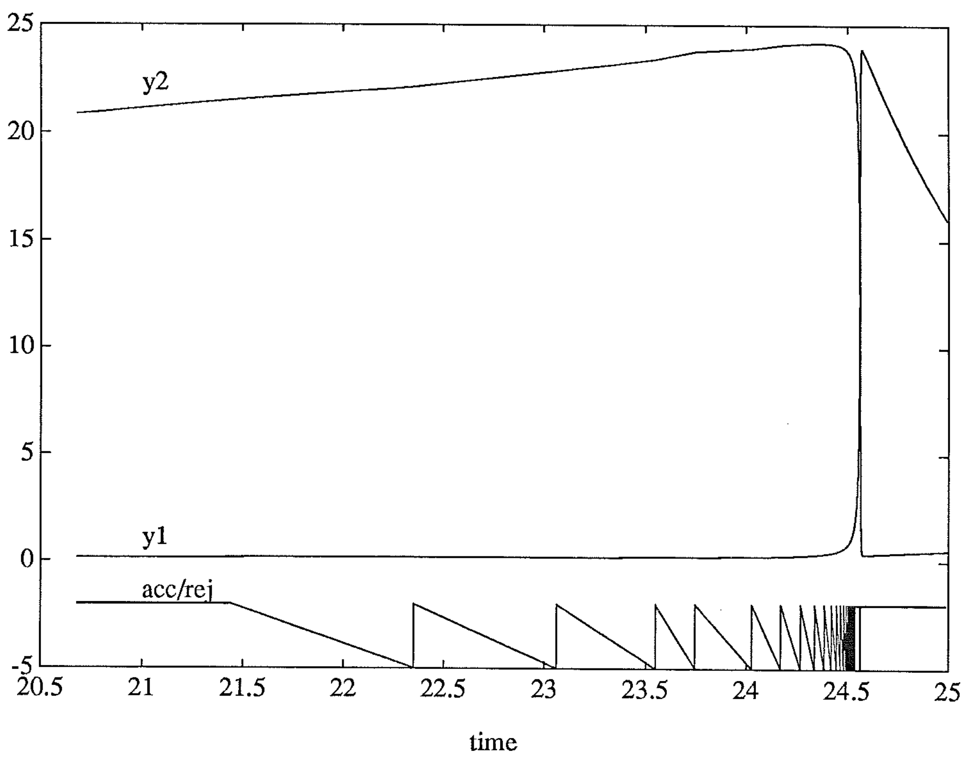

Figure 2. A simulation of the Brusselator. The two upper curves are the state variables in the Brusselator, while the lower third curve indicates if the step in the integration method was rejected (level -5 in the plot) or accepted (level -2 in the plot). When the problem changes character at $t = 24.5$ many steps are rejected. From $t = 21.0$ to $t = 24.5$ there are 24 accepted and 21 rejected steps.

focused mainly on describing and characterizing the behavior [Shampine 1975, Hall 1985, Hall 1986, Hall and Higham 1987]. An exception is the recent study [Higham and Hall 1987], in which an explicit Runge-Kutta method is modified to behave properly together with the standard stepsize controller. The modification trades the stability region and/or the precision of the integration method for a more well behaved stepsize sequence. Another interesting early study is [Zonneveld 1964], where an extrapolation scheme on stepsizes is used to try to produce a smoother stepsize sequence.

The Proposed Solution

The problem of stepsize control can be analyzed using tools from control theory. The analysis, which will be presented later, reveals the causes for the misbehavior described above. It also gives a solid foundation from which new algorithms for stepsize adjustment can be derived.

By using logarithmic variables, the standard stepsize controller can be expressed as a discrete time I-controller. I stands for *integrating* and reflects the fact that the controller forms its output (the logarithm of the stepsize) as the integral* of its input (the control errors). To resolve the problems with oscillating stepsize sequences, the use of a discrete time PI-controller *(proportional, integrating)* [Gustafsson et al. 1988] is suggested. Such a controller is of the form

$$h_n = \left(\frac{tol}{r_n}\right)^{k_I} \left(\frac{r_{n-1}}{r_n}\right)^{k_P} h_{n-1}$$

which should be compared with the standard control algorithm

$$h_n = \left(\frac{tol}{r_n}\right)^{1/k} h_{n-1}.$$

In both expressions h is the stepsize, r the estimated error, and tol the user-specified tolerance. Coefficients k_I and k_P are chosen depending on the integration method used. In the standard controller, k_I is chosen as

* Strictly speaking, the output is formed by *summing* the input since the controller is discrete time. However, since the controller is derived from its continuous time counterpart, the standard continuous time terminology will be used.

$1/k$ where k is the order of the error estimator in the selected integration method. This is not an optimal choice, but better control can be achieved by choosing k_I differently.

This new controller has been tested on a variety of problems and been found to behave properly [Gustafsson et al. 1988]. Not only is the problem with stepsize oscillations resolved, but the controller in general produces smoother stepsize sequences. As a result, the error estimates show a more regular behavior. The latter property might be particularly useful in multistep methods, since their numerical performance may depend significantly on the stepsize sequence. Ideally, the stepsize sequence should be as smooth as the solution itself.

In the proposed solution only the stepsize control algorithm is altered, while the integration method is kept as is. Consequently, a more well behaved stepsize sequence is obtained without sacrificing neither the stability region nor the precision of the integration method.

The analysis presented here is only pursued for explicit Runge-Kutta methods, although we believe that a control theoretic viewpoint is beneficial also for other types of integration methods.

2. Today's Stepsize Control

In this section the stepsize control found in most production codes written today will be motivated and derived. Although much of the discussion and the results probably are well known to the reader, the purpose is to introduce a control theoretic framework and also to sort out some matters of notation.

Viewed As A Control System

Throughout this article a control theoretic viewpoint of stepsize control will be pursued. In control theory a common problem is to design a controller for a given process. The purpose of the controller is to produce an input signal to the process such that the output of the process behaves in a certain way.

In our case the process consists of the integration method, the differential equation, and the error estimator. The process has one input: the stepsize h, and two outputs: the solution of the differential equation y and

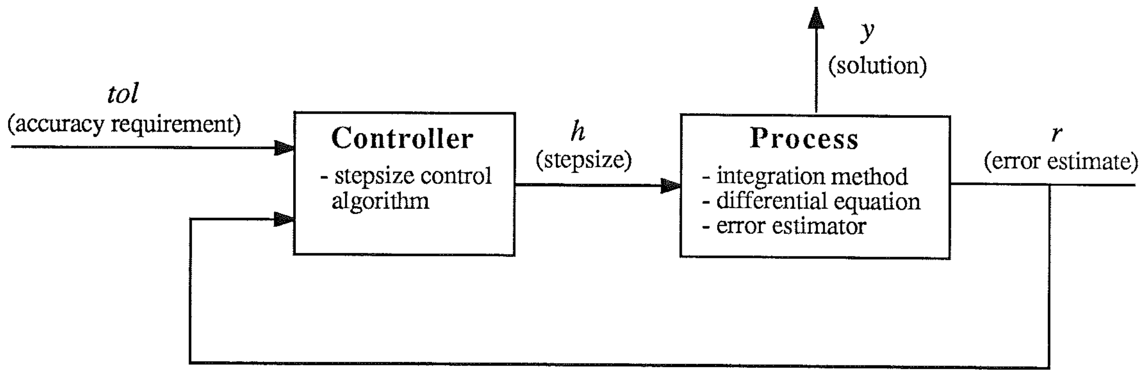

Figure 3. Control system view of stepsize control.

the error estimate r. A user regards y as the most important signal, but from the stepsize control point of view it is only a by-product. Instead, the useful output is the error estimate r, which tells the quality of the solution y. The stepsize controller uses this quality measure together with a user-specified tolerance *tol*, to decide upon the next stepsize to use, i.e the stepsize controller tries to control the behavior of the output r of the process using the stepsize h as control variable.

Control Objectives

Consider the initial value problem

$$\begin{aligned} \dot{y} &= f(t,y), \quad 0 \le t \le T, \qquad y,f \in \mathcal{R}^l \\ y(0) &= y_0 \end{aligned} \tag{1}$$

which has the exact solution $y(t)$. An integration method discretizes the problem and forms a solution $\{y_n\}_{n=0}^N$, where y_n is the numerical solution at time t_n.

Definition 1
The global error g_n is defined as the difference between the numerical solution and the exact solution, i.e. $g_n = y_n - y(t_n)$. □

When solving (1) the ultimate goal is to produce, in an efficient way, a numerical solution meeting some prescribed accuracy requirement. This means keeping the global error below some user specified level. Thus the main objective of the stepsize controller is to choose stepsizes such that the problem is solved efficiently under the constraint of producing a solution of acceptable quality.

To measure the quality of the solution implies the ability to estimate the global error, which is usually a difficult and expensive operation. However, the global error may be bounded in terms of the local truncation error per unit step [Dahlquist et al. 1974]. Therefore, by controlling this error, the goal of accuracy can be achieved.

DEFINITION 2
For an explicit one-step integration method, the local truncation error per step e_n is defined as $e_n = y_n - y(t_n)$, provided $y_{n-1} = y(t_{n-1})$. Moreover, the local truncation error per unit step d_n is defined by normalizing e_n with respect to the stepsize, i.e. $d_n = e_n/h_{n-1}$ where $h_{n-1} = t_n - t_{n-1}$.
□

Remark. The definition is generalized to explicit multistep methods by requiring $y_{n-j} = y(t_{n-j})$ for $j = 1, 2, \ldots, k$, where k is the number of steps in the method. For implicit integration methods a corresponding definition can be given, but that requires further elaboration.

The Process

For an integration method of order p the local truncation errors e and d depend on the stepsize asymptotically as

$$e \sim h^{p+1}, \qquad d \sim h^p$$

Most integration methods will produce estimates $\hat{e}_n$ or $\hat{d}_n$ of the local truncation errors e and d, while constructing the solution y_n. Typically an integration method implements two different updating formulae, and by comparing their individual results an error estimate is arrived at. Then

$$\hat{e} \sim h^{p_e}, \qquad \hat{d} \sim h^{p_e-1} \tag{2}$$

where p_e is the order of the error estimator. Normally one strives to get an estimate that is asymptotically correct making p_e equal $p+1$. However, methods exist in which $p_e \neq p+1$. One such exception is embedded Runge-Kutta methods using local extrapolation, in which the order of the error estimator equals the order of the method, i.e. $p_e = p$.

Independently of the relation between p_e and p, (2) may be written

$$\hat{e} = \phi h^{p_e}, \qquad \hat{d} = \phi h^{p_e-1} \tag{3}$$

with the coefficient vector ϕ being $O(1)$ as $h \to 0$. In addition, ϕ depends on the solution of the differential equation, and in this respect it may be regarded as a function of time.

To illustrate the discussion consider the following example.

EXAMPLE 1—Explicit Euler with error estimate using modified Euler
For an initial value problem in the form (1), the explicit Euler method results in the difference equation

$$y_{n+1} = y_n + h_n f(y_n) \tag{4}$$

For the same problem modified Euler results in

$$\bar{y}_{n+1} = y_n + \frac{h_n}{2}\left(f(y_n) + f(y_{n+1})\right)$$

with y_{n+1} defined by (4). The local truncation error per step resulting from one explicit Euler step can be estimated by the difference between y_{n+1} and $\bar{y}_{n+1}$, i.e.

$$\begin{aligned} \hat{e}_{n+1} = y_{n+1} - \bar{y}_{n+1} &= \frac{h_n}{2}\left(f(y_n) - f(y_{n+1})\right) \\ &= \frac{h_n}{2}\left(f(y_n) - f(y_n + h_n f(y_n))\right) \end{aligned} \tag{5}$$

$$\hat{e}_{n+1} \approx -\frac{h_n^2}{2} f'(y_n) f(y_n). \tag{6}$$

To verify that (6) is an asymptotically correct estimate, consider the Taylor expansion of the correct solution of (1) at time $t_n + h_n$,

$$y(t_n + h_n) = y(t_n) + h_n f(y(t_n)) + \frac{h_n^2}{2}\left(f'(y(t_n)) f(y(t_n))\right) + \ldots$$

The local truncation error per step for explicit Euler may then be expressed as

$$\begin{aligned} e_{n+1} = y_{n+1} - y(t_n + h_n) &\approx -\frac{h_n^2}{2} f'(y(t_n)) f(y(t_n)) \\ &\approx -\frac{h_n^2}{2} f'(y_n) f(y_n) \end{aligned}$$

which proves the correctness of (6). Also observe that

$$\phi_n \approx -\frac{f'(y_n)f(y_n)}{2}.$$

$\square$

Note that in the example ϕ depends on the elementary differential of order 1 of the function f. Typically, ϕ will contain the elementary differentials of order $p_e - 1$ for an error estimator of order p_e.

The error estimate is used for stepsize control, in order to produce a solution within the specified accuracy bound. Either $\hat{e}_n$ or $\hat{d}_n$ may be used. The former will be referred to as *error per step* (EPS), and the latter as *error per unit step* (EPUS). The motive to use EPUS is to get the same "accumulated" global error for a fixed integration time regardless of the number of steps used.

The output from the process is a scalar error estimate r, which is to be compared with the user-specified *tol*. Since $\hat{e}_n$ and $\hat{d}_n$ are vectors some norm has to be used to produce r. For robustness, a mixed absolute-relative norm is often used. A typical example is

$$r = \|\hat{e}\| = \max_i \left| \frac{\hat{e}_i}{|y_i| + \eta_i} \right| \tag{7}$$

where η_i is a scaling factor for the i:th component of y, creating the mixed absolute-relative error measure.

Combining (3) and the norm (7) yields the approximate process description

$$r = \|\phi\| h^k$$
$$k = \begin{cases} p_e, & \text{EPS} \\ p_e - 1, & \text{EPUS} \end{cases}$$

The Stepsize Controller

The user specifies the desired accuracy of the solution by giving an upper bound *tol* for the local error. The local error is approximated by r. This strategy is motivated by the fact that the global error can be bounded in terms of the local truncation error per unit step d.

To take steps as long as possible without violating *tol*, the stepsize should be chosen to fulfill

$$r = tol \tag{8}$$

If ϕ varies slowly, it can be estimated from the last step, and then used to calculate the next stepsize to fulfill (8). Hence

$$\begin{cases} \|\phi\| = \dfrac{r_{n+1}}{h_n^k} \\ tol = \|\phi\| h_{n+1}^k \end{cases} \Rightarrow \quad h_{n+1} = \left(\frac{tol}{r_{n+1}}\right)^{1/k} h_n \tag{9}$$

If the error is too large in one step (i.e. $r > \rho \cdot tol$), the step is rejected, and a new try is made with a recalculated stepsize. A common choice is $\rho = 1.2$.

To reduce the risk of rejection a safety factor γ is introduced in (9) resulting in

$$\begin{aligned} h_{n+1} &= \theta h_n \\ \theta &= \gamma \left(\frac{tol}{r_n}\right)^{1/k}, \end{aligned} \tag{10}$$

where γ is chosen less or equal to 1. A typical value is 0.9.

In some integration methods, changing stepsize may be an expensive operation requiring additional matrix factorizations. Therefore small stepsize changes are sometimes prevented by introducing a dead-zone. If θ is close to one, no stepsize change is done.

Finally, the amount the stepsize is allowed to increase in one step is limited. This is due to the fact that some integration methods (e.g. multistep methods) may go unstable if the stepsize is increased too rapidly.

Hence (10) is modified with

$$\theta := \begin{cases} 1, & \text{if } \theta_{lo} \le \theta \le \theta_{hi} \\ \theta_{max}, & \text{if } \theta > \theta_{max} \\ \theta, & \text{otherwise} \end{cases} \tag{11}$$

where ':=' means assignment. Typical values of the parameters are: $\theta_{lo} = 1.0$, $\theta_{hi} = 1.2$, and $\theta_{max} = 2.0$.

The stepsize controller condensed into (10) and (11) is found in most codes written today. It is also the one presented in modern textbooks [Hairer et al. 1987, Gear 1971].

3. Modeling the Process

To be able to analyze the control system, a model of the process is needed. Preferably, it should be simple, yet capture the main properties of the process.

A Model of the Process

In the previous section, asymptotic properties of the process gave rise to the approximate model

$$r_{n+1} = \|\phi_n\| h_n^k$$

By regarding $\log h$ as process input and $\log r$ as process output the model is turned into an affine relation. Using the forward shift operator q, the model reads

$$\log r_n = \frac{1}{q}\left(k \log h_n + \log \|\phi_n\|\right) \tag{12}$$

The process is just a constant gain k, depending on the order of the error estimator in the integration method, and a disturbance $\log \|\phi_n\|$ depending on the properties of the problem and its solution. The delay q^{-1} in the model is a consequence of the indexing conventions, i.e. the stepsize h_n is used to advance y_n to y_{n+1} giving r_{n+1} as output.

How good is the model? To answer the question consider the initial value problem

$$\begin{aligned} \dot{y} &= \lambda y \qquad t \geq 0 \quad \lambda \leq 0 \\ y(0) &= y_0 \end{aligned} \tag{13}$$

For such a problem an explicit Runge-Kutta method can be expressed as

$$\begin{aligned} y_{n+1} &= P(h_n\lambda) y_n \\ \hat{e}_{n+1} &= E(h_n\lambda) y_n \\ r_{n+1} &= \|\hat{d}_{n+1}\| \qquad \text{(EPUS)} \end{aligned} \tag{14}$$

where $P(h_n\lambda)$ and $E(h_n\lambda)$ are polynomials in $h_n\lambda$. The nonlinear difference equations (14) constitute the true process for the simple problem (13). The stability of the process is governed by $P(h_n\lambda)$, and the process is stable as long as $h\lambda$ (fixed h) is inside the stability region $\mathcal{S} = \{h\lambda : |P(h\lambda)| \leq 1\}$ of the Runge-Kutta method.

When $h_n\lambda$ is small and well inside $\mathcal{S}$, the process is well described by the static model (12) resulting from a linearization of (14) around $h = 0$. To verify, observe that $E(h_n\lambda)$ takes the form

$$E(h_n\lambda) = \kappa_0(h_n\lambda)^{p_e} + \kappa_1(h_n\lambda)^{p_e+1} + \dots$$

Then it holds

$$\log r_n = \frac{1}{q}\left(k \log h_n + \log \|y_n \lambda^{p_e}(\kappa_0 + \kappa_1 h_n \lambda + \dots)\|\right) \tag{15}$$

which is identical to the process model assumed earlier. The linearization even results in an explicit expression for ϕ. Note that ϕ is varying along the solution y_n and also (weakly) dependent on h_n. In addition, compare with Example 1 in Section 2, where (5) is the true process and (6) is the linearized model.

