BIT 28 (1988), 270–287

A *PI* STEPSIZE CONTROL FOR THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

KJELL GUSTAFSSON[1], MICHAEL LUNDH[1] and GUSTAF SÖDERLIND[2]

[1] *Department of Automatic Control, Lund Institute of Technology, P.O. Box 118, S-221 00, Lund, Sweden*

[2] *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.

Subject classification: AMS 65L05.

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 [10]). 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

Received May 1987. Revised October 1987.

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 [4, 5].

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 ith 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 fulfil

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

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

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

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 ($\|r\|/h > \varrho \cdot tol$), 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.

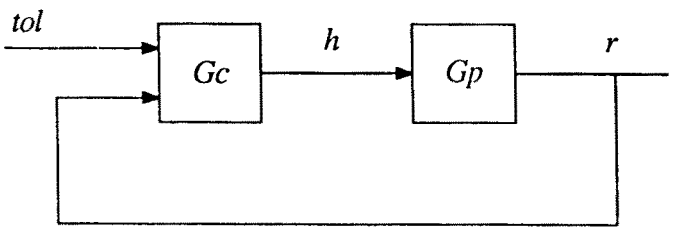

Fig. 1. Stepsize control viewed as an automatic control problem.

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. [3, 12]. Thus the controller can be expressed with the following equations:

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

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 [6, 7], and earlier in [11]. 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 [6] and complex [7] values of λ, respectively. More recently, these criteria have been used [8, 9] 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 [9]. Rather than constructing new numerical methods that go 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 [3]. P stands for *proportional* and I for *integral*. The output of such

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

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:

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

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 ($\|r\|/h > \varrho \cdot tol$). 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.

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

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

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

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
$K_P = 0.13$	$K_P = 0$
$K_I = 1/15$	$K_I = 1/5$
$\theta_{max} = 2.0$	$\theta_{max} = 2.0$
$\varrho = 1.2$	$\varrho = 1.2$

4. Numerical tests.

The integration method used in this paper is DOPRI45 [5], 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 [1], 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

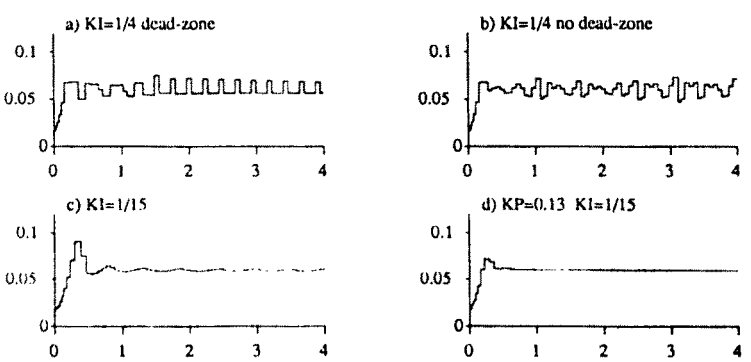

Fig. 2. The effect on stepsize of some of the differences between the old and the new algorithm (Problem 6).

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*.

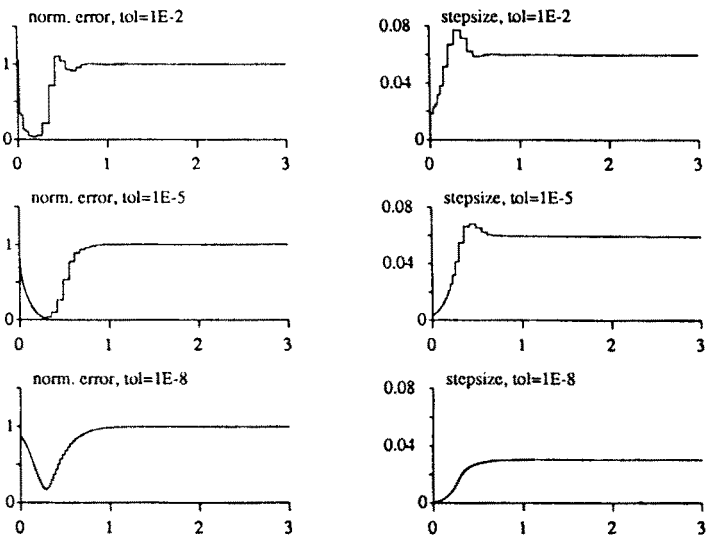

Fig. 3. Error estimate and stepsize for different tolerances (Problem 6). New controller.

