Validated Explicit and Implicit Runge-Kutta Methods

Alexandre Chapoutot

joint work with Julien Alexandre dit Sandretto and Olivier Mullier
U2IS, ENSTA ParisTech

8th Small Workshop on Interval Methods, Praha
June 11, 2015

Initial Value Problem of Ordinary Differential Equations

Consider an IVP for ODE, over the time interval $[0, T]$

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

IVP has a unique solution $\mathbf{y}(t; \mathbf{y}_0)$ if $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz in $\mathbf{y}$
but for our purpose we suppose f smooth enough i.e., of class C^k

Goal of numerical integration

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

$$\forall i \in [0, n], \quad \mathbf{y}_i \approx \mathbf{y}(t_i; \mathbf{y}_0) \ .$$

- s.t. $\mathbf{y}_{n+1} \approx \mathbf{y}(t_n + h; \mathbf{y}_n)$ with an error $\mathcal{O}(h^{p+1})$ where
  - h is the integration **step-size**
  - p is the **order** of the method
  - true with *localization assumption* i.e., $\mathbf{y}_n = \mathbf{y}(t_n; y_0)$.

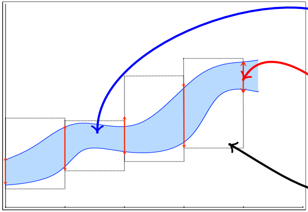
Validated solution of IVP for ODE

Goal of validated numerical integration

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

$$\forall i \in [0, n], \quad [\mathbf{y}_i] \ni \mathbf{y}(t_i; \mathbf{y}_0) \enspace .$$

A two-step approach



- Exact solution of $\dot{\mathbf{y}} = f(\mathbf{y}(t))$ with $\mathbf{y}(0) \in \mathcal{Y}_0$
- Safe approximation at discrete time instants
- Safe approximation between time instants

State of the art

Taylor methods

They have been developed since 60's (Moore, Lohner, Makino and Berz, Rhim, Jackson and Nedialkov, etc.)

- prove the existence and uniqueness: **high order interval Picard-Lindelöf**
- works very well on various kinds of problems:
  - **non stiff** and **moderately stiff** linear and non-linear systems,
  - with **thin uncertainties on initial conditions**
  - with (a writing process) **thin uncertainties on parameters**
- **very efficient** with automatic differentiation techniques
- **wrapping effect fighting**: interval centered form and QR decomposition
- **many software**: AWA, COSY infinity, VNODE-LP, CAPD, etc.

Some extensions

- Taylor polynomial with Hermite-Obreskov (Jackson and Nedialkov)
- Taylor polynomial in Chebyshev basis (T. Dzetkulic)

One question

Why bother to define new methods?

Answer 1: it may fail

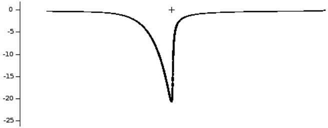
A chemical reaction simulated with VNODE-LP

$$\begin{cases} \dot{y} = z \\ \dot{z} = z^2 - \dfrac{3}{0.001 + y^2} \end{cases} \quad \text{with} \quad \begin{cases} y(0) = 10 \\ z(0) = 0 \end{cases} \quad \text{and} \quad t \in [0, 50]$$

Result: it is stuck around $t = 1$ with various order between 5 and 40.

With validated Lobatto-3C (order 4) method with tolerance 10^{-10}, we get in about 7.6s (Intel i7 3.4Ghz)

- $\text{width}(y_1(50.0)) = 7.67807 \cdot 10^{-5}$
- $\text{width}(y_2(50.0)) = 2.338 \cdot 10^{-6}$



Note: CAPD can solve this problem

Answer 2: there is no silver bullet

Numerical solutions of IVP for ODEs are produced by

- Adams-Bashworth/Moulton methods
- BDF methods
- Runge-Kutta methods
- etc.

each of these methods is adapted to a particular class of ODEs

Runge-Kutta methods

- have **strong stability** properties for various kinds of problems (A-stable, L-stable, algebraic stability, etc.)
- may preserve quadratic algebraic invariant (symplectic methods)
- can produce continuous output (polynomial approximation of $y(t)$)

Can we benefit these properties in validated computations?

