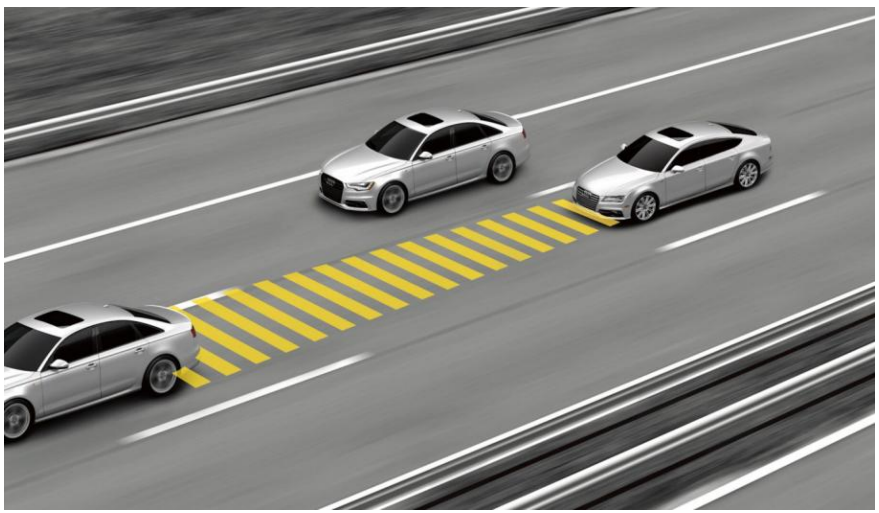
Guaranteed co-simulation of continuous-time dynamical systems

Adrien Le Coënt

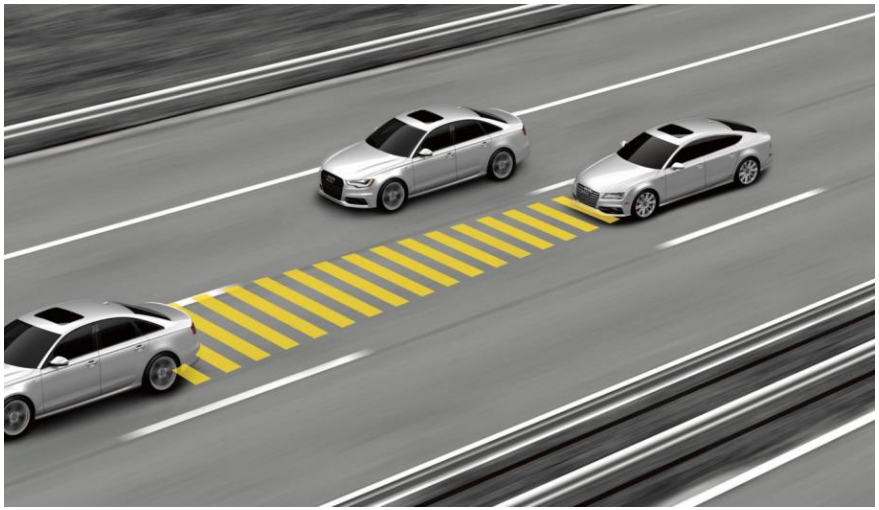
joint work with Julien Alexandre dit Sandretto and Alexandre Chapoutot
U2IS, ENSTA Paris, Palaiseau, France

Journée scientifique conjointe Chaire ISC et projet DGA AID
March 13, 2020

Context : Cyber-Physical Systems



Context : Cyber-Physical Systems



Model-based design of cyber-physical systems :

modeling, analysis, controller synthesis, **simulation**, deployment

Guaranteed Simulation Cyber-Physical Systems

Applications : verification, parameter and/or control synthesis, safety critical systems, **reachability analysis**

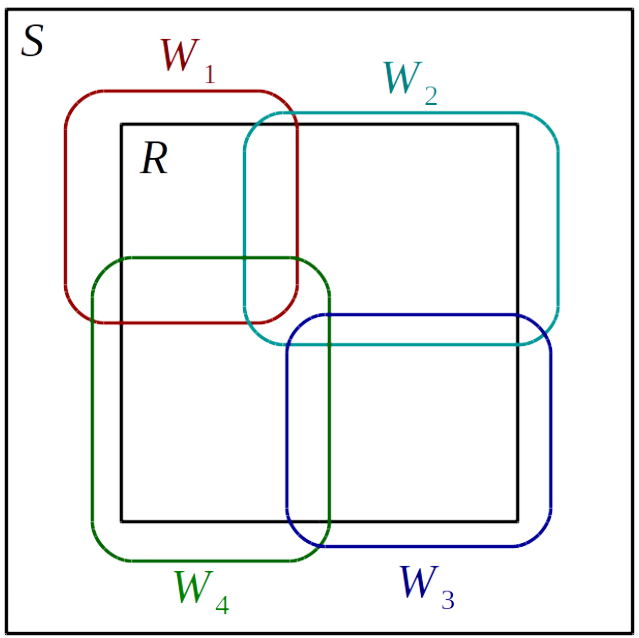
Guaranteed Simulation Cyber-Physical Systems

Applications : verification, parameter and/or control synthesis, safety critical systems, **reachability analysis**

Example of a controller synthesis algorithm for a switched system:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$

Goal: from any $x \in R$, return in R while always staying in S.



Basic idea:

- Generate a covering of R

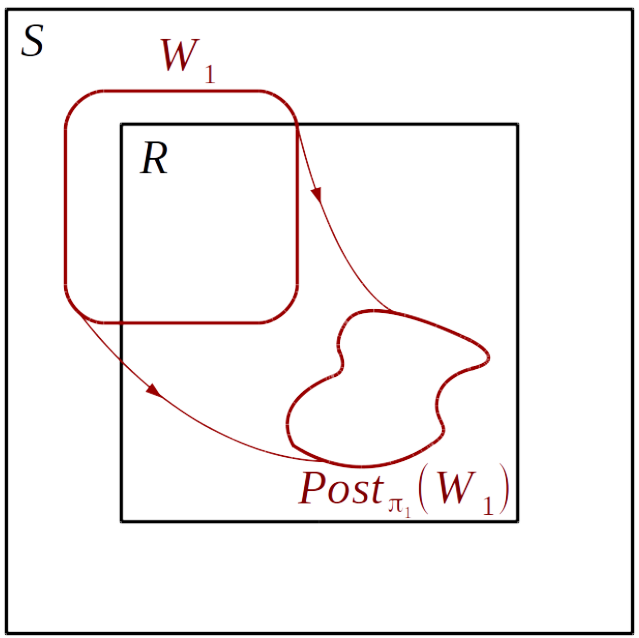
Guaranteed Simulation Cyber-Physical Systems

Applications : verification, parameter and/or control synthesis, safety critical systems, **reachability analysis**

Example of a controller synthesis algorithm for a switched system:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$

Goal: from any $x \in R$, return in R while always staying in S.



Basic idea:

- Generate a covering of R
- Look for patterns (input sequences) mapping the tiles into R while always staying in S

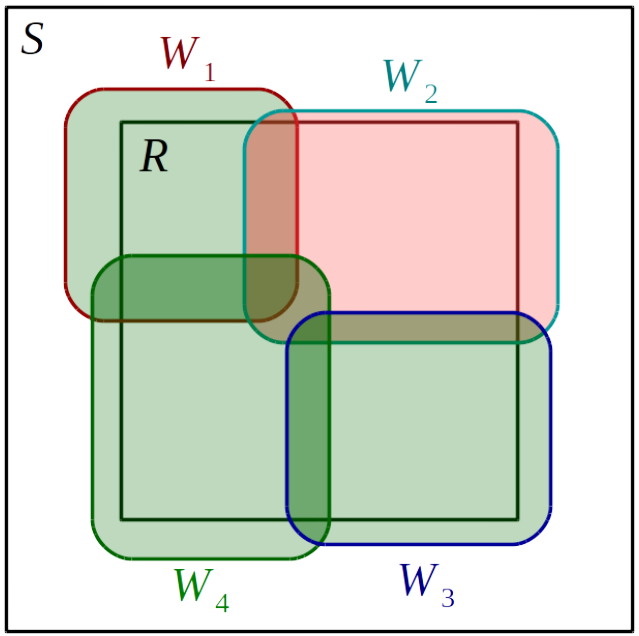
Guaranteed Simulation Cyber-Physical Systems

Applications : verification, parameter and/or control synthesis, safety critical systems, **reachability analysis**

Example of a controller synthesis algorithm for a switched system:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$

Goal: from any $x \in R$, return in R while always staying in S.