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 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.

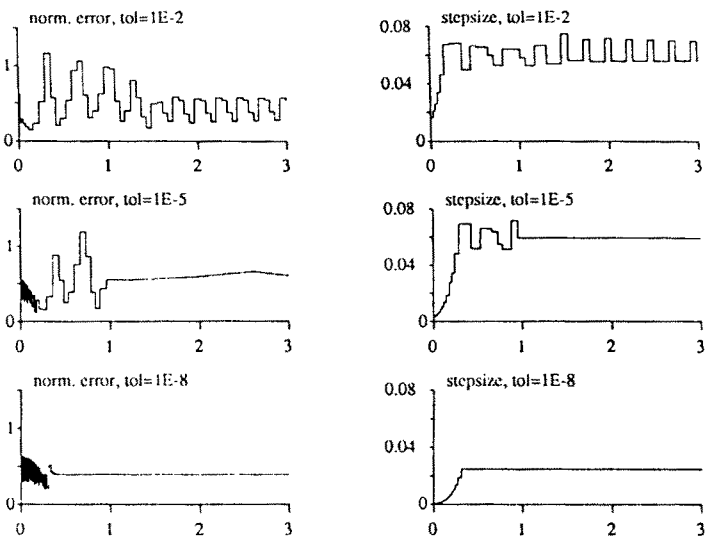

Fig. 4. Error estimate and stepsize for different tolerances (Problem 6). Old controller.

In Figure 5 we see the effect of incorporating a very small dead-zone ($\theta_{hi} = 1.02, \theta_{lo} = 0.995$) 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

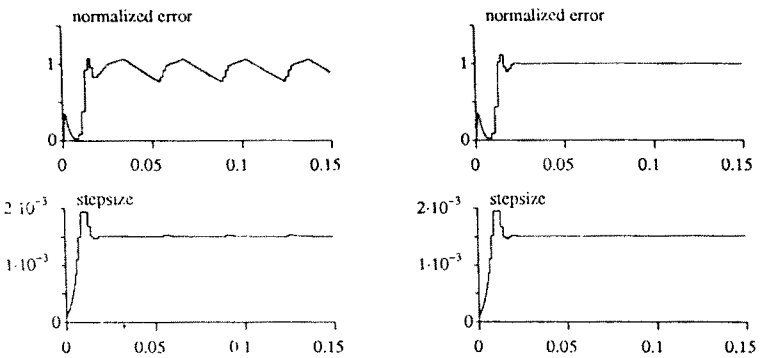

Fig. 5. New controller with (left) and without (right) dead-zone.

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

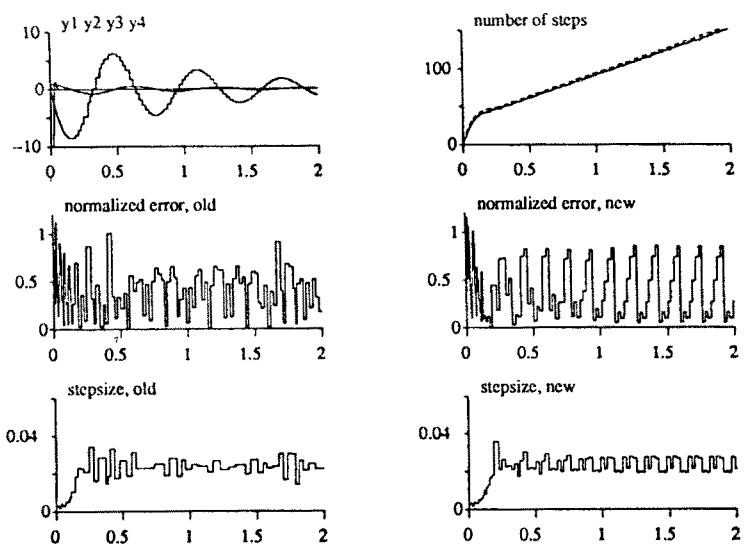

Fig. 6. Solving Problem 1, *tol* = 0.01.

