Errors in Numerical Approximations

To "solve" (numerically approximate) the (IVP), we discretize time into discrete timesteps $t_0 < t_1 < \dots < t_n < \dots < t_N$ and construct numerical approximation $Y_n \approx y(t_n)$ by some method, e.g. Euler scheme.

Continuous problem: ODE	Discrete problem: Finite Difference Equ
solution: $y = y(t)$, $t_0 < t < t_{end}$	solution: Y_n, $n = 0 : N$
$ODE[y] := y' - F(t,y) = 0$	$FDE[Y_n] := \frac{Y_{n+1} - Y_n}{\Delta t} - F(t_n, Y_n) = 0$
IC: $y(t_0) = y_0$	$Y_0 = y_0$ for Euler

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

compares exact sol $y(t)$ of $ODE[y] = 0$ with exact sol Y_n of $FDE[Y_n] = 0$

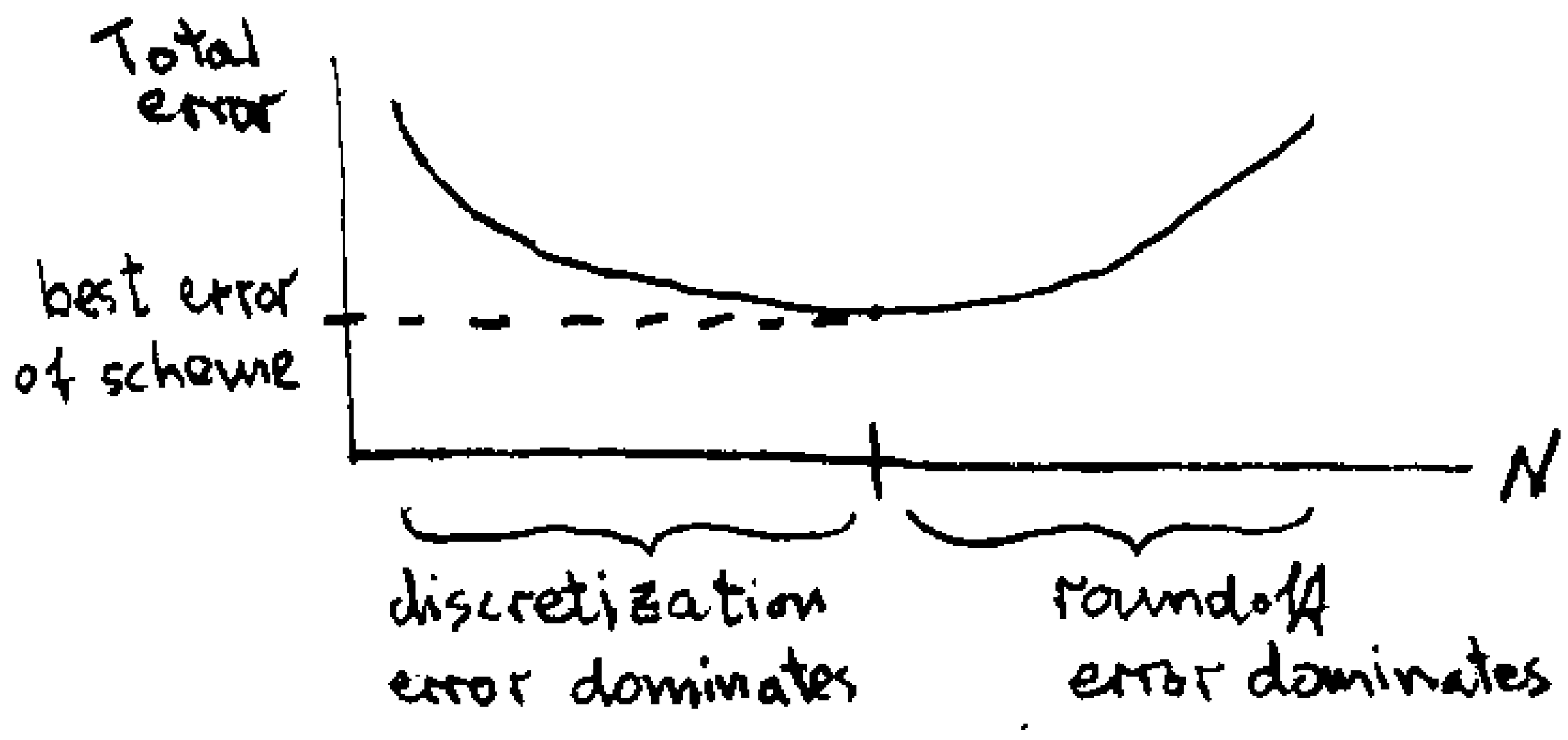
Roundoff error at n^{th} timestep: $re_n = Y_n - \tilde{Y}_n$

compares exact sol. of $FDE[Y_n] = 0$ with actual sol. from calculation in finite precision.

Actual Total error $= y(t_n) - \tilde{Y}_n \equiv \underbrace{y(t_n) - Y_n}_{\text{discret. error}} + \underbrace{Y_n - \tilde{Y}_n}_{\text{roundoff error}}$

We can improve discr. error by taking smaller Δt, i.e. bigger N: $\Delta t = \frac{t_{end} - t_0}{N}$.

