Numerical methods for dynamical systems

Julien Alexandre dit Sandretto

Department U2IS
ENSTA Paris

INF6561 2022-2023

Contents

Initial Value Problem of Ordinary Differential Equatic

Consider an IVP for ODE, over the time interval $[0, t_{\text{end}}]$

$$\dot{\mathbf{y}} = f(t, \mathbf{y}) \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0$$

IVP has a unique solution $\mathbf{y}(t; \mathbf{y}_0)$ if $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz in $\mathbf{y}$

$$\forall t, \forall \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^n, \exists L > 0, \quad \| f(t, \mathbf{y}_1) - f(t, \mathbf{y}_2) \| \leq L \| \mathbf{y}_1 - \mathbf{y}_2 \| \quad .$$

Goal of numerical integration

- Compute a sequence of time instants: $t_0 = 0 < t_1 < \cdots < t_n = t_{\text{end}}$
- Compute a sequence of values: $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n$ such that

$$\forall \ell \in [0, n], \quad \mathbf{y}_\ell \approx \mathbf{y}(t_\ell; \mathbf{y}_0) \ .$$

- s.t. $\mathbf{y}_{\ell+1} \approx \mathbf{y}(t_\ell + h; \mathbf{y}_\ell)$ with an error $\mathcal{O}(h^{p+1})$ where
  - h is the integration **step-size**
  - p is the **order** of the method

Simulation algorithm

```
Data: $f$ the flow, $\mathbf{y}_0$ initial condition, $t_0$ starting time, $t_{\text{end}}$ end
      time, $h$ integration step-size
$t \leftarrow t_0$  $\mathbf{y} \leftarrow \mathbf{y}_0$  while $\underline{t < t_{\text{end}}}$ do
    Print($t$, $\mathbf{y}$)  $y \leftarrow$ Euler($f$,$t$,$\mathbf{y}$,$h$)  $t \leftarrow t + h$
end
```

with, the Euler's method defined by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + hf(t_n, \mathbf{y}_n) \quad \text{and} \quad t_{n+1} = t_n + h \ .$$

Examples of Runge-Kutta methods

Single-step fixed step-size explicit Runge-Kutta method

e.g. explicit Trapezoidal method (or Heun's method)[1] is defined by:

$$\mathbf{k}_1 = f(t_\ell, \mathbf{y}_\ell)\ , \qquad \mathbf{k}_2 = f(t_\ell + 1h, \mathbf{y}_\ell + h1\mathbf{k}_1)$$

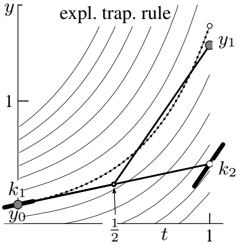
$$\mathbf{y}_{i+1} = \mathbf{y}_\ell + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right)$$

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Intuition

- $\dot{y} = t^2 + y^2$
- $y_0 = 0.46$
- $h = 1.0$

dotted line is the exact solution.



[1] example coming from "Geometric Numerical Integration", Hairer, Lubich and Wanner, 2006.

Examples of Runge-Kutta methods

Single-step variable step-size explicit Runge-Kutta method

e.g. Bogacki-Shampine (`ode23`) is defined by:

$$
\begin{aligned}
\mathbf{k}_1 &= f(t_n, \mathbf{y}_n)\\
\mathbf{k}_2 &= f\left(t_n + \frac{1}{2}h_n, \mathbf{y}_n + \frac{1}{2}h\mathbf{k}_1\right)\\
\mathbf{k}_3 &= f\left(t_n + \frac{3}{4}h_n, \mathbf{y}_n + \frac{3}{4}h\mathbf{k}_2\right)\\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\left(\frac{2}{9}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{4}{9}\mathbf{k}_3\right)\\
\mathbf{k}_4 &= f(t_n + 1h_n, \mathbf{y}_{n+1})\\
\mathbf{z}_{n+1} &= \mathbf{y}_n + h\left(\frac{7}{24}\mathbf{k}_1 + \frac{1}{4}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{8}\mathbf{k}_4\right)
\end{aligned}
$$

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{3}{4} & 0 & \frac{3}{4} & & \\
1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
\hline
 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
 & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8}
\end{array}
$$

Remark: the step-size h is adapted following $\parallel \mathbf{y}_{n+1} - \mathbf{z}_{n+1} \parallel$

[1] example coming from “Geometric Numerical Integration”, Hairer, Lubich and Wanner, 2006.