Examples of Runge-Kutta methods

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

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

$$\mathbf{k}_1 = f(t_n, \mathbf{y}_n) \;, \qquad \mathbf{k}_2 = f(t_n + 1h_n, \mathbf{y}_n + h1\mathbf{k}_1)$$

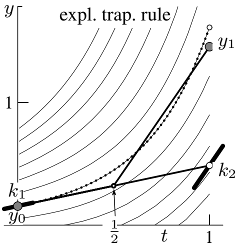
$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right)$$

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

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.

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

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

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

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

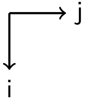
$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right) \tag{1c}$$

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

Runge-Kutta methods

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

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



Which induces the following recurrence:

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

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

Validated Runge-Kutta methods

Challenges

1. Computing with sets of values taking into account dependency problem and wrapping effect;
2. Bounding the approximation error of Runge-Kutta formula.

Our approach

- **Problem 1** is solved using **affine arithmetic** avoiding centered form and QR decomposition
- **Problem 2** is solved by bounding the **Local truncation error** of Runge-Kutta method based on **B-series**

We focus on Problem 2 in this talk

Order condition for Runge-Kutta methods

Method order of Runge-Kutta methods and Local Truncation Error (LTE)

$$\mathbf{y}(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n = C \cdot \mathcal{O}\left(h^{p+1}\right) \quad \text{with} \quad C \in \mathbb{R}.$$

we want to bound this!

Order condition

This condition states that a method of Runge-Kutta family is of order p **iff**

- the Taylor expansion of the exact solution
- and the Taylor expansion of the numerical methods

have the same $p+1$ first coefficients.

Consequence

The LTE is the **difference of Lagrange remainders of two Taylor expansions**

. . . but how to compute it?

A quick view of Runge-Kutta order condition theory[2]

Starting from $\mathbf{y}^{(q)} = (f(\mathbf{y}))^{(q-1)}$ and with the Chain rule, we have

High order derivatives of exact solution y

$$
\begin{aligned}
\dot{\mathbf{y}} &= f(\mathbf{y}) \\
\ddot{\mathbf{y}} &= f'(\mathbf{y})\dot{\mathbf{y}} && f'(\mathbf{y}) \text{ is a linear map} \\
\mathbf{y}^{(3)} &= f''(\mathbf{y})(\dot{\mathbf{y}}, \dot{\mathbf{y}}) + f'(\mathbf{y})\ddot{\mathbf{y}} && f''(\mathbf{y}) \text{ is a bi-linear map} \\
\mathbf{y}^{(4)} &= f'''(\mathbf{y})(\dot{\mathbf{y}}, \dot{\mathbf{y}}, \dot{\mathbf{y}}) + 3f''(\mathbf{y})(\ddot{\mathbf{y}}, \dot{\mathbf{y}}) + f'(\mathbf{y})\mathbf{y}^{(3)} && f'''(\mathbf{y}) \text{ is a tri-linear map} \\
\mathbf{y}^{(5)} &= f^{(4)}(\mathbf{y})(\dot{\mathbf{y}}, \dot{\mathbf{y}}, \dot{\mathbf{y}}, \dot{\mathbf{y}}) + 6f'''(\mathbf{y})(\ddot{\mathbf{y}}, \dot{\mathbf{y}}, \dot{\mathbf{y}}) && \vdots \\
&\quad + 4f''(\mathbf{y})(\mathbf{y}^{(3)}, \dot{\mathbf{y}}) + 3f''(\mathbf{y})(\ddot{\mathbf{y}}, \ddot{\mathbf{y}}) + f'(\mathbf{y})\mathbf{y}^{(4)} \\
&\vdots
\end{aligned}
$$

[2] strongly inspired from "Geometric Numerical Integration", Hairer, Lubich and Wanner.