For the linear problem (13), $\phi \to 0$ as $t \to \infty$, and to keep r equal to the tolerance the stepsize controller will increase the stepsize. For a sufficiently large stepsize h_n, $h_n\lambda$ will leave the stability region of the method. Such a stepsize makes the nonlinear difference equation (14) unstable, and the stepsize is said to be limited by numerical stability. The behavior of (14) changes when $h_n\lambda$ approaches $\partial\mathcal{S} = \{h\lambda : |P(h\lambda)| = 1\}$, the stability boundary of the method, and the linearized model (12) no longer holds. A new process model can be derived for this case. The derivation, which is partly similar in spirit to [Higham and Hall 1987], is somewhat technical and has been relegated to Appendix A. For the problem (13) the resulting process model is

$$\log r_n = \frac{(C_1 - 1)q + C_2 - C_1 + 1}{q(q-1)}(\log h_n - \log h_s) - \log h_s, \tag{16}$$

where h_s is such that $h_s\lambda$ is placed on $\partial\mathcal{S}$. The coefficients C_1 and C_2 are defined by

$$C_1(h_s\lambda) = h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}, \qquad C_2(h_s\lambda) = h_s\lambda\frac{P'(h_s\lambda)}{P(h_s\lambda)}. \tag{17}$$

The results (12), (16) and (17) can be generalized to wider classes of problems than (13). A similar derivation can also be done for the case EPS. In the latter case (12) remains unchanged while (16) has to be slightly modified. Both the generalizations and the EPS case are treated in Appendix A.

Identification of the Process Model

To verify the process models derived above, an identification of the transfer function was performed when integrating the nonlinear problem 4 in Appendix D. The problem was solved for different tolerances using DOPRI45, an order 4/5 explicit Runge-Kutta method [Hairer et al. 1987]. DOPRI45 is optimized to be used with local extrapolation, i.e. as a fifth order method, although this may asymptotically overestimate the error. EPUS was used for stepsize control.

For DOPRI45 using local extrapolation the polynomials $P(z)$ and $E(z)$ are

$$\begin{aligned} P(z) &= 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \frac{z^6}{600} \\ E(z) &= -\frac{97z^5}{120000} + \frac{13z^6}{40000} - \frac{z^7}{24000}. \end{aligned} \tag{18}$$

and the stability region $\mathcal{S}$ shown in Figure 4 can be evaluated from $P(z)$.

The problem was solved for a number of different tolerances. For each tolerance a transfer function between $\log h_n$ and $\log r_n$ was identified using the system identification toolbox in PRO-MATLAB [Moler et al. 1987]. The full identification procedure as well as the nonlinear problem is described in Appendix B.

Let λ_{max} be the "dominating*" most negative real eigenvalue of the Jacobian of the nonlinear differential equation. For $h_n\lambda_{max} \approx -0.5$, i.e. $h_n\lambda_{max}$ well inside $\mathcal{S}$, the following model was arrived at

$$\log r_n = \frac{4.25}{q} \log h_n. \tag{19}$$

Since the error estimator in DOPRI45 is of fifth order and EPUS was used one would expect the value 4.0 instead of 4.25. The discrepancy is

* By dominating we mean the eigenvalue to first reach $\partial\mathcal{S}$ when h is increased.

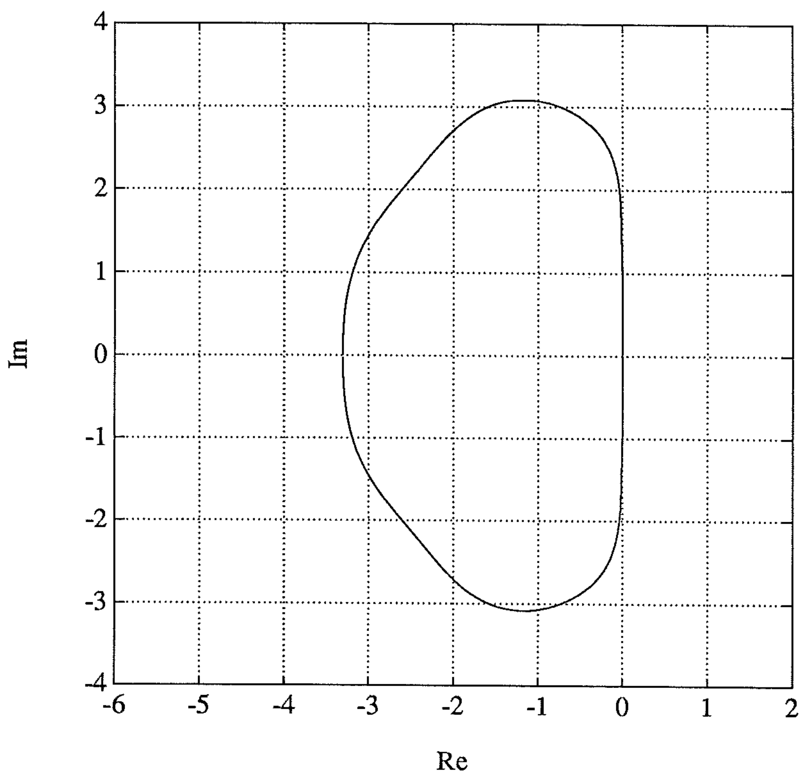

Figure 4. Stability region for DOPRI45 with local extrapolation.

explained by the fact that $h\lambda$ differs significantly from zero. As a result, one does not observe the asymptotic behavior but a slightly modified behavior. It is possible to analytically estimate this modified behavior. By assuming that the behavior of the nonlinear equation is completely governed by λ_{max}, one gets

$$\hat{e}_{n+1} \approx E(h_n\lambda_{max})y_n, \qquad r_{n+1} = \|\hat{d}_{n+1}\| \approx \frac{|E(h_n\lambda_{max})|\,\|y_n\|}{h_n}$$

Now observe that

$$\begin{aligned} \frac{\partial \log r_{n+1}}{\partial \log h_n} &= h_n \frac{\partial \log r_{n+1}}{\partial h_n} \approx h_n \frac{\partial}{\partial h_n} \log\left(\frac{|E(h_n\lambda_{max})|\,\|y_n\|}{h_n}\right) \\ &= h_n\lambda_{max}\frac{E'(h_n\lambda_{max})}{E(h_n\lambda_{max})} - 1 = C_1(h_n\lambda_{max}) - 1 \end{aligned} \tag{20}$$

For $h_n\lambda_{max} = -0.5$ the formula (20) evaluates to 4.19, which should be compared with the value 4.25 from (19). As $h_n\lambda_{max}$ increases higher order terms in $E(z)$ will play a larger roll. Hence (20) predicts the gain 4.51 for $h_n\lambda_{max} \approx -1.6$, while the identification gives 4.60.

As $h_n\lambda_{max}$ is further increased it will approach $\partial\mathcal{S}$ where a model in the form (16) is expected. Although not verified by theoretical derivations the identification indicates a gradual change from (12) to (16). As an example consider $h_n\lambda_{max} \approx -2.4$. For this value the identification resulted in

$$\log r_n = \frac{4.87q - 0.15}{q(q-0.24)} \log h_n$$

For DOPRI45 the negative real axis intersects $\partial\mathcal{S}$ at -3.31. At this point $C_1 = 5.85$ and $C_2 = 6.07$, and according to (16) one would expect the model

$$\log r_n = \frac{4.85q + 1.22}{q(q-1)} \log h_n$$

This is in almost perfect agreement with the identified model, which was

$$\log r_n = \frac{4.85q + 1.23}{q(q-1)} \log h_n$$

for $h_n\lambda_{max} \approx -3.3$.

4. The Closed Loop Using the Standard Controller

The Standard Controller

By regarding $\log h$ as control variable, the standard stepsize controller (10) can be expressed as a pure integrating controller [Gustafsson et al. 1988]. To illustrate this, $\log h_{n+1}$ is expressed as a function of $\log h_n$ using (10). Then

$$\log h_{n+1} = \log h_n + \frac{1}{k}\left(\log(\gamma^k \cdot tol) - \log(r_{n+1})\right),$$

and using the forward shift operator q

$$\log h_n = \frac{1}{k}\frac{q}{q-1}\left(\log(\gamma^k \cdot tol) - \log(r_n)\right). \qquad (21)$$

The set point of the controller is $\log(\gamma^k \cdot tol)$, and $\log h_n$ is the controller state. Note that the safety factor γ, which was introduced to reduce the risk of step rejection, is equivalent to decreasing the set point from $\log(tol)$ to $\log(\gamma^k \cdot tol)$.

A small control error results in small stepsize changes that are prevented by the dead-zone (11). This is one of the two cases where the control signal is chosen differently from the value given by (10). The other case occur when the stepsize increase is too large in one step. In both cases the controller state needs to be updated to reflect the discrepancy. The way this is handled is equivalent to a deadbeat antiwindup (for a description of antiwindup see e.g. [Åström and Wittenmark 1984, Franklin et al. 1986]).

To facilitate the analysis of the dynamics of the closed loop system when using the standard controller (10), the dead-zone and the limitation on control signal rate are first neglected. In addition γ is set to 1, and step rejections are prohibited (i.e. choose ρ large). Normally the *integration gain* is chosen as $1/k$ but here it will be kept as a free parameter k_I to investigate its influence on the closed loop system. With these modifications the controller (21) can be expressed as

$$\begin{aligned}\log h_n &= G_{c_I}(q)\,(\log tol - \log r_n)\\ G_{c_I}(q) &= \frac{k_I q}{q-1}\end{aligned} \tag{22}$$

Asymptotically Small Stepsizes

For asymptotically small stepsizes the process is well approximated by a constant gain k, defined by the asymptotic behavior of the error estimator, and a disturbance $\log \|\phi\|$. The same model structure is valid for both EPS and EPUS. Hence

$$\begin{aligned}\log r_n &= G_p(q)\log h_n + \log \|\phi\|\\ G_p(q) &= \frac{k}{q},\end{aligned}$$

where the delay in $G_p(q)$ is a consequence of the indexing conventions.

When looking at the process model the integrating controller seems reasonable. The disturbance, which is regarded as constant or slowly varying, is compensated by the integrator in the controller.

The closed loop system (see Figure 5) may be written

$$\log r_n = G_{tol}(q)\log tol + G_{\phi}(q)\log \|\phi_n\|$$

where

$$\begin{aligned} G_{tol}(q) &= \frac{G_{c_I}(q)G_p(q)}{1+G_{c_I}(q)G_p(q)} = \frac{kk_I}{q-1+kk_I} \\ G_{\phi}(q) &= \frac{1}{q(1+G_{c_I}(q)G_p(q))} = \frac{q-1}{q(q-1+kk_I)} \end{aligned} \tag{23}$$

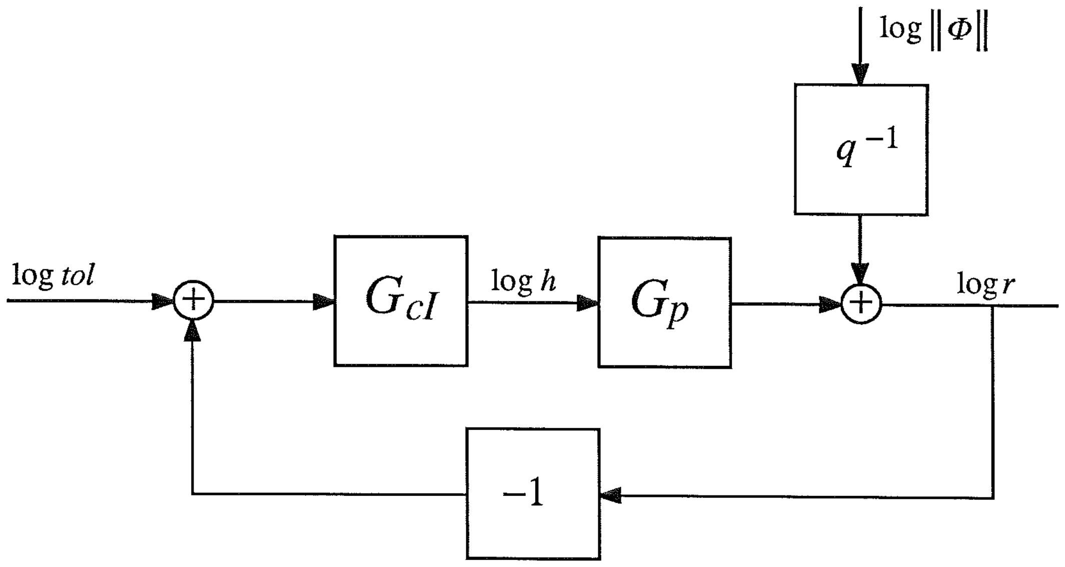

Figure 5. Closed loop system

For all choices of k_I yielding a stable control system (i.e. avoiding a pole outside the unit disc, $0 < k_I k < 2$) and a constant disturbance $\log \|\phi\|$, the error r will eventually approach tol, since $G_{tol}(1) = 1$ and $G_{\phi}(1) = 0$. The constant disturbance is removed by the "differentiation" $q-1$ in the numerator of $G_{\phi}(q)$. How fast r reaches tol is governed by the pole location. By changing k_I it can be placed anywhere on the real axis. Choosing k_I as $1/k$, as usually in the standard controller, places the pole at the origin. This choice makes the response of the system as fast as possible and is referred to as deadbeat control (a presentation of deadbeat control is given in e.g. [Åström and Wittenmark 1984]).

Whether it is desirable to make the system deadbeat depends on the properties of the disturbance $\log\|\phi\|$. If it is mainly constant or slowly varying it might be a good idea. The result is fast disturbance rejection. If, on the other hand, $\log\|\phi\|$ contains higher frequency components, they will be amplified by the "differentiation" in $G_\phi(q)$. It would then be wise to include some filtering in $G_\phi(q)$ by placing the pole on the real axis somewhere in the interval [0,1]. Moving the pole closer to 1 increases the filtering. The pole placement is a tradeoff between response time and sensitivity to high-frequency components in $\log\|\phi\|$.

Another way to look at the problem is to regard the controller as an attempt to predict $\log\|\phi\|$, and to remove the disturbance accordingly. For each $\log r_n$, $\log\|\phi_{n-1}\|$ can be calculated using $\log h_{n-1}$ (see formula 12). The value is then used as a prediction of $\log\|\phi_n\|$, making it possible to calculate h_n. Placing the pole at the origin means basing the prediction on the last error and stepsize only. Moving the pole from the origin to $\xi = 1 - kk_I$, a position between 0 and 1 on the real axis, is equivalent to forming a prediction based on an exponential average of old $\log\|\phi\|$:s. To illustrate this, r_n is eliminated from (22) and (23):

$$\begin{aligned}\log h_n &= \frac{k_I}{q-1+kk_I}\left(\log tol - \log\|\phi_n\|\right)\\ &= \frac{1}{k}\frac{1-\xi}{q-\xi}\left(\log tol - \log\|\phi_n\|\right).\end{aligned}$$

Since *tol* is constant, we have, in explicit form

$$\log h_n = \frac{1}{k}\left(\log tol - \log\|\hat{\phi}_n\|\right),$$

where $\log\|\phi_n\|$ is estimated by

$$\log\|\hat{\phi}_n\| = \begin{cases} \dfrac{1-\xi}{\xi}\displaystyle\sum_{i=1}^{\infty}\xi^i \log\|\phi_{n-i}\|, & \xi \neq 0\\[2ex] \log\|\phi_{n-1}\|, & \xi = 0.\end{cases}$$

It is important to understand that the choice of k_I is a design choice affecting the properties of the system. In numerical analysis literature k_I is normally considered as determined by the order of the error estimator in the integration method used, but from a control theoretic point of view the choice $k_I = 1/k$, is by no means necessary.

Stepsize Limited by Stability

As shown in Section 3 the process changes character when numerical stability ($|P(h\lambda)| \leq 1$) limits the stepsize. The process model depends on the integration method, the problem being solved and if EPS or EPUS is used (see Appendix A). From now on only the EPUS case will be considered. The derivations and results for EPS are completely analogous.

Using the process model for EPUS (16)

$$\log r_n = G_p(q)(\log h_n - \log h_s) - \log h_s$$
$$G_p(q) = \frac{(C_1 - 1)q + C_2 - C_1 + 1}{q(q-1)}$$

and the standard stepsize controller (22) results in the closed loop system

$$\begin{aligned}
\log r_n &= G_{tol}(q) \log tol + G_{h_s}(q) \log h_s \\
G_{tol}(q) &= \frac{G_{c_I}(q) G_p(q)}{1 + G_{c_I}(q) G_p(q)} \\
&= \frac{k_I \left((C_1 - 1)q + 1 - C_1 + C_2\right)}{q^2 + \left(-2 + k_I(C_1 - 1)\right) q + 1 + k_I(1 - C_1 + C_2)} \qquad (24) \\
G_{h_s}(q) &= -\frac{1 + G_p(q)}{1 + G_{c_I}(q) G_p(q)} \\
&= -\frac{(q-1)(q^2 + (C_1 - 2)q + 1 - C_1 + C_2)}{q\left(q^2 + \left(-2 + k_I(C_1 - 1)\right) q + 1 + k_I(1 - C_1 + C_2)\right)}
\end{aligned}$$

Here $G_{tol}(1) = 1$ and $G_{h_s}(1) = 0$. Therefore $\log h_s$, which is constant (or slowly varying), will be removed and eventually $\log r$ will equal $\log tol$, provided the closed loop control system is stable.

The transient behavior as well as the stability of the control system is governed by the transfer function poles, i.e. the roots of

$$q^2 + \left(-2 + k_I(C_1 - 1)\right) q + 1 + k_I(1 - C_1 + C_2) = 0 \qquad (25)$$

The system is stable if these roots are inside the unit circle. In [Hall 1985, 1986] another stability test is derived. It consists of checking the eigenvalues of a 2 by 2 matrix. In the special case, $k_I = 1/k$ the characteristic equation of that matrix equals the polynomial in (25).

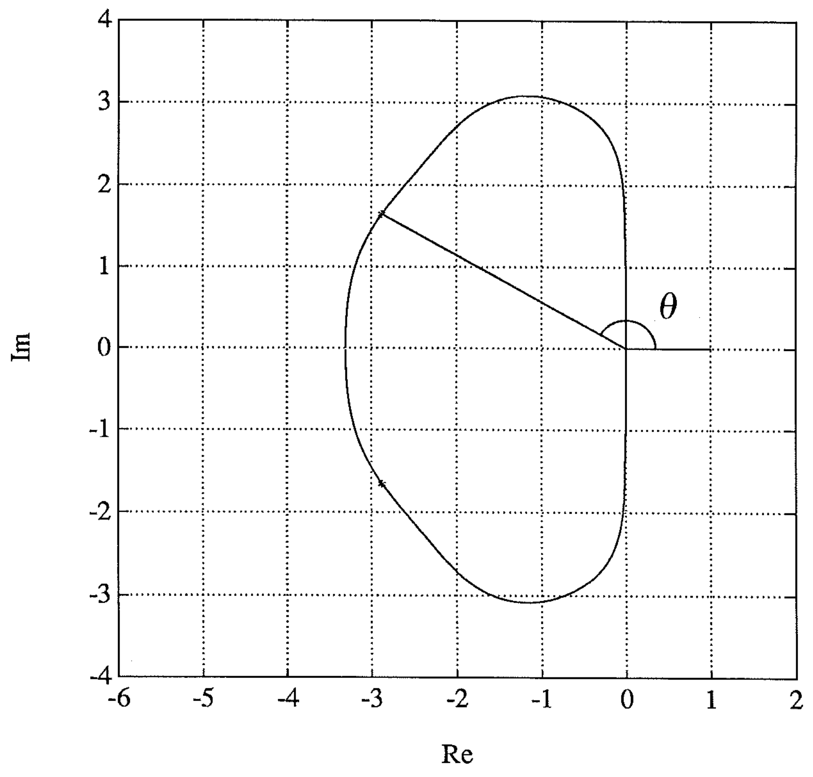

Figure 6. Definition of θ.

It is important to note that (25) having a root outside the unit disc only implies local instability. The model (16) is only valid when $h \approx h_s$, and the behavior of the process changes as soon as the instability causes the stepsize to move away from the vicinity of h_s.