Examples of Runge-Kutta methods

Single-step fixed step-size implicit Runge-Kutta method

e.g. Runge-Kutta Gauss method (order 4) is defined by:

$$\mathbf{k}_1 = f\left(t_n + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)h_n, \quad \mathbf{y}_n + h\left(\frac{1}{4}\mathbf{k}_1 + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)\mathbf{k}_2\right)\right) \tag{1a}$$

$$\mathbf{k}_2 = f\left(t_n + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)h_n, \quad \mathbf{y}_n + h\left(\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right)\mathbf{k}_1 + \frac{1}{4}\mathbf{k}_2\right)\right) \tag{1b}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right) \tag{1c}$$

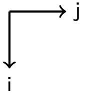
Remark: A non-linear system of equations must be solved at each step.

[1] example coming from “Geometric Numerical Integration”, Hairer, Lubich and Wanner, 2006.

Runge-Kutta methods

s-stage Runge-Kutta methods are described by a Butcher tableau:

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
\vdots & \vdots & \vdots & & \vdots \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s \\
 & b'_1 & b'_2 & \cdots & b'_s & \text{(optional)}
\end{array}$$



Which induces the following recurrence formula:

$$\mathbf{k}_i = f\left(t_n + c_i h_n, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right) \qquad \mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i\mathbf{k}_i \qquad (2)$$

- **Explicit** method (ERK) if $a_{ij} = 0$ is $i \leq j$
- **Diagonal Implicit** method (DIRK) if $a_{ij} = 0$ is $i \leq j$ and at least one $a_{ii} \neq 0$
- **Implicit** method (IRK) otherwise

Building RK methods: Order condition

Every numerical method member of the Runge-Kutta family follows the *condition order*.

Order condition

This condition states that a method of this family is of order p if and only if the $p+1$ first coefficients of the Taylor expansion of the true solution and the Taylor expansion of the numerical methods are equal.

In other terms, a **RK method has order** p if

$$\mathbf{y}(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n = h^{p+1}\Psi_f(\mathbf{y}_n) + \mathcal{O}(h^{p+2})$$

Building RK methods: Order condition

Taylor expansion of the exact and the numerical solutions

- At a time instant t_n the Taylor expansion of the true solution with the Lagrange remainder states that there exists $\xi \in]t_n, t_{n+1}[$ such that:

$$\begin{aligned}\mathbf{y}(t_{n+1}; \mathbf{y}_0) &= \mathbf{y}(t_n; \mathbf{y}_0) + \sum_{i=1}^{p} \frac{h_n^i}{i!} \mathbf{y}^{(i)}(t_n; \mathbf{y}_0) + \mathcal{O}(h^{p+1}) \\ &= \mathbf{y}(t_n; \mathbf{y}_0) + \sum_{i=1}^{p} \frac{h_n^i}{i!} f^{(i-1)}\left(t_n, \mathbf{y}(t_n; \mathbf{y}_0)\right) + \mathcal{O}(h^{p+1})\end{aligned}$$

- The Taylor expansion (very complex expression) of the numerical solution is given by expanding, around $(t_n, \mathbf{y}_n)$, the expression:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i$$

Consequence of the condition order

The definition of RK methods (Butcher table coefficients) is based on the solution of under-determined system of algebraic equations.

Example: 3-stages explicit RK method (scalar IVP)

One considers a scalar ODE $\dot{y} = f(t, y)$ with $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
One tries to determine the coefficients b_i $(i = 1, 2, 3)$, c_2, c_3, a_{32} such that

$$\begin{aligned} y_{n+1} &= y_n + h\left(b_1 k_1 + b_2 k_2 + b_3 k_3\right) \\ k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + c_2 h, y_n + h c_2 k_1) \\ k_3 &= f(t_n + c_3 h, y_n + h(c_3 - a_{32}) k_1 + h a_{32} k_2 \end{aligned}$$

Some notations (evaluation at point $(t_n, y(t_n))$):

$$f = f(t, y) \quad f_t = \frac{\partial f(t, y)}{\partial t} \quad f_{tt} = \frac{\partial^2 f(t, y)}{\partial t^2} \quad f_{ty} = \frac{\partial f(t, y)}{\partial t \partial y} \quad \dots$$

