Guaranteed Control of Switched Control Systems Using Model Order Reduction and Bisection

Adrien Le Coënt [1], Florian De Vuyst [1], Christian Rey [2],
Ludovic Chamoin [2], Laurent Fribourg [3]

April 11, 2015

[1]CMLA Centre de Mathématiques et de Leurs Applications
[2]LMT-Cachan Laboratoire de Mécanique et Technologie
[3]LSV Laboratoire de Spécification et Vérification

Introduction

Framework

- Goal: control the evolution of an operating system with the help of actuators
- Framework of the switched control systems: one selects the working modes of the system over time, every mode is described by differential equations (ODEs or PDEs)
- Application to medium/high dimensional systems:
  - Model Order Reduction
  - Error bounding
  - State space bisection

Outline

Outline

Sampled Switched Systems

A switched system

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

is a family of continuous-time dynamical systems with a rule σ that determines at each time which one is active

Sampled Switched Systems

A switched system

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

is a family of continuous-time dynamical systems with a rule σ that determines at each time which one is active

- state $x \in \mathbb{R}^n$

Sampled Switched Systems

A switched system

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

is a family of continuous-time dynamical systems with a rule σ that determines at each time which one is active

- state $x \in \mathbb{R}^n$
- switching signal $\sigma : [0, \infty) \to U$

Sampled Switched Systems

A switched system

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

is a family of continuous-time dynamical systems with a rule σ that determines at each time which one is active

- state $x \in \mathbb{R}^n$
- switching signal $\sigma : [0, \infty) \to U$
- $U = \{1, \dots, N\}$ finite set of modes associated with dynamics

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

Sampled Switched Systems

A switched system

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

is a family of continuous-time dynamical systems with a rule σ that determines at each time which one is active

- state $x \in \mathbb{R}^n$
- switching signal $\sigma : [0, \infty) \to U$
- $U = \{1, \dots, N\}$ finite set of modes associated with dynamics

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

We focus here on sampled switched systems: switching instants occur periodically every τ

Sampled Switched Systems

A switched system

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

is a family of continuous-time dynamical systems with a rule σ that determines at each time which one is active

- state $x \in \mathbb{R}^n$
- switching signal $\sigma : [0, \infty) \to U$
- $U = \{1, \ldots, N\}$ finite set of modes associated with dynamics

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

We focus here on sampled switched systems: switching instants occur periodically every τ ($\rightsquigarrow \sigma$ is constant on $[i\tau, (i+1)\tau)$)

Affine Sampled Switched Systems

Affine Sampled Switched Systems

- An affine sampled switched system is of the form
  $\{\dot{x} = A_u x + b_u\}_{u \in U}$ with $A_u \in \mathbb{R}^{n \times n}$ and $b_u \in \mathbb{R}$

Affine Sampled Switched Systems

- An affine sampled switched system is of the form
  $\{\dot{x} = A_u x + b_u\}_{u \in U}$ with $A_u \in \mathbb{R}^{n \times n}$ and $b_u \in \mathbb{R}$
- Between two sampling times $i\tau$ and $(i+1)\tau$, the system is governed by a differential affine equation of the form

$$\dot{x}(t) = A_u x(t) + b_u.$$

Affine Sampled Switched Systems

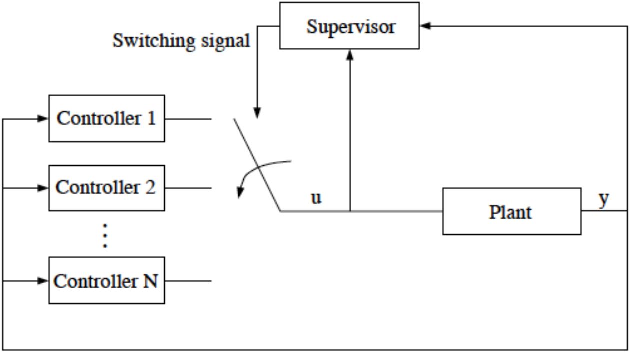
- An affine sampled switched system is of the form
  $\{\dot{x} = A_u x + b_u\}_{u \in U}$ with $A_u \in \mathbb{R}^{n \times n}$ and $b_u \in \mathbb{R}$

- Between two sampling times $i\tau$ and $(i+1)\tau$, the system is governed by a differential affine equation of the form

$$\dot{x}(t) = A_u x(t) + b_u.$$

- Instead of considering all the continuous evolution, one observes the system only at periodic switching instants at times: τ, 2τ, $\ldots$

Controlled Switched Systems: Schematic View



Control Synthesis Problem

We consider the state-dependent control problem of synthesizing σ:

Control Synthesis Problem

We consider the state-dependent control problem of synthesizing σ:

At each τ, find the appropriate switched mode $u \in U$ according to the current value of x, in order to achieve some objectives:

Control Synthesis Problem

We consider the state-dependent control problem of synthesizing σ:

At each τ, find the appropriate switched mode $u \in U$ according to the current value of x, in order to achieve some objectives:

- stability (x should converge to and stay in the neighborhood R of a reference point Ω)

Control Synthesis Problem

We consider the state-dependent control problem of synthesizing σ:

At each τ, find the appropriate switched mode $u \in U$ according to the current value of x, in order to achieve some objectives:

- stability (x should converge to and stay in the neighborhood R of a reference point Ω)
- safety (x should never exit from a safe zone S)

Control Synthesis Problem

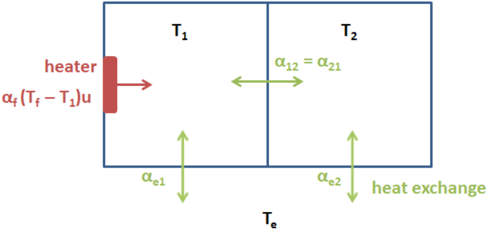
We consider the state-dependent control problem of synthesizing σ:

At each τ, find the appropriate switched mode $u \in U$ according to the current value of x, in order to achieve some objectives:

- stability (x should converge to and stay in the neighborhood R of a reference point Ω)
- safety (x should never exit from a safe zone S)

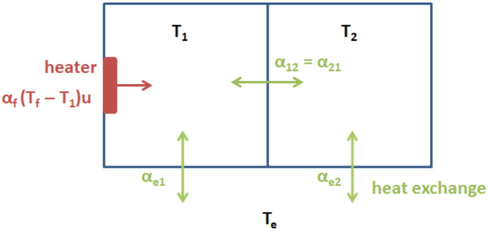
NB: classic stabilization impossible here (no common equilibrium pt)
$\leadsto$ practical stability

Example: Two-room apartment



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

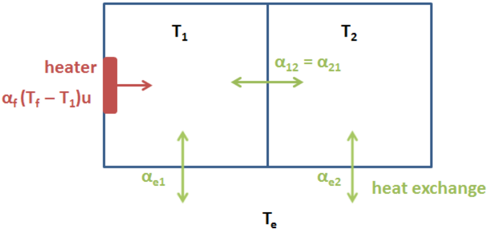
Example: Two-room apartment



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

- Modes: $u = 0, 1$; sampling period τ

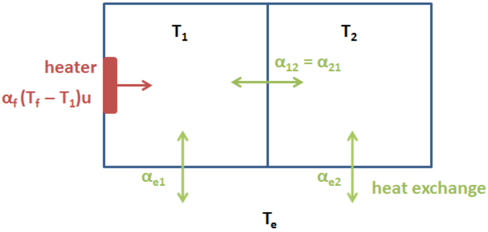
Example: Two-room apartment



$$\begin{pmatrix} \dot{T_1} \\ T_2 \end{pmatrix} = \begin{pmatrix} -\alpha_{21}-\alpha_{e1}-\alpha_f u & \alpha_{21} \\ \alpha_{12} & -\alpha_{12}-\alpha_{e2} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} + \begin{pmatrix} \alpha_{e1}T_e+\alpha_f T_f u \\ \alpha_{e2}T_e \end{pmatrix}.$$