Basic idea:

- Generate a covering of R
- Look for patterns (input sequences) mapping the tiles into R while always staying in S
- If it fails,

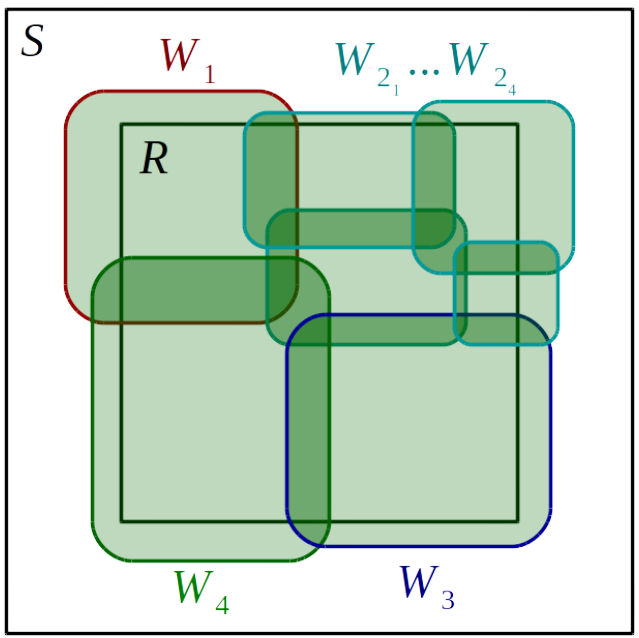
Guaranteed Simulation Cyber-Physical Systems

Applications : verification, parameter and/or control synthesis, safety critical systems, **reachability analysis**

Example of a controller synthesis algorithm for a switched system:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t))$$

Goal: from any $x \in R$, return in R while always staying in S.



Basic idea:

- Generate a covering of R
- Look for patterns (input sequences) mapping the tiles into R while always staying in S
- If it fails, generate another covering.

Numerical integration, reachability analysis, state-of-the-art

- Classical (non guaranteed) methods: Euler, Runge-Kutta, implicit, explicit schemes...
- Guaranteed reachability analysis: Enclosing solutions, error bounding, additional hypotheses

Numerical integration, reachability analysis, state-of-the-art

- Classical (non guaranteed) methods: Euler, Runge-Kutta, implicit, explicit schemes...
- Guaranteed reachability analysis: Enclosing solutions, error bounding, additional hypotheses
- State-of-the-art:
  - Monotonicity, ISS, incremental stability [Girard, Sontag, Zamani, Tabuada...]
  - Validated simulation, guaranteed integration [Moore, Lohner, Bertz, Makino, Nedialkov, Jackson, Corliss, Chen, Ábrahám, Sankaranarayanan, Taha, Chapoutot,...]
  - Sensitivity Analysis [Donzé, Maler...]

Numerical integration, reachability analysis, state-of-the-art

- Classical (non guaranteed) methods: Euler, Runge-Kutta, implicit, explicit schemes...
- Guaranteed reachability analysis: Enclosing solutions, error bounding, additional hypotheses
- State-of-the-art:
  - Monotonicity, ISS, incremental stability [Girard, Sontag, Zamani, Tabuada...]
  - Validated simulation, guaranteed integration [Moore, Lohner, Bertz, Makino, Nedialkov, Jackson, Corliss, Chen, Ábrahám, Sankaranarayanan, Taha, Chapoutot,...]
  - Sensitivity Analysis [Donzé, Maler...]
- Data structures:
  - Reachability analysis using zonotopes [Dang, Girard, Althoff...]
  - Ellipsoid methods [Kurzhanski, Varaiya, Dang...]

Initial Value Problem of Ordinary Differential Equations

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

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{p}) \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0, \ \mathbf{p} \text{ bounded},$$

for a given perturbation $\mathbf{p}(\cdot)$, IVP has a unique solution $\mathbf{x}(t; \mathbf{y}_0, \mathbf{p})$ if $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz in $\mathbf{y}$ and $f(\cdot, \mathbf{p}(\cdot))$ continuous
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: $t_0 = 0 < t_1 < \cdots < t_n = t_{\text{end}}$
- Compute a sequence of values: $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$ such that

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

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

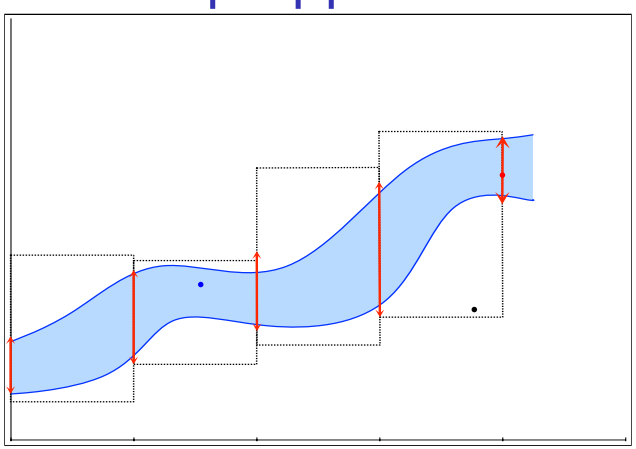
Guaranteed solution of IVP for ODE

Goal of guaranteed numerical integration

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

$$\forall \ell \in [0, n], \quad [\mathbf{x}_\ell] \ni \mathbf{x}(t_\ell; \mathbf{x}_{\ell-1}, \mathbf{p}) \ .$$

A two-step approach



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

Taylor Methods, State-of-the-art

Taylor methods

They have been developed since 60's (Moore, Lohner, Makino and Berz, Corliss and Rhim, Neher *et al.*, 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)
- etc.

One question

Why bother to define
new guaranteed numerical integration methods?

An answer: 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 $\mathbf{x}(t; \mathbf{x}_0)$)

Can we benefit these properties in guaranteed computations?

History on Interval Runge-Kutta methods

- Andrzej Marciniak *et al.* work on this topic since 1999

  "The form of $\psi(t, x(t))$ is very complicated and cannot be written in a general form for an arbitrary p"

  The implementation OOIRK is not freely avalaible.

- Hartmann and Petras, ICIAM 1999
  No more information than an abstract of 5 lines.

- Bouissou and Martel, SCAN 2006 (only RK4 method)
  Implementation GRKLib is not avaliable

- Bouissou, Chapoutot and Djoudi, NFM 2013 (any explicit RK)
  Implementation is not avaliable

- Alexandre dit Sandretto and Chapoutot, 2016 (any explicit and implicit RK)
  implementation DynIBEX is open-source, combine with IBEX

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:

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

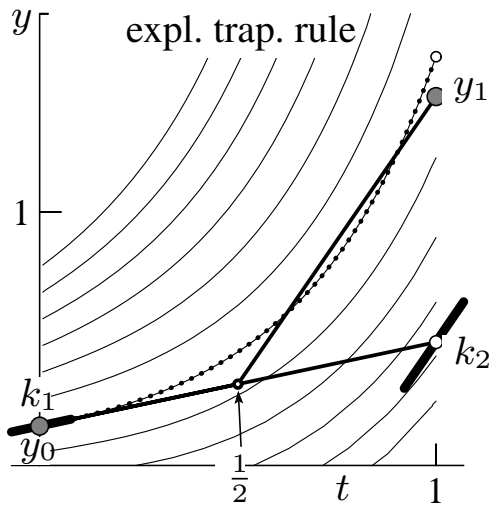
$$\mathbf{x}_{i+1} = \mathbf{x}_\ell + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right)$$

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

Intuition

- $\dot{x} = t^2 + x^2$
- $x_0 = 0.46$
- $h = 1.0$

dotted line is the exact solution.



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

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:

$$\mathbf{k}_1 = f\left(t_\ell + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)h, \quad \mathbf{x}_\ell + h\left(\frac{1}{4}\mathbf{k}_1 + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)\mathbf{k}_2\right)\right)$$

$$\mathbf{k}_2 = f\left(t_\ell + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)h, \quad \mathbf{x}_\ell + h\left(\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right)\mathbf{k}_1 + \frac{1}{4}\mathbf{k}_2\right)\right)$$

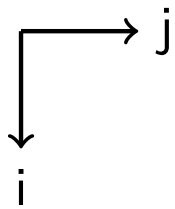
$$\mathbf{x}_{\ell+1} = \mathbf{x}_\ell + h\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2\right)$$

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

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



which induces the following algorithm

$$\mathbf{k}_i = f\left(t_\ell + c_i h_\ell, \quad \mathbf{x}_\ell + h_\ell \sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right), \qquad \mathbf{x}_{\ell+1} = \mathbf{x}_\ell + h_\ell \sum_{i=1}^{s} b_i \mathbf{k}_i$$

- **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

Guaranteed Runge-Kutta methods

A guaranteed algorithm

$$[\mathbf{x}_{\ell+1}] = [\mathrm{RK}]\,(h, [\mathbf{x}_\ell]) + \mathrm{LTE} \ .$$

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** replacing centered form and QR decomposition
- **Problem 2** is solved by bounding the **Local Truncation Error** (LTE) of Runge-Kutta methods 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)

$$\mathbf{x}(t_\ell; \mathbf{x}_{\ell-1}) - \mathbf{x}_\ell = C \cdot h^{p+1} \quad \text{with} \quad C \in \mathbb{R}.$$

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?