Note: in Butcher tableau we always have the *row-sum condition*

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, 2, \dots, s \ .$$

Example: 3-stages explicit RK method (scalar IVP)

Taylor expansion of $y(t_{n+1})$, the exact solution, around t_n:

$$y(t_{n+1}) = y(t_n) + hy^{(1)}(t_n) + \frac{h^2}{2}y^{(2)}(t_n) + \frac{h^3}{6}y^{(3)}(t_n) + \mathcal{O}(h^4)$$

Moreover,

$$\begin{aligned} y^{(1)}(t_n) &= f \\ y^{(2)}(t_n) &= f_t + f_y\dot{y} = f_t + ff_y \\ y^{(3)}(t_n) &= f_{tt} + f_{ty}f + f(f_{ty} + f_{yy}f) + f_y(f_y + ff_y) \\ &= f_{tt} + 2ff_{ty} + f^2 f_{yy} + f_y(f_t + ff_y) \end{aligned}$$

With $F = f_t + ff_y$ and $G = f_{tt} + 2ff_{ty} + f^2 f_{ty}$, one has:

$$y(t_{n+1}) = y(t_n) + hf + \frac{h^2}{2}F + \frac{h^3}{6}(Ff_y + G) + \mathcal{O}(h^4)$$

Example: 3-stages explicit RK method (scalar IVP)

Taylor expansion k_i around t_n

$$\begin{aligned}
k_2 &= f + hc_2\left(f_t + k_1 f_y\right) + \frac{h^2}{2}c_2^2\left(f_{tt} + 2k_1 f_{ty} + k_1^2 f_{yy}\right) + \mathcal{O}(h^3)\\
&= f + hc_2 F + \frac{h^2}{2}c_2^2 G + \mathcal{O}(h^3)\\
k_3 &= f + h\left\{c_3 f_t + \left[(c_3 - a_{32})k_1 + a_{32}k_2\right] f_y\right\}\\
&\quad + \frac{h^2}{2}\Big\{c_3^2 f_{tt} + 2c_3\left[(c_3 - a_{32})k_1 + a_{32}k_2\right] f_{ty}\\
&\quad + \left[(c_3 - a_{32})k_1 + a_{32}k_2\right]^2 f_{yy}\Big\} + \mathcal{O}(h^3)\\
&= f + hc_3 F + h^2(c_2 a_{32} F f_y + \frac{1}{2}c_3^2 G + \mathcal{O}(h^3) \quad \text{(substituting } k_1 = f \text{ and } k_2\text{)}
\end{aligned}$$

Taylor expansion of y_{n+1} (localizing assumption $y_n = y(t_n)$)

$$\begin{aligned}
y_{n+1} = y(t_n) + h(b_1 + b_2 + b_3)f + h^2(b_2 c_2 + b_3 c_3)F&\\
+ \frac{h^3}{2}\left[2b_3 c_2 a_{32} F f_y + (b_2 c_2^2 + b_3 c_3^2)G\right] + \mathcal{O}(h^4)&
\end{aligned}$$

Example: 3-stages explicit RK method (scalar IVP)

Building one stage method

We fix $b_2 = b_3 = 0$, so one gets

$$y_{n+1} = y(t_n) + hb_1 f + \mathcal{O}(h^2)$$

In consequence $b_1 = 1$ (by identification) so one gets Euler's method (order 1)

Example: 3-stages explicit RK method (scalar IVP)

Building two stages method

We fix $b_3 = 0$, so one gets

$$y_{n+1} = y(t_n) + h(b_1 + b_2)f + h^2 b_2 c_2 F + \frac{1}{2} h^3 b_2 c_2^2 G + \mathcal{O}(h^3)$$

In consequence to get order 2 methods, we need to solve

$$b_1 + b_2 = 1 \qquad b_2 c_2 = \frac{1}{2}$$

Remark: there is a (singly) infinite number of solutions.
Two particular solutions of order 2:

$$\begin{array}{c|cc} 0 & & \\ 1 & \frac{1}{2} & \\ \hline & 0 & 1 \end{array} \qquad\qquad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Example: 3-stages explicit RK method (scalar IVP)

