CODEN: LUTFD2/(TFRT-7358)/1-15/(1987)

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

Kjell Gustafsson
Michael Lundh
Gustaf Söderlind

Department of Automatic Control
Lund Institute of Technology
May 1987

Department of Automatic Control **Lund Institute of Technology** P.O. Box 118 S-221 00 Lund Sweden	*Document name* Internal Report *Date of issue* May 1987 *Document Number* CODEN: LUTFD2/(TFRT-7358)/1-15/(1987)
Author(s) Kjell Gustafsson, Michael Lundh, Gustaf Söderlind	*Supervisor* *Sponsoring organisation*

Title and subtitle

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

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.

Key words

Numerical Integration, Stepsize Control

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title		*ISBN*
Language English	*Number of pages* 15	*Recipient's notes*
Security classification		

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

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

Kjell Gustafsson Michael Lundh
Department of Automatic Control
Lund Institute of Technology
Lund, Sweden

Gustaf Söderlind
Department of Computer Sciences
Lund University
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 dynamic simulation, ordinary differential equations are usually solved numerically. The user specifies the desired accuracy, and the integration method tries to find an approximate solution in accordance with this requirement.

To minimize the computational effort, the stepsize is adapted to the local smoothness of the solution. This is accomplished by an automatic stepsize control algorithm. Such algorithms usually adjust the stepsize to keep an estimate of the local truncation error per unit step at a constant level. This strategy is motivated by the fact that the global error can be bounded in terms of the local truncation error per unit step.

A good integration method should, while calculating a solution within the specified accuracy, try 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 parameter α reflects the cost of taking one step in the integration routine and β reflects the cost of changing the stepsize. Both α and β will be method and problem dependent. In some methods (e.g. explicit Runge-Kutta methods) changing the stepsize does not invoke extra computations, i.e. $\beta = 0$. In connection with stiff methods, a stepsize change may on the other hand require additional matrix factorizations, thus causing the second term of (1) to be significant.

Despite the fact that stepsize control is the most important means to make an integration method efficient, most stepsize control algorithms used today are quite simplistic. It seems that no effort has been devoted to using control theory for the design of such algorithms.

The paper is organized as follows. In Section 2, a typical stepsize control algorithm is analyzed from a control theory point of view. A new algorithm, based on a discrete PID controller is presented in Section 3. Section 4 contains comparative simulations for the two algorithms.

In this investigation we have limited ourselves to tuning the controller parameters for an explicit Runge-Kutta method. For such a method, the typical controller sometimes oscillates violently, in particular if there is a conflict between accuracy and numerical stability. Since this will happen in any stiff problem, most of our test problems are stiff. The new controller overcomes the oscillatory behavior and thus has much improved stability characteristics. It is likely that our algorithm will be superior also in stiff integration methods. This will, however, require a separate analysis which has not been pursued in this paper.

2. Present Stepsize Control Algorithm

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], [Hall 1985], [Hall 1986].

A Typical Stepsize Control Algorithm

The user specifies the desired accuracy of the solution by deciding an acceptable local error per unit step *tol*. For a method of order p the local error r depends on the stepsize h asymptotically as

$$r \sim h^{p+1} \tag{2}$$

The local error is often measured with the following 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

$$\frac{\|r\|}{h} = tol \tag{4}$$

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

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

where γ is a "safety factor" chosen less or equal to 1. A typical choice is $\gamma = 0.9$. To prevent many small stepsize changes a dead-zone is introduced. If θ is close to 1 no stepsize change is done. There is also a limit in how much the stepsize may increase in one step. Hence

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

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

If the error per unit step is too big in one step ($\|r\|/h > \rho \cdot tol$) the step is rejected, and it is recalculated with a new stepsize. A typical value of ρ is 1.2.

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

Analysis from a Control Theory Point of View

Regarding 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 a local error r as output. The controller G_c is the stepsize control algorithm. It tries to choose the stepsize such that the local error per unit step comes as close as possible to the prescribed tolerance.

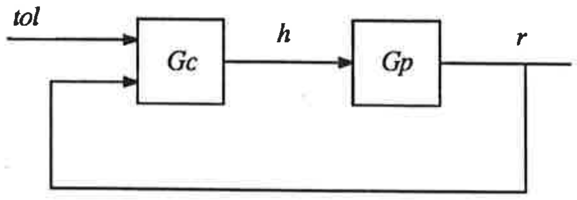

Figure 1. Stepsize control viewed as an automatic control problem.

The plant is nonlinear and time-varying. 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, the known part of the nonlinearity will turn into a constant gain, i.e. $\log\|r\| \sim (p+1)\log h$.