- Modes: $u = 0, 1$; sampling period τ
- A pattern π is a finite sequence of modes (e.g. $(1 \cdot 0 \cdot 0 \cdot 0)$)

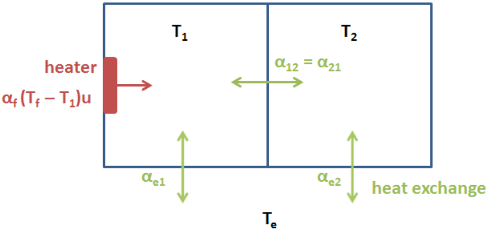
Example: Two-room apartment



$$\begin{pmatrix} \dot{T_1} \\ T_2 \end{pmatrix} = \begin{pmatrix} -\alpha_{21}-\alpha_{e1}-\alpha_f u & \alpha_{21} \\ \alpha_{12} & -\alpha_{12}-\alpha_{e2} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} + \begin{pmatrix} \alpha_{e1}T_e+\alpha_f T_f u \\ \alpha_{e2}T_e \end{pmatrix}.$$

- Modes: $u = 0, 1$; sampling period τ
- A pattern π is a finite sequence of modes (e.g. $(1 \cdot 0 \cdot 0 \cdot 0)$)
- A state dependent control consists to select at each τ a mode (or a pattern) according to the current value of the state.

Example: Two-room apartment



$$\begin{pmatrix} \dot{T_1} \\ T_2 \end{pmatrix} = \begin{pmatrix} -\alpha_{21}-\alpha_{e1}-\alpha_f u & \alpha_{21} \\ \alpha_{12} & -\alpha_{12}-\alpha_{e2} \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} + \begin{pmatrix} \alpha_{e1}T_e+\alpha_f T_f u \\ \alpha_{e2}T_e \end{pmatrix}.$$

- Modes: $u = 0, 1$; sampling period τ
- A pattern π is a finite sequence of modes (e.g. $(1 \cdot 0 \cdot 0 \cdot 0)$)
- A state dependent control consists to select at each τ a mode (or a pattern) according to the current value of the state.

NB: Each mode has its basic proper equilibrium point; by appropriate switching, one can drive the system to a specific stability zone

Safety and Stability Properties for the two-room apartment

Safety and Stability Properties for the two-room apartment

- Example of safety property to be checked: satisfactory temperature

$$\forall t \geq 0 : \qquad T_{min} \leq T_i(t) \leq T_{max}$$

Safety and Stability Properties for the two-room apartment

- Example of safety property to be checked: satisfactory temperature

$$\forall t \geq 0 : \qquad T_{min} \leq T_i(t) \leq T_{max}$$

- Example of stability property to be checked: temperature regulation

$$|T_i(t) - T_{reference}| \leq \varepsilon \text{ as } t \to \infty$$

Affine Sampled Switched Systems (cont'd)

- We introduce the transition relation $\rightarrow^u_\tau$ to denote the point reached at time t under mode u from initial condition x,

Affine Sampled Switched Systems (cont'd)

- We introduce the transition relation $\rightarrow_{\tau}^{u}$ to denote the point reached at time t under mode u from initial condition x, defined by:

$$x \rightarrow_{\tau}^{u} x' \quad \text{iff} \quad x' = C_u x + d_u$$

Affine Sampled Switched Systems (cont'd)

- We introduce the transition relation $\rightarrow^u_\tau$ to denote the point reached at time t under mode u from initial condition x, defined by:

$$x \rightarrow^u_\tau x' \quad \text{iff} \quad x' = C_u x + d_u$$

with $C_u = e^{A_u \tau}$

Affine Sampled Switched Systems (cont'd)

- We introduce the transition relation $\rightarrow_{\tau}^{u}$ to denote the point reached at time t under mode u from initial condition x, defined by:

$$x \rightarrow_{\tau}^{u} x' \quad \text{iff} \quad x' = C_u x + d_u$$

with $C_u = e^{A_u \tau}$
and $d_u = \int_o^\tau e^{A_u(\tau - t)} b_u dt = (e^{A_u}\tau - \mathcal{I})A_u^{-1} b_u$

Affine Sampled Switched Systems (cont'd)

- We introduce the transition relation $\rightarrow_{\tau}^{u}$ to denote the point reached at time t under mode u from initial condition x, defined by:

$$x \rightarrow_{\tau}^{u} x' \quad \text{iff} \quad x' = C_u x + d_u$$

  with $C_u = e^{A_u \tau}$
  and $d_u = \int_o^{\tau} e^{A_u(\tau - t)} b_u dt = (e^{A_u}\tau - \mathcal{I}) A_u^{-1} b_u$

- A sampled switched system can thus be viewed as a piecewise affine discrete-time system.

Post Set Operators

- $Post_u(X) = \{x' \mid x \to_\tau^u x' \text{ for some } x \in X\}$

Post Set Operators

- $Post_u(X) = \{x' \mid x \rightarrow^{u}_{\tau} x' \text{ for some } x \in X\}$
- $Post_\pi(X) = \{x' \mid x \rightarrow^{u_1}_{\tau} \cdots \rightarrow^{u_m}_{\tau} x' \text{ for some } x \in X\}$
  if π is a pattern of the form $(u_1 \cdots u_m)$

Post Set Operators

- $Post_u(X) = \{x' \mid x \to_\tau^u x' \text{ for some } x \in X\}$
- $Post_\pi(X) = \{x' \mid x \to_\tau^{u_1} \cdots \to_\tau^{u_m} x' \text{ for some } x \in X\}$
  if π is a pattern of the form $(u_1 \cdots u_m)$
- The unfolding of $Post_\pi(X)$ is the union of X, $Post_\pi(X)$ and the intermediate sets:

  $X \cup Post_{u_1}(X) \cup Post_{u_1 \cdot u_2}(X) \cup \cdots \cup Post_{u_1 \cdots u_{m-1}}(X) \cup Post_\pi(X)$

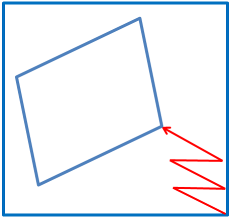
Outline

Bisection Method

Bisection Method

Given a zone R (selected around a reference point Ω)

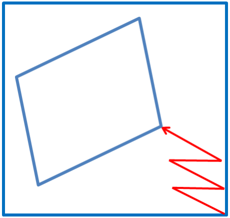
- Look for a *pattern* which maps R into R



Bisection Method

Given a zone R (selected around a reference point Ω)

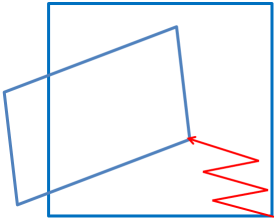
- Look for a *pattern* which maps R into R
- If such a *pattern* exists, then uniform control over the whole R



Bisection Method

Given a zone R (selected around a reference point Ω)

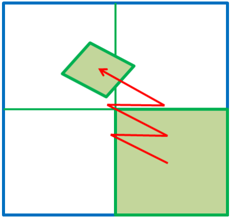
- Look for a *pattern* which maps R into R
- If such a *pattern* exists, then uniform control over the whole R
- Otherwise,



Bisection Method

Given a zone R (selected around a reference point Ω)

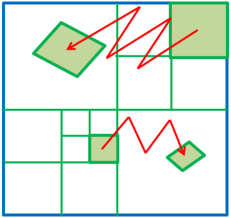
- Look for a *pattern* which maps R into R
- If such a *pattern* exists, then uniform control over the whole R
- Otherwise, bisect of R into subparts, and search for *patterns* mapping these subparts into R



Bisection Method

Given a zone R (selected around a reference point Ω)

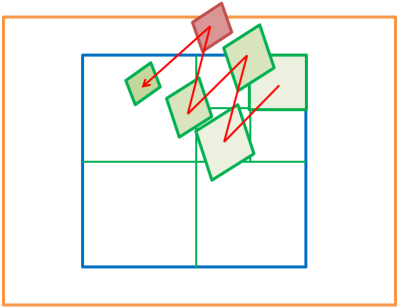
- Look for a *pattern* which maps R into R
- If such a *pattern* exists, then uniform control over the whole R
- Otherwise, bisect of R into subparts, and search for *patterns* mapping these subparts into R
- In case of failure, iterate the bisection

