Runge-Kutta Theory and Constraint Programming

Julien Alexandre dit Sandretto
Alexandre Chapoutot

Department U2IS
ENSTA ParisTech
SCAN 2016 - Uppsala

Numerical integration

Initial value problem

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

Time discretization $t_0 = 0 < t_1 < \cdots < t_n = T$

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. $\mathbf{y}_{n+1} \approx \mathbf{y}(t_n + h; \mathbf{y}_n) = \mathbf{y}_n + \int_0^h f(y(s))ds$

Tool: integration scheme to approx $\int_0^h f(y(s))ds$

with an error $\mathcal{O}(h^{p+1})$

- h is the integration **step-size**
- p is the **order** of the method (Taylor series or Runge-Kutta)

Numerical integration

Initial value problem

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

Time discretization $t_0 = 0 < t_1 < \cdots < t_n = T$

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. $\mathbf{y}_{n+1} \approx \mathbf{y}(t_n + h; \mathbf{y}_n) = \mathbf{y}_n + \int_0^h f(y(s))ds$

Tool: integration scheme to approx $\int_0^h f(y(s))ds$

with an error $\mathcal{O}(h^{p+1})$

- h is the integration **step-size**
- p is the **order** of the method (Taylor series or Runge-Kutta)

Numerical integration

Initial value problem

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

Time discretization $t_0 = 0 < t_1 < \cdots < t_n = T$

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. $\mathbf{y}_{n+1} \approx \mathbf{y}(t_n + h; \mathbf{y}_n) = \mathbf{y}_n + \int_0^h f(y(s))ds$

Tool: integration scheme to approx $\int_0^h f(y(s))ds$

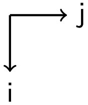
with an error $\mathcal{O}(h^{p+1})$

- h is the integration **step-size**
- p is the **order** of the method (Taylor series or Runge-Kutta)

Runge-Kutta schemes

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

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$



The integration scheme is given by:

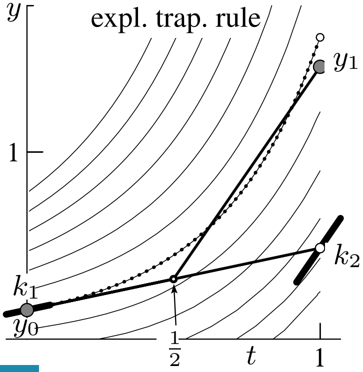
$$\mathbf{k}_i = f\left(t_n + c_i h_n, \quad \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right) \quad \mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{k}_i$$

- **Explicit** method (ERK): $a_{ij} = 0$ for $i \leqslant j$
- **Diagonal Implicit** method (DIRK): $a_{ij} = 0$ for $i < j$
- **Implicit** method (IRK) otherwise

Order p if $\mathbf{y}(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n = C \cdot h^{p+1}$ with $C \in \mathbb{R}$.

Runge-Kutta vs Taylor

Taylor: only one method computed for different order (till 120 !)
$\mathbf{y}_{n+1} = \mathbf{y}_n + \sum_{i=1}^{s} h^i f^{[i]}(\mathbf{y}_n) \Rightarrow$ computation from only one point !
Runge-Kutta: many methods with different order (often 4), computation from different points:



strong stability properties for various kinds of problems (A-stable, L-stable, algebraic stability, etc.), and may preserve quadratic algebraic invariant (symplectic methods)

Race for higher order schemes

Why higher order ?

High order implies low difference between solution and approximation !

A global competition

- Explicit Runge-Kutta order 14 with 35 stages with Maple [1]
- Radau order 17 with 9 stages with Mathematica [2]

[1] Feagin, Terry, "High-order Explicit Runge-Kutta Methods Using M-Symmetry", Neural, Parallel & Scientific Computations, Vol. 20, No. 4,December 2012, pp. 437-458
[2] J Martín-Vaquero, "A 17th-order Radau IIA method for package RADAU", Applications in mechanical systems, Computers & Mathematics with Applications, 2010

New scheme: a complex problem

Needs to solve constraints

High order polynomials (till p), number of constraints increases exponentially (4 for $p = 3$, 8 for $p = 4$, 17, 37, 85, 200)

1. $\sum_1^s b_i = 1$
2. $\sum_1^s b_i c_i = 1/2$
3. $\sum_1^s b_i c_i^2 = 1/3 \quad \sum_1^s \sum_1^s b_i a_{ij} c_j = 1/6$

Classical approach

Solve by using polynomials with known exact zeros such as Legendre (for Gauss) or Jacobi (for Radau) [see Butcher]

Problems

- Discovery of new methods guided by solver and not by requirements !
- Solved numerically: additive approximations !

