Lecture 8: Error Analysis for Time-Stepping Routines

Error analysis for time-stepping routines

- Accuracy and stability are fundamental to numerical analysis and are the key factors in evaluating any numerical integration technique. Therefore, it is essential to evaluate the accuracy and stability of the time-stepping schemes (时间步进方法) developed.

- Rarely does it occur that both accuracy and stability work in concert. In fact, they often are offsetting and work directly against each other.

- Thus a highly accurate scheme may compromise (妥协 折衷) stability, whereas a low accuracy scheme may have excellent stability properties.

exploring accuracy

In the context of time-stepping schemes, the natural place to begin is with Taylor expansions. Thus we consider the expansion

$$\mathbf{y}(t+\Delta t) = \mathbf{y}(t) + \Delta t \cdot \frac{d\mathbf{y}(t)}{dt} + \frac{\Delta t^2}{2} \cdot \frac{d^2\mathbf{y}(c)}{dt^2}$$

where $c \in [t, t+\Delta t]$. Since we are considering *dy/dt = f(t, y)*, **the above formula reduces to the Euler iteration scheme**

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \cdot f(t_n, \mathbf{y}_n) + O(\Delta t^2) .$$

It is clear from this that the truncation error is $O(t^2)$. Specifically, the truncation error is given by $\Delta t^2/2 \cdot d^2\mathbf{y}(c)/dt^2$.

Of importance is how this truncation error contributes to the overall error in the numerical solution. Two types of error are important to identify: local and global error.

Each is significant in its own right. However, in practice we are only concerned with the global (cumulative) error. The **global discretization error** is given by

$$E_k = \mathbf{y}(t_k) - \mathbf{y}_k$$

where $y(t_k)$ is the exact solution and y_k is the numerical solution.

The **local discretization error** is given by

$$\epsilon_{k+1} = \mathbf{y}(t_{k+1}) - (\mathbf{y}(t_k) + \Delta t \cdot \phi)$$

where $y(t_{k+1})$ is the exact solution and $\mathbf{y}(t_k) + \Delta t \cdot \phi$ is a one-step approximation (单步逼近) over the time interval $t \in [t_n, t_{n+1}]$.

For the Euler method, we can **calculate both the local and global error**.

Given a time-step Δt and a specified time interval $t \in [a, b]$, we have after *K* steps that $\Delta t \cdot K = b - a$. Thus we find

$$\text{local:} \quad \epsilon_k = \frac{\Delta t^2}{2} \frac{d^2\, y(c_k)}{dt^2} \sim O(\Delta t^2)$$

$$\text{global:} \quad E_k = \sum_{j=1}^{K} \frac{\Delta t^2\, y(c_j)}{2} \frac{d^2\, y(c_j)}{dt^2} \approx \frac{\Delta t^2\, d^2\, y(\mathbf{c})}{dt^2} \cdot K$$

$$= \frac{\Delta t^2}{2} \frac{d^2\, y(c)}{dt^2} \cdot \frac{b-a}{\Delta t} = \frac{b-a}{2} \Delta t \cdot \frac{d^2\, y(\mathbf{c})}{dt^2} \sim O(\Delta t)$$

which gives a local error for the Euler scheme which is $O(\Delta t^2)$ and a global error which is $O(\Delta t)$. Thus the cumulative error is poor for the Euler scheme, i.e. it is not very accurate.

A similar procedure can be carried out for all the schemes discussed thus far, including the multi-step Adams schemes.

Table 7 illustrates various schemes and their associated local and global errors. The error analysis suggests that the error will always decrease in some power of Δt.

Thus it is tempting to conclude that higher accuracy is easily achieved by taking smaller time steps Δt. This would be true if not for round-off error in the computer.

scheme	local error ϵ_k	global error E_k
Euler	$O(\Delta t^2)$	$O(\Delta t)$
2nd order Rung-Kutta	$O(\Delta t^3)$	$O(\Delta t^2)$
4nd order Rung-Kutta	$O(\Delta t^5)$	$O(\Delta t^4)$
2nd order Adams-Bnshforth	$O(\Delta t^3)$	$O(\Delta t^2)$

Round-off and step-size

An unavoidable consequence of working with numerical computations is round-off error. When working with most computations, double precision numbers (双精度实数) are used. This allows for 16-digit accuracy in the representation of a given number. This round-off has significant impact upon numerical computations and the issue of time-stepping.