A quick view of Runge-Kutta order condition theory[2]

Inserting the value of $\dot{\mathbf{y}}$, $\ddot{\mathbf{y}}$, ..., we have:

High order derivatives of exact solution y

$$
\begin{aligned}
\dot{\mathbf{y}} &= f \\
\ddot{\mathbf{y}} &= f'(f) \\
\mathbf{y}^{(3)} &= f''(f,f) + f'(f'(f)) \\
\mathbf{y}^{(4)} &= \ f'''(f,f,f) \ + \ {\color{red}3}f''(f'f,f) \ + \ f'(f''(f,f)) \ + \ f'(f'(f'(f))) \\
&\ \vdots
\end{aligned}
$$

- `Elementary differentials`, are denoted by $F(\tau)$

Remark a tree structure is made apparent in these computations

[2]strongly inspired from "Geometric Numerical Integration", Hairer, Lubich and Wanner.

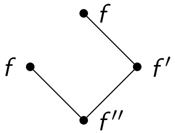
A quick view of Runge-Kutta order condition theory[2]

Rooted trees

- f is a leaf
- f' is a tree with one branch, ..., $f^{(k)}$ is a tree with k branches

Example

$f''(f'f, f)$ is associated to



Remark: this tree is not unique e.g., symmetry

[2] strongly inspired from "Geometric Numerical Integration", Hairer, Lubich and Wanner.

A quick view of Runge-Kutta order condition theory[2]

Theorem 1 (Butcher, 1963)

The qth derivative of the **exact solution** is given by

$$\mathbf{y}^{(q)} = \sum_{r(\tau)=q} \alpha(\tau) F(\tau)(\mathbf{y}_0) \quad \text{with} \quad \begin{array}{l} r(\tau) \text{ the order of } \tau \text{ i.e., number of nodes} \\ \alpha(\tau) \text{ a positive integer} \end{array}$$

We can do the same for the numerical solution

Theorem 2 (Butcher, 1963)

The qth derivative of the **numerical solution** is given by

$$\mathbf{y}_1^{(q)} = \sum_{r(\tau)=q} \gamma(\tau)\phi(\tau)\alpha(\tau) F(\tau)(\mathbf{y}_0) \quad \text{with} \quad \begin{array}{l} \gamma(\tau) \text{ a positive integer} \\ \phi(\tau) \text{ depending on a Butcher tableau} \end{array}$$

Theorem 3, order condition (Butcher, 1963)

A Runge-Kutta method has order p iff $\phi(\tau) = \frac{1}{\gamma(\tau)} \quad \forall \tau, r(\tau) \leqslant p$

[2] strongly inspired from "Geometric Numerical Integration", Hairer, Lubich and Wanner.

LTE formula for **explicit and implicit** Runge-Kutta

From Theorem 1 and Theorem 2, if a Runge-Kutta has order p then

$$\mathbf{y}(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n = \frac{h^{p+1}}{(p+1)!} \sum_{r(\tau)=p+1} \alpha(\tau) \left[1 - \gamma(\tau)\phi(\tau)\right] F(\tau)(\mathbf{y}(\xi))$$

$$\xi \in [t_{n-1}, t_n]$$

- $\alpha(\tau)$ and $\gamma(\tau)$ are positive integer (with some combinatorial meaning)
- $\phi(\tau)$ function of the coefficients of the RK method,

Example

$\phi\Big($  $\Big)$ is associated to $\displaystyle\sum_{i,j=1}^{s} b_i a_{ij} c_j$ with $\displaystyle c_j = \sum_{k=1}^{s} a_{jk}$

Note: $y(\xi)$ may be over-approximated using Interval Picard-Lindelöf operator.

