


REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

AD-A281 551

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE Feb. 1994	3. REPORT TYPE AND DATES COVERED	
4. TITLE AND SUBTITLE A Stepsize Control Strategy For Stiff Systems Of Ordinary Differential Equations			5. FUNDING NUMBERS DAAL03-92-G-0247	
6. AUTHOR(S) Peter K. Moore & Linda R. Petzold				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Peter K. Moore, Dept. of Mathematics, T-lane Univ., New Orelans LA 70118 Linda R. Petzold, Dept. of Computer Science, Univ. of Minnesota, Mpls MN 55455			8. PERFORMING ORGANIZATION REPORT NUMBER 94-08	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709-2211			SPONSORING/MONITORING AGENCY REPORT NUMBER ARO 29850.2-MA	
11. SUPPLEMENTARY NOTES The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) In solving stiff systems of ordinary differential equations using BDF methods, Jacobians needed for quasi-Newton iteration are frequently computed using finite differences. Round-off errors in the finite-difference approximation can lead to Newton failures forcing the code to choose its time steps based on "stability" rather than accuracy considerations. When standard stepsize control is used the code can experience thrashing which increases the total number of time steps, Jacobian evaluations, & function evaluations. In this paper we investigate this situation, explaining some surprising time step selection behavior produced by the standard control mechanism. A new control mechanism is proposed which attempts to find & use a "stability" stepsize. A comparison of the new strategy with the standard strategy & with two PI controllers introduced earlier is made using the stiff test set.				
14. SUBJECT TERMS		2288 94-21064		15. NUMBER OF PAGES 20
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

SDTC ELECTE JUL 12 1994 G D

04 2 11 062

**Computer Science Department
University of Minnesota
Twin Cities
4-192 EE/CSci Building
200 Union Street S.E.
Minneapolis, MN 55455**

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

**A Stepsize Control Strategy
for Stiff Systems of Ordinary
Differential Equations**

by
**by Peter K. Moore and Linda R.
Petzold**

**TR 94-08
1994
Technical Report**

**A STEPSIZE CONTROL STRATEGY FOR STIFF
SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS**

Peter K. Moore

Department of Mathematics, Tulane University, New Orleans, Louisiana 70118,

and

Linda R. Petzold *

*Department of Computer Science, University of Minnesota, Minneapolis, Minnesota
55455*

Abstract. In solving stiff systems of ordinary differential equations using BDF methods, Jacobians needed for quasi-Newton iteration are frequently computed using finite differences. Round-off errors in the finite-difference approximation can lead to Newton failures forcing the code to choose its time steps based on "stability" rather than accuracy considerations. When standard stepsize control is used the code can experience thrashing which increases the total number of time steps, Jacobian evaluations, and function evaluations. In this paper we investigate this situation, explaining some surprising time step selection behavior produced by the standard control mechanism. A new control mechanism is proposed which attempts to find and use a "stability" stepsize. A comparison of the new strategy with the standard strategy and with two PI controllers introduced earlier is made using the stiff test set.

1. Introduction.

In solving stiff systems of ordinary differential equations using BDF methods, Jacobians which are needed for the quasi-Newton iteration are often approximated

* The work of this author was partially supported by ARO Contract Number DAAL03-89-C-0038 with the U. of Minn. AHPCRC, and by ARO Contract Number DAAL03-92-G-0247.

using finite differences [1, 2, 9]. Smaller stepsizes than allowed by accuracy considerations may be needed to guarantee convergence of the Newton iteration due to round-off errors in the finite-difference Jacobians. The standard stepsize control mechanism, such as that used in DASSL [2], is

$$h_n = (TOL/EST_{n-1})^{1/p} h_{n-1}, \quad (1.1)$$

where TOL is the user-prescribed tolerance, EST_{n-1} is the local error estimate computed at time t_{n-1} , and p is the method order. However, (1.1) is based solely on accuracy considerations. This can lead to highly oscillatory stepsize behavior (see Figure 1.1).

Here we apply DASSL to the stiff test set [4, 5] with an approximate Jacobian to simulate the effects of an inaccurate finite difference Jacobian. The stepsize behavior shown in Figure 1.1 when DASSL is applied to problem D2 is typical. After periods of taking relatively small stepsizes the algorithm suddenly increases the stepsize by several orders of magnitude. It remains at this larger stepsize for several time steps, and then decreases the stepsize dramatically whereupon the process begins again. Although most of the time steps taken by DASSL are of the smaller size, the solution on most of the time interval is found using the larger time steps.