For the standard choice $k_I = 1/k$ the roots of (25) are often to be found outside the unit disc, resulting in a locally unstable closed loop system. Take for instance the following example:

EXAMPLE 2—DOPRI45 with local extrapolation, EPUS
For a second order problem with complex eigenvalues, C_1 and C_2 are expressed (see Appendix A) as

$$C_1 = \mathrm{Re}\left(h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}\right), \qquad C_2 = \mathrm{Re}\left(h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}\right)$$

Expressions for $P(z)$ and $E(z)$ for DOPRI45 were given in (18). Using these expressions, the coefficients C_1 and C_2 can be calculated for all $h_s\lambda$ on ∂S. Assume the standard controller (22) with $k_I = 1/k = 1/4$ is being

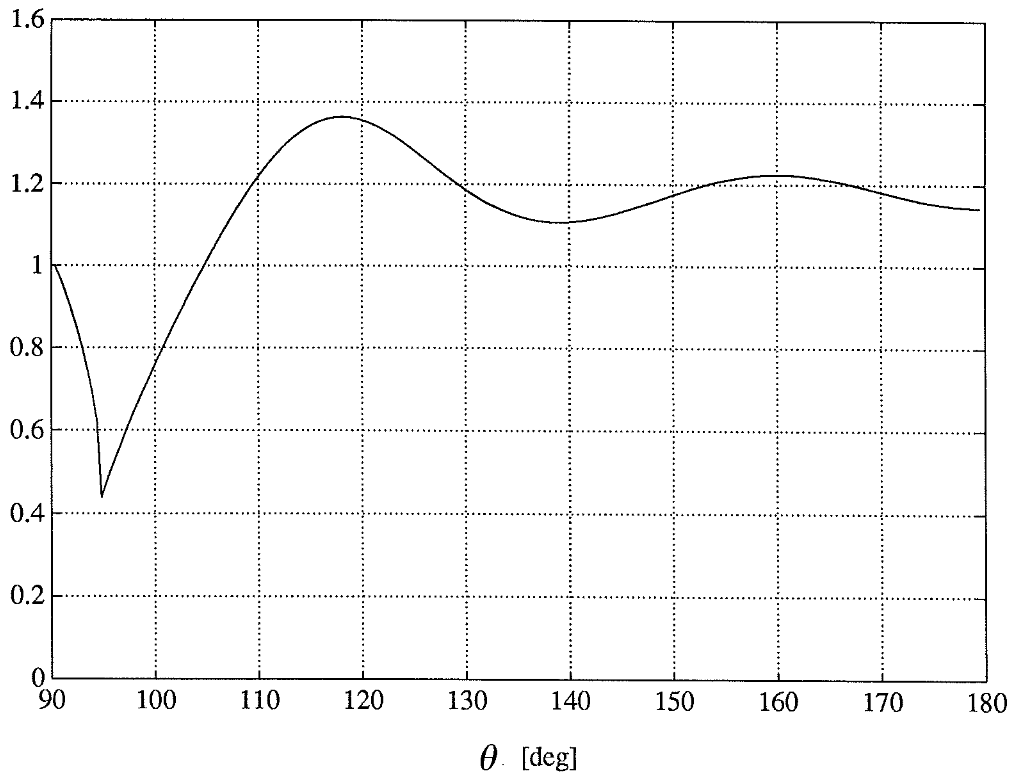

Figure 7. Magnitude of the largest closed loop pole of the stepsize control loop, as a function of $\theta = \arg(h_s\lambda)$.

used. For $h_s\lambda$ on $\partial\mathcal{S}$, let $\theta = \arg(h_s\lambda)$ (see Figure 6). For each value of θ, ($\theta \in [\pi/2, \pi]$), the magnitude of the largest closed loop pole is calculated. The result is plotted in Figure 7. The closed loop control system is locally unstable for almost all values of θ.

□

Explaining the Anomalies of the Standard Controller

From the results above it is quite easy to explain the oscillating stepsize sequence often encountered when solving a stiff differential equation using an integration method with bounded stability region.

As described above, the process (the nonlinear difference equation created by the differential equation, the integration method and the error estimator) changes character, when stability limits the stepsize. The standard stepsize controller is not designed to handle this case, and the result is a locally unstable closed loop system. The instability causes the error to grow and the stepsize controller will reduce the stepsize to keep the error below *tol*. The reduction of stepsize causes the behavior of the process to change ($h_n\lambda$ moves inside the stability region), making the control system regain stability. The cycle repeats itself creating a highly irregular stepsize sequence.

In the standard controller sometimes a dead-zone on stepsize changes is included. If the controller asks for a change that is too small it is prevented by the dead-zone. Although introduced to improve efficiency by preventing "unnecessary" stepsize changes, the dead-zone sometimes has a positive effect also on the behavior of the closed loop stability. If r is below, but sufficiently close to *tol*, the control error is small and due to the dead-zone there will not be any stepsize changes. Hence h will not be increased such that $h\lambda$ ends up on ∂S, and the system stays stable.

The other misbehavior reported is long sequences of rejected steps. This can be observed for problems showing drastic changes in behavior. During such changes $||\phi||$ varies considerably, sometimes by many orders of magnitude causing $\log ||\phi||$ to vary by several hundred percent. The standard controller is designed with the assumption that $\log ||\phi||$ is slowly varying, and it does not have any abilities to track fast changes in $\log ||\phi||$. For a strongly decreasing $\log ||\phi||$, $\log r$ will decrease since the controller does not succeed in changing $\log h$ fast enough to track the changing disturbance. The situation is similar for $\log ||\phi||$ increasing, and the controller fails to decrease $\log h$ fast enough to keep $\log r$ below $\log tol$. The result is too large errors, causing rejected steps.

5. A New Controller

Design Alternatives

The stepsize control problems may be attacked in different ways. The properties of the closed loop system depend on the controller as well as on the process. Thus one may change either one to improve the behavior of the system.

Hall and Higham [1987] approach the problem by changing the process, i.e. the integration algorithm. When constructing an explicit Runge-Kutta method there is some freedom in the choice of parameters. Normally this freedom is used to minimize error coefficients or to maximize the stability region of the method, but Higham and Hall exploit it to change C_1 and C_2 such that the closed loop system is stable when the standard controller is used.

The other way to approach the problem is to change the controller. It is our opinion that the freedom in choice of parameters in the method should be used to improve its numerical properties, while the stepsize control problem should be attacked by improving the stepsize controller.

Design Goals

A good stepsize controller should have the following properties

1. Steady state gain from $\log tol$ to $\log r$ equal to one, i.e. use $\log tol$ as set point.
2. Good disturbance rejection for the process model (12), i.e. the influence of $\log \|\phi\|$ on $\log r$ in the closed loop system should be small.
3. The process model (16) should be stabilized for all relevant values of C_1 and C_2.
4. In the identification experiment some intermediate process models were found when the process changed character from (12) to (16) (see Appendix B). These processes should be stabilized as well.
5. Computationally simple to minimize overhead.

A PI Controller

The above mentioned goals cannot be achieved by merely changing k_I in the standard controller. This is easily seen from the following example.

EXAMPLE 3—DOPRI45 with local extrapolation, standard controller
Apply DOPRI45 (EPUS) to a first order linear autonomous problem, with a negative eigenvalue. At stationarity, stability limits the stepsize and the model (16) with $C_1 = 5.85$ and $C_2 = 6.07$ holds. The denominator in (24) evaluates to

$$q^2 + (-2 + 4.85 \cdot k_I)q + 1 + 1.23 \cdot k_I.$$

The roots of this polynomial are outside the unit circle for all $k_I > 0$, and hence the closed loop system is locally unstable for any choice $k_I > 0$. □

As was indicated in Section 1, a simple but effective modification of the standard controller is to add a proportional term in the control law (22), i.e.

$$G_{c_{PI}}(q) = k_I \frac{q}{q-1} + k_P = \frac{(k_I + k_P)q - k_P}{q-1}. \tag{26}$$

This change is equivalent to introducing a zero in the transfer function of the controller. In the standard controller the stepsize is constructed from the "integral" of old control errors, while the modified one also uses a term directly proportional to the current control error. It is a general fact that a PI-controller (proportional, integral) exhibits better stabilizing properties than a pure I-controller.

It is hard to give general formulae for how to choose the controller parameters k_I and k_P. Their values are a compromise between stability and response time. Since C_1 and C_2 vary for different integration methods one cannot expect to find values that will be acceptable for all integration methods. Here we will confine ourselves to DOPRI45 and EPUS. For this combination a good choice is

$$k_I = 0.06, \qquad k_P = 0.13.$$

The derivation of these parameter values is described in Appendix C.

Explicit Formulation of the PI-controller

The PI-controller may be rewritten on a form resembling the original standard stepsize controller (10) [Gustafsson et al. 1988]. Some manipulations applied to

$$\log h_n = G_{c_{PI}}(q)(\log tol - \log r_n)$$

using (26) yield

$$h_n = \left(\frac{tol}{r_n}\right)^{k_I} \left(\frac{r_{n-1}}{r_n}\right)^{k_P} h_{n-1} \tag{27}$$

From this expression it is clear that the proportional part corresponds to taking the most recent development of r into account when deciding upon the next stepsize. It is also clear that this type of controller is trivial to implement in existing ODE-codes.

Limitations and Dead-zones

Some of the limitations present in the standard controller should be included in the new controller as well. The error r is just an estimate of the true error and r may occasionally (and erroneously) be very small in one step. It is therefore sound to include a limit on the rate of stepsize increase. Moreover, multistep methods have a tendency to become unstable if the stepsize is increased too rapidly. Therefore a rate limit has to be present in that case.

The standard stepsize controller includes a dead-zone on stepsize changes. A stepsize change may invoke extra computations (e.g. refactorizations of the Jacobian in multistep methods), and for efficiency reasons small "unnecessary" changes may be prevented by using a dead-zone.

There is no reason to use the dead-zone in a one-step methods, since a stepsize change does not invoke any extra computations. With the new controller it may also be useful to remove it for multistep methods. The new controller generally produces smoother stepsize sequences (see Section 7) than the standard controller, and with a smooth and regular stepsize sequence there is little or no need to prevent small stepsize changes.

Rejected Steps

When a step is rejected due to too large an error, the stepsize controller should not be used in the usual way. The next step to be taken is a retry, and from the last attempt it is known what to expect ahead. This information should be used to get a good restart of the controller.

The most likely reason for the rejected step is a major increase in the disturbance $\log \|\phi\|$. For the old standard controller an alternative reason would have been an irregular error sequence caused by an oscillating stepsize sequence. With the new controller this is less likely since the controller stabilizes all the different process models encountered.

Suppose r_{n+1} was too large, i.e. h_n could not be used to update y_n to y_{n+1}. Instead a new stepsize h_n^* has to be found such that the resulting r_{n+1}^* is acceptable. A simple way to do this is to use the algorithm of the standard controller: the stepsize and the error from the rejected step is used to calculate $\log \|\phi_n\|$, and then a new stepsize such that $\log r_{n+1}^*$

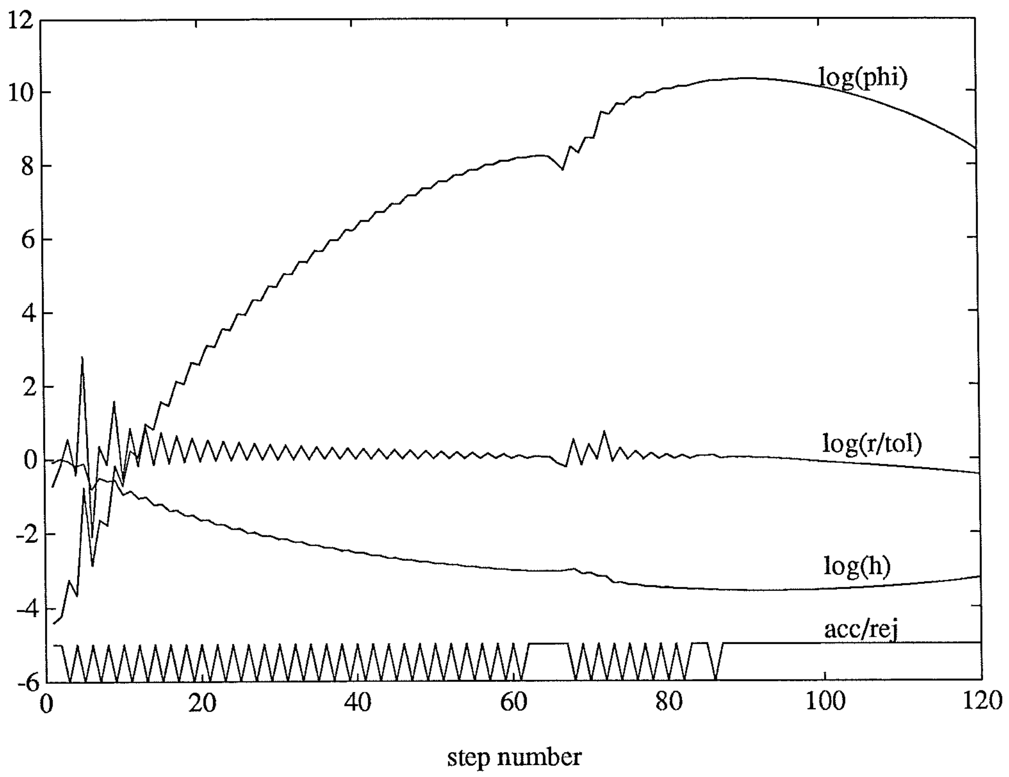

Figure 8. Some curves originating from the Brusselator (problem 5 in Appendix D, *tol* $= 10^{-4}$). Note the long sequence of alternating accepted and rejected steps caused by the increasing $\log\|\phi\|$. The variables $\log\|\phi\|$, $\log r$ and $\log h$ are plotted using base 10 logarithm. In the curve acc/rej an accepted step is represented by the value -5 and an rejected step by the value -6.

equals $\log tol$ is calculated using $\log\|\phi_n\|$ and (12). Hence

$$h_n^* = \left(\frac{tol}{r_{n+1}}\right)^{1/k} h_n \tag{28}$$

One objection to (28) is that the disturbance $\log\|\phi_n^*\|$ at the step h_n^* will differ from $\log\|\phi_n\|$ since $h_n^* \neq h_n$. However, the previous step was rejected due to an increase in $\log\|\phi\|$ and $\log\|\phi_n^*\|$ will therefore generally be smaller than $\log\|\phi_n\|$ due to $h_n^* < h_n$. Hence the stepsize h_n^* given by (28) is likely to be a little bit smaller than needed, which actually increases the chance that h_n^* will result in an accepted step.

After a rejected step it does not suffice to determine a new stepsize only. Also the internal state in the PI-controller has to be updated. A first approach would be to use a deadbeat antiwindup strategy, i.e. give the

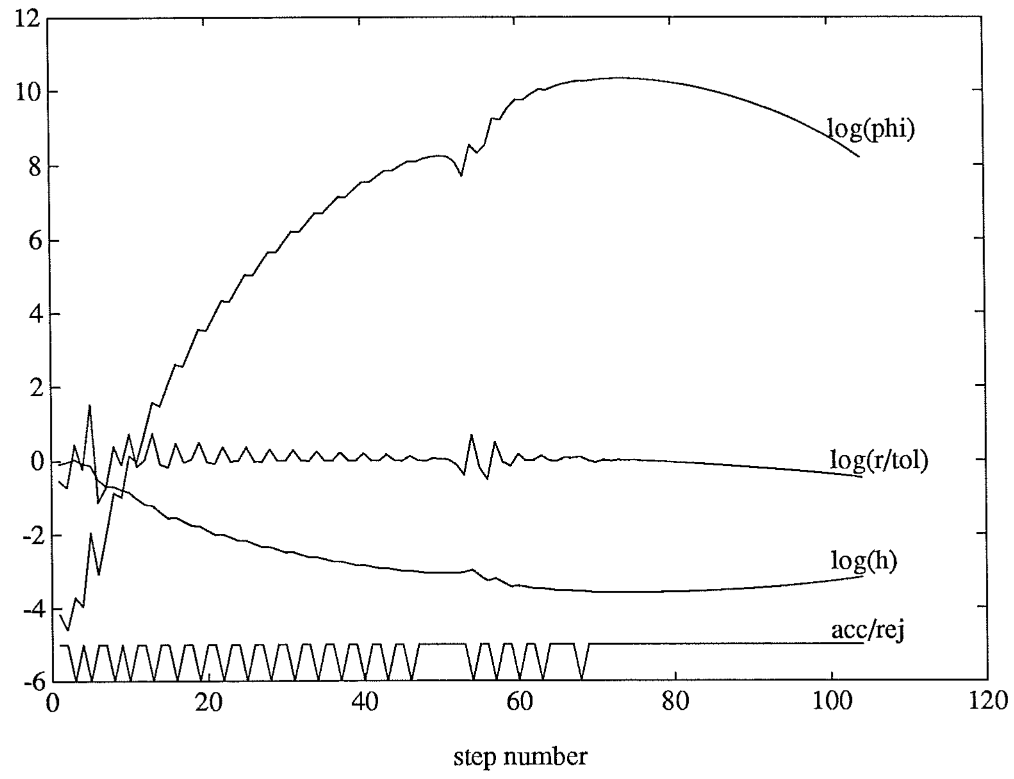

Figure 9. Some curves originating from the Brusselator (problem 5 in Appendix D, $tol = 10^{-4}$). A special update of the controller state is used after each rejected step, resulting in a drastic decrease in the number of rejected steps. The variables $\log \|\phi\|$, $\log r$ and $\log h$ are plotted using base 10 logarithm. In the curve acc/rej an accepted step is represented by the value -5 and an rejected step by the value -6.

state such a value that for a zero input (last step perfect, $\log r = \log tol$) the controller would have produced h_n^* as output. This is however not a good approach. The disturbance $\log \|\phi\|$ exhibits a lot of structure. Since $\log \|\phi\|$ increased in the last step it is very likely that it will increase in the next step too. Then h_n^* will be accepted but the very next step will be rejected due to the additional increase in $\log \|\phi\|$. This effect can be seen in Figure 8. Here the Brusselator (problem 5 in Appendix D) is solved with DOPRI45, EPUS and the PI-controller with $tol = 10^{-4}$. The same time sequence as in Figure 2 is shown, but this time the variables are plotted versus step number instead of time. The phenomenon of alternating accepted and rejected steps due to an increasing $\log \|\phi\|$ is very pronounced. Note that in the figure $\log \|\phi\|$, $\log r$ and $\log h$ are plotted with base 10 logarithm. Moreover, $\log \|\phi\|$ is calculated from $\log r$ and $\log tol$ using

(12). The sawtooth pattern in the log $||\phi||$ curve when log $||\phi||$ increases is not due to any irregularities in log $||\phi||$, but rather to the fact that both accepted and rejected steps are plotted. After a rejected step the next step will sens a similar (or smaller) value on log $||\phi||$, resulting in the sawtooth pattern in the figure.

The structure in log $||\phi||$ suggests that not only should h_n be decreased to h_n^* but a stepsize decrease in the following step as well would probably be wise. The state in the controller could be updated to achieve this end. Hence, at a rejected step:

1. Calculate h_n^* from (28) and use it as the next stepsize.
2. Update the controller state such that if the next step is perfect ($r_n^* = tol$) there will still be a stepsize decrease of the same factor as the one between h_n and h_n^*.

