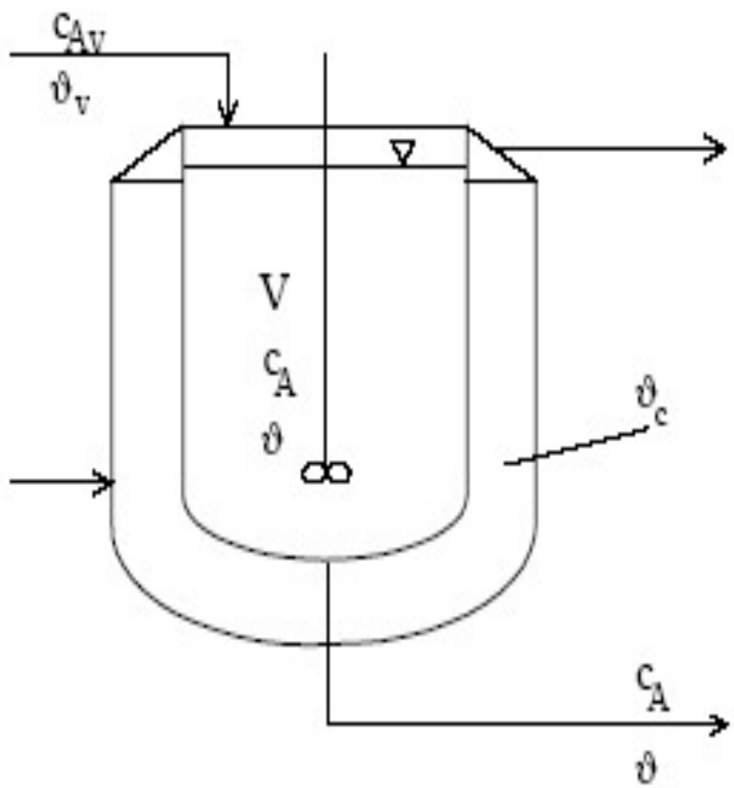
Hybrid Systems Seminar
Part 1: Motivation

Michal Kvasnica

Good control using limited resources

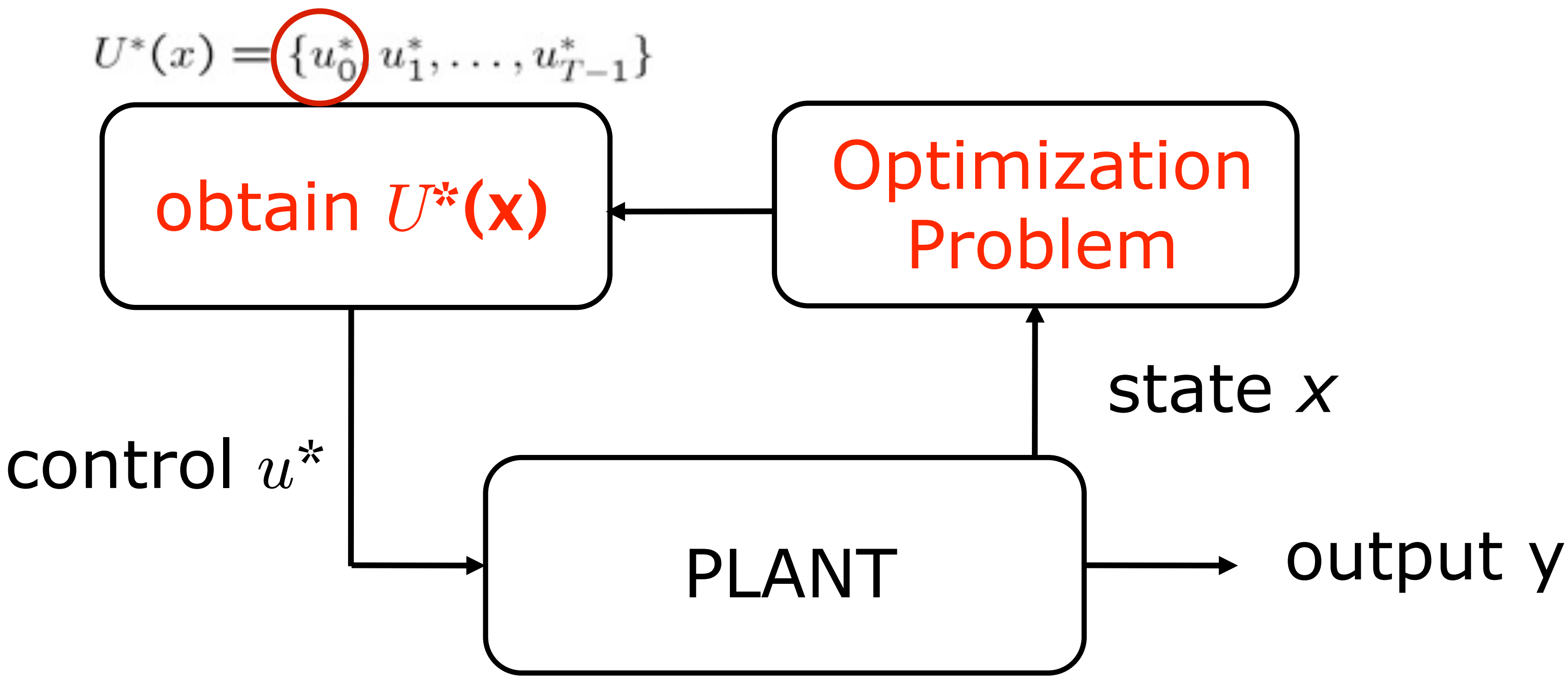
CSTR Control



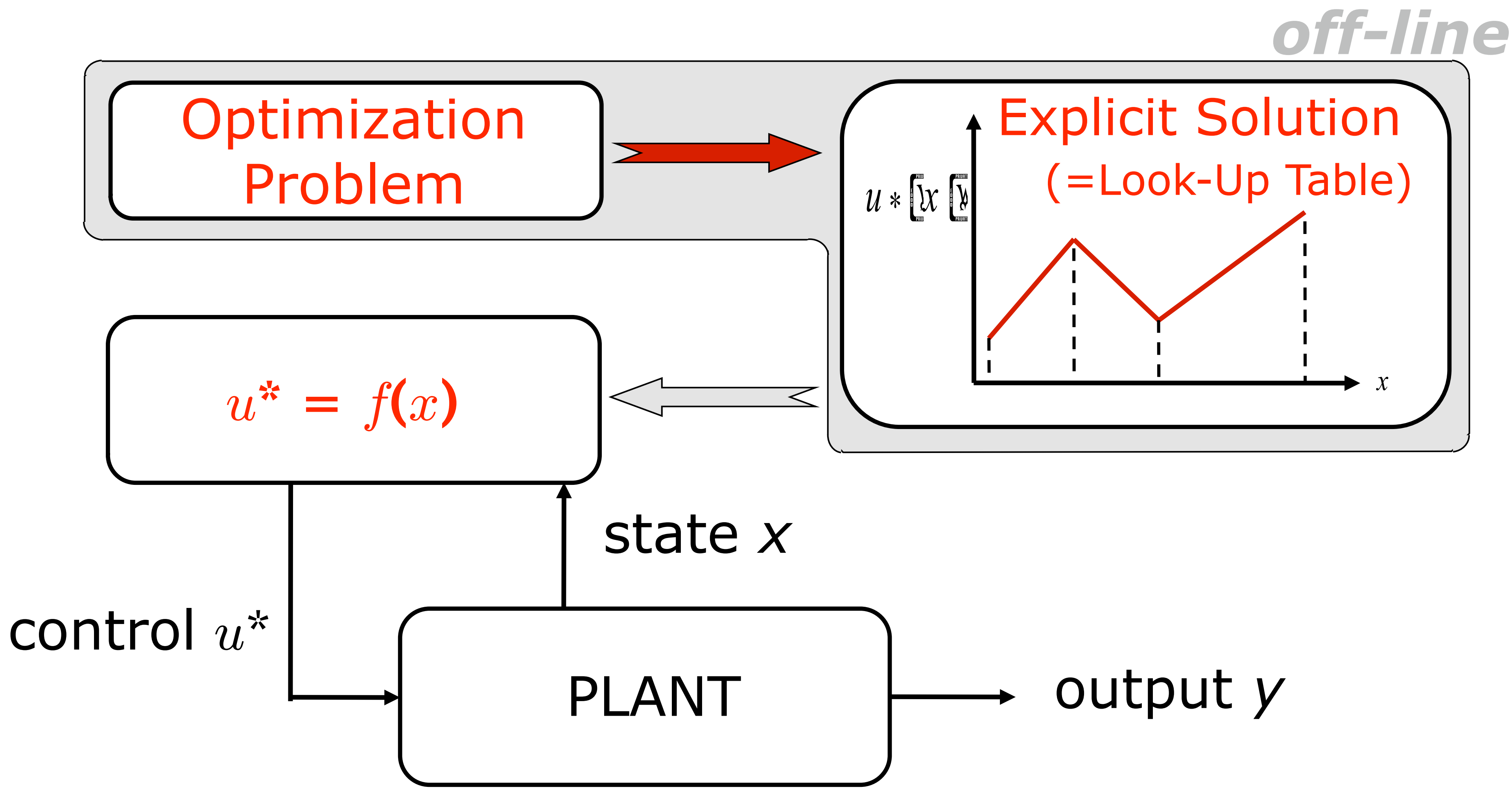
Challenges:

- constraints
- nonlinear behavior
- optimal operation
- cheap implementation in real time