Building three stages method

In consequence to get order 3 methods, we need to solve

$$b_1 + b_2 + b_3 = 1 \qquad b_2 c_2 + b_3 c_3 = \frac{1}{2}$$

$$b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3} \qquad b_3 c_2 a_{32} = \frac{1}{6}$$

Remark: there is a (doubly) infinite number of solutions.
Two particular solutions of order 3:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array} \qquad \begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Relation between order and stage

Limitation of ERK

s-stage ERK method cannot have order greater than s

Moreover, this upper bound is reached for order less or equal to 4. For now, we only know:

order	1	2	3	4	5	6	7	8	9	10
s	1	2	3	4	6	7	9	11	$[12, 17]$	$[13, 17]$
cond	1	2	4	8	17	37	85	200	486	1205

Remark: order 10 is highest order known for an ERK (with rational coefficients).

Optimal order for IRK methods

We know s-stage method having order $2s$ (Gauss' methods).

Note on building IRK Gauss' method

$$\dot{\mathbf{y}} = f(\mathbf{y}) \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0 \Leftrightarrow \mathbf{y}(t) = \mathbf{y}_0 + \int_{t_n}^{t_{n+1}} f(\mathbf{y}(s))ds$$

We solve this equation using quadrature formula.
IRK Gauss method is associated to a collocation method (polynomial approximation of the integral) such that for $i, j = 1, \ldots, s$:

$$a_{ij} = \int_0^{c_i} \ell_j(t)dt \qquad \text{and} \qquad b_j = \int_0^1 \ell_j(t)dt$$

with $\ell_j(t) = \prod_{k\neq j} \frac{t-c_k}{c_j-c_k}$ the Lagrange polynomial.
And the c_i are chosen as the solution of the Shifted Legendre polynomial of degree s:

$$P_s(x) = (-1)^s \sum_{k=0}^{s} \binom{s}{k} \binom{s+k}{s} (-x)^k$$

1, x, $0.5(3x^2 - 1)$, $0.5(5x^3 - 3x)$, etc.

Example (order 3): Radau family $(2s-1)$

Based on different polynomial, one can build different IRK methods with a particular structure. For examples, Radau family consider as endpoints of time interval either 0 or 1.

Radau IA (0 endpoint)

$$\begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$$

Radau IIA (1 endpoint)

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

Example (order 4): Lobatto family ($2s-2$)

Based on different polynomial, one can build different IRK methods with a particula structure. For examples, Lobatto family always consider 0 and 1 as endpoints of time interval.

Lobatto IIIA

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Lobatto IIIB

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} & 0 \\ 1 & \frac{1}{6} & \frac{5}{6} & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Lobatto IIIC

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Local error estimation in ERK

Goal: monitoring the step length to

- increase it if the norm of the error is below a given tolerance (with a factor)
- decrease it if the norm of the error is above a given tolerance

The trade-off between Accuracy versus Performance
An old solution: *Richardson extrapolation*, from a method of order p:

- solve the IVP for one step h to get $\tilde{\mathbf{y}}_n$
- solve the IVP for two steps $h/2$ to get $\tilde{\mathbf{z}}_n$
- error estimation if given by: $(\tilde{\mathbf{z}}_n - \tilde{\mathbf{y}}_n)/(2^{p+1} - 1)$

Drawback: time consuming

Other approach

embedding two ERK (or suitable IRK) methods of order p and $p + 1$ and compute the difference, as

$$\mathbf{y}^*_{n+1} - \mathbf{y}_{n+1} = [\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}] - [\mathbf{y}(t_{n+1}) - \mathbf{y}^*_{n+1}] = h^{p+1}\Psi_f(\mathbf{y}_n) + \mathcal{O}(h^{p+2})$$

with $\mathbf{y}_{n+1}$ solution of order p and $\mathbf{y}^*_{n+1}$ solution of order $p^* > p$

Example: explicit Runge-Kutta-Fehlberg (RKF45)

Fehlberg's method of order 4 and 5

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
 & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\
 & \frac{16}{135} & 0 & -\frac{128}{4275} & -\frac{2197}{75240} & \frac{1}{50} & \frac{2}{55}
\end{array}
$$

Remark

- coefficient chosen to minimize the coefficient of the Taylor expansion remainder