New scheme: a complex problem

Needs to solve constraints

High order polynomials (till p), number of constraints increases exponentially (4 for $p=3$, 8 for $p=4$, 17, 37, 85, 200)

1. $\sum_1^s b_i = 1$
2. $\sum_1^s b_i c_i = 1/2$
3. $\sum_1^s b_i c_i^2 = 1/3 \quad \sum_1^s \sum_1^s b_i a_{ij} c_j = 1/6$

Classical approach

Solve by using polynomials with known exact zeros such as Legendre (for Gauss) or Jacobi (for Radau) [see Butcher]

Problems

- Discovery of new methods guided by solver and not by requirements !
- Solved numerically: additive approximations !

New scheme: a complex problem

Needs to solve constraints

High order polynomials (till p), number of constraints increases exponentially (4 for $p=3$, 8 for $p=4$, 17, 37, 85, 200)

1. $\sum_1^s b_i = 1$
2. $\sum_1^s b_i c_i = 1/2$
3. $\sum_1^s b_i c_i^2 = 1/3 \quad \sum_1^s \sum_1^s b_i a_{ij} c_j = 1/6$

Classical approach

Solve by using polynomials with known exact zeros such as Legendre (for Gauss) or Jacobi (for Radau) [see Butcher]

Problems

- Discovery of new methods guided by solver and not by requirements !
- Solved numerically: additive approximations !

Example of RK (35,14)

c[2]=.111111111111111111111111111111111111111111111 11111111111111111111111111111111111111111,
c[3]=.555555555555555555555555555555555555555555555 55555555555555555555555555555555555555556,
c[4]=.833333333333333333333333333333333333333333333 33333333333333333333333333333333333333333,
c[5]=.333333333333333333333333333333333333333333333 33333333333333333333333333333333333333333,
c[6]=1., c[7]=.669986979272772921764683785505998513938845229
6384603532851421391683474428303956826239, c[8]=.297068384213818357389584716808219413223332094
6989156873791682903324708698499266217383, c[9]=.727272727272727272727272727272727272727272727
2727272727272727272727272727272727272727, c[10]=.14015279904218876527618748796694671762980646
30825329362873230163439023340348096838456, c[11]=.70070103977015073715109985483074933794140704
92655464089692218490447945746638665522966, c[12]=.36363636363636363636363636363636363636363636
36363636363636363636363636363636363636364, c[13]=.26315789473684210526315789473684210526315789
47368421052631578947368421052631578947368, c[14]=.39217224665027085912519664250120864886371431
5266128052078483e-1, c[15]=.81291750292837676298339315927803650618961237
26172385507744269795906758195776958783707, c[16]=.16666666666666666666666666666666666666666666
66666666666666666666666666666666666666667, c[17]=.9,
c[18]=.64129925745196692331277119389668280948109665 16150832254029235721305050295351572963693e-1,
c[19]=.20414990928342884892774463430102340502714950 52413337516288702042649259099754335560687,
c[20]=.39535039104876056561567136982732437235222729 74566594505545766538389345381768585023057,
c[21]=.60464960895123943438432863017267562764777270 25433405494454233461610654618231414976943,
c[22]=.79585009071657115107225536569897659497285049 47586662483711297957350740900245664439313,
c[23]=.93587007425480330766872288061033171905189033 48384916774597076427869494970464842703631,
c[24]=.16666666666666666666666666666666666666666666 66666666666666666666666666666666666666667,
c[25]=.81291750292837676298339315927803650618961237 26172385507744269795906758195776958783707,
c[26]=.39217224665027085912519664250120864886371431 5266128052078483e-1,
c[27]=.36363636363636363636363636363636363636363636 36363636363636363636363636363636363636364,
c[28]=.70070103977015073715109985483074933794140704 92655464089692218490447945746638665522966,
c[29]=.14015279904218876527618748796694671762980646 30825329362873230163439023340348096838456,
c[30]=.29706838421381835738958471680821941322333209 46989156873791682903324708698499266217383,
c[31]=.66998697927277292176468378550599851393884522 96384603532851421391683474428303956826239, ...

