The Boring Topic of Stepsize Selection

Fred T. Krogh[1]

Boring as it may be, stepsize selection can make a big difference in the effectiveness of an algorithm. We begin with supporting evidence and then give a brief description of a new algorithm for doing this job. A more complete description is in [1].

We give results for the two-body problem

i	f_i	$y_i(2k\pi)$	$y_i((2k+1)\pi)$
1	y_2	$1-e$	$-1-e$
2	$-y_1/\sqrt{y_1^2+y_3^2}$	0	0
3	y_4	0	0
4	$-y_3/\sqrt{y_1^2+y_3^2}$	$\sqrt{(1+e)/(1-e)}$	$-\sqrt{(1-e)/(1+e)}$,

solved for t from 0 to 16π, with errors only computed at multiples of π, and for the the Euler Equations

i	f_i	$y_i(4kc)$	$y_i((4k+1)c)$	$y_i((4k+2)c)$	$y_i((4k+3)c)$
1	y_2y_3	0	1	0	-1
2	$-y_1y_3$	1	0	-1	0
3	$-.51y_1y_2$	1	.7	1	.7,

from $t = 0$ to $t = 28c$, where $c = 1.86264080233273855203028122057 9$, and errors only computed at multiples of c. The codes tested are

dop853: The code from [2], thanks Ernst and happy 60^{th}.
dxrk8: A code, [3], derived from dop853, using the stepsize selection introduced here.
rksuite8: The 8^{th} order Fortran 77 version of the code described in [4].
rksuite5: As for rksuite8, but 5^{th} order
dxrk8s: Like dxrk8, but using the stepsize selection described in [5] / [6]. We used Eq. (31) in [5], with $k = 8$ and $b = 4$.
diva: A variable order Adams code of ours similar to what is described in [7].
diva2: Diva with the two-body problems modeled as second order equations.

Since different codes use a different internal scaling of the users input tolerance, we have modified the value of the tolerance passed to the codes so that results are more comparable. An absolute error tolerance is used in all cases.

In the tables, E is the largest ratio for (global error) / tolerance, E_8 is the 8^{th} largest such ratio, $\overline{\text{NF}}$ is the average number of function evaluations required for a single case, Secs is the running time and the remaining columns give the counts for (global error) / tolerance for the ranges indicated.

If dop853 only did the extra derivatives required for interpolation when they were needed the 4134 below would be about 3370 and 2310 would be about 1940.

[1]email: fkrogh@mathalacarte.com, Math à la Carte, Inc., P.O. Box 616, Tujunga, CA 91043

Elliptical Two Body Problems, for
tol=$10^{-3} \times .96^j,\ j = 0, \ldots, 400$ $\quad e = .1 + .01j,\ j = 0, \ldots, 80$

code	E	E_8	$\overline{\mathbf{NF}}$	**Secs**	10^0-10^1	10^1-10^2	10^2-10^3	10^3-10^4	10^4-10^5
dop853	15684	8181	4134	52.2	7824	15219	7018	2417	3
dxrk8	12591	12190	2287	29.6	1765	9475	17211	3927	103
dxrk8s	81864	64940	2746	36.7	68	1102	14630	13595	3086
rksuite8	16231	13062	2629	98.5	2431	12734	14032	3254	30
rksuite5	24825	21001	4758	196.4	4987	14369	7512	5434	179
diva	22559	18206	1918	136.6	3171	14353	8768	5929	260
diva2	20063	17242	1739	130.7	5121	14798	7212	5106	244

Euler Equations, for tol=$10^{-3} \times .96^j,\ j = 0, \ldots,\ 400$

code	E	E_8	$\overline{\mathbf{NF}}$	**Secs**	10^0-10^1	10^1-10^2	10^2-10^3	10^3-10^4
dop853	1564.1	1216.9	2310	0.28	15	54	308	24
dxkr8	10.2	7.1	1510	0.20	400	1	0	0
dxrk8s	9.1	8.7	1516	0.21	401	0	0	0
rksuite8	37.6	28.2	1704	0.65	366	35	0	0
rksuite5	26.0	16.9	2710	1.16	365	36	0	0
diva	3.9	2.4	1267	0.72	401	0	0	0

We offer the following observations.