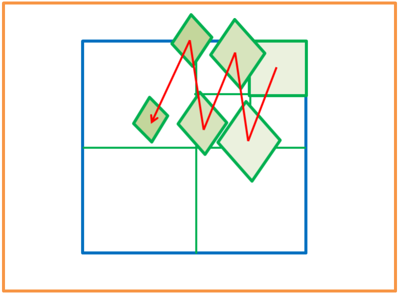


Bisection Method

Given a zone R (selected around a reference point Ω)

- Look for a *pattern* which maps R into R
- If such a *pattern* exists, then uniform control over the whole R
- Otherwise, bisect of R into subparts, and search for *patterns* mapping these subparts into R
- In case of failure, iterate the bisection
- Extension for safety: the unfolding must stay in the safety set S.





definition

A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

definition

A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R$

definition

A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R$
- $\forall i \in I \ Post_{\pi_i}(V_i) \subseteq R$

definition

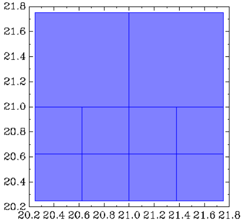
A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R$
- $\forall i \in I \ Post_{\pi_i}(V_i) \subseteq R$
- (Extension for safety: and $\forall i \in I \ Unf_{\pi_i}(V_i) \subseteq S$).

definition

A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

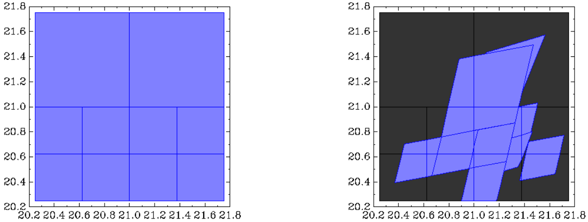
- $\bigcup_{i \in I} V_i = R$
- $\forall i \in I \ Post_{\pi_i}(V_i) \subseteq R$
- (Extension for safety: and $\forall i \in I \ Unf_{\pi_i}(V_i) \subseteq S$).



definition

A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R$
- $\forall i \in I \; Post_{\pi_i}(V_i) \subseteq R$
- (Extension for safety: and $\forall i \in I \; Unf_{\pi_i}(V_i) \subseteq S$).



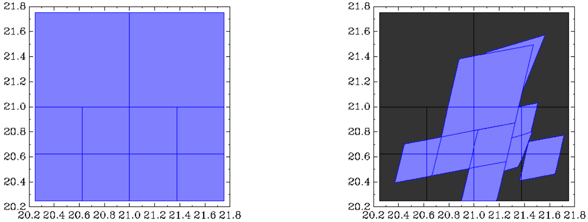
definition and property

Let $Post_{\Delta}(X) =_{def} \bigcup_{i \in I} Post_{\pi_i}(X \cap V_i)$.

definition

A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R$
- $\forall i \in I \ Post_{\pi_i}(V_i) \subseteq R$
- (Extension for safety: and $\forall i \in I \ Unf_{\pi_i}(V_i) \subseteq S$).



definition and property

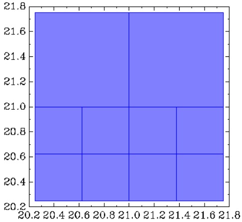
Let $Post_\Delta(X) =_{def} \bigcup_{i \in I} Post_{\pi_i}(X \cap V_i)$. We have:

$$Post_\Delta(R) \subseteq R$$

definition

A decomposition Δ of R is a set of couples $\{(V_i, \pi_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R$
- $\forall i \in I \ Post_{\pi_i}(V_i) \subseteq R$
- (Extension for safety: and $\forall i \in I \ Unf_{\pi_i}(V_i) \subseteq S$).



definition and property

Let $Post_\Delta(X) =_{def} \bigcup_{i \in I} Post_{\pi_i}(X \cap V_i)$. We have:

$$Post_\Delta(R) \subseteq R \quad (\text{and} \quad Unf_\Delta(R) \subseteq S)$$

Control and Trajectories Induced by Δ

The decomposition $\Delta = \{(V_i, \pi_i)\}_{i \in I}$ induces a natural control:

Control and Trajectories Induced by Δ

The decomposition $\Delta = \{(V_i, \pi_i)\}_{i \in I}$ induces a natural control:

1. $x(t) \in R$, therefore $\exists i \in I$ such that $x(t) \in V_i$

Control and Trajectories Induced by Δ

The decomposition $\Delta = \{(V_i, \pi_i)\}_{i \in I}$ induces a natural control:

1. $x(t) \in R$, therefore $\exists i \in I$ such that $x(t) \in V_i$
2. Apply pattern π_i to $x(t)$

Control and Trajectories Induced by Δ

The decomposition $\Delta = \{(V_i, \pi_i)\}_{i \in I}$ induces a natural control:

1. $x(t) \in R$, therefore $\exists i \in I$ such that $x(t) \in V_i$
2. Apply pattern π_i to $x(t)$
3. At the end of π_i, $x(t') \in R$, iterate by going back to step (1)

Control and Trajectories Induced by Δ

The decomposition $\Delta = \{(V_i, \pi_i)\}_{i \in I}$ induces a natural control:

1. $x(t) \in R$, therefore $\exists i \in I$ such that $x(t) \in V_i$
2. Apply pattern π_i to $x(t)$
3. At the end of π_i, $x(t') \in R$, iterate by going back to step (1)

Property

Under the Δ-control,

- any trajectory $x_0 \rightarrow_{\pi_{i_1}} x_1 \rightarrow_{\pi_{i_2}} x_2 \rightarrow_{\pi_{i_3}} \cdots$ always stays in R

Control and Trajectories Induced by Δ

The decomposition $\Delta = \{(V_i, \pi_i)\}_{i \in I}$ induces a natural control:

1. $x(t) \in R$, therefore $\exists i \in I$ such that $x(t) \in V_i$
2. Apply pattern π_i to $x(t)$
3. At the end of π_i, $x(t') \in R$, iterate by going back to step (1)

Property

Under the Δ-control,

- any trajectory $x_0 \rightarrow_{\pi_{i_1}} x_1 \rightarrow_{\pi_{i_2}} x_2 \rightarrow_{\pi_{i_3}} \cdots$ always stays in R
- The unfolding of the trajectory always stays in S

Decomposition for the two-room apartment

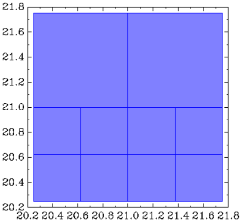
For: $\alpha_{12} = 5 \times 10^{-2}, \alpha_{21} = 5 \times 10^{-2}, \alpha_{e1} = 5 \times 10^{-3}, \alpha_{e2} = 3.3 \times 10^{-3}, \alpha_f = 8.3 \times 10^{-3}, T_e = 10, T_f = 50$ and $\tau = 5$.

Decomposition for the two-room apartment

For: $\alpha_{12} = 5 \times 10^{-2}, \alpha_{21} = 5 \times 10^{-2}, \alpha_{e1} = 5 \times 10^{-3}, \alpha_{e2} = 3.3 \times 10^{-3}, \alpha_f = 8.3 \times 10^{-3}, T_e = 10, T_f = 50$ and $\tau = 5$.
$\Omega = (21, 21)$, $R = [20.25, 21.75] \times [20.25, 21.75]$, $S = [20, 22] \times [20, 22]$

Decomposition for the two-room apartment

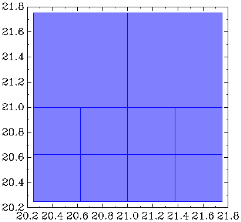
For: $\alpha_{12} = 5 \times 10^{-2}, \alpha_{21} = 5 \times 10^{-2}, \alpha_{e1} = 5 \times 10^{-3}, \alpha_{e2} = 3.3 \times 10^{-3}, \alpha_f = 8.3 \times 10^{-3}, T_e = 10, T_f = 50$ and $\tau = 5$.
$\Omega = (21, 21)$, $R = [20.25, 21.75] \times [20.25, 21.75]$, $S = [20, 22] \times [20, 22]$

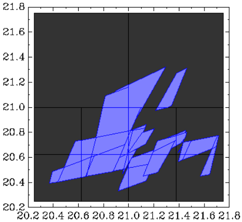


Decomposition for the two-room apartment

For: $\alpha_{12} = 5 \times 10^{-2}, \alpha_{21} = 5 \times 10^{-2}, \alpha_{e1} = 5 \times 10^{-3}, \alpha_{e2} = 3.3 \times 10^{-3}, \alpha_f = 8.3 \times 10^{-3}, T_e = 10, T_f = 50$ and $\tau = 5$.

