Euler's Method applied to the control of switched systems

FORMATS 2017 - Berlin

Laurent Fribourg[1]

September 6, 2017

[1]LSV - CNRS & ENS Cachan

Motivation

- Objective: control of switched systems: determine the active mode of the system over time (each mode described by ODE)

Motivation

- Objective: control of switched systems: determine the active mode of the system over time (each mode described by ODE)
- basic techniques:
  - space discretization (with tiles/boxes/interval vectors)
  - interval arithmetic used for set-valued integration

Motivation

- Objective: control of switched systems: determine the active mode of the system over time (each mode described by ODE)
- basic techniques:
  - space discretization (with tiles/boxes/interval vectors)
  - interval arithmetic used for set-valued integration
- alternative techniques:
  - error bound for Euler's method sharply estimated
  - application to controlled stability of switched systems

Motivation

- Objective: control of switched systems: determine the active mode of the system over time (each mode described by ODE)

- basic techniques:
  - space discretization (with tiles/boxes/interval vectors)
  - interval arithmetic used for set-valued integration

- alternative techniques:
  - error bound for Euler's method sharply estimated
  - application to controlled stability of switched systems

joint work with: A. Le Cöent, F. de Vuyst, L. Chamoin, J. Alexandre dit Sandretto, A. Chapoutot

Outline

Outline

Switched systems

A continuous switched system

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

- state $x(t) \in \mathbb{R}^n$
- control rule $\sigma(\cdot) : \mathbb{R}^+ \longrightarrow U$
- finite set of modes $U = \{1, \ldots, N\}$

Switched systems

A continuous switched system

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

- state $x(t) \in \mathbb{R}^n$
- control rule $\sigma(\cdot) : \mathbb{R}^+ \longrightarrow U$
- finite set of modes $U = \{1, \ldots, N\}$

Focus on time-sampled switched systems:
given a stepsize (or "sampling period") $\tau > 0$,
the mode switching occurs at times τ, 2τ, ...

Switched systems

A continuous switched system

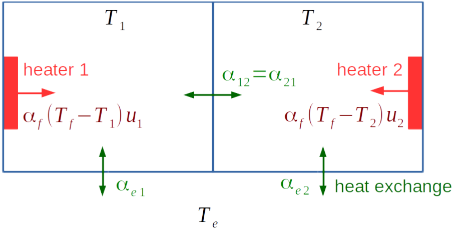
$$\dot{x}(t) = f_{\sigma(t)}(x(t))$$

- state $x(t) \in \mathbb{R}^n$
- control rule $\sigma(\cdot) : \mathbb{R}^+ \longrightarrow U$
- finite set of modes $U = \{1, \ldots, N\}$

Focus on time-sampled switched systems:
given a stepsize (or "sampling period") $\tau > 0$,
the mode switching occurs at times τ, 2τ, ...

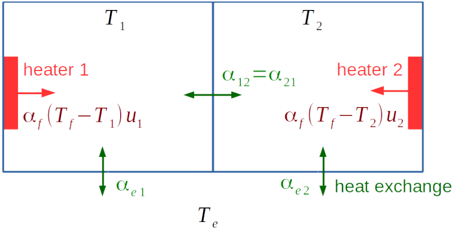
The control σ is a piecewise constant function
with equal steps of length τ, and height value in U

Example: Two-room apartment



$$\begin{pmatrix}\dot{T}_1\\ \dot{T}_2\end{pmatrix} = \begin{pmatrix}-\alpha_{21}-\alpha_{e1}-\alpha_f u_1 & \alpha_{21}\\ \alpha_{12} & -\alpha_{12}-\alpha_{e2}-\alpha_f u_2\end{pmatrix}\begin{pmatrix}T_1\\ T_2\end{pmatrix}+\begin{pmatrix}\alpha_{e1}T_e+\alpha_f T_f u_1\\ \alpha_{e2}T_e+\alpha_f T_f u_2\end{pmatrix}.$$

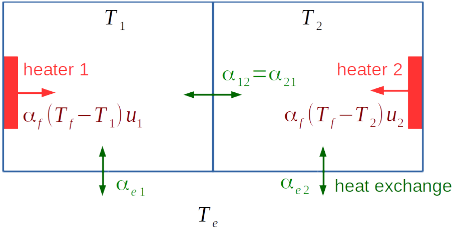
Example: Two-room apartment



$$\begin{pmatrix} \dot{T}_1 \\ \dot{T}_2 \end{pmatrix} = \begin{pmatrix} -\alpha_{21} - \alpha_{e1} - \alpha_f u_1 & \alpha_{21} \\ \alpha_{12} & -\alpha_{12} - \alpha_{e2} - \alpha_f u_2 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} + \begin{pmatrix} \alpha_{e1} T_e + \alpha_f T_f u_1 \\ \alpha_{e2} T_e + \alpha_f T_f u_2 \end{pmatrix}.$$

- Modes: $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$; stepsize τ

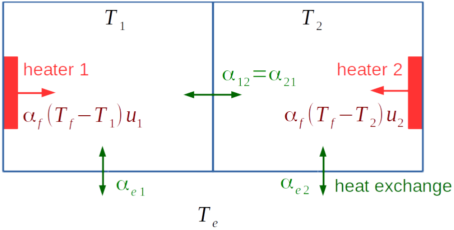
Example: Two-room apartment



$$\dot{T}_1(t) = f_1(T_1(t), T_2(t), u_1)$$
$$\dot{T}_2(t) = f_2(T_1(t), T_2(t), u_2)$$

- Modes: $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$; stepsize τ

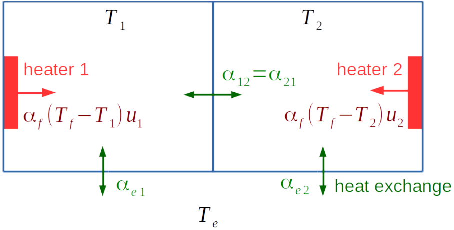
Example: Two-room apartment



$$\dot{T}_1(t) = f_1(T_1(t), T_2(t), u_1)$$
$$\dot{T}_2(t) = f_2(T_1(t), T_2(t), u_2)$$

- Modes: $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$; stepsize τ
- pattern: π is a finite sequence of modes, e.g. $\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right)$

Example: Two-room apartment



$$\dot{T}_1(t) = f_1(T_1(t), T_2(t), u_1)$$
$$\dot{T}_2(t) = f_2(T_1(t), T_2(t), u_2)$$

- Modes: $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$; stepsize τ
- pattern: π is a finite sequence of modes, e.g. $\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right)$
- state-dependent control: select at each τ a mode/pattern according to current state value x, in order to satisfy a desired property (eg: stability)

Controlled stability

Controlled stability

Given a a "safety" set S and a "recurrence" set $R \subseteq S$,

Controlled stability

Given a a "safety" set S and a "recurrence" set $R \subseteq S$,

select at each $t = \tau, 2\tau, \ldots$, a mode $j \in U$ (according to value $x(t)$) in order to satisfy

Controlled stability

Given a a "safety" set S and a "recurrence" set $R \subseteq S$,

select at each $t = \tau, 2\tau, \ldots$, a mode $j \in U$ (according to value $x(t)$) in order to satisfy

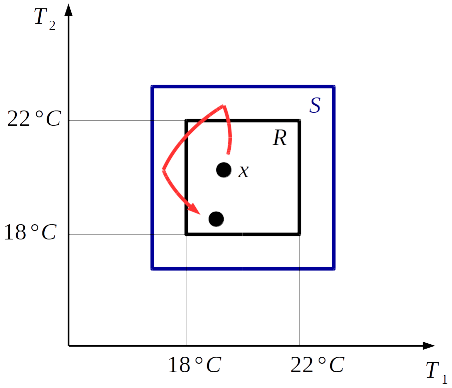
(R, S)-stability:

Controlled stability

Given a a "safety" set S and a "recurrence" set $R \subseteq S$,

select at each $t = \tau, 2\tau, \ldots,$ a mode $j \in U$ (according to value $x(t)$) in order to satisfy

(R, S)-stability:
$x(t)$ returns to R
while never leaving S



Outline

Interval arithmetic vs. standard arithmetic[1]

- standard numerical methods compute approximations to a mathematically correct result (due to finite representation of reals).

