Error control and stepsize selection.

Anne Kværnø

April 1, 2009

A user of some numerical black box software will usually require one thing: The accuracy of the numerical solution should be within some user specified tolerance. To accomplish this we have to measure the error, and if the error is too large, it has to be reduced. For ordinary differential equations, this means to reduce the stepsize. On the other hand, we would like our algorithm to be as efficient as possible, that is, to use large stepsizes. This leaves us with two problems: How to measure the error, and how to get the right balance between accuracy and efficiency.

Local error estimate. As demonstrated in Figure 1, the global error $y(t_n) - y_n$ comes from two sources: the local truncation error and the propagation of errors produced in preceding steps. This makes it difficult (but not impossible) to measure the global error. Fortunately it is surprisingly easy to measure the *local error*, l_{n+1}, the error produced in one step when starting at (t_n, y_n), see Figure 1. Let $y(t; t_n, y_n)$ be the exact solution of the ODE through the point t_n, y_n. For a method of order p we get

$$l_{n+1} = y(t_n + h; t_n, y_n) - y_{n+1} = \Psi(t_n, y_n)h^{p+1} + \mathcal{O}(h^{p+2}),$$

where $\mathcal{O}(h^{p+1})$ refer to higher order terms [1] . The term $\Psi(t_n, y_n)h^{p+1}$ is called *the principal error term*, and we assume that this term is the dominating part of the error. This assumption is true if the stepsize h is sufficiently small. Taking a step from the same point t_n, y_n with a method of order $\hat{p} = p + 1$ gives a solution $\hat{y}_{n+1}$ with a local error satisfying

$$y(t_n + h; t_n, y_n) - \hat{y}_{n+1} = \mathcal{O}(h^{p+2}).$$

The *local error estimate* is given by

$$le_{n+1} = \hat{y}_{n+1} - y_{n+1} = \Psi(t_n, y_n)h^{p+1} + \mathcal{O}h^{p+2} \approx l_{n+1}.$$

[1]Strictly speaking, the Landau-symbol $\mathcal{O}$ is defined by

$$f(x) = \mathcal{O}(g(x)) \quad \text{for } x \to x_0 \qquad \text{if} \qquad \lim_{x\to x_0} \frac{\|f(x)\|}{\|g(x)\|} < K < \infty$$

for some unspecified constant K. Thus $f(h) = \mathcal{O}(h^q)$ means that $\|f(h)\| \leq Kh^q$ when $h \to 0$, and refer to the remainder terms of a truncated series.

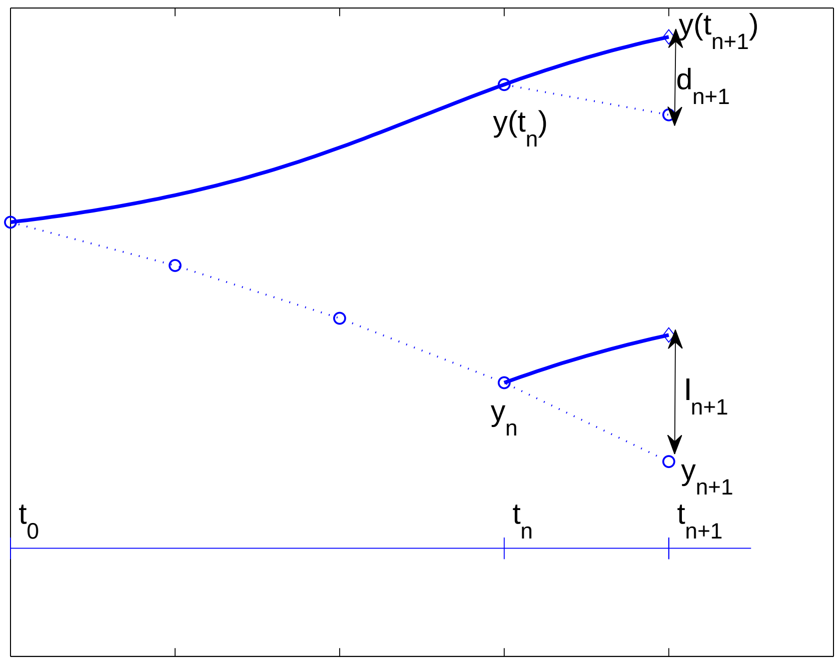

Figure 1: Lady Windermere's Fan

Embedded Runge-Kutta pair Given a Runge-Kutta method of order p. To be able to measure the local error, we need a method of order $p+1$ (or higher). But we do not want to spend more work (in terms of f-evaluations) than necessary. The solution is *embedded Runge-Kutta pairs*, which, for explicit methods are given by

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & \\
\hline
 & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_{s-1} & \hat{b}_s
\end{array}$$