In §2 we analyze the behavior of DASSL on a simple linear system which leads to an understanding of the stepsize behavior discussed above. We also present a modification of the time step selection strategy used in DASSL based on the quasi-Newton algorithm of Dennis and Schnabel [3]. The revised strategy prevents the larger "anomalous" steps but leads to a much larger number of time steps with no significant improvement in accuracy. The stepsize controller of Gustafsson et al. [6], referred to henceforth as the PI controller I, is presented in §3 with a new interpretation. This controller was developed for explicit time integrators. Gustafsson [8]

developed a PI controller (referred to herein as PI controller II) for implicit methods. We also discuss this controller in §3. In §4 we present a new control strategy and compare it with the standard stepsize controller and the PI controllers on the stiff test set [4, 5]. Brief conclusions are given in §5.

2. Analysis of a stiff system.

Consider first the standard test problem

$$y' = \lambda y, \quad y(0) = 1 \quad (2.1)$$

where $Re(\lambda) \leq 0$. Applying the backward-Euler method together with quasi-Newton iteration yields

$$(1 - \alpha\lambda h)\Delta y_{n+1}^i = -y_{n+1}^{i-1}(1 - \lambda h) + y_n, \quad i \geq 1. \quad (2.2)$$

Typically, if an analytic Jacobian is used $\alpha = 1$. To model the effects of an inaccurate matrix approximation, we choose α different from 1. After $k+1$ iterations of the quasi-Newton method we obtain

$$y_{n+1}^{k+1} = y_{n+1}^0 \frac{(\alpha - 1)^{k+1}(-\lambda h)^{k+1}}{(1 - \alpha\lambda h)^{k+1}} + \frac{y_n}{1 - \alpha\lambda h} \left[1 - \frac{(\alpha - 1)\lambda h}{1 - \alpha\lambda h} + \dots + \frac{(\alpha - 1)^k(-\lambda h)^k}{(1 - \alpha\lambda h)^k} \right], \quad (2.3)$$

where y_{n+1}^0 is the predicted solution. If $|\lambda h(\alpha - 1)/(1 - \alpha\lambda h)| < 1$ the quasi-Newton iterates converge to the true solution where the rate of convergence, ρ , is given by

$$\rho = \frac{|y_{n+1}^{k+1} - y_n/(1 - \lambda h)|}{|y_{n+1}^k - y_n/(1 - \lambda h)|} = |\lambda h(\alpha - 1)/(1 - \alpha\lambda h)|. \quad (2.4)$$

In DASSL [2] the quasi-Newton iteration is said to converge if

$$\frac{\rho}{1 - \rho} |y_{n+1}^{k+1} - y_{n+1}^k| < 0.33 \quad (2.5a)$$

where $\hat{\rho}$ is an approximation of the rate ρ given by

$$\hat{\rho} = (|y_{n+1}^{k+1} - y_{n+1}^k| / |y_{n+1}^1 - y_{n+1}^0|)^{1/k}. \quad (2.5b)$$

Thus the number of iterations before convergence is determined by both the accuracy of the predictor and the rate of convergence $\hat{\rho}$.

For systems of equations, the analysis is complicated by the norm used. Although DASSL uses a weighted rms norm, herein we consider the l^2 norm which displays the same type of behavior. Consider the diagonal system

$$y' = Dy, \quad y(0) = 1, \quad (2.6)$$

where $D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_m)$ with $\text{Re}(\lambda_i) < 0$, $i = 1, 2, \dots, m$. After two iterations the rate of convergence is given by

$$\frac{[(\rho_1(y_{n+1,1}^0 - y_{n,1}/(1 - \lambda_1 h)))^2 + \dots + (\rho_m(y_{n+1,m}^0 - y_{n,m}/(1 - \lambda_m h)))^2]^{1/2}}{[(y_{n+1,1}^0 - y_{n,1}/(1 - \lambda_1 h))^2 + \dots + (y_{n+1,m}^0 - y_{n,m}/(1 - \lambda_m h))^2]^{1/2}} \quad (2.7)$$

where the second subscript indicates the component of the vector y and ρ_i is given by (2.4) with the appropriate λ_i . Unlike (2.1) the rate of convergence of (2.7) is not constant, but is instead determined by the stiffness (through ρ_i) and the accuracy of the predictor $(y_{n+1,i}^0 - y_{n,i}/(1 - \lambda_i h))$. Thus if the stiff components are sufficiently more accurately predicted than the nonstiff components (which is likely since the stiff components change little from step-to-step) the rate of convergence will be controlled by the rate of convergence for the nonstiff components which have smaller rate constants. The stepsize controller may then wish to increase the stepsize (which may be low for the nonstiff components) until the errors in the stiff components are excited whereupon the stepsize undergoes a drastic reduction since the rate is now being determined by the stiff components.