MPC: On-Line Solution



MPC: Off-Line Solution

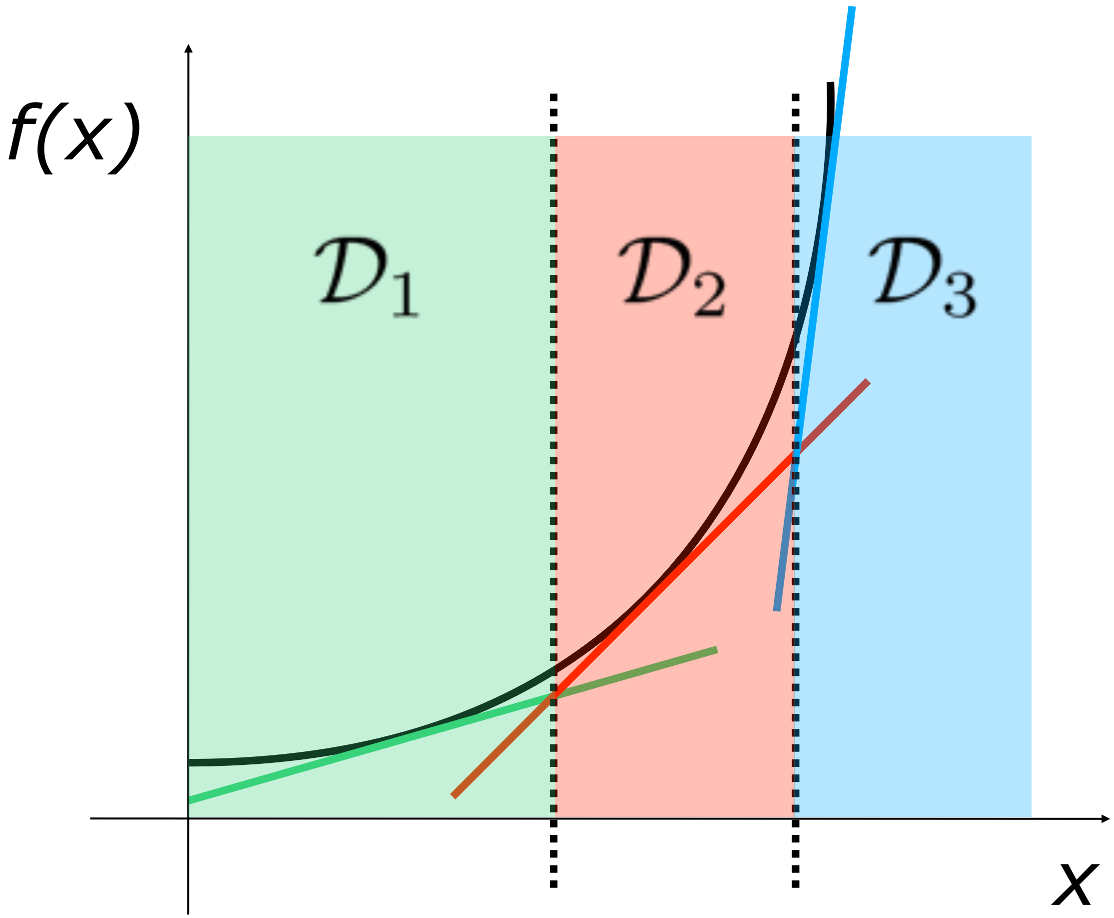


On-Line vs. Off-Line

	On-Line	Off-Line
Fast implementation	✘	✓
Cheap implementation	✘	✓
Nonlinear models	✓	✘

Idea: approximate nonlinearities by a hybrid linear system

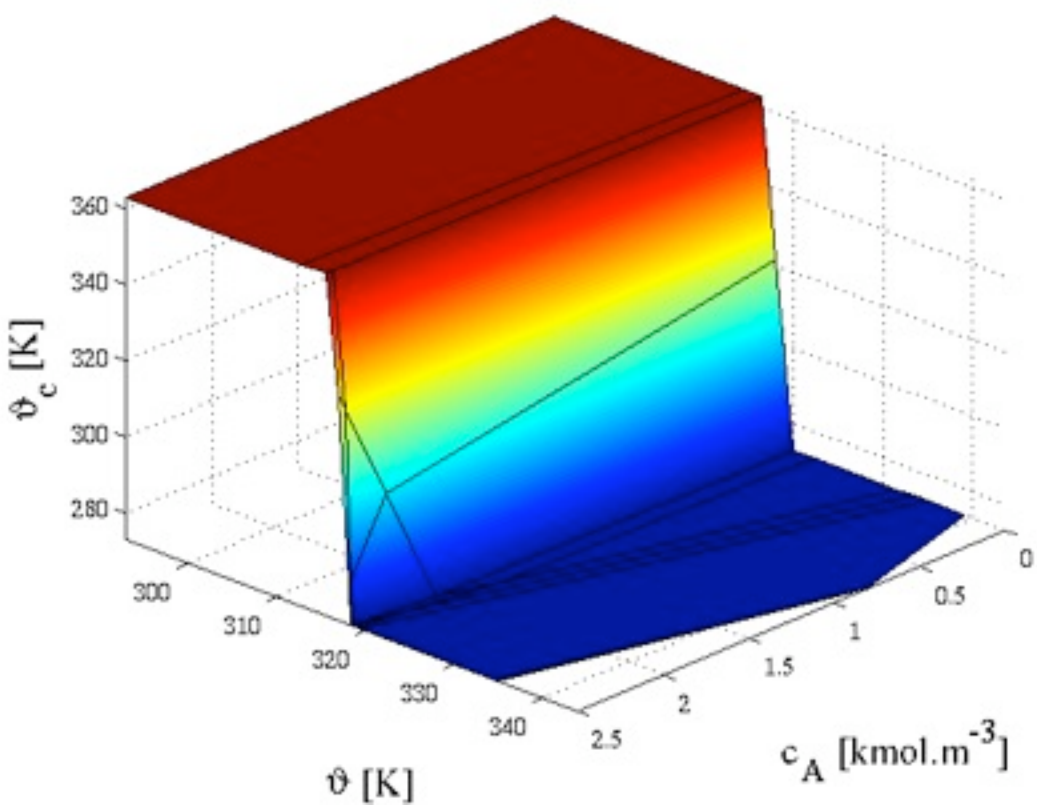
PWA Approximation



- IF-THEN rules translate into an mixed-integer model
- arbitrary precision can be achieved by adding more linearizations

$$x_{k+1} = A_i x_k + B_i u_k + f_i \quad \text{IF} \quad x_k \in \mathcal{D}_i$$

CSTR: Off-Line MPC



- track temperature reference
- use the PWA model to form predictions
- MPT calculates the off-line solution
- 210 regions in 3D

MPT: Multi-Parametric Toolbox, M. Kvasnica et al.

Evaluation

	Nonlinear	PWA	Linear
Performance	100 %	**85 %**	30 %
Runtime	600 ms	**0.5 ms**	0.5 ms
Expenses	1000 €	**10 €**	10 €

Conclusions

- MPC to handle constraints & performance
- Off-line MPC to allow real-time implementation
- PWA approximations to deal with nonlinearities

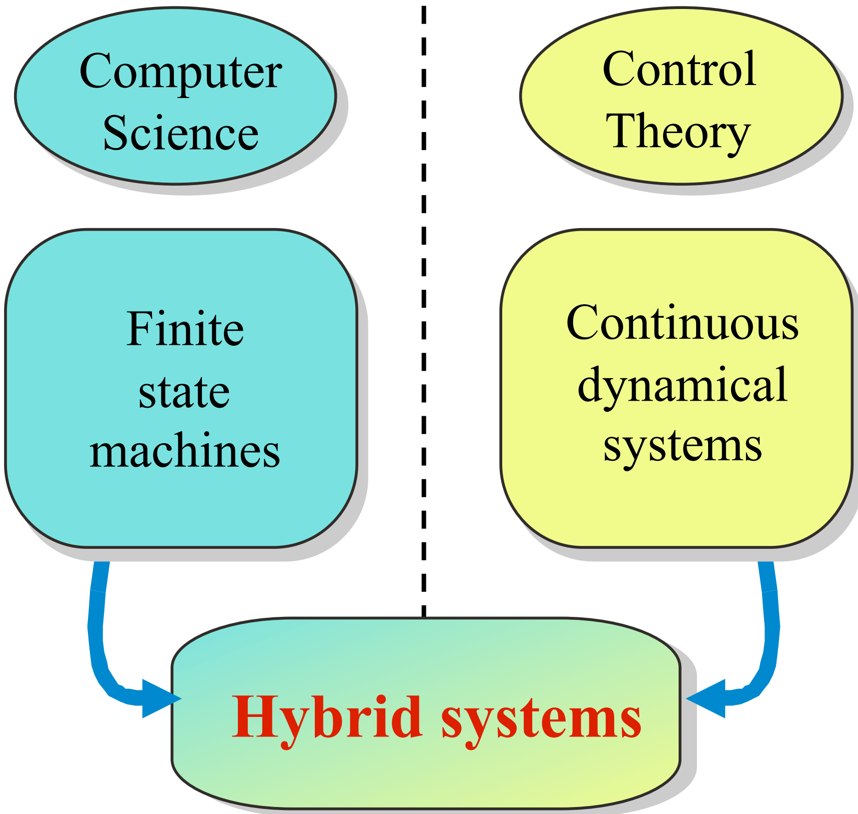
- Summary: well performing control using cheap hardware