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

$$\log h_{n+1} = \log h_n + \frac{1}{p}\Big(\log(\gamma^p \cdot tol) - \log(\frac{\|r\|}{h_n})\Big) \tag{7}$$

Now we need to see how the dead-zone and the limitation of θ come in. When the dead-zone or the limitation is active it means invoking a different control signal than the calculated. The states in the controller must be updated to reflect this difference, or the controller may behave improperly. In control engineering this special update, when the control signal is limited, is refered to as anti-windup [Franklin et al 1986]. Thus the controller can be expressed with the following equations

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

We recognize $\log(\gamma^p \cdot tol)$ as the set point, i.e. the algorithm will try to make the local error per unit step as close as possible to $\log(\gamma^p \cdot tol)$ in order to bring the control error to zero.

In the algorithm the integration gain is chosen to $1/p$. This is a quite large value. It means that the local error per unit step will converge to *tol* very quickly (approximately in $2p$ steps). This is of course good, but may also be dangerous. In control theory it is well known that a pure integrating controller has quite poor stabilizing capabilities. This property is further accentuated if the integration time constant ($\approx p$) is small compared to the dominating time constant of the plant. As noted above instabilities can clearly be seen when applying the algorithm to certain problems.

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 violently [Hall 1985], [Hall 1986].

Roughly speaking, one can say that the purpose of the dead-zone is twofold: first to reduce the number of (small) stepsize changes, and second, to prevent oscillations. However, while the former objective is successfully achieved, the latter is not. In view of this, we shall propose a new control algorithm with improved stability characteristics.

3. A New Stepsize Control Algorithm

The plant G_p is nonlinear and time-varying. It will have different characteristics for each differential equation we try to solve. We have to find a controller that performs well for a large class of differential equations. The controller should preferably have a simple structure and not have too many parameters to tune. The pure integrating controller often performs quite well. We therefore choose to generalize this structure and suggest the use of a standard discrete PID controller [Franklin et al 1986]. $\log(tol)$ is regarded as set point, $\log(\|r\|)$ as plant output and $\log(h)$ as control signal.

The PID 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 PID controller is derived from the corresponding continuous time equivalent. Replace the integration with a summation, and the differentiation with a difference. As in continuous time PID controllers we filter the D-part. The filter is first order and has a pole at κ. Its gain is normalized to one for "high frequencies" ($q = -1$). Using the forward shift operator q we get the following expressions

$$\begin{aligned} e_n &= \log(tol) - \log\Big(\frac{\|r_n\|}{h_n}\Big) \\ P_n &= k e_n \\ I_n &= \frac{1}{T_I} \cdot \frac{1}{q-1} \cdot e_n \\ D_n &= T_D \cdot \frac{1+\kappa}{2(q-\kappa)} \cdot (q-1) \cdot e_n \\ h_{n+1} &= \exp(P_n + I_n + D_n) \end{aligned} \tag{9}$$

where $1/T_I$ is the integral gain and T_D is the differential gain.

Dead-zone on Stepsize Changes

The PID controller performs very well (see Section 4). It manages to control the local error per unit step perfectly, but at the price of many stepsize changes. The stepsize is changed almost every step. This is no problem if stepsize change is a computationally cheap operation. The local error per unit step is well controlled and the integration algorithm will always take as long steps as possible, thus reducing the amount of computation needed to calculate the solution.

In some integration algorithms the change of stepsize requires a lot of extra computation. In the old algorithm there is a dead-zone to prevent small stepsize changes. To really affect the number of stepsize changes the dead-zone had to be quite big ($\theta_{hi} = 1.2$). Such big dead-zones may cause trouble. When a stepsize change is accepted it will be quite large and thus cause a large transient in the local error per unit step. This transient may cause new stepsize changes, and the stepsize may even start to oscillate (see simulations in Section 4).

To prevent stepsize changes at every step, a dead-zone on stepsize changes is introduced in the new algorithm too. Small stepsize changes are required to be able to control the local error per unit step accurately (Note that this accuracy is needed not from the numerical point of view, but for the performance of the controller.). The dead-zone is therefore made much smaller than in the old algorithm. If the local error per unit step is too big the step will be rejected. Until the stepsize control algorithm finds a new stepsize that gives an acceptable error, there will be no update of the solution of the differential equation. The dead-zone is chosen unsymmetric to facilitate the finding of a new suitable stepsize in this situation, i.e. we are more willing to accept a decrease in stepsize than an increase. Even though the dead-zone is much smaller than the one used in the old algorithm, the number of stepsize changes will be moderate due to the better control supplied by the new algorithm.

To make the controller function properly with the dead-zone we need to include anti-windup in the integral part. The integral part will always try to bring the control error to zero. If the dead-zone prevents stepsize changes the integral part will continue to increase/decrease until a stepsize change is done. Thus each time a stepsize change is prevented we also adjust the integral part to compensate that.