Implementation of LTE formula

Elementary differentials

$$F(\tau)(y) = f^{(m)}(y)\,(F(\tau_1)(y), \ldots, F(\tau_m)(y)) \quad \text{for} \quad \tau = [\tau_1, \ldots, \tau_m]$$

translate as a sum of partial derivatives of f associated to sub-trees

Notations

- n the state-space dimension
- p the order of a Rung-Kutta method

Two ways of computing $F(\tau)$

1. **Direct form** (current): complexity $\mathcal{O}(n^{p+1})$
2. **Factorized form** (under test): complexity $\mathcal{O}(n(p+1)^{\frac{5}{2}})$
   based on the work of Ferenc Bartha and Hans Munthe-Kaas
   "Computing of B-series by automatic differentiation", 2014

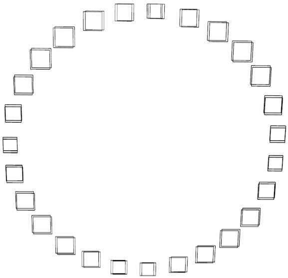
Experimentation

Toy example

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} -y_2 \\ y_1 \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} y_1(0) = [0, 0.1] \\ y_2(0) = [0.95, 1.05] \end{pmatrix}$$

Validated RK4 method with tolerance 10^{-8} we get in about 3s (Intel i7 3.4Ghz)

- $\mathrm{width}(y_1(100.0)) = 0.146808$
- $\mathrm{width}(y_2(100.0)) = 0.146902$



Experimentation

Usefulness of affine arithmetic

$$\begin{cases} \dot{y}_1 = 1, & y_1(0) = 0 \\ \dot{y}_2 = y_3, & y_2(0) = 0 \\ \dot{y}_3 = \dfrac{1}{6}y_2^3 - y_2 + 2\sin(p \cdot y_1) \quad \text{with} \quad p \in [2.78, 2.79], & y_3(0) = 0 \end{cases} .$$

Validated RK4 method with tolerance 10^{-6} we get in about 2.3s (Intel i7 3.4Ghz)

- $\text{width}(y_1(10.0)) = 7.10543 \cdot 10^{-15}$
- $\text{width}(y_2(10.0)) = 6.11703$
- $\text{width}(y_3(10.0)) = 7.47225$

Note: none of the method in the Vericomp benchmark can reach 10s
Note 2: CAPD can solve it

Experimentation

Based on Vericomp benchmark [3] (around 70 problems)

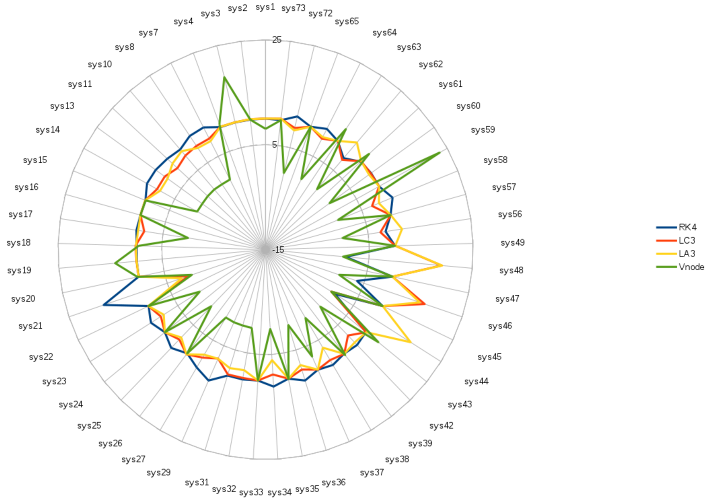
$$
\begin{tikzcd}
 & & & \text{simple (A)} \arrow[r] & \text{Uncertain (U) or not} \\
 & & \textcolor{blue}{\text{linear (L)}} \arrow[ur] \arrow[r] \arrow[dr] & \text{moderate (B)} \arrow[ur, dotted] & \\
\text{IVP} \arrow[r] & \text{non-stiff (P.I)} \arrow[ur] \arrow[dr] & & \text{complicate (C)} \arrow[uur, dotted] & \\
 & & \textcolor{orange}{\text{nonlinear } (\overline{L})} \arrow[r, dotted] & \text{idem} &
\end{tikzcd}
$$

with the following metrics:

- c5t: user time taken to simulate the problem for 1 second.
- c5w: the final diameter of the solution (infinity norm is used).
- c6t: the time to breakdown the method with a maximal limit of 10 seconds.
- c6w: the diameter of the solution a the breakdown time.

