Stability

While the accuracy (truncation error) of a method can be determined by using Taylor series, in some cases a particular method for a particular problem may be unstable:

small deviations from the true solution (what the algorithm ideally gives) grow as the algorithm progresses.

Consider Euler algorithm for $\dot{y} = f(y, t)$

$$\text{(1)} \quad y_{n+1} = y_n + \Delta t\, f(y_n, t_n) = y_n + \Delta t\, f_n$$

Assume that there is an error, δy_n, arising initially from e.g. roundoff error, associated with each step

$$f_n = f(y_n, t_n) \to f(y_n + \delta y_n, t_n)$$

$$= f(y_n, t_n) + \delta y_n \frac{\partial f}{\partial y}\Big|_n$$

Euler becomes

$$y_{n+1} + \delta y_{n+1} \doteq y_n + \delta y_n + \Delta t \left[f(y_n, t_n) + \delta y_n \frac{\partial f_n}{\partial y} \right]$$

$$\delta y_{n+1} = \delta y_n + \Delta t \frac{\partial f_n}{\partial y} \delta y_n$$

$$= \left(1 + \Delta t \frac{\partial f_n}{\partial y}\right) \delta y_n$$

$$\delta y_{n+1} = g\, \delta y_n$$

Error δy_n will increase at each step if $|g| > 1$

Euler is stable if $|g| < 1$

Consider three archetypal cases (α positive)

growth $y = y_0 e^{\alpha t}$ $\rightarrow$ $\dot{y} = \alpha y$ $\rightarrow$ $f = \alpha y$

decay $y = y_0 e^{-\alpha t}$ $\rightarrow$ $\dot{y} = -\alpha y$ $\rightarrow$ $f = -\alpha y$

oscillation $y = y_0 e^{i\alpha t}$ $\rightarrow$ $\dot{y} = i\alpha y$ $\rightarrow$ $f = i\alpha y$

For growth $g = 1 + \Delta t \frac{\partial f}{\partial y} = 1 + \Delta t \alpha > 1$

$\therefore$ unstable

decay $g = 1 - \Delta t \alpha$ $\rightarrow$ $|g| < 1$

$\rightarrow$ $-1 < g < 1$

$-1 < g$

$-1 < 1 - \Delta t \alpha$

$-2 < -\alpha \Delta t$

or $\Delta t < \frac{2}{\alpha}$

conditionally stable : Δt must be less than $\frac{2}{\alpha}$

oscillation $g = 1 + i\alpha\Delta t$

$|g|^2 = (1 + i\alpha\Delta t)(1 - i\alpha\Delta t) = 1 + \alpha^2\Delta t^2 > 1$

$\therefore$ Euler is unstable for oscillation

What about 2^{nd} order RK (midpoint method)?

Midpoint Method

$$y_{n+1/2} = y_n + \tfrac{1}{2}\Delta t\, f(y_n, t_n)$$
$$y_{n+1} = y_n + \Delta t\, f(y_{n+1/2}, t_{n+1/2})$$

$$y_{n+1} = y_n + \Delta t\, f\left(y_n + \tfrac{1}{2}\Delta t\, f(y_n, t_n),\ t + \Delta t/2\right)$$

$$= y_n + \Delta t\left[f(y_n, t_n) + \tfrac{1}{2}\Delta t\, f_n \frac{\partial f}{\partial y}\Big|_n + \frac{\Delta t}{2}\frac{\partial f}{\partial t}\Big|_n\right] + \mathcal{O}(\Delta t^3)$$

$$y_{n+1} = y_n + \Delta t\, f_n + \tfrac{1}{2}\Delta t^2\left(f_n \frac{\partial f}{\partial y}\Big|_n + \frac{\partial f}{\partial t}\right)$$

As before, $y_n \to y_n + \delta y_n$

$f_n \to f(y_n + \delta y_n, t_n) \approx f_n + \delta y_n \frac{\partial f}{\partial y}\Big|_n$

$$y_{n+1} + \delta y_{n+1} = y_n + \delta y_n + \Delta t\left[f_n + \delta y_n \frac{\partial f_n}{\partial y}\right] + \frac{\Delta t^2}{2}\left[\left(f_n + \delta y_n \frac{\partial f_n}{\partial y}\right)\frac{\partial f_n}{\partial y} + \frac{\partial f_n}{\partial t}\right]$$

$$\delta y_{n+1} = \delta y_n + \Delta t\, \delta y_n \frac{\partial f_n}{\partial y} + \frac{\Delta t^2}{2}\delta y_n \left(\frac{\partial f_n}{\partial y}\right)^2$$

$$= g\, \delta y_n \quad \text{with} \quad g = 1 + \Delta t \frac{\partial f_n}{\partial y} + \frac{\Delta t^2}{2}\left(\frac{\partial f_n}{\partial y}\right)^2$$

growth: $\frac{\partial f_n}{\partial y} = \alpha$ $\quad g = 1 + \alpha\Delta t + \alpha^2 \frac{\Delta t^2}{2} > 1$ unstable

decay: $\frac{\partial f_n}{\partial y} = -\alpha$ $\quad g = 1 - \alpha\Delta t + \alpha^2 \frac{\Delta t^2}{2}$

$-1 < g < 1 \rightarrow \quad 1 - \alpha \Delta t + \alpha^2 \Delta t^2/2 < 1$

$-\alpha \Delta t + \alpha^2 \Delta t^2/2 < 0$

$-1 + \frac{\Delta t}{2} \alpha < 0$

$\Delta t < \frac{2}{\alpha}$ (same as for Euler)

($-1 < g$ is also satisfied with this condition)

oscillation $\frac{\partial f_n}{\partial y} = i\alpha$, $g = 1 + i\alpha \Delta t - \frac{\Delta t^2}{2}\alpha^2$

$= 1 - \frac{\alpha^2 \Delta t^2}{2} + i\alpha \Delta t$

$|g|^2 = \left[\left(1 - \frac{\alpha^2 \Delta t^2}{2}\right) + i\alpha\Delta t\right]\left[\left(1 - \frac{\alpha^2 \Delta t^2}{2}\right) - i\alpha \Delta t\right]$

$= \left(1 - \frac{\alpha^2 \Delta t^2}{2}\right)^2 + \alpha^2 \Delta t^2$

$= 1 - \alpha^2 \Delta t^2 + \frac{\alpha^4 \Delta t^4}{4} + \alpha^2 \Delta t^2 = 1 + \frac{1}{4}(\alpha \Delta t)^4$

unstable

but $g \approx 1$ for $\alpha \Delta t \ll 1$

say $\alpha = 1$, $\Delta t = 10^{-3}$

$|g|^2 = 1 + \frac{10^{-12}}{4} \rightarrow |g| \simeq 1 + \frac{10^{-12}}{8}$

$(1+x)^{1/2} \approx 1 + \frac{x}{2}$

errors will accumulate quite slowly

Monte Carlo Simulations

- based on taking averages of many different realizations of a system. These configurations are generated using (pseudo) random numbers.

- Most random number generators use a chaotic sequence.

e.g. multiplicative congruent method
- based on large integers a and b that have no common factors

$$x_{n+1} = (a\, x_n) \% b$$

where $x \% y$ is the remainder after dividing x by y
C ↗ Fortran mod(x, y)

$$x \% y = x - \text{int}\left(\frac{x}{y}\right) * y$$

e.g. $\text{mod}(12, 5) = 12 - \text{int}\left(\frac{12}{5}\right) \times 5$
$= 12 - \text{int}(2.4) * 5 = 12 - 2 \cdot 5 = 2$

This sequence generates integers less than b $[1, b-1]$
in a "random" order.
- first value x_0 is called the seed
- for a given x_0, sequence x_n is completely determined
- for a new x_0, a new sequence is generated
- not at all random → pseudorandom

For this simple algorithm, quality of pseudorandom sequence depends a lot on choice of a and b

A good pair is (Lewis, Goodman, Miller 1969)

$$a = 7^5 = 16807 \qquad \text{(largest 32-bit unsigned}$$
$$b = 2^{31} - 1 = 2\,147\,483\,647 \qquad \text{integer is } 2^{32} - 1 \text{)}$$

When implementing, need to take care that integer product can be stored ($b * b \approx 2^{62}$)

- either use unsigned long integer (64-bit integer) or use computational tricks (see Numerical Recipes)

A slightly more general class of pseudo-RNG is the linear congruential generator

$$x_{n+1} = \mathrm{mod}(a x_n + c, b)$$

(congruent map: 

linear: $ax + c$)

Basic idea: multiply two big integers → least significant digits look random

To get result on $[0,1)$, simply take $\dfrac{x_{n+1}}{b}$ } or $[0,1]$ $\dfrac{x_{n+1}}{b-1}$

Many algorithms exist for generating RNs on $[0,1)$ and $[0,1]$ see NR, www, google "Mersenne Twister"