

Mathematical Tripos Part IB: Lent Term 2024

Numerical Analysis – Lecture 11

11 Numerical implementation

Problem 11.1 The step size h is not some preordained quantity: it is a parameter of the method (in reality, many parameters, since we may vary it from step to step). The basic input of a well-written computer package for ODEs is not the step size but the *error tolerance*: the level of precision, as required by the user. The choice of $h > 0$ is an important tool at our disposal to keep a local estimate of the error beneath the required tolerance in the solution interval. In other words, we need not just a *time-stepping algorithm*, but also mechanisms for *error control* and for amending the step size.

Technique 11.2 (Milne device) Suppose that we wish to monitor the error of the trapezoidal rule (TR),

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \left[\frac{1}{2} \mathbf{f}_n + \frac{1}{2} \mathbf{f}_{n+1} \right].$$

We already know that the order is 2, more precisely (see Remark 6.9) the truncation error satisfies

$$\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}^{\text{TR}} = c_{\text{TR}} h^3 \mathbf{y}'''(t_n) + \mathcal{O}(h^4), \quad c_{\text{TR}} = -\frac{1}{12}, \quad (11.1)$$

where the number $c_{\text{TR}} = -\frac{1}{12}$ is called the *error constant* of TR. Unfortunately, this error estimate does not help much because the value $\mathbf{y}'''(t_n)$ is unknown. However, each multistep method (but not RK!) has its own error constant. For example, the 2-nd order 2-step Adams–Bashforth method (AB)

$$\mathbf{y}_{n+1} - \mathbf{y}_n = h \left[\frac{3}{2} \mathbf{f}_n - \frac{1}{2} \mathbf{f}_{n-1} \right],$$

has the error constant $c_{\text{AB}} = \frac{5}{12}$ (see Example 7.5), i.e.,

$$\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}^{\text{AB}} = c_{\text{AB}} h^3 \mathbf{y}'''(t_n) + \mathcal{O}(h^4), \quad c_{\text{AB}} = \frac{5}{12}. \quad (11.2)$$

The idea behind the *Milne device* is to use two multistep methods of the same order, one explicit and the second implicit (e.g., two methods just mentioned), to estimate the local error of the implicit method. Subtracting (11.2) from (11.1), we obtain the estimate $h^3 \mathbf{y}'''(t_n) \approx -\frac{\mathbf{y}_{n+1}^{\text{AB}} - \mathbf{y}_{n+1}^{\text{TR}}}{c_{\text{AB}} - c_{\text{TR}}}$, therefore

$$\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}^{\text{TR}} \approx -\frac{c_{\text{TR}}}{c_{\text{AB}} - c_{\text{TR}}} (\mathbf{y}_{n+1}^{\text{AB}} - \mathbf{y}_{n+1}^{\text{TR}}) = \frac{1}{6} (\mathbf{y}_{n+1}^{\text{AB}} - \mathbf{y}_{n+1}^{\text{TR}}), \quad (11.3)$$

and we use the right hand side as an estimate of the local error.

Note that TR is a far better method than AB: it is A-stable, hence its *global* behaviour is superior. We employ AB *solely* to estimate the local error (well, and for one more thing, see below). This adds very little to the overall cost of TR, since AB is an explicit method.

Implementation 11.3 To implement the *Milne device*, we work with a *pair* of multistep methods of the same order, one explicit (*predictor*) and the other implicit (*corrector*), e.g., two second order methods as in the example above – Adams–Bashforth and trapezoidal rule:

$$\begin{aligned} \text{Predictor:} \quad \mathbf{y}_{n+1}^{\text{P}} &= \mathbf{y}_n + h \left[\frac{3}{2} \mathbf{f}_n - \frac{1}{2} \mathbf{f}_{n-1} \right], \\ \text{Corrector:} \quad \mathbf{y}_{n+1}^{\text{C}} &= \mathbf{y}_n + h \left[\frac{1}{2} \mathbf{f}_n + \frac{1}{2} \mathbf{f}_{n+1} \right]. \end{aligned} \quad (11.4)$$

Depending on whether a user-specified tolerance $\epsilon > 0$ has been achieved (from the estimate (11.3)) we amend the step size h say by the factor of 2 (getting previous values with a smaller step size can be done by polynomial interpolation).

- (a) $\frac{1}{10}\epsilon \leq \|\text{error}\| \leq \epsilon$: accept the step and move to t_{n+2} with the same step size.
- (b) $\|\text{error}\| < \frac{1}{10}\epsilon$: accept the step and increase the step length.
- (c) $\|\text{error}\| > \epsilon$: reject the step, recommence integration with smaller step length.