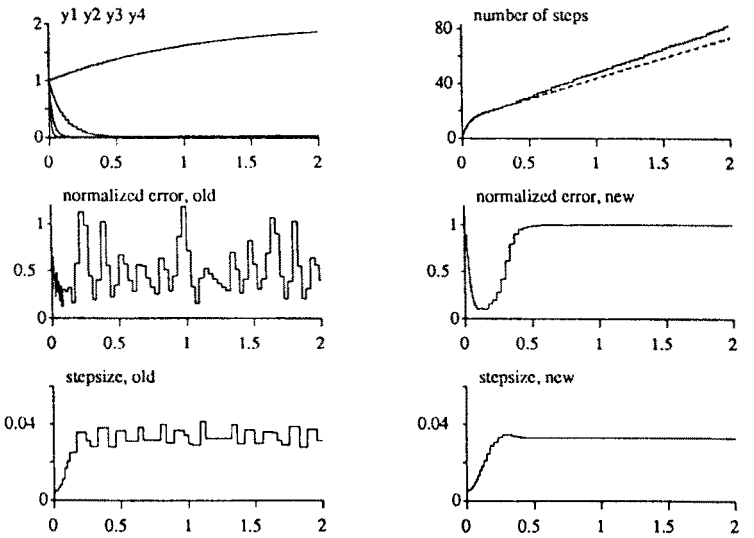

Fig. 7. Solving Problem 2, $tol = 0.01$.

where both controllers have some difficulties. Even for constant steps, the slowly damped oscillations would lead to fluctuating errors to which the controllers respond.

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

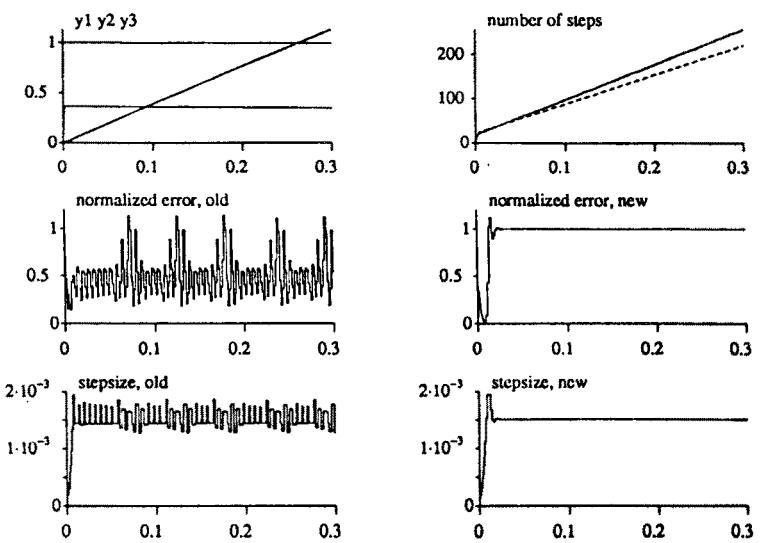

Fig. 8. Solving Problem 3, $tol = 0.01$.

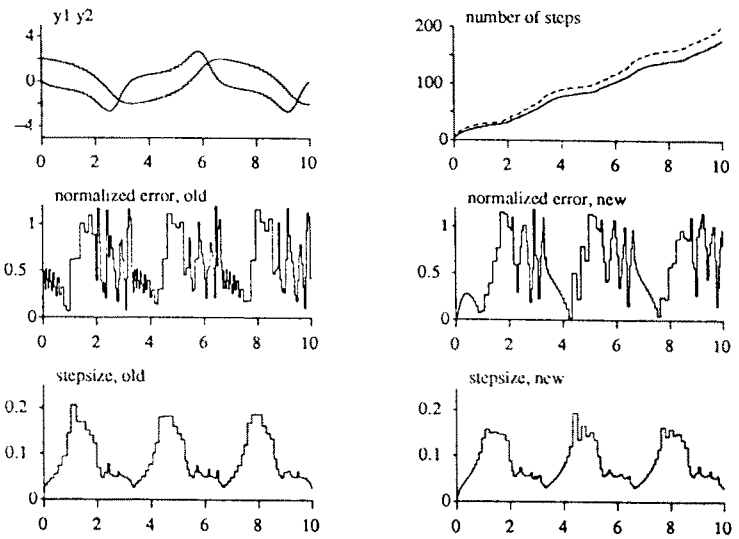

Fig. 9. Solving Problem 4, $tol = 10^{-6}$.