As an example of the impact of round-off, we consider the Euler approximation to the derivative

$$\frac{d\mathbf{y}}{dt} \approx \frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{\Delta t} + \epsilon(\mathbf{y}_n, \Delta t)$$

where $\epsilon(\mathbf{y}_n, \Delta t)$ measures the **truncation error**. Upon evaluating this expression in the computer, **round-off error** occurs so that

$$\mathbf{y}_{n+1} = \mathbf{Y}_{n+1} + \mathbf{e}_{n+1}$$

Thus the combined error between the round-off and truncation gives the following expression for the derivative:

$$\frac{d\mathbf{y}}{dt} = \frac{\mathbf{Y}_{n+1} - \mathbf{Y}_n}{\Delta t} + E_n(\mathbf{y}_n, \Delta t)$$

where the total error, E_n, is the **combination of round-off and truncation** such that

$$E_n = E_{\text{round}} + E_{\text{trunc}} = \frac{\mathbf{e}_{n+1} - \mathbf{e}_n}{\Delta t} - \frac{\Delta t^2}{2}\frac{d^2\mathbf{y}(c)}{dt^2}$$

We now determine the maximum size of the error. In particular, we can bound the maximum value of round-off and the second derivative to be

$$|\mathbf{e}_{n+1}| \le e_r$$
$$|-\mathbf{e}_n| \le e_r$$
$$M = \max_{c\in[t_n, t_{n+1}]} \left\{ \left| \frac{d^2\mathbf{y}(c)}{dt^2} \right| \right\}$$

This then gives the maximum error to be

$$|E_n| \leq \frac{e_r + e_r}{\Delta t} + \frac{\Delta t}{2} M = \frac{2e_r}{\Delta t} + \frac{\Delta t M}{2}.$$

To minimize the error, we require that $\partial |E_n| / \partial(\Delta t) = 0.$. Calculating this derivative gives

$$\frac{\partial |E_n|}{\partial(\Delta t)} = -\frac{2e_r}{\Delta t^2} + \frac{M}{2} = 0$$

so that

$$\Delta t = \left(\frac{4e_r}{M} \right)^{1/2}$$

This gives the step-size resulting in a minimum error. Thus the smallest step-size is not necessarily the most accurate. Rather, a balance between round-off error and truncation error is achieved to obtain the optimal step-size.

Stability

The accuracy of any scheme is certainly important. However, it is meaningless if the scheme is not stable numerically. The essense of a stable scheme: the numerical solutions do not blow up to infinity (发散到无穷). As an example, consider the simple differential equation

$$\frac{dy}{dt} = \lambda y$$

with

$$y(0) = y_0$$

The analytic solution is easily calculated to be $y(t) = y_0 \exp(\lambda t)$. However, if we solve this problem numerically with a forward Euler method we find

$$y_{n+1} = y_n + \Delta t \cdot \lambda y_n = (1 + \lambda \Delta t) y_n.$$

After *N* steps, we find this iteration scheme yields

$$y_N = (1 + \lambda \Delta t)^N y_0$$

Given that we have a certain amount of round-off error, the numerical solution would then be given by

$$y_N = (1 + \lambda \Delta t)^N (y_0 + e).$$

The error then associated with this scheme is given by

$$E = (1 + \lambda \Delta t)^N e.$$

At this point, the following observations can be made.

For $\lambda > 0$, the solution $y_N \to \infty$ in Eq. (7.2.18) as $N \to \infty$. So although the error also grows, it may not be significant in comparison to the size of the numerical solution.

In contrast, Eq. (7.2.18) for $\lambda < 0$ is markedly different. For this case, $y_N \to 0$ in Eq. (7.2.18) as $N \to \infty$. The error, however, can dominate in this case. In particular, we have the following two cases for the error given by (7.2.19):

$$\begin{aligned} \text{I}: &\quad |1+\lambda\Delta t| < 1 \quad \text{then} \quad E \to 0 \\ \text{II}: &\quad |1+\lambda\Delta t| > 1 \quad \text{then} \quad E \to \infty. \end{aligned}$$

In case I, the scheme would be considered stable. However, case II holds and is unstable provided $\Delta t > -2/\lambda$.

A **general theory of stability** can be developed for any one-step time-stepping scheme.

Consider the one-step recursion relation for an $M \times M$ system

$$\mathbf{y}_{n+1} = \mathbf{A}\mathbf{y}_n.$$

After N steps, the algorithm yields the solution

$$\mathbf{y}_N = \mathbf{A}^N \mathbf{y}_0$$

where y_0 is the initial vector. A well-known result from linear algebra is that

$$\mathbf{A}^N = \mathbf{S}\mathbf{\Lambda}^N \mathbf{S}^{-1}$$

where S is the matrix whose columns are the eigenvectors of A, and

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_M \end{pmatrix} \rightarrow \Lambda^N = \begin{pmatrix} \lambda_1^N & 0 & \cdots & 0 \\ 0 & \lambda_2^N & 0 & \cdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \vdots & \\ 0 & \cdots & 0 & \lambda_M^N \end{pmatrix}$$

is a diagonal matrix whose entries are the eigenvalues of *A*. Thus upon calculating Λ^N, we are only **concerned with the eigenvalues**. In particular, **instability occurs** if $|\lambda_i| > 1$ for i = 1, 2, . . . ,M. This method can be easily generalized to two-step schemes (Adams methods) by considering $y_{n+1} = Ay_n + By_{n-1}$.