As in the old algorithm we limit the stepsize increase in one step. This is done since there are integration methods (linear multistep methods) that may go unstable if the stepsize is increased too much in one step.

Rejected Steps

If the local error per unit step is too big, i.e. $\|r\|/h > \rho \cdot tol$, the step is rejected. Then the solution of the differential equation is not updated. Instead the stepsize control algorithm is called to get a new stepsize and a new try is made.

This situation may occur when there are sudden changes that call for a drastic decrease in stepsize. The new algorithm is designed to be more stable than the old one. This will also make it a little slower when following large transients. When a step has been rejected we will keep on rejecting steps until a stepsize that gives an acceptable local error per unit step is found.

Ideally one would like to have a controller that responds fast but still has very good stabilizing properties. However, these two properties are conflicting. Here we solve the problem by having two sets of parameters for the PID controller. The first set is chosen to optimize the stabilizing behavior of the algorithm. This set is used almost always. Parameter set two gives a faster response and is used when a step has been rejected. Typically, parameter set two is used in less than one percent of the calls to the stepsize control algorithm.

Complete Algorithm

Finally we state the complete control algorithm. The anti-windup, the limitation and the dead-zone has been incorporated. It has also been rewritten in an explicit form to facilitate coding.

$$\begin{aligned}
e_n &\leftarrow \log(tol) - \log\Big(\frac{\|r_n\|}{h_n}\Big) \\
P_n &\leftarrow k e_n \\
D_n &\leftarrow \kappa D_{n-1} + T_D \cdot \frac{1+\kappa}{2}(e_n - e_{n-1}) \\
h_{temp} &\leftarrow \exp(P_n + I_n + D_n) \\
h_{n+1} &\leftarrow \begin{cases} h_n, & \text{if } \theta_{lo} h_n \le h_{temp} \le \theta_{hi} h_n \\ \theta_{max} h_n, & \text{if } h_{temp} > \theta_{max} h_n \\ h_{temp}, & \text{otherwise} \end{cases} \\
I_{n+1} &\leftarrow I_n + \frac{e_n}{T_I} + \frac{1}{T_R}(\log h_{n+1} - \log h_{temp})
\end{aligned} \tag{10}$$

There are two sets of parameters used. One for normal behavior, and one with faster response to be used when a step has been rejected.

4. Simulations

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 1985], which gives a convenient way to change parameters in the routine. There are also good plotting facilities included in the package.

Repeated simulations suggested the following two parameter sets.

Set for normal case		Set for rejected case	
K	$= 0.2$	K	$= 0.2$
T_I	$= 25.0$	T_I	$= 5.0$
T_D	$= 0.08$	T_D	$= 0.0$
κ	$= 0.5$	κ	$= 0.0$
T_R	$= 1.0$	T_R	$= 1.0$
θ_{lo}	$= 0.995$	θ_{lo}	$= 1.0$
θ_{hi}	$= 1.020$	θ_{hi}	$= 1.0$
θ_{max}	$= 2.0$	θ_{max}	$= 2.0$
ρ	$= 1.2$	ρ	$= 1.2$

A first set of simulations solving problem 8 (all problems can be found in the Appendix) shows how some of the differences between the new and the old algorithm affects the stepsize. The stepsize for a pure integrating controller with dead-band (old strategy) is shown in Figure 2a. Figure 2b shows the stepsize when the dead-band is decreased to [0.995 1.02]. In Figure 2c the integral gain $1/T_I$ is changed from 1/4 to 1/25 (Note that in the original stepsize control, the integral 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.

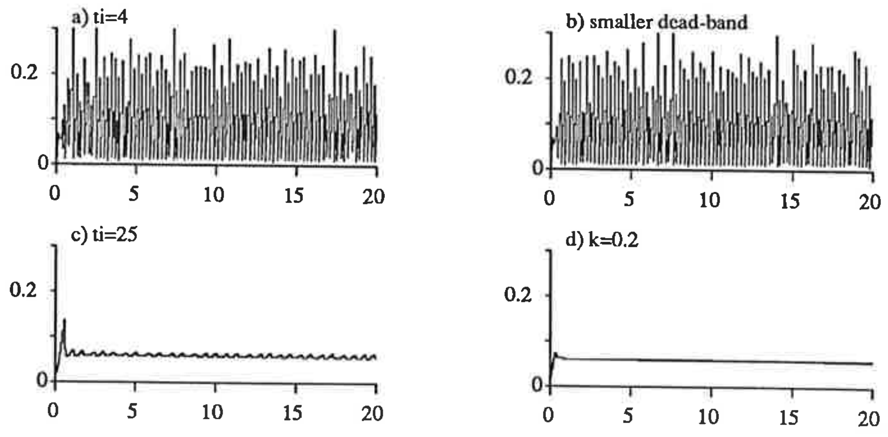

Figure 2. The effect on stepsize of some of the differences between the old and the new algorithm (problem 8).