$\Rightarrow \approx$ 40 slides...

Coefficients given in floating numbers

Problems:

Constraints not satisfied $\Rightarrow$ Method not at order p, but lower...

Validated integration (in a very short view):

Based on Local truncature error:
Validated bounds of $[lte] \triangleq \mathbf{y}(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n$, then
$\mathbf{y}(t_n; \mathbf{y}_{n-1}) \in \mathbf{y}_n + [lte]$

Coefficients given in floating numbers

Problems:

Constraints not satisfied $\Rightarrow$ Method not at order p, but lower...

Validated integration (in a very short view):

Based on Local truncature error:
Validated bounds of $[lte] \triangleq \mathbf{y}(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n$, then
$\mathbf{y}(t_n; \mathbf{y}_{n-1}) \in \mathbf{y}_n + [lte]$

Wrong with floating numbers !

Runge-Kutta with interval coefficients

A Runge-Kutta method of order p approximates a solution $\mathbf{y}(\mathbf{t})$ by computing its Taylor series expansion (till p) without any derivative computation.

Indeed

A method of order p is defined in the way to be equal to the sum of Taylor expansion till p^{th} term
$\Rightarrow$ the order conditions or also called Butcher rules.

With interval coefficients

These constraints are fulfilled by inclusion, then RK methods with interval coefficients are valid !

Runge-Kutta with interval coefficients

A Runge-Kutta method of order p approximates a solution $\mathbf{y}(\mathbf{t})$ by computing its Taylor series expansion (till p) without any derivative computation.

Indeed

A method of order p is defined in the way to be equal to the sum of Taylor expansion till p^{th} term
$\Rightarrow$ the order conditions or also called Butcher rules.

With interval coefficients

These constraints are fulfilled by inclusion, then RK methods with interval coefficients are valid !
But intervals have to be tight...

Properties are preserved

Main interest of Runge-Kutta w.r.t. Taylor series is the properties:

- Stability: linear, algebraic, etc.
- Symplecticity (conservation of energy)
- Structural properties: singly diagonal, explicit, diagonal implicit, explicit first line, stiffly accurate (easier to solve, better behavior, etc)

Properties are preserved

Main interest of Runge-Kutta w.r.t. Taylor series is the properties:

- Stability: linear, algebraic, etc.
- Symplecticity (conservation of energy)
- Structural properties: singly diagonal, explicit, diagonal implicit, explicit first line, stiffly accurate (easier to solve, better behavior, etc)

$\Rightarrow$ We want to preserve these properties !

Stability

"no analytical solution of a problem [...] numerical solutions [...] obtained for specified initial values. [...] stability behavior of the solutions for all initial values in the neighbourhood of a certain equilibrium point." [Hairer]

Example on linear problem

- $\dot{x} = \mathbf{A}x$, with exact solution: $x(t) = exp(\mathbf{A}t)x_0$
  Analytically stable if all trajectories remain bounded as $t \to \infty$
  $\Rightarrow$ If and only if $Re\{Eig(\mathbf{A})\} < 0$
- Euler: $x(t^* + h) \approx x(t^*) + \mathbf{A}hx(t^*) = (\mathbf{I} + \mathbf{A}h)x(t^*) = \mathbf{F}x(t^*)$
  Method is analytically stable if $x_{k+1} = \mathbf{F}x_k$ is analytically stable

Many classes of stability (A-, B-, A(α), Algebraic,...), linked to the problem (linear or not, stiff component or not, etc)

Linear Stability

Example of explicit methods (s=p) [Hairer]

$$R(z) = 1 + z\sum_j b_j + z^2 \sum_{j,k} b_j a_{jk} + z^3 \sum_{j,k,l} b_j a_{jk} a_{kl} + \dots$$