Hybrid Systems Seminar
Part 2: Models of Hybrid Systems

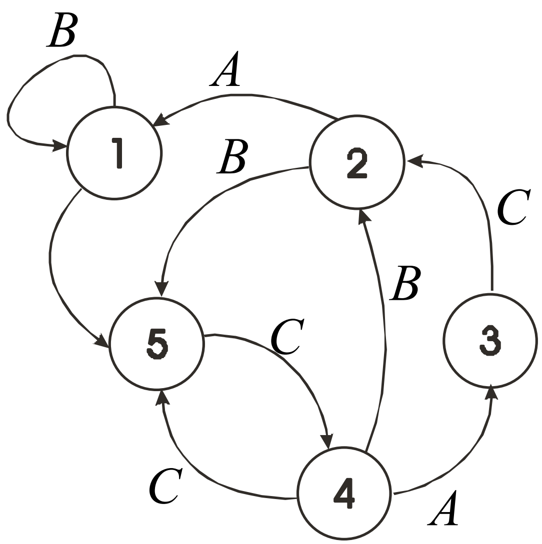
Michal Kvasnica

Hybrid Systems

$$X = \{1, 2, 3, 4, 5\}$$
$$U = \{A, B, C\}$$

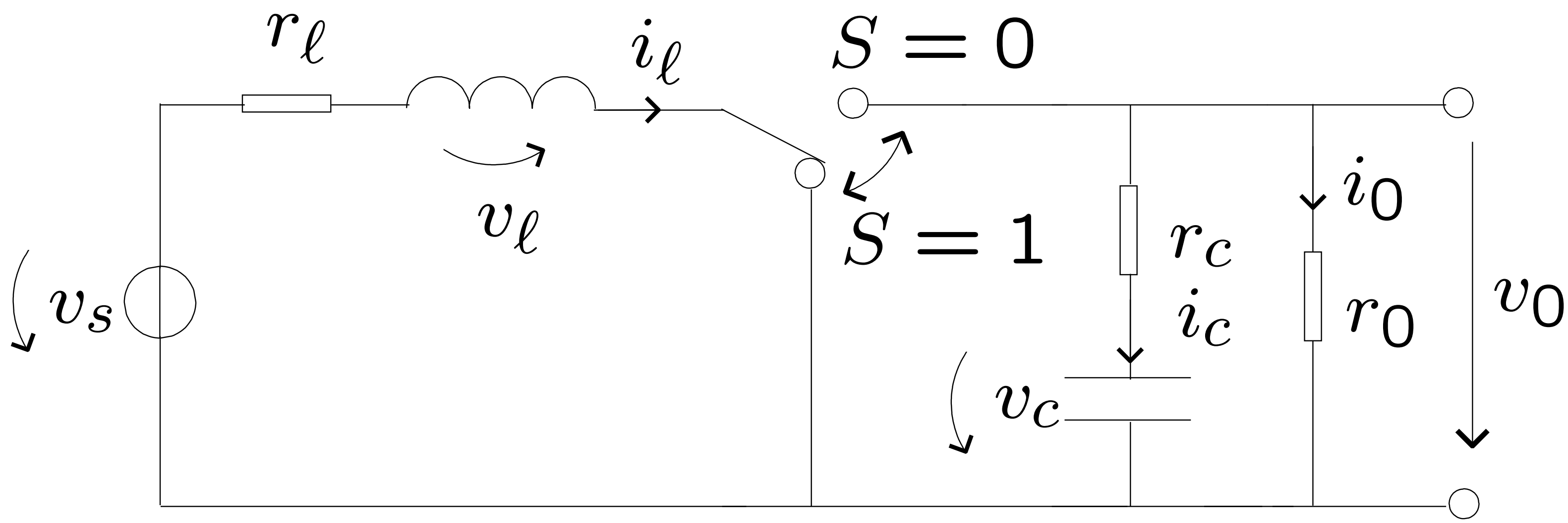


$$x \in \mathbb{R}^{n_x}$$
$$u \in \mathbb{R}^{n_u}$$
$$y \in \mathbb{R}^{n_y}$$



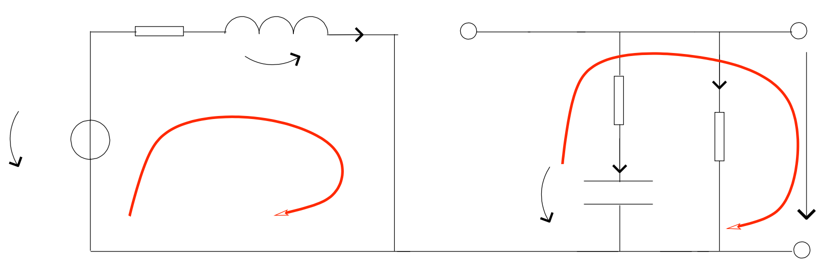
$$\begin{cases} x(k+1) & = & f(x(k), u(k)) \\ y(k) & = & g(x(k), u(k)) \end{cases}$$

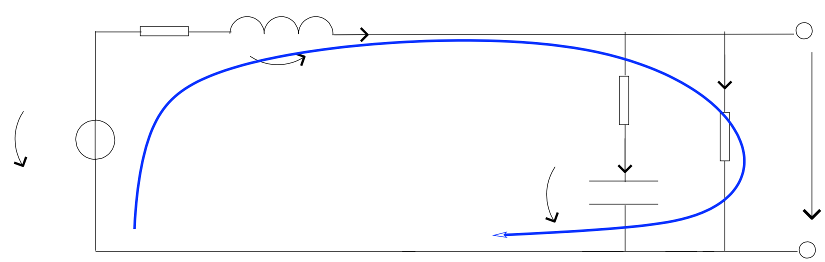
DC-DC Converter



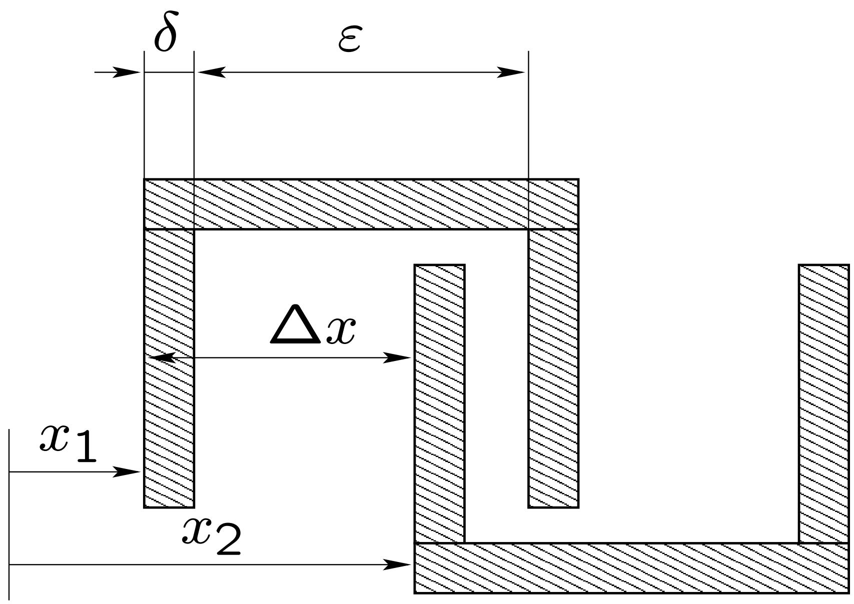
- Continuous states, discrete inputs
- Linear dynamics switches depending on the value of input

MODE 1 ($S = 1$)	MODE 2 ($S = 0$)



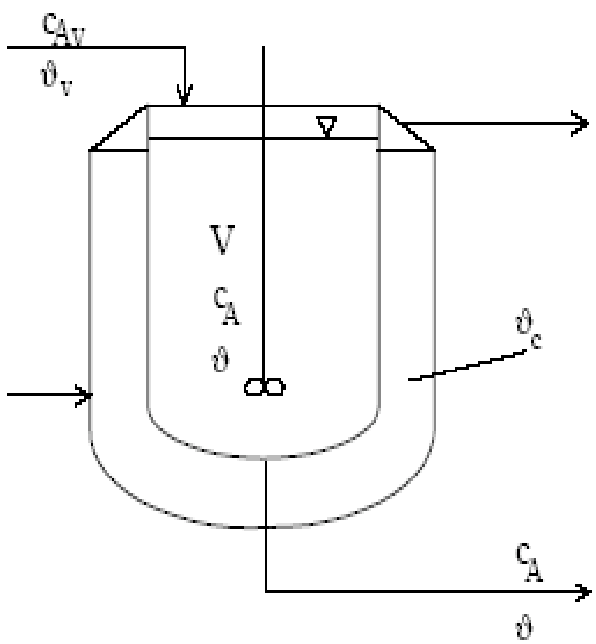


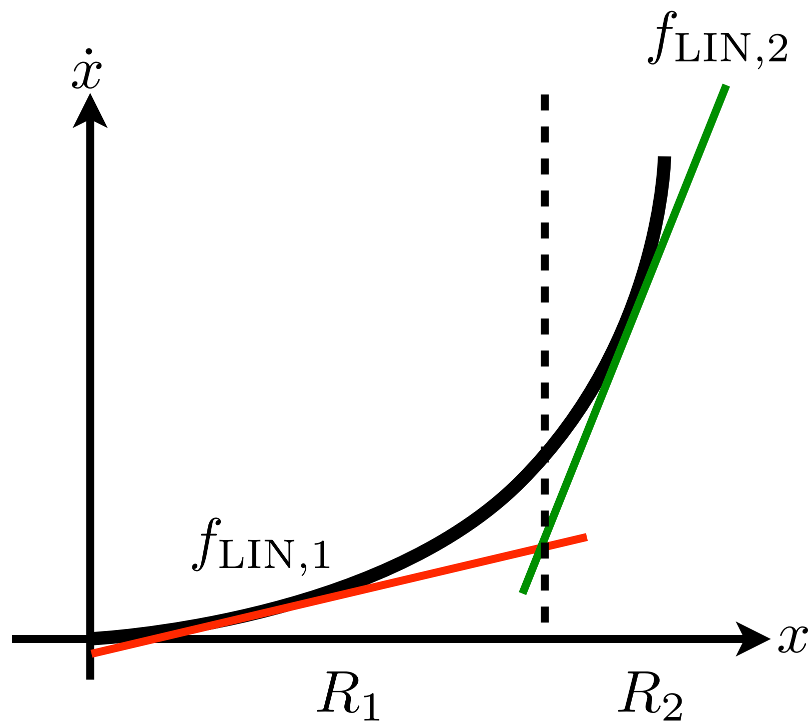
Mechanical System with Backlash



- Continuous states
- Linear dynamics switches between two modes:
  - contact mode $[(\Delta x = \delta) \wedge (\dot{x}_1 > \dot{x}_2)] \vee [(\Delta x = \epsilon) \wedge (\dot{x}_2 > \dot{x}_1)]$
  - backlash mode otherwise