1. Comparing dop853 and dxrk8, we see a better control of the stepsize makes a very noticeable difference.
2. The comparison of dxrk8 and dxrk8s, suggests that the least squares error control used in dxrk8 works better than the digital filtering approach of dxrk8s when big stepsize changes are needed. The two are essentially the same for the Euler equations. (These codes were as close as we could make them in terms of the additional restrictions placed on the stepsize.)
3. Dxrk8 does better than rksuite8 in terms of function evaluations and the overhead for rksuite is surprisingly high. Rksuite8 does not have an interpolation facility, and thus it could not provide the G-Stop feature (finding zeros of functions of the variables involved in the solution) which is provided by dxrk8. Rksuite5 is only included to show the value of high order.
4. Diva results are given as (unneeded) further evidence that Adams methods are good if derivatives are expensive to compute, and bad if they are not.

The least squares error control assumes errors can be modeled with

$$\left| \frac{\text{estimated error}}{\text{requested accuracy}} \right| = \rho_n \approx e^{\phi_n} h_n^p, \tag{1}$$

and then fits ϕ_{n+1-k} with a linear or quadratic function of k. The fit is a least squares fit assuming that past residuals are to be weighted down by $w^{k/2}$, where

$0 < w < 1$, and $k = 1,\ 2,\ \ldots,\ \infty$. The results here used $w = .1$, but the results are not very dependent on the choice of w. We believe the quadratic model is likely to work better for a multistep code or low order Runge-Kutta code, but for the high order Runge-Kutta code we have been working with here, the linear model does a better job.

This least squares problem requires very little computation. At the end of step n, we can compute ϕ_n given ρ_n and h_n. The residuals for the normal equations for this least squares problem can be computed with

$$\begin{aligned} r_1 &= \phi_n + w r_1 \\ r_2 &= r_1 + w r_2 \\ r_3 &= r_2 + w r_3 \quad \text{(Not needed for the linear case)}, \end{aligned} \tag{2}$$

where the "=" sign is being used for assignment. The value of the constant term, $\hat{\phi}_{n+1}$ = the "expected" value for ϕ_{n+1}, is simply

$$\frac{1-w^2}{w} r_1 - \frac{(1-w)^2}{w} r_2 \qquad \text{linear,} \tag{3}$$

$$\frac{1-w}{w^2}\left[(1+w+w^2)r_1 + (-2+w+w^2)r_2 + (1-2w+w^2)r_3\right] \qquad \text{quadratic.} \tag{4}$$

The coefficients used here can of course be precomputed in parameter statements. Given $\hat{\phi}_{n+1}$ it is an easy matter to compute h_{n+1}. After the second step, or after a rejected step, the linear model can be used to generate the r_i using

$$\begin{aligned} r_1 &= \sum_{k=1}^{\infty} w^{k-1}\left[\phi_2 - (\phi_2 - \phi_1)(k-1)\right] &&= \frac{w\phi_1 + (1-2w)\phi_2}{(1-w)^2} \\ r_2 &= \sum_{k=1}^{\infty} k w^{k-1}\left[\phi_2 - (\phi_2 - \phi_1)(k-1)\right] &&= \frac{2w\phi_1 + (1-3w)\phi_2}{(1-w)^3} \\ r_3 &= \sum_{k=1}^{\infty} \frac{k(k+1)}{2} w^{k-1}\left[\phi_2 - (\phi_2 - \phi_1)(k-1)\right] &&= \frac{3w\phi_1 + (1-4w)\phi_2}{(1-w)^4}. \end{aligned} \tag{5}$$

That's the easy part. For the hard part we used an approach described in [8], and that sort of approach is more important than the details here. Although what we have seems to work, some of the choices are difficult to justify. Especially for large values of e in the two body problems, there are very sudden and dramatic changes in the behavior of ϕ. In such cases it is desirable that a code avoids bad choices that have been made earlier in the computation.