[1] Jackson 2011: https://cs.uwaterloo.ca/.../hybrid2011/slides/KenJackson.pdf

Interval arithmetic vs. standard arithmetic[1]

- standard numerical methods compute approximations to a mathematically correct result (due to finite representation of reals).
- interval methods [Moore66] manipulate set-valued real expressions: "interval vectors" or "boxes"

[1] Jackson 2011: https://cs.uwaterloo.ca/.../hybrid2011/slides/KenJackson.pdf

Interval arithmetic vs. standard arithmetic[1]

- standard numerical methods compute approximations to a mathematically correct result (due to finite representation of reals).
- interval methods [Moore66] manipulate set-valued real expressions: "interval vectors" or "boxes"
- they give bounds that are guaranteed to contain the mathematically correct result, using rules of the form:

[1] Jackson 2011: https://cs.uwaterloo.ca/.../hybrid2011/slides/KenJackson.pdf

Interval arithmetic vs. standard arithmetic[1]

- standard numerical methods compute approximations to a mathematically correct result (due to finite representation of reals).
- interval methods [Moore66] manipulate set-valued real expressions: "interval vectors" or "boxes"
- they give bounds that are guaranteed to contain the mathematically correct result, using rules of the form:
  - $[a] + [b] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$

[1] Jackson 2011: https://cs.uwaterloo.ca/.../hybrid2011/slides/KenJackson.pdf

Interval arithmetic vs. standard arithmetic[1]

- standard numerical methods compute approximations to a mathematically correct result (due to finite representation of reals).
- interval methods [Moore66] manipulate set-valued real expressions: "interval vectors" or "boxes"
- they give bounds that are guaranteed to contain the mathematically correct result, using rules of the form:
    - $[a] + [b] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$
    - $[a] \cdot [b] = [\min\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}, \max\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}]$

[1] Jackson 2011: https://cs.uwaterloo.ca/.../hybrid2011/slides/KenJackson.pdf

Interval arithmetic vs. standard arithmetic[1]

- standard numerical methods compute approximations to a mathematically correct result (due to finite representation of reals).
- interval methods [Moore66] manipulate set-valued real expressions: "interval vectors" or "boxes"
- they give bounds that are guaranteed to contain the mathematically correct result, using rules of the form:
  - $[a] + [b] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$
  - $[a] \cdot [b] = [\min\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}, \max\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}]$
- they can account for
  - rounding errors
  - inaccuracies in measurements of inputs
  - uncertainty on parameters, disturbance, errors from the model

[1] Jackson 2011: https://cs.uwaterloo.ca/.../hybrid2011/slides/KenJackson.pdf

Interval-based integration[2]

For $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, we consider the ODE

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

solution denoted by $x(t; x_0)$ (or simply $x(t)$)

[2] idem.

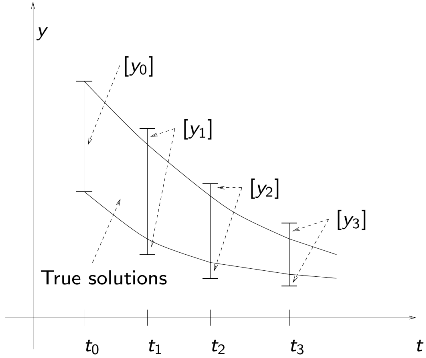
Interval-based integration[2]

For $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, we consider the ODE

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

solution denoted by $x(t; x_0)$ (or simply $x(t)$)

Goal: Given an interval I_0 at $t = t_0$, construct a sequence of intervals:



[2] idem.

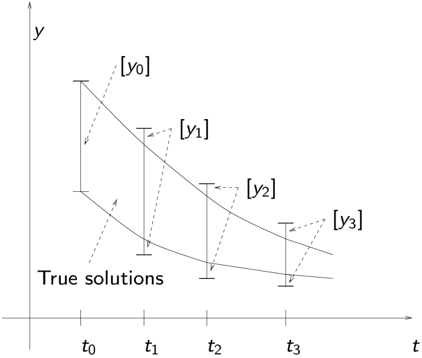
Interval-based integration[2]

For $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, we consider the ODE

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

solution denoted by $x(t; x_0)$ (or simply $x(t)$)

Goal: Given an interval I_0 at $t = t_0$, construct a sequence of intervals:



1. I_1 containing at $t_1 = t_0 + \tau$: $x(t_1; I_0) \equiv \{x(t_1; x_0) \mid x_0 \in I_0\}$

[2] idem.

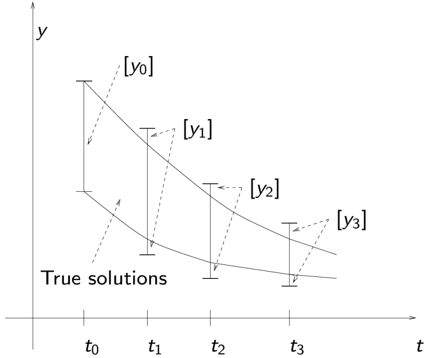
Interval-based integration[2]

For $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, we consider the ODE

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

solution denoted by $x(t; x_0)$ (or simply $x(t)$)

Goal: Given an interval I_0 at $t = t_0$, construct a sequence of intervals:



1. I_1 containing at $t_1 = t_0 + \tau$:
   $x(t_1; I_0) \equiv \{x(t_1; x_0) \mid x_0 \in I_0\}$
2. I_2 containing at $t_2 = t_1 + \tau$:
   $x(t_2; I_1) \equiv \{x(t_2; x_1) \mid x_1 \in I_1\}$

[2] idem.

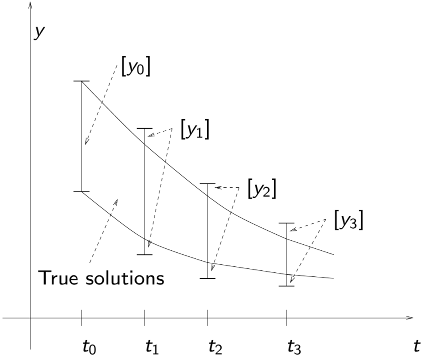
Interval-based integration[2]

For $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, we consider the ODE

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

solution denoted by $x(t; x_0)$ (or simply $x(t)$)

Goal: Given an interval I_0 at $t = t_0$, construct a sequence of intervals:



1. I_1 containing at $t_1 = t_0 + \tau$: $x(t_1; I_0) \equiv \{x(t_1; x_0) \mid x_0 \in I_0\}$
2. I_2 containing at $t_2 = t_1 + \tau$: $x(t_2; I_1) \equiv \{x(t_2; x_1) \mid x_1 \in I_1\}$
3. ...

[2] idem.

Interval-based integration[3]

Given I_j an interval for $t = t_j$, compute a (super)set of solutions I_{j+1} at $t_{j+1} = t_j + \tau$ via a two-step method:

[3] idem.

Interval-based integration[3]

Given I_j an interval for $t = t_j$, compute a (super)set of solutions I_{j+1} at $t_{j+1} = t_j + \tau$ via a two-step method:

1. Algorithm I: compute an a priori enclosure F_j:

$$x(t; I_j) \subseteq F_j \text{ for all } t \in [t_j, t_{j+1}]$$

[3] idem.

Interval-based integration[3]

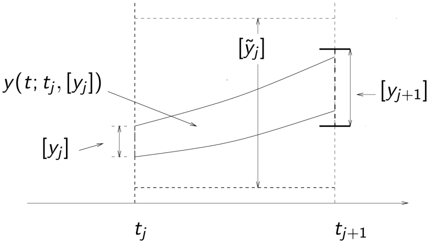
Given I_j an interval for $t = t_j$, compute a (super)set of solutions I_{j+1} at $t_{j+1} = t_j + \tau$ via a two-step method:

1. Algorithm I: compute an a priori enclosure F_j:

$$x(t; I_j) \subseteq F_j \text{ for all } t \in [t_j, t_{j+1}]$$

2. Algorithm II: compute a tighter enclosure I_{j+1}:

$$x(t; I_j) \subseteq I_{j+1} \subset F_j \text{ at } t = t_{j+1}$$



[3] idem.

Algorithm I: a priori enclosure method[4]

Basic property: If there exists an interval F:

1. $I_0 \subseteq F$, and
2. $I_0 + [0, \tau] \cdot f(F) \subseteq F$

[4] idem

Algorithm I: a priori enclosure method[4]

Basic property: If there exists an interval F:

1. $I_0 \subseteq F$, and
2. $I_0 + [0, \tau] \cdot f(F) \subseteq F$

then there exists a unique solution $x(t; x_0)$ for all $t \in [0, \tau]$, $x_0 \in I_0$. Furthermore: $x(t; x_0) \in F$.

[4] idem

Algorithm I: a priori enclosure method[4]

Basic property: If there exists an interval F:

1. $I_0 \subseteq F$, and
2. $I_0 + [0, \tau] \cdot f(F) \subseteq F$

then there exists a unique solution $x(t; x_0)$ for all $t \in [0, \tau]$, $x_0 \in I_0$.
Furthermore: $x(t; x_0) \in F$.

Proof based on Banach fixed-point th., and Picard-Lindelöf operator

$$(Tu)(t) = x_0 + \int_0^t f(u(s))ds.$$

[4] idem

Algorithm I: a priori enclosure method[4]

Basic property: If there exists an interval F:

1. $I_0 \subseteq F$, and
2. $I_0 + [0, \tau] \cdot f(F) \subseteq F$

then there exists a unique solution $x(t; x_0)$ for all $t \in [0, \tau]$, $x_0 \in I_0$.
Furthermore: $x(t; x_0) \in F$.

Proof based on Banach fixed-point th., and Picard-Lindelöf operator

$$(Tu)(t) = x_0 + \int_0^t f(u(s))ds.$$

The construction of F relies on fixed-point acceleration heuristics ("widening") using adjustment of stepsize τ.

[4] idem

Algorithm II: tighter enclosure[5]

Using F, compute a tighter enclosure I_1 of $x(t; I_0)$ for $\underline{t = \tau}$.

[5] idem

Algorithm II: tighter enclosure[5]

Using F, compute a tighter enclosure I_1 of $x(t; I_0)$ for $\underline{t = \tau}$.