The strategy is equivalent to using h_n^* given by (28), but then when using (27) to calculate the next stepsize pretend that the last used stepsize was not h_n^* but instead h' with

$$h' = h^* \cdot \frac{h^*}{h}$$

Using the described strategy when solving the previously presented Brusselator example results in Figure 9. Compare Figure 9 with Figure 8, and note the reduction in rejected steps. Now it takes 104 steps to produce the solution compared to the 121 steps needed before. Observe that now each rejected step is normally followed by (at least) two accepted steps. The first accepted step is explained by (28) while the second is due to the special update of the controller state.

What if log $||\phi||$ does not behave as assumed? This will not cause any problems. If log $||\phi||$ increases much more than expected the special update of the controller state will not succeed in creating a second accepted step. Instead it will be rejected and (28) will be used to restart the controller again. On the other hand, if log $||\phi||$ does not increase the updated controller state will result in a too small stepsize. As a consequence the error will be small and in the next step the controller will increase the stepsize again.

Algorithmic Formulation of the PI-controller

The inclusion of the strategy concerning rejected steps makes the controller more complicated than (26). It is no longer possible to have h as both controller state and output. In addition, the controller uses different formulae after accepted and rejected steps. Therefore, to summarize this section, an outline of a code implementing the PI-controller is presented in Listing 1.

```
if current_step_accepeted then
    if previous_step_rejected then
        $x := h \cdot h/x$
    endif
    $x := \left(\frac{tol}{r}\right)^{k_I} \left(\frac{oldr}{r}\right)^{k_P} x$
    $h := x$
    $oldr := r$
else
    $h := \left(\frac{tol}{r}\right)^{1/k} h$
endif
```

Listing 1. An outline of the code needed to implement the PI-controller including the restart strategy after rejected steps.

The controller is called after each step in the integration routine, and calculates the stepsize to be used in the next step. The variable x is the controller state, and as before h is the stepsize, and r is the corresponding error estimate.

To make the algorithm complete the limitation on stepsize increase should be included. The algorithm should also include some safety net to prevent division by zero if $r = 0$.

6. A Controller with Prediction

In this paper a control theoretic viewpoint of stepsize control has been advocated. First the process was modeled. The model was constructed such that it captured all important properties of the process. The model brought understanding of the behavior of the process, and the gained understanding could then be used when designing a new controller.

A good process model facilitates the controller design. It gives a means to analyze and compare different controllers, making the final controller choice better founded.

To exemplify this a PI-controller that includes prediction of changes in $\log \|\phi\|$ will be considered.

Prediction of the Disturbance

The disturbance $\log \|\phi\|$ contains much structure. In the previous section this structure was exploited to decrease the number of rejected steps (see the discussion about rejected steps in Section 5). When $\log \|\phi\|$ changes the controller will sometimes have trouble increasing or decreasing the stepsize sufficiently fast. The result is either too large an error leading to rejected steps or an error that is much smaller than the tolerance, i.e. an unnecessarily accurate solution. In these situations the controller might function better if it includes a prediction of the change in $\log \|\phi\|$. A first order prediction using (12) makes the PI-controller read

$$\log h_n = (k_P + k_I \frac{q}{q-1})(\log tol - \log r_n) - \frac{q}{k(q-1)} \Delta \log \|\phi_n\| \quad (29)$$

where we use the same k as in (12). The change in $\log \|\phi\|$ is predicted by the term $\Delta \log \|\phi\|$, which is formed as an exponential average of old changes in $\log \|\phi\|$, i.e.

$$\Delta \log \|\phi_n\| = \xi \Delta \log \|\phi_{n-1}\| + (1-\xi)(\log \|\phi_{n-1}\| - \log \|\phi_{n-2}\|) \quad (30)$$

with $\xi \in [0,1]$. In (30) old values of $\log \|\phi\|$ are required. These can be calculated from old errors and old stepsizes using (12).

The controller (29) can be expressed in closed form by combining

(12), (29) and (30). This results in

$$\begin{aligned}
&\log h_n = G_{tol}(q)\log tol + G_r(q)\log r_n \\
&G_{tol}(q) = \frac{(q-\xi)\left((k_I+k_P)q - k_P\right)}{(q-1)^2} \\
&G_r(q) = \\
&\qquad \frac{(\xi - 1 - k(k_I+k_P))q^2 + (1 + kk_P - \xi + \xi k(k_I+k_P))q + \xi k k_P}{k(q-1)^2}.
\end{aligned} \tag{31}$$

The controller includes two poles at 1, i.e. a double integrator. This is a natural consequence of requiring the controller to handle a linearly changing $\log\|\phi\|$ instead of just a constant $\log\|\phi\|$.

The long sequences of alternating accepted and rejected steps seen in Figure 8 and Figure 9 might suggest a controller where the set point is varied in relation to the changes in $\log\|\phi\|$, i.e. when $\log\|\phi\|$ increases rapidly, the set point is decreased to reduce the risk of rejecting steps, and conversely during a decrease in $\log\|\phi\|$ the set point is increased. Such a controller is similar to the predicting controller (31). The only thing that differs is the way the prediction is included, which depends on how the set point was varied in relation to changes in $\log\|\phi\|$. The controller introduced by Zonneveld [1964] can also be viewed as a predicting controller. Zonneveld uses an extrapolation scheme on old stepsizes to calculate a new stepsize. The extrapolation scheme is equivalent to modifying the standard controller (22) with a kind of prediction of changes in $\log\|\phi\|$.

The controller (31) functions well for differential equations and tolerances such that (12) holds during the whole simulation. This can be seen in Figure 10, where the Brusselator (problem 5 in Appendix D) is solved from $t = 0$ to $t = 30$ with $tol = 10^{-4}$. In Figure 11 the same problem is solved with the ordinary PI-controller as a comparison. Note that when using the predicting controller an (almost) linear change in $\log\|\phi\|$ no longer poses any problems. Instead the predicting controller has some problems when the trend in $\log\|\phi\|$ changes direction. With the PI-controller a total of 426 steps are needed to solve the problem while the controller with prediction only needs 401. The reduction in number of steps is a result of both a reduction in the number of rejected steps and better control of $\log r$ ($\log r$ is kept closer to $\log tol$).

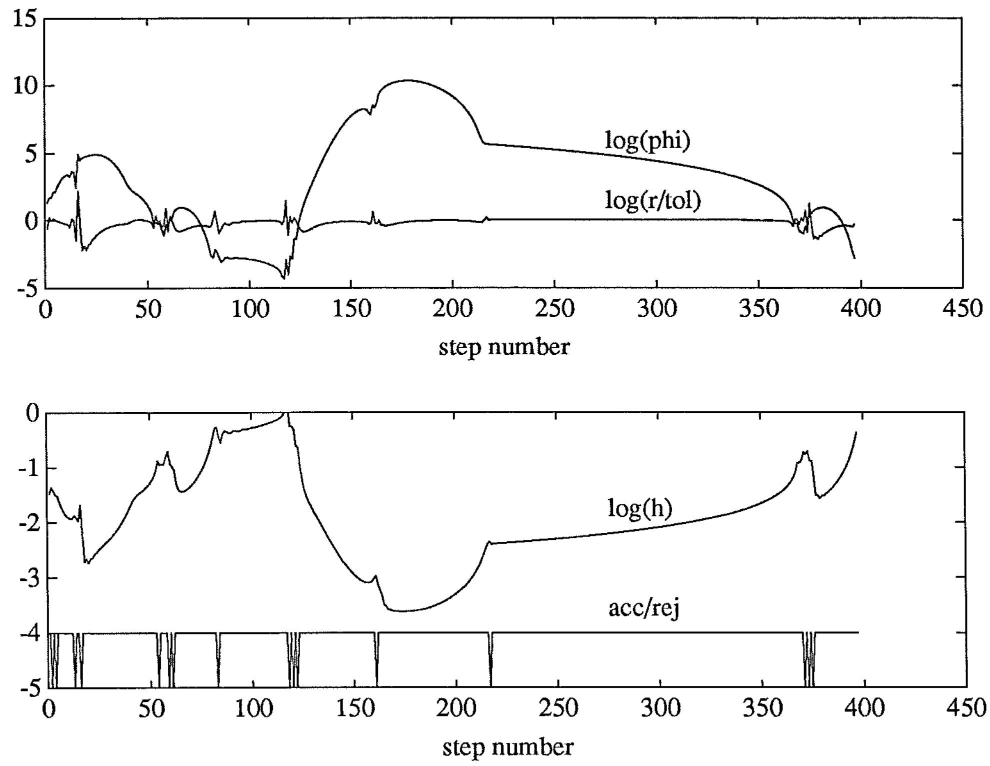

Figure 10. The Brusselator (problem 5 in Appendix D) simulated from $t = 0$ to $t = 30$ using a PI-controller with prediction for stepsize control. The parameter ξ in (31) was chosen as 0.5. All variables are plotted as functions of step number. The variables $\log \|\phi\|$, $\log r$ and $\log h$ are plotted using base 10 logarithm. In the curve acc/rej the value -4 represents an accepted step and -5 a rejected.

The simulation example above was chosen carefully. The model (12) describes the process well throughout the whole simulation interval. Also, in this particular problem the disturbance $\log \|\phi\|$ changes considerably and hence the prediction really improves the stepsize control.

As soon as the differential equation and/or the tolerance are such that the process is described by (16) instead of (12), the predicting controller may behave poorly. The situation is similar to the one with the old standard controller. The predicting controller does not manage to stabilize (16) for all relevant values on C_1 and C_2. For some problems the closed loop system will be locally unstable, resulting in an oscillating stepsize sequence, which may cause the produced solution to be erroneous. An example of the instability is shown in Figure 12. The problem 4 in Appendix D (the problem used in the identification experiment in Section 3) is solved from $t = 0$ to $t = 0.3$ with $tol = 10^{-4}$. During the initial

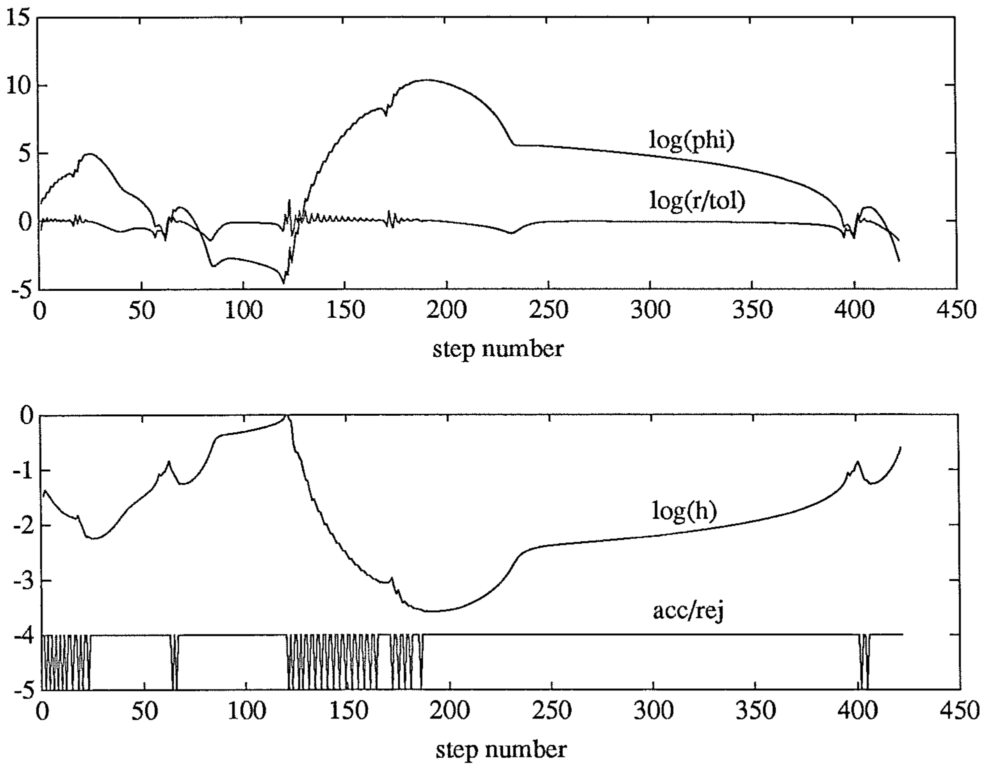

Figure 11. The Brusselator (problem 5 in Appendix D) simulated from $t = 0$ to $t = 30$ using the normal PI-controller for stepsize control. All variables are plotted as functions of step number. The variables $\log \|\phi\|$, $\log r$ and $\log h$ are plotted using base 10 logarithm. In the curve acc/rej the value -4 represents an accepted step and -5 a rejected.

transient (approximately the 50 first steps) the problem is well described by the model (12), and the predicting controller works well. During the transient $\log \|\phi\|$ decreases and the stepsize increases. Eventually the point where (16) should be considered instead of (12) is reached. Then the predicting controller fails to stabilize the process and the result is an oscillating stepsize and error sequence, with a lot of rejected steps.

The bottom line of this excursion is that for the integration methods considered here, i.e. explicit Runge-Kutta methods, it may not be a good idea to include the prediction of $\log \|\phi\|$ in the controller. The resulting controller would not manage to stabilize all the different process models encountered. However, the concept may still be of interest for other types of integration methods with a stability region of a different shape (e.g. A-stable, $A(\alpha)$-stable methods), or for specialized problems and tolerances where (12) always holds.

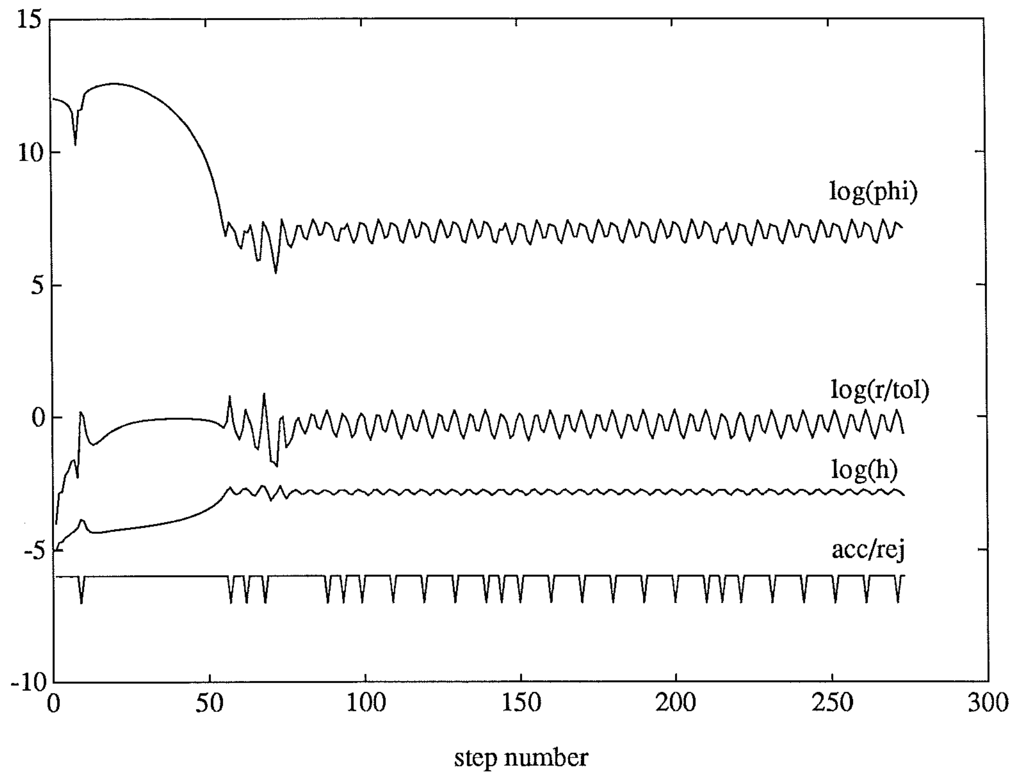

Figure 12. A simulation of problem 4 in Appendix D. The problem is solved from $t = 0$ to $t = 0.3$ using the predicting PI-controller for stepsize control. The parameter ξ in (31) was chosen as 0.5. All variables are plotted as function of step number. The variables $\log \|\phi\|$, $\log r$ and $\log h$ are plotted using base 10 logarithm. In the curve acc/rej the value -6 represents an accepted step and -7 a rejected.

7. Numerical Tests

To demonstrate some of the properties of both the old standard controller and the new PI-controller the problems in Appendix D were simulated using DOPRI45 [Hairer et al. 1987]. DOPRI45 is an explicit Runge-Kutta method which implements two different formulae of order four and five respectively. The fifth order formula was used to update the solution. The embedded error estimator is also of fifth order and since EPUS was used for stepsize control the gain k in (12) equals 4.

The error was measured with a mixed absolute-relative 2-norm, i.e.

$$r = \sqrt{\sum_i \left(\frac{\hat{e}_i}{|y_i| + \eta_i} \right)^2}$$

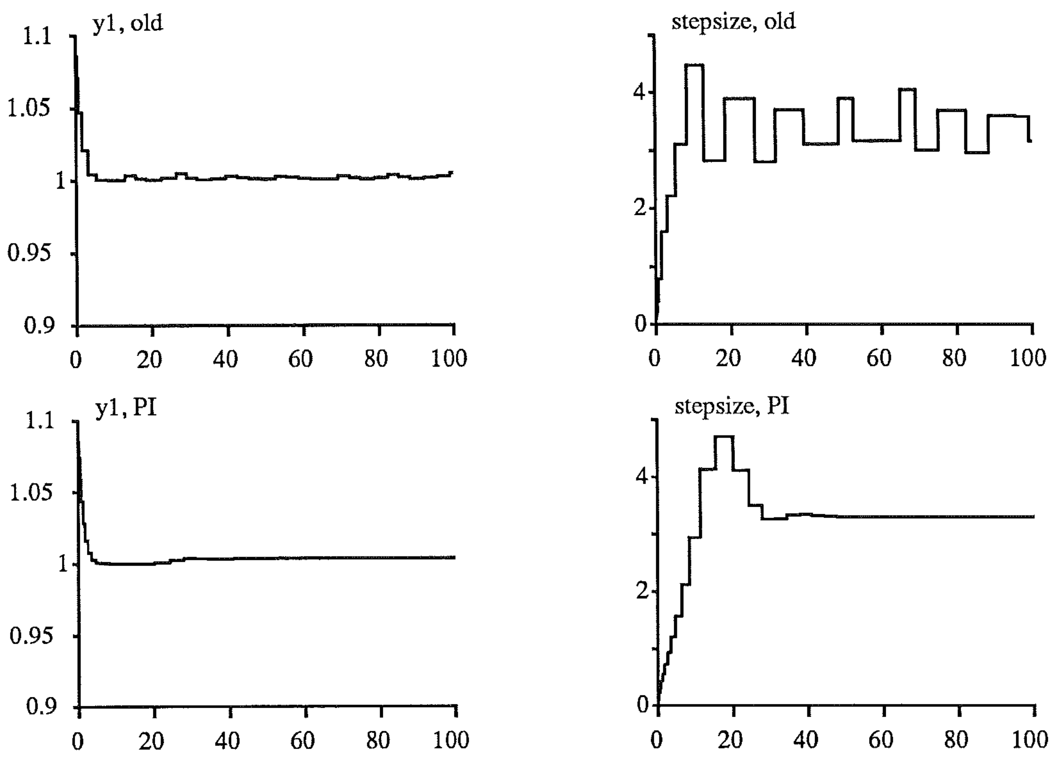

Figure 13. Simulation of Problem 1 with $tol = 10^{-3}$. At stationarity the old standard controller fails to produce a smooth stepsize sequence.

with $\eta_i = 0.1, \forall i$. In one of the simulations the max-norm was used to demonstrate some properties depending on the norm. The explicit expression for the mixed absolute-relative max-norm is

$$r = \max_i \left| \frac{\hat{e}_i}{|y_i| + \eta_i} \right|$$

with the same choice for η_i as above.