Example: building an order 2 explicit Runge-Kutta method

Consider, **a scalar IVP**

$$\dot{x}(t) = f(x(t)) \quad \text{with} \quad x(t_0) = x_0$$

Consider **an explicit Runge-Kutta method** of the form

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_1 & a_{21} & 0 \\ \hline & b_1 & b_2 \end{array}$$

i.e.,

$$x_1 = x_0 + h(b_1 k_1 + b_2 k_2) \quad \text{with} \quad \begin{cases} k_1 = f(x_0) \\ k_2 = f(x_0 + h a_{21} k_1) \end{cases}$$

and **we want to make it of order** 2

Example: building an order 2 explicit Runge-Kutta method

Taylor expansion of the exact solution, *i.e.*,

$$\dot{x}(t) = f(x(t)) \quad \text{with} \quad x(0) = x_0$$

We have, up to order 3

$$\begin{aligned} x(t_0 + h) &= x(t_0) + h\dot{x}(t_0) + \frac{h^2}{2}\ddot{x}(t_0) + \mathcal{O}(h^3) \\ &= x(t_0) + hf(x(t_0)) + \frac{h^2}{2}\frac{\partial f}{\partial x}(x(t_0))f\left(x(t_0)\right) + \mathcal{O}(h^3) \end{aligned}$$

Example: building an order 2 explicit Runge-Kutta method

Taylor expansion of the numerical solution, *i.e.*,

$$x_1 = x_0 + h(b_1 k_1 + b_2 k_2) \quad \text{with} \quad \begin{cases} k_1 = f(x_0) \\ k_2 = f(x_0 + h a_{21} k_1) \end{cases}$$

We have ($h = t - t_0$)

$$\frac{dk_1}{dh} = 0$$

$$\frac{dk_2}{dh} = \left(a_{21} k_1 + h a_{21} \frac{dk_1}{dh} \right) \cdot \frac{\partial f}{\partial x}(x_0 + h a_{21} k_1) \quad = a_{21} f(x_0) \frac{\partial f}{\partial x}(x_0 + h a_{21} f(x_0))$$

so,

$$\frac{dx_1}{dh} = b_1 k_1 + b_2 k_2 + h(b1 \frac{dk_1}{dh} + b_2 \frac{dk_2}{dh})$$

$$= b_1 f(x_0) + b_2 f(x_0 + h a_{21} f(x_0)) + h b_2 a_{21} f(x_0) \frac{\partial f}{\partial x}(x_0 + h a_{21} f(x_0))$$

When $h = 0$, we have

$$\frac{dx_1}{dh} = b_1 f(x_0) + b_2 f(x_0) = (b_1 + b_2) f(x_0)$$

Example: building an order 2 explicit Runge-Kutta method

Taylor expansion of the numerical solution, *i.e.*,

$$x_1 = x_0 + h(b_1 k_1 + b_2 k_2) \quad \text{with} \quad \begin{cases} k_1 = f(x_0) \\ k_2 = f(x_0 + h a_{21} k_1) \end{cases}$$

We can pursue the process at the second order to get

$$\frac{d^2 x_1}{dh^2} = b_2 a_{21} f(x_0) \frac{\partial f}{\partial x}(x_0 + h a_{21} f(x_0)) + (b_2 a_{21} f(x_0)) \frac{\partial f}{\partial x}(x_0 + h a_{21} f(x_0))$$
$$+ h(b_2 a_{21} f(x_0)) a_{21} f(x_0)) \frac{\partial^2 f}{\partial x^2}(x_0 + h a_{21} f(x_0))$$

When $h = 0$, we have

$$\frac{d^2 x_1}{dh^2} = 2 b_2 a_{21} f(x_0) \frac{\partial f}{\partial x}(x_0)$$

Example: building an order 2 explicit Runge-Kutta method

Hence, we get the **Taylor expansion of the numerical solution**

$$x_1 = x_0 + h\,(b_1 + b_2)\,f(x_0) + h^2\,b_2\,a_{21}\,f(x_0)\frac{\partial f}{\partial x}(x_0) + \mathcal{O}(h^3)$$

w.r.t., the **Taylor expansion of the exact solution**

$$x(t+h) = x(t_0) + h1f\,(x(t_0)) + h^2\frac{1}{2}f(x(t_0))\frac{\partial f}{\partial x}(x(t_0)) + \mathcal{O}(h^3)$$

with **localization assumption**, *i.e.*, $x(t_0) = x_0$, we have the constraints

$$\begin{cases} b_1 + b_2 = 1 \\ \quad b_2\,a_{21} = \dfrac{1}{2} \end{cases}$$

Note: there is an infinity set of solutions of order 2 methods. Two notorious

- $b_1 = 0$, $b_2 = 1$ and $a_{21} = \frac{1}{2}$ (Explicit midpoint method)
- $b_1 = \frac{1}{2}$, $b_2 = \frac{1}{2}$ and $a_{21} = 1$ (Heun's method)

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

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

High order derivatives of exact solution x

$$
\begin{aligned}
\dot{\mathbf{x}} &= f(\mathbf{x}) \\
\ddot{\mathbf{x}} &= f'(\mathbf{x})\dot{\mathbf{x}} \\
\mathbf{x}^{(3)} &= f''(\mathbf{x})(\dot{\mathbf{x}},\dot{\mathbf{x}}) + f'(\mathbf{x})\ddot{\mathbf{x}} \\
\mathbf{x}^{(4)} &= f'''(\mathbf{x})(\dot{\mathbf{x}},\dot{\mathbf{x}},\dot{\mathbf{x}}) + 3f''(\mathbf{x})(\ddot{\mathbf{x}},\dot{\mathbf{x}}) + f'(\mathbf{x})\mathbf{x}^{(3)} \\
\mathbf{x}^{(5)} &= f^{(4)}(\mathbf{x})(\dot{\mathbf{x}},\dot{\mathbf{x}},\dot{\mathbf{x}},\dot{\mathbf{x}}) + 6f'''(\mathbf{x})(\ddot{\mathbf{x}},\dot{\mathbf{x}},\dot{\mathbf{x}}) \\
&\quad + 4f''(\mathbf{x})(\mathbf{x}^{(3)},\dot{\mathbf{x}}) + 3f''(\mathbf{x})(\ddot{\mathbf{x}},\ddot{\mathbf{x}}) + f'(\mathbf{x})\mathbf{x}^{(4)} \\
&\vdots
\end{aligned}
$$

$f'(\mathbf{x})$ is a linear map

$f''(\mathbf{x})$ is a bi-linear map

$f'''(\mathbf{x})$ is a tri-linear map

$\vdots$

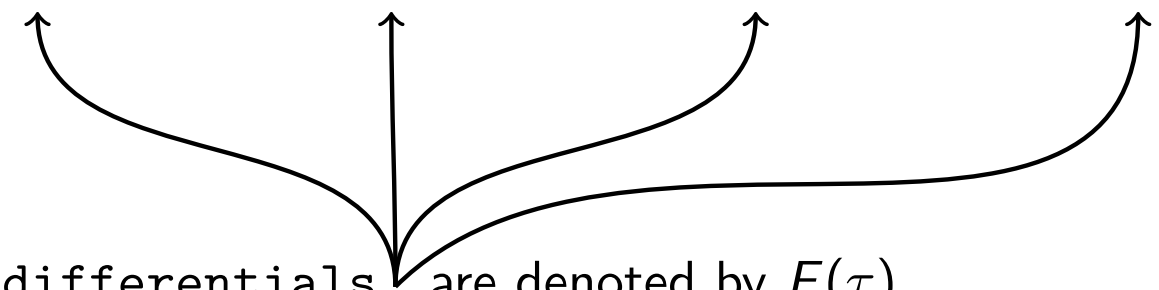
[2] strongly inspired from "Geometric Numerical Integration", Hairer, Lubich and Wanner.