That this actually happens can be seen by applying DASSL to problem A4 of the

stiff test set [5] which has $\lambda_i = -i^5$, $i = 1, 2, \dots, 10$. We solved this problem on $0 < t \leq 0.4$ with absolute and relative error tolerances of 0.01 and with $\alpha = 0.5$. Figure 2.1 shows the time steps and the error in the stiffest component over the interval. When the stiff component becomes sufficiently accurate (after very small time steps) the time step increases rapidly, reaching a value controlled by the error in nonstiff components. Such large time steps excite errors in the stiff components until the rate of convergence becomes dominated by the stiff components and the time step is drastically reduced, beginning the process again. Such behavior is also seen in non-diagonal systems as shown in Figure 1.1.

The question arises, is it desirable to permit the large stepsizes. We observed that on some successful steps at the larger stepsizes $\|g(t, Y, Y')\|$ increased where Y is the BDF solution when we are solving

$$g(t, y, y') = 0. \quad (2.8)$$

One approach we implemented to correct this problem was to insist that $\|g(t, Y, Y')\|$ decrease before a step was converged. Specifically we required that on each Newton step

$$\frac{\|g(Y + \Delta Y)\|^2 - \|g(Y)\|^2}{-2\|g(Y)\|^2} \geq 0.05 \quad (2.9)$$

(cf. Dennis and Schnabel [3]) where ΔY is the quasi-Newton direction. Although using (2.9) did reduce the number and extent of the large stepsize regions, more time steps were used with no significant improvement in accuracy. Thus, it seems reasonable to allow the large steps and subsequently we do not use (2.9).

3. PI control.

As was seen in §2 the standard stepsize control mechanism (1.1) leads to oscilla-

tory time steps. The difficulty is finding a way to smooth out the selection of small time steps without eliminating the selection of the large time steps. One possible approach is to use PI controller I introduced by Gustafsson et al., [6, 7, 8] for explicit methods or PI controller II of Gustafsson [8] for implicit methods. In this section we present a new interpretation of these strategies along with some modifications for use in our situation.

PI controller I can be written in the form (with some modifications for maximum rate of stepsize increase and decrease) [7]

$$h_n = (TOL/EST_{n-1})^{K_I} (EST_{n-2}/EST_{n-1})^{K_P} h_{n-1} \quad (3.1)$$

where EST_{n-2} and EST_{n-1} are estimates of the local truncation error at time t_{n-2} and t_{n-1} , respectively, and K_I and K_P are parameters whose values depend only on whether a step is successful or not. Values for K_I and K_P are given in [6] and [7] although they differ slightly. For our purposes it is important to note first that in the case of a rejected step $K_I = 1/p$ and $K_P = 0$. Thus, when a step is rejected the standard controller is used. Second, in the case of an accepted step $K_I + K_P = 1/p$.

We can rewrite (3.1) as

$$h_n = (EST_{n-2}/TOL)^{K_P} (TOL/EST_{n-1})^{(K_P + K_I)} h_{n-1} \quad (3.2)$$

Assuming $K_I + K_P = 1/p$ and using

$$h_{acc,n-1} = (TOL/EST_{n-1})^{1/p} h_{n-1} \quad (3.3a)$$

we obtain

$$h_n = (EST_{n-2}/TOL)^{K_P} h_{acc,n-1} \quad (3.3b)$$

where $h_{acc,n-1}$ represents the stepsize based on the local truncation error that could have been taken at time t_{n-2} . We note that $h_{acc,n-1}$ also represents the stepsize based

on accuracy that is typically used for the next time step $[t_{n-1}, t_n]$. Now, since (3.3a) holds at time t_{n-2} we obtain

$$h_n = (h_{n-2}/h_{acc,n-2})^{K_G} h_{acc,n-1} \quad (3.4)$$

where $K_G = K_P/(K_P + K_I)$. Thus the new stepsize is chosen to be the stepsize based on accuracy multiplied by a factor that represents the ratio of the actual stepsize to accuracy stepsize we could have chosen on the previous step. If on the last step the accuracy and actual time step were the same, the accuracy stepsize is used on the present step. Otherwise (the previous stepsize can never be larger than the previous stepsize based on accuracy), if the previous stepsize was much smaller than the previous stepsize based on accuracy only a fraction of the accuracy stepsize is used.

A similar analysis shows that PI controller II has the form

$$h_n = (h_{n-2}/h_{acc,n-2})^{K_G} (h_{n-1}/h_{n-2}) h_{acc,n-1} \quad (3.5)$$

if two or more successive accepted time steps have been taken (otherwise the standard controller is used). Now the accuracy time step $h_{acc,n-1}$ is multiplied by an additional factor representing the ratio of successive accepted time steps.

The version of the PI algorithms we used in our testing consists of three cases.