Lending further significance to this stability analysis is its connection with practical implementation. We contrast the difference in stability between the forward and backward Euler schemes. The forward Euler scheme has already been considered in (7.2.16)–(7.2.19). The backward Euler displays significant differences in stability. If we again consider (7.2.14) with (7.2.15), the **backward Euler method** gives the iteration scheme

$$y_{n+1} = y_n + \Delta t \cdot \lambda y_{n+1}$$

which after *N* steps leads to

$$y_N = \left(\frac{1}{1-\lambda\Delta t}\right)^N y_0$$

The round-off error associated with this scheme is given by

$$E = \left(\frac{1}{1-\lambda\Delta t}\right)^N e.$$

By letting $z = \lambda\Delta t$ be a complex number, we find the following criteria to yield unstable behavior based upon (7.2.19) and (7.2.27):

$$\begin{aligned} &\text{Forward Euler :} && |1+z| > 1 \\ &\text{Backward Euler :} && \left|\frac{1}{1-z}\right| > 1 \end{aligned}$$

Figure 7.3 shows the regions of stable and unstable behavior as a function of z. It is observed that the forward Euler scheme has a very small range of stability whereas the backward Euler scheme has a large range of stability. This large stability region is part of what makes implicit methods so attractive. Thus stability regions can be calculated. However, control of the accuracy is also essential.

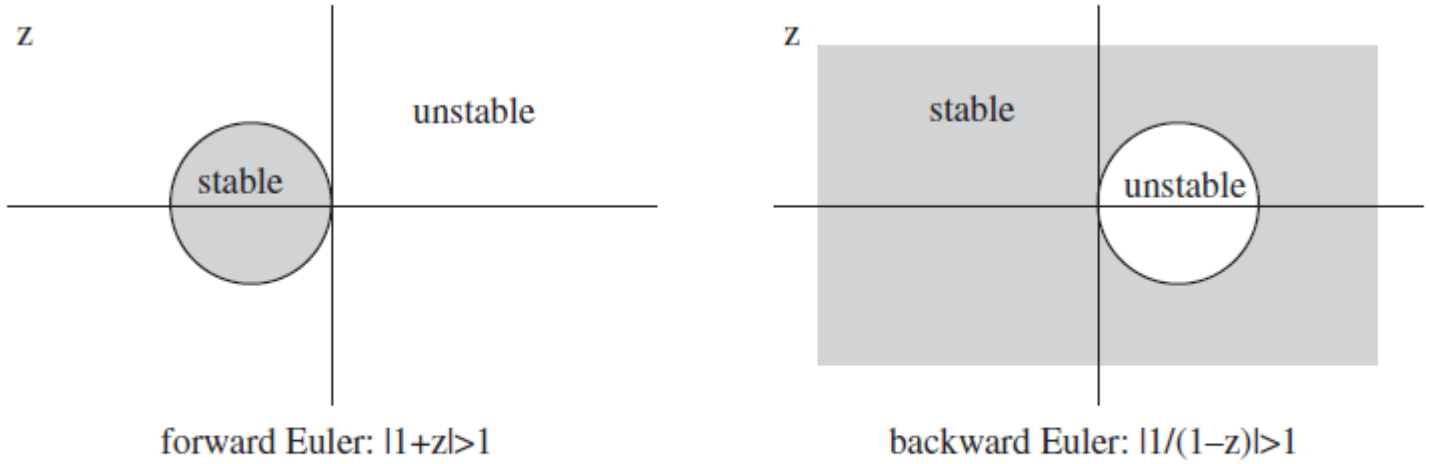

Figure 7.3: Regions for stable stepping (shaded) for the forward Euler and backward Euler schemes. The criteria for instability are also given for each stepping method.

Forward Euler method

$$\frac{dx}{dt} = \lim_{\Delta_t \to 0} \frac{\Delta x}{\Delta t}$$

We can rewrite the ODE as

$$\frac{x_{k+1} - x_k}{\Delta t} \approx \dot{x}_k = f(x_k)$$

thus the forward Euler approximation to the exact solution at x_{k+1} is

$$x_{k+1} = x_k + \Delta t f(x_k)$$

For $\dot{x} = Ax, \quad x(0) = x_0$, the approximation at x_{k+1} can be

$$x_{k+1} = x_k + \Delta t A x_k$$

$$= (I + \Delta t A) x_k$$

Not very stable!

Backward Euler method

$$\frac{dx}{dt} = \lim_{\Delta_t \to 0} \frac{\Delta x}{\Delta t}$$

We can rewrite the ODE as

$$\frac{x_{k+1} - x_k}{\Delta t} \approx \dot{x}_{k+1} = f(x_{k+1})$$

thus the backward Euler approximation to the exact solution at x_{k+1} is

$$x_{k+1} = x_k + \Delta t f(x_{k+1})$$

For $\dot{x} = Ax, \quad x(0) = x_0$, the approximation at x_{k+1} can be

$$\begin{aligned} x_{k+1} &= x_k + \Delta t A x_{k+1} \\ &= (I - \Delta t A)^{-1} x_k \end{aligned}$$

More stable!