Chemical Reactor





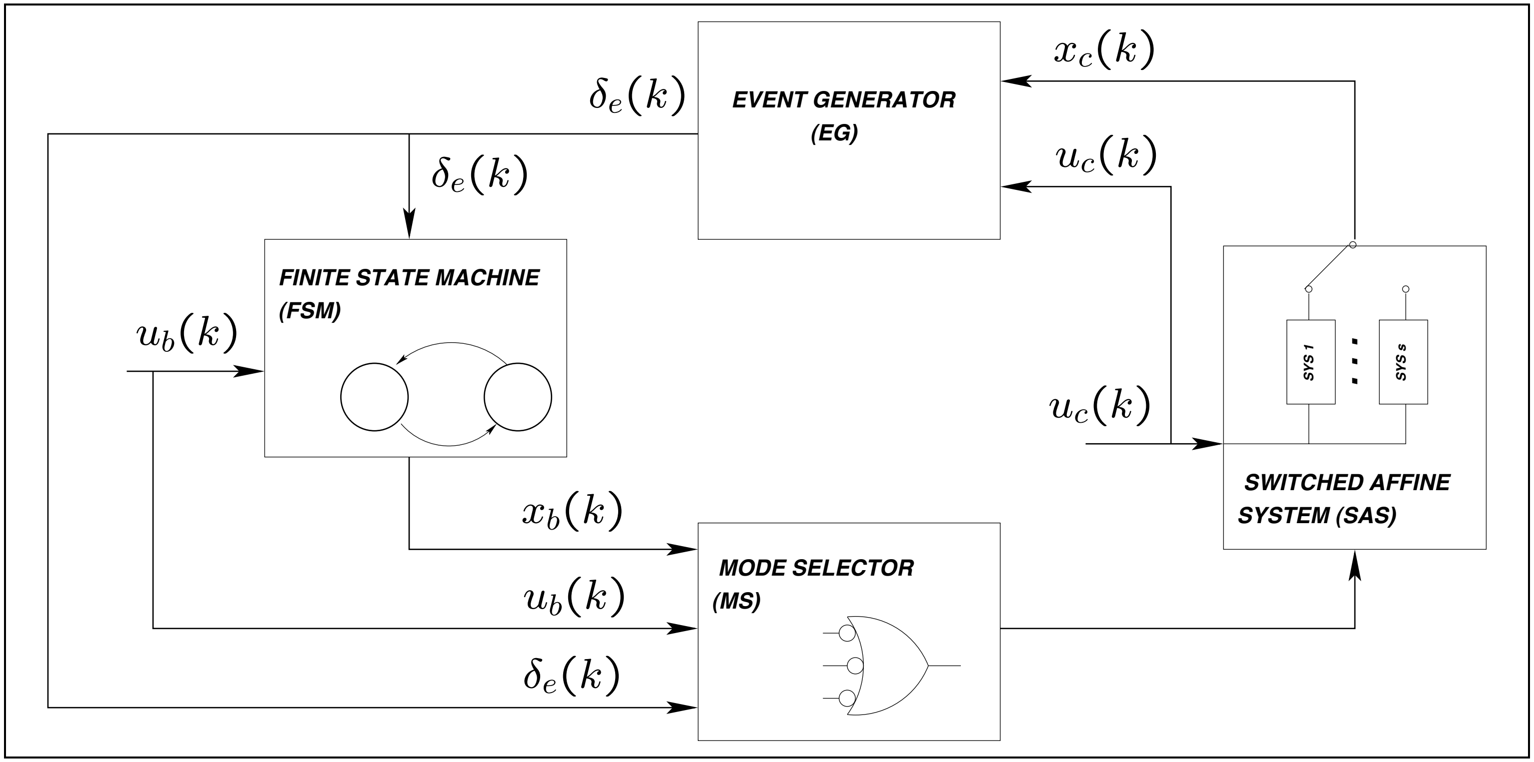
- Continuous states and inputs
- Nonlinear dynamics approximated by multiple linearizations

$$\dot{x} = \begin{cases} f_{\mathrm{LIN},1} \text{ if } x \in R_1 \\ \\ f_{\mathrm{LIN},2} \text{ if } x \in R_2 \end{cases}$$

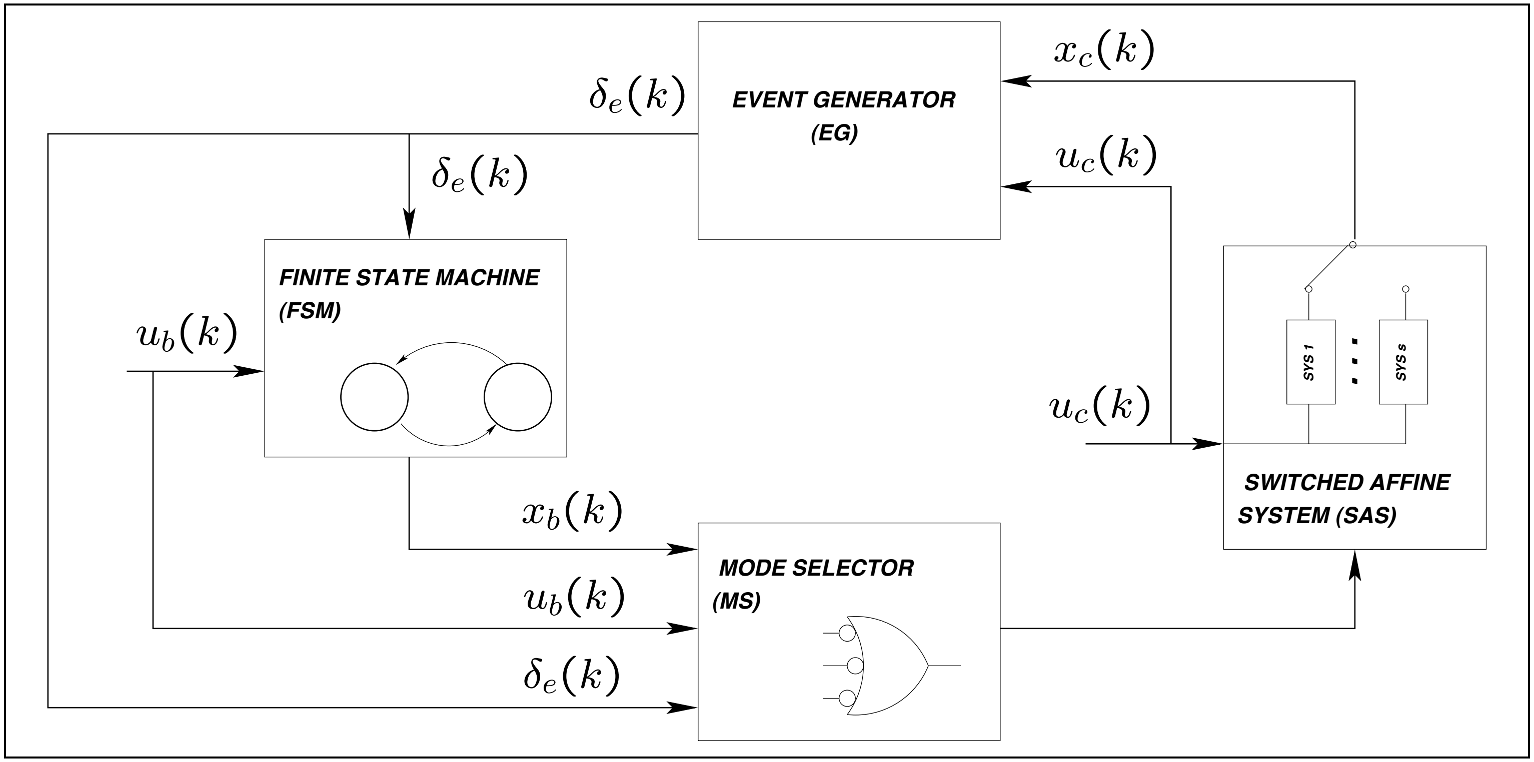
Modeling of Hybrid Systems

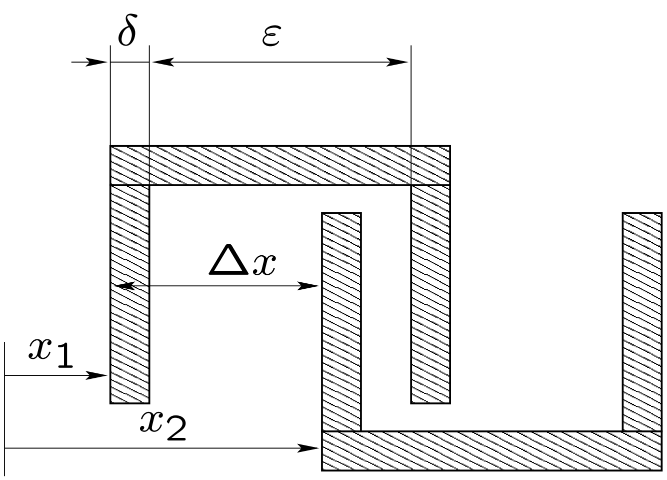
- Suitable mathematical abstraction needed
- For simulations:
  - detailed process description
  - individual modes usually involve nonlinear dynamics
  - can be modeled e.g. using Stateflow / Simulink
- For control:
  - descriptive enough to capture behavior of the plant
  - simple enough to allow controller synthesis
  - dynamics in each mode approximated by an affine expression
  - due to presence of switches the overall dynamics is still nonlinear
  - mathematical representation of the whole system is needed