- 1: If the present step is rejected due to the error test, set $h_n = h_{acc,n-1}$, $K_G = K_{LO}$
- 2: If the present step is rejected due to Newton divergence, set $h_n = h_{n-1}/4$,
 $K_G = K_{HI}$
- 3: If the present step is accepted update $K_G = \max(\text{fac} * K_G, K_{LO})$ if the previous step was not a Newton failure. For PI controller I set

$$h_n = \min(1, (h_{n-2}/h_{acc,n-2})^{K_G}) h_{acc,n-1} \text{ and set}$$

$$h_n = \min(1, (h_{n-2}/h_{acc,n-2})^{K_G} (h_{n-1}/h_{n-2})) h_{acc,n-1} \text{ for PI controller II.}$$

where $\text{fac} = 0.9$, $K_{LO} = 0.5$, and $K_{HI} = 0.7$ are fixed parameters. This differs slightly from the approach taken by Gustafsson [6, 8], since we allow K_G to vary. We found varying K_G resulted in slightly better performance over fixed K_G .

Gustafsson [8] offers some additional ideas for PI controller II. He has a stepsize algorithm for the case of successive stepsize failure due to error control. In our situation, however, we encounter successive stepsize failure due to divergence in Newton's method so we did not incorporate this heuristic in our algorithm. In certain cases of Newton divergence he computes a second, "stability" stepsize which is based on the size of the norm of the Jacobian. Since we are also interested in solving differential-algebraic equations, we are very reluctant to use a scale-dependent quantity in our algorithm so we have neglected this feature.

4. A new controller.

For reasons that will become clear from the examples in this section neither the standard controller nor the PI controllers possess the desired stepsize behavior. Using the analysis from §2 we present a new controller which we refer to as the STAB controller. We then present a numerical comparison of the the standard, PI, and STAB controllers applied to the stiff test set [4, 5].

From our observations in §2 we desire a controller that interferes with the standard controller as little as possible. Our goal was to smooth out the time step selection strategy only when the code is thrashing due to Newton convergence difficulties. When the code is able to use larger stepsizes because of a good predictor for the stiff components, we want to let it do this because this is where it makes most of its progress. Additionally, as indicated in §2, no accuracy is lost in accepting the large steps.

We begin with the observation that there are two important time step sizes, an accuracy size h_{acc} and a stability size h_{stab} , where here, stability refers to the convergence of Newton's method. Normally h_{acc} is smaller than h_{stab} for BDF methods but when the Jacobian is poorly approximated the reverse can occur. In the graph on the right side of Figure 1.1, the peaks in curve A indicate stepsizes chosen by accuracy but which caused the quasi-Newton iteration to diverge. Thus the best guess at h_{acc} is represented by the peaks, although it may be quite a bit larger (curve C on the right, Figure 1.1). After each peak two successful, smaller steps are taken. The value h_{stab} is approximated by curve B in Figure 1.1. Our controller seeks to detect when h_{stab} is smaller than h_{acc} and then makes two attempts at finding h_{stab} . The algorithm then limits the time step size for 10 steps to $0.87h_{stab}$ after which it reverts to the standard controller. The value 0.87 is chosen to reduce the number of step failures and to reduce the number of Newton iterates required for convergence. If the time step used was h_{stab} , Newton may take several steps to converge due to a larger rate and a poorer predictor.

The new controller is invoked only when the Newton iteration fails to converge (as long as the convergence is not due to a singular Jacobian), i.e., when the criteria (2.5a) fails and when the last successful time step h_{n-1} is smaller than the first failed (Newton) step h_n . Two attempts are made at finding h_{stab} . After the first Newton failure, if $h_{n-1}/h_n \geq 0.8$, $h_n = 0.87h_{n-1}$ and the time step is not allowed to become larger than this value for 10 time steps (of course it can become smaller due to subsequent Newton failures or error failure). If $h_{n-1}/h_n < 0.8$ then $h_n = 0.8h_{n-1} + 0.2h_n$. Now, however, the stepsize is allowed to increase, but at a reduced maximum rate of 1.18. If Newton fails for a second (but not second consecutive) time within the 10 step limit, $h_n = 0.87h_{n-1}$ and no increase above this value is allowed for 10 steps.

Consecutive Newton failures result in the algorithm reverting to the standard controller since we no longer seem to have a good approximation to h_{stab} . We limit our controller to 10 steps so that larger stepsize increases are allowed from time to time which should preserve the desirable property of the standard controller.