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

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

High order derivatives of exact solution x

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



- `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]

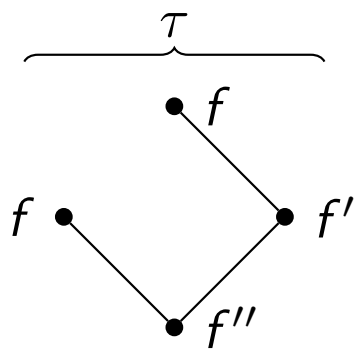
Rooted trees: a combinatorial view of elementary differentials

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

Example

$$\underbrace{f''(f'f, f)}_{F(\tau)}$$

is associated to



Note: τ is not unique, *e.g.*, symmetry

Consequences

At a given order, an **enumeration of all the trees** is possible $\Rightarrow$ all the elementary differentials are enumerable

Order	1	2	3	4	5	6	7	8	9	10
Number of τ	1	1	2	4	9	20	48	115	286	719

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

Theorem 1 (Butcher, 1963)

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

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

We can do the same for the numerical solution

Theorem 2 (Butcher, 1963)

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

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

Theorem 3, order condition (Butcher, 1963)

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

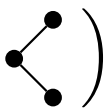
LTE formula for **explicit and implicit** Runge-Kutta

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

$$\mathbf{x}(t_1;\mathbf{x}_0) - \mathbf{x}_1 = \frac{h^{p+1}}{(p+1)!} \sum_{r(\tau)=p+1} \alpha(\tau)\big[1-\gamma(\tau)\phi(\tau)\big]F(\tau)(\mathbf{x}(\xi)), \quad \xi \in [t_1, t_0]$$

- $\alpha(\tau)$ 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 $c_j = \displaystyle\sum_{k=1}^{s} a_{jk}$

Remark

Making guaranteed Runge-Kutta is a simpler problem, *i.e.*, for each method the Butcher tableau and the order are already available

Note: $\mathbf{x}(\xi)$ can be enclosed by $[\tilde{\mathbf{x}}]$ using Interval Picard-Lindelöf operator

Implementation of LTE formula

Notations

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

Two ways of computing $F(\tau)$

1. **Symbolic differentiation** : complexity $\mathcal{O}(n^p)$
   - compute all partial derivatives symbolically
   - combine them following the rooted tree structures

2. **Automatic differentiation** : complexity $\mathcal{O}(n3^p)$
   based on the work of Ferenc Bartha and Hans Munthe-Kaas
   "Computing of B-Series by Automatic Differentiation", 2014

A guaranteed numerical integration based on Runge-Kutta

A guaranteed algorithm

$$[\mathbf{x}_{\ell+1}] = [\mathrm{RK}]\,(h, [\mathbf{x}_\ell]) + \mathrm{LTE}\ .$$

Note on the implementation of $[\mathrm{RK}]\,(h, [\mathbf{x}_\ell])$

- If **explicit** method is considered
  [RK] is an **inclusion function**

- If **implicit** method is considered
  [RK] is an **interval contractor operator**

Example: implicit midpoint

$$\mathbf{k}_1 = f\left(t_\ell + \frac{h}{2}, \mathbf{x}_\ell + \frac{h}{2}\mathbf{k}_1\right) \quad \Rightarrow \quad [\mathbf{k}_1] = [\mathbf{k}_1] \cap [f]\left(t_\ell + \frac{h}{2}, [\mathbf{x}_\ell] + \frac{h}{2}[\mathbf{k}_1]\right)$$

$$\mathbf{x}_{\ell+1} = \mathbf{x}_\ell + h\mathbf{k}_1 \qquad [\mathbf{x}_{\ell+1}] = [\mathbf{x}_\ell] + h[\mathbf{k}_1]$$

Starting the contraction with $[\tilde{\mathbf{x}}]$ the result of interval Picard-Lindelöf operator

RK-based of interval Picard-Lindelöf operator

Starting from the expression

$$\mathbf{k}_i(t) = f\left(t_\ell + c_i(t - t_\ell), \mathbf{x}_\ell + (t - t_\ell)\sum_{n=1}^{s} a_{in}\mathbf{k}_n\right), \quad 1 \leqslant i \leqslant s$$

$$\mathbf{x}_{\ell+1}(t, \xi) = \mathbf{x}_\ell + (t - t_\ell)\sum_{i=1}^{s} b_i\mathbf{k}_i(t) + \mathrm{LTE}(t, \mathbf{x}(\xi))$$

We can define an inclusion function with $h = t_{j+1} - t_j$ such that

$$P([t_\ell, t_{\ell+1}], [\tilde{\mathbf{x}}]) := [\mathbf{x}_\ell] + [0, h]\sum_{i=1}^{s} b_i[\mathbf{k}_i]\left([t_\ell, t_{\ell+1}]\right) + \mathrm{LTE}([t_\ell, t_{\ell+1}], [\tilde{\mathbf{x}}])$$

Hence, it is sufficient to have

$$[\tilde{\mathbf{x}}] \supseteq P\left([t_\ell, t_{\ell+1}], [\tilde{\mathbf{x}}]\right)$$

to prove the existence and uniqueness of the solution of IVP ODE.

Interval RK methods summary

Given a RK scheme, guaranteed integration works with a two step approach:

- Prove existence and uniqueness of solution :
  Picard-Lindelöf operator contraction proof
  $\Rightarrow$ Box bounding the state **over** the next time interval
- Apply interval scheme with bounded LTE :
  Computation and evaluation of LTE
  $\Rightarrow$ Box bounding solution **at** next time step

Interval RK methods summary

Given a RK scheme, guaranteed integration works with a two step approach:

- Prove existence and uniqueness of solution :
  Picard-Lindelöf operator contraction proof
  $\Rightarrow$ Box bounding the state **over** the next time interval
- Apply interval scheme with bounded LTE :
  Computation and evaluation of LTE
  $\Rightarrow$ Box bounding solution **at** next time step

Complexity comes from :

- Computation of the LTE

Interval RK methods summary

Given a RK scheme, guaranteed integration works with a two step approach:

- Prove existence and uniqueness of solution :
  Picard-Lindelöf operator contraction proof
  $\Rightarrow$ Box bounding the state **over** the next time interval
- Apply interval scheme with bounded LTE :
  Computation and evaluation of LTE
  $\Rightarrow$ Box bounding solution **at** next time step

Complexity comes from :

- Computation of the LTE

In practice :

- Dimension limited to ≈ 50

Co-simulation

Let us suppose a **decomposed** dynamics :

$$\begin{aligned}
&\dot{\mathbf{x}}_1 \in f_1(t, \mathbf{x}_1, \mathbf{u}_1) \quad \text{with} \quad \mathbf{x}_1(0) \in [\mathbf{x}_1^0],\ \mathbf{u}_1 \in [\mathbf{u}_1],\\
&\dot{\mathbf{x}}_2 \in f_2(t, \mathbf{x}_2, \mathbf{u}_2) \quad \text{with} \quad \mathbf{x}_2(0) \in [\mathbf{x}_2^0],\ \mathbf{u}_2 \in [\mathbf{u}_2],\\
&\dots\\
&\dot{\mathbf{x}}_m \in f_m(t, \mathbf{x}_m, \mathbf{u}_m) \quad \text{with} \quad \mathbf{x}_m(0) \in [\mathbf{x}_m^0],\ \mathbf{u}_m \in [\mathbf{u}_m],\\
&L(\mathbf{x}_1, \dots, \mathbf{x}_m, \mathbf{u}_1, \dots, \mathbf{u}_m) = 0,
\end{aligned}$$

where the state $\mathbf{x}$ is decomposed in m components $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_m)$, for all $i \in \{1, \dots, m\}$, $\mathbf{x}_i \in X_i$, $X_1 \times \cdots \times X_m = \mathbb{R}^d$, and L is a coupling function between the components.

Co-simulation

Let us suppose a **decomposed** dynamics :

$$\begin{aligned}
&\dot{\mathbf{x}}_1 \in f_1(t, \mathbf{x}_1, \mathbf{u}_1) \quad \text{with} \quad \mathbf{x}_1(0) \in [\mathbf{x}_1^0],\ \mathbf{u}_1 \in [\mathbf{u}_1],\\
&\dot{\mathbf{x}}_2 \in f_2(t, \mathbf{x}_2, \mathbf{u}_2) \quad \text{with} \quad \mathbf{x}_2(0) \in [\mathbf{x}_2^0],\ \mathbf{u}_2 \in [\mathbf{u}_2],\\
&\dots\\
&\dot{\mathbf{x}}_m \in f_m(t, \mathbf{x}_m, \mathbf{u}_m) \quad \text{with} \quad \mathbf{x}_m(0) \in [\mathbf{x}_m^0],\ \mathbf{u}_m \in [\mathbf{u}_m],\\
&L(\mathbf{x}_1, \dots, \mathbf{x}_m, \mathbf{u}_1, \dots, \mathbf{u}_m) = 0,
\end{aligned}$$

where the state $\mathbf{x}$ is decomposed in m components $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_m)$, for all $i \in \{1, \dots, m\}$, $\mathbf{x}_i \in X_i$, $X_1 \times \cdots \times X_m = \mathbb{R}^d$, and inputs are given by $\mathbf{u}_i = K_i(\mathbf{x}_1, \dots, \mathbf{x}_m)$.