Approach: Taylor series + remainder term.

$$x_1 = x_0 + \sum_{i=1}^{k-1} \tau^i \cdot f^{(i)}(x_0) + \tau^k \cdot f^{(k)}(y), \text{ for some } y \in F.$$

[5] idem

Algorithm II: tighter enclosure[5]

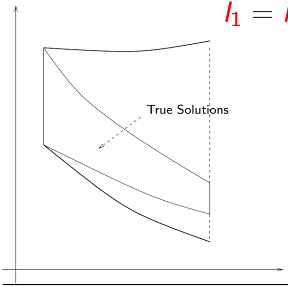
Using F, compute a tighter enclosure I_1 of $x(t; I_0)$ for $\underline{t = \tau}$.

Approach: Taylor series + remainder term.

$$x_1 = x_0 + \sum_{i=1}^{k-1} \tau^i \cdot f^{(i)}(x_0) + \tau^k \cdot f^{(k)}(y), \text{ for some } y \in F.$$

Hence

$$I_1 = I_0 + \sum_{i=1}^{k-1} \tau^i \cdot f^{(i)}(I_0) + \tau^k \cdot f^{(k)}(F)$$



[5] idem

Algorithm II: tighter enclosure[5]

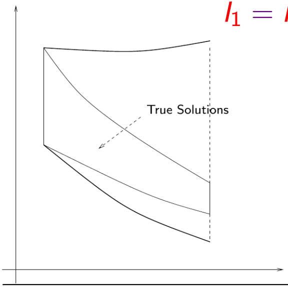
Using F, compute a tighter enclosure I_1 of $x(t; I_0)$ for $\underline{t = \tau}$.

Approach: Taylor series + remainder term.

$$x_1 = x_0 + \sum_{i=1}^{k-1} \tau^i \cdot f^{(i)}(x_0) + \tau^k \cdot f^{(k)}(y), \text{ for some } y \in F.$$

Hence

$$I_1 = I_0 + \sum_{i=1}^{k-1} \tau^i \cdot f^{(i)}(I_0) + \tau^k \cdot f^{(k)}(F)$$



NB: with this algo, $|I_1| > |I_0|$

– even if the true solutions contract!

$\rightarrow$ further refinement needed

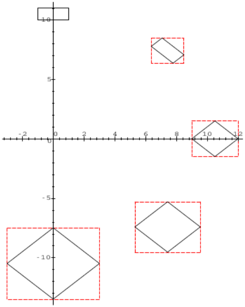
[5] idem

Wrapping effect[6]

A simple rotation:

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x; \quad x_0 \in I_0$$

The solution is $x(t) = \begin{pmatrix} cos(t) & sin(t) \\ -sin(t) & cos(t) \end{pmatrix} x_0$, where $x_0 \in I_0$



I_0 can be viewed as a parallelepiped.

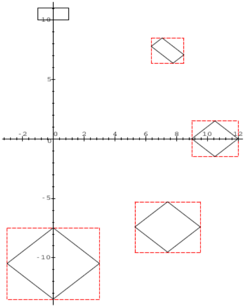
[6] idem

Wrapping effect[6]

A simple rotation:

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x; \quad x_0 \in I_0$$

The solution is $x(t) = \begin{pmatrix} cos(t) & sin(t) \\ -sin(t) & cos(t) \end{pmatrix} x_0$, where $x_0 \in I_0$



I_0 can be viewed as a parallelepiped.

At each step, the parallelepiped is rotated and has to be wrapped by another one.

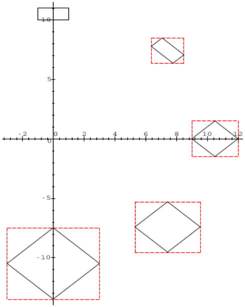
[6] idem

Wrapping effect[6]

A simple rotation:

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x; \quad x_0 \in I_0$$

The solution is $x(t) = \begin{pmatrix} cos(t) & sin(t) \\ -sin(t) & cos(t) \end{pmatrix} x_0$, where $x_0 \in I_0$



I_0 can be viewed as a parallelepiped.

At each step, the parallelepiped is rotated and has to be wrapped by another one.

At $t = 2\pi$, the blow up factor is by a factor $e^{2\pi} \approx 535$, as the stepsize tends to zero.

[6] idem

(Dis)advantages of interval methods[7]

Advantages over standard numerical methods:

1. ensure a unique solution exists
2. provide guaranteed bounds on the solution
3. can be efficient for problems with ranges of parameters

[7] idem

(Dis)advantages of interval methods[7]

Advantages over standard numerical methods:

1. ensure a unique solution exists
2. provide guaranteed bounds on the solution
3. can be efficient for problems with ranges of parameters

Disadvantages

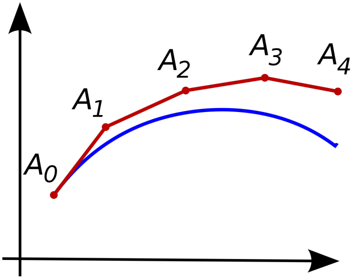
1. computation is time consuming
2. harder to implement than standard numerical methods
3. error bounds may be too large

[7] idem

Outline

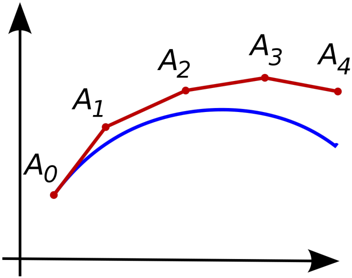
Euler's approximation $\tilde{x}(t)$ of $x(t)$

$$\tilde{x}(t) = \tilde{x}(t_i) + (t - t_i) \cdot f(\tilde{x}(t_i))$$



Euler's approximation $\tilde{x}(t)$ of $x(t)$

$$\tilde{x}(t) = \tilde{x}(t_i) + (t - t_i) \cdot f(\tilde{x}(t_i))$$



Piecewise linear fn.:
at each step, constant derivative of $\tilde{x}(t)$ $(= f(\tilde{x}(t_i))$ deriv. at starting pt)

Classical error bound (using Lipschitz constant L)

- The error at $t = t_0 + k\tau$ is: $\|x(t) - \tilde{x}(t)\|$.

Classical error bound (using Lipschitz constant L)

- The error at $t = t_0 + k\tau$ is: $\|x(t) - \tilde{x}(t)\|$.

  If f is Lipschitz cont. ($\|f(y) - f(x)\| \leq L\|y - x\|$), then:

  $$\text{error}(t) \leq \frac{\tau M}{2L}(e^{L(t-t_0)} - 1)$$

  where L is the Lipschitz constant of f (and M an upper bound on f'').

Classical error bound (using Lipschitz constant L)

- The error at $t = t_0 + k\tau$ is: $\|x(t) - \tilde{x}(t)\|$.

  If f is Lipschitz cont. ($\|f(y) - f(x)\| \leq L\|y - x\|$), then:

  $$\text{error}(t) \leq \frac{\tau M}{2L}(e^{L(t-t_0)} - 1)$$

  where L is the Lipschitz constant of f (and M an upper bound on f'').

- In case of "stiff" equations, L can be very big!

Classical error bound (using Lipschitz constant L)

- The error at $t = t_0 + k\tau$ is: $\|x(t) - \tilde{x}(t)\|$.

  If f is Lipschitz cont. ($\|f(y) - f(x)\| \leq L\|y - x\|$), then:

  $$\text{error}(t) \leq \frac{\tau M}{2L}(e^{L(t-t_0)} - 1)$$

  where L is the Lipschitz constant of f (and M an upper bound on f'').

- In case of “stiff” equations, L can be very big!

Idea: exploit another constant λ that will allow for a sharper estimation of Euler’s error

One-sided Lipschitz (OSL) constant λ

- $\lambda \in \mathbb{R}$ is a constant s.t., for all $x, y \in S$:

$$\langle f(y) - f(x), y - x \rangle \leq \lambda \|y - x\|^2$$

where $\langle \cdot, \cdot \rangle$ denote the scalar product of two vectors of $\mathbb{R}^n$

One-sided Lipschitz (OSL) constant λ

- $\lambda \in \mathbb{R}$ is a constant s.t., for all $x, y \in S$:

$$\langle f(y) - f(x), y - x \rangle \leq \lambda \|y - x\|^2$$

  where $\langle \cdot, \cdot \rangle$ denote the scalar product of two vectors of $\mathbb{R}^n$

- λ can be < 0 ($\rightarrow$ contractivity)

One-sided Lipschitz (OSL) constant λ

- $\lambda \in \mathbb{R}$ is a constant s.t., for all $x, y \in S$:

$$\langle f(y) - f(x), y - x \rangle \leq \lambda \|y - x\|^2$$

  where $\langle \cdot, \cdot \rangle$ denote the scalar product of two vectors of $\mathbb{R}^n$

- λ can be < 0 ($\rightarrow$ contractivity)

- even in case $\lambda > 0$, in practice: $\lambda \ll L$
  $\longrightarrow$ sharper estimation of Euler error

One-sided Lipschitz (OSL) constant λ

- $\lambda \in \mathbb{R}$ is a constant s.t., for all $x, y \in S$:

$$\langle f(y) - f(x), y - x \rangle \leq \lambda \|y - x\|^2$$

  where $\langle \cdot, \cdot \rangle$ denote the scalar product of two vectors of $\mathbb{R}^n$

- λ can be < 0 ($\rightarrow$ contractivity)

- even in case $\lambda > 0$, in practice: $\lambda \ll L$
  $\longrightarrow$ sharper estimation of Euler error