We solved the stiff set using DASSL with the three controllers and absolute and relative error tolerances of 10^{-k} , $k = 2, 3, \dots, 6$ and $\alpha = 0.5$. Table 4.1 contains the number of time steps used (including successful and unsuccessful time steps) by each of the three algorithms with tolerances of 0.0001. None of the algorithms was able to solve F1 or F4 in 10000 time steps and only PI controller I was able to do so for F5 with this poor approximation to the iteration matrix. In almost all cases the STAB controller outperforms the standard controller. For problems A2, D3, and D5 where the standard controller appears to outperform the STAB controller, the error produced using the STAB controller was at least a factor of three smaller. Also note the anomalous behavior of the standard controller on F3. PI controller II is slightly better than PI controller I. The STAB controller is more efficient than the PI controllers for a significant number of the test problems. Although PI controller II appears to perform better than the STAB controller on the early problems A2, A3, and A4, the STAB controller produced solutions with errors at least a factor of 10 smaller than PI controller II. On problems D4 and D5 the STAB controller is more accurate by a factor of at least 2.5. The results at this tolerance were typical of the performance of the control algorithms at the other tolerances.

In Table 4.2 we have listed the number of Jacobian evaluations, function evaluations, and errors (in the l^2 norm) for several cases. The STAB controller is clearly more efficient in almost every case. For several problems it uses a factor of 6 fewer Jacobian evaluations and less than half the number of function evaluations while

obtaining a smaller error. Even in the cases where it uses more function evaluations and time steps, such as D4, its solution is more accurate. PI controller II is generally superior to PI controller I. The PI controllers have, in general, larger errors and are often between the STAB and standard controllers in the other measures.

Figure 4.1 presents the time steps used in the standard controller and the STAB controller for $0 < t \leq 8$ with tolerances of 0.00001 and $\alpha = 0.5$. Clearly the STAB controller produces a smoother time step history than the standard controller.

The only problem which presented any difficulty to DASSL with the standard controller and $\alpha = 1$ was F5. We solved F5 with the standard, the STAB, and the PI controllers with $\alpha = 1$ and with both analytic and finite difference Jacobians for tolerances of 10^{-k} , $k = 2, 3, \dots, 6$. The results shown in Table 4.3a (analytic Jacobians) and Table 4.3b (finite-difference Jacobians) indicate that the STAB controller enhances the performance of DASSL in both cases.

5. Conclusions.

When finite-difference approximations to the Jacobian are used in stiff solvers such as DASSL, thrashing of the time step can occur because the accuracy stepsize exceeds the stepsize required for convergence of the quasi-Newton iteration. After damping the stiff components at small stepsizes, such algorithms using the standard stepsize control mechanism are able to take larger time steps based on the error in the nonstiff components. However, the stiff components then become excited resulting in drastic decreases in the stepsize. We analyzed a simple linear system to explain this stepsize phenomena. To smooth out the smaller stepsizes we tried modified versions of two PI controllers proposed by Gustafsson et al. [6, 7, 8] for explicit Runge-Kutta methods and implicit Runge-Kutta methods, respectively, which we reinterpreted to our

situation. A new controller was also proposed which attempts to find and use a "stability" stepsize. In comparing the four controllers on the stiff test set with altered Jacobian we found the new controller superior in almost every case. In fact for a number of cases it offers dramatic improvement over the standard and the PI controllers.

Although most of the test results of §4 were somewhat artificial the new controller does produce significantly better results especially in terms of the number of Jacobian evaluations. This continued to be true even when analytic and finite difference Jacobians were used for problem F5. Smoothing out the smaller stepsizes also seems to be beneficial for the error. More testing needs to be done to verify the usefulness of the algorithm in a wider setting, when used to solve partial differential equations by the method-of-lines and differential-algebraic equations.

References

1. M. Berzins, R.M. Furzeland, and P.M. Dew, Software tools for time-dependent differential equations, in *Simulation and Optimization of Large Systems*, A.J. Osiadacz, Ed., Oxford University Press, (1988).
2. K.E. Brenan, S.L. Campbell, and L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, North Holland, New York, 1989.
3. J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, 1983.
4. W.H. Enright and T.E. Hull, Comparing numerical methods for the solution of stiff systems of ODEs arising in chemistry, in *Numerical Methods for Differential Systems, Recent Developments in Algorithms, Software and Applications*, L. Lapidus and W.E. Schiesser, Eds., Academic Press, New York, (1976), 45-66.
5. W.H. Enright, T.E. Hull and B. Lindberg, Comparing numerical methods for stiff systems of ODEs, *BIT* 15 (1975), 10-48.
6. K. Gustafsson, M. Lundh, and G. Soderlind, A PI stepsize control for the numerical solution of ordinary differential equations, *BIT* 28 (1988), 270-287.
7. K. Gustafsson, Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods, *ACM Trans. Math. Soft.* 17 (1991), 533-554.

8. K. Gustafsson, Control of error and convergence in ODE solvers, Ph.D. Thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1992.
9. A.C. Hindmarsh, ODEPACK, A systematized collection of ODE solvers, in *Scientific Computing*, R.S. Stepleman et al., Eds., North Holland, Amsterdam, (1983), 55-64.