The predictor is employed not just to estimate the error of the corrector, but also to provide *an initial guess in the solution of the implicit corrector equations*, which then can be solved, say, by the fixed point iteration (see p. 20). In the latter case, the formulas are as follows

$$\begin{aligned} \mathbf{y}_{n+1}^{(1)} &= \mathbf{y}_n + h \left[\frac{3}{2}\mathbf{f}_n - \frac{1}{2}\mathbf{f}_{n-1} \right], \\ \mathbf{y}_{n+1}^{(\ell+1)} &= \mathbf{y}_n + h \left[\frac{1}{2}\mathbf{f}_n + \frac{1}{2}\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}^{(\ell)}) \right], \quad \ell = 1, 2, \dots \end{aligned}$$

Usually, one stops the iteration when $\|\mathbf{y}_{n+1}^{(\ell+1)} - \mathbf{y}_{n+1}^{(\ell)}\| < \epsilon'$ is achieved.

Technique 11.4 (Embedded Runge–Kutta methods) The situation is more complicated with RK, since no single error constant determines local growth of the error. The approach of *embedded RK* requires, again, two (typically explicit) methods: an RK method of s stages and order p , say, and another method, of $s + m$ stages, $m \geq 1$, and order $p + 1$, such that *the first s stages of both methods are identical*. (This means that the cost of implementing the higher-order method is marginal, once we have computed the lower-order approximation.) For example, consider

$$\left. \begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n), \\ \mathbf{k}_2 &= \mathbf{f}\left(t_n + \frac{1}{2}h, \mathbf{y}_n + \frac{1}{2}h\mathbf{k}_1\right), \\ \mathbf{y}_{n+1}^{[1]} &= \mathbf{y}_n + h\mathbf{k}_2; \\ \mathbf{k}_3 &= \mathbf{f}(t_n + h, \mathbf{y}_n - h\mathbf{k}_1 + 2h\mathbf{k}_2), \\ \mathbf{y}_{n+1}^{[2]} &= \mathbf{y}_n + \frac{1}{6}h(\mathbf{k}_1 + 4\mathbf{k}_2 + \mathbf{k}_3). \end{aligned} \right\} \begin{array}{l} \text{order 2} \\ \text{order 3} \end{array}$$

We thus estimate $\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}^{[1]} \approx \mathbf{y}_{n+1}^{[2]} - \mathbf{y}_{n+1}^{[1]}$. (It might look paradoxical, at least at first glance, but the only purpose of the higher-order method is to provide error control for the lower-order one!)

Technique 11.5 (Zadunaisky device) Suppose that the ODE $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$, $\mathbf{y}(0) = \mathbf{y}_0$, is solved by an arbitrary numerical method of order p and that we have stored (not necessarily equidistant) past solution values $\mathbf{y}_n, \mathbf{y}_{n-1}, \dots, \mathbf{y}_{n-p}$. We form an interpolating p th degree polynomial (with vector coefficients) $\mathbf{q}$ such that $\mathbf{q}(t_{n-i}) = \mathbf{y}_{n-i}$, $i = 0, 1, \dots, p$, and consider the differential equation

$$\mathbf{z}' = \mathbf{f}(t, \mathbf{z}) + \mathbf{q}'(t) - \mathbf{f}(t, \mathbf{q}), \quad \mathbf{z}(t_n) = \mathbf{y}_n. \quad (11.5)$$

There are two important observations with regard to (11.5)

(1) Since $\mathbf{q}(t) - \mathbf{y}(t) = \mathcal{O}(h^{p+1})$, the term $\mathbf{q}'(t) - \mathbf{f}(t, \mathbf{q})$ is usually small (because $\mathbf{y}'(t) - \mathbf{f}(t, \mathbf{y}(t)) \equiv 0$). Therefore, (11.5) is a small perturbation of the original ODE.

(2) The exact solution of (11.5) is known: $\mathbf{z}(t) = \mathbf{q}(t)$. Now, having produced $\mathbf{y}_{n+1}$ with our numerical method, we proceed to evaluate $\mathbf{z}_{n+1}$ as well, *using exactly the same method and implementation details*. We then evaluate the error in $\mathbf{z}_{n+1}$, namely $\mathbf{z}_{n+1} - \mathbf{q}(t_{n+1})$, and use it as an estimate of the error in $\mathbf{y}_{n+1}$.