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 more efficient. In the latter, the new controller has a much smoother behavior,

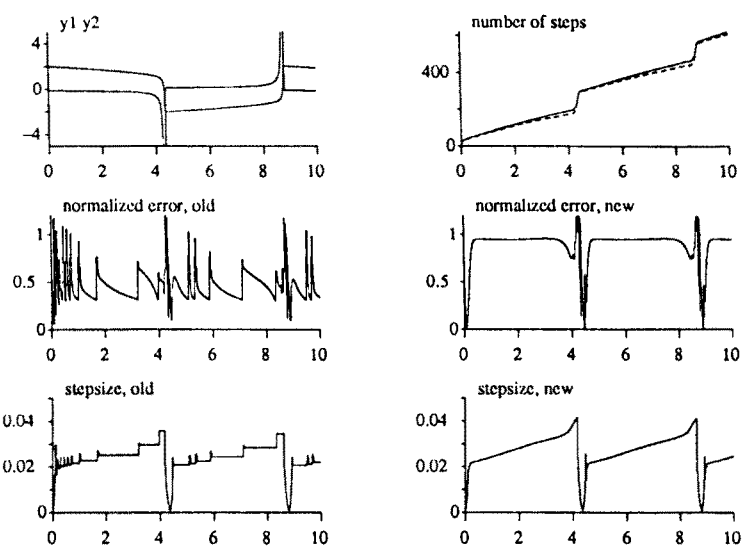

Fig. 10. Solving Problem 5, $tol = 10^{-4}$.

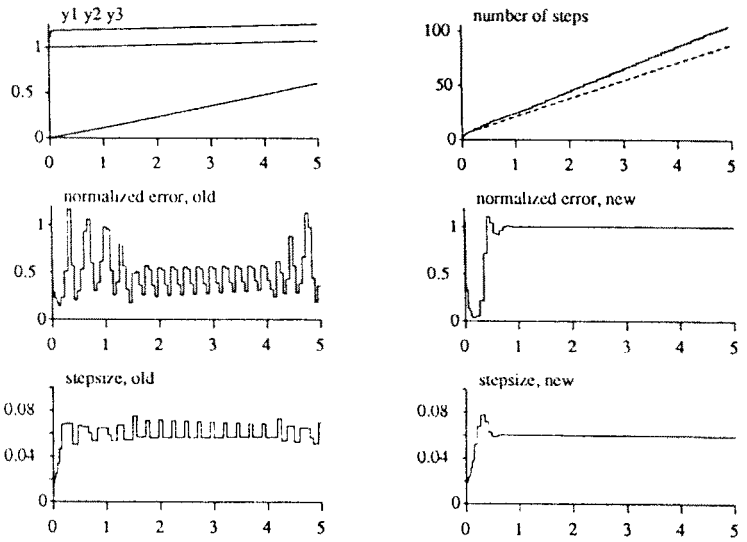

Fig. 11. Solving Problem 6, $tol = 0.01$.

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 7a and b are so-called "Brusselators", a type of nonlinear oscillating

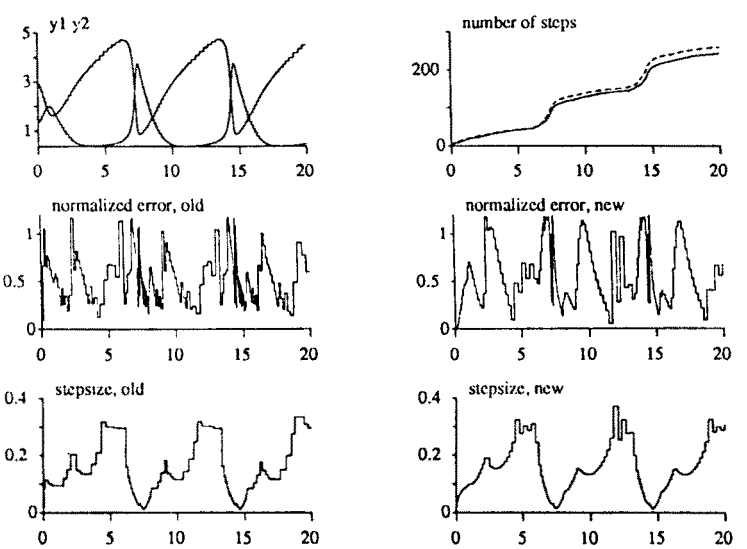

Fig. 12. Solving Problem 7a, $tol = 10^{-6}$.