Co-simulation

Let us suppose a **decomposed** dynamics :

$$\begin{aligned}
&\dot{\mathbf{x}}_1 \in f_1(t, \mathbf{x}_1, \mathbf{u}_1) \quad \text{with} \quad \mathbf{x}_1(0) \in [\mathbf{x}_1^0],\ \mathbf{u}_1 \in [\mathbf{u}_1],\\
&\dot{\mathbf{x}}_2 \in f_2(t, \mathbf{x}_2, \mathbf{u}_2) \quad \text{with} \quad \mathbf{x}_2(0) \in [\mathbf{x}_2^0],\ \mathbf{u}_2 \in [\mathbf{u}_2],\\
&\dots\\
&\dot{\mathbf{x}}_m \in f_m(t, \mathbf{x}_m, \mathbf{u}_m) \quad \text{with} \quad \mathbf{x}_m(0) \in [\mathbf{x}_m^0],\ \mathbf{u}_m \in [\mathbf{u}_m],\\
&L(\mathbf{x}_1, \dots, \mathbf{x}_m, \mathbf{u}_1, \dots, \mathbf{u}_m) = 0,
\end{aligned}$$

where the state $\mathbf{x}$ is decomposed in m components $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_m)$, for all $i \in \{1, \dots, m\}$, $\mathbf{x}_i \in X_i$, $X_1 \times \cdots \times X_m = \mathbb{R}^d$, and inputs are given by $\mathbf{u}_i = K_i(\mathbf{x}_1, \dots, \mathbf{x}_m)$.

Principle of co-simulation:

- Simulate subsystems in a **distributed** manner
- At some instants, **exchange information** between subsystems

Advantages:

- Faster than simulating the entire system
- Enables large scale systems
- Different solvers can be used (*i.e.* applications to CPS and multi-physics)

Co-simulation, distributed reachability, state-of-the-art

- Co-simulation is widely used (at least 48 industrial applications reported) [survey by C. Gomes et al.]
- Most of the tools developed rely on FMI/FMU standard [M. Arnold, E. Lee]: Modelica, Simulink...
- No guaranteed co-simulation but some work on error bounding [M. Arnold]
- Guaranteed distributed reachability is less common
  - Compositional abstractions based on hybrid automata [Chen, Sankaranarayanan], linear arithmetic relations [Chen, Mover, Sankaranarayanan]
  - Local numerical integration with compositional splitting [Blanes, Casas, Murua]

Co-simulation formalism (Gomes et al.)

Continuous-time simulation unit:

$$
\begin{aligned}
& S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, \mathbf{x}_i(0), \mathbf{\Phi}_{U_I} \rangle, \\
& \delta_i : \mathbb{R} \times X_i \times U_i \to X_i, \\
& \lambda_i : \mathbb{R} \times X_i \times U_i \to Y_i, \text{ or } \mathbb{R} \times X_i \to Y_i, \\
& \mathbf{x}_i(0) \in X_i, \\
& \mathbf{\Phi}_{U_i} : \mathbb{R} \times U_i \times \cdots \times U_I \to U_i,
\end{aligned}
\tag{1}
$$

where

- X_i is the state vector space,
- U_i is the input vector space,
- Y_i is the output vector space,
- $\delta_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)) = \mathbf{x}_i(t + H)$ advances the simulation (using extrapolation function $\mathbf{\Phi}_{U_i}$)
- $\lambda_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)) = \mathbf{y}_i(t)$ or $\lambda_i(t, \mathbf{x}_i(t)) = \mathbf{y}_i(t)$ is the output function; and
- $\mathbf{x}_i(0)$ is the initial state.

Co-simulation formalism (Gomes et al.)

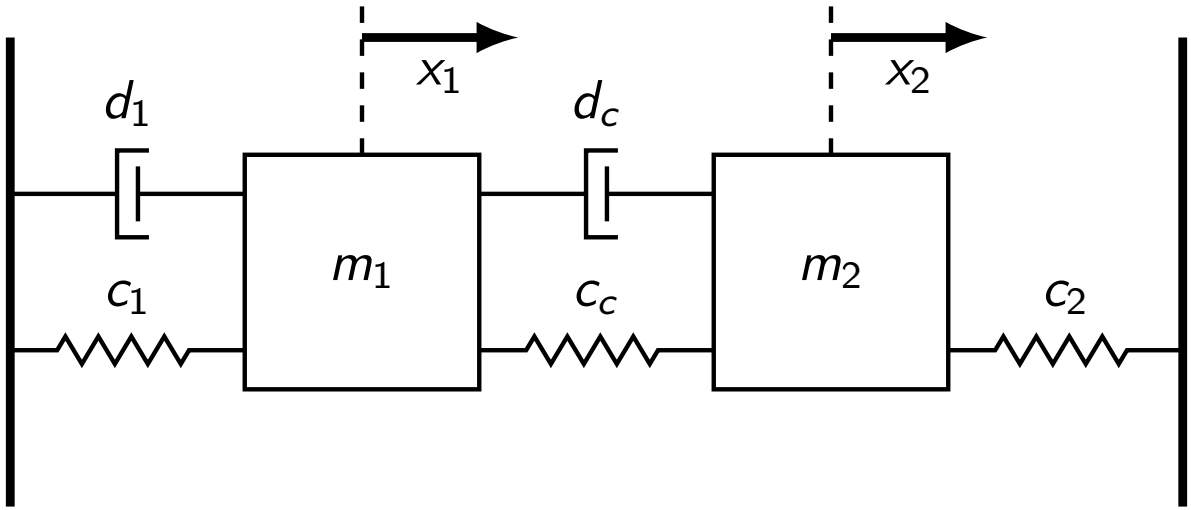
Continuous-time simulation unit:

$$
\begin{aligned}
&S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, \mathbf{x}_i(0), \mathbf{\Phi}_{U_I} \rangle, \\
&\delta_i : \mathbb{R} \times X_i \times U_i \to X_i, \\
&\lambda_i : \mathbb{R} \times X_i \times U_i \to Y_i, \text{ or } \mathbb{R} \times X_i \to Y_i, \\
&\mathbf{x}_i(0) \in X_i, \\
&\mathbf{\Phi}_{U_i} : \mathbb{R} \times U_i \times \cdots \times U_I \to U_i,
\end{aligned}
\tag{1}
$$

where

- X_i is the state vector space,
- U_i is the input vector space,
- Y_i is the output vector space,
- $\delta_i([t, t'], [\mathbf{x}_i], [\mathbf{u}_i]) = ([\mathbf{x}_i]', \{[\mathbf{x}_i^k]'\}_k)$ advances the simulation (using extrapolation function $\mathbf{\Phi}_{U_i}$)
- $\lambda_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)) = \mathbf{y}_i(t)$ or $\lambda_i(t, \mathbf{x}_i(t)) = \mathbf{y}_i(t)$ is the output function; and
- $\mathbf{x}_i(0)$ is the initial state.

Example of composed system



The dynamics of the system is given by the following system of equations:

$$\begin{cases} \dot{x}_1 = v_1 \\ m_1 \dot{v}_1 = -c_1 x_1 - d_1 v_1 + c_c(x_2 - x_1) + d_c(v_2 - v_1) \\ \dot{x}_2 = v_2 \\ m_2 \dot{v}_2 = -c_c(x_2 - x_1) - c_2 x_2 - d_c(v_2 - v_1) \end{cases}$$

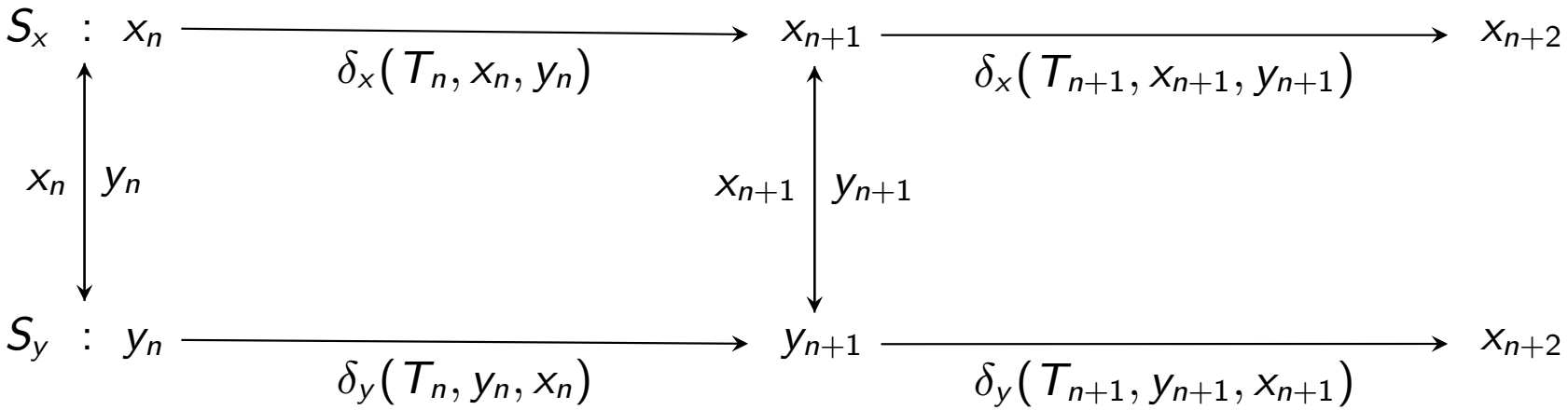
with the initial conditions $x_1(0) = x_2(0) = v_1(0) = v_2(0) = [1, 1]$ (a point interval).