The next simulation (Figure 3) shows that it is possible to drive the local error per unit-step to the desired value (*tol*). This is achieved by eliminating the dead-band. The stepsize will then be almost constant. However the cost for this performance is a change in stepsize every step. While this is perfectly acceptable for Runge-Kutta methods, it may not be desirable for multistep methods. The simulated example is problem 6.

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 4 – 11, with each figure consisting of six small plots. The upper left shows the solution of the differential equation. In the upper right four curves appear showing the cost for solving the

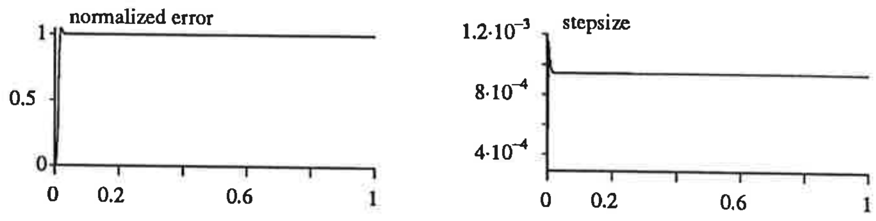

Figure 3. Local error and stepsize without dead-zone (problem 6).

differential equation. It is the number of integration routine calls for the old (1) and for the new (2) method, and the number of stepsize changes for the old (3) and for the new (4) method. The two plots in the middle show the local error per unit step ($\|r\|/h_n$) for the old (left) and 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. Notice the few stepsize changes for the new controller, and that it quickly finds the proper stepsize for the problem.

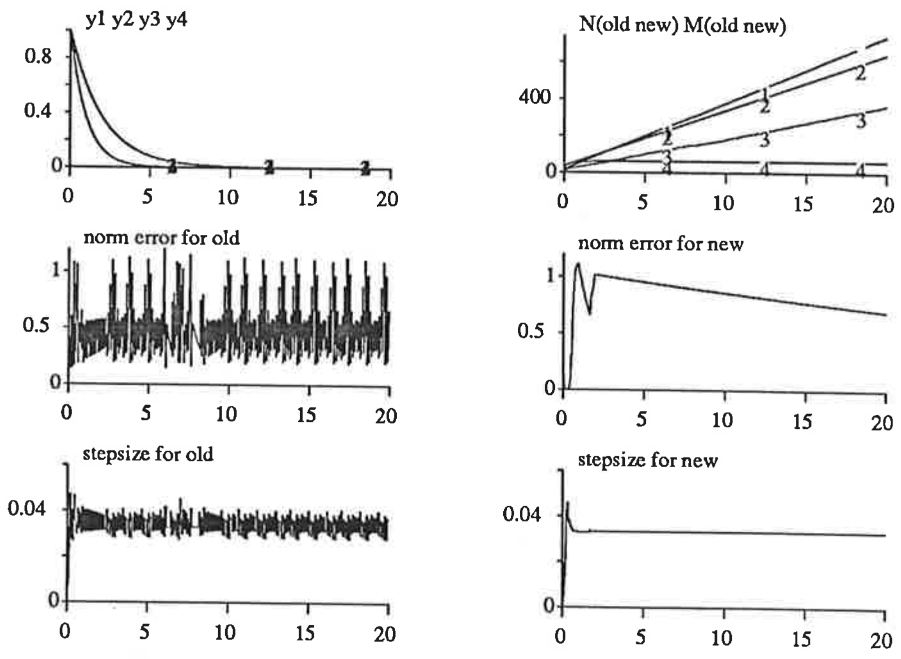

Figure 4. Solving problem 1.

Next, problem 2 is solved. Here, both control algorithms have some difficulties. This is probably due to the slowly damped oscillations in the solution. Even for constant steps, this would lead to fluctuating errors. However, the new method solves the problem using fewer steps.

The nonlinear problem 3 is solved with less computation in the new algorithm. Here, too, the PID controller finds the maximum stable stepsize and stays there without oscillations.

The same is true for problem 4. After the transient, the stepsize is held at an almost constant level.

In problem 5, we see an example of a problem where the dead-zone of the old controller manages to prevent stepsize oscillations in a short interval.