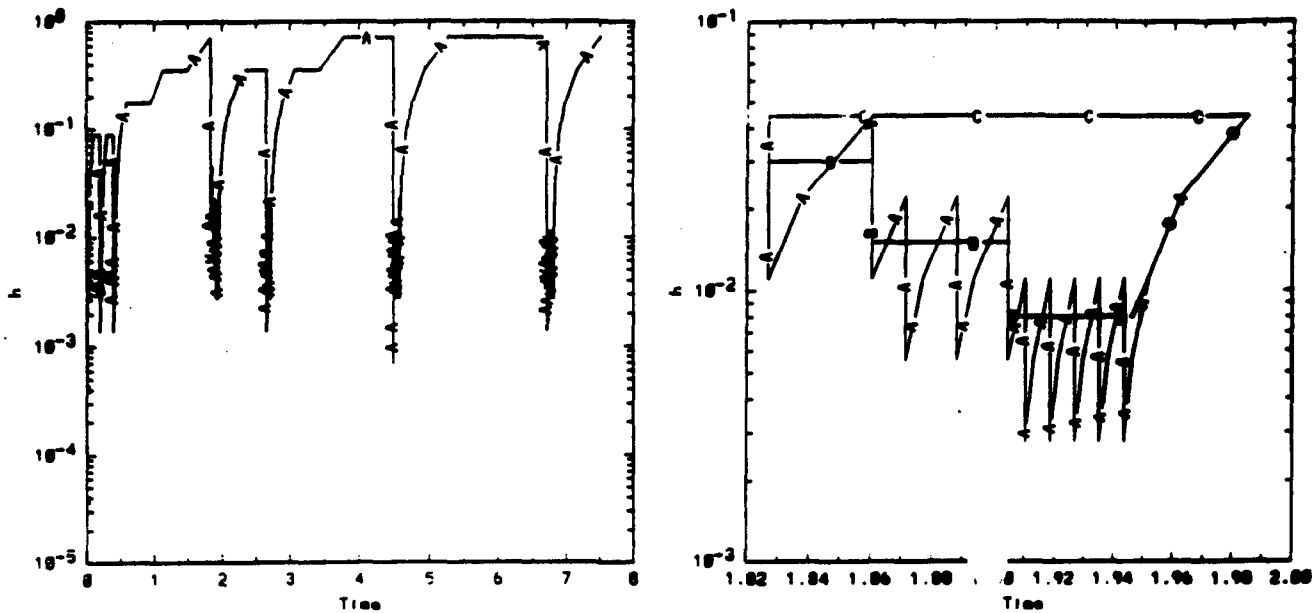


Figure 1.1. Time steps used in solving problem D2 from the stiff test set [5] on $0 < t \leq 8$ (left) and on $1.83 \leq t \leq 1.98$ (right denoted by A) using DASSL with absolute and relative error tolerances of 0.01 and $\alpha = 0.5$. Estimates of the stepsize based on "stability", h_{stab} (right, denoted by B) and accuracy, h_{acc} (right, denoted by C).

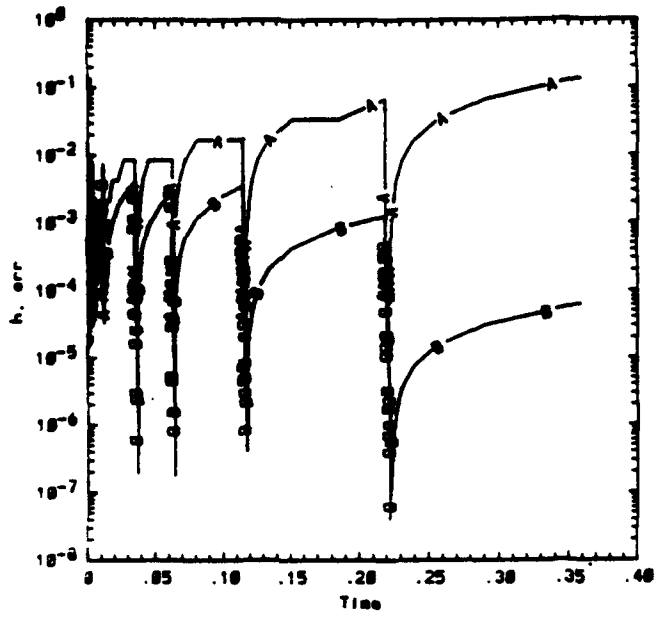


Figure 2.1. Time steps (A) and error in y_{10} (B) in solving A4 from the stiff test set [5] on $0 < t \leq 0.4$ with DASSL with absolute and relative error tolerances of 0.01 and $\alpha = 0.5$.