The integration method was coded as a Pascal system within the simulation program Simnon [Elmquist et al. 1986]. This provides good means for plotting and a convenient way to change parameters in the stepsize control routines.

Linear Problems

The first simulation example is Problem 1 (All problems can be found in Appendix D). During the transient the stepsize is small and $h\lambda$ is well inside the stability region of DOPRI45. When the solution of the problem reaches stationarity, $\|\phi\|$ decreases and consequently the stepsize

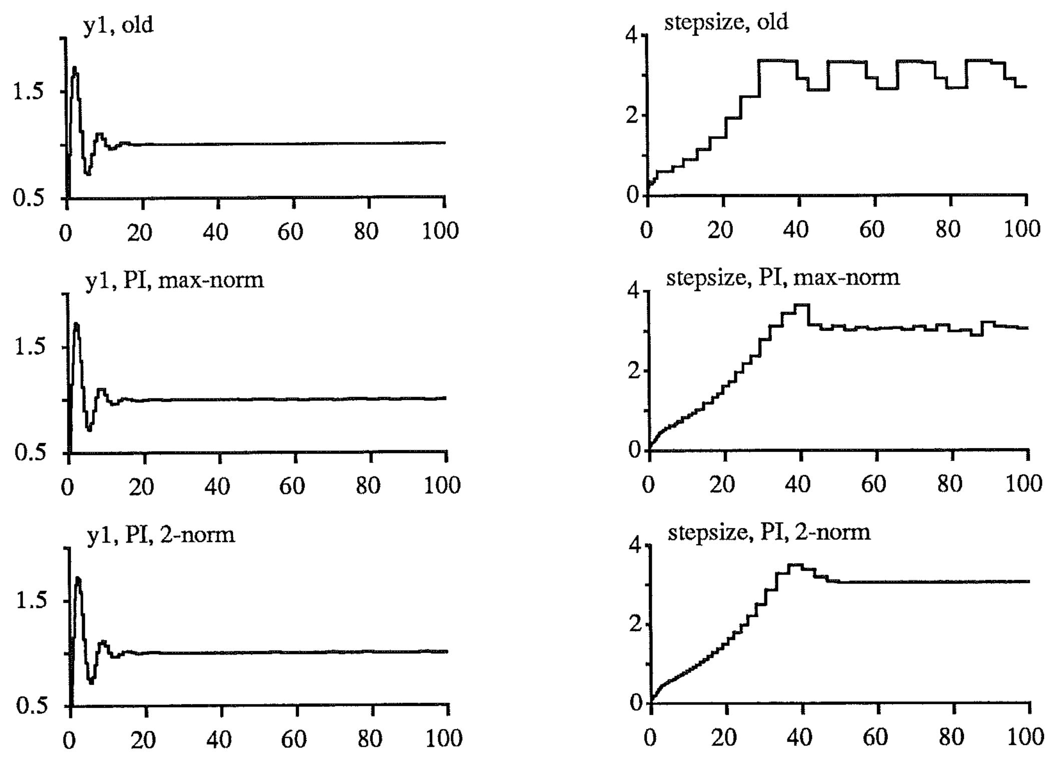

Figure 14. Simulation of Problem 2 with $tol = 10^{-4}$. At stationarity the old standard controller fails to produce a smooth stepsize sequence. This is achieved with the PI-controller if the 2-norm is used.

controller will increase the stepsize to make r equal tol. Since $\|\phi\| \rightarrow 0$, $h\lambda$ will eventually reach ∂S and the process model changes from (12) to (16). For this case the standard stepsize controller fails to produce a smooth stepsize sequence for this case, while the new PI-controller does not have any problems. The solution and the stepsize sequences are shown in Figure 13. Note that in stationarity the stepsize is approximately 3.3 and hence $h\lambda = -3.3$. For DOPRI45, ∂S intersects the negative real axis at -3.31.

In Problem 1 a forcing term equal to 1 is included. This gives a steady state solution $y_1 = 1$. The same behavior as in Figure 13 would have been observed for any constant value on the forcing term; also the value zero. In addition, there is nothing in Figure 13 that can be attributed to the transition from relative to absolute norm, since the forcing term and the initial value $y_1(0)$ were chosen such that the solution could not pass zero.

Problem 2 is a second order linear problem with two complex eigenvalues. Apart from the constant forcing term, it has the same form as the second example in Appendix A. One of the state variables (y_1) and the stepsize sequences for this problem is shown in Figure 14. The behavior is similar to the one observed for Problem 1, but here the choice of norm also makes a difference. In Appendix A the 2-norm had to be used to obtain a stationary solution with $h\lambda$ on $\partial\mathcal{S}$. This shows up in Figure 14, where the max-norm leads to a varying stepsize at stationarity, while the 2-norm results in a constant stepsize.

The behavior observed in Figure 13 and Figure 14 is the same for all linear problems with constant forcing term. At stationarity $\|\phi\| \to 0$ and to make r non-decreasing the difference equation (resulting from the discretization of the differential equation) has to put an eigenvalue on $\partial\mathcal{S}$.

For the two previous problems the quality of the produced solutions were acceptable despite the irregular stepsize sequences. This is not the case for Problem 3, a linear problem originating from a small control system. The problem consists of a continuous time PID-controller and a fourth order plant. The variable y_6 is the output from the plant and y_{PID} is the output from the controller (and thus the input to the plant). The system is described by

$$\begin{aligned} y_6 &= \frac{1}{(\rho+1)^4}\, y_{PID} && \text{(plant)} \\ y_{PID} &= k\Big(y_r - y_6 + \frac{1}{\rho T_i}(y_r - y_6) - \frac{\rho T_d}{\rho T_d/N + 1}\, y_6\Big) && \text{(controller)} \\ y_r &= 1 && \text{(reference signal)} \end{aligned}$$

with ρ being the differential operator. The parameter values given in Appendix D yields a PID-controller well tuned for the plant.

Problem 3 has four complex eigenvalues and two real. Five of the eigenvalues have a magnitude approximately equal to 1, while the sixth eigenvalue $\lambda_6 \approx -40$. The eigenvalue λ_6 is related to the filtering of the D-part in the controller.

When solving Problem 3 the transient corresponding to λ_6 dies out very fast. Consequently the stepsize controller increases the stepsize and soon $h\lambda_6$ is placed on $\partial\mathcal{S}$. For the standard controller this results in an irregular stepsize sequence which will excite the fast mode corresponding to λ_6. The error estimator fails to recover this mode properly and the produced solution is erroneous. A plot of y_{PID} was shown in Figure 1 in the beginning of this paper. The incorrect solution is the result of a simulation using the old standard stepsize controller for $tol = 10^{-2}$.

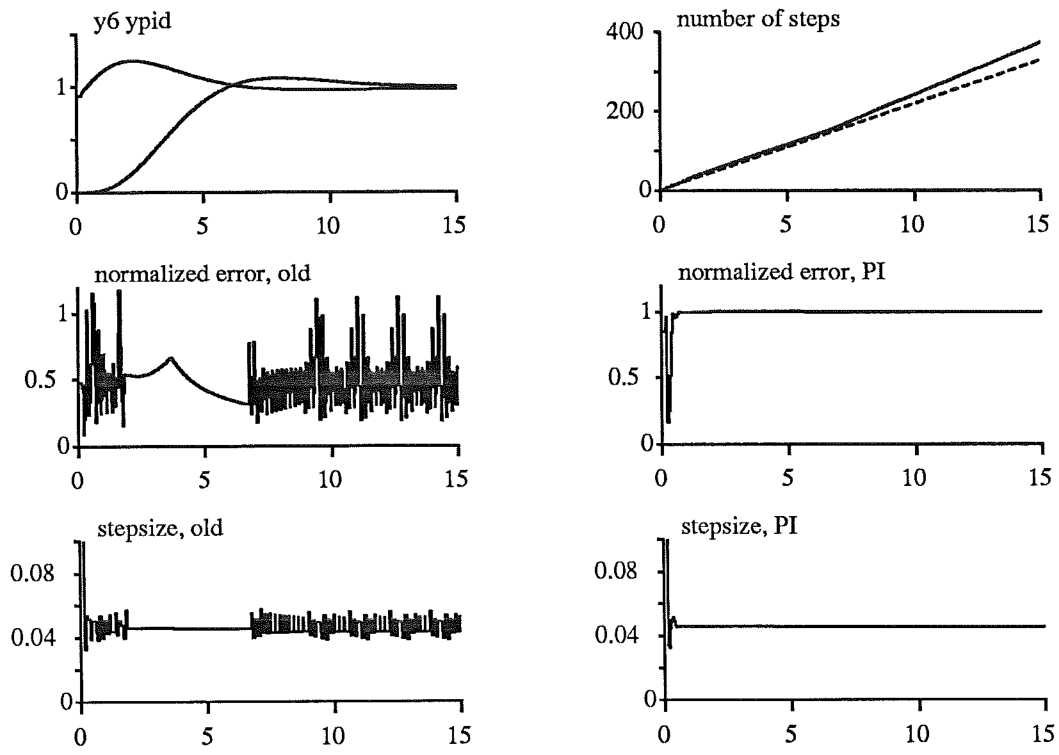

Figure 15. Simulation of Problem 3 with $tol = 10^{-2}$.

The correct solution in the same figure is produce by instead using the PI-controller for stepsize control.

Figure 15 shows some more signals from the simulation of Problem 3. The figure consists of six small plots where all signals are plotted as functions of time. The upper left plot shows the correct solution to the problem. The upper right plot shows two curves corresponding to the work needed to solve the problem. It is the total number of integration routine calls for both the old controller (solid line) and the new PI-controller (dashed line). Note that also the rejected steps are included to properly reflect the total work. The two plots in the middle show the estimated error r normalized with tol for the old (left) and the new (right) controller. The two lower plots compare the stepsize for the controllers.

From Figure 1 it is known that the irregular stepsize sequence produced by the old controller results in an erroneous solution to Problem 3. Additionally, it is also worth noting (see Figure 15) that the irregular stepsize sequence produces a lot of rejected steps, making the total amount of work needed to solve the problem being 20 % larger than that for the PI-controller. Note how the dead-zone in the standard controller manages to prevent the stepsize oscillation from $t = 2$ to $t = 7$. In con-

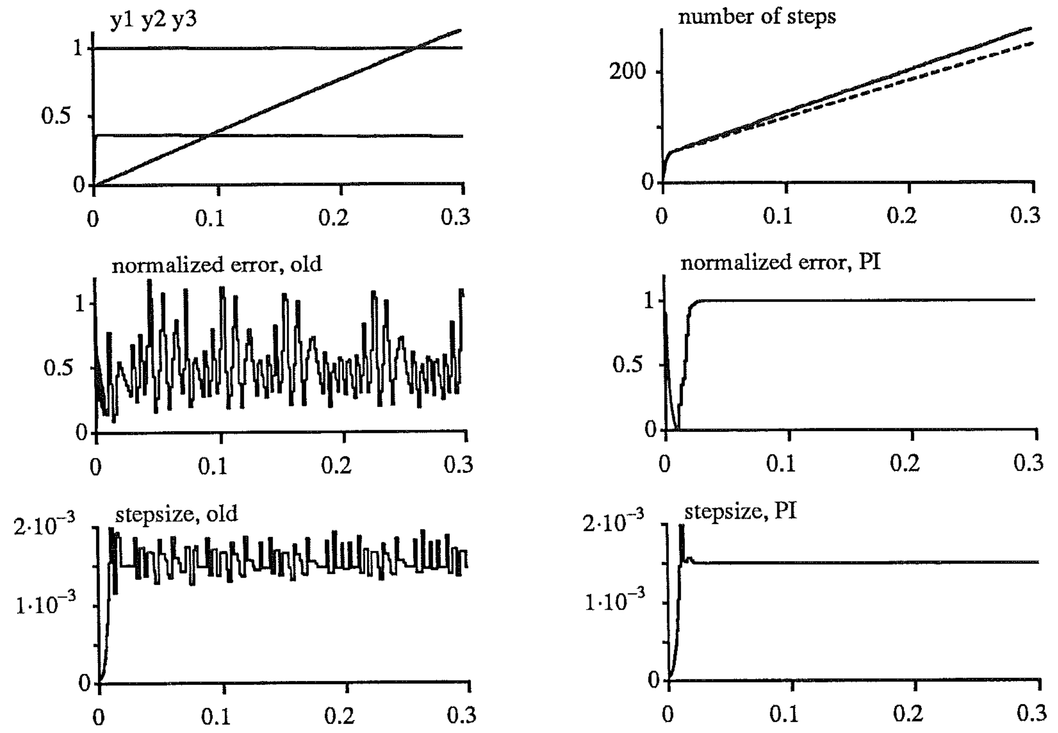

Figure 16. Simulation of Problem 4 with *tol* $= 10^{-4}$. For this tolerance $h\lambda_{max}$ will end up at ∂S at stationarity, and consequently the old controller behaves badly.

trast to the standard controller, the PI-controller quickly finds the correct stepsize and manages to control the error almost perfectly.

Nonlinear Problems

For linear problems, $\|\phi\|$ tends to zero in stationarity, which causes the stepsize to be increased and eventually $h\lambda$ ends up on ∂S. For nonlinear problems the situation is different. In stationarity $\|\phi\| \to C$, where C depends on the problem. From (12) it is now clear that C will affect what stepsize it takes to make r equal *tol* in stationarity. The larger *tol* the larger stepsize, and for sufficiently large *tol*, $h\lambda_{max}$ (λ_{max} is the dominating eigenvalue of the jacobian of the problem) will end up on ∂S. Hence for the same problem we expect to see different behavior in stationarity for different values of *tol*.

This effect can be seen for Problem 4 (the problem used in the identification experiment in Section 3 and Appendix C). At stationarity, the dominating eigenvalue $\lambda_{max} \approx -2200$, and hence a stepsize $h \approx 1.5 \cdot 10^{-3}$ will put $h\lambda_{max}$ on ∂S. For the standard controller this case leads to an

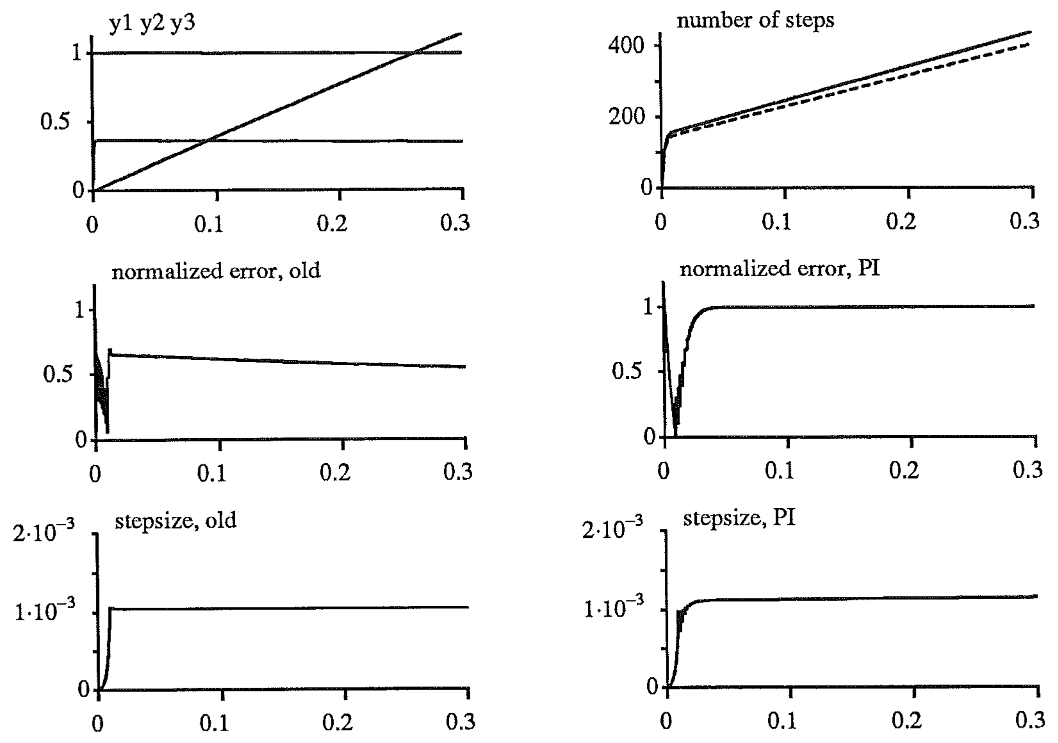

Figure 17. Simulation of Problem 4 with $tol = 10^{-6}$. For this tolerance $h\lambda_{max}$ is inside $\mathcal{S}$ at stationarity, and both controllers behave well.

irregular stepsize sequence and many rejected steps readily seen in Figure 16, where the problem is solved for $tol = 10^{-4}$. For the same case the PI-controller quickly finds the correct stepsize and solves the problem with 10-15 % less work. In Figure 17 Problem 4 is solved for $tol = 10^{-6}$. Now $h\lambda_{max}$ stays inside $\mathcal{S}$ at stationarity, and both controllers behave well. Still the PI-controller requires less work. This is due to the safety factor γ in the old controller, which leads to shorter steady state stepsize than that used by the PI-controller. On the other hand, if the safety factor is dropped there is a significant increase in rejected steps for the case $tol = 10^{-4}$.

Next Problem 5, the Brusselator, a moderately stiff problem is solved. The PI-controller manages to produce a smoother stepsize sequence than the old controller. Also due to the improved handling of rejected steps, the PI-controller requires less steps during the fast state transitions at $t \approx 25$.