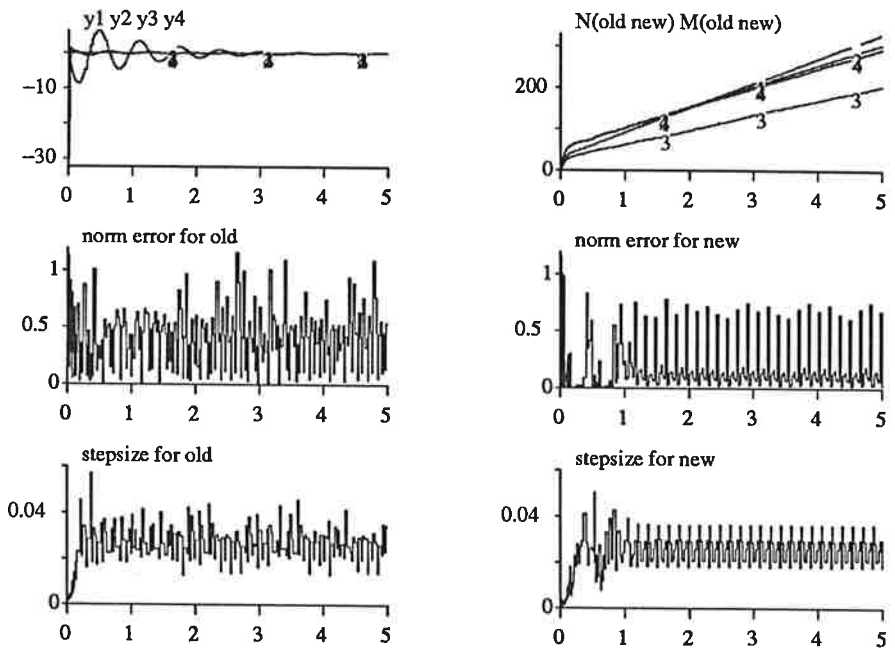

Figure 5. Solving problem 2.

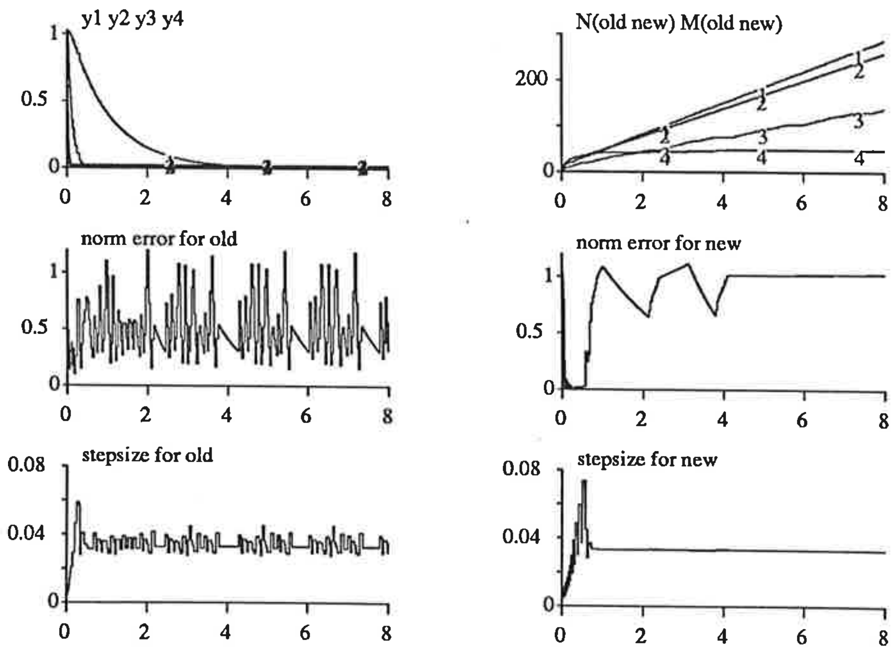

Figure 6. Solving problem 3.

This is mainly due to the extraordinary smoothness of the solution. The PID controller is again using the maximum stable stepsize.

Problem 6 is another typical example, showing the superior stabilizing effect of the PID controller. Also note the oscillations in the error for this problem. Even the most minute stepsize changes will cause error growth or decay. This is a sign of the stepsize being limited by numerical stability and that the algorithm is using the maximum stable stepsize without the closed loop system becoming unstable. This is also seen in problem 8.