- λ can be computed using constraint optimization algorithms

Hypotheses

$(H0)$ (Lipschitz): for all $j \in U$, there exists a constant $L_j > 0$ such that:

$$\|f_j(y) - f_j(x)\| \leq L_j \|y - x\| \quad \forall x, y \in S.$$

[8] T is the one-step expansion of S under all the modes j of U

Hypotheses

$(H0)$ (Lipschitz): for all $j \in U$, there exists a constant $L_j > 0$ such that:

$$\|f_j(y) - f_j(x)\| \leq L_j \|y - x\| \quad \forall x, y \in S.$$

$(H1)$ (one-sided Lipschitz): for all $j \in U$, there exists a constant $\lambda_j \in \mathbb{R}$ such that

$$\langle f_j(y) - f_j(x), y - x \rangle \leq \lambda_j \|y - x\|^2 \quad \forall x, y \in T^8,$$

[8] T is the one-step expansion of S under all the modes j of U

Hypotheses

$(H0)$ (Lipschitz): for all $j \in U$, there exists a constant $L_j > 0$ such that:

$$\|f_j(y) - f_j(x)\| \leq L_j \|y - x\| \quad \forall x, y \in S.$$

$(H1)$ (one-sided Lipschitz): for all $j \in U$, there exists a constant $\lambda_j \in \mathbb{R}$ such that

$$\langle f_j(y) - f_j(x), y - x \rangle \leq \lambda_j \|y - x\|^2 \quad \forall x, y \in T^8,$$

The constants C_j for all $j \in U$ are defined as follows:

$$C_j = \sup_{x \in S} \; L_j \|f_j(x)\|.$$

[8] T is the one-step expansion of S under all the modes j of U

Notations

Let $x_j(t)$ the solution at time t of the system under mode j with (implicit) initial point x^0

$$\dot{x}(t) = f_j(x(t)),$$
$$x(0) = x^0.$$

Notations

Let $x_j(t)$ the solution at time t of the system under mode j with (implicit) initial point x^0

$$\dot{x}(t) = f_j(x(t)),$$
$$x(0) = x^0.$$

Given an (approximate) initial point $\tilde{x}^0 \in S$ and a mode $j \in U$,

the Euler approximate, denoted by $\tilde{x}_j(t; \tilde{x}^0)$, is defined by:

$$\tilde{x}_j(t; \tilde{x}^0) = \tilde{x}^0 + t \cdot f_j(\tilde{x}^0), \quad \text{with } t \in [0, \tau]$$

Notations

Let $x_j(t)$ the solution at time t of the system under mode j with (implicit) initial point x^0

$$\dot{x}(t) = f_j(x(t)),$$
$$x(0) = x^0.$$

Given an (approximate) initial point $\tilde{x}^0 \in S$ and a mode $j \in U$,

the Euler approximate, denoted by $\tilde{x}_j(t;\tilde{x}^0)$, is defined by:

$$\tilde{x}_j(t;\tilde{x}^0) = \tilde{x}^0 + t \cdot f_j(\tilde{x}^0), \quad \text{with } t \in [0,\tau]$$

We are going to determine an upper bound $\delta_j(t)$ to

$$\text{error}_j(t) \equiv \|x_j(t;x^0) - \tilde{x}_j(t;\tilde{x}^0)\|,$$

Notations

Let $x_j(t)$ the solution at time t of the system under mode j with (implicit) initial point x^0

$$\dot{x}(t) = f_j(x(t)),$$
$$x(0) = x^0.$$

Given an (approximate) initial point $\tilde{x}^0 \in S$ and a mode $j \in U$,

the Euler approximate, denoted by $\tilde{x}_j(t; \tilde{x}^0)$, is defined by:

$$\tilde{x}_j(t; \tilde{x}^0) = \tilde{x}^0 + t \cdot f_j(\tilde{x}^0), \quad \text{with } t \in [0, \tau]$$

We are going to determine an upper bound $\delta_j(t)$ to

$$\text{error}_j(t) \equiv \| x_j(t; x^0) - \tilde{x}_j(t; \tilde{x}^0) \|,$$

assuming $\text{error}_j(0) \equiv \| x^0 - \tilde{x}^0 \| \leq \delta^0$ for some $\delta^0 \in \mathbb{R}_+$.

Basic result: local error $\delta_j(t)$ using λ_j

Basic result: local error $\delta_j(t)$ using λ_j

Theorem

Given a system satisfying (H0-H1), an approximate initial pt $\tilde{x}^0$*, a positive real* δ^0 *and* $j \in U$*, we have:*

Basic result: local error $\delta_j(t)$ using λ_j

Theorem

Given a system satisfying (H0-H1), an approximate initial pt $\tilde{x}^0$, a positive real δ^0 and $j \in U$, we have:

For all initial point $x^0 \in B(\tilde{x}^0, \delta^0)$,

$$x_j(t; x^0) \in B(\tilde{x}_j(t; \tilde{x}^0), \delta_j(t)) \quad \textit{for all } t \in [0, \tau].$$

Basic result: local error $\delta_j(t)$ using λ_j

Theorem

Given a system satisfying (H0-H1), an approximate initial pt $\tilde{x}^0$*, a positive real* δ^0 *and* $j \in U$*, we have:*

For all initial point $x^0 \in B(\tilde{x}^0, \delta^0)$,

$$x_j(t; x^0) \in B(\tilde{x}_j(t; \tilde{x}^0), \delta_j(t)) \quad \textit{for all } t \in [0, \tau].$$

with

- *if* $\lambda_j < 0: \ \delta_j(t) = \left((\delta^0)^2 e^{\lambda_j t} + \frac{C_j^2}{\lambda_j{}^2} \left(t^2 + \frac{2t}{\lambda_j} + \frac{2}{\lambda_j{}^2} \left(1 - e^{\lambda_j t} \right) \right) \right)^{\frac{1}{2}}$

Basic result: local error $\delta_j(t)$ using λ_j

Theorem

Given a system satisfying (H0-H1), an approximate initial pt $\tilde{x}^0$*, a positive real* δ^0 *and* $j \in U$*, we have:*

For all initial point $x^0 \in B(\tilde{x}^0, \delta^0)$,

$$x_j(t; x^0) \in B(\tilde{x}_j(t; \tilde{x}^0), \delta_j(t)) \quad \textit{for all } t \in [0, \tau].$$

with

- *if* $\lambda_j < 0$: $\delta_j(t) = \left((\delta^0)^2 e^{\lambda_j t} + \frac{C_j^2}{\lambda_j{}^2} \left(t^2 + \frac{2t}{\lambda_j} + \frac{2}{\lambda_j{}^2} \left(1 - e^{\lambda_j t} \right) \right) \right)^{\frac{1}{2}}$
- *if* $\lambda_j = 0$: $\delta_j(t) = \left((\delta^0)^2 e^t + C_j^2(-t^2 - 2t + 2(e^t - 1)) \right)^{\frac{1}{2}}$

Basic result: local error $\delta_j(t)$ using λ_j

Theorem

Given a system satisfying (H0-H1), an approximate initial pt $\tilde{x}^0$, a positive real δ^0 and $j \in U$, we have:

For all initial point $x^0 \in B(\tilde{x}^0, \delta^0)$,

$$x_j(t; x^0) \in B(\tilde{x}_j(t; \tilde{x}^0), \delta_j(t)) \quad \textit{for all } t \in [0, \tau].$$

with

- *if* $\lambda_j < 0$: $\delta_j(t) = \left((\delta^0)^2 e^{\lambda_j t} + \frac{C_j^2}{\lambda_j^2} \left(t^2 + \frac{2t}{\lambda_j} + \frac{2}{\lambda_j^2} \left(1 - e^{\lambda_j t} \right) \right) \right)^{\frac{1}{2}}$
- *if* $\lambda_j = 0$: $\delta_j(t) = \left((\delta^0)^2 e^t + C_j^2 (-t^2 - 2t + 2(e^t - 1)) \right)^{\frac{1}{2}}$
- *if* $\lambda_j > 0$:

$$\delta_j(t) = \left((\delta^0)^2 e^{3\lambda_j t} + \frac{C_j^2}{3\lambda_j^2} \left(-t^2 - \frac{2t}{3\lambda_j} + \frac{2}{9\lambda_j^2} \left(e^{3\lambda_j t} - 1 \right) \right) \right)^{\frac{1}{2}}$$

Application to one-step controlled safety

Application to one-step controlled safety

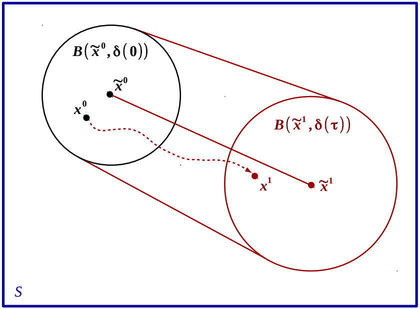
Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subset S$, safely control B^0 during one step, select $j \in U$:

$$x_j(t; B^0) \subseteq S, \qquad \forall t \in [0, \tau]$$

Application to one-step controlled safety

Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subset S$, safely control B^0 during one step, select $j \in U$:

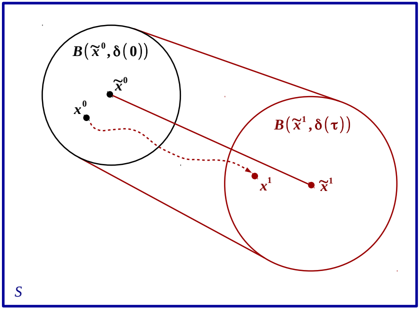
$$x_j(t; B^0) \subseteq S, \qquad \forall t \in [0, \tau]$$



