Error sources in the numerical solution of ODEs

1 Numerical Solution of ODEs

As with numerical differentiation and quadrature, the numerical solution of ordinary differential equations also involves errors that need to be understood and controlled. With quadrature and differentiation we were able to exploit Taylor's series with remainder to derive error estimates that could be used to construct adaptive algorithms for controlling overall error. For the numerical solution of ordinary differential equations our goal will be a slight bit more modest. We will try to derive an expression for the error in the solution that will allow us to understand what influences it; however, we will not take the next step of deriving an adaptive time-stepping methodology.

The general problem of interest is the first order ordinary differential equation written in standard form

$$\dot{y} = f(t, y)\,. \tag{1}$$

We will be interested in the solution of this equation at times t_n where $t_{n+1} = t_n + h$ with h being a fixed step size. In what follows we will denote the exact solution $y(t_n)$ as y_n. The approximate solution at the same time will be written as $\tilde{y}_n$ and the error in the approximation will written as $e_n = y_n - \tilde{y}_n$.

2 Forward Euler: Error Analysis

To keep the discussion concrete, let us consider the forward Euler method which is written as

$$\tilde{y}_{n+1} = \tilde{y}_n + hf(t_n, \tilde{y}_n)\,. \tag{2}$$

To determine an error expression let us first start with a Taylor expansion of the exact solution:

$$y_{n+1} = y_n + h\dot{y}_n + h^2\ddot{y}_{n+\theta}/2\,, \tag{3}$$

where by $\ddot{y}_{n+\theta}$ we mean $\ddot{y}(t)$ evaluated at some time between t_n and t_{n+1} (by Taylor's series with remainder). In the second term on the right in (3) we can use (1) to exactly replace $\dot{y}_n$ by $f(t_n, y_n)$. If we do this and then subtract (2) from (3), we will find that

$$e_{n+1} = e_n + h\left(f(t_n, y_n) - f(t_n, \tilde{y}_n)\right) + h^2\ddot{y}_{n+\theta}/2\,. \tag{4}$$

The second term on the right can be further reduced with a second exploitation of the Taylor series with remainder:

$$f(t_n, y_n) - f(t_n, \tilde{y}_n) = f(t_n, y_n) - \left(f(t_n, y_n) + \frac{\partial f}{\partial y}(t_n, \hat{y}_n)(\tilde{y}_n - y_n)\right)\,, \tag{5}$$

for some $\hat{y}_n$ between y_n and $\tilde{y}_n$. Noting that $\tilde{y}_n - y_n = -e_n$, we can substitute (5) back into (4) to yield

$$e_{n+1} = \underbrace{\left(1 + h\frac{\partial f}{\partial y}(t_n, \hat{y}_n)\right) e_n}_{\text{Stability contribution}} + \underbrace{h^2 \ddot{y}_{n+\theta}/2}_{\text{Truncation contribution}}. \tag{6}$$

Equation (6) tells us what contributes to the error in the numerical solution at each moment in time. There are two contributions. The first is associated with the propagation of an error that we have committed at one time step on to the next time step. This contribution to the error is known as stability error. The second contribution is known as truncation error and is quite similar to the errors we have already studied with respect to quadrature and differentiation.

For an overall accurate numerical solution of an ordinary differential equation one has to control both errors. For an integrator like forward Euler, this means that h needs to be sufficiently small in order to control both contributions. For other integrators, one can derive similar formulae. They will always involve a stability contribution and a truncation contribution.

Note that the stability error is closely connected to the derivative $\partial f/\partial y$. Equations for which this derivative makes the stability error term important are known as *stiff* equations. When faced with stiff equations it is usually important to select a time stepping scheme designed for stiff problems. In MATLAB for example, one could choose `ode23s` instead of `ode23`. Since both of these methods adaptively control the overall error they will both give reasonable answers; however, using `ode23s` on a stiff equation will often be more efficient than `ode23`.