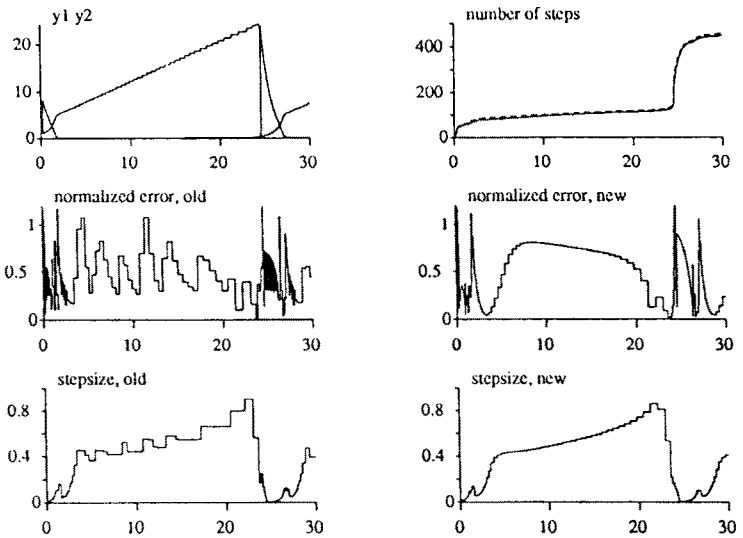

Fig. 13. Solving Problem 7b, $tol = 10^{-4}$.

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.

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.

Appendix.

Problem 1 Problem B1 in [2].

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

Problem 2 Problem C2 in [2] 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 [2].

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

Problem 4 Problem E2 in [2].

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

Problem 5 Problem E2 in [2] (slightly 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 6 Problem E3 in [2].

$$\begin{aligned} \dot{y}_1 &= -(55 + y_3)y_1 + 65y_2 & y_1(0) &= 1.0 \\ \dot{y}_2 &= 0.0785(y_1 - y_2) & y_2(0) &= 1.0 \\ \dot{y}_3 &= 0.1y_1 & y_3(0) &= 0.0 \end{aligned}$$

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

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

Note added in proof.

As is normally the case, the method DOPRI45 has been used in the numerical tests *with local extrapolation*, i.e. as a *fifth* order method. The parameter p in the

standard controller (called "the order of the method") should still have the value 4, since the embedded error estimate is of fifth order and the local truncation error per unit step is controlled. Unfortunately, the use of local extrapolation leads to a few similar inconsistencies in the way the term *order* is used in the paper. However, this fact does not change any results or conclusions of the paper.

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

1. H. Elmqvist, K. J. Åström and T. Schönthal, *SIMNON, User's Guide for MS-DOS Computers*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden (1986).
2. W. H. Enright, T. E. Hull and B. Lindberg, *Comparing numerical methods for stiff systems of ODE's*, BIT 15 (1975), 28–33.
3. G. F. Franklin, J. D. Powell and A. Emami-Naeini, *Feedback Control Systems*, Addison-Wesley (1986), pp. 99–103.
4. C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall (1971).
5. E. Hairer, S. P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations, I*, Springer (1987).
6. G. Hall, *Equilibrium states of Runge-Kutta schemes: Part I*, ACM Transactions on Mathematical Software, 11, 3 (1985), 289–301.
7. G. Hall, *Equilibrium states of Runge-Kutta schemes: Part II*, ACM Transactions on Mathematical Software, 12, 3 (1986), 183–192.
8. G. Hall and D. J. Higham, *Analysis of stepsize selection for Runge-Kutta codes*, NA report No. 137, University of Manchester (1987).
9. D. J. Higham and G. Hall, *Embedded Runge-Kutta formulae with stable equilibrium states*, NA report No. 140, University of Manchester (1987).
10. B. Lindberg, *Characterization of optimal stepsize sequences for methods for stiff differential equations*, SINUM, 14 (1977), 859–887.
11. L. F. Shampine, *Stiffness and nonstiff differential equation solvers*, in L. Collatz (Ed.): *Numerische Behandlung von Differentialgleichungen*, Information Series of Numerical Mathematics 27, Birkhäuser Verlag, Basel (1975), pp. 287–301.
12. K. J. Åström and B. Wittenmark, *Computer Controlled Systems*, Prentice-Hall, Englewood Cliffs, N.J. (1984), pp. 369–373.