Example: explicit DOPRI54

Dormand-Prince's method of order 5 and 4

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{55} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40} \\
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0
\end{array}
$$

Remarks

- 7 stage for an order 5 method but **FSAL (First Same As Last)**
- *Local extrapolation* order 5 approximation is used to solve the next step

Example (order 4): SDIRK Family

$$
\begin{array}{c|ccccc}
\frac{1}{4} & \frac{1}{4} & & & & \\
\frac{3}{4} & \frac{1}{2} & \frac{1}{4} & & & \\
\frac{1}{20} & \frac{17}{50} & -\frac{1}{25} & \frac{1}{4} & & \\
\frac{1}{2} & \frac{371}{1360} & -\frac{137}{2720} & \frac{15}{544} & \frac{1}{4} & \\
1 & \frac{25}{24} & -\frac{49}{48} & \frac{125}{16} & -\frac{85}{12} & \frac{1}{4} \\
\hline
 & \frac{25}{24} & -\frac{49}{48} & \frac{125}{16} & -\frac{85}{12} & \frac{1}{4} \\
 & \frac{59}{48} & -\frac{17}{96} & \frac{225}{32} & -\frac{85}{12} & 0
\end{array}
$$

Remarks:

- this an embedded implicit RK method (difficult to find one for IRK)
- simplification of the iteration to solve the fixpoint equations

Step size control - simple case

Simple strategy:

$$\text{err} = \| \mathbf{y}_{n+1} - \mathbf{z}_{n+1} \|$$

with $\mathbf{y}_{n+1}$ the approximation of order p and $\mathbf{z}_{n+1}$ the approximation of order $p+1$.

Simple step-size update strategy

From an user defined tolerance TOL:

- if err $>$ TOL then solve IVP with $h/2$
- if err $\leq \frac{\text{TOL}}{2^{p+1}}$ then solve IVP with $2h$

Step size control - more evolved case

Validation of the integration step

For adaptive step-size method: for all continuous state variables

$$\text{err} = \underbrace{\| \mathbf{y}_{n+1} - \mathbf{z}_{n+1} \|}_{\text{Estimated error}} \leq \underbrace{\max\left(\text{atol}, \text{rtol} \times \max\left(\| \mathbf{y}_{n+1} \|, \| \mathbf{y}_n \|\right)\right)}_{\text{Maximal acceptable error}} .$$

Note: different norms can be considered.

Strategy:

- **Success:** may increase the step-size: $h_{n+1} = h_n \sqrt[p+1]{1/\mathrm{err}}$ (p is the minimal order of the embedded methods).
- **Failure:** reduce the step-size h_n in general only a division by 2, and restart the integration step with the new step-size.

Remark

The reduction of the step-size is done until the h_{min} is reached. In that case a simulation error may happen.

More details on the step-size control

Some care is necessary to reduce probability the next step is rejected:

$$h_{n+1} = h_n \min\left(\text{facmax}, \max\left(\text{facmin}, \text{fac}\sqrt[p+1]{1/\text{err}}\right)\right)$$

and to prevent that h is increased or decreased too quickly.
Usually:

- $\text{fac} = 0.8, 0.9, (0.25)^{1/(p+1)}, (0.38)^{1/(p+1)}$
- facmax is between 1.5 and 5
- facmin is equal to 0.5

Note

after a rejection (i.e., a valid step coming from a reject step) it is advisable to let h unchanged.

Implicit Runge-Kutta Methods

The $\mathbf{k}_i$, $i = 1, \ldots, s$, form a nonlinear system of equations in,

$$\mathbf{k}_i = f\left(t_n + c_i h_n, \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j\right) \qquad \mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i$$

Theorem: existence and uniqueness of the solution

Let $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ be continuous and satisfy a Lipschitz conditions with constant L (w.r.t. $\mathbf{y}$). If

$$h < \frac{1}{L \max_i \sum_j \mid a_{ij} \mid}$$

there exists a unique solution which can be obtained by iteration.

Remark: in case of stiff problems (see lecture on DAE), larger value of L is bad has it may cause numerical instability in iterations.

