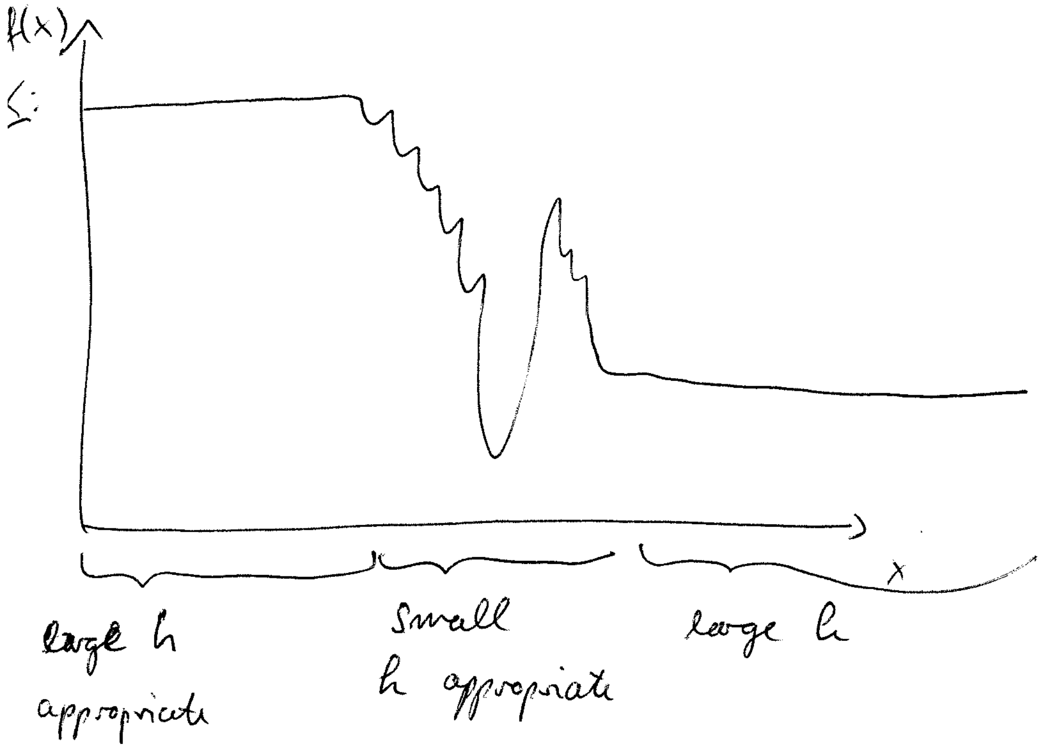
Adaptive Quadrature



many functions have regions where they barely change and regions where they change <u>a lot</u>

Integration becomes expensive when minimum necessary h is used everywhere

Idea: Change h locally depending on behavior of $f(x)$

Implementation (Recursive)

- Subdivide $[a,b]$ into coarsest grid (e.g. $N = 100$)
- on each subinterval, estimate the error of the integration by locally performing <u>a finer</u> quadrature (say, $N = 200$).
- if the two quadratures do not differ much, we are done (large h region)
- otherwise, refine the discretization (small h region)

Jumerical Integration of ODEs

$$(*) \qquad \frac{du}{dt} = f(t,u) \quad , \qquad u(0) = u_0$$ (initial value problem)

icard-Lindelöf theorem: if $f(t,u)$ is Lipschitz,
en there is a unique solution to $(*)$ on some
terval $[-\varepsilon, +\varepsilon]$.

-> But we do not know in general the form of the solution $u(t)$.