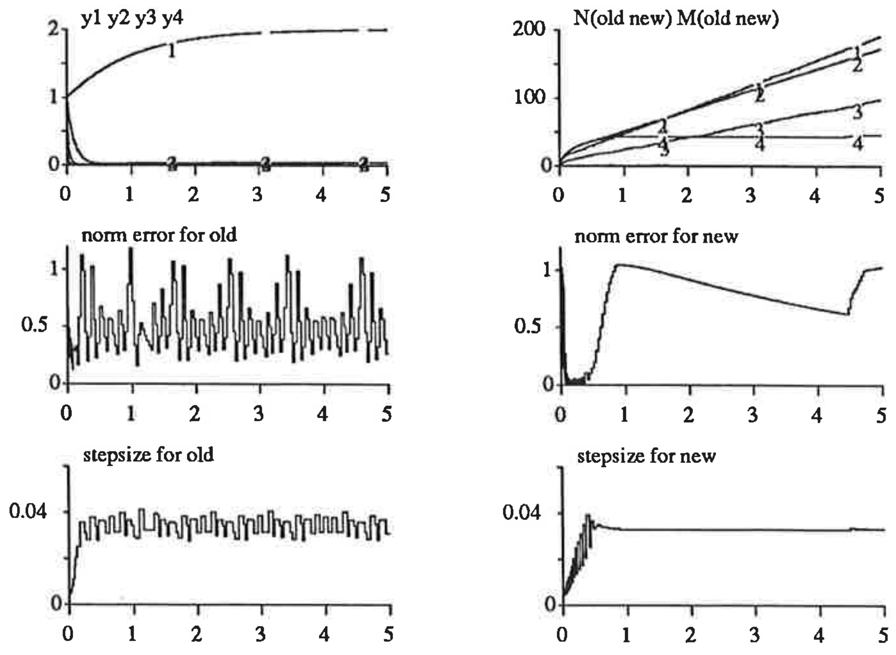

Figure 7. Solving problem 4.

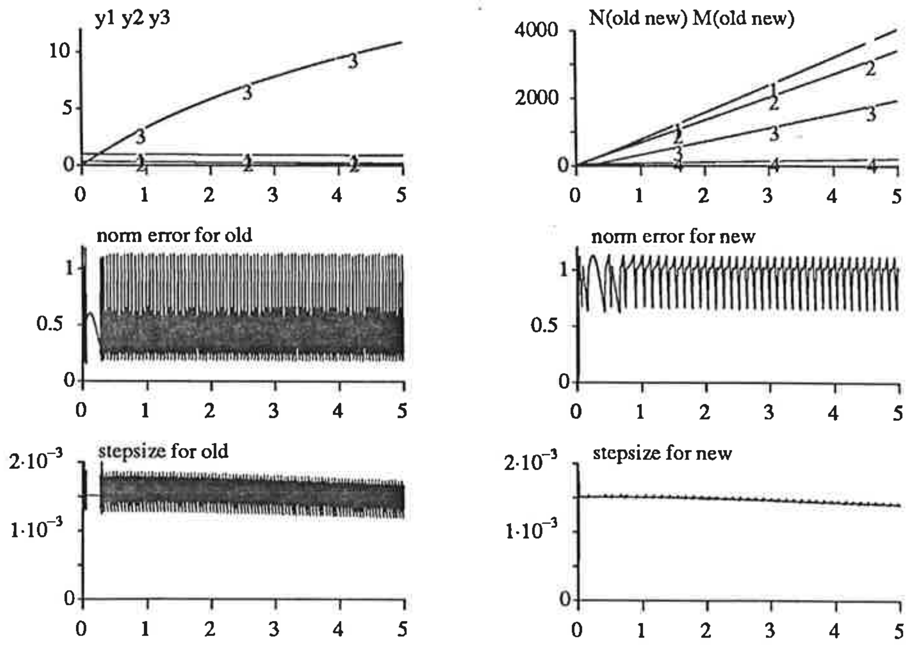

Figure 8. Solving problem 5.

The nonlinear oscillator in problem 7 has piecewise smooth solutions (in the stiff intervals) and the old controller performs quite well. However, the new algorithm, with its smaller stepsize changes, is even better although at a slightly higher expense.