Discrete Hybrid Automata



Discrete Hybrid Automata



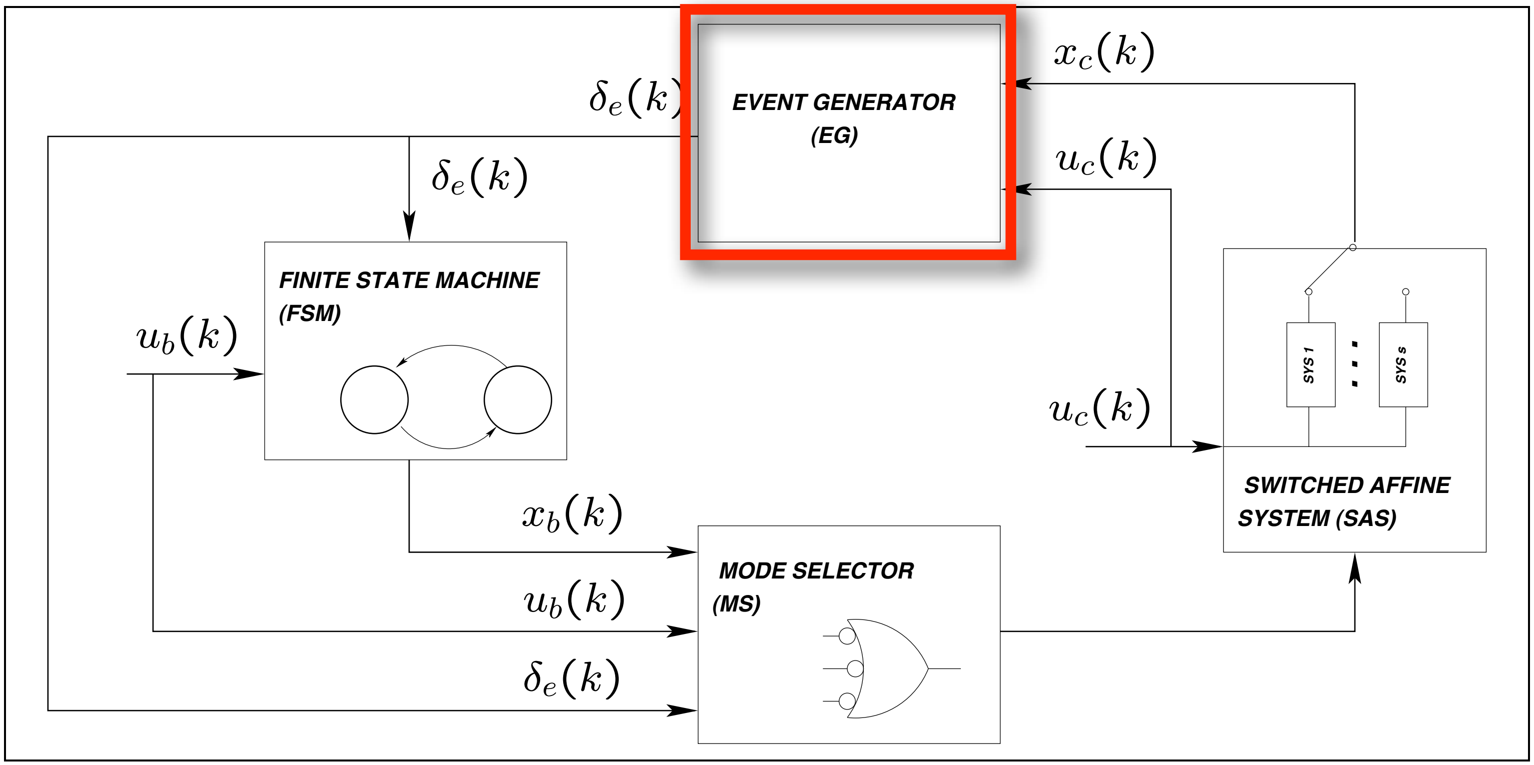


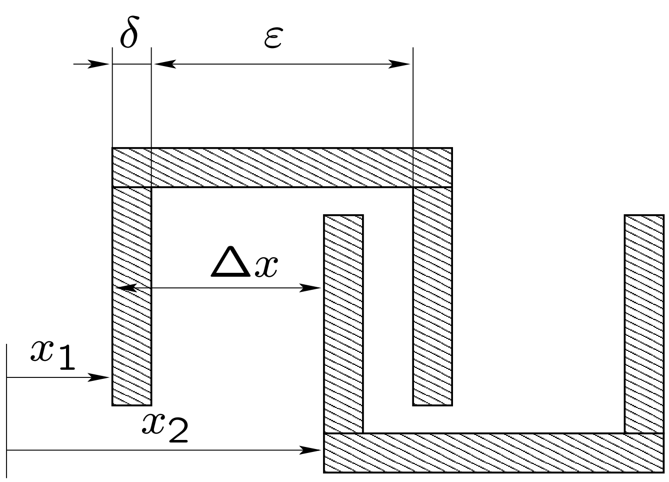
- Contact mode:

$$[(\Delta x = \delta) \wedge (\dot{x}_1 > \dot{x}_2)] \vee [(\Delta x = \epsilon) \wedge (\dot{x}_2 > \dot{x}_1)]$$

- Backlash mode

Discrete Hybrid Automata



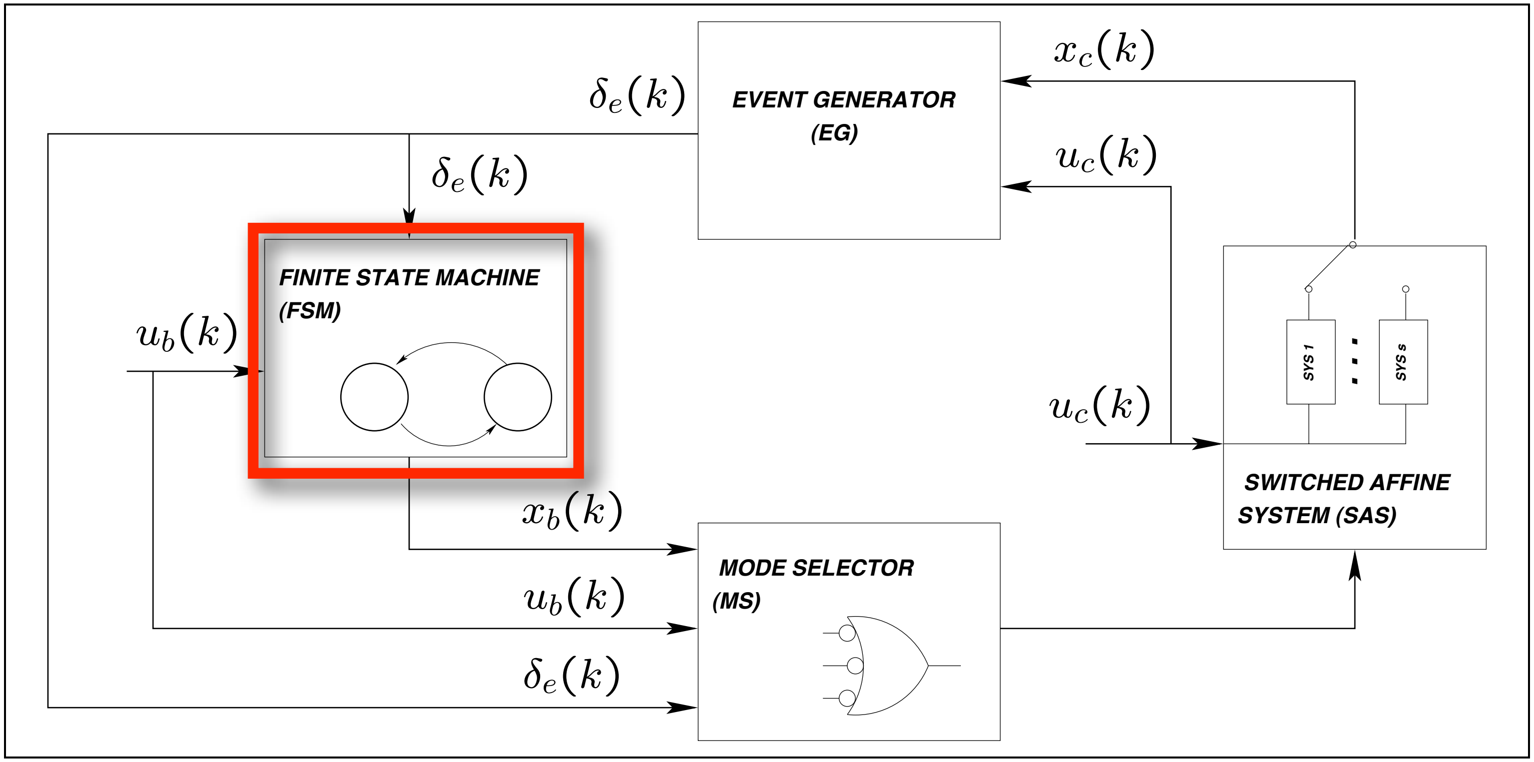


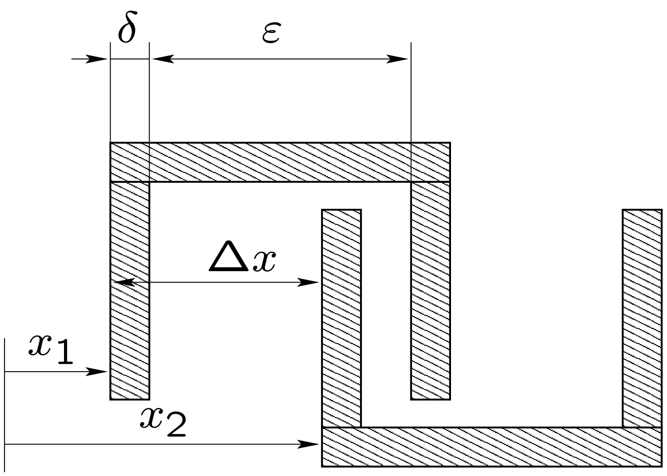
- Contact mode:

$$[(\boxed{\Delta x = \delta}) \wedge (\boxed{\dot{x}_1 > \dot{x}_2})] \vee [(\boxed{\Delta x = \epsilon}) \wedge (\boxed{\dot{x}_2 > \dot{x}_1})]$$

- Backlash mode

Discrete Hybrid Automata



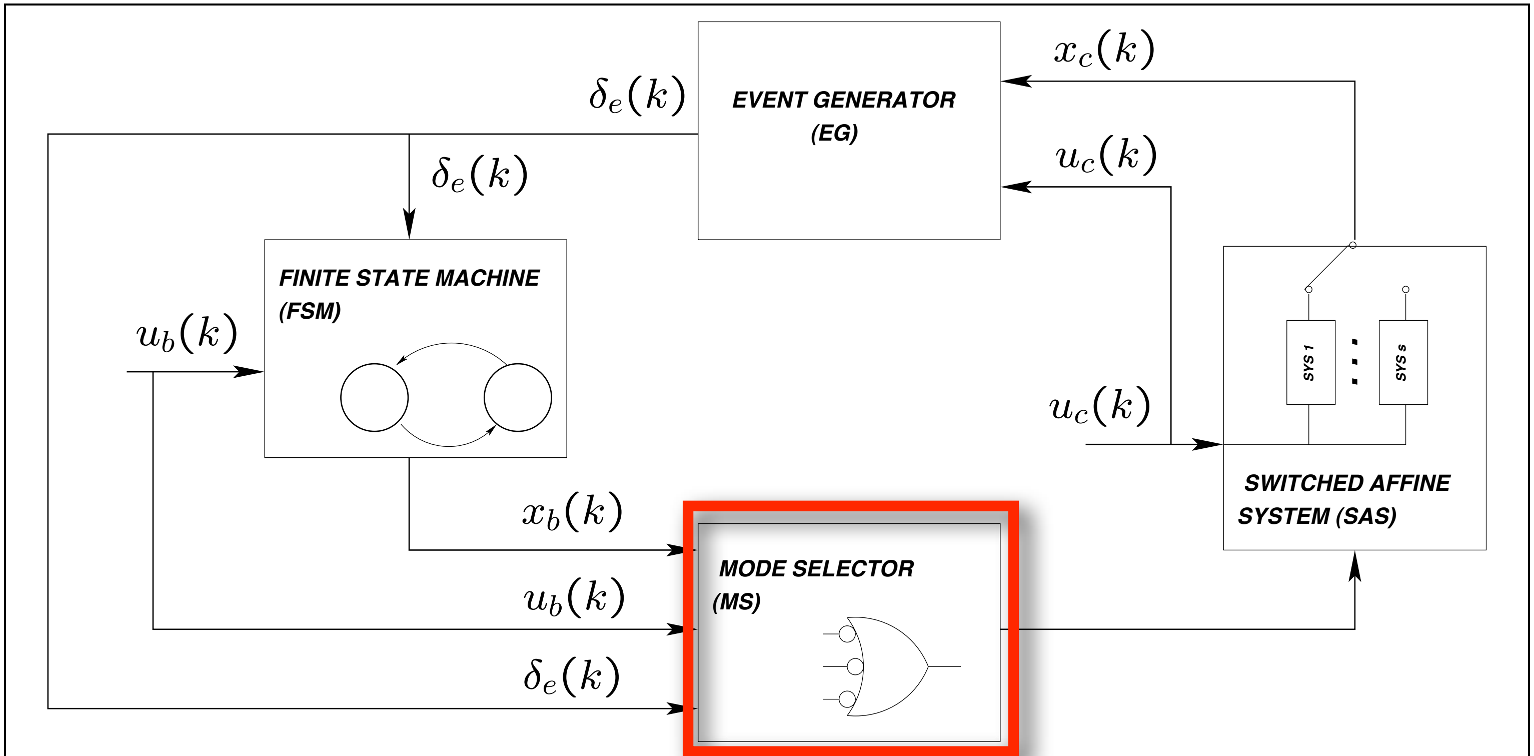


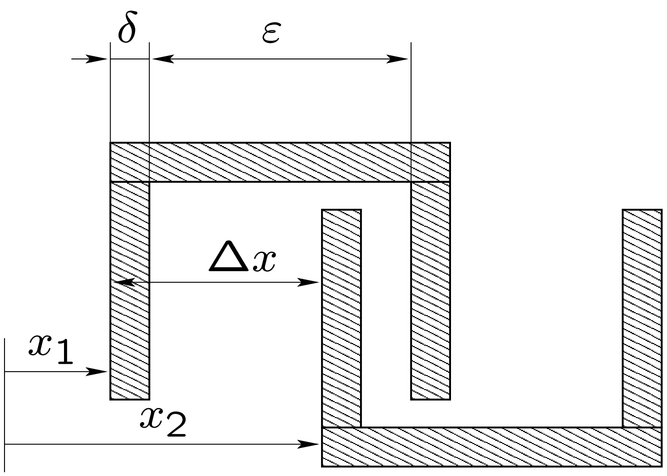
- Contact mode:

$$[(\Delta x = \delta) \boxed{\wedge} (\dot{x}_1 > \dot{x}_2)] \boxed{\vee} [(\Delta x = \epsilon) \boxed{\wedge} (\dot{x}_2 > \dot{x}_1)]$$