Application to one-step controlled safety

Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subset S$, safely control B^0 during one step, select $j \in U$:

$$x_j(t; B^0) \subseteq S, \qquad \forall t \in [0, \tau]$$

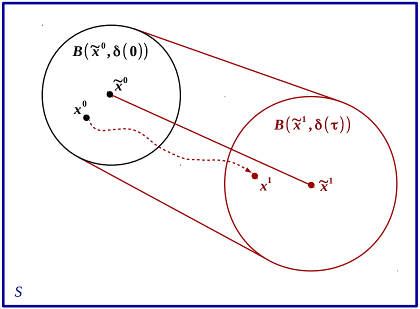


it suffices to find $j \in U$: $\quad B^1 \equiv B(\tilde{x}^1, \delta_j(\tau)) \subseteq S \quad$ with $\tilde{x}^1 \equiv \tilde{x}^0 + \tau \cdot f_j(\tilde{x}^0)$

Application to one-step controlled safety

Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subset S$, safely control B^0 during one step, select $j \in U$:

$$x_j(t; B^0) \subseteq S, \qquad \forall t \in [0, \tau]$$



it suffices to find $j \in U$: $\quad B^1 \equiv B(\tilde{x}^1, \delta_j(\tau)) \subseteq S \quad$ with $\tilde{x}^1 \equiv \tilde{x}^0 + \tau \cdot f_j(\tilde{x}^0)$
provided δ_j verified to be convex on $[0, \tau]$

Remarks on the form of $\delta_j(\cdot)$

ex: DC-DC converter

modes given by $\dot{x}(t) = A_{\sigma(t)}x(t) + B_{\sigma(t)}$ with $\sigma(t) \in U = \{1, 2\}$,

$$A_1 = \begin{pmatrix} -\frac{r_l}{x_l} & 0 \\ 0 & -\frac{1}{x_c}\frac{1}{r_0+r_c} \end{pmatrix} \quad B_1 = \begin{pmatrix} \frac{v_s}{x_l} \\ 0 \end{pmatrix}$$

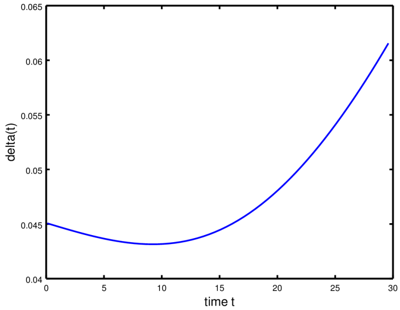
$$A_2 = \begin{pmatrix} -\frac{1}{x_l}(r_l + \frac{r_0.r_c}{r_0+r_c}) & -\frac{1}{x_l}\frac{r_0}{r_0+r_c} \\ \frac{1}{x_c}\frac{r_0}{r_0+r_c} & -\frac{1}{x_c}\frac{r_0}{r_0+r_c} \end{pmatrix} \quad B_2 = \begin{pmatrix} \frac{v_s}{x_l} \\ 0 \end{pmatrix}$$

with $x_c = 70$, $x_l = 3$, $r_c = 0.005$, $r_l = 0.05$, $r_0 = 1$, $v_s = 1$.

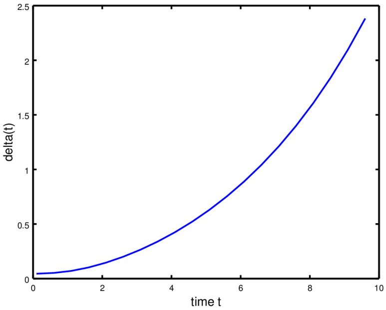
λ_1	-0.0142
λ_2	0.142
C_1	6.7126×10^{-5}
C_2	2.6229×10^{-2}

Remarks on the form of $\delta_j(\cdot)$

ex: DC-DC converter



$\lambda_1 = -0.0142 < 0$



$\lambda_2 = 0.142 > 0$

For mode 1 ($\lambda_1 < 0$): optimal stepsize τ corresponding to minimum of δ_1

For mode 2 ($\lambda_2 > 0$): δ_2 always $\nearrow$

$\rightarrow$ suggests subsampling of τ for achieving better precision

No wrapping effect in the rotation example

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x$$

No wrapping effect in the rotation example

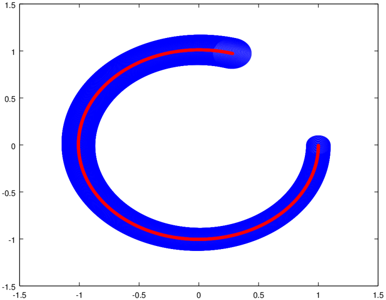
$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x$$

constants: $\lambda = 0$, $C = 4.2$, $L = 1$
initial error: $\delta^0 = 0.1$
stepsize: $\tau = 0.005$

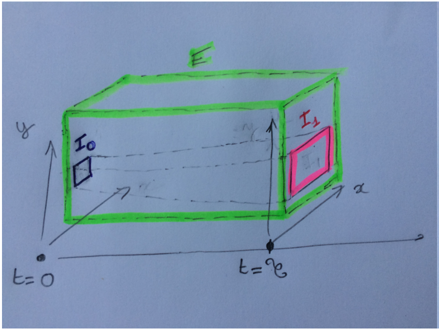
No wrapping effect in the rotation example

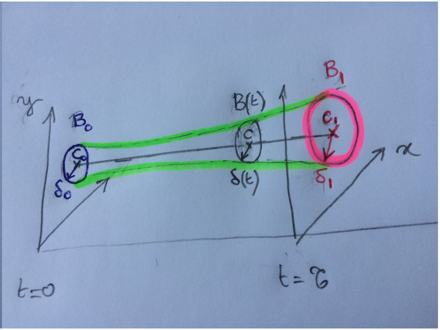
$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} x$$

constants: $\lambda = 0$, $C = 4.2$, $L = 1$
initial error: $\delta^0 = 0.1$
stepsize: $\tau = 0.005$

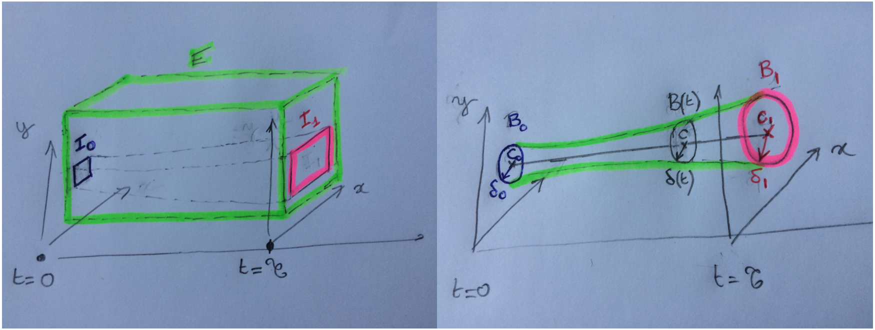


Recap': Interval-based vs. Euler-based method



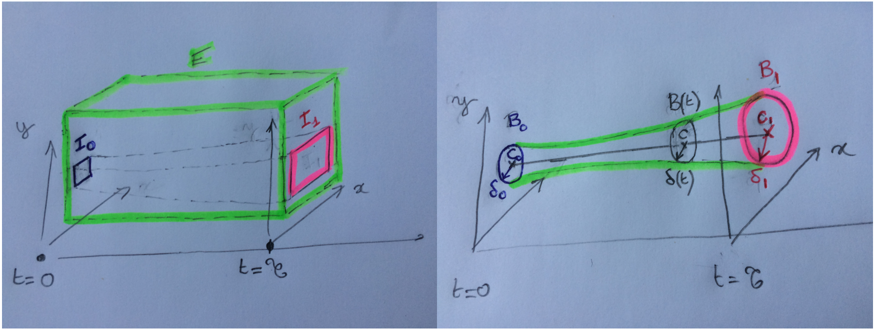


Recap': Interval-based vs. Euler-based method



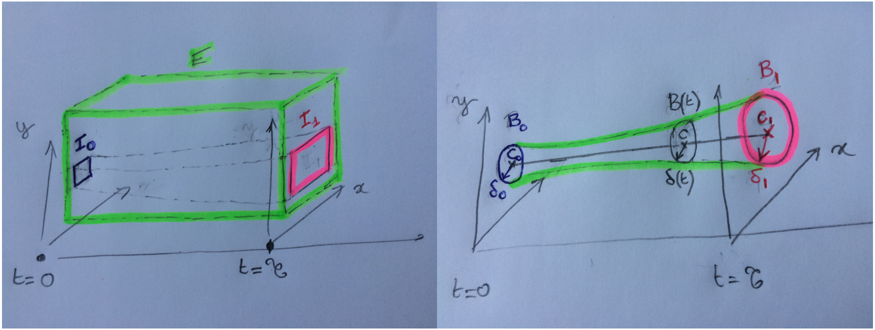
- <u>input/output</u>: intervals I_0, I_1 vs ball $B_0 \equiv B(C_0, \delta_0)$, $B_1 \equiv B(C_1, \delta_1)$