But bigger N implies more computations and more chances for roundoff to pile up.



As N grows, roundoff grows and eventually becomes bigger than discret. error, so total error gets worse!

Each scheme can achieve some minimal error that cannot be further improved.

To reduce roundoff: careful coding, higher precision computation (uses more memory, slower)

If total error still too large for our purposes, will have to use another, higher order, method. There are many ODE integrators...

There is NO best method for all classes of problems!

Errors ...

Consistency error = FDE$[y(t_n)]$ − ODE$[y(t_n)]$

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

compares the ODE and FDE problems, measuring how well the FDE approximates the ODE itself.
Can be approximated via Taylor expansion, and reveals the order of the method

A numerical method (scheme) is called:

consistent if consistency error → 0 as $\Delta t \to 0$ $(N \to \infty)$. [Euler is consistent]

convergent if discretization error → 0 as $\Delta t \to 0$ [Euler is convergent]

stable if any error at any step remains bounded in later steps (it may grow but not keep on growing) [Euler is stable]
Unstable schemes are totally useless!

We want schemes that are consistent, convergent, stable and also efficient and robust and accurate

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

Consistency error in Euler Method = FDE$[y(t_n)]$ − ODE$[y(t_n)]$

$$= \frac{y(t_{n+1}) - y(t_n)}{\Delta t} - F(t_n, y(t_n)) - 0$$

By Taylor: $y(t_n + \Delta t) = y(t_n) + y'(t_n)\cdot\Delta t + \frac{y''(z_n)}{2!}\cdot\Delta t^2$ for some $t_n < z_n < t_{n+1}$

$$\Rightarrow \quad = y'(t_n) + \frac{y''(z_n)}{2!}\Delta t - F(t_n, y(t_n))$$

$$= \underbrace{\mathrm{ODE}[y(t_n)]}_{0} + \frac{y''(z_n)}{2}\cdot\Delta t$$

$$= O(\Delta t)$$ so 1st order accurate

Errors ....

Accuracy of Euler Method (convergence estimate): The discritization error of Euler scheme satisfies

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

where K = Lipschitz const. of $F \leq \sup\left|\frac{\partial F}{\partial y}\right| < \infty$, $M_2 = \sup|y''| < \infty$ assumed

Remarks: 1. Even a small initial error can become big for large t_n (long term computation) or large K

but always remains bounded.

2. Apart from the initial error, $|\text{error}| \leq (\text{const.})\ \Delta t = \mathcal{O}(\Delta t)$
($\Rightarrow$ first order method)

So convergent $\therefore$ also stable.

3. 1^{st} order accuracy means error is proportional to Δt.
Halving Δt we expect half the error.

Big O notation: $f(x) = \mathcal{O}(g(x))$ as $x \to x_0$ means $\left|\frac{f(x)}{g(x)}\right| \leq M$ for x near x_0
i.e. $|f(x)| \leq (\text{const.})\,|g(x)|$ near x_0

Excellent ODE solvers are available: VODE, rksuite, ... in netlib.org
GSL

There is **no** best method for all ODEs or even for classes of ODEs.

Types of ODE integrators:

explicit - implicit - BDF for "stiff"
single step - multistep
Taylor type - RK
symplectic - nonsymplectic
non-adaptive - adaptive

Matlab has 8 integrators: rk4, rk45, ode113, ode113s, ode15i, 15s, ode23, 23s, 23t, ode23tb

Runge-Kutta (RK) methods

characterized by number of stages and order, very well studied
1^{st} order RK is Euler.
Most famous and popular: classical 4^{th} order RK, has 4 stages K_i

rk4

$$Y_0 = y_0$$
$$Y_{n+1} = Y_n + \frac{\Delta t}{6}\left[K_1 + 2K_2 + 2K_3 + K_4\right]$$

with stages
$$K_1 = F(t_n, Y_n)$$
$$K_2 = F\left(t_n + \frac{\Delta t}{2},\; Y_n + \frac{\Delta t}{2}\cdot K_1\right)$$
$$K_3 = F\left(t_n + \frac{\Delta t}{2},\; Y_n + \frac{\Delta t}{2}\cdot K_2\right)$$
$$K_4 = F\left(t_n + \Delta t,\; Y_n + \Delta t\cdot K_3\right)$$

It is 4^{th} order accurate, i.e. discretization error $= O(\Delta t^4)$
so $\frac{\Delta t}{2}$ reduces error by $\frac{1}{2^4} = \frac{1}{16}$

It has 4 stages, so requires 4 function evaluations, so 4 times costlier than Euler,
so about 4 times more expensive per step than Euler,

but achieves much higher accuracy, so perhaps can use larger Δt.

Classical RK4 is optimal: uses 4 evaluations for 4^{th} order
but any 5^{th} order RK needs 6 stages
6^{th} 7 or 8 stages

RKF (rk45): Fehlberg (1969) devised the famous RKF method;
adaptive
uses a 4^{th} order and a 5^{th} order, with total 6 evaluations per step
to estimate local error and adapt (select) next Δt.
If error is low, increase Δt
if error is high, decrease Δt
hoping to get to t_{end} in fewer steps overall.

works well on many ODEs, excellent implementations exist (rk45 in Matlab)
(rksuite package)