Numerical Solution of Differential Equations I

Endre Süli

Mathematical Institute
University of Oxford
2019

Lecture 10

Adaptivity for stiff problems

We would like to compute an approximate solution of the following initial-value problem for a system of first-order ODEs:

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}), \qquad \mathbf{y}(x_0) = \mathbf{y}_0, \tag{1}$$

for all $x \in [x_0, X_M]$, and make sure that this approximation is accurate up to a certain (absolute/relative) precision.

In addition, we would like to achieve such a precision in the fastest/cheapest way possible. How should this be done?

We shall describe two attempts, the first attempt being conceptually simpler, while the second attempt being the one preferred in practice for reasons which we shall explain.

First attempt

A simple strategy could be to:

1. choose a one-step method of order p;
2. choose a natural number $N \in \mathbb{N}$ and compute the approximate solution $\{\mathbf{y}_n\}_{n=0}^{N}$ using the step size $h = (X_M - x_0)/N$;
3. choose a large natural number $\tilde{N} \in \mathbb{N}$ with $\tilde{N} > N$ and compute the approximate solution $\{\tilde{\mathbf{y}}_n\}_{n=0}^{\tilde{N}}$ using the step size $\tilde{h} = (X_M - x_0)/\tilde{N}$.

This way, we obtain two approximations $\mathbf{y}_N$ and $\tilde{\mathbf{y}}_{\tilde{N}}$ of $\mathbf{y}(X_M)$.

We may then use the (computable) difference $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\|$ to estimate the (noncomputable) error $\|\mathbf{y}(X_M) - \mathbf{y}_N\|$.

If $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\|$ is smaller than a target absolute tolerance $\mathtt{TOL}$, then we finish the computation.

Else, we

1. increase N so that $N > \tilde{N}$;
2. compute the approximate solution $\{\mathbf{y}_n\}_{n=0}^{N}$ using $h = (X_M - x_0)/N$;
3. check whether $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\| < \mathtt{TOL}$.

If $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\|$ is smaller than the target absolute tolerance $\mathtt{TOL}$, then we finish the computation. Otherwise, we select a new $\tilde{N}$ such that $\tilde{N} > N$, and compute $\{\tilde{\mathbf{y}}_n\}_{n=0}^{\tilde{N}}$ using $\tilde{h} = (X_M - x_0)/\tilde{N}$.

This procedure is repeated (alternating N and $\tilde{N}$) until $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\|$ falls below the target absolute tolerance $\mathtt{TOL}$.

The following argument suggests that the (computable) difference $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\|$ can be used to estimate the error $\|\mathbf{y}(X_M) - \mathbf{y}_N\|$.

The idea to use $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\|$ to estimate $\|\mathbf{y}(X_M) - \mathbf{y}_N\|$ is based on the following calculations. Let us assume that $\tilde{N} > N$, and define $\alpha := \tilde{h}/h = N/\tilde{N} < 1$. For h sufficiently small,

$$\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\| = \|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}(X_M) + \mathbf{y}(X_M) - \mathbf{y}_N\| \le C(\tilde{h}^p + h^p) = (1+\alpha^p)Ch^p.$$

Thus,

$$\begin{aligned} \|\mathbf{y}(X_M) - \mathbf{y}_N\| &= \|\mathbf{y}(X_M) - \tilde{\mathbf{y}}_{\tilde{N}} + \tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\| \\ &\le \|\mathbf{y}(X_M) - \tilde{\mathbf{y}}_{\tilde{N}}\| + \|\mathbf{y}_{\tilde{N}} - \mathbf{y}_N\| \\ &\le C\tilde{h}^p + (1+\alpha^p)Ch^p \\ &\le \alpha^p\big(Ch^p\big) + (1+\alpha^p)\big(Ch^p\big). \end{aligned}$$

For $\alpha < 1$, $\alpha^p \ll 1 + \alpha^p$ (in relative terms).

Therefore, the term $\|\mathbf{y}(X_M) - \tilde{\mathbf{y}}_{\tilde{N}}\|$ has a minor contribution, and $\|\tilde{\mathbf{y}}_{\tilde{N}} - \mathbf{y}_N\|$ may be used to estimate $\|\mathbf{y}(X_M) - \mathbf{y}_N\|$.

This first adaptive strategy could deliver an accurate solution, but it is likely to be inefficient: whenever the target tolerance is not met we need to compute another solution from scratch on a finer mesh on the entire interval $[x_0, X_M]$.

Second attempt

To improve efficiency, we can try to control the consistency error for each mesh point x_n.

We have shown that (up to a constant factor, independent of h) the global error of a one-step method is bounded by the maximum of the consistency error.

The hope is therefore that we may compute a sufficiently accurate numerical solution by adapting the step size locally, that is, by selecting a suitable h_n for every x_n to control the local size of the consistency error.

To estimate the consistency error at $x = x_n$, in addition to the one step method

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\boldsymbol{\Phi}(x_n, \mathbf{y}_n; h) =: \boldsymbol{\Psi}(x_n, \mathbf{y}_n; h), \qquad n = 0, 1, \ldots;$$

of order p being used, we consider an additional one-step method

$$\tilde{\mathbf{y}}_{n+1} = \tilde{\mathbf{y}}_n + h\tilde{\boldsymbol{\Phi}}(x_n, \tilde{\mathbf{y}}_n; h) =: \tilde{\boldsymbol{\Psi}}(x_n, \tilde{\mathbf{y}}_n; h), \qquad n = 0, 1, \ldots,$$

of order $\tilde{p}$, with $\tilde{p} > p$, and we compute

$$\mathrm{ERR}(x_n; h) \coloneqq \|\tilde{\boldsymbol{\Psi}}(x_n, \mathbf{y}_n; h) - \boldsymbol{\Psi}(x_n, \mathbf{y}_n; h)\|. \qquad (2)$$

The idea behind using (2) to estimate the consistency error T_n is that, if the error has been controlled from x_0 up until x_n, for some $n \geq 1$, then the difference between $\mathbf{y}(x_n)$ and $\mathbf{y}_n$ is “negligible”, and therefore $\mathbf{y}_n$ can be assumed to be equal to $\tilde{\mathbf{y}}_n$.