Recap': Interval-based vs. Euler-based method



- input/output: intervals I_0, I_1 vs ball $B_0 \equiv B(C_0, \delta_0)$, $B_1 \equiv B(C_1, \delta_1)$
- method: I_1 computed from I_0 using intermediate structure F

Recap': Interval-based vs. Euler-based method



- input/output: intervals I_0, I_1 vs ball $B_0 \equiv B(C_0, \delta_0)$, $B_1 \equiv B(C_1, \delta_1)$
- method: I_1 computed from I_0 using intermediate structure F

  vs. B_1 evaluated directly from C_0 and δ_0

Euler-based integration (vs. interval integration)

- Advantages:
  1. Computationally very cheap (standard arithmetic, no need for computation of f derivatives, δ_j pre-computed)
  2. allows a priori for longer stepsize τ (often)
  3. reduces wrapping effect (sometimes)
  4. well-suited to controlled safety

Euler-based integration (vs. interval integration)

- Advantages:
  1. Computationally very cheap (standard arithmetic, no need for computation of f derivatives, δ_j pre-computed)
  2. allows a priori for longer stepsize τ (often)
  3. reduces wrapping effect (sometimes)
  4. well-suited to controlled safety
- Limits:
  - less precise than interval-based integration method

    (1st order Taylor method vs. higher order Taylor method)

Outline

One-step controlled safety

One-step controlled safety

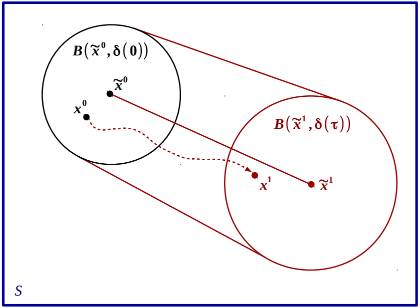
Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subseteq S$, select a mode j:

$$x_j(t; B^0) \subseteq S \qquad \text{for all } t \in [0, \tau]$$

One-step controlled safety

Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subseteq S$, select a mode j:

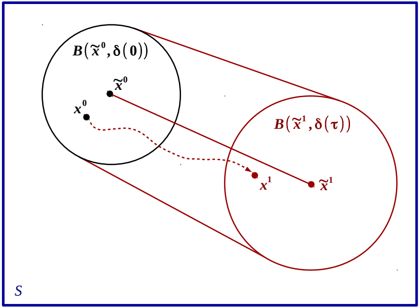
$$x_j(t; B^0) \subseteq S \qquad \text{for all } t \in [0, \tau]$$



One-step controlled safety

Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subseteq S$, select a mode j:

$$x_j(t; B^0) \subseteq S \qquad \text{for all } t \in [0, \tau]$$



It suffices to find j:

$$B^1 \equiv B(\tilde{x}^1, \delta^1) \subseteq S \text{ with } \quad \tilde{x}^1 = \tilde{x}^0 + \tau \cdot f_j(\tilde{x}^0) \quad \text{and} \quad \delta^1 = \delta_j(\tau)$$

assuming $\delta_j(\cdot)$ convex

Multi-step controlled safety

Multi-step controlled safety

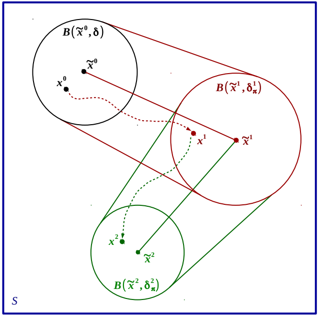
Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subset S$, select a pattern π (of length k):

$$x(t; B^0) \in S \quad \text{for all } t \in [0, k\tau]$$

Multi-step controlled safety

Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subset S$, select a pattern π (of length k):

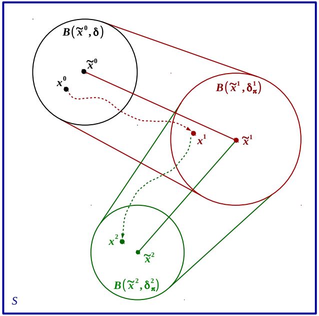
$$x(t; B^0) \in S \quad \text{for all } t \in [0, k\tau]$$



Multi-step controlled safety

Given a ball $B^0 \equiv B(\tilde{x}^0, \delta^0) \subset S$, select a pattern π (of length k):

$$x(t; B^0) \in S \quad \text{for all } t \in [0, k\tau]$$



It suffices to find a pattern $\pi \equiv j_1 \cdots j_k$:

$$B^1 \equiv B(\tilde{x}^1, \delta^1_{j_1}) \subset S, \quad \ldots, \quad B^k \equiv B(\tilde{x}^k, \delta^k_{j_k}) \subset S$$

Controlled (R,S)-stability

Controlled (R,S)-stability

1. Find a set of initial balls $B_i^0 \equiv B(\tilde{x}_i^0, \delta^0) \subset S$ covering R

Controlled (R,S)-stability

1. Find a set of initial balls $B_i^0 \equiv B(\tilde{x}_i^0, \delta^0) \subset S$ covering R
2. For each B_i^0 select a pattern π_i of the form $j_1 \cdots j_{k_i}$:

Controlled (R,S)-stability

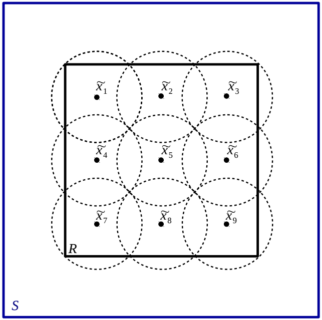
1. Find a set of initial balls $B_i^0 \equiv B(\tilde{x}_i^0, \delta^0) \subset S$ covering R
2. For each B_i^0 select a pattern π_i of the form $j_1 \cdots j_{k_i}$:
   - safety: all the balls $B_i^1 \equiv B(\tilde{x}_i^1, \delta^1), ..., B_i^{k_i} \equiv B(\tilde{x}_i^{k_i}, \delta^{k_i})$ are $\subseteq S$, and

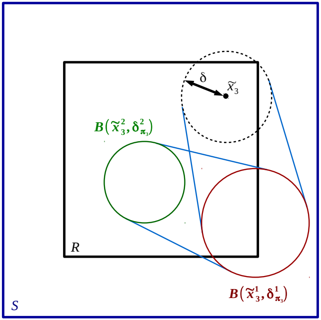
Controlled (R,S)-stability

1. Find a set of initial balls $B_i^0 \equiv B(\tilde{x}_i^0, \delta^0) \subset S$ covering R
2. For each B_i^0 select a pattern π_i of the form $j_1 \cdots j_{k_i}$:
   - safety: all the balls $B_i^1 \equiv B(\tilde{x}_i^1, \delta^1), ..., B_i^{k_i} \equiv B(\tilde{x}_i^{k_i}, \delta^{k_i})$ are $\subseteq S$, and
   - recurrence: the last ball $B_i^{k_i}$ is $\subseteq R$

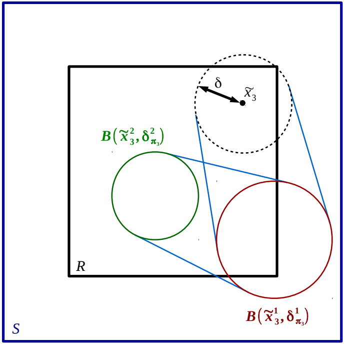
Controlled (R,S)-stability

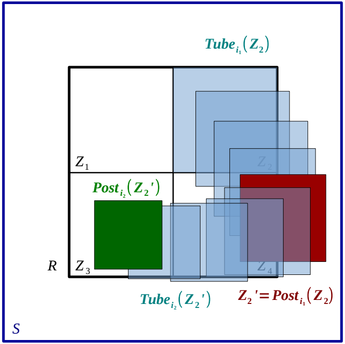
1. Find a set of initial balls $B_i^0 \equiv B(\tilde{x}_i^0, \delta^0) \subset S$ covering R
2. For each B_i^0 select a pattern π_i of the form $j_1 \cdots j_{k_i}$:
   - safety: all the balls $B_i^1 \equiv B(\tilde{x}_i^1, \delta^1), \dots, B_i^{k_i} \equiv B(\tilde{x}_i^{k_i}, \delta^{k_i})$ are $\subseteq S$, and
   - recurrence: the last ball $B_i^{k_i}$ is $\subseteq R$





Euler-based control vs. interval-based control





Example: Building ventilation

[Meyer, Nazarpour, Girard, Witrant, 2014]

Dynamics of a four-room apartment:

$$\frac{dT_i}{dt} = \sum_{j \in \mathcal{U}^*} a_{ij}(T_j - T_i) + \delta_{s_i} b_i (T_{s_i}^4 - T_i^4) + c_i \max\left(0, \frac{V_i - V_i^*}{\bar{V}_i - V_i^*}\right)(T_u - T_i).$$

with $\mathcal{U}^* = \{1, 2, 3, 4, u, o, c\}$

16 switching modes

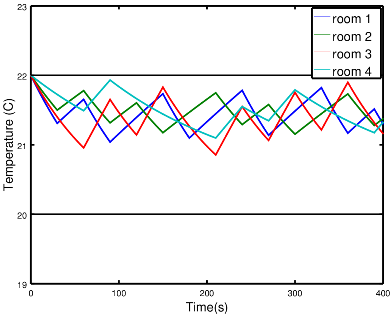
(control inputs: V_1, $V_4 \in \{$ 0V, 3.5V$\}$, and V_2, $V_3 \in \{$ 0V, 3V$\}$)

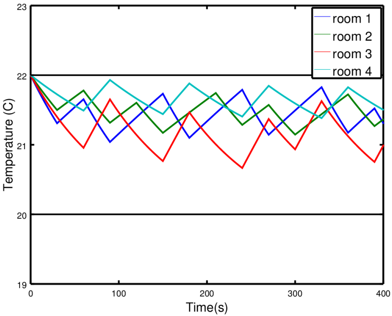
Building ventilation

	Euler	DynIBEX
R S	$[20,22]^4$ $[19,23]^4$	
τ	30	
Time subsampling	No	
Complete control	Yes	Yes
$\max_{j=1,\ldots,16} \lambda_j$ $\max_{j=1,\ldots,16} C_j$	-6.30×10^{-3} 4.18×10^{-6}	
Number of balls/tiles Pattern length	4096 1	252 1
CPU time	63 seconds	249 seconds

Control based on Euler (left) and interval (right).