Finally Problem 6, a version of the van der Pol oscillator, is solved. It is an interesting problem combining properties from both Problem 4 and Problem 5. The solution of the problem consists of some almost sta-

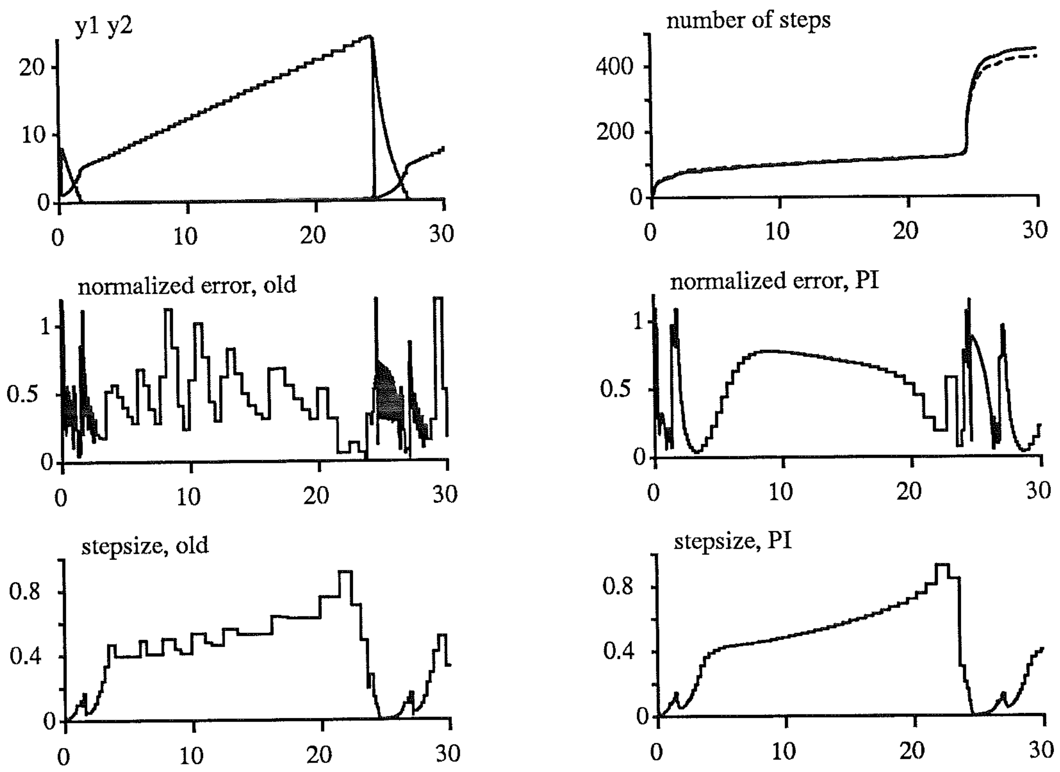

Figure 18. Simulation of Problem 5 with $tol = 10^{-4}$.

tionary parts and some very fast state transitions. The PI-controller is more efficient during both these parts of the solution. During the stationary parts the situation is much like the one for Problem 4, and the old controller produces a very irregular stepsize sequence resulting in many rejected steps. Then during the fast state transitions the PI-controller requires less steps due to its way of restarting after a rejected step. The overall benefits of using the PI-controller is a smoother stepsize sequence and a saving of some 10-15 % of work.

In [Gustafsson et al. 1988] some more problems are simulated. For some of these problems the PI-controller required more total work than the standard controller. The reason is that the used PI-controller did not incorporate the restart strategy after a rejected step described in Section 5. By including the restart strategy the PI-controller has been found to perform better or equally well for all problems tested.

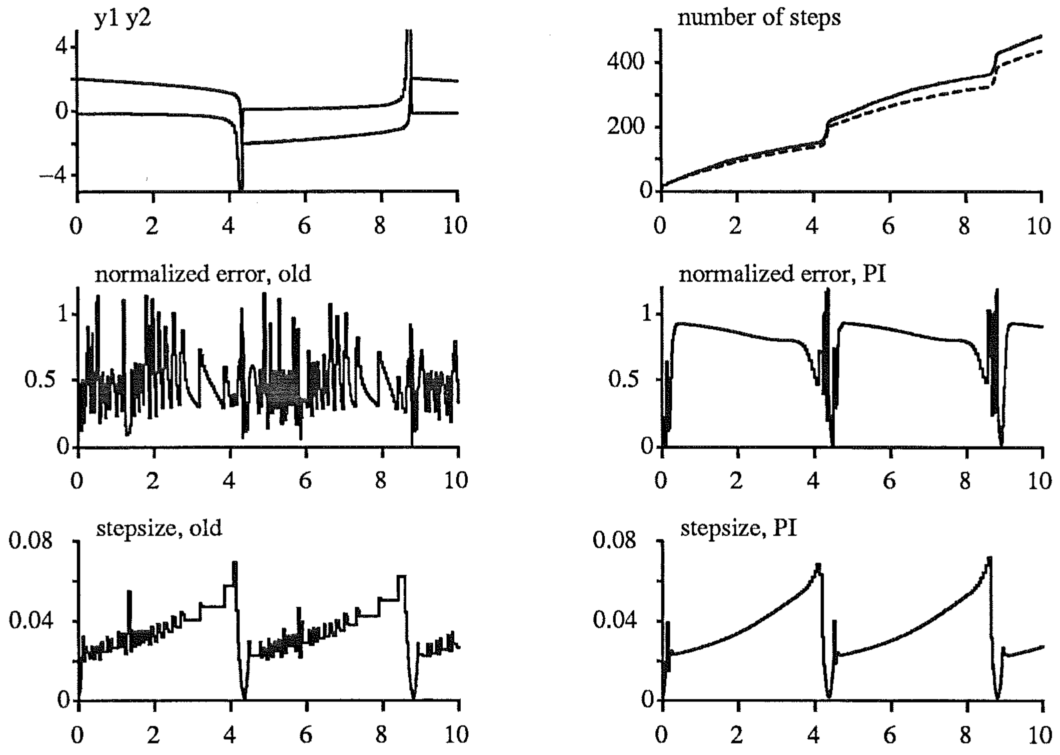

Figure 19. Simulation of Problem 6 with $tol = 10^{-3}$.

8. Conclusions

Control theory provides efficient means to analyze the problem of stepsize control in numerical integration. It naturally separates an integration routine into two parts: the process (integration method, differential equation and error estimator) and the stepsize controller. Hence an integration method can be constructed for optimal numerical behavior, and then a fitting stepsize controller is designed.

To design the stepsize controller a process model is needed. The static asymptotic relation, between the stepsize and the local truncation error, normally assumed is not sufficient. When the stepsize is limited by numerical stability, a dynamic model has to be used. Such a model was derived and numerically verified for explicit Runge-Kutta methods.

Using the dynamic model it is straightforward to analyze the standard stepsize controller. The analysis gives insight and clearly points out that the standard stepsize controller combined with a problem where numerical stability limits the stepsize leads to a locally unstable closed loop system.

The standard stepsize controller can be regarded as an I-controller. A generalization to a PI-controller is then natural, and using design techniques from automatic control its parameters can be tuned such that good control is achieved also when numerical stability limits the stepsize. The PI-controller gives better overall performance at little extra expense.

Here only explicit Runge-Kutta methods were considered, and the proposed controller was a PI-controller. There is, however, nothing that limits the used methodology to these cases. Similar analytical models can probably be derived for other types of integration methods, and once a model is obtained it can be used to analyze and improve the stepsize control.

Acknowledgements

The author wishes to express his gratitude to Gustaf Söderlind and Per Hagander. First to Gustaf Söderlind, for teaching the subject of numerical integration and introducing the problem of stepsize control, and to Per Hagander for persuading and convincing the author that a dynamic model of an integration method would be possible to derive.

In addition, Per Hagander, Gustaf Söderlind and Sven Erik Mattsson have read several versions of the manuscript and their comments and suggestions have greatly improved the quality of the paper.

The work has been supported by the Swedish National Board for Technical Development (STU) under contract 88-01611P.

References

Åström, K. J. and B. Wittenmark (1984): *Computer Controlled Systems, Theory and Design*, Prentice-Hall, Englewood Cliffs.

Dahlquist G, Å. Björk and N. Andersson (1974): *Numerical Methods*, Prentice-Hall, Englewood Cliffs, New Jersey, p. 336.

Elmqvist, H, K. J. Åström and T. Schönthal (1986): *SIMNON, User's Guide for MS-DOS Computers*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Enright, W. H, T. E. Hull and B. Lindberg (1975): "Comparing Numerical Methods for Stiff Systems of ODE's," *BIT*, 15, 28 – 33.

Franklin, G. F, J. D. Powell and A. Emami-Naeini (1986): *Feedback Control Systems*, Addison-Wesley, pp. 99 – 103.

Gear, C. W. (1971): *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall.

Gustafsson, K, M. Lundh, G. Söderlind (to appear): "A PI Stepsize Control for the Numerical Solution of Ordinary Differential Equations," *To appear in BIT.*

Hairer, E, S. P. Nørsett and G. Wanner (1987): *Solving Ordinary Differential Equations, I*, Springer.

Hall, G. (1985): "Equilibrium States of Runge-Kutta Schemes: Part I," *ACM Transactions on Mathematical Software*, **11**, 3, 289 – 301.

Hall, G. (1986): "Equilibrium States of Runge-Kutta Schemes: Part II," *ACM Transactions on Mathematical Software*, **12**, 3, 183 – 192.

Hall, G. and D. J. Higham (1987): "Analysis of Stepsize Selection for Runge-Kutta Codes," NA report No. 137, University of Manchester.

Higham, D. J. and G. Hall (1987): "Embedded Runge-Kutta Formulae with Stable Equilibrium States," NA report No. 140, University of Manchester.

Hoffmann de Visme, G. (1971): *Binary Sequences*, The English Universities Press Ltd, St Pauls House, Warwick Lane, London.

LJUNG, L. and T. SÖDERSTRÖM (1983): *Theory and Practice of Recursive Identification*, The MIT Press.

MOLER, C., J. LITTLE, S. BANGERT and S. KLEIMAN (1987): *PRO-MATLAB, Users Guide*, The MathWorks, Inc..

SHAMPINE L. F. (1975): "Stiffness and Nonstiff Differential Equation Solvers," in L. Collatz (Ed.): *Numerische Behandlung von Differentialgleichungen*, Information Series of Numerical Mathematics 27, Birkhauser Verlag, Basel, pp. 287–301.

ZONNEVELD, J. A. (1964): "Automatic Numerical Integration," Akademisch Proefschrift, Matematisch Centrum, Amsterdam.

Appendix A: A Model for Explicit Runge-Kutta Methods

A model that relates the scalar error estimate r to the stepsize h of an explicit Runge-Kutta method will be derived. If the logarithms of r and h are used, the model can be written as a linear transfer function.

To derive a model two common test problems will be considered. First a model for the case of the integration method being applied to a real scalar test problem is derived. Then a test problem with two complex eigenvalues is considered. Apart for some minor modifications the same model holds for this case as well. Finally the case of a general linear problem is discussed.

A derivation similar in spirit can be found in [Hall 1985, 1986].

The Integration Method

An explicit m-stage Runge-Kutta method [Hairer et al. 1987] applied to the initial-value problem

$$\begin{aligned} \dot{y} &= f(t,y) \qquad 0 \le t \le T, \quad y, f \in \mathcal{R}^l \\ y(0) &= y_0 \end{aligned}$$

takes the following form

$$\begin{aligned} \dot{Y}_1 &= f(t_n, y_n) \\ \dot{Y}_2 &= f(t_n + c_2 h_n, y_n + h_n a_{21} \dot{Y}_1) \\ &\vdots \\ \dot{Y}_m &= f(t_n + c_m h_n, y_n + h_n \sum_{j=1}^{m-1} a_{mj} \dot{Y}_j) \\ y_{n+1} &= y_n + h_n \sum_{j=1}^{m} b_j \dot{Y}_j \\ t_{n+1} &= t_n + h_n \end{aligned} \tag{A.1}$$

$$
\begin{aligned}
\hat{e}_{n+1} &= h_n \sum_{j=1}^{m} (b_j - \hat{b}_j)\dot{Y}_j \\
\hat{d}_{n+1} &= \sum_{j=1}^{m} (b_j - \hat{b}_j)\dot{Y}_j \qquad \text{(A.1 continued)} \\
r_{n+1} &= \begin{cases} \|\hat{e}_{n+1}\|, & \text{(EPS)} \\ \|\hat{d}_{n+1}\|, & \text{(EPUS)} \end{cases}
\end{aligned}
$$

The m-stage method supports two formulas of orders p and $p+1$. They are represented by the coefficients $\{b_j\}$ and $\{\hat{b}_j\}$ respectively. One of these coefficient sets is used to advance the integration while the other is used for the error estimate. The case when the coefficient set corresponding to order $p+1$ is used to advance the solution is referred to as local extrapolation. Irrespectively of which coefficient set is used for the integration the error estimator is of order $p_e = p+1$.

When creating the scalar error estimate r the individual components of $\hat{e}$ (or $\hat{d}$) has to be related to the size of the solution y, and therefore a relative norm is preferred. However, for the case $y = 0$ not to pose any problems typically a mixed absolute-relative norm is used. Two common choices are

$$
\|\hat{e}\| = \max_i \left| \frac{\hat{e}_i}{|y_i| + \eta_i} \right|, \qquad \|\hat{e}\| = \sqrt{\sum_i \left(\frac{\hat{e}_i}{|y_i| + \eta_i} \right)^2} \tag{A.2}
$$

where η_i is a scaling factor for the i:th component of y, creating the mixed absolute-relative error measure.

The nonlinear difference equations in (A.1) constitute the true process. The behavior of the process is complicated, but its main properties can normally be captured by using a linearized model.

A Real Scalar Test Problem

Consider the real scalar initial value problem

$$
\begin{aligned}
\dot{x} &= \lambda(x - x_{stat}) \qquad t \geq 0, \quad \lambda < 0 \\
x(0) &= x_0
\end{aligned} \tag{A.3}
$$

Applied to this problem the Runge-Kutta algorithm can be expressed as the exact process

$$\begin{aligned} x_{n+1} &= P(h_n\lambda)(x_n - x_{stat}) + x_{stat} \\ \hat{e}_{n+1} &= E(h_n\lambda)(x_n - x_{stat}) \end{aligned}$$

where $P(h_n\lambda)$ and $E(h_n\lambda)$ are polynomials in $h_n\lambda$. Introduce y_n as the deviation of x_n from stationarity, i.e. $y_n = x_n - x_{stat}$. Then

$$\begin{aligned} \dot{y} &= \lambda y \qquad t \geq 0, \quad \lambda < 0 \\ y(0) &= x_0 - x_{stat} \end{aligned} \tag{A.4}$$

and

$$\begin{aligned} y_{n+1} &= P(h_n\lambda)y_n \\ \hat{e}_{n+1} &= E(h_n\lambda)y_n \end{aligned} \tag{A.5}$$

Behavior during transient (linearization around $h = 0$). The polynomial $E(h_n\lambda)$ can be written

$$E(h_n\lambda) = \kappa_0(h_n\lambda)^{p_e} + \kappa_1(h_n\lambda)^{p_e+1} + \ldots$$

Using this relation we may write

$$\begin{aligned} \log r_{n+1} = \log \|\hat{e}_{n+1}\| &= k \log h_n + \log \|\phi_n\| \\ \phi_n &= y_n\lambda^{p_e}(\kappa_0 + \kappa_1 h_n\lambda + \ldots) \end{aligned} \tag{A.6}$$

with $k = p_e$ (EPS) or $k = p_e - 1$ (EPUS). Here ϕ_n is measured with the same norm as $\hat{e}$ (or $\hat{d}$).

During the transient the deviation y_n is significant, and ϕ_n takes a large value (in relation to h and tol). From (A.6) it is clear that h_n has to be small for r_n not to violate the prescribed tolerance. Then it is also clear that $\kappa_0 \gg \kappa_1 h_n\lambda$ and the dependence between ϕ_n and h_n is weak.

Hence, using the forward shift operator q the model reads

$$\log r_n = \frac{1}{q}\left(k \log h_n + \log \|\phi_n\|\right) \tag{A.7}$$

where ϕ_n is a time-varying disturbance almost independent of h_n.

Steady state behavior (linearization around $h = h_s$). From (A.6) it is obvious that $\phi \to 0$ as $y_n \to 0$. To make r_n equal the tolerance, the stepsize has to increase, and for y_n sufficiently small $h_n = h_s$, where h_s puts $h_s\lambda$ on $\partial\mathcal{S}$ the border of the stability region $\mathcal{S}$, $\mathcal{S} = \{h\lambda : |P(h\lambda)| \leq 1\}$ of the integration method. A constant stepsize h_s leads to a stationary solution $|y_{n+1}| = |y_n|$ since $|P(h_s\lambda)| = 1$. Denote the steady state values by r_s and h_s, and consider small perturbations, i.e. $h_n = h_s(1 + \epsilon_n)$. It no longer holds that y_n is almost independent of h_{n-1}, and thus

$$\begin{aligned}
\hat{e}_{n+1} &= E(h_n\lambda)y_n = E(h_n\lambda)P(h_{n-1}\lambda)y_{n-1} = E(h_n\lambda)\frac{P(h_{n-1}\lambda)}{E(h_{n-1}\lambda)}\hat{e}_n \\
&= E\left(h_s\lambda(1+\epsilon_n)\right)\frac{P\left(h_s\lambda(1+\epsilon_{n-1})\right)}{E\left(h_s\lambda(1+\epsilon_{n-1})\right)}\hat{e}_n \\
&\approx E(h_s\lambda)\left(1 + \epsilon_n h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}\right) \\
&\quad \cdot \frac{P(h_s\lambda)\left(1 + \epsilon_{n-1}h_s\lambda\frac{P'(h_s\lambda)}{P(h_s\lambda)}\right)}{E(h_s\lambda)\left(1 + \epsilon_{n-1}h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}\right)}\hat{e}_n
\end{aligned} \tag{A.8}$$

Introduce

$$C_1(h_s\lambda) = h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}, \qquad C_2(h_s\lambda) = h_s\lambda\frac{P'(h_s\lambda)}{P(h_s\lambda)} \tag{A.9}$$

and (A.8) can be written

$$\begin{aligned}
\hat{e}_{n+1} &\approx E(h_s\lambda)\left(1 + \epsilon_n C_1\right)\frac{P(h_s\lambda)\left(1 + \epsilon_{n-1}C_2\right)}{E(h_s\lambda)\left(1 + \epsilon_{n-1}C_1\right)}\hat{e}_n \\
&\approx P(h_s\lambda)\left(1+\epsilon_n\right)^{C_1}\left(1+\epsilon_{n-1}\right)^{-C_1+C_2}\hat{e}_n \\
&= P(h_s\lambda)\left(\frac{h_n}{h_s}\right)^{C_1}\left(\frac{h_{n-1}}{h_s}\right)^{-C_1+C_2}\hat{e}_n
\end{aligned} \tag{A.10}$$

At steady state y_n is small, i.e. $r_n = |\hat{e}_n|/x_n \approx |\hat{e}_n|/x_{stat}$ (or if $x_{stat} < \eta$, $r_n \approx |\hat{e}_n|/\eta$. Using EPS and noting that on the stability boundary $\partial\mathcal{S}$,

$|P(h_s\lambda)| = 1$, (A.10) can be written

$$\log r_n = \frac{C_1 q + C_2 - C_1}{q(q-1)} (\log h_n - \log h_s) , \tag{A.11}$$

where q is the forward shift operator. Using EPUS changes the relation (A.11), and instead

$$\log r_n = \frac{(C_1 - 1)q + C_2 - C_1 + 1}{q(q-1)} (\log h_n - \log h_s) - \log h_s . \tag{A.12}$$

The last term $-\log h_s$ is just a consequence of the stationary difference between EPS and EPUS.

A Test Problem With Two Complex Eigenvalues

The attention is now turned to the initial value problem

$$\begin{aligned} \dot{y} &= Ay \qquad t \geq 0, \qquad A = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix}, \quad \alpha < 0 \\ y(0) &= y_0 \end{aligned} \tag{A.13}$$

The matrix A has eigenvalues $\lambda = \alpha + i\beta$ and $\bar{\lambda} = \alpha - i\beta$. An explicit Runge-Kutta method applied to the problem makes the process take the form

$$\begin{aligned} y_{n+1} &= P(h_n A) y_n \\ \hat{e}_{n+1} &= E(h_n A) y_n \end{aligned} \tag{A.14}$$

Behavior during transient (linearization around $h = 0$). With the same kind of reasoning as for problem (A.3) we may write

$$\begin{aligned} \log r_{n+1} &= \log \|\hat{e}_{n+1}\| = k \log h_n + \log \|\phi_n\| \\ \phi_n^T &= y_n^T A^{p+1} (\kappa_0 + \kappa_1 h_n A + \ldots) \end{aligned} \tag{A.15}$$

with $k = p_e$ (EPS) or $k = p_e - 1$ (EPUS). And thus

$$\log r_n = \frac{1}{q} (k \log h_n + \log \|\phi_n\|) \tag{A.16}$$

This model is the same as the one derived for the real scalar test problem (A.3), and with the same arguments used there, it is clear that the dependence between $\log \|\phi_n\|$ and $\log h_n$ is weak.