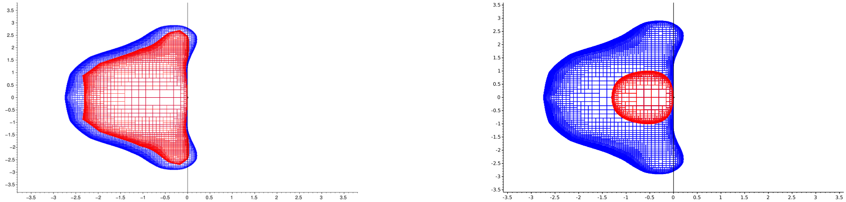
Stability domain given by $S = \{z \in \mathcal{C} : |R(z)| \leq 1\}$

For RK4: $R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}$

After $z = x + iy$, and some processing:

$$|R(x,y)| = \sqrt{(((((((((0.166667 * x^3) * y) + ((0.5 * x^2) * y)) - ((0.166667 * x) * y^3)) + ((1 * x) * y)) - (0.166667 * y^3)) + y)^2 + ((((((((0.0416667 * x^4) + (0.166667 * x^3)) - ((0.25 * x^2) * y^2)) + (0.5 * x^2)) - ((0.5 * x) * y^2)) + x) + (0.0416667 * y^4)) - (0.5 * y^2)) + 1)^2))} \leq 1$$

Linear Stability



Paving of stability domain for RK4 method with high precision coefficients (blue) and with error (10^{-8} and 10^{-2}) on coefficients (red).

Algebraically stable

Algebraically stable if:

- $b_i \geq 0$, for all $i = 1, \ldots, s$
- $M = (m_{ij}) = (b_i a_{ij} + b_j a_{ji} - b_i b_j)_{i,j=1}^{s}$ is non-negative definite

Problem to solve

Solving the eigenvalue problem $\det(A - \lambda I) = 0 \quad (1)$
and proving $\lambda > 0$.

For 3-stage Runge-Kutta methods:
$(m_{11} - \lambda) * ((m_{22} - \lambda) * (m_{33} - \lambda) - m_{23} * m_{32}) - m_{12} * (m_{21} * (m_{33} - \lambda) - m_{23} * m_{31}) + m_{13} * (m_{21} * m_{32} - (m_{22} - \lambda) * m_{31}) = 0$

With contractor programming (Fwd/Bwd + Newton)

Eq.(1) has no solution in $]-\infty, 0[\equiv M$ is non-negative definite.

Algebraically stable

Verification of theory

- Lobatto IIIC: contraction to empty set $\Rightarrow$ algebraically stable
- Lobatto IIIA: solution found $(-0.0481125) \Rightarrow$ not algebraically stable

With floating number

Lobatto IIIC with error of 10^{-9} on a_{ij}: solution found $(-1.03041 \cdot 10^{-05}) \Rightarrow$ not algebraically stable

Algebraically stable

Verification of theory

- Lobatto IIIC: contraction to empty set $\Rightarrow$ algebraically stable
- Lobatto IIIA: solution found (-0.0481125) $\Rightarrow$ not algebraically stable

With floating number

Lobatto IIIC with error of 10^{-9} on a_{ij}: solution found $(-1.03041 \cdot 10^{-05})$ $\Rightarrow$ not algebraically stable

Symplectic

Symplectic if $M = 0$, with $M = (m_{ij}) = (b_i a_{ij} + b_j a_{ji} - b_i b_j)_{i,j=1}^{s}$

Problem to solve

$0 \in [M]$ with interval arithmetic

Verification of theory with Gauss-Legendre:

$$M = 10^{-17} \cdot \begin{pmatrix} [-1.38, 1.38] & [-2.77, 2.77] & [-2.77, 1.38] \\ [-2.77, 2.77] & [-2.77, 2.77] & [-1.38, 4.16] \\ [-2.77, 1.38] & [-1.38, 4.16] & [-1.38, 1.38] \end{pmatrix}$$

With $a_{1,2} = 2.0/9.0 - \sqrt{15.0}/15.0$ computed with `float`

$$M = \begin{pmatrix} [-1.38e^{-17}, 1.38e^{-17}] & [\mathbf{-1.91e^{-09}}, \mathbf{-1.91e^{-09}}] & [-2.77e^{-17}, 1.38e^{-17}] \\ [\mathbf{-1.91e^{-09}}, \mathbf{-1.91e^{-09}}] & [-2.77e^{-17}, 2.77e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] \\ [-2.77e^{-17}, 1.38e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] & [-1.38e^{-17}, 1.38e^{-17}] \end{pmatrix}$$