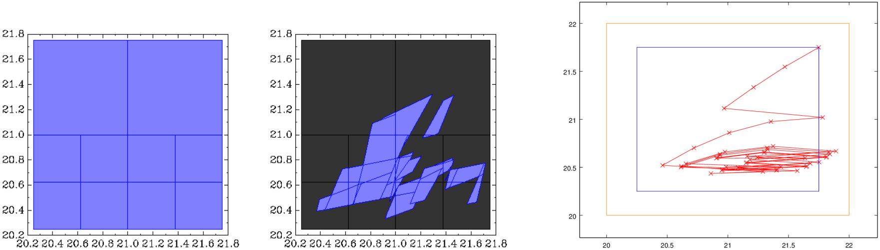
$\Omega = (21, 21)$, $R = [20.25, 21.75] \times [20.25, 21.75]$, $S = [20, 22] \times [20, 22]$





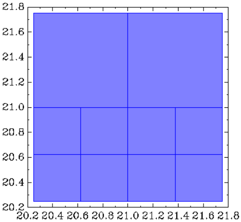
Decomposition for the two-room apartment

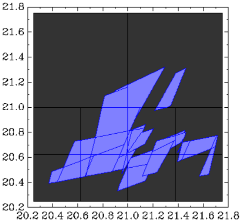
For: $\alpha_{12} = 5 \times 10^{-2}, \alpha_{21} = 5 \times 10^{-2}, \alpha_{e1} = 5 \times 10^{-3}, \alpha_{e2} = 3.3 \times 10^{-3}, \alpha_f = 8.3 \times 10^{-3}, T_e = 10, T_f = 50$ and $\tau = 5$.
$\Omega = (21, 21)$, $R = [20.25, 21.75] \times [20.25, 21.75]$, $S = [20, 22] \times [20, 22]$



Decomposition for the two-room apartment

For: $\alpha_{12} = 5 \times 10^{-2}, \alpha_{21} = 5 \times 10^{-2}, \alpha_{e1} = 5 \times 10^{-3}, \alpha_{e2} = 3.3 \times 10^{-3}, \alpha_f = 8.3 \times 10^{-3}, T_e = 10, T_f = 50$ and $\tau = 5$.
$\Omega = (21, 21)$, $R = [20.25, 21.75] \times [20.25, 21.75]$, $S = [20, 22] \times [20, 22]$





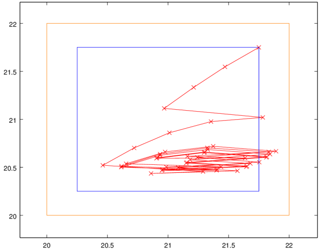
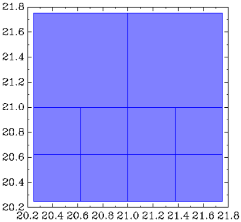
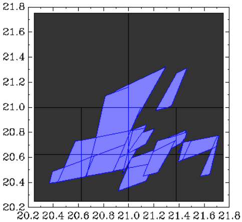


Figure : Decomposition (left) ; unfolding (middle) ; unfolded trajectory (right) in plane (T_1, T_2)

Decomposition for the two-room apartment

For: $\alpha_{12} = 5 \times 10^{-2}, \alpha_{21} = 5 \times 10^{-2}, \alpha_{e1} = 5 \times 10^{-3}, \alpha_{e2} = 3.3 \times 10^{-3}, \alpha_f = 8.3 \times 10^{-3}, T_e = 10, T_f = 50$ and $\tau = 5$.
$\Omega = (21, 21)$, $R = [20.25, 21.75] \times [20.25, 21.75]$, $S = [20, 22] \times [20, 22]$





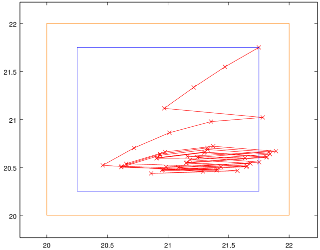


Figure : Decomposition (left) ; unfolding (middle) ; unfolded trajectory (right) in plane (T_1, T_2)

Decomposition found for $k = 4, d = 3$.

Outline

A Sampled Switched System with Output

- Described by the differential equation:

$$\begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Constraint: x of "high" dimension.

A Sampled Switched System with Output

- Described by the differential equation:

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{cases}$$

  - $x \in \mathbb{R}^n$: state variable
  - $y \in \mathbb{R}^m$: output
  - $u \in \mathbb{R}^p$: control input, takes a finite number of values (modes)
  - A,B,C: matrices of appropriate dimensions

- Constraint: x of “high” dimension.

A Sampled Switched System with Output

- Described by the differential equation:

$$\begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

  - $x \in \mathbb{R}^n$: state variable
  - $y \in \mathbb{R}^m$: output
  - $u \in \mathbb{R}^p$: control input, takes a finite number of values (modes)
  - A,B,C: matrices of appropriate dimensions
- Idea: impose the right $u(t)$ such that x and y verify some properties (stability, reachability...)

- Constraint: x of “high” dimension.

A Sampled Switched System with Output

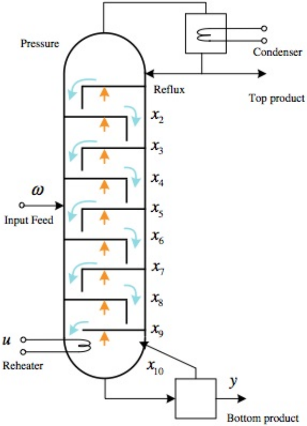
- Described by the differential equation:

$$\begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

  - $x \in \mathbb{R}^n$: state variable
  - $y \in \mathbb{R}^m$: output
  - $u \in \mathbb{R}^p$: control input, takes a finite number of values (modes)
  - A,B,C: matrices of appropriate dimensions
- Idea: impose the right $u(t)$ such that x and y verify some properties (stability, reachability...)
- Objectives:
  1. *x-stabilization*: make all the state trajectories starting in a compact interest set $R_x \subset \mathbb{R}^n$ return to R_x;
  2. *y-convergence*: send the output of all the trajectories starting in R_x into an objective set $R_y \subset \mathbb{R}^m$;
- Constraint: x of “high” dimension.

A Sampled Switched System with Output

A distillation column



Output Post Set Operators

- $Post_{u,C}(X) = \{y = Cx' \mid x \rightarrow_{\tau}^{u} x' \text{ for some } x \in X\}$

Output Post Set Operators

- $Post_{u,C}(X) = \{y = Cx' \mid x \rightarrow_{\tau}^{u} x' \text{ for some } x \in X\}$
- $Post_{Pat,C}(X) = \{y = Cx' \mid x \rightarrow_{\tau}^{u_1} \cdots \rightarrow_{\tau}^{u_m} x' \text{ for some } x \in X\}$
  if Pat is a pattern of the form $(u_1 \cdots u_m)$

New Decomposition

definition

A decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i \in I}$ such that:

New Decomposition

definition

A decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i\in I}$ such that:

- $\bigcup_{i\in I} V_i = R_x$

New Decomposition

definition

A decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R_x$
- $\forall i \in I \ Post_{Pat_i}(V_i) \subseteq R_x$ (x-stabilization)

New Decomposition

definition

A decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i\in I}$ such that:

- $\bigcup_{i\in I} V_i = R_x$
- $\forall i \in I \; Post_{Pat_i}(V_i) \subseteq R_x$ (x-stabilization)
- $\forall i \in I \; Post_{Pat_i,C}(V_i) \subseteq R_y$ (y-convergence)

New Decomposition

definition

A decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R_x$
- $\forall i \in I \; Post_{Pat_i}(V_i) \subseteq R_x$ (x-stabilization)
- $\forall i \in I \; Post_{Pat_i, C}(V_i) \subseteq R_y$ (y-convergence)

definition and property

Let $Post_{\Delta}(X) =_{def} \bigcup_{i \in I} Post_{\pi_i}(X \cap V_i)$.