Behavior during steady state (linearization around $h = h_s$). As for the problem (A.3), ϕ will decrease and therefore, to keep r_n equal to the tolerance, the stepsize has to be increased. Eventually the stepsize will put $h_n\lambda$ (as well as $h_n\bar{\lambda}$) on the stability boundary $\partial\mathcal{S}$ of the method. The linearization is facilitated if we have a stationary solution, i.e. a solution where a constant stepsize h_s gives rise to a constant r_n. Generally this does not occur for a problem dominated by a complex conjugate pair of eigenvalues. It is achieved, but only for a very special choice of norm tailored to the solution of the problem.

First note that the matrix A in (A.13) is normal, $A^TA = \lambda\bar{\lambda}I$ and $A^T + A = (\lambda + \bar{\lambda})I$. Hence $A^{Tp} + A^p = (\lambda^p + \bar{\lambda}^p)I$ and

$$\begin{aligned}E(h_nA)^TE(h_nA) &= E(h_n\lambda)E(h_n\bar{\lambda})I\\ P(h_nA)^TP(h_nA) &= P(h_n\lambda)P(h_n\bar{\lambda})I\end{aligned}$$

By using the 2-norm,

$$\begin{aligned}\|\hat{e}_n\|_2^2 &= y_n^TE(h_nA)^TE(h_nA)y_n\\ &= E(h_n\lambda)E(h_n\bar{\lambda})\|y_n\|_2^2 = |E(h_n\lambda)|^2\|y_n\|_2^2\\ \|y_n\|_2^2 &= y_n^TP(h_nA)^TP(h_nA)y_n\\ &= P(h_n\lambda)P(h_n\bar{\lambda})\|y_n\|_2^2 = |P(h_n\lambda)|^2\|y_n\|_2^2\end{aligned}$$

There exists a stepsize h_s such that $|P(h_s\lambda)| = 1$. For such a stepsize the norm of y is constant. Consider small perturbations, i.e. $h_n = h_s(1 + \epsilon_n)$, and use the same derivations as for problem (A.3).

$$\begin{aligned}\|\hat{e}_{n+1}\|_2^2 &= |E(h_n\lambda)|^2\|y_n\|_2^2\\ &= |E(h_s\lambda(1+\epsilon_n))|^2\frac{|P(h_s\lambda(1+\epsilon_{n-1}))|^2}{|E(h_s\lambda(1+\epsilon_{n-1}))|^2}\|\hat{e}_n\|_2^2\\ &\approx |1+\epsilon_nC_E|^2\frac{|1+\epsilon_{n-1}C_P|^2}{|1+\epsilon_{n-1}C_E|^2}\|\hat{e}_n\|_2^2\end{aligned}$$

where

$$C_E(h_s\lambda) = h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}, \qquad C_P(h_s\lambda) = h_s\lambda\frac{P'(h_s\lambda)}{P(h_s\lambda)}.$$

Moreover, using $|1+\epsilon C|^2 \approx 1 + 2\epsilon \operatorname{Re}(C)$

$$\begin{aligned} \|\hat{e}_{n+1}\|_2^2 &\approx (1+2\epsilon_n C_1)\frac{1+2\epsilon_{n-1}C_2}{1+2\epsilon_{n-1}C_1}\|\hat{e}_n\|_2^2 \\ &= \left(\frac{h_n}{h_s}\right)^{2C_1}\left(\frac{h_{n-1}}{h_s}\right)^{-2C_1+2C_2}\|\hat{e}_n\|_2^2 \end{aligned} \tag{A.17}$$

where

$$\begin{aligned} C_1(h_s\lambda) &= \operatorname{Re}\,(C_E) = \operatorname{Re}\left(h_s\lambda\frac{E'(h_s\lambda)}{E(h_s\lambda)}\right), \\ C_2(h_s\lambda) &= \operatorname{Re}\,(C_P) = \operatorname{Re}\left(h_s\lambda\frac{P'(h_s\lambda)}{P(h_s\lambda)}\right) \end{aligned} \tag{A.18}$$

Hence using EPS (A.17) may be written

$$\log r_n = \frac{C_1 q + C_2 - C_1}{q(q-1)}(\log h_n - \log h_s), \tag{A.19}$$

while EPUS yields

$$\log r_n = \frac{(C_1-1)q + C_2 - C_1 + 1}{q(q-1)}(\log h_n - \log h_s) - \log h_s. \tag{A.20}$$

Note that apart from the modified definition of C_1 and C_2 the models are identical to (A.11) and (A.12) which were derived for the real scalar test problem (A.3).

The results (A.19) and (A.20) are very special. In the case of dominating complex eigenvalues it only holds for the special eigenstructure of the matrix A. If the problem (A.13) is transformed with the nonsingular matrix T ($y = T\tilde{y}$), giving rise to the system matrix $T^{-1}AT$, the norm $\|\tilde{y}\|_T = \|T\tilde{y}\|_2$ has to be used to establish (A.19) and (A.20).

Even if the results only can be established for a very special norm, it is still likely that the transformed system behaves as (A.19) or (A.20) with the exception that h_s may be a time-varying entity.

Appendix B: Identification of a Model for DOPRI45

The explicit Runge-Kutta method DOPRI45 was used in the identification. For more details about the method see Section 3 or [Harier et al. 1987].

The Problem

The identification was done using the nonlinear problem 4 in Appendix D. It has its origin in chemical reaction kinetics and is due to Robertson [Enright et al. 1975]. The first 0.3 seconds of the solution to the problem is shown in Figure B.1. After a fast initial transient y_1 and y_2 are almost constant (slowly decreasing) while y_3 increases almost linearly.

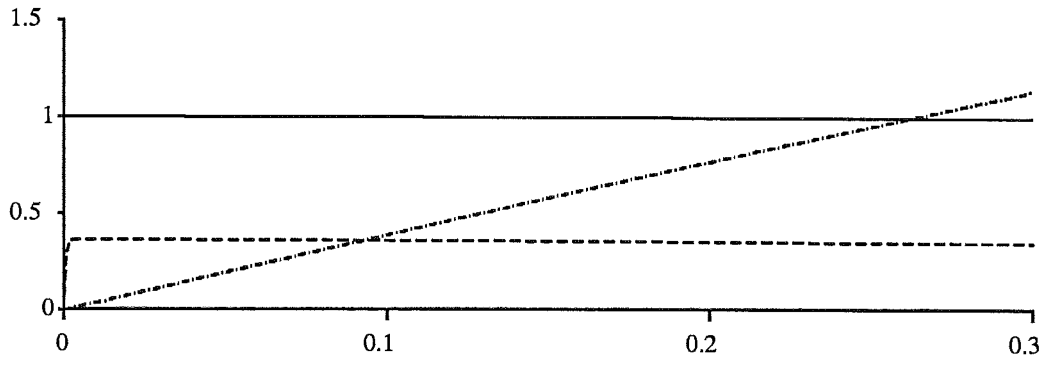

Figure B.1 The solution of the nonlinear problem 1 in Appendix D. y_1 (solid line), y_2 (dashed line) and y_3 (dash-dotted line).

The Jacobian of the problem is

$$J = \begin{pmatrix} -0.04 & 0.01y_3 & 0.01y_2 \\ 400 & -100y_3 - 6000y_2 & -100y_2 \\ 0 & 60y_2 & 0 \end{pmatrix}$$

At $t = 0$ the eigenvalues of J are 0, 0 and -0.040. Due to mass conservation one of the eigenvalues at the origin will stay there for all t. During the initial transient ($0 \leq t \leq 0.005$) the other eigenvalue at the origin moves out to -2180. The small negative eigenvalue moves to -0.40. As time increases the large negative eigenvalue will continue to move out from the origin while the small negative eigenvalue moves back towards

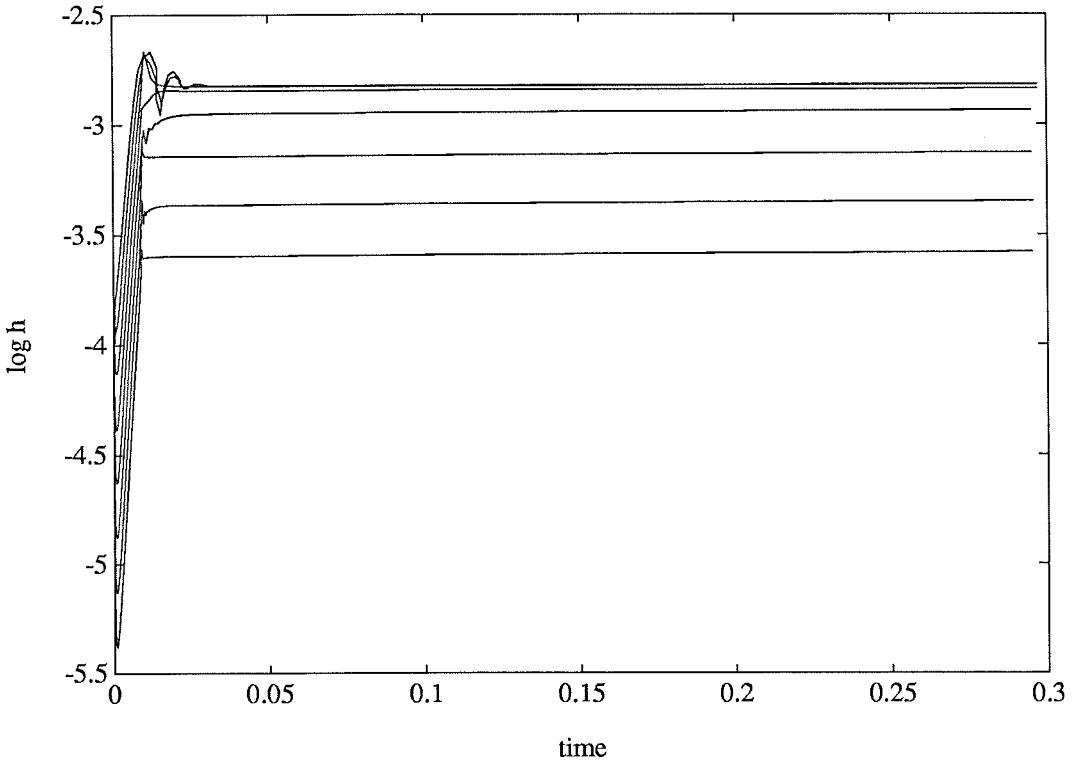

Figure B.2 Stepsize for different tolerances. Note that it is the logarithm of the stepsize that is plotted. The curves come in order, i.e. the lower one corresponds to $tol = 10^{-9}$, the second from the bottom corresponds to $tol = 10^{-8}$, and so on. For $tol = 10^{-4}, \dots, 10^{-2}$ the stationary value of the stepsize is identical and the curves overlap.

the origin again. At $t = 3.0$ they have reached the values -2244 and -0.13 respectively.

The problem was solved for eight different tolerances ($tol = 10^{-2}$, 10^{-3}, ... , 10^{-9}). The stepsize sequences obtained are shown in Figure B.2. After the initial transient the stepsize is almost constant. The PI-controller (26) described in Section 5 was used, since for the solved problem the stepsize is limited by numerical stability for some tolerances, and then the standard controller yields a locally unstable closed loop system (see Section 4).

The stepsize controller tries to keep the error measure constant and equal to *tol*. For a linear problem without time-varying forcing terms this eventually requires a stepsize h_n that puts $h_n\lambda$ on $\partial\mathcal{S}$. This is a consequence of the fact that for linear problems $\phi \rightarrow 0$ at stationarity. For a nonlinear problem higher order effects may very well make ϕ not

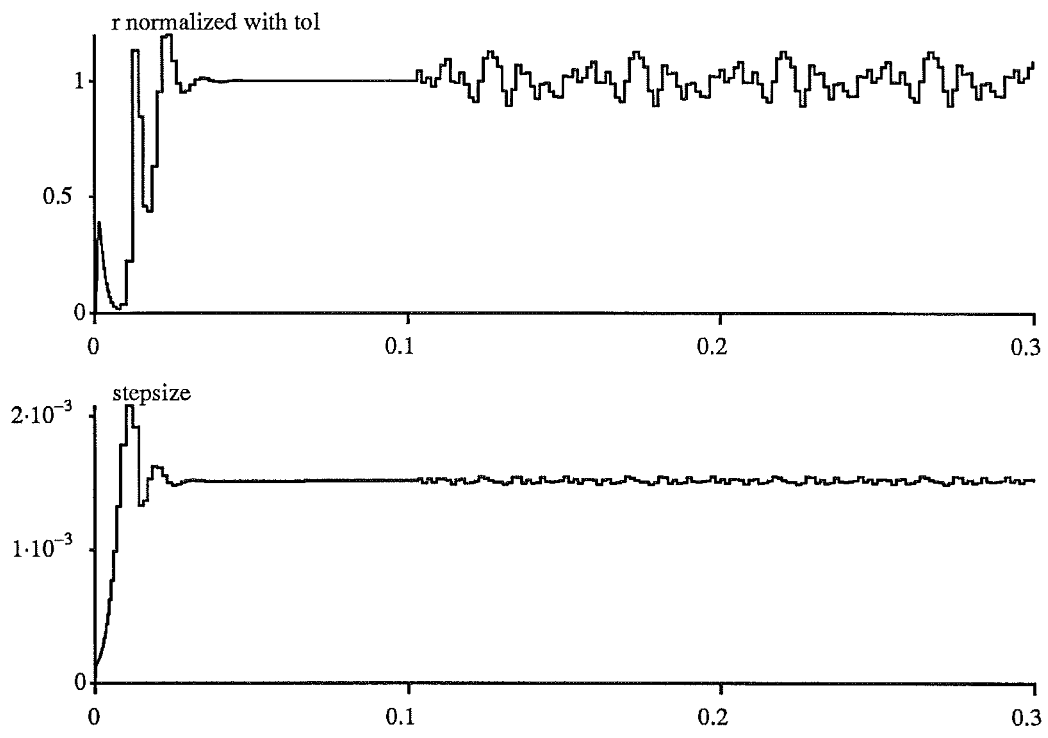

Figure B.3 Stepsize and error estimate recorded at the identification experiment with $tol = 10^{-2}$. The irregularities in the data sequences is caused by the perturbation of tol (A.20).

tend to zero at stationarity, and it is possible to get constant error for constant stepsizes where $h_n\lambda$ is strictly inside $\mathcal{S}$. The smaller the stepsize the smaller the error.

This effect can readily be observed in the problem above. For different values of tol the stationary value of h_n will take different values: the larger the value of tol, the larger the stepsize. This holds true for tolerances below 10^{-5}. For larger values of tol the stepsize will be sufficiently large to put $h\lambda_{max}$ on $\partial\mathcal{S}$ for DOPRI45, i.e. $h\lambda_{max} = 1.51 \cdot 10^{-3} \cdot (-2180) = -3.3$.

Identification Procedure

The identification was carried out using Simnon [Elmqvist et al. 1986] and PRO-MATLAB [Moler et al. 1987]. Simnon was used to code the integration routine while the identification toolbox in PRO-MATLAB supplied the routines for identification.

Data collecting. The problem was solved using DOPRI45 for different values on tol. At $t = 0.1$, after the transient has died out completely, an excitation signal was added by perturbing $\log tol$ according to

$$\log tol = \log tol_0 + 0.05\Delta tol_n. \tag{B.1}$$

Here Δtol_n was a PRBS (pseudo random binary signal [Hoffmann de Visme 1971]) sequence alternating between $+1$ and -1. The perturbation was small and the stepsize only varied a few percent around its stationary value. For each value of tol_0 the stepsize h_n and the error estimate r_n were recorded and stored. During the data logging there were no rejected steps. The experiment was done for $tol_0 = 10^{-2}, 10^{-3}, \ldots, 10^{-9}$. In Figure B.3 the stepsize and the error estimate recorded for $tol = 10^{-2}$ is shown. Note that the irregular stepsize and error sequence is a consequence of perturbing tol according to (B.1), and not due to poor stepsize control.

Data conditioning. The data sequences have non-zero mean values. They also contain a slow ramp due to the large negative eigenvalue that slowly increases. To remove these effects a least squares fit of a ramp was done to each data sequence, and thereafter the ramp was subtracted from the data sequence.

Identification. For each pair of data signals (stepsize sequence and error sequence) an ARMA-model was identified. The Akaike test and the statistics of the residuals were used to decide upon a correct model order [Moler et al. 1987, Ljung and Söderström 1983]. The identified transfer functions from $\log h_n$ to $\log r_n$ were

$$tol_0 = 10^{-2} \qquad G(q) = \frac{4.85q + 1.23}{q(q-1.00)} \qquad h_n\lambda_{max} \approx -3.3$$

$$tol_0 = 10^{-3} \qquad G(q) = \frac{4.85q + 1.23}{q(q-1.00)} \qquad h_n\lambda_{max} \approx -3.3$$

$$tol_0 = 10^{-4} \qquad G(q) = \frac{4.86q + 1.18}{q(q-1.00)} \qquad h_n\lambda_{max} \approx -3.3$$

$$tol_0 = 10^{-5} \qquad G(q) = \frac{4.88q + 0.84}{q(q-0.75)} \qquad h_n\lambda_{max} \approx -3.1$$

$$tol_0 = 10^{-6} \qquad G(q) = \frac{4.87q - 0.15}{q(q-0.24)} \qquad h_n\lambda_{max} \approx -2.4$$

$$tol_0 = 10^{-7} \qquad G(q) = \frac{4.60}{q} \qquad h_n\lambda_{max} \approx -1.6$$

$$tol_0 = 10^{-8} \qquad G(q) = \frac{4.41}{q} \qquad h_n\lambda_{max} \approx -0.90$$

$$tol_0 = 10^{-9} \qquad G(q) = \frac{4.25}{q} \qquad h_n\lambda_{max} \approx -0.54$$

As $h_n\lambda_{max}$ increases to the value that puts $h_n\lambda_{max}$ on $\partial\mathcal{S}$ the process gradually changes between the two models derived in Appendix A. For $h_n\lambda_{max}$ small and $h_n\lambda_{max}$ on $\partial\mathcal{S}$ the models above agree very well with the theoretical results derived in Appendix A.

(12) and (16) were found:

$$G_6(q) = \frac{4.88q - 0.09}{q(q - 0.29)}$$
$$G_7(q) = \frac{4.88q + 0.91}{q(q - 0.80)}.$$

Choosing Parameters

The span in behavior for the seven process models is large. It is therefore hard to find one parameter choice that will give good behavior for all cases. Since $G_1(q)$ is the most common and most important case, it is the case to start with.