Symplectic

Symplectic if $M = 0$, with $M = (m_{ij}) = (b_i a_{ij} + b_j a_{ji} - b_i b_j)_{i,j=1}^{s}$

Problem to solve

$0 \in [M]$ with interval arithmetic

Verification of theory with Gauss-Legendre:

$$M = 10^{-17} \cdot \begin{pmatrix} [-1.38, 1.38] & [-2.77, 2.77] & [-2.77, 1.38] \\ [-2.77, 2.77] & [-2.77, 2.77] & [-1.38, 4.16] \\ [-2.77, 1.38] & [-1.38, 4.16] & [-1.38, 1.38] \end{pmatrix}$$

With $a_{1,2} = 2.0/9.0 - \sqrt{15.0}/15.0$ computed with `float`

$$M = \begin{pmatrix} [-1.38e^{-17}, 1.38e^{-17}] & [\mathbf{-1.91e^{-09}}, \mathbf{-1.91e^{-09}}] & [-2.77e^{-17}, 1.38e^{-17}] \\ [\mathbf{-1.91e^{-09}}, \mathbf{-1.91e^{-09}}] & [-2.77e^{-17}, 2.77e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] \\ [-2.77e^{-17}, 1.38e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] & [-1.38e^{-17}, 1.38e^{-17}] \end{pmatrix}$$

Constraint approach to define new schemes

Consistency

- $c_i = \sum a_{ij}$ with $c_1 < \cdots < c_s$

Order conditions

1. $\sum b_i = 1$
2. $\sum b_i a_{ij} = 1/2$
3. $\sum c_i b_i a_{ij} = 1/6$, $\sum b_i c_i^2 = 1/3$
4. $\sum b_i c_i^3 = 1/4$, $\sum b_i c_i a_{ij} c_j = 1/8$, $\sum b_i a_{ij} c_i^2 = 1/12$, $\sum b_i a_{ij} a_{jk} c_k = 1/24$

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Properies by construction

- Singly diagonal: $a_{1,1} = \cdots = a_{s,s}$
- Explicit: $a_{ij} = 0, \forall j \geq i$
- Diagonal implicit: $a_{ij} = 0, \forall j > i$
- Explicit first line: $a_{1,1} = \cdots = a_{1,s} = 0$
- Stiffly accurate: $a_{s,i} = b_i, \forall i = 1, \ldots, s$

Constraint solver

User interface (Python-Sympy) to describe the desired method

- Choice of number of stages and the order (≤ 5)
- Choice of structure: Singly diagonal, Explicit method, DIRK method, Explicit first line and/or Stiffly accurate
- Generation of Constraint Satisfaction Problem

Solver Branch & Contract (Ibex)

- Contraction with Fwd/Bwd (or HC4)
- Bisection "largest first"

Re-discover the theory

Only one 2-stage method of order 4

```
Variables
b[2] in [-1,1];
c[2] in [0,1];
a[2][2] in [-1,1];
Constraints
b(1) +b(2) -1.0=0;
b(1)*c(1) +b(2)*c(2) -1.0/2.0=0;
b(1)*(c(1))^2 +b(2)*(c(2))^2 -1.0/3.0=0;
b(1)*a(1)(1)*c(1) +b(1)*a(1)(2)*c(2) +
    b(2)*a(2)(1)*c(1) +b(2)*a(2)(2)*c(2)
    -1.0/6.0=0;
b(1)*(c(1))^3 +b(2)*(c(2))^3 -1.0/4.0=0;
b(1)*c(1)*a(1)(1)*c(1) +b(1)*c(1)*a(1)(2)*c(2) +
    b(2)*c(2)*a(2)(1)*c(1) +b(2)*c(2)*a(2)(2)*c(2)
    -1.0/8.0=0;
b(1)*a(1)(1)*(c(1))^2 +b(1)*a(1)(2)*(c(2))^2 +
    b(2)*a(2)(1)*(c(1))^2 +b(2)*a(2)(2)*(c(2))^2
    -1.0/12.0=0;
b(1)*a(1)(1)*a(1)(1)*c(1) +b(1)*a(1)(1)*a(1)(2)*c(2) +
    b(1)*a(1)(2)*a(2)(1)*c(1) +b(1)*a(1)(2)*a(2)(2)*c(2) +
    b(2)*a(2)(1)*a(1)(1)*c(1) +b(2)*a(2)(1)*a(1)(2)*c(2) +
    b(2)*a(2)(2)*a(2)(1)*c(1) +b(2)*a(2)(2)*a(2)(2)*c(2)
    -1.0/24.0=0;
a(1)(1)+a(1)(2)-c(1) = 0; a(2)(1)+a(2)(2)-c(2) = 0;
c(1) < c(2);
end
```

```
number of solutions=1
cpu time used=0.013073s.
([0.5, 0.5] ; [0.5, 0.5] ;
[0.2113248, 0.2113248] ;[0.788675, 0.788675] ;
[0.25, 0.25] ; [-0.038675, -0.038675] ;
[0.538675, 0.538675] ;[0.25, 0.25])
```