New Decomposition

definition

A decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R_x$
- $\forall i \in I \; Post_{Pat_i}(V_i) \subseteq R_x$ (x-stabilization)
- $\forall i \in I \; Post_{Pat_i, C}(V_i) \subseteq R_y$ (y-convergence)

definition and property

Let $Post_{\Delta}(X) =_{def} \bigcup_{i \in I} Post_{\pi_i}(X \cap V_i)$. We have:

$$Post_{\Delta}(R_x) \subseteq R_x \quad \text{and} \quad Post_{\Delta, C}(R_x) \subseteq R_y.$$

New Decomposition

definition

A decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i \in I}$ such that:

- $\bigcup_{i \in I} V_i = R_x$
- $\forall i \in I \ Post_{Pat_i}(V_i) \subseteq R_x$ (x-stabilization)
- $\forall i \in I \ Post_{Pat_i,C}(V_i) \subseteq R_y$ (y-convergence)

definition and property

Let $Post_\Delta(X) =_{def} \bigcup_{i \in I} Post_{\pi_i}(X \cap V_i)$. We have:

$$Post_\Delta(R_x) \subseteq R_x \quad \text{and} \quad Post_{\Delta,C}(R_x) \subseteq R_y.$$

Computational cost of decomposition: at most in $O(2^{nd} N^k)$.

Outline

Model Order Reduction by Projection

Construction of a reduced order system $\hat{\Sigma}$ of order n_r:

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Model Order Reduction by Projection

Construction of a reduced order system $\hat{\Sigma}$ of order n_r:

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Reduction by a projection (constructed by balanced truncation) $\pi = \pi_L \pi_R$, $\pi_L \in \mathbb{R}^{n \times n_r}$, $\pi_R \in \mathbb{R}^{n_r \times n}$:

$$\hat{A} = \pi_R A \pi_L, \quad \hat{B} = \pi_R B, \quad \hat{C} = C \pi_L.$$

Model Order Reduction by Projection

Construction of a reduced order system $\hat{\Sigma}$ of order n_r:

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) &= \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) &= \hat{C}\hat{x}(t). \end{cases}$$

Reduction by a projection (constructed by balanced truncation) $\pi = \pi_L \pi_R$, $\pi_L \in \mathbb{R}^{n \times n_r}$, $\pi_R \in \mathbb{R}^{n_r \times n}$:

$$\hat{A} = \pi_R A \pi_L, \quad \hat{B} = \pi_R B, \quad \hat{C} = C \pi_L.$$

Goal: design a controle rule $u(\cdot)$ at the low-order level and apply it at the full-order level.

Model Order Reduction by Projection

Construction of a reduced order system $\hat{\Sigma}$ of order n_r:

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Reduction by a projection (constructed by balanced truncation)
$\pi = \pi_L \pi_R$, $\pi_L \in \mathbb{R}^{n \times n_r}$, $\pi_R \in \mathbb{R}^{n_r \times n}$:

$$\hat{A} = \pi_R A \pi_L, \quad \hat{B} = \pi_R B, \quad \hat{C} = C\pi_L.$$

Goal: design a controle rule $u(\cdot)$ at the low-order level and apply it at the full-order level.
Requirements:

- projection of the interest set $\hat{R}_x = \pi_R R_x$

Model Order Reduction by Projection

Construction of a reduced order system $\hat{\Sigma}$ of order n_r:

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Reduction by a projection (constructed by balanced truncation) $\pi = \pi_L \pi_R$, $\pi_L \in \mathbb{R}^{n \times n_r}$, $\pi_R \in \mathbb{R}^{n_r \times n}$:

$$\hat{A} = \pi_R A \pi_L, \quad \hat{B} = \pi_R B, \quad \hat{C} = C \pi_L.$$

Goal: design a controle rule $u(\cdot)$ at the low-order level and apply it at the full-order level.

Requirements:

- projection of the interest set $\hat{R}_x = \pi_R R_x$
- error bounding of the state and output trajectory

Output trajectory error [4]

Defined by (for a pattern Pat):

$$e_y(|Pat|\tau) = \|CPost_{Pat}(x) - \hat{C}Post_{Pat}(\pi_R x)\|$$

Output trajectory error [4]

Defined by (for a pattern Pat):

$$e_y(|Pat|\tau) = \|CPost_{Pat}(x) - \hat{C}Post_{Pat}(\pi_R x)\|$$

For a pattern of length j, bounded by ε_y^j:

$$\forall t = j\tau > 0, \quad e_y(j\tau) \leq \varepsilon_y^j$$

Output trajectory error [4]

Defined by (for a pattern Pat):

$$e_y(|Pat|\tau) = \|CPost_{Pat}(x) - \hat{C}Post_{Pat}(\pi_R x)\|$$

For a pattern of length j, bounded by ε_y^j:

$$\forall t = j\tau > 0, \quad e_y(j\tau) \leq \varepsilon_y^j$$

where:

$$\varepsilon_y^j = \|u(\cdot)\|_\infty^{[0,j\tau]} \int_0^{j\tau} \| \begin{bmatrix} C & -\hat{C} \end{bmatrix} \begin{bmatrix} e^{tA} & \\ & e^{t\hat{A}} \end{bmatrix} \begin{bmatrix} B \\ \hat{B} \end{bmatrix} \| dt \quad + \sup_{x_0 \in R_x} \| \begin{bmatrix} C & -\hat{C} \end{bmatrix} \begin{bmatrix} e^{j\tau A} & \\ & e^{j\tau\hat{A}} \end{bmatrix} \begin{bmatrix} x_0 \\ \pi_R x_0 \end{bmatrix} \|.$$

State trajectory error

Defined by (for a pattern Pat):

$$e_x(|Pat|\tau) = \|\pi_R Post_{Pat}(x) - Post_{Pat}(\pi_R x)\|$$

State trajectory error

Defined by (for a pattern Pat):

$$e_x(|Pat|\tau) = \|\pi_R Post_{Pat}(x) - Post_{Pat}(\pi_R x)\|$$

For a pattern of length j, bounded by ε_x^j:

$$\forall t = j\tau > 0, \quad e_x(j\tau) \leq \varepsilon_x^j$$

State trajectory error

Defined by (for a pattern Pat):

$$e_x(|Pat|\tau) = \|\pi_R Post_{Pat}(x) - Post_{Pat}(\pi_R x)\|$$

For a pattern of length j, bounded by ε_x^j:

$$\forall t = j\tau > 0, \quad e_x(j\tau) \leq \varepsilon_x^j$$

where:

$$\varepsilon_x^j = \|u(\cdot)\|_\infty^{[0,j\tau]} \int_0^{j\tau} \| \begin{bmatrix} \pi_R & -I_{n_r} \end{bmatrix} \begin{bmatrix} e^{tA} & \\ & e^{t\hat{A}} \end{bmatrix} \begin{bmatrix} B \\ \hat{B} \end{bmatrix} \| dt \quad + \sup_{x_0 \in R_x} \| \begin{bmatrix} \pi_R & -I_{n_r} \end{bmatrix} \begin{bmatrix} e^{j\tau A} & \\ & e^{j\tau \hat{A}} \end{bmatrix} \begin{bmatrix} x_0 \\ \pi_R x_0 \end{bmatrix} \|.$$

Outline

Reduced Order Control

Two systems:

- Full-order system: Σ, R_x, R_y

$$\Sigma : \begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Reduced-order system: $\hat{\Sigma}$, $\hat{R}_x$, R_y

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Reduced Order Control

Two systems:

- Full-order system: Σ, R_x, R_y

$$\Sigma : \begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Reduced-order system: $\hat{\Sigma}$, $\hat{R}_x$, R_y

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Control synthesis (decomposition) for the reduced-order system.