Controlling $G_1(q)$ with the PI-controller results in a transfer function from $\log h_n$ to $\log r_n$ equal to

$$G(q) = \frac{G_{c_{PI}}(q)G_1(q)}{1 + G_{c_{PI}}(q)G_1(q)} = \frac{k(k_I + k_P)q - kk_P}{q^2 + q(-1 + k(k_I + k_P)) - kk_P} \quad \text{(C.1)}$$

For $k_I > 0$ and $k_P > 0$ the poles of (C.1) are real. One is positioned on the positive part of the real axis and the other on the negative part. The negative pole corresponds to an oscillating time signal and therefore its magnitude should preferably be small. Regarding the positive pole, it should not be placed to close to the unit circle since that leads to a very slow system. Hence the rejection of the varying disturbance ($\log \|\phi\|$) will be poor. On the other hand the system should not be made too fast (the poles positioned close to the origin) since that will make it sensitive to noisy fluctuations in $\log \|\phi\|$.

In Figure C.1 the magnitude of the largest closed loop pole is plotted for different values on k_P and k_I. In view of the discussion above it seems as choosing $k_I \in [0.05, 0.20]$ and $k_P \in [0.02, 0.15]$ would be appropriate. Note however, that a large value on k_P cannot be combined with a large value on k_I.

For each one of the remaining transfer functions ($G_2(q) - G_7(q)$), the closed loop poles where evaluated and plotted (root locus plots) when varying k_I and k_P. In Figure C.2 the root locus plot when controlling $G_4(q)$ is shown. The magnitude for the largest closed loop pole is

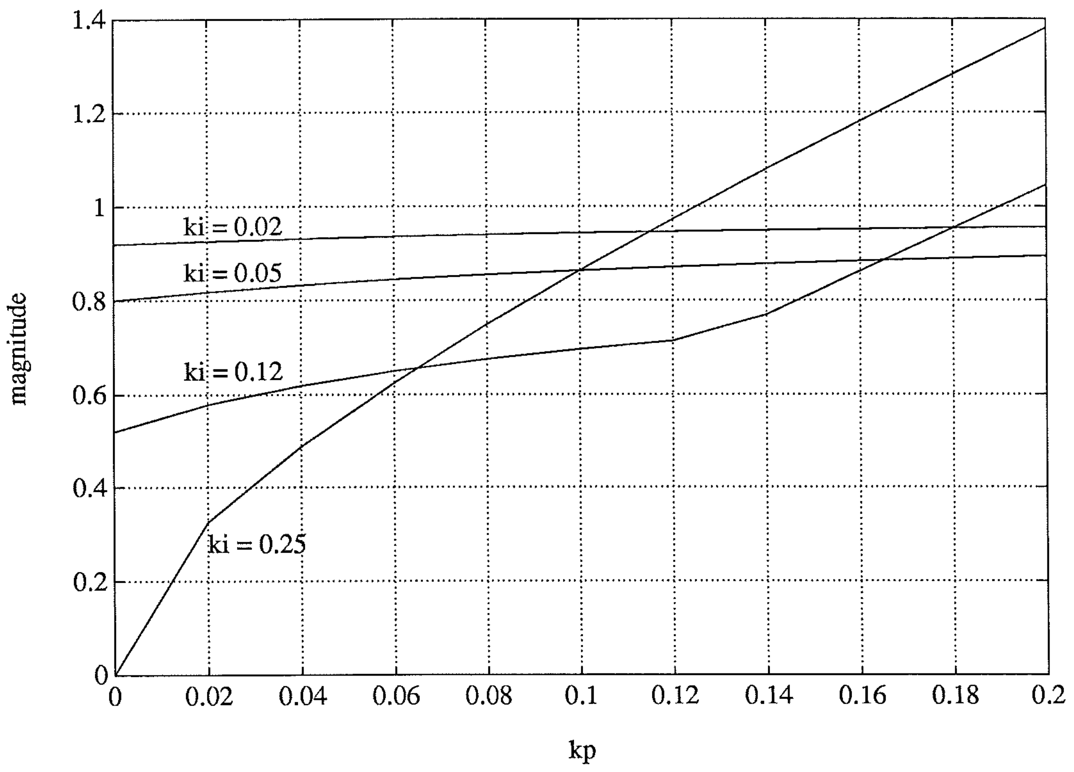

Figure C.1 Magnitude of largest closed loop pole when controlling $G_1(q)$ with the PI-controller.

plotted in Figure C.3. Figure C.2 and Figure C.3 advocate the choice $k_I \in [0.05, 0.12]$ and $k_P \in [0.06, 0.20]$.

By continuing like this, inspecting root locus plots for all the different processes, it was found that the choice

$$k_I = 0.06, \qquad k_P = 0.13$$

gives good overall performance.

Evaluation of the controller parameters

To compare the new controller with the standard one, the loop transfer function of the system was plotted as Nyquist plots (see Figure C.4 and Figure C.5). (For a discussion of Nyquist plots see e.g. [Franklin et al. 1986]).

In Figure C.5 some of the curves pass above the -1 point on the negative real axis, and hence the closed loop system is unstable for the corresponding process models. These cases gave rise to the oscillating

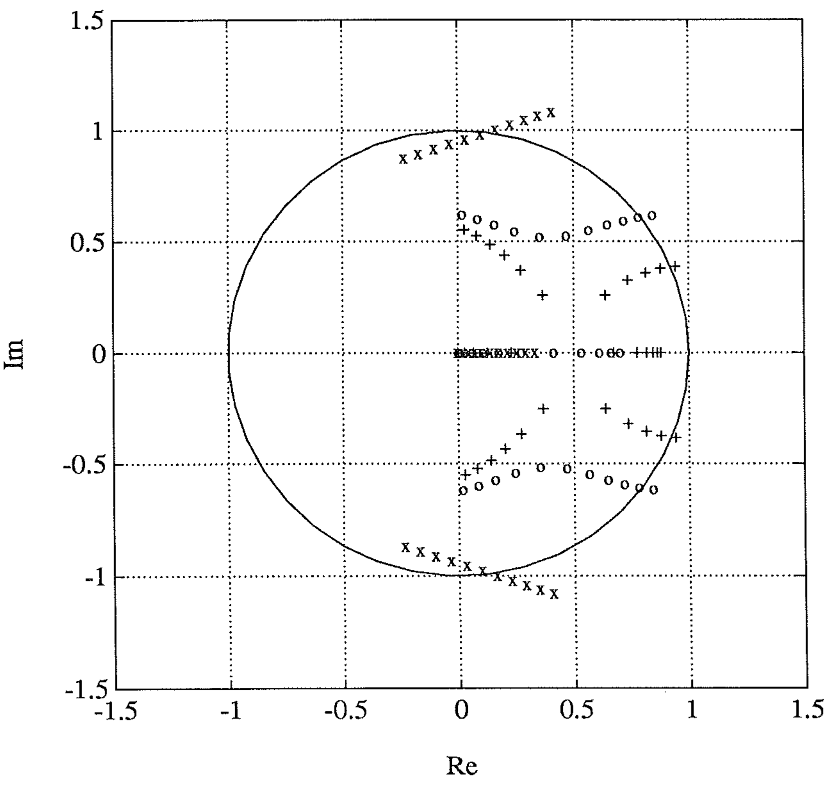

Figure C.2 Root locus for the case $G_4(q)$. k_P is varied from 0 to 0.2 for three values of k_I: $k_I = 0.25$ (x), $k_I = 0.12$ (o) and $k_I = 0.05$ (+).

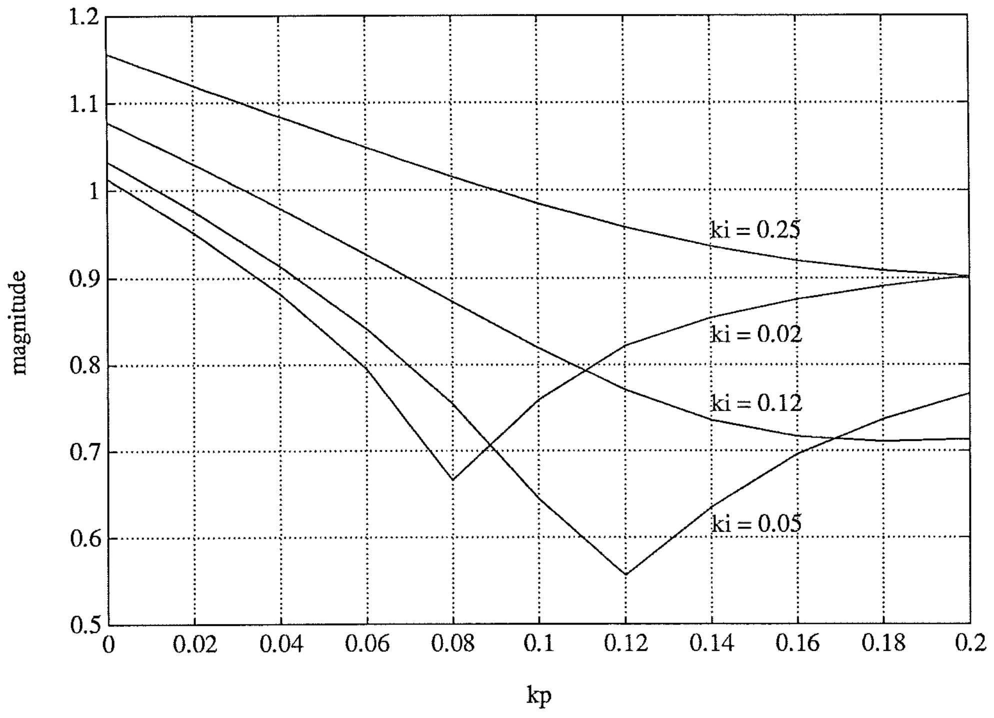

Figure C.3 Magnitude of largest closed loop pole when controlling $G_4(q)$ with the PI-controller.

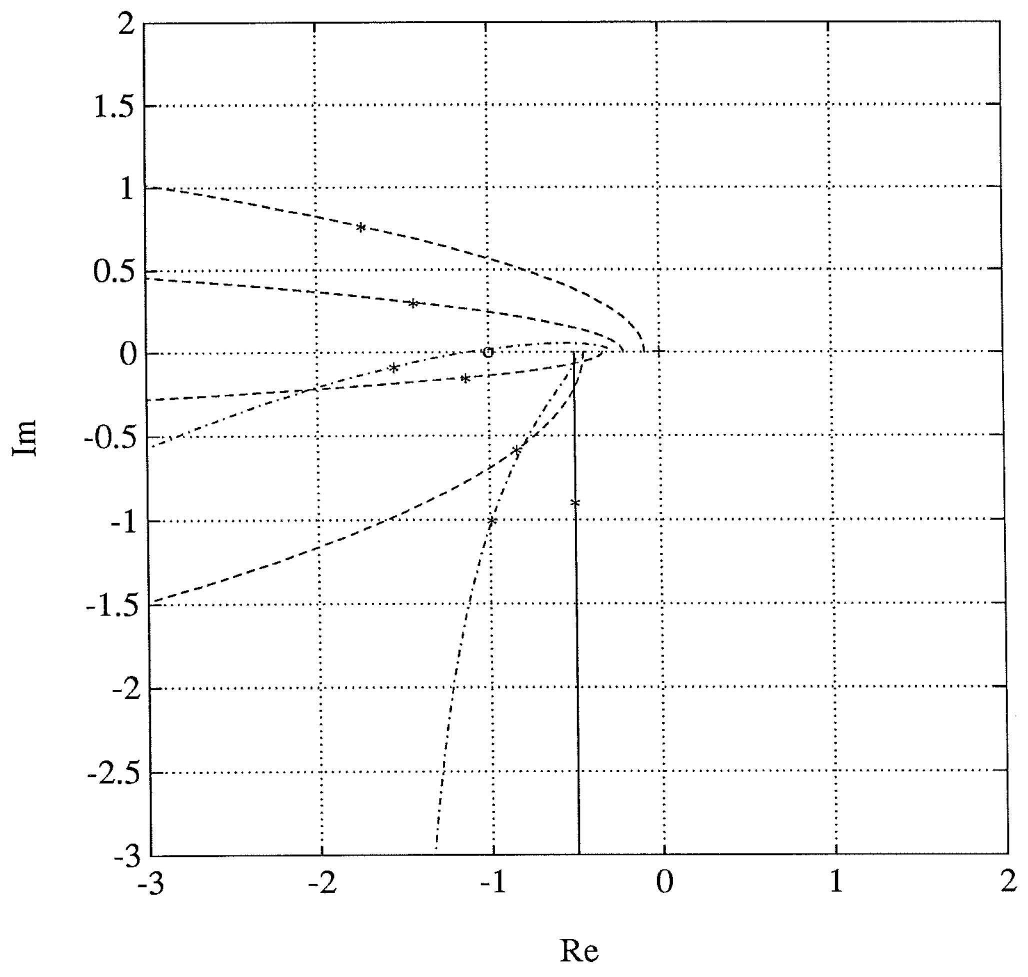

Figure C.4 Nyquist plot of loop transfer function for the process models G_1 - G_7 and the standard controller, $ki = 1/4$. The process in set 1 is plotted with a solid line, processes in set 2 with dashed lines and processes in set 3 with dash-dotted lines.

stepsize sequences among other things seen in some of the simulations in Section 7. When the PI-controller is used all curves pass below the -1 point, and the closed loop systems are all stable.

On each curve the point corresponding to a frequency of 1 is marked with an asterisk (*). Observe that for the new PI-controller this point is closer to the origin than for the old standard controller. This indicates that the loop gain is smaller when the PI-controller is used, and the result is a somewhat slower time response.

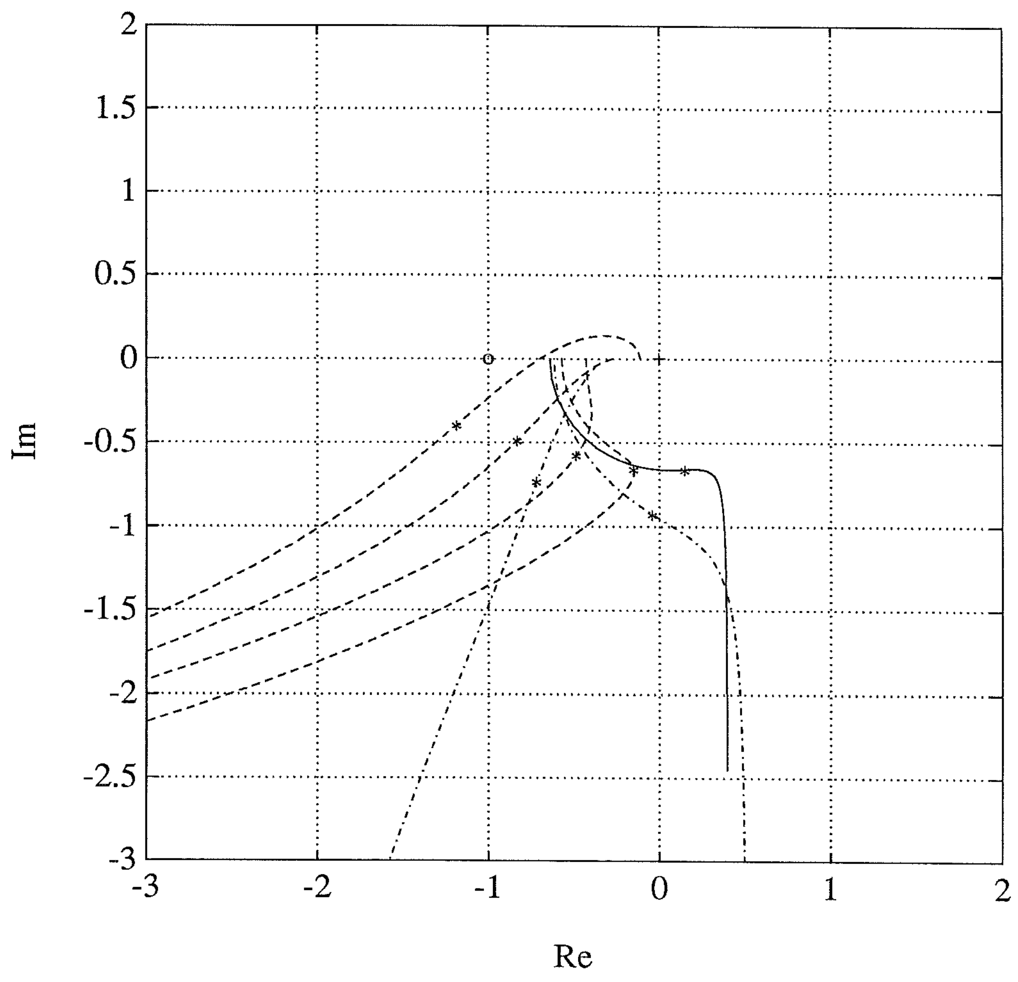

Figure C.5 Nyquist plot of loop transfer function for the process models $G_1 - G_7$ and the new controller, $ki = 0.06$ and $kp = 0.13$. The process in set 1 is plotted with a solid line, processes in set 2 with dashed lines and processes in set 3 with dash-dotted lines.

Appendix D: Problems

Problem 1 Linear, first order.

$$\dot{y}_1 = -y_1 + 1 \qquad y_1(0) = 1.1$$

Problem 2 Linear, second order with complex eigenvalues.

$$\begin{aligned} \dot{y}_1 &= -0.3y_1 - y_2 + 1.3 & y_1(0) &= 0.0 \\ \dot{y}_2 &= y_1 - 0.3y_2 - 0.7 & y_2(0) &= 0.0 \end{aligned}$$

Problem 3 Control system, PID-controller ($k = 0.87$, $T_i = 2.7$, $T_d = 0.69$ and $N = 30$) and a fourth order plant. The control signal (the output from the controller) is denoted y_{PID}.

$$\begin{aligned} \dot{y}_1 &= (1 - y_6)/T_i & y_1(0) &= 0.0 \\ \dot{y}_2 &= (-y_2 + y_6)N/T_d & y_2(0) &= 0.0 \\ y_{PID} &= k(1 - y_6 + y_1 + N(y_2 - y_6)) & & \\ \dot{y}_3 &= -y_3 + y_{PID} & y_3(0) &= 0.0 \\ \dot{y}_4 &= -y_4 + y_3 & y_4(0) &= 0.0 \\ \dot{y}_5 &= -y_5 + y_4 & y_5(0) &= 0.0 \\ \dot{y}_6 &= -y_6 + y_5 & y_6(0) &= 0.0 \end{aligned}$$

Problem 4 Problem D2 in [Enright et al. 1975].

$$\begin{aligned} \dot{y}_1 &= -0.04y_1 + 0.01y_2y_3 & y_1(0) &= 1.0 \\ \dot{y}_2 &= 400y_1 - 100y_2y_3 - 3000y_2^2 & y_2(0) &= 0.0 \\ \dot{y}_3 &= 30y_2^2 & y_3(0) &= 0.0 \end{aligned}$$

Problem 5 Brusselator with $\beta = 8.533$.

$$\begin{aligned} \dot{y}_1 &= 1.0 + y_1^2y_2 - (\beta + 1.0)y_1 & y_1(0) &= 1.3 \\ \dot{y}_2 &= \beta y_1 - y_1^2y_2 & y_2(0) &= \beta \end{aligned}$$

Problem 6 Van der Pol oscillator. Slightly changed version of Problem E2 in [Enright et al. 1975].

$$\begin{aligned} \dot{y}_1 &= y_2 & y_1(0) &= 2.0 \\ \dot{y}_2 &= 50(1 - y_1^2)y_2 - 10y_1 & y_2(0) &= 0.0 \end{aligned}$$

Lunds Universitet
Reprocentralen,1988