Building ventilation





Control based on Euler (left) and interval (right).

Outline

Robust OSL condition w.r.t. disturbance $w \in W$

Consider: $\dot{x}(t) = f(x(t), w(t))$ with $w(t) \in W$ for all $t \in [0, \tau]$.

Robust OSL condition w.r.t. disturbance $w \in W$

Consider: $\dot{x}(t) = f(x(t), w(t))$ with $w(t) \in W$ for all $t \in [0, \tau]$.

The eq. $\dot{x} = f(x, w)$ with $w \in W$ is said to be robustly OSL w.r.t disturbance set W if

Robust OSL condition w.r.t. disturbance $w \in W$

Consider: $\dot{x}(t) = f(x(t), w(t))$ with $w(t) \in W$ for all $t \in [0, \tau]$.

The eq. $\dot{x} = f(x, w)$ with $w \in W$ is said to be robustly OSL
w.r.t disturbance set W if

$\exists\ \lambda \in \mathbb{R}$ and $\gamma \in \mathbb{R}_{\geq 0}$ s.t.

Robust OSL condition w.r.t. disturbance $w \in W$

Consider: $\dot{x}(t) = f(x(t), w(t))$ with $w(t) \in W$ for all $t \in [0, \tau]$.

The eq. $\dot{x} = f(x, w)$ with $w \in W$ is said to be robustly OSL w.r.t disturbance set W if

$\exists\ \lambda \in \mathbb{R}$ and $\gamma \in \mathbb{R}_{\geq 0}$ s.t.

(H_W): $\forall x, x' \in T$, $\forall w, w' \in W$

$$\langle f(x, w) - f(x', w'), x - x' \rangle \leq \lambda \|x - x'\|^2 + \gamma \|x - x'\| \|w - w'\|.$$

E. error function δ_W in presence of disturbance $w \in W$

Consider the ODE: $\dot{x}(t) = f(x(t), w(t))$ with $w(t) \in W$ for all $t \in [0, \tau]$.

E. error function δ_W in presence of disturbance $w \in W$

The fn δ_W (s.t: for all $t \in [0, \tau]$, $w(t) \in W$: $\|x(t) - \tilde{x}(t)\| \leq \delta_W(t)$)
can now be defined by:

E. error function δ_W in presence of disturbance $w \in W$

- if $\lambda < 0$,

$$\delta_W(t) = \left(\frac{C^2}{-\lambda^4} \left(-\lambda^2 t^2 - 2\lambda t + 2e^{\lambda t} - 2 \right) \right.$$
$$+ \frac{1}{\lambda^2} \left(\frac{C\gamma|W|}{-\lambda} \left(-\lambda t + e^{\lambda t} - 1 \right) \right.$$
$$\left. \left. + \lambda \left(\frac{(\gamma)^2(|W|/2)^2}{-\lambda}(e^{\lambda t} - 1) + \lambda(\delta^0)^2 e^{\lambda t} \right) \right) \right)^{1/2} \quad (1)$$

E. error function δ_W in presence of disturbance $w \in W$

- if $\lambda < 0$,

$$\delta_W(t) = \left(\frac{C^2}{-\lambda^4} \left(-\lambda^2 t^2 - 2\lambda t + 2e^{\lambda t} - 2 \right) \right.$$
$$+ \frac{1}{\lambda^2} \left(\frac{C\gamma|W|}{-\lambda} \left(-\lambda t + e^{\lambda t} - 1 \right) \right.$$
$$\left. \left. + \lambda \left(\frac{(\gamma)^2(|W|/2)^2}{-\lambda} (e^{\lambda t} - 1) + \lambda(\delta^0)^2 e^{\lambda t} \right) \right) \right)^{1/2} \quad (1)$$

- if $\lambda > 0$,

$$\delta_W(t) = \frac{1}{(3\lambda)^{3/2}} \left(\frac{C^2}{\lambda} \left(-9\lambda^2 t^2 - 6\lambda t + 2e^{3\lambda t} - 2 \right) \right.$$
$$+ 3\lambda \left(\frac{C\gamma|W|}{\lambda} \left(-3\lambda t + e^{3\lambda t} - 1 \right) \right.$$
$$\left. \left. + 3\lambda \left(\frac{(\gamma)^2(|W|/2)^2}{\lambda} (e^{3\lambda t} - 1) + 3\lambda(\delta^0)^2 e^{3\lambda t} \right) \right) \right)^{1/2} \quad (2)$$

E. error function δ_W in presence of disturbance $w \in W$

- if $\lambda < 0$,

$$\delta_W(t) = \left(\frac{C^2}{-\lambda^4} \left(-\lambda^2 t^2 - 2\lambda t + 2e^{\lambda t} - 2 \right) + \frac{1}{\lambda^2} \left(\frac{C\gamma|W|}{-\lambda} \left(-\lambda t + e^{\lambda t} - 1 \right) + \lambda \left(\frac{(\gamma)^2(|W|/2)^2}{-\lambda}(e^{\lambda t} - 1) + \lambda(\delta^0)^2 e^{\lambda t} \right) \right) \right)^{1/2} \tag{1}$$

- if $\lambda > 0$,

$$\delta_W(t) = \frac{1}{(3\lambda)^{3/2}} \left(\frac{C^2}{\lambda} \left(-9\lambda^2 t^2 - 6\lambda t + 2e^{3\lambda t} - 2 \right) + 3\lambda \left(\frac{C\gamma|W|}{\lambda} \left(-3\lambda t + e^{3\lambda t} - 1 \right) + 3\lambda \left(\frac{(\gamma)^2(|W|/2)^2}{\lambda}(e^{3\lambda t} - 1) + 3\lambda(\delta^0)^2 e^{3\lambda t} \right) \right) \right)^{1/2} \tag{2}$$

- if $\lambda = 0$,

$$\delta_W(t) = \left(C^2 \left(-t^2 - 2t + 2e^t - 2 \right) + \left(C\gamma|W| \left(-t + e^t - 1 \right) + ((\gamma)^2(|W|/2)^2(e^t - 1) + (\delta^0)^2 e^t) \right) \right)^{1/2} \tag{3}$$

Compositional $(R_1 \times R_2, S_1 \times S_2)$-stability

$$\dot{x}_1 = f^1(x_1, x_2)$$
$$\dot{x}_2 = f^2(x_1, x_2)$$

Compositional $(R_1 \times R_2, S_1 \times S_2)$-stability

$$\dot{x}_1 = f^1(x_1, x_2)$$
$$\dot{x}_2 = f^2(x_1, x_2)$$

Suppose:

- (H_{1,T_2}): $\dot{x}_1 = f^1(x_1, x_2)$ is robustly OSL w.r.t $x_2 \in T_2$, with λ^1, γ^1.
- (H_{2,T_1}): $\dot{x}_2 = f^2(x_1, x_2)$ is robustly OSL w.r.t $x_1 \in T_1$, with λ^2, γ^2.

Compositional $(R_1 \times R_2, S_1 \times S_2)$-stability

$$\dot{x}_1 = f^1(x_1, x_2)$$
$$\dot{x}_2 = f^2(x_1, x_2)$$

Suppose:

- (H_{1,T_2}): $\dot{x}_1 = f^1(x_1, x_2)$ is robustly OSL w.r.t $x_2 \in T_2$, with λ^1, γ^1.
- (H_{2,T_1}): $\dot{x}_2 = f^2(x_1, x_2)$ is robustly OSL w.r.t $x_1 \in T_1$, with λ^2, γ^2.

Theorem (compositionality): If

- σ_1 is an (R_1, S_1)-stable control of $x_1(t)$ with T_2 as domain of disturbance,

Compositional $(R_1 \times R_2, S_1 \times S_2)$-stability

$$\dot{x}_1 = f^1(x_1, x_2)$$
$$\dot{x}_2 = f^2(x_1, x_2)$$

Suppose:

- (H_{1,T_2}): $\dot{x}_1 = f^1(x_1, x_2)$ is robustly OSL w.r.t $x_2 \in T_2$, with λ^1, γ^1.
- (H_{2,T_1}): $\dot{x}_2 = f^2(x_1, x_2)$ is robustly OSL w.r.t $x_1 \in T_1$, with λ^2, γ^2.