-> ODE integrator takes $(*)$ and approximates

$$(t_0, u_0) \; , \; (t_1, u_1), \; (t_2, u_2), \ldots$$

for given $t_0, t_1, \ldots$

oupled ODEs

Ex:

$$\frac{du_1}{dt} = (\gamma_1 - \gamma_a) u_1 + \gamma_b u_2$$

$$\frac{du_2}{dt} = \gamma_a u_1 + (\gamma_2 - \gamma_b) u_2$$

(population dynamics)

$$\Rightarrow \quad \frac{d}{dt} \underbrace{\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}}_{= \vec{u}} = \vec{f}(t, \vec{u})$$

or linear systems,

$$\vec{f}(t,\vec{u}) = \underbrace{A}_{\text{matrix}}\vec{u}$$

nd $\vec{u}(t) = e^{At}\vec{u}_0$.

mpossible for nonlinear $\vec{f}(t,\vec{u})$ (in general)

<u>gher order ODEs</u>

<u>Ex:</u> $$\frac{d^2 u_1}{dt^2} = \frac{1}{m} F(u_1)$$ (Newton's law)

n into first-order system by defining

$$\left.\begin{aligned} \frac{du_1}{dt} &= u_2 \\ \frac{du_2}{dt} &= \frac{1}{m} F(u_1) \end{aligned}\right\} \text{system of first-order ODEs}$$

, only need a theory to solve first-order systems x:

$$\dddot{x} + A\ddot{x} - (\dot{x})^2 + x = 0$$

fine $u_1 = x$

$u_2 = \dot{x} = \dot{u}_1$

$u_3 = \ddot{x} = \dot{u}_2$

$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} u_2 \\ u_3 \\ -u_1 + u_2^2 - Au_3 \end{pmatrix}$$

Comparison to numerical quadrature

Ex. $\frac{du}{dt} = f(t,u)$ $\iff$ $u(t) = \int_a^t \underbrace{f(t', u(t'))}_{\text{integrand depends on } u(t)!} dt'$

$u(a) = 0$

ntegrand depends on the integral at previous values of t!

quadrature must be done <u>incrementally</u>, one point at a time xamples:

· planetary motion $$m\ddot{\vec{r}} = -G\frac{Mm}{r^2}\hat{r}$$

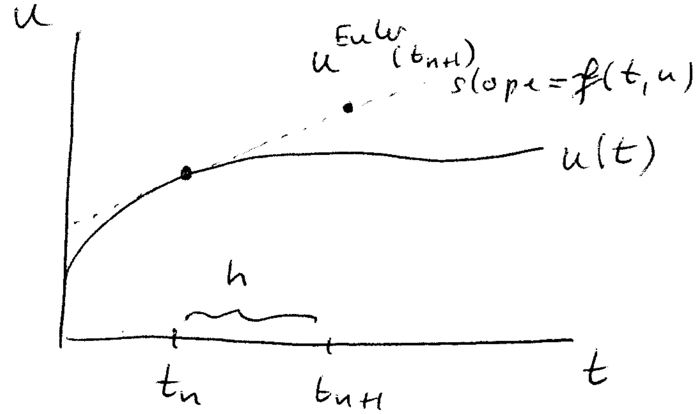
molecular dynamics
$$m\ddot{\vec{r}}_i = -\vec{\nabla}\,\mathcal{U}(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_i, \ldots, \vec{x}_N)$$

chemical kinetics
$$O_3 + O_2 \underset{k_2}{\overset{k_1}{\rightleftharpoons}} O + 2O_2$$
$$O_3 + O \xrightarrow{k_3} 2O_2$$
(ozone in the atmosphere)

$$\frac{d}{dt}\begin{pmatrix}[O]\\ [O_2]\\ [O_3]\end{pmatrix} = \begin{pmatrix} k_1[O_3][O_2] - k_2[O][O_2]^2 - k_3[O_3][O] \\ -k_1[O_3][O_2] + k_2[O][O_2]^2 + k_3[O_3][O] \\ -k_1[O_3][O_2] + k_2[O][O_2]^2 - k_3[O_3][O] \end{pmatrix}$$

ODE Algorithms

Forward Euler



$$\frac{du}{dt} = f(t, u)$$

given $u(t)$, t, compute <u>slope</u> $f(t,u)$ of the solution curve and move in that direction with step length h.