There were two things in addition to the algorithm above involved in the choice of h. Most important is to remember values of h that led to rejected steps, and to be cautious about increasing the stepsize when it is close to values that failed in the past. The other is to make use of the ratio of the order 5 and order 3 error estimates. Making use of this ratio not only provides a little additional caution when such is desirable, but it also provides a very cheap test for stiffness that at least for $y' = -cy$,with large and small values of c works very well. We have just enough space to describe what was done.

Set $\alpha = 1.5$ when starting. Never choose $\hat{\phi}_{n+1} > \hat{\phi}_n + .75 \log(\alpha * O_5/O_3)$, where O_5 and O_3 are the order 5 and order 3 error estimates. If this test results in a $\hat{\phi}_{n+1}$ smaller than we would have picked otherwise, set $\alpha = .6\alpha$. Then if $\alpha < .1$ and $O_5 > O_3$ give a one time "stiff" diagnostic and don't touch α again. If $\alpha < .1$ and $O_5 \leq O_3$, set $\alpha = 1.5$. When this test does not restrict $\hat{\phi}_{n+1}$, set $\alpha = 1.5$.

In the case of multistep methods we sample ϕ more frequently and thus in effect extrapolate a shorter distance. For Adams and BDF codes (no matter what representation of the interpolating polynomial is used), since the p^{th} divided difference approximates the p^{th} derivative we see from [7, p. 24, Eq. 2.1] a more exact model would have the error ratio given by $\rho_n = h_n(h_n + h_{n-1}) \cdots (h_n + \cdots + h_{n-p+1})e^{\phi_n}$. Using this model should do a much better job, than what we have used in the past. Given $\hat{\phi}_{n+1}$ we have a nonlinear equation in h_{n+1} to solve (with $\rho_{n+1} = 1$).

References

[1] Fred T. Krogh, **Least Squares Based Stepsize Selection for Ordinary Differential Equations**. Technical report, Math à la Carte, Inc. (2000). Submitted to TOMS, Currently at `http:\\mathalacarte.com/fkrogh`.

[2] E. Hairer, S. P. Nørsett, and G. Wanner, **Solving Ordinary Differential Equations I**, Springer Verlag, Berlin, second revised edition (1993).

[3] Fred T. Krogh, **Explicit Runge-Kutta Method for Ordinary Differential Equations (DXRK8)**. Technical report, Math à la Carte, Inc., Tujunga, CA (Feb. 1997) 14.2–01—14.2–10. With registration, available from `http://mathalacarte.com/cb/mom.fcg/ya65`.

[4] R. W. Brankin and I. Gladwell, *Algorithm 771.* `rksuite_90`*: Fortran software for ordinary differential equation initial value problems*, **ACM Transactions on Mathematical Software 23**, 3 (Sept. 1997) 402–415.

[5] Gustaf Söderlind, *Digital filters in adaptive time-stepping*, **ACM Transactions on Mathematical Software 29**, 1 (March 2003) 1–26.

[6] Kjell Gustafsson, *Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods*, **ACM Transactions on Mathematical Software 17**, 4 (Dec. 1991) 533–554.

[7] Fred T. Krogh, *Changing stepsize in the integration of differential equations using modified divided differences*, in Dale G. Bettis, editor, **Proceedings of the Conference on the Numerical Solution of Ordinary Differential Equations**, *Lecture Notes in Mathematics 362*, 22–71, Springer Verlag, Berlin (1974).

[8] Fred T. Krogh, *On developing mathematical software*, **J. Computational and Applied Mathematics 185** (Jan. 2006) 196–202. Preprint at `http://mathalacarte.com/fkrogh`.