$\Rightarrow$ Validated Gauss-Legendre !

Re-discover the theory and ...

No 2-stage method of order 5

Proof in 0.04s !

Now find new methods

Remark: it is hard to be sure that a method is new...

A method order 4, 3 stages, singly, stiffly accurate

This method is promising: capabilities wanted for a stiff problem, singly to optimize the Newton solving and stiffly accurate to be more efficient w.r.t. stiff problems (and DAEs).

[0.161097, 0.161097]	[0.105662, 0.105662]	[0.172855, 0.172855]	[-0.117419, -0.117419]
[0.655889, 0.655889]	[0.482099, 0.482099]	[0.105662, 0.105662]	[0.068127, 0.068127]
[1, 1]	[0.388545, 0.388545]	[0.505792, 0.505792]	[0.105662, 0.105662]
	[0.388545, 0.388545]	[0.505792, 0.505792]	[0.105662, 0.105662]

Table : New method S3O4

A method order 5, 3 stages, explicit first line

With only 6 non zero coefficients, this method is a good agreement between a method with order 4 and 4 intermediate computations (Gauss4) and order 6 with 9 intermediate computations (Gauss6).
NB: there is no Gauss at order 5...

[0, 0]	[0, 0]	[0, 0]	[0, 0]
[0.355051, 0.355051]	[0.152659, 0.152659]	[0.220412, 0.220412]	[-0.018021, -0.018021]
[0.844948, 0.844948]	[0.087340, 0.087340]	[0.578021, 0.578021]	[0.179587, 0.179587]
	[0.111111, 0.111111]	[0.512485, 0.512485]	[0.376403, 0.376403]

Table : New method S3O5

Integration with the new schemes

- Implemented in DynIbex (a tool for validated simulation)
- Norm of diameter of final solution bounds the global error

Methods	time (s)	nb of steps	norm of diameter of final solution
S3O4	39	1821	$5.9 \cdot 10^{-5}$
Radau3	52	7509	$2 \cdot 10^{-4}$
Radau5	81	954	$7.6 \cdot 10^{-5}$

Table : S3O4 on a stiff problem (oil problem)

Methods	time (s)	nb of steps	norm of diameter of final solution
S3O5	92	195	5.9
Gauss4	45	544	93.9
Gauss6	570	157	7.0

Table : S3O5 on a problem with interval param. (vericomp p.61)

Discussion

- S3O4: Singly to optimize the Newton solving and stiffly accurate to be more efficient
  $\Rightarrow$ As efficient than Radau at order 5, but faster than order 3 !
- S3O5: With only 6 non zero coefficients, this method is a good agreement between a method with order 4 and 4 intermediate computations (Gauss4) and order 6 with 9 intermediate computations (Gauss6)
  $\Rightarrow$ More efficient than Gauss6 and 5 time faster !

Cost function to define optimal schemes

Problem: continuum of solutions

CSP can be under constrained (e.g., $p \leq s$)

Example of countless methods

Countless number of 2-stage; order 2; stiffly accurate; fully implicit

Optimization

- We could find the best one!
- How choose the cost function?

Cost function to define optimal schemes

Problem: continuum of solutions

CSP can be under constrained (e.g., $p \leq s$)

Example of countless methods

Countless number of 2-stage; order 2; stiffly accurate; fully implicit

Optimization

- We could find the best one!
- How choose the cost function?