Quick remainder on Newton-Raphson methods

Goal of the method

finding zeroes of non-linear continuously differentiable functions $G : \mathbb{R}^n \to \mathbb{R}^n$

Based on the idea (in 1D) to approximate non-linear function by its tangent equation and starting from a sufficiently close solution $\mathbf{x}_0$ we can produce an approximation $\mathbf{x}_1$ closer to the solution, such that

$$\mathbf{x}_1 = \mathbf{x}_0 - J_G^{-1}(\mathbf{x}_0)G(\mathbf{x}_0)$$

where J_G is the Jacobian matrix of G. This process is repeated until we are close enough
Usually instead of computing inverse of matrices, it is more efficient to solve the linear system (e.g., with LU decomposition)

$$J_G(\mathbf{x}_0)\delta_x = -G(\mathbf{x}_0) \quad \text{with unknown} \quad \delta_x = \mathbf{x}_1 - \mathbf{x}_0$$

and so $\mathbf{x}_1 = \mathbf{x}_0 + \delta_x$

Reformulating non-linear system of $\mathbf{k}_i$'s

Solution of the nonlinear system of equations using Newton's method: first we can rewrite the system:

$$\mathbf{k}_i = f\left(t_n + c_i h_n, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right)$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i\mathbf{k}_i$$

with $\mathbf{k}'_i = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j$ into

$$\mathbf{k}'_i = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij} f(t_n + c_i h_n, \mathbf{k}'_j)$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{j=1}^{s} b_i f(t_n + c_i h_n, \mathbf{k}'_j)$$

Reformulating non-linear system of $\mathbf{k}_i$'s

Next, let $\mathbf{z}_i = \mathbf{k}_i' - \mathbf{y}_n$ we have:

$$\begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_s \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1s} \\ \vdots & \ddots & \vdots \\ a_{s1} & \cdots & a_{ss} \end{pmatrix} \begin{pmatrix} hf(t_n + c_1 h_n, \mathbf{y}_n + \mathbf{z}_1) \\ \vdots \\ hf(t_n + c_s h_n, \mathbf{y}_n + \mathbf{z}_s) \end{pmatrix} \tag{3}$$

$$\mathbf{z} = hAF(\mathbf{z})$$

hence, with $\mathbf{z}^k$ the solution of Equation (3):

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \sum_{i=1}^{s} d_i \mathbf{z}_i^k \quad \text{with} \quad (d_1, \cdots, d_s) = (b_1, \cdots, b_s) A^{-1}$$

with $A = \{a_{ij}\}$ if A is invertible (it is the case for Gauss' method).

Reformulating non-linear system of k_i's

Now we have to solve:

$$g(\mathbf{z}) = 0 \quad \text{with} \quad g(\mathbf{z}) = \mathbf{z} - hAF(\mathbf{z})$$

with Newton's method where Jacobian matrix $\nabla g(\mathbf{z})$ of g is:

$$\nabla g(\mathbf{z}) = \begin{pmatrix} I - ha_{11}J(\mathbf{z}_1) & -ha_{12}J(\mathbf{z}_2) & \dots & -ha_{1s}J(\mathbf{z}_s) \\ -ha_{21}J(\mathbf{z}_1) & I - ha_{22}J(\mathbf{z}_2) & \dots & -ha_{2s}J(\mathbf{z}_s) \\ \vdots & \vdots & \ddots & \vdots \\ -ha_{1s}J(\mathbf{z}_1) & -ha_{2s}J(\mathbf{z}_2) & \dots & I - ha_{ss}J(\mathbf{z}_s) \end{pmatrix}$$

with $J(\mathbf{z}_i) = \frac{\partial f}{\partial \mathbf{y}}(t_n + c_i h_n, \mathbf{y}_n + \mathbf{z}_i)$. And the Newton iteration is defined by:

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \mathbf{p}_k \quad \text{with} \quad \mathbf{p}_k \text{ solution of } \nabla g(\mathbf{z}^k)\mathbf{p} = -g(\mathbf{z}^k)$$

Remarks: Usually we use $\frac{\partial f}{\partial \mathbf{y}}(t_n, \mathbf{y}_n) \approx \frac{\partial f}{\partial \mathbf{y}}(t_n + c_i h_n, \mathbf{y}_n + \mathbf{z}_i)$ and we have strategy to update the computation of $\nabla g(\mathbf{z})$