Stiff Set Problem	Number of Time Steps			
	Standard Controller	STAB Controller	PI Controller I	PI Controller II
A1	367	217	286	343
A2	1546	1627	1990	984
A3	2446	1467	1946	746
A4	1469	1417	1201	805
B1	1154	892	2578	2372
B2	110	106	120	106
B3	145	120	132	127
B4	277	184	360	328
B5	1633	1071	1413	1334
C1	347	238	153	200
C2	328	242	258	274
C3	322	152	257	303
C4	348	281	420	383
C5	272	233	393	358
D1	2099	621	1928	2023
D2	2767	1576	2408	1810
D3	376	449	358	364
D4	154	497	99	106
D5	144	182	131	107
D6	517	604	476	510
E1	62	40	73	73
E2	156	156	871	768
E3	2121	1063	1973	1635
E4	1298	972	1339	1270
E5	19	19	19	19
F2	159	135	112	138
F3	15002	152	15002	15002

Table 4.1. The number of time steps needed by DASSL with the standard controller, the STAB controller, and the PI controllers in solving the problems of the stiff test set [4, 5] with tolerances 0.0001 and $\alpha = 0.5$. None of the three algorithms solved F1 or F4 in fewer than 10000 steps and only PI controller I was able to solve F5 in fewer than 10000 steps.

Stiff Set Problem	TOL	Standard Controller	STAB Controller	PI Controller I	PI Controller II	
A2	10 ⁻⁵	3349 8649 3595 0.42 × 10 ⁻⁵	697 6423 2632 0.11 × 10 ⁻⁶	2361 7182 2918 0.68 × 10 ⁻⁵	1030 3632 1483 0.14 × 10 ⁻⁴	JACS FNS STEPS l ² ERROR
	10 ⁻⁶	8241 22570 8630 0.19 × 10 ⁻⁵	1077 10237 4110 0.54 × 10 ⁻⁷	2679 10037 3958 0.25 × 10 ⁻⁵	1117 6218 2742 0.46 × 10 ⁻⁵	JACS FNS STEPS l ² ERROR
A3	10 ⁻⁵	6002 15458 4375 0.36 × 10 ⁻²	905 8554 2933 0.43 × 10 ⁻³	3666 11065 4312 0.35 × 10 ⁻²	492 3079 1375 0.62 × 10 ⁻²	JACS FNS STEPS l ² ERROR
	10 ⁻⁶	9901 25928 10595 0.96 × 10 ⁻³	1207 11786 4594 0.14 × 10 ⁻³	1906 10087 4109 0.20 × 10 ⁻²	2883 10608 4384 0.21 × 10 ⁻²	JACS FNS STEPS l ² ERROR
D2	10 ⁻⁵	5223 13226 5377 0.11 × 10 ⁻¹	704 6431 2607 0.12 × 10 ⁻²	3092 10014 3788 0.12 × 10 ⁻¹	2988 8863 3420 0.20 × 10 ⁻¹	JACS FNS STEPS l ² ERROR
	10 ⁻⁶	9123 24408 9403 0.28 × 10 ⁻²	967 9806 3761 0.29 × 10 ⁻³	4519 18768 6762 0.44 × 10 ⁻²	7224 20259 7554 0.51 × 10 ⁻²	JACS FNS STEPS l ² ERROR
D4	10 ⁻⁵	488 1129 506 0.19 × 10 ⁻³	434 3520 1538 0.12 × 10 ⁻³	1029 2500 1066 0.64 × 10 ⁻³	311 824 341 0.10 × 10 ⁻²	JACS FNS STEPS l ² ERROR
	10 ⁻⁶	1998 4859 2129 0.18 × 10 ⁻³	589 5353 2330 0.35 × 10 ⁻⁴	209 1058 482 0.30 × 10 ⁻³	166 856 389 0.30 × 10 ⁻³	JACS FNS STEPS l ² ERROR
E3	10 ⁻⁶	5403 14102 5609 0.21 × 10 ⁻⁴	521 5905 2120 0.28 × 10 ⁻⁴	3393 12103 4293 0.31 × 10 ⁻³	4599 12439 4873 0.23 × 10 ⁻³	JACS FNS STEPS l ² ERROR

Table 4.2. The number of Jacobian evaluations (JACS), function evaluations (FNS), time steps (STEPS), and error in the l² norm for selected problems from the stiff test set [5] using DASSL with standard, STAB, and PI controllers.