Principle of classical co-simulation

With a simpler dynamics:

$$\dot{x} = f(x, y)$$
$$\dot{y} = g(y, x)$$

where $x(0) \in [x_0]$, $y(0) \in [y_0]$, the principle is :

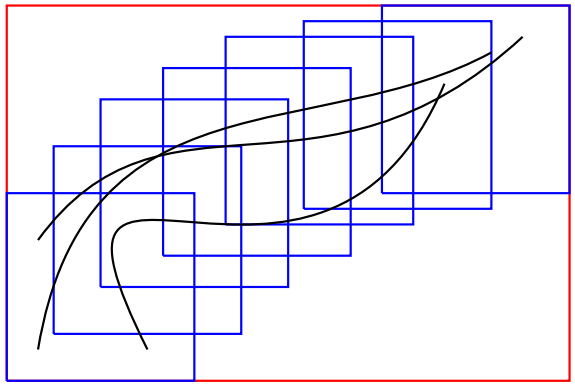


- on macro-steps, sub-systems advance simulation independently
- at communication times, they exchange output values (x_n and y_n)

Principle guaranteed of co-simulation

With a simpler dynamics:

$$\dot{x} = f(x, y)$$
$$\dot{y} = g(y, x)$$



where $x(0) \in [x_0]$, $y(0) \in [y_0]$, the principle is :

$$\begin{array}{ccccccc}
S_x & : & [x_n] & \xrightarrow[\delta_x(T_n, [x_n], [y_n^H])]{} & [x_{n+1}] & \xrightarrow[\delta_x(T_{n+1}, [x_{n+1}], [y_{n+1}^H])]{} & [x_{n+2}] \\
 & & \updownarrow [x_n^H]\ [y_n^H] & & \updownarrow [x_{n+1}^H]\ [y_{n+1}^H] & & \\
S_y & : & [y_n] & \xrightarrow[\delta_y(T_n, [y_n], [x_n^H])]{} & [y_{n+1}] & \xrightarrow[\delta_y(T_{n+1}, [y_{n+1}], [x_{n+1}^H])]{} & [x_{n+2}]
\end{array}$$

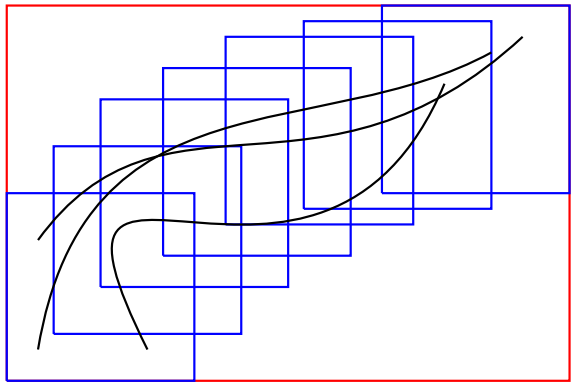
with

- $[x_n]$ and $[y_n]$ interval state at communication times
- $[x_n^H]$ and $[y_n^H]$ over-approximation of the state over the next macro-step $[T_n, T_n + H]$

Principle guaranteed of co-simulation

With a simpler dynamics:

$$\dot{x} = f(x, y)$$
$$\dot{y} = g(y, x)$$



where $x(0) \in [x_0]$, $y(0) \in [y_0]$, the principle is :

$$\begin{array}{ccccc}
S_x : & [x_n] & \xrightarrow[\delta_x(T_n, [x_n], [y_n^H])]{} & [x_{n+1}] & \xrightarrow[\delta_x(T_{n+1}, [x_{n+1}], [y_{n+1}^H])]{} & [x_{n+2}] \\
& \uparrow [x_n^H] \; \downarrow [y_n^H] & & \uparrow [x_{n+1}^H] \; \downarrow [y_{n+1}^H] & & \\
S_y : & [y_n] & \xrightarrow[\delta_y(T_n, [y_n], [x_n^H])]{} & [y_{n+1}] & \xrightarrow[\delta_y(T_{n+1}, [y_{n+1}], [x_{n+1}^H])]{} & [x_{n+2}]
\end{array}$$

with

- $[x_n]$ and $[y_n]$ interval state at communication times
- $[x_n^H]$ and $[y_n^H]$ over-approximation of the state over the next macro-step $[T_n, T_n + H]$

Question:
How to compute $[x_n^H]$ and $[y_n^H]$?

The cross-Picard operator

Local Picard-Lindelöf operators on macro-step $[T_n, T_{n+1}]$:

$$P_f([T_n, T_{n+1}], [\tilde{x}], [\tilde{y}]) := [x_n] + \sum_{k=0}^{N} f^{[k]}([x_n], [\tilde{y}])[0, H^k] + f^{[N+1]}([\tilde{x}], [\tilde{y}])[0, H^{N+1}].$$

$$P_g([T_n, T_{n+1}], [\tilde{y}], [\tilde{x}]) := [y_n] + \sum_{k=0}^{N} g^{[k]}([y_n], [\tilde{x}])[0, H^k] + f^{[N+1]}([\tilde{y}], [\tilde{x}])[0, H^{N+1}].$$

The cross-Picard operator

Local Picard-Lindelöf operators on macro-step $[T_n, T_{n+1}]$:

$$P_f([T_n, T_{n+1}], [\tilde{x}], [\tilde{y}]) := [x_n] + \sum_{k=0}^{N} f^{[k]}([x_n], [\tilde{y}])[0, H^k] + f^{[N+1]}([\tilde{x}], [\tilde{y}])[0, H^{N+1}].$$

$$P_g([T_n, T_{n+1}], [\tilde{y}], [\tilde{x}]) := [y_n] + \sum_{k=0}^{N} g^{[k]}([y_n], [\tilde{x}])[0, H^k] + f^{[N+1]}([\tilde{y}], [\tilde{x}])[0, H^{N+1}].$$

In order to prove that $[x_n^H]$ and $[y_n^H]$ are indeed over-approximating $x(t)$ and $y(t)$ over the next macro-step, the condition to verify is:

$$\mathcal{P}_f([T_n, T_{n+1}], [x_n^H], [y_n^H]) \subset \text{Int}([x_n^H]) \text{ and } \mathcal{P}_g([T_n, T_{n+1}], [y_n^H], [x_n^H]) \subset \text{Int}([y_n^H])$$

The cross-Picard operator

Local Picard-Lindelöf operators on macro-step $[T_n, T_{n+1}]$:

$$P_f([T_n, T_{n+1}], [\tilde{x}], [\tilde{y}]) := [x_n] + \sum_{k=0}^{N} f^{[k]}([x_n], [\tilde{y}])[0, H^k] + f^{[N+1]}([\tilde{x}], [\tilde{y}])[0, H^{N+1}].$$

$$P_g([T_n, T_{n+1}], [\tilde{y}], [\tilde{x}]) := [y_n] + \sum_{k=0}^{N} g^{[k]}([y_n], [\tilde{x}])[0, H^k] + f^{[N+1]}([\tilde{y}], [\tilde{x}])[0, H^{N+1}].$$

In order to prove that $[x_n^H]$ and $[y_n^H]$ are indeed over-approximating $x(t)$ and $y(t)$ over the next macro-step, the condition to verify is:

$\mathcal{P}_f([T_n, T_{n+1}], [x_n^H], [y_n^H]) \subset \text{Int}([x_n^H])$ and $\mathcal{P}_g([T_n, T_{n+1}], [y_n^H], [x_n^H]) \subset \text{Int}([y_n^H])$

Procedure: start with an initial guess (heuristics) and contract (fixed point)

The cross-Picard operator

Algorithm 1 Computation of the cross-Picard operator

Data: $cs = \langle \emptyset, Y_{cs}, D = \{1, \ldots, m\}, \{S_i\}_{i \in D}, L, \emptyset \rangle$, a time interval $[t, t + H]$, initial intervals $[x_{i,n}]$ and initial guesses $[r^H_{i,n}]$

Result: $\{[X^H_i]\}_{i=1,\ldots,m}$, a set of boxes over-approximating the global state on $[T_n, T_n + H]$