Hence,

$$\begin{aligned} hT_n &= \mathbf{y}(x_{n+1}) - \mathbf{\Psi}(x_n, \mathbf{y}(x_n); h) \\ &= \mathbf{y}(x_{n+1}) - \tilde{\mathbf{\Psi}}(x_n, \mathbf{y}(x_n); h) + \tilde{\mathbf{\Psi}}(x_n, \mathbf{y}(x_n); h) - \mathbf{\Psi}(x_n, \mathbf{y}(x_n); h) \\ &\approx \mathbf{y}(x_{n+1}) - \tilde{\mathbf{\Psi}}(x_n, \mathbf{y}(x_n); h) + \tilde{\mathbf{\Psi}}(x_n, \mathbf{y}_n; h) - \mathbf{\Psi}(x_n, \mathbf{y}_n; h) \\ &\approx Ch^{\tilde{p}+1} + \tilde{\mathbf{\Psi}}(x_n, \mathbf{y}_n; h) - \mathbf{\Psi}(x_n, \mathbf{y}_n; h). \end{aligned}$$

Since the left-hand side is of the order $\mathcal{O}(h \times h^p) = \mathcal{O}(h^{p+1})$ and $\tilde{p} > p$, it follows that the term $\approx Ch^{\tilde{p}+1}$ on the right-hand side is "negligible" compared to the "leading term"

$$\tilde{\mathbf{\Psi}}(x_n, \mathbf{y}_n; h) - \mathbf{\Psi}(x_n, \mathbf{y}_n; h).$$

Hence,

$$hT_n \approx \tilde{\mathbf{\Psi}}(x_n, \mathbf{y}_n; h) - \mathbf{\Psi}(x_n, \mathbf{y}_n; h).$$

In summary, the locally adaptive strategy proceeds as follows:

at every step x_n

1. select an initial local step size h_n;
2. compute $\mathrm{ERR}(x_n; h_n)$;
3. if this is smaller than a target tolerance `TOL`, we set $\mathbf{y}_{n+1} = \mathbf{\Psi}(x_n, \mathbf{y}_n; h_n)$; else, we choose a smaller h_n and return to step 2.

To make this algorithm more efficient, it is common to increase the step h_n every time this step has been accepted, that is, to select βh_n for a suitable $\beta > 1$.

Remark

Let TOL *be a target absolute error tolerance and let* $\mathrm{ERR}(x_n; h_n) < \mathtt{TOL}$*. Then, the "optimal"* β *is*

$$\beta = \beta_n = \sqrt[(p+1)]{\mathtt{TOL}/\mathrm{ERR}(x_n; h_n)}. \tag{3}$$

Indeed, if $\mathrm{ERR}(x_n; h_n) < \mathtt{TOL}$*, we could have chosen a larger* h_n *and still satisfied the tolerance criterion.*

Let β_n *be such that* $\mathrm{ERR}(x_n, \beta_n h_n) = \mathtt{TOL}$*, so that* $\beta_n h_n$ *is the ideal step size. Then, we deduce* (3)*, because*

$$\mathrm{ERR}(x_n; \beta_n h_n) \approx C(\beta_n h_n)^{p+1} = \beta_n^{p+1} C h_n^{p+1} \approx \beta_n^{p+1} \mathrm{ERR}(x_n; h_n).$$

To further improve the efficiency of this adaptive algorithm, it is convenient to use embedded Runge–Kutta methods, which limit the number of function evaluations.

Definition

Two Runge–Kutta methods are *embedded* if they use the same stages. The Butcher tableau of two embedded Runge–Kutta methods can be written as

$$\begin{array}{c|c} \mathbf{a} & \mathbf{B} \\ \hline & \mathbf{c}_2^{\mathrm{T}} \\ & \mathbf{c}_1^{\mathrm{T}} \end{array}, \quad \text{where} \quad \begin{array}{c|c} \mathbf{a} & \mathbf{B} \\ \hline & \mathbf{c}_2^{\mathrm{T}} \end{array} \quad \text{and} \quad \begin{array}{c|c} \mathbf{a} & \mathbf{B} \\ \hline & \mathbf{c}_1^{\mathrm{T}} \end{array}$$

are the Butcher tableaux of the two Runge–Kutta methods, respectively.

Example

The Heun[1]–Euler method has the Butcher tableau:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \\ & 1 & 0 \end{array}, \quad \text{where} \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array} \quad \text{and} \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1 & 0 \end{array}$$

are the Butcher tableaux of Heun's method

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n))]$$

and the explicit Euler method $y_{n+1} = y_n + hf(x_n, y_n)$, respectively.

Example

MATLAB integrators for ODEs (such as the functions `ode45`, `ode23`, etc.) are based on embedded Runge–Kutta methods.[2]

[1]Karl Heun (3 April 1859, Wiesbaden – 10 January 1929, Karlsruhe)
[2]See L. F. Shampine and M. W. Reichelt, *The MATLAB ODE suite* (1997).