TOL	Analytic Jacobian				
	Standard Controller	STAB Controller	PI Controller I	PI Controller II	
10^{-2}	49	28	51	49	JACS
	82	83	86	82	FNS
	49	44	52	49	STEPS
10^{-3}	39	51	84	39	JACS
	71	170	142	71	FNS
	39	87	85	39	STEPS
10^{-4}	356	100	313	421	JACS
	637	314	545	687	FNS
	370	145	329	430	STEPS
10^{-5}	2575	119	2137	1175	JACS
	4138	435	3495	2035	FNS
	2604	195	2180	1200	STEPS
10^{-6}	9296	3547	12576	12211	JACS
	14798	12166	20006	19950	FNS
	9339	5675	12710	12326	STEPS

Table 4.3a. The number of Jacobian evaluations (JACS), function evaluations (FNS), and time steps (STEPS) needed to solve problem F5 from the stiff test set with DASSL and analytic Jacobians using the standard, STAB, and PI controllers.

TOL	Finite-Difference Jacobian				
	Standard Controller	STAB Controller	PI Controller I	PI Controller II	
10^{-2}	41	34	43	35	JACS
	202	226	248	202	FNS
	35	47	44	35	STEPS
10^{-3}	74	48	39	74	JACS
	418	346	230	418	FNS
	74	74	40	74	STEPS
10^{-4}	411	146	38	253	JACS
	2328	1043	261	1451	FNS
	425	197	54	262	STEPS
10^{-5}	2805	897	3202	1390	JACS
	15783	6764	17954	7923	FNS
	2834	1443	3245	1423	STEPS
10^{-6}	2683	1383	4655	6645	JACS
	15160	10362	26317	37297	FNS
	2726	2269	4789	6760	STEPS

Table 4.3b. The number of Jacobian evaluations (JACS), function evaluations (FNS), and time steps (STEPS) needed to solve problem F5 from the stiff test set with DASSL and finite-difference Jacobians using the standard, STAB, and PI controllers.

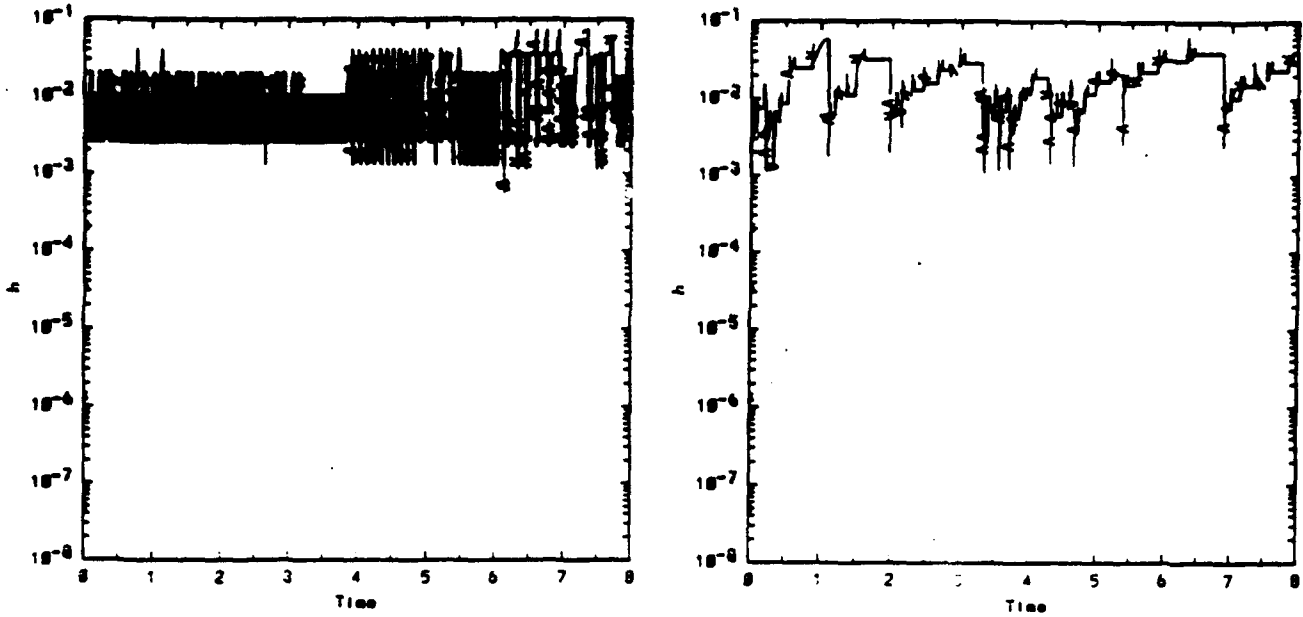


Figure 4.1. Time steps used in solving problem D2 from the stiff test set [4] on $0 < t \leq 8$ using DASSL with the standard controller (left) and the STAB controller (right) and with absolute and relative error tolerances of 0.00001 and $\alpha = 0.5$.