- Backlash mode

Discrete Hybrid Automata



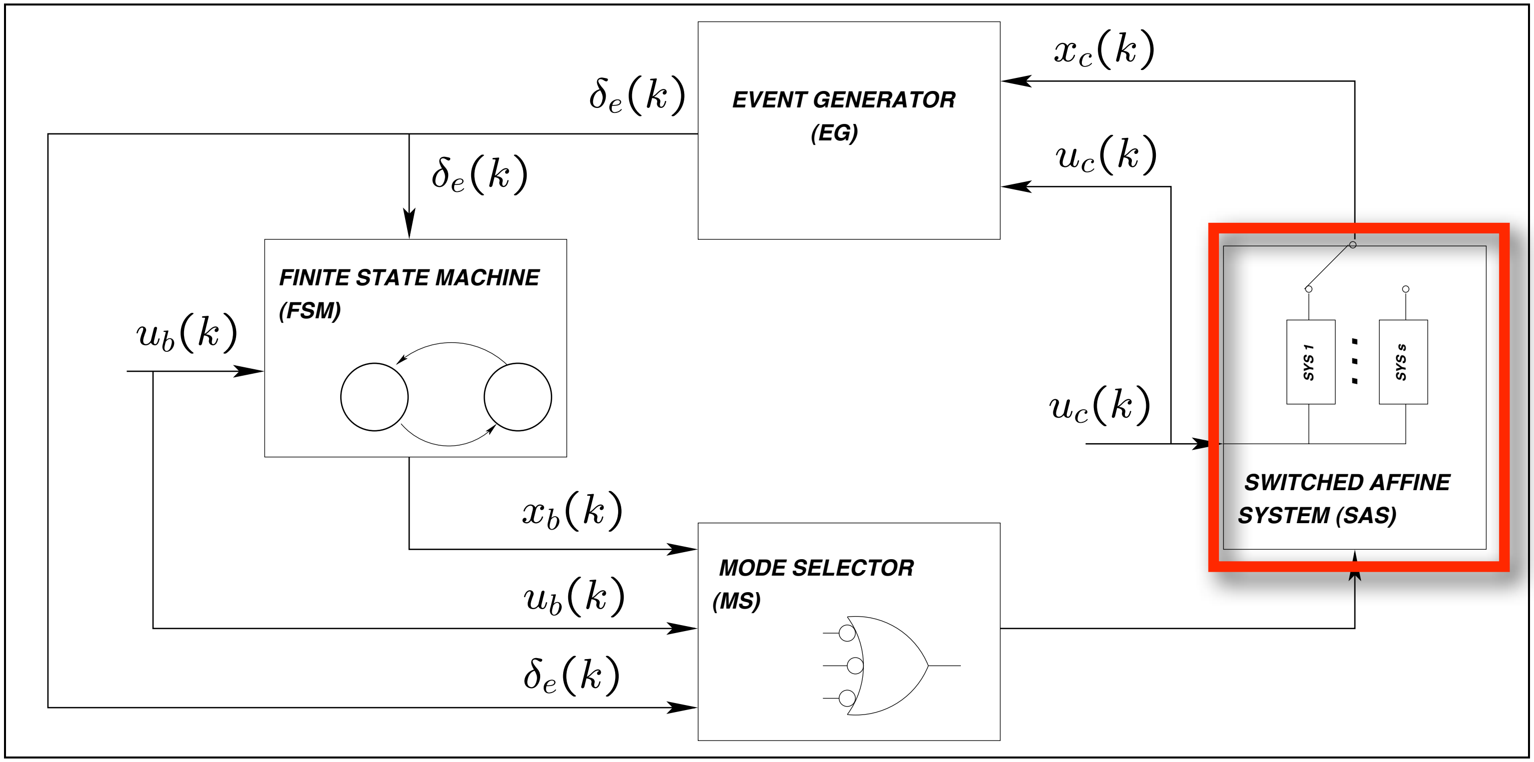


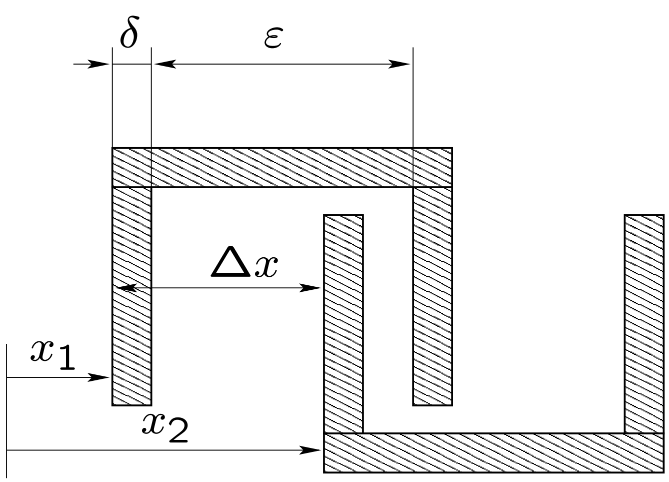
- Contact mode:

$$\boxed{[(\Delta x = \delta) \wedge (\dot{x}_1 > \dot{x}_2)] \vee [(\Delta x = \epsilon) \wedge (\dot{x}_2 > \dot{x}_1)]}$$

- Backlash mode

Discrete Hybrid Automata





- **Contact mode:**

$$[(\Delta x = \delta) \wedge (\dot{x}_1 > \dot{x}_2)] \vee [(\Delta x = \epsilon) \wedge (\dot{x}_2 > \dot{x}_1)]$$

- **Backlash mode**

Mathematical Modeling of DHAs

- Two key issues:
  - how to describe logic components (FSM, event generator, mode selector)
  - how to capture the interaction between binary logic and continuous dynamics?
- Key idea:
  - write logic expressions as a set of inequalities involving binary variables
- Example:

$$
\begin{array}{rl}
\overline{\delta_i} & 1 - \delta_i \\
\delta_i \vee \delta_j & \delta_i + \delta_j \geq 1 \\
\delta_i \wedge \delta_j & \delta_i + \delta_j \geq 2 \\
\delta_i \Rightarrow \delta_j & \delta_i - \delta_j \geq 0 \\
\delta_i \Leftrightarrow \delta_j & \delta_i - \delta_j = 0
\end{array}
$$

Mathematical Modeling of DHAs

- More complex example:

$$\underbrace{(\delta_1 \wedge \delta_2)}_{\delta_a} \Rightarrow \underbrace{(\delta_3 \vee \delta_4)}_{\delta_b}$$

$$(\delta_a \Rightarrow \delta_b) \Leftrightarrow (\delta_a \geq \delta_b)$$

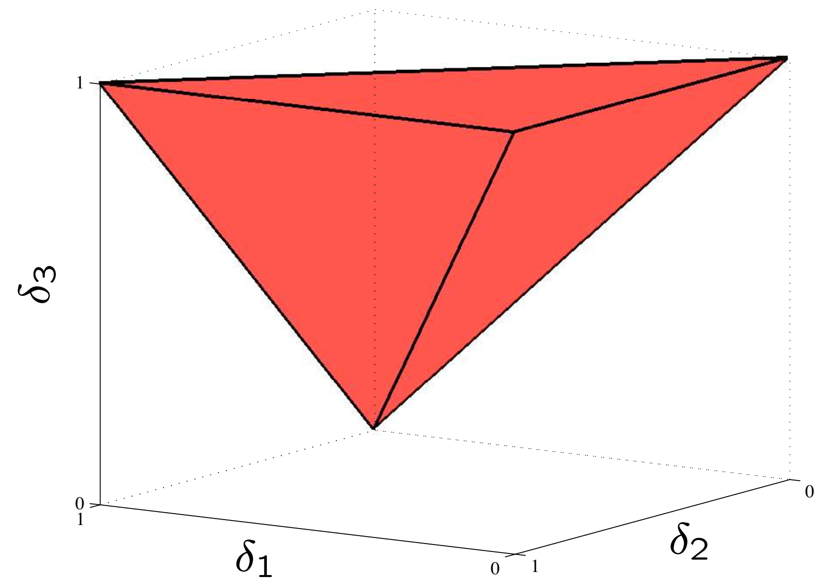
$$\delta_a = (\delta_1 \wedge \delta_2) \;\Leftrightarrow\; \begin{cases} \delta_a \leq \delta_1 \\ \delta_a \leq \delta_2 \\ \delta_1 + \delta_2 \leq 1 + \delta_a \end{cases}$$

$$\delta_b = (\delta_3 \vee \delta_4) \;\Leftrightarrow\; \begin{cases} \delta_b \geq \delta_1 \\ \delta_b \geq \delta_2 \\ \delta_1 + \delta_2 \geq \delta_b \end{cases}$$

Geometric Approach

- Consider any logic expression, e.g. $\delta_3 = (\delta_1 \Rightarrow \delta_2)$
- Create the truth table

δ_1	δ_2	δ_3
0	0	1
0	1	1
1	0	0
1	1	1



- Calculate the convex hull

$$hull\left\{\begin{bmatrix}0\\0\\1\end{bmatrix},\begin{bmatrix}0\\1\\1\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}1\\1\\1\end{bmatrix}\right\} = \left\{\begin{array}{rcr} \delta_2 - \delta_3 & \leq & 0 \\ \delta_3 & \leq & 1 \\ \delta_1 - \delta_2 + \delta_3 & \leq & 1 \\ -\delta_1 - \delta_3 & \leq & -1 \end{array}\right.$$

Mathematical Modeling of DHAs

- Relations between logic and continuous variables modeled in a similar fashion
- Assume a bounded function $m \leq f(x) \leq M$
- Mathematical representation of the event generator:

$$([f(x) \leq 0] \Leftrightarrow [\delta = 1]) \quad \Leftrightarrow \quad \begin{cases} f(x) \leq M(1-\delta) \\ f(x) \geq \epsilon + (m - \epsilon)\delta \end{cases}$$

Mathematical Modeling of DHAs

- Mode selector and switched affine system:

$$x(t+1) = \begin{cases} f_1(x) \text{ if } (\delta_1 = 1) \\ \vdots \\ f_n(x) \text{ if } (\delta_n = 1) \end{cases}$$

- Rewrite as $x(t+1) = z_1 + \cdots + z_n$ with $z_i = f_i(x)\delta_i$
- Corresponding mathematical representation:

$$\begin{aligned} z_i &\leq M\delta_i \\ z_i &\geq m\delta_i \\ z_i &\leq f_i(x) - m(1-\delta_i) \\ z_i &\geq f_i(x) - M(1-\delta_i) \end{aligned}$$

Mixed Logical Dynamical (MLD) Systems

- Compact mathematical representation of hybrid systems

$$\boxed{\begin{aligned}&x(t+1) = Ax(t) + B_u u(t) + B_\delta \delta(t) + B_z z(t)\\&y(t) = Cx(t) + D_u u(t) + D_\delta \delta(t) + D_z z(t)\\&E_x x(t) + E_u u(t) + E_\delta \delta(t) + E_z z(t) \leq E_0\end{aligned}}$$

- Involves continuous and binary states, inputs, outputs
- Auxiliary variables:
  - binary selectors $\delta(t)$
  - continuous variables $z(t)$
- Mixed-integer linear constraints:
  - include physical constraints on state, inputs, outputs
  - capture events, FSM, mode selection

Automatic Generation of MLD Descriptions?