Reduced Order Control

Two systems:

- Full-order system: Σ, R_x, R_y

$$\Sigma : \begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Reduced-order system: $\hat{\Sigma}$, $\hat{R}_x$, R_y

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Control synthesis (decomposition) for the reduced-order system.
$\Rightarrow$ reduced-order control

Reduced Order Control

Two systems:

- Full-order system: Σ, R_x, R_y

$$\Sigma : \begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Reduced-order system: $\hat{\Sigma}$, $\hat{R}_x$, R_y

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Control synthesis (decomposition) for the reduced-order system.

$\Rightarrow$ reduced-order control

$\Rightarrow$ application of the reduced-order control to the full-order system

Reduced Order Control

Two systems:

- Full-order system: Σ, R_x, R_y

$$\Sigma : \begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Reduced-order system: $\hat{\Sigma}$, $\hat{R}_x$, R_y

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Control synthesis (decomposition) for the reduced-order system.
$\Rightarrow$ reduced-order control
$\Rightarrow$ application of the reduced-order control to the full-order system
Questions:

Reduced Order Control

Two systems:

- Full-order system: Σ, R_x, R_y

$$\Sigma : \begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Reduced-order system: $\hat{\Sigma}$, $\hat{R}_x$, R_y

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Control synthesis (decomposition) for the reduced-order system.

$\Rightarrow$ reduced-order control

$\Rightarrow$ application of the reduced-order control to the full-order system

Questions:

- How is it applied?

Reduced Order Control

Two systems:

- Full-order system: Σ, R_x, R_y

$$\Sigma : \begin{cases} \dot{x}(t) & = Ax(t) + Bu(t) \\ y(t) & = Cx(t) \end{cases}$$

- Reduced-order system: $\hat{\Sigma}$, $\hat{R}_x$, R_y

$$\hat{\Sigma} : \begin{cases} \dot{\hat{x}}(t) & = \hat{A}\hat{x}(t) + \hat{B}u(t), \\ y_r(t) & = \hat{C}\hat{x}(t). \end{cases}$$

Control synthesis (decomposition) for the reduced-order system.
$\Rightarrow$ reduced-order control
$\Rightarrow$ application of the reduced-order control to the full-order system
Questions:

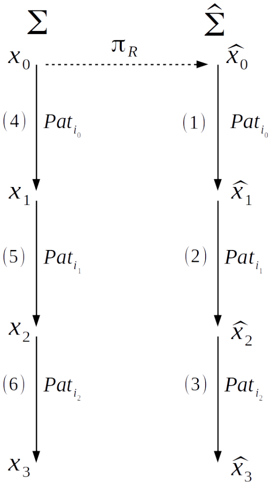
- How is it applied?
- Is the reduced-order control effective at the full-order level?

Outline

Offline Procedure

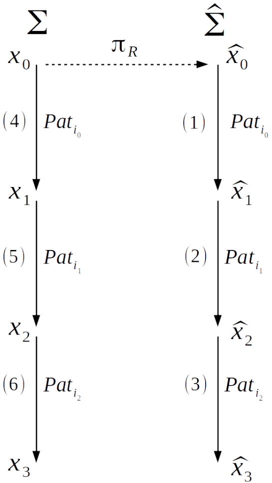
$$
\begin{tikzcd}[row sep=large, column sep=huge]
\Sigma & \widehat{\Sigma} \\
x_0 \arrow[r, dashed, "\pi_R"] \arrow[d, "(4)"', "Pat_{i_0}"] & \widehat{x}_0 \arrow[d, "(1)"', "Pat_{i_0}"] \\
x_1 \arrow[d, "(5)"', "Pat_{i_1}"] & \widehat{x}_1 \arrow[d, "(2)"', "Pat_{i_1}"] \\
x_2 \arrow[d, "(6)"', "Pat_{i_2}"] & \widehat{x}_2 \arrow[d, "(3)"', "Pat_{i_2}"] \\
x_3 & \widehat{x}_3
\end{tikzcd}
$$

Offline Procedure



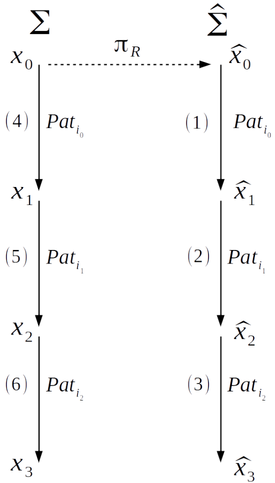
1. Projection of the initial state x_0

Offline Procedure



1. Projection of the initial state x_0
2. Computation of a pattern sequence at the low-order level Pat_{i_0}, Pat_{i_1}... (steps (1),(2),(3))

Offline Procedure



1. Projection of the initial state x_0
2. Computation of a pattern sequence at the low-order level Pat_{i_0}, Pat_{i_1}... (steps (1),(2),(3))
3. Application of the pattern sequence at the full-order level (steps (4),(5),(6)).

Guaranteed Offline Control

Application of the same pattern sequence:

Guaranteed Offline Control

Application of the same pattern sequence:

$$\Rightarrow \quad \forall t = j\tau > 0, \quad \|y(t) - y_r(t)\| \leq \varepsilon_y^j$$

Guaranteed Offline Control

Application of the same pattern sequence:

$$\Rightarrow \quad \forall t = j\tau > 0, \quad \|y(t) - y_r(t)\| \leq \varepsilon_y^j$$

$$\Rightarrow \quad \forall t = j\tau > 0, \quad \|y(t) - y_r(t)\| \leq \varepsilon_y^\infty = \sup_{j>0} \varepsilon_y^j$$

Guaranteed Offline Control

Application of the same pattern sequence:

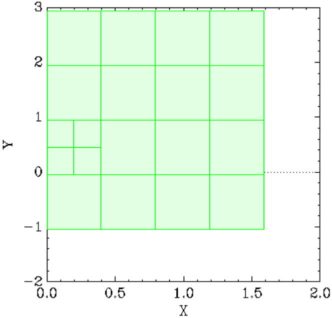
$$\Rightarrow \quad \forall t = j\tau > 0, \quad \|y(t) - y_r(t)\| \leq \varepsilon_y^j$$

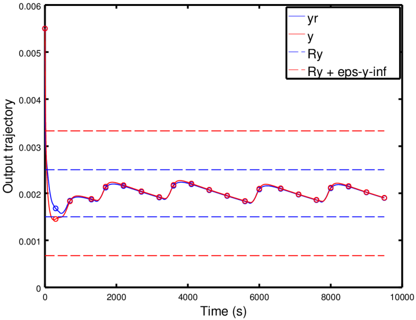
$$\Rightarrow \quad \forall t = j\tau > 0, \quad \|y(t) - y_r(t)\| \leq \varepsilon_y^\infty = \sup_{j>0} \varepsilon_y^j$$

Consequence: the output of the full order system is sent in $R_y + \varepsilon_y^\infty$.