Cost function to define optimal schemes

Problem: continuum of solutions

CSP can be under constrained (e.g., $p \leq s$)

Example of countless methods

Countless number of 2-stage; order 2; stiffly accurate; fully implicit

Optimization

- We could find the best one!
- How choose the cost function?

Cost function

Minimizing local truncature error

- Method with lower error for the same order
- Example of general form of ERK with 2 stages and order 2

0	0	0
α	α	0
	1-1/(2 α)	1/(2 α)

Ralston[1]: $\alpha = 2/3$ minimizes the sum of square of coefficients of rooted trees in the lte computation

Our approach: maximizing the order

- Minimizing the sum of squares of order constraints
- Cost easy to compute: direct from constraints
- Same result $\alpha \in [0.666...6, 0.666...7]$!

[1] Ralston, Anthony. "Runge-Kutta methods with minimum error bounds." Mathematics of computation (1962).

Cost function

Minimizing local truncature error

- Method with lower error for the same order
- Example of general form of ERK with 2 stages and order 2

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \alpha & \alpha & 0 \\ \hline & \text{1-1}/(2\ \alpha) & 1/(2\ \alpha) \end{array}$$

Ralston[1]: $\alpha = 2/3$ minimizes the sum of square of coefficients of rooted trees in the lte computation

Our approach: maximizing the order

- Minimizing the sum of squares of order constraints
- Cost easy to compute: direct from constraints
- Same result $\alpha \in [0.666...6, 0.666...7]$!

[1] Ralston, Anthony. "Runge-Kutta methods with minimum error bounds." Mathematics of computation (1962).

Detail of optimizer

Method in Ibex

Based on a branch and bound algorithm, with epsilon relaxation of constraints

Problem of relaxation

Implies to verify coefficients by a second step with solver and some fixed values (without relaxation)

Re-discover the theory

Theory

Countless 2-stage order 2 stiffly accurate fully implicit. But there is only one method at order 3: RadauIIA.

Optimization of (2,2)

```
best feasible point (0.749999939992 ; 0.250000060009 ;
0.333333280449 ; 0.999999998633 ;
0.416655823215 ; -0.0833225527662 ;
0.749999932909 ; 0.250000055725)
cpu time used 0.3879s.
```

with a cost of $[-\infty, 2.89787805696 \cdot 10^{-11}]$: there is an order 3 !

Verification

We add constraints $b_1 = 0.75$ and $c_2 = 1$, then we find RadauIIA

Explicit 3 stages 3 order

Theory (again)

There is countless explicit (3,3)-methods, but there is no order 4 method with 3 stages.

With optimizer: Erk33

[0, 0]	[0, 0]	[0, 0]	[0, 0]
[0.465904, 0.465904]	[0.465904, 0.465904]	[0, 0]	[0, 0]
[0.800685, 0.800685]	[-0.154577, -0.154577]	[0.955262, 0.955262]	[0, 0]
	[0.195905, 0.195906]	[0.429613, 0.429614]	[0.374480, 0.374480]

Comparison to Kutta (known to be the best)

Euclidean distance between fourth order conditions (1/4, 1/8, 1/12, 1/24) and obtained values:

- ERK33: $[0.045221, 0.045221]$
- Kutta: 0.058925

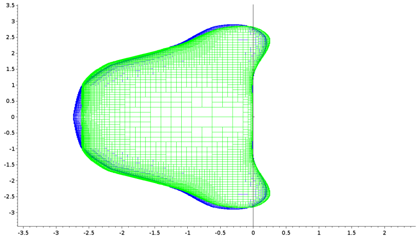
$\Rightarrow$ Our method is then closer to fourth order than Kutta.

Integration with Erk33, on VanDerPol

Methods	time	nb of steps	norm of diameter of final solution
ERK33	3.7	647	$2.2 \cdot 10^{-5}$
Kutta (3,3)	3.55	663	$3.4 \cdot 10^{-5}$
RK4 (4,4)	4.3	280	$1.9 \cdot 10^{-5}$

Paving of stability domain

For RK4 method (blue) and for Erk33 (green): really close !



Conclusion

Done

- Solver to find new validated Runge-Kutta methods, which preserve properties
- Optimizer to tend to an higher order
- Some testes which prove that our approach is valid

Future

- Automatic generation of order conditions greater than 5
- Branch with high-level properties (not only structure), such as stability, symplecticity...

Questions ?