- Example:

$$x(t+1) = \begin{cases} 0.8x(t) + u(t) & \text{if } x(t) \le 0 \\ -0.8x(t) + u(t) & \text{if } x(t) > 0 \end{cases}$$

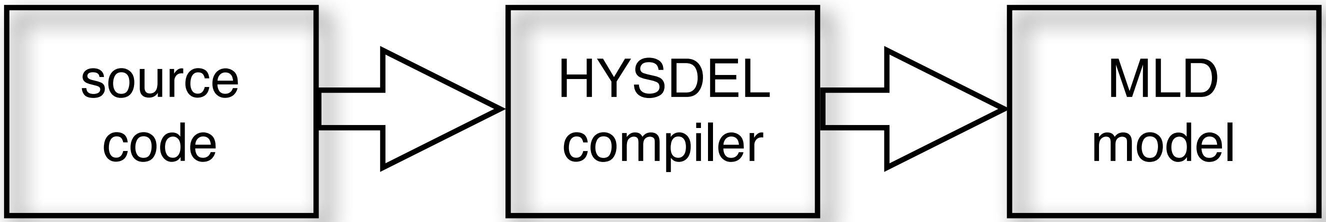
- Associate $(\delta(t) = 1) \Leftrightarrow (x(t) \le 0)$
- Rewrite state-update equation $x(t+1) = 1.6\delta(t)x(t) - 0.8x(t) + u(t)$
- Introduce auxiliary variable $z(t) = \delta(t)x(t)$

$$x(t+1) = 1.6z(t) - 0.8x(t) + u(t)$$

- Formulate constraints:

$$\begin{aligned} x(t) &\le M(1 - \delta(t)) \\ x(t) &\ge \epsilon + (m - \epsilon)\delta(t) \\ z(t) &\le M\delta(t) \\ z(t) &\ge m\delta(t) \\ z(t) &\le x(t) - m(1 - \delta(t)) \\ z(t) &\le x(t) - M(1 - \delta(t)) \end{aligned}$$

HYbrid Systems DEscription Language (HYSDEL)

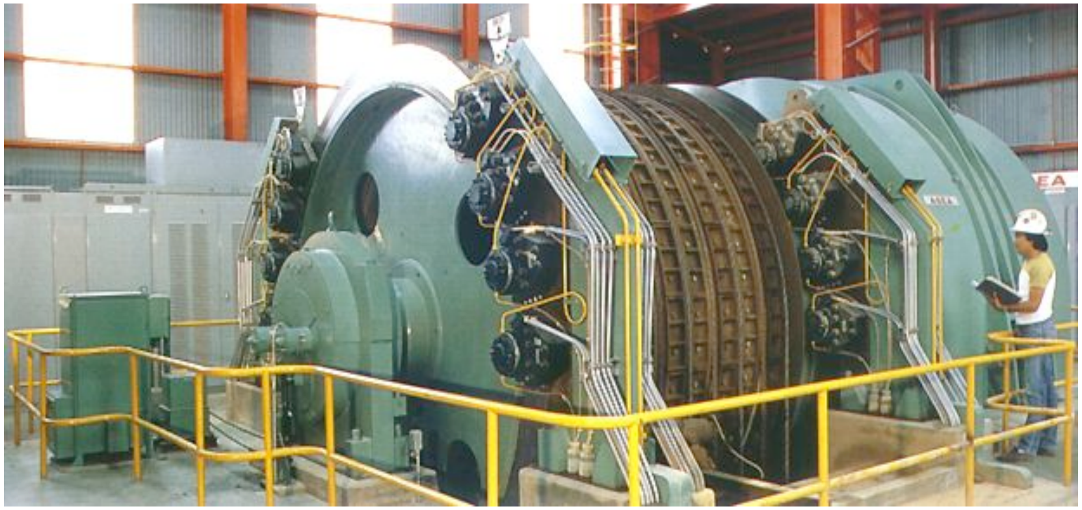



```
SYSTEM switched_system {
   INTERFACE {
      STATE { REAL x [-10, 10]; }
      INPUT { REAL u [-1, 1]; }
   }
   IMPLEMENTATION {
      AUX { BOOL delta; REAL z; }
      AD  { delta = (x <= 0); }
      DA  { z = {IF delta THEN  0.8*x ELSE -0.8*x}; }
      CONTINUOUS { x = z + u; }
   }
}
```

Event Generator = AD Section


```
SYSTEM tank {
   INTERFACE {
      STATE {
         REAL h; }
      INPUT {
         REAL Q; }
      OUTPUT {
         BOOL overflow; }
      PARAMETER {
  REAL k     = 1; }
   } /* end interface */
   IMPLEMENTATION {
      AUX {
         BOOL s; }
      AD {
         s = (h >= hmax); }
      CONTINUOUS {
         h = h + k * Q; }
      OUTPUT {
         overflow = s; }
   } /* end implementation */
} /* end system */
```

Mode Selector + Switched System = DA Section



Nonlinear amplification unit

$$u_{comp} = \begin{cases} u & (u < u_t) \\ 2.3u - 1.3u_t & (u \geq u_t) \end{cases}$$

```
SYSTEM motor {
    INTERFACE {
        STATE {
            REAL ucomp; }
        INPUT {
            REAL u [0, umax];}
        PARAMETER {
            REAL ut   = 1;
            REAL umax = 10;}
    } /* end interface */
```

```
IMPLEMENTATION {
        AUX {
            REAL unl;
            BOOL th; }
        AD {
            th = (u >= ut); }
        DA {
            unl = { IF th THEN 2.3*u - 1.3*ut
                           ELSE u}; }
        CONTINUOUS {
            ucomp = unl; }
    } /* end implementation */
} /* end system */
```

Logic Expressions

$$u_{brake} = u_{alarm} \wedge (\neg s_{tunnel} \vee s_{fire})$$

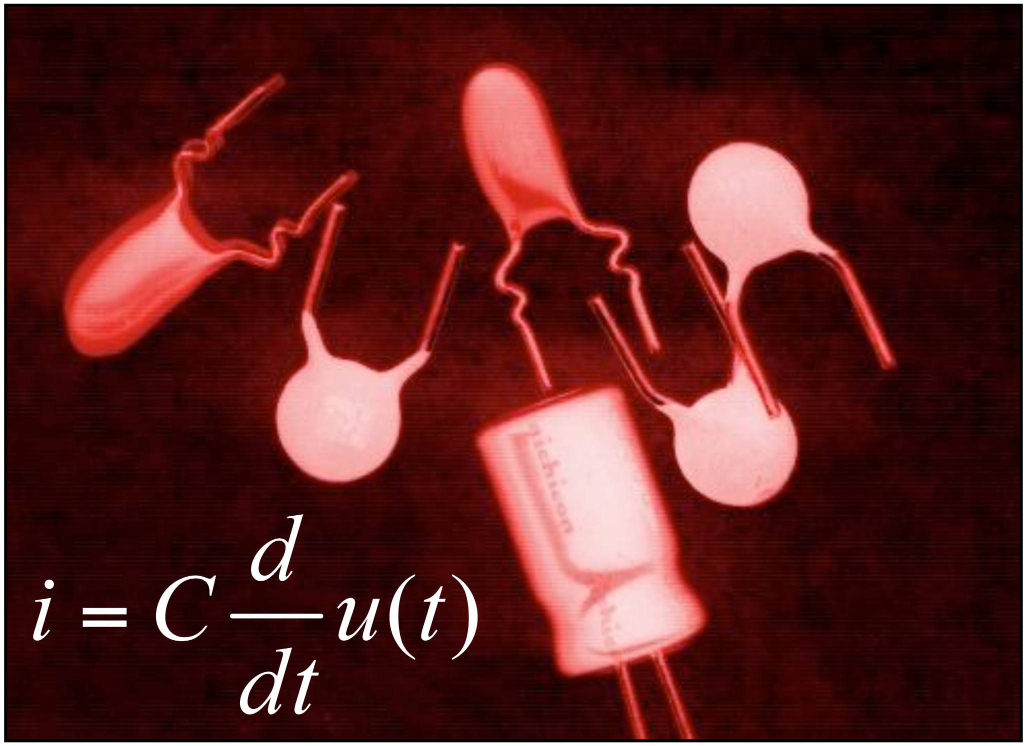
$$s_{fire} \longrightarrow u_{alarm}$$

```
SYSTEM train {
   INTERFACE {
      STATE {
         BOOL brake; }
      INPUT {
         BOOL alarm, tunnel, fire; }

   } /* end interface */


   IMPLEMENTATION {
      AUX {
         BOOL decision; }
      LOGIC {
         decision =
            alarm & (~tunnel | fire); }
      AUTOMATA {
         brake = decision; }
      MUST {
         fire -> alarm; }
   } /* end implementation */
} /* end system */
```

Discrete-Time Dynamics



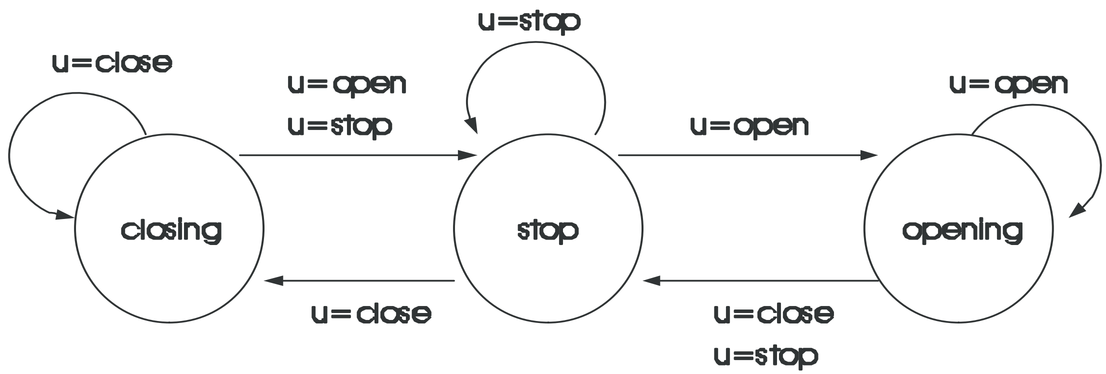

```
SYSTEM capacitorD {
   INTERFACE {
      STATE {
         REAL u; }
      PARAMETER {
         REAL R = 1e4;
         REAL C = 1e-4;
         REAL T = 1e-1; }
   } /* end interface */


   IMPLEMENTATION {
      CONTINUOUS {
        u = u - T/C/R*i;  }
   } /* end implementation */
} /* end system */
```

Forward Euler discretization:

$$u(k+1) = u(k) + \frac{T}{C}i(k)$$

Finite State Machines



```
SYSTEM outflow {
   INTERFACE {
      STATE {
         BOOL closing, stop, opening; }
      INPUