Errors in numerical approximations

Continuous problem: $y' = f(t,y)$, $y(t_0) = y_0$
$t_0 < t < t_{end}$

We discretize time: $\Delta t = \frac{t_{end} - t_0}{N}$, $t_n = t_0 + n \cdot \Delta t$, $n = 0, \dots, N$

and approximate the ODE by a Finite Difference Equation (FDE)

via some method, e.g. Euler scheme, and to compute discrete $Y_n \approx y(t_n)$, $n = 0 : N$

Continuous problem: $ODE[y] = y' - f(t,y) = 0$, $y(t_0) = y_0$

Discretized problem: $FDE[Y] = \frac{Y_{n+1} - Y_n}{\Delta t} - f(t_n, Y_n) \overset{=0}{,}$ $Y_0 = y_0$ for Euler

$y(t)$ = exact sol of $ODE[y] = 0$

Y_n = exact sol of $FDE[Y] = 0$ (at infinite precision) $\approx y(t_n)$

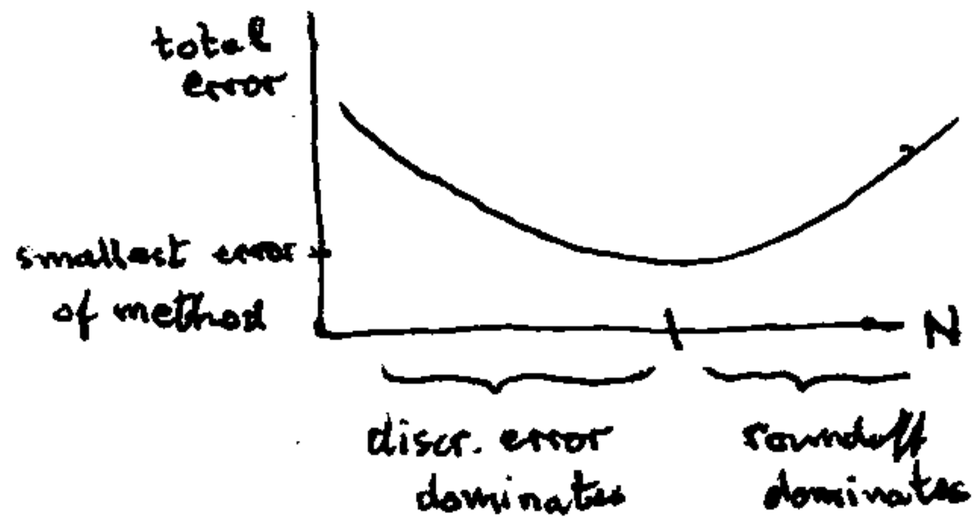
$\tilde{Y}_n$ = actual computed values (at finite precision) $\approx Y_n$

Discretization error at n-th timestep: $de_n = y(t_n) - Y_n$.

Roundoff error at n-th step: $re_n = Y_n - \tilde{Y}_n$

Total error (actual) at n-th step: $y(t_n) - \tilde{Y}_n = \underbrace{y(t_n) - Y_n}_{\text{discr. error}} + \underbrace{Y_n - \tilde{Y}_n}_{\text{roundoff error}}$

To improve the approximation we reduce Δt (increase N),
so more computation, slower, and more chances for roundoff to increase!



As N grows, roundoff grows,
and eventually dominates!
Accuracy cannot be improved further
with this method!

If the smallest achievable error is too big, then another method must be used!
(a higher order method)

Consistancy error: how well the FDE approximates the ODE

(local truncation error)

$$ce_n = FDE[y(t_n)] - ODE[y(t_n)]$$

= amount by which the exact sol $y(t_n)$ of ODE fails to satisfy the FDE

for Euler

$$= \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n))$$

can be estimated by Taylor expansion and reveals order of method

Definitions: A num. scheme is called

convergent if discretization error $|y(t_n) - Y_n| \to 0$ as $\Delta t \to 0$

consistent if consistency error $\to 0$ as $\Delta t \to 0$

stable if any error introduced at any step remains bounded in subsequent steps. Messy business...

(cannot hope for roundoff to $\to 0$, in fact will increase, but it should remain bounded, not blow up!)

Clearly, want num. methods to be consistent and convergent and stable, else useless...

Lax Equivalence Theorem: For a well-posed problem, a consistent method is convergent iff stable.

Also want num. methods to be

Efficient: get acceptable accuracy at reasonable cost (# of f-evaluations)

Robust: perform reliably on large classes of problems.

Fact: There is no best method for all problems!

Consistency error in Euler scheme

$$FDE[y(t_n)] - \cancel{ODE[y(t_n)]}^{\to 0} = \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - f(t_n, y(t_n))$$

by Taylor expansion of $y(t)$ about t_n:

$$= \frac{1}{\Delta t}\left[y'(t_n)\cdot\Delta t + \frac{1}{2} y''(\xi_n)\cdot\Delta t^2 \right] - f(t_n, y(t_n))$$

$$= y'(t_n) - f(t_n, y(t_n)) + \frac{1}{2} y''(\xi_n)\cdot\Delta t$$

$$= ODE[y(t_n)] + \frac{1}{2} y''(t_n)\cdot\Delta t$$

$$\Rightarrow \quad |\text{consistency error}| \equiv |FDE[y(t_n)]| = (\text{const})\cdot\Delta t = O(\Delta t)$$

assuming $|y''(t)| \le M$

$\to 0$ as $\Delta t \to 0$

Therefore Euler scheme is consistent and 1^{st} order accurate.

Discretization error in Euler scheme: (convergence estimate)

$$|y(t_n) - Y_n| \le e^{K\cdot(t_n - t_0)}\cdot|y_0 - Y_0| + \left(\frac{M_2}{2K}\left[e^{K(t_n - t_0)} - 1\right]\right)\cdot\Delta t$$

where K = Lipschitz const. of f w.r.t. y $\left(\left|\frac{\partial f}{\partial y}\right| \le K\right)$

M_2 = upper bound on y'' $\left(|y''(t)| \le M_2\right)$

so, apart from initial error $|y_0 - Y_0|$, $|\text{discr. error}| \le (\text{const.})\cdot\Delta t = O(\Delta t)$

so indeed 1^{st} order

Note that even a small initial error $|y_0 - Y_0|$ can grow big for large $t_n - t_0$ (long time) or large K

Total error in Euler: If $\varepsilon_0, \varepsilon_1, \ldots$ are roundoff errors in Euler steps

i.e. $\tilde{Y}_0 = y_0 + \varepsilon_0$, $\tilde{Y}_{n+1} = \tilde{Y}_n + \Delta t\cdot f(t_n, \tilde{Y}_n) + \varepsilon_{n+1}$, $i = 0:N$, $\varepsilon = \max_n \varepsilon_n$

then total error at any step is

$$|y(t_n) - \tilde{Y}_n| \le \underbrace{\left(\frac{M_2}{2K}\cdot\Delta t + \frac{\varepsilon}{\Delta t}\right)\cdot\left[e^{K(t_n - t_0)} - 1\right]}_{\text{discretiz. error}} + \underbrace{|\varepsilon_0|\cdot e^{K(t_n - t_0)}}_{\text{roundoff}}$$

$\searrow$ $\nearrow$ as $\Delta t \to 0$ $\Rightarrow$

Stability of Euler scheme

The above estimate also shows that Euler is stable (for any fixed Δt),
i.e. errors remain bounded (not blowing up).
This is analogous to a problem being well-conditioned.

For more complicated schemes, it is hard to obtain such estimates on total error,
so another approach is used:

Apply the scheme to the model linear problem $\begin{cases} y' = \lambda y \\ y(0) = y_0 \neq 0 \end{cases}$, solution $y(t) = y_0 e^{\lambda t}$

Euler: $Y_{n+1} = Y_n + h \cdot \lambda \cdot Y_n = (1+\lambda h) Y_n = (1+\lambda h)^2 Y_{n-1} = \cdots = (1+\lambda h)^n Y_0$

amplification factor at each step is $1+\lambda h$, so

if $|1+\lambda h| \leq 1$ then errors do not grow $\Rightarrow$ scheme is stable

if $|1+\lambda h| > 1$ then errors do grow $\Rightarrow$ scheme is unstable

so Euler is stable at step size h if $-1 \leq 1+\lambda h \leq 1$ i.e. $-2 \leq \lambda h \leq 0$

$[-2, 0]$ is called the interval of absolute stability of the scheme

When $\lambda > 0$, the ODE itself is "unstable" (trajectories diverge),
Y_n must grow, errors will grow. It's the DE's fault,
not scheme's fault.

When $\lambda < 0$, the DE is "stable", Y_n should decay, so need $-2 \leq \lambda h$
$\Rightarrow h < \frac{2}{-\lambda}$, so time step $\Delta t = h$ must be restricted $< \frac{2}{-\lambda}$
to be stable. Euler scheme is conditionally stable.

All explicit schemes are conditionally stable, Δt must be restricted for stability.

Stability of Backward Euler scheme: $Y_{n+1} = Y_n + h \cdot f(t_{n+1}, Y_{n+1})$

Apply it to $y' = \lambda y$, $\lambda < 0$: $Y_{n+1} = Y_n + h \cdot \lambda Y_{n+1}$

$$(1 - \lambda h) Y_{n+1} = Y_n$$

$$\Rightarrow \quad Y_{n+1} = \frac{1}{1-\lambda h} Y_n = \left(\frac{1}{1-\lambda h}\right)^2 Y_{n-1} = \dots = \left(\frac{1}{1-\lambda h}\right)^n Y_0$$

To decay: $\left|\frac{1}{1-\lambda h}\right| < 1 \Rightarrow |1-\lambda h| > 1 \Rightarrow 1-\lambda h < -1$ or $1-\lambda h > 1$

$2 < \lambda h$ or $\lambda h < 0$

$\therefore$ no restriction for stability (unconditionally stable)

only for accuracy h needs to be chosen small enough

This is the advantage of implicit schemes: better stability!

λ plays the role of $\frac{\partial f}{\partial y}(t_n, Y_n) \overset{=J}{}$. For systems λ plays role of eigenvalues of Jacobian matrix $\frac{\partial \vec{f}}{\partial \vec{y}} = J$

Stiff problems: when solution has components that vary in widely different time scales (very small and very large eigenvalues).
Some components may decay very fast (transients) while other components persist. For very small times solution would be very different than later, which is what we really want to capture.

This situation gets many methods into trouble (especially explicit schemes):
need very small Δt to remain stable, unreasonably small for tracking the persistant slow-varying components!

BDF (Backward Differentiation Formula) schemes of Gear-type were developed in 1960s