Guaranteed Offline Control

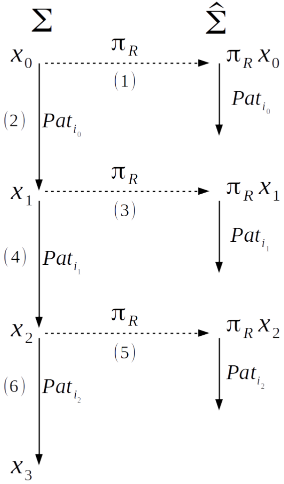
Simulation on a linearized model of a distillation column [5]: $n = 11$ and $n_r = 2$:



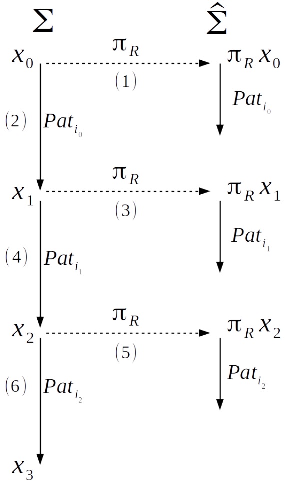


Outline

Online Procedure

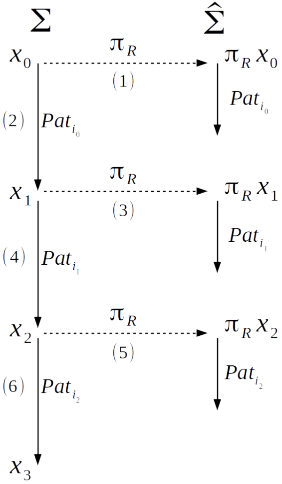


Online Procedure



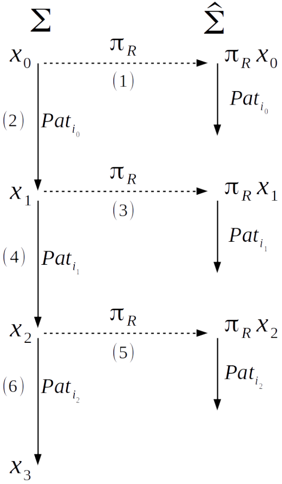
1. Projection of the initial state x_0 (step (1))

Online Procedure



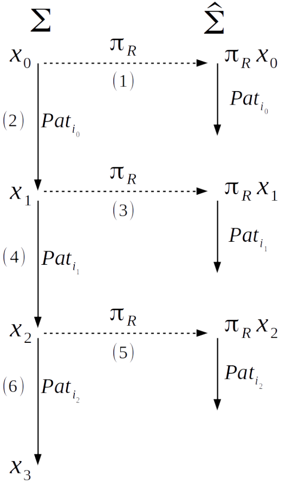
1. Projection of the initial state x_0 (step (1))
2. Computation of the pattern Pat_{i_0} at the reduced-order level

Online Procedure



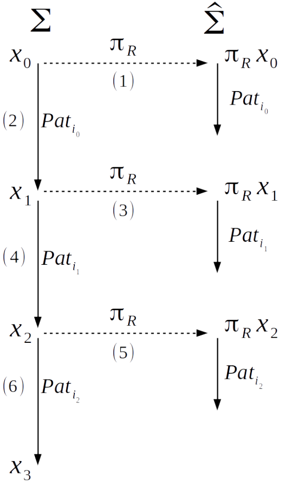
1. Projection of the initial state x_0 (step (1))
2. Computation of the pattern Pat_{i_0} at the reduced-order level
3. Application of the pattern Pat_{i_0} at the full-order level, Σ is sent to a state x_1 (step (2))

Online Procedure



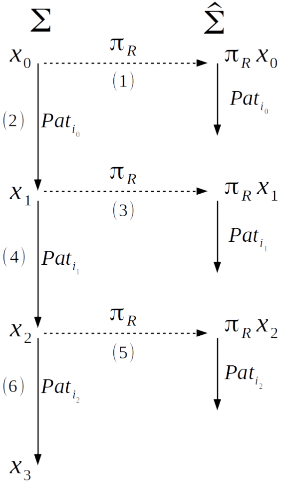
1. Projection of the initial state x_0 (step (1))
2. Computation of the pattern Pat_{i_0} at the reduced-order level
3. Application of the pattern Pat_{i_0} at the full-order level, Σ is sent to a state x_1 (step (2))
4. Projection of the (new initial) state x_1 (step (3))

Online Procedure



1. Projection of the initial state x_0 (step (1))
2. Computation of the pattern Pat_{i_0} at the reduced-order level
3. Application of the pattern Pat_{i_0} at the full-order level, Σ is sent to a state x_1 (step (2))
4. Projection of the (new initial) state x_1 (step (3))
5. Computation of the pattern Pat_{i_1} at the reduced-order level

Online Procedure



1. Projection of the initial state x_0 (step (1))
2. Computation of the pattern Pat_{i_0} at the reduced-order level
3. Application of the pattern Pat_{i_0} at the full-order level, Σ is sent to a state x_1 (step (2))
4. Projection of the (new initial) state x_1 (step (3))
5. Computation of the pattern Pat_{i_1} at the reduced-order level
6. Application of the pattern Pat_{i_1} at the full-order level, Σ is sent to a state x_2 (step (4))...

Guaranteed Online Control

Requirement to apply the online procedure:

Guaranteed Online Control

Requirement to apply the online procedure:

- Ensure that $\pi_R Post_{Pat_i}(x) \in \hat{R}_x$ at every step.

Guaranteed Online Control

Requirement to apply the online procedure:

- Ensure that $\pi_R Post_{Pat_i}(x) \in \hat{R}_x$ at every step.

Solution: Compute an ε-decomposition

definition

A ε-decomposition Δ of R_x is a set of couples $\{(V_i, Pat_i)\}_{i\in I}$ such that:

- $\bigcup_{i\in I} V_i = R_x$
- $\forall i \in I\ Post_{Pat_i}(V_i) \subseteq R_x - \varepsilon_x^{|Pat_i|}$
- $\forall i \in I\ Post_{Pat_i,C}(V_i) \subseteq R_y$ (y-convergence)

Guaranteed Online Control

An ε-decomposition performed on $\hat{\Sigma}$ permits to iterate the online procedure:

Guaranteed Online Control

An ε-decomposition performed on $\hat{\Sigma}$ permits to iterate the online procedure:

- At a step k, $\pi_R x_k$ is sent in $R_x - \varepsilon_x^{|Pat_{i_k}|}$

Guaranteed Online Control

An ε-decomposition performed on $\hat{\Sigma}$ permits to iterate the online procedure:

- At a step k, $\pi_R x_k$ is sent in $R_x - \varepsilon_x^{|Pat_{i_k}|}$
- we have:

$$\|\pi_R Post_{Pat}(x) - Post_{Pat}(\pi_R x)\| \leq \varepsilon_x^{|Pat_{i_k}|}$$

Guaranteed Online Control

An ε-decomposition performed on $\hat{\Sigma}$ permits to iterate the online procedure:

- At a step k, $\pi_R x_k$ is sent in $R_x - \varepsilon_x^{|Pat_{i_k}|}$
- we have:

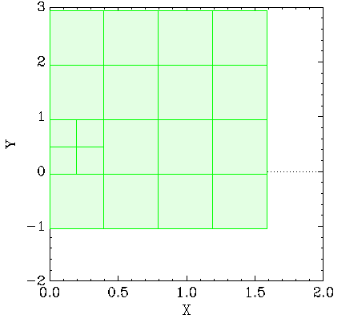
$$\|\pi_R Post_{Pat}(x) - Post_{Pat}(\pi_R x)\| \leq \varepsilon_x^{|Pat_{i_k}|}$$

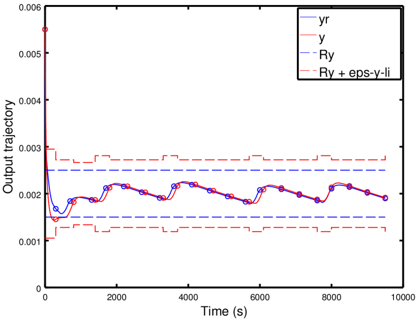
- thus, at every step k:

$$\pi_R Post_{Pat_{i_k}}(x_k) \in \hat{R}_x$$

Guaranteed Online Control

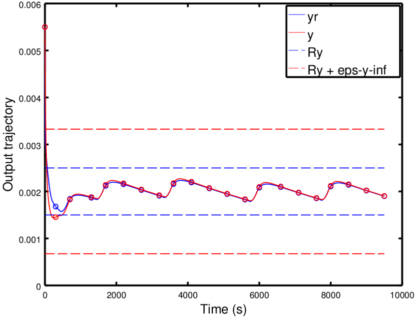
Simulation on a linearized model of a distillation column [5]: $n = 11$ and $n_r = 2$:

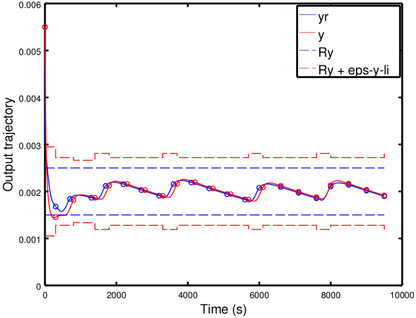




Remark: Output trajectory error depending on the length of the applied pattern: much lower than the infinite bound ε_y^∞

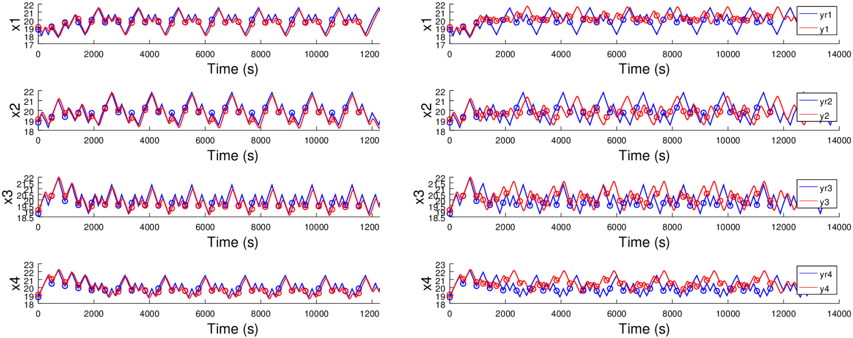
Comparison of the Two Procedures





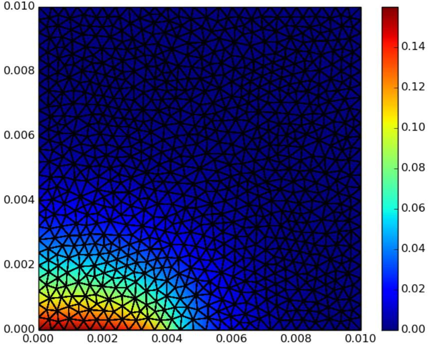
Other Applications

- Control of the temperature of a 4 room appartment: offline and online control
  $n = 8$ and $n_r = 4$



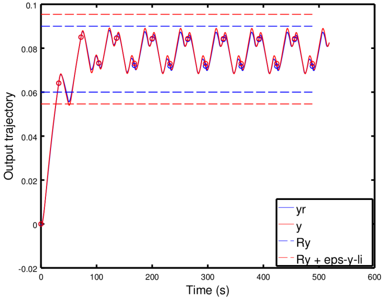
Other Applications

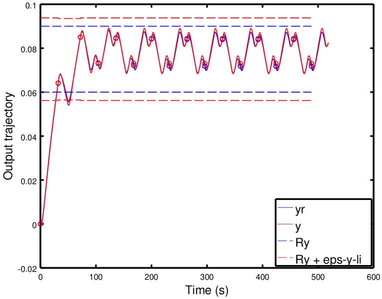
- Control of the temperature of a square plate discretized by finite elements: offline and online control
  $n = 897$ and $n_r = 2$



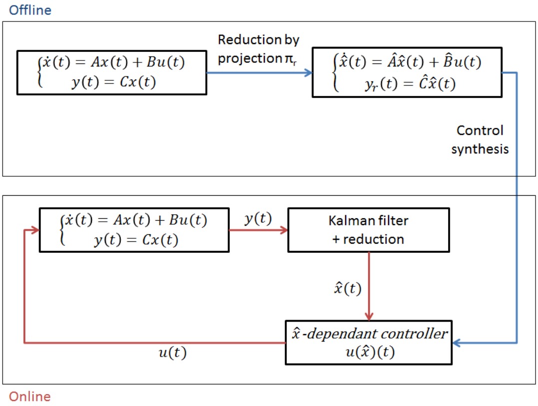
Other Applications

- Control of the temperature of a square plate discretized by finite elements: offline and online control
  $n = 897$ and $n_r = 2$

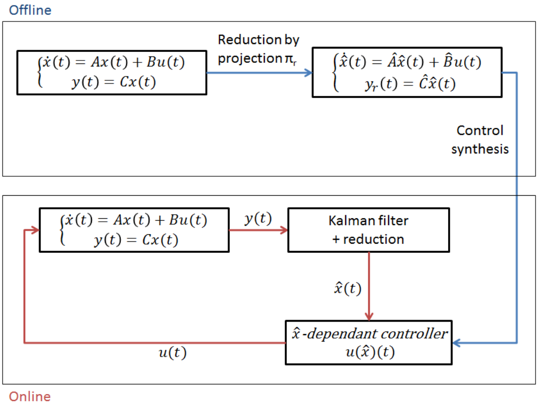




Open Questions and Future Work

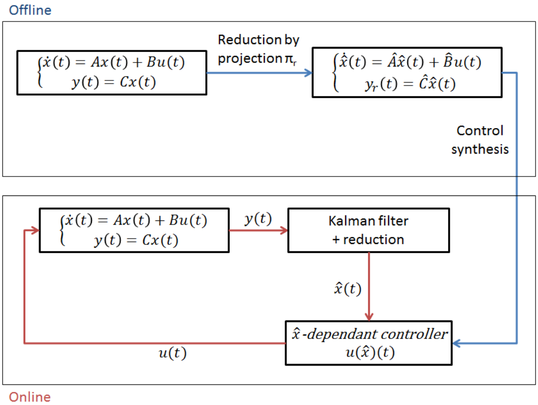


Open Questions and Future Work



- Online reconstruction of the reduced state
  - $\Rightarrow$ reduced Kalman filter
  - $\Rightarrow$ reconstruction error estimation

Open Questions and Future Work



- Online reconstruction of the reduced state
  - $\Rightarrow$ reduced Kalman filter
  - $\Rightarrow$ reconstruction error estimation
- Application to large scale systems

Some References

Peter Benner and André Schneider.
Balanced truncation model order reduction for lti systems with many inputs or outputs.
In *Proceedings of the 19th international symposium on mathematical theory of networks and systems–MTNS*, volume 5, 2010.

Laurent Fribourg, Ulrich Kühne, and Romain Soulat.
Minimator: a tool for controller synthesis and computation of minimal invariant sets for linear switched systems, March 2013.

Laurent Fribourg and Romain Soulat.
Control of Switching Systems by Invariance Analysis.
Focus series. John Wiley and Sons, ISTE Ltd, 2013.

Zhi Han and Bruce Krogh.
Reachability analysis of hybrid systems using reduced-order models.
In *American Control Conference*, pages 1183–1189. IEEE, 2004.

D. Tong, W. Zhou, A. Dai, H. Wang, X. Mou, and Y. Xu.
$\mathcal{H}_\infty$ model reduction for the distillation column linear system.
Circuits Syst Signal Process, 2014.

Some References

Peter Benner and André Schneider.
Balanced truncation model order reduction for lti systems with many inputs or outputs.
In *Proceedings of the 19th international symposium on mathematical theory of networks and systems–MTNS*, volume 5, 2010.

Laurent Fribourg, Ulrich Kühne, and Romain Soulat.
Minimator: a tool for controller synthesis and computation of minimal invariant sets for linear switched systems, March 2013.

Laurent Fribourg and Romain Soulat.
Control of Switching Systems by Invariance Analysis.
Focus series. John Wiley and Sons, ISTE Ltd, 2013.

Zhi Han and Bruce Krogh.
Reachability analysis of hybrid systems using reduced-order models.
In *American Control Conference*, pages 1183–1189. IEEE, 2004.

D. Tong, W. Zhou, A. Dai, H. Wang, X. Mou, and Y. Xu.
$\mathcal{H}_\infty$ model reduction for the distillation column linear system.
Circuits Syst Signal Process, 2014.

Thank you ! Questions?