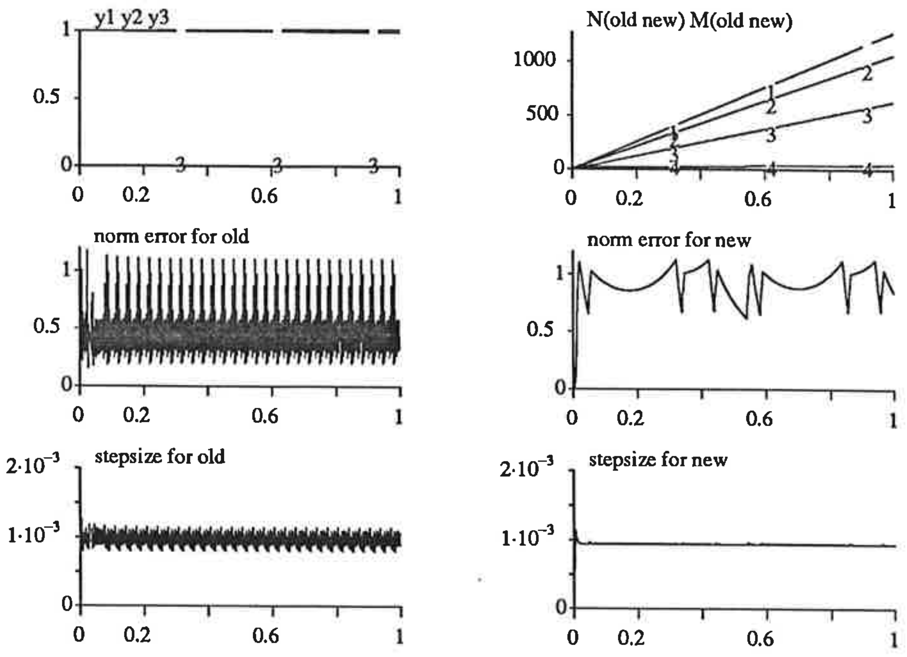

Figure 9. Solving problem 6.

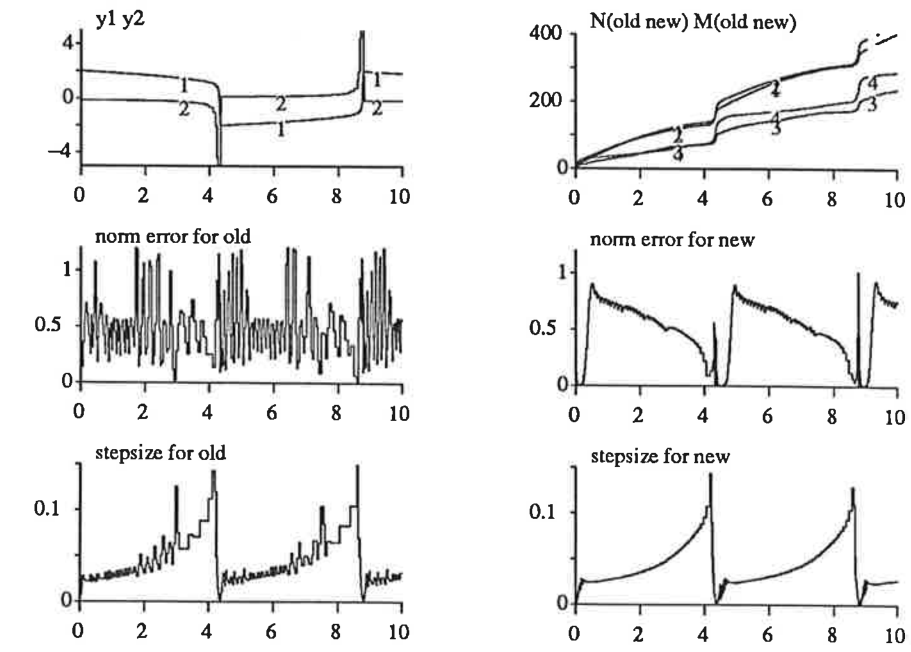

Figure 10. Solving problem 7.

5. Conclusions

By using standard control theory much insight and understanding of the stepsize control problem can be gained. The analysis of the standard control algorithm (Section 2) explains way it sometimes results in oscillating stepsize. A remedy is to use a standard PID algorithm as control algorithm. It gives significantly better results and more consistent performance. The addition of a proportional part increases the stabilizing properties drastically, while the

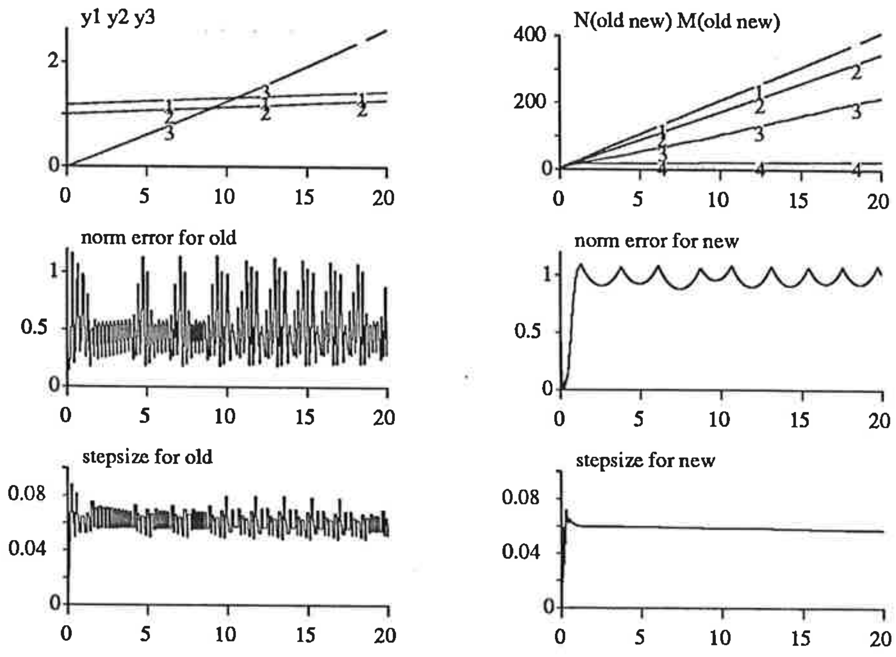

Figure 11. Solving problem 8.

derivative part seems to be especially important when having a dead-zone.