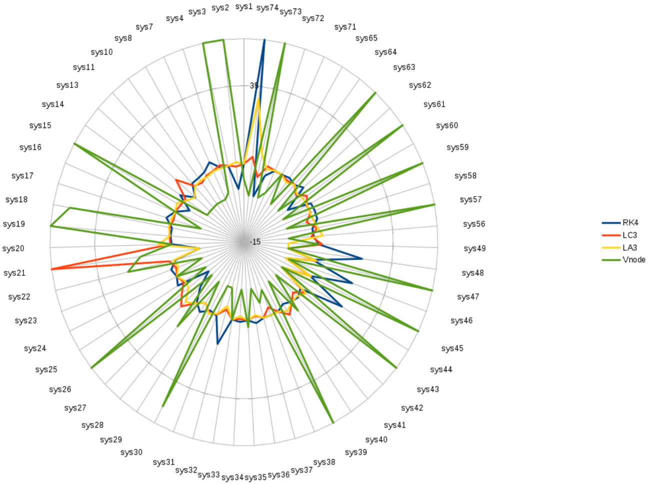
[3] `http://vericomp.inf.uni-due.de/`

Summary – RK vs Vnode-LP – c5w



- Vnode-LP: order 15, 20, 25 (tolerances 10^{-14})
- RK4, LC3, LA3: tolerances 10^{-8} to 10^{-14} (order 4)

Summary – RK vs Vnode-LP – c6w



- Vnode-LP: order 15, 20, 25 (tolerances 10^{-14})
- RK4, LC3, LA3: tolerances 10^{-8} to 10^{-14} (order 4)

Conclusion

We presented a new approach to validate Runge-Kutta methods

- a new formula to compute LTE based on B-series
- fully parametrized by a Butcher tableau
- affine arithmetic avoiding QR decomposition

implementation as a plugin of IBEX, code name DynIbex, available at
`http://perso.ensta-paristech.fr/~chapoutot/dynibex/`

Future work

- finish testing the implementation of LTE with automatic differentiation
- implement new *a priori* enclosure methods based on Runge-Kutta
- define new methods mixing different Runge-Kutta in one simulation
- solve new IVP problems such as for DAE (next talk) or DDE

BACKUP

Note on the number of trees (up to order 11 (left)):

```
Number of Rooted Trees
1842 719 286 115 48 20 9 4 2 1 1  (total 3047)
```

Quick remainder: Taylor series method

Taylor series development of $\mathbf{y}(t)$ (assume $\mathbf{y}(t_n) \in [\boldsymbol{y}_n]$)

$$\begin{aligned}
\mathbf{y}(t_{n+1}) &= \mathbf{y}(t_n) + \sum_{i=1}^{N-1} \frac{h^i}{i!} \frac{d^i \mathbf{y}}{dt^i}(t_n) + \frac{h_{n+1}^N}{N!} \frac{d^{\mathbf{Nx}}}{dt^N}(t') \\
&\in [\boldsymbol{y}_n] + \sum_{i=1}^{N-1} h^i f^{[i-1]}(\mathbf{y}(t_n)) + h^N f^{[N-1]}(\mathbf{y}(t')) \\
&\in [\boldsymbol{y}_n] + \sum_{i=1}^{N-1} h^i f^{[i-1]}([\boldsymbol{y}_n]) + h^N f^{[N-1]}([\tilde{\boldsymbol{y}}_n]) \triangleq [\boldsymbol{y}_{n+1}]
\end{aligned}$$

Challenges

- Computation of $[\tilde{\boldsymbol{y}}_n]$ such that $\forall t \in [t_n, t_{n+1}],\ \mathbf{y}(t) \in [\tilde{\boldsymbol{y}}_n]$
  Solution: interval Picard-Lindelöf operator
- With that formula: $\text{width}([\boldsymbol{y}_{n+1}]) \geqslant \text{width}([\boldsymbol{y}_n])$
  Solutions: interval centered form + QR decomposition

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

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

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

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

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

Gauss-Legendre methods

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

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

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

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

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right) \tag{3c}$$

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

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:

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

Example: 1, $2x - 1$, $6x^2 - 6x + 1$, $20x^3 - 30x^2 + 12x - 1$, etc.