Theorem (compositionality): If

- σ_1 is an (R_1, S_1)-stable control of $x_1(t)$ with T_2 as domain of disturbance,
- σ_2 is an (R_2, S_2)-stable control of $x_2(t)$ with T_1 as domain of disturbance,

Compositional $(R_1 \times R_2, S_1 \times S_2)$-stability

$$\dot{x}_1 = f^1(x_1, x_2)$$
$$\dot{x}_2 = f^2(x_1, x_2)$$

Suppose:

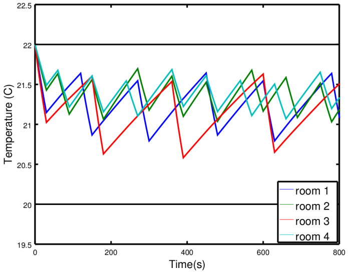
- (H_{1,T_2}): $\dot{x}_1 = f^1(x_1, x_2)$ is robustly OSL w.r.t $x_2 \in T_2$, with λ^1, γ^1.
- (H_{2,T_1}): $\dot{x}_2 = f^2(x_1, x_2)$ is robustly OSL w.r.t $x_1 \in T_1$, with λ^2, γ^2.

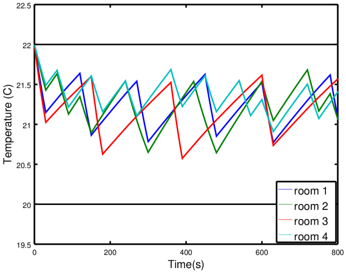
Theorem (compositionality): If

- σ_1 is an (R_1, S_1)-stable control of $x_1(t)$ with T_2 as domain of disturbance,
- σ_2 is an (R_2, S_2)-stable control of $x_2(t)$ with T_1 as domain of disturbance,

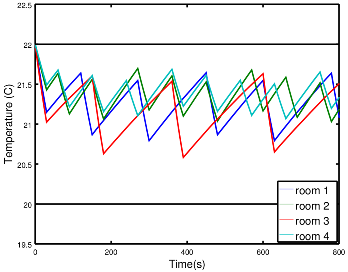
Then: $\sigma = \sigma_1 | \sigma_2$ is an $(R_1 \times R_2, S_1 \times S_2)$-stable control of $x(t) = (x_1(t), x_2(t))$.

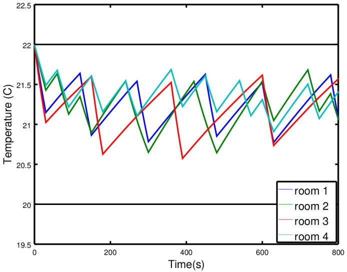
Ex. of centralized vs. distributed Euler-based control





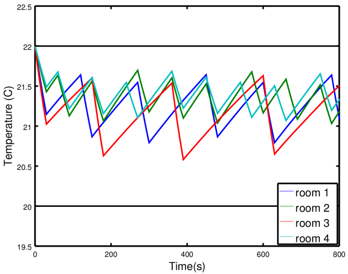
Ex. of centralized vs. distributed Euler-based control

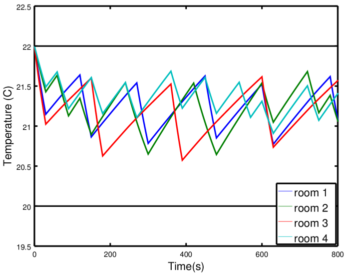




- (left) E.-based centralized control: subsampling $h = \frac{\tau}{20}$, 2^4 modes, 256 balls $\quad\rightarrow\quad$ 48 s. of CPU time.

Ex. of centralized vs. distributed Euler-based control





- (left) E.-based centralized control: subsampling $h = \frac{\tau}{20}$,
  2^4 modes, 256 balls $\rightarrow$ 48 s. of CPU time.

- (right) E.-based distributed control: subsampling $h = \frac{\tau}{10} \mid h = \frac{\tau}{1}$,
  $2^2 \mid 2^2$ modes, $16 \mid 16$ balls $\rightarrow$ < 1 s. of CPU time.

Outline

Final remarks

Final remarks

1. Very simple method

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)
3. Fast, but may lack precision
   w.r.t. sophisticated refinements of interval-based methods

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)
3. Fast, but may lack precision
   w.r.t. sophisticated refinements of interval-based methods
4. Method can be adapted to control reachability (instead of stability)

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)
3. Fast, but may lack precision
   w.r.t. sophisticated refinements of interval-based methods
4. Method can be adapted to control reachability (instead of stability)
5. Replacement of forward Euler's method by better numerical schemes
   (e.g.: backward Euler, Runge-Kutta of order 4)
   does not seem to gain much in the control framework

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)
3. Fast, but may lack precision
   w.r.t. sophisticated refinements of interval-based methods
4. Method can be adapted to control reachability (instead of stability)
5. Replacement of forward Euler's method by better numerical schemes
   (e.g.: backward Euler, Runge-Kutta of order 4)
   does not seem to gain much in the control framework
6. Several examples for which Euler-based control
   - beats state-of-art interval-based control (e.g.: building ventilation)

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)
3. Fast, but may lack precision
   w.r.t. sophisticated refinements of interval-based methods
4. Method can be adapted to control reachability (instead of stability)
5. Replacement of forward Euler's method by better numerical schemes
   (e.g.: backward Euler, Runge-Kutta of order 4)
   does not seem to gain much in the control framework
6. Several examples for which Euler-based control
   - beats state-of-art interval-based control (e.g.: building ventilation)
   - but the converse is also true! (e.g.: DC-DC converter)

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)
3. Fast, but may lack precision
   w.r.t. sophisticated refinements of interval-based methods
4. Method can be adapted to control reachability (instead of stability)
5. Replacement of forward Euler's method by better numerical schemes
   (e.g.: backward Euler, Runge-Kutta of order 4)
   does not seem to gain much in the control framework
6. Several examples for which Euler-based control
   - beats state-of-art interval-based control (e.g.: building ventilation)
   - but the converse is also true! (e.g.: DC-DC converter)
7. Further experimentations ongoing ...

Final remarks

1. Very simple method
2. Very easy to implement (a few hundreds of lines of Octave)
3. Fast, but may lack precision
   w.r.t. sophisticated refinements of interval-based methods
4. Method can be adapted to control reachability (instead of stability)
5. Replacement of forward Euler's method by better numerical schemes
   (e.g.: backward Euler, Runge-Kutta of order 4)
   does not seem to gain much in the control framework
6. Several examples for which Euler-based control
   - beats state-of-art interval-based control (e.g.: building ventilation)
   - but the converse is also true! (e.g.: DC-DC converter)
7. Further experimentations ongoing ...

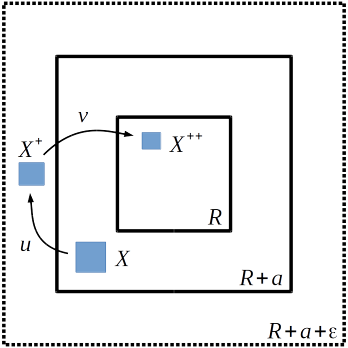
THANKS!

Distributed vs. Centralized Control

Centralized control synthesis

$$\dot{x}(t) = f_u(x(t))$$

Example of a validated pattern of length 2 mapping the "ball" X into R with $S = R + a + \varepsilon$ as safety box:



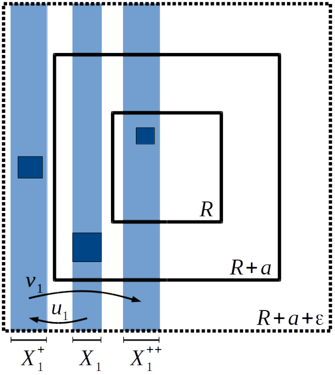
- $X \subset R$
- $X^+ = f_u(X) \subset S$
- $X^{++} = f_v(X^+) \subset R$

- Pattern $u \cdot v$ depends on X

Distrib. Control Synth. (of x_1 using S_2 as approx. of x_2)

$$\dot{x}_1(t) = f^1_{u_1}(x_1(t), x_2(t))$$
$$\dot{x}_2(t) = f^2_{u_2}(x_1(t), x_2(t))$$

Target zone: $R = R_1 \times R_2$



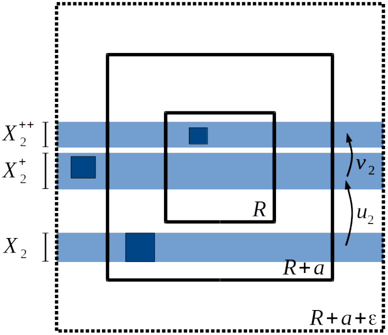
- $X_1 \subset R_1$
- $X_1^+ = f^1_{u_1}(X_1, S_2) \subset S_1$
- $X_1^{++} = f^1_{v_1}(X_1^+, S_2) \subset R_1$

- Pattern $u_1 \cdot v_1$ depends only on X_1

Distrib. Control Synth. (of x_2 using S_1 as approx. of x_1)

$$\dot{x}_1(t) = f^1_{u_1}(x_1(t), x_2(t))$$
$$\dot{x}_2(t) = f^2_{u_2}(x_1(t), x_2(t))$$

Target zone: $R = R_1 \times R_2$



- $X_2 \subset R_2$
- $X_2^+ = f^2_{u_2}(S_1, X_2) \in S_2$
- $X_2^{++} = f^2_{v_2}(S_1, X_2^+) \in R_2$
- Pattern $u_2 \cdot v_2$ depends only on X_2