A dead-zone on the stepsize change should not be incorporated thoughtlessly. If no extra computation is needed to change stepsize there should not be any dead-zones. The cheapness of stepsize change should be used to try to get as large steps as possible without violating the prescribed *tol*. If changing the stepsize invokes extra computation there is a trade-off. The dead-zone reduces the number of stepsize changes but the stepsize will now on the average be smaller. For each type of integration method the dead-zone and the parameters in the controller need to be tuned to minimize (1).

Since we have used a Runge-Kutta method in our simulations there should not have been a dead-zone on stepsize changes. Still, one has been used to more clearly demonstrate the properties of the control algorithm.

In the tests we have seen that the PID controller gives much improved stability characteristics. 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.

6. Acknowledgements

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

7. 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.

Appendix

Problem 1 Problem A1 in [Enright et al 1975].

$$\begin{aligned} \dot{y}_1 &= -0.5y_1 & y_1(0) &= 1 \\ \dot{y}_2 &= -y_2 & y_2(0) &= 1 \\ \dot{y}_3 &= -100y_3 & y_3(0) &= 1 \\ \dot{y}_4 &= -90y_4 & y_4(0) &= 1 \end{aligned}$$

Problem 2 Problem B1 in [Enright et al 1975].

$$\begin{aligned} \dot{y}_1 &= -y_1 + y_2 & y_1(0) &= 1 \\ \dot{y}_2 &= -100y_1 - y_2 & y_2(0) &= 0 \\ \dot{y}_3 &= -100y_3 + y_4 & y_3(0) &= 1 \\ \dot{y}_4 &= -10000y_3 - 100y_4 & y_4(0) &= 0 \end{aligned}$$

Problem 3 Problem C1 in [Enright et al 1975].

$$\begin{aligned} \dot{y}_1 &= -y_1 + y_2^2 + y_3^2 + y_4^2 & y_1(0) &= 1 \\ \dot{y}_2 &= -10y_2 + 10(y_3^2 + y_4^2) & y_2(0) &= 1 \\ \dot{y}_3 &= -40y_3 + 40y_4^2 & y_3(0) &= 1 \\ \dot{y}_4 &= -100y_4 + 2 & y_4(0) &= 1 \end{aligned}$$

Problem 4 Problem C2 in [Enright et al 1975] with $\beta = 0.1$.

$$\begin{aligned} \dot{y}_1 &= -y_1 + 2 & y_1(0) &= 1 \\ \dot{y}_2 &= -10y_2 + \beta y_1^2 & y_2(0) &= 1 \\ \dot{y}_3 &= -40y_3 + 4\beta \cdot (y_1^2 + y_2^2) & y_3(0) &= 1 \\ \dot{y}_4 &= -100y_4 + 10\beta \cdot (y_1^2 + y_2^2 + y_3^2) & y_4(0) &= 1 \end{aligned}$$

Problem 5 Problem D2 in [Enright et al 1975].

$$\begin{aligned} \dot{y}_1 &= -0.04y_1 + 0.01y_2y_3 & y_1(0) &= 1 \\ \dot{y}_2 &= 400y_1 - 100y_2y_3 - 3000y_2^2 & y_2(0) &= 0 \\ \dot{y}_3 &= 30y_2^2 & y_3(0) &= 0 \end{aligned}$$

Problem 6 Problem D4 in [Enright et al 1975].

$$\begin{aligned} \dot{y}_1 &= -0.013y_1 - 1000y_1y_3 & y_1(0) &= 1 \\ \dot{y}_2 &= -2500y_2y_3 & y_2(0) &= 1 \\ \dot{y}_3 &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3 & y_3(0) &= 0 \end{aligned}$$

Problem 7 Problem E2 in [Enright et al 1975] (sligthly changed).

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

Problem 8 Problem E3 in [Enright et al 1975].

$$\begin{aligned} \dot{y}_1 &= -(55 + y_3)y_1 + 65y_2 & y_1(0) &= 1 \\ \dot{y}_2 &= 0.0785(y_1 - y_2) & y_2(0) &= 1 \\ \dot{y}_3 &= 0.1y_1 & y_3(0) &= 0 \end{aligned}$$