Implementation of fixed step size ERK

```python
def euler_one_step (f, t, y, h):
    return y + h * f(t, y)

def heun_one_step (f, t, y, h):
    y1 = euler_one_step (f,t,y,h)
    return y + h * 0.5 * ( f(t, y) + f(t+h, y1))

def solve (f, t0, y0, tend, nsteps):
    h = (tend - t0) / nsteps
    time = np.linspace(t0, tend, nsteps)
    yn = y0
    y = []
    print (h)
    for t in time:
        y.append(yn)
        # change the method here
        yn = heun_one_step (f, t, yn, h)
    return [ time, y]

def dynamic (t, y):
    return np.array([-y[1], y[0]])
```

Implementation of fixed step size IRK

```
def backward_euler_one_step (f, t, y, h):
    yn = y; err = 10000000
    while (err > 1e-14):
        yn1 = y + h * f(t + h, yn)
        err = LA.norm (yn1 - yn, 2)
        yn = yn1
    return yn1

def implicit_trapezoidal_one_step (f, t, y, h):
    aux = lambda yn :  y + h * 0.5 * (f(t, y) + f(t+h, yn)) - yn
    res = fsolve(aux, y)
    return res

def solve (f, t0, y0, tend, nsteps):
    h = (tend - t0) / nsteps
    time = np.linspace(t0, tend, nsteps)
    yn = y0; y = []
    for t in time:
        y.append(yn)
        yn = implicit_trapezoidal_one_step (f, t, yn, h)
    return [ time, y]

def dynamic (t, y):
    return np.array([-y[1], y[0]])

[t, y] = solve (dynamic, 0.0, np.array([1., 0.]), 2*np.pi*10, 100)
```

Implementation of variable step size ERK

```
def heun_euler_one_step (f, t, y, h):
    k1 = f(t, y); k2 = f(t + h, y + h * k1); ynp1 = y + h * 0.5 * (k1 + k2)
    znp1 = y + h * k1; err = ynp1 - znp1
    return (ynp1, err)

def compute_h (err, hn, order, tolerance):
    if LA.norm(err, 2) <= (tolerance / pow(2.0, order + 1)):
        return 2 * hn
    else:
        return hn

def solve (f, t0, y0, tend, tolerance):
    t = t0; yn = y0; hn = 0.5; y = [y0]; time = [t0]; h = [hn]
    while t <= tend:
        (yn, err) = heun_euler_one_step (f, t, yn, hn)
        if LA.norm(err, 2) <= tolerance:
            y.append(yn); t = t + hn; time.append(t)
            hn = compute_h (err, hn, 1, tolerance); h.append(hn)
        else:
            hn = hn / 2.0
    return [ time, y, h]

def dynamic (t, y):
    return np.array([-y[1], y[0]])

[t, y, h] = solve (dynamic, 0.0, np.array([1., 0.]), 2*np.pi*10, 1e-2)
```

Hamiltonian systems

We consider **conservative** (i.e., energy is preserved) Hamiltonian dynamical systems the form

$$H(q,p) = K(p) + V(q)$$

where H the Hamiltonian, K is the kinetic energy and V is the potential energy. And so can be write as

$$\begin{cases} \dfrac{dq}{dt} = \dfrac{\partial H}{\partial p} \\ \dfrac{dp}{dt} = -\dfrac{\partial H}{\partial q} \end{cases}$$

Classical example: harmonic oscillator

We have

$$H = \frac{1}{2}p^2 + \frac{1}{2}q^2$$

so

$$\begin{cases} \dfrac{dq}{dt} = p \\ \dfrac{dq}{dt} = -q \end{cases}$$

Symplectic Euler's method

- Applying directly explicit Euler's method on conservative Hamiltonian system cannot guaranteed the preservation of energy along the simulation.
- But we can do a small change to make the Euler's method symplectic i.e., energy preserving as

Solution 1

$$q_{n+1} = q_n + h\frac{\partial K}{\partial p}(p_n)$$
$$p_{n+1} = p_n + h\frac{\partial K}{\partial p}(q_{n+1})$$

Note: q has to be solved first
Solution 2

$$q_{n+1} = q_n + h\frac{\partial K}{\partial p}(p_{n+1})$$
$$p_{n+1} = p_n + h\frac{\partial K}{\partial p}(q_n)$$

Note: p has to be solved first
Note: In that case, it is a fixed step-size explicit order 1 method