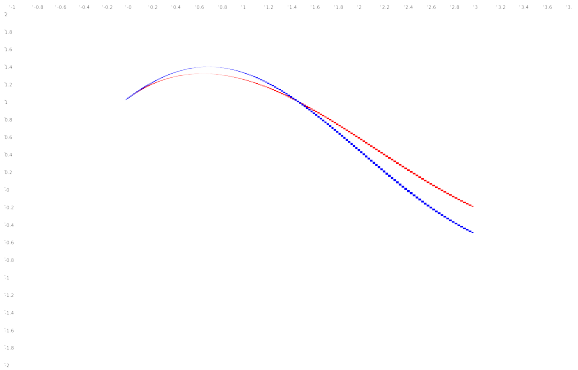
for $i = 1, \ldots, m$ (in parallel) **do**
$[\tilde{X}^H_i] := [r^H_{i,n}]$
$[U^H_i] := K_i([\tilde{X}^H_{1,n}], \ldots, [\tilde{X}^H_{1,n}])$
$[X^H_i] := \mathcal{P}^H_{[x_{i,n}],[U^H_i]}$

while $[X^H_i] \not\subseteq [\tilde{X}^H_i]$ for all i **do**
for $i = 1, \ldots, m$ (in parallel) **do**
$[\tilde{X}^H_i] := [X^H_i]$
$[U^H_i] := K_i([\tilde{X}^H_{1,n}], \ldots, [\tilde{X}^H_{1,n}])$
$[X^H_i] := \mathcal{P}^H_{[x_{i,n}],[U^H_i]}$

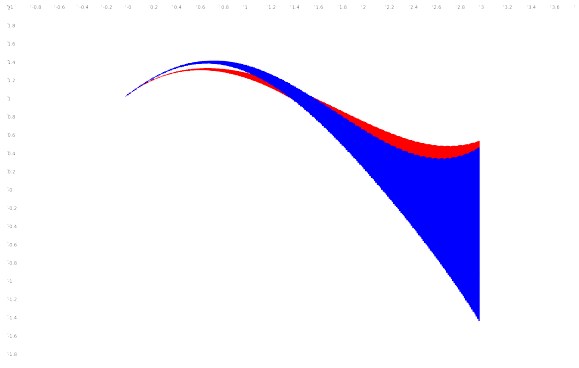
return $[X^H_i]$

Application

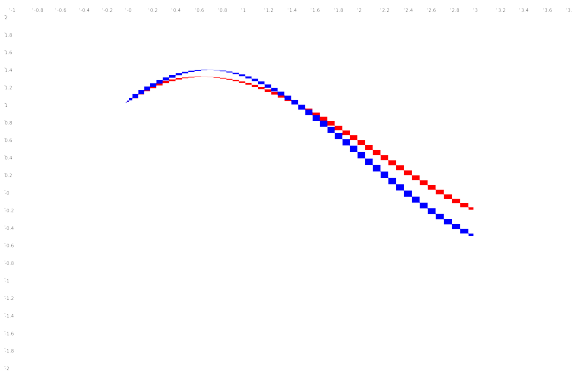
Double mass-spring-damper oscillator



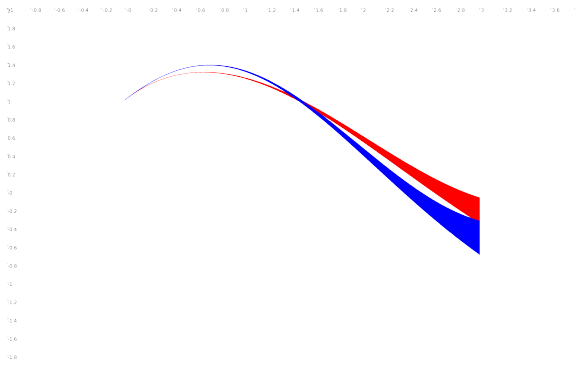
Heun



co-Heun (H=0.05)



RK4



co-RK4 (H=0.01)

More information exchange for more accuracy : guaranteed extrapolation

A simulation unit $S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, \mathbf{x}_i(0), \mathbf{\Phi}_{U_i} \rangle$ can **extrapolate inputs** for the next macro-step based on **previous behavior**:

Classical approach : interpolation polynomials

$$\mathbf{\Phi}_{U_i,n}(t) = \sum_{l=0}^{k} \mathbf{u}_i(T_{n-l}) \prod_{\substack{p=0 \\ p \neq l}}^{k} \frac{t - T_{n-p}}{T_{n-l} - T_{n-p}} = \mathbf{u}_i(t) + O(H^{k+1}) \qquad (2)$$

More information exchange for more accuracy : guaranteed extrapolation

A simulation unit $S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, \mathbf{x}_i(0), \mathbf{\Phi}_{U_i} \rangle$ can **extrapolate inputs** for the next macro-step based on **previous behavior**:

Classical approach : interpolation polynomials

$$\mathbf{\Phi}_{U_i,n}(t) = \sum_{l=0}^{k} \mathbf{u}_i(T_{n-l}) \prod_{\substack{p=0 \\ p \neq l}}^{k} \frac{t - T_{n-p}}{T_{n-l} - T_{n-p}} = \mathbf{u}_i(t) + O(H^{k+1}) \tag{2}$$

Interpolation error :

$$\mathbf{\Phi}_{U_i,n}(t) - \mathbf{u}_i(t) = \frac{1}{(k+1)!} \mathbf{u}_i^{(k+1)}(\xi) \prod_{i=0}^{k} (t - T_{n-k}) \quad \xi \in [T_n, T_{n+1}]$$

More information exchange for more accuracy : guaranteed extrapolation

A simulation unit $S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, \mathbf{x}_i(0), \mathbf{\Phi}_{U_i} \rangle$ can **extrapolate inputs** for the next macro-step based on **previous behavior**:

Higher chain formula:

$$\mathbf{u}_i^{(k)}(t) = k! \frac{\partial^{r_1+\cdots+r_m} K_i}{\partial \mathbf{x}_1^{r_1} \ldots \partial \mathbf{x}_m^{r_m}} \prod_{j=1}^{s} \prod_{l=1}^{m} \frac{1}{m_{jl}!} \left[\frac{1}{p_j!} \mathbf{x}_i^{(p_j)} \right]^{m_{jl}}$$

More information exchange for more accuracy : guaranteed extrapolation

A simulation unit $S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, \mathbf{x}_i(0), \mathbf{\Phi}_{U_i} \rangle$ can **extrapolate inputs** for the next macro-step based on **previous behavior**:

Higher chain formula:

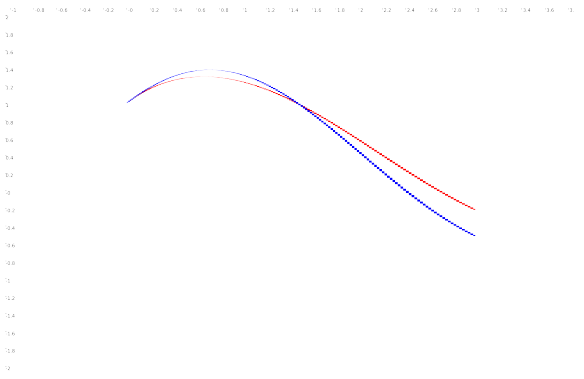
$$\mathbf{u}_i^{(k)}(t) = k! \frac{\partial^{r_1+\cdots+r_m} K_i}{\partial \mathbf{x}_1^{r_1} \ldots \partial \mathbf{x}_m^{r_m}} \prod_{j=1}^{s} \prod_{l=1}^{m} \frac{1}{m_{jl}!} \left[\frac{1}{p_j!} \mathbf{x}_i^{(p_j)} \right]^{m_{jl}}$$

Guaranteed interval extrapolation :

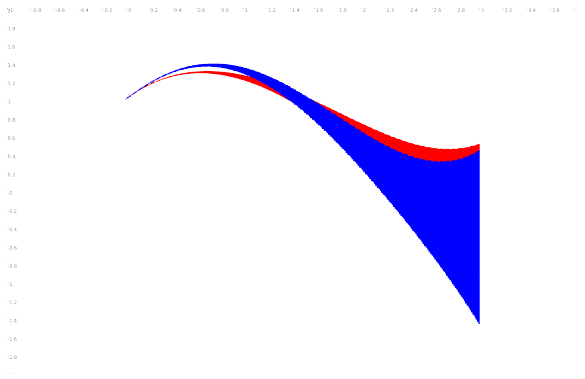
$$[\mathbf{\Phi}_{U_i,n}](t) = \sum_{l=0}^{k} [\mathbf{u}_{i,n-l}] \prod_{\substack{p=0 \\ p \neq l}}^{k} \frac{t - T_{n-p}}{T_{n-l} - T_{n-p}} + \frac{1}{(k+1)!} [\mathbf{u}_{i,n}^{(k),H}] \prod_{i=0}^{k} (t - T_{n-k}) \quad (3)$$