The method given by the b_i's is of order p, the error estimating method given by the $\hat{b}_i$'s is of order $p+1$. (Sometimes it is the other way round. The important thing is to have two methods of different order.) The local error estimate of y_{n+1} is then given by

$$le_{n+1} = \hat{y}_{n+1} - y_{n+1} = h\sum_{i=1}^{s}(\hat{b}_i - b_i)k_i.$$

Example 0.1. *A combination of the Euler method and improved Euler will result in the* *following pair*

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

so that

$$k_1 = f(t_n, y_n), \quad k_2 = f(t_n + h, y_n + hk_1), \quad y_{n+1} = y_n + hk_1, \quad l_{n+1} \approx le_{n+1} = \frac{h}{2}(-k_1 + k_2).$$

Example 0.2. *Assume that you have decided to use improved Euler, which is of order 2, as your advancing method, and you would like to find an error estimating method of order 3. There are no 2-stage order 3 ERKs, so you have to add one stage to your method. This gives a method like*

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ c_3 & a_{31} & a_{32} & \\ \hline & \frac{1}{2} & \frac{1}{2} & \\ \hline & \hat{b}_1 & \hat{b}_2 & \hat{b}_3 \end{array}$$

where we require $c_3 = a_{31} + a_{32}$, which give us five free parameters. These have to satisfy all four order condition for an order 3 method. Using c_3 as a free parameter, we get the following class of 3th order methods:

$$b_1 = \frac{3c_3 - 1}{6c_3}, \quad b_2 = \frac{2 - 3c_3}{6(1 - c_3)}, \quad b_3 = \frac{1}{6c_3(1 - c_3)}, \quad a_{31} = c_3^2, \quad a_{31} = c_3 - c_3^2.$$

It is also possible to use the highest order method to advance the solution. In this case, we still measure the local error estimate of the lowest order order solution, but we get a more accurate numerical solution for free. This idea is called *local extrapolation.*

MATLAB has two integrators based on explicit Runge-Kutta schemes, `ODE23` which is based on an order 3/2 pair by Bogacki and Shampine, (a 3th order advancing and a 2nd order error estimating method), and `ODE45` based on an order 5/4 pair by Dormand and Prince. Both use local extrapolation.

Stepsize control Let the user specify a tolerance Tol, and a norm $\|\cdot\|$ in which the error is measured. Let us start with t_n, y_n, and do one step forward in time with a stepsize h_n, giving y_{n+1} and le_{n+1}. If $\|le_{n+1}\| \leq Tol$ the step is accepted, and we proceed till the next step, maybe with an increased stepsize. If $\|le_{n+1}\| > Tol$ the step is rejected and we try again with a smaller stepsize. In both cases, we would like to find a stepsize h_{new} which gives a local error estimate smaller than Tol, but at the same time as close to Tol as possible. To find the right stepsize, we make one assumption: The function $\Psi(t_n, y_n)$ of the principle error term do not change much from one step to the next, thus $\|\Psi(t_n, y_n)\| \approx \|\Psi(t_{n+1}, y_{n+1})\| \approx C$. Then

$$\begin{aligned} \text{we have:} \qquad & \|le_{n+1}\| &\approx C \cdot h_n^{p+1} \\ \\ \text{we want:} \qquad & Tol &\approx C \cdot h_{new}^{p+1} \end{aligned}$$

We get rid of the unknown C by dividing the two equations with each other, and h_{new} can be solved from

$$\frac{\|le_{n+1}\|}{Tol} \approx \left(\frac{h_n}{h_{new}}\right)^{p+1}.$$

Rejected steps are wasted work, and it should be avoided. Thus we choose the new stepsize somewhat conservative. The new stepsize is computed by

$$h_{new} = P \cdot \left(\frac{Tol}{\|le_{n+1}\|}\right)^{\frac{1}{p+1}} h_n. \tag{1}$$

where P is a *pessimist factor*, usually chosen somewhere in the interval [0.5,0.95]. In the discussion so far we have used the requirement $\|le_{n+1}\| \le Tol$, that is *error pr. step* (EPS). This do not take into account the fact that the smaller the step is, the more steps you take, and the local errors from each step adds up. From this point of view, it would make sense to rather use the requirement $le\|_{n+1} \le Tol \cdot h_n$, that is *error pr. unit step* (EPUS). The stepsize selection is then given by

$$h_{new} = P \cdot \left(\frac{Tol}{\|le_{n+1}\|}\right)^{\frac{1}{p}} h_n. \tag{2}$$

Careful analysis has proved that the local extrapolation together with EPS gives proportionality between the global error and the tolerance. The same is true for the use of the lower order method to advance the solution in combination with EPUS.