$$(t_n, \vec{u}_n) \longrightarrow (t_{n+1}, \vec{u}_{n+1})$$

$$t_{n+1} = t_n + h$$

$$\vec{u}_{n+1} = \vec{u}_n + h\, \vec{f}(t_n, \vec{u}_n)$$

<u>Ex:</u> linear system $\vec{f}(t, \vec{u}) = A\vec{u}$

$$\rightarrow \quad \vec{u}_{n+1} = \vec{u}_n + h A \vec{u}_n$$

$$= (\mathbb{1} + hA)\, \vec{u}_n$$

$\rightsquigarrow$ single matrix-vector multiplication ($\mathcal{O}(n)$ operations for sparse A)

rror analysis

let $u(t)$ be the true solution to

$$\frac{du}{dt} = f(t,u), \quad u(t_0) = u_0$$

Taylor - expand:

$$u(t) = \underbrace{u(t_0)}_{=u_0} + \underbrace{(t-t_0)}_{h}\underbrace{u'(t_0)}_{=f(t_0,u_0)} + \frac{1}{2}(t-t_0)^2 u''(t_0) + \dots$$

$$\Rightarrow \quad = \underbrace{u_0 + h f(t_0,u_0)}_{= u^{Euler}(t_0+h)} + \frac{1}{2}h^2 u''(t_0) + \dots$$

$\Rightarrow$ error $\quad u(t+h) - u^{Euler}(t+h) = \frac{1}{2}h^2 u''(t_0) + \dots$

$$\sim h^2$$

each step of the Euler method has error $\sim h^2$

on an interval $[t_0, t_1]$ subdivided into N intervals,

$N = \frac{t_b - t_a}{h}$ the total error accumulates

$$\boxed{\text{error} \sim N h^2 \sim h}$$

$\rightarrow$ "order-1 method"

<u>Improved Euler method</u>

<u>Idea:</u> use a better estimate for the slope

$$\frac{1}{2}(s + s'), \quad s = \vec{f}(t_n, u_n)$$

$$s' = f(t_{n+1}, u_{n+1}^{Euler})$$

$$t_{n+1} = t_n + h$$

$$\vec{u}_{n+1} = \vec{u}_n + \frac{h}{2}\left(\vec{f}(t_n, \vec{u}_n) + \vec{f}(t_{n+1}, \vec{u}_{n+1}^{Euler}) \right)$$

$$\vec{u}_{n+1}^{Euler} = \vec{u}_n + h \vec{f}(t_n, \vec{u}_n)$$

rror analysis

$$u(t_0+h) = \underbrace{u(t_0)}_{=u_0} + h\underbrace{u'(t_0)}_{=f(t_0,u_0)} + \frac{h^2}{2}u''(t_0) + \frac{h^3}{6}u'''(t_0) + \dots$$

valuate

$$u''(t_0) = \frac{d}{dt}u'(t_0)$$

$$= \frac{d}{dt}f(t_0,u_0)$$

$$= \underbrace{\frac{\partial f}{\partial t}\Big|_{t_0,u_0}}_{\equiv f_t} + \underbrace{\frac{du}{dt}\Big|_{t_0,u_0}}_{=f(t_0,u_0)} \underbrace{\frac{\partial f}{\partial u}\Big|_{t_0,u_0}}_{\equiv f_u}$$

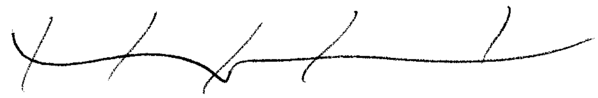
$$\Rightarrow u(t_0+h) = u_0 + h f(t_0,u_0) + \frac{h^2}{2}\left(f_t + f_u f(t_0,u_0)\right) + \mathcal{O}(h^3)$$

nd improved Euler:

$$u^{IE}(t_0+h) = u_0 + \frac{h}{2}\Big(f(t_0,u_0) + f(t_0+h, u_0 + h f(t_0,u_0))\Big)$$

$$= u_0 + \frac{h}{2}\left(f_0 + f_0 + h f_t + h f_0 f_u\right) + \mathcal{O}(h^3)$$

$$= u_0 + h f(t_0,u_0) + \frac{h^2}{2}\left(f_t + f(t_0,u_0) f_u\right) + \mathcal{O}(h^3)$$



$$u(t_0+h) - u^{IE}(t_0+h) = \mathcal{O}(h^3)$$

$\Rightarrow$ Error per step $\sim h^3$

$\Rightarrow$ total error $\sim h^2$

$\Rightarrow$ improved Euler is an $\boxed{\text{order-2 method}}$