Application

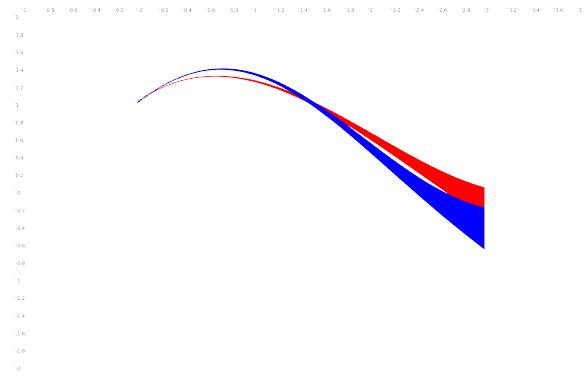
Double mass-spring-damper oscillator



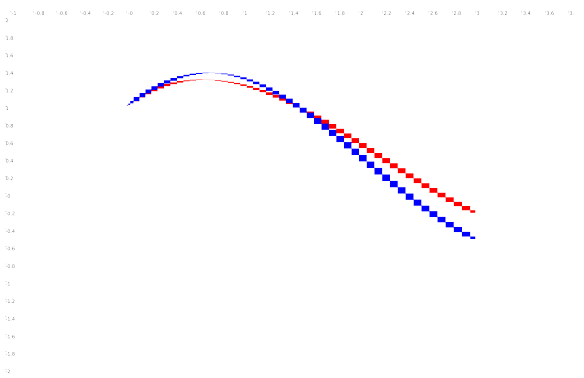
Heun



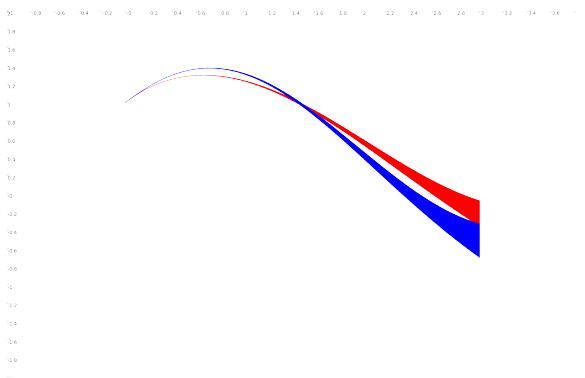
co-Heun (H=0.05)



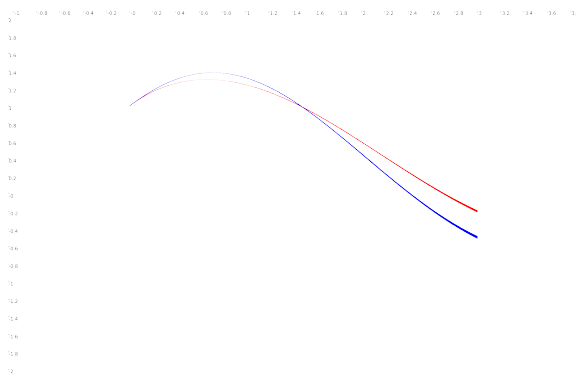
co-Heun-interp (H=0.05)



RK4



co-RK4 (H=0.01)



co-RK4-interp (H=0.01)

Seluxit case study

[K.G. Larsen et al. *Online and Compositional Learning of Controllers with Application to Floor Heating*, TACAS 2016.]

Seluxit case study

[K.G. Larsen et al. *Online and Compositional Learning of Controllers with Application to Floor Heating*, TACAS 2016.]

System dynamics:

$$\frac{d}{dt} T_i(t) = \sum_{j=1}^{n} A^d_{i,j}(T_j(t) - T_i(t)) + B_i(T_{env}(t) - T_i(t)) + H_{i,j}.v_j$$

- System of dimension 11
- 2^{11} combinations of v_j (not all admissible, constraint on the number of open valves)
- Pipes heating a room may influence other rooms
- Doors opening and closing (here: average between open and closed)
- Varying external temperature (here: $T_{env} = 10^\circ C$)
- Measures and switching every 15 minutes

Seluxit case study

[K.G. Larsen et al. *Online and Compositional Learning of Controllers with Application to Floor Heating*, TACAS 2016.]

Simulation over 10 switching times for a given switching sequence

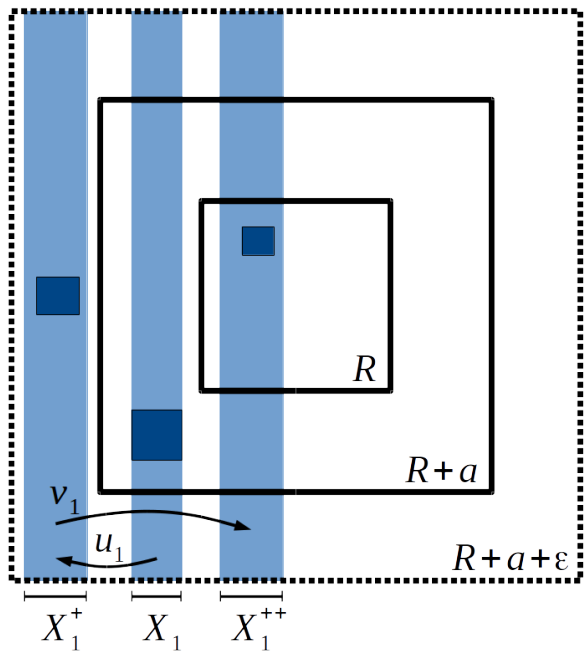
Scheme	Computation time (s)	Final area (m^2)
HEUN	7,96	0.2165
co-HEUN	5,95	0.2407
co-HEUN-interp	27,05	0.2335
RK4	27,60	0.1821
co-RK4	17,87	0.1932
co-RK4-interp	122,17	0.1854

Towards scalable CPS design

Model-based design of cyber-physical systems :

modeling, analysis, **controller synthesis**, **simulation**, deployment

- Progress is being made in scalability of controller synthesis methods
  - **Compositional methods**
    Break exponential complexity in dimension

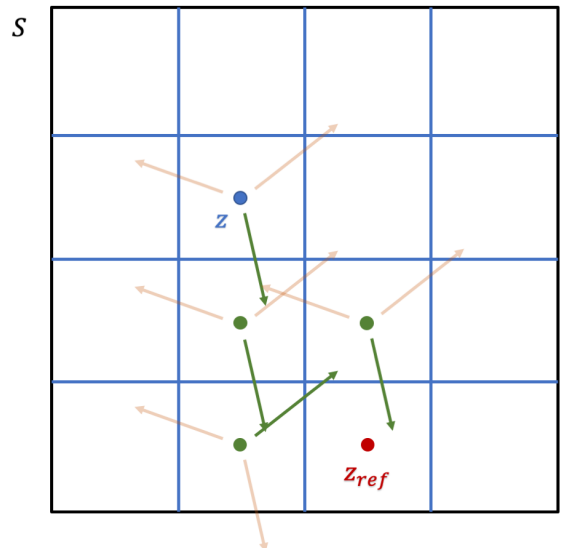


Towards scalable CPS design

Model-based design of cyber-physical systems :

modeling, analysis, **controller synthesis**, **simulation**, deployment

- Progress is being made in scalability of controller synthesis methods
  - **Compositional methods**
    Break exponential complexity in dimension
  - **Optimal control methods**
    From exponential to linear complexity in horizon length



Conclusions and future work

Conclusions:

- Guaranteed RK schemes available for systems with perturbations
- Guaranteed co-simulation possible using only local computations
- Co-simulation provides some significant computation time improvements

Conclusions and future work

Conclusions:

- Guaranteed RK schemes available for systems with perturbations
- Guaranteed co-simulation possible using only local computations
- Co-simulation provides some significant computation time improvements

Future work:

- Keep original coupling function formulation with DAE tools
- Parallelize computations and implement in DynIbex
- Test use in control synthesis algorithms
- More case studies (multi-physics)

Bibliography

- **Interval Runge-Kutta methods**
  [Julien Alexandre dit Sandretto and Alexandre Chapoutot. *Validated explicit and implicit runge-kutta methods*, Reliable Computing, 2016]
- **Automatic differentiation**
  [Olivier Mullier, Alexandre Chapoutot, and Julien Alexandre Dit Sandretto, *Validated computation of the local truncation error of runge-kutta methods with automatic differentiation*, Optimization Methods and Software, 2018]
- **Differential Algebraic Equations**
  [Julien Alexandre dit Sandretto and Alexandre Chapoutot. *Validated simulation of differential algebraic equations with runge-kutta methods*, Reliable Computing, 2016]
- **Guaranteed co-simulation**
  [Adrien Le Coënt, Julien Alexandre Dit Sandretto, Alexandre Chapoutot. *Guaranteed cosimulation of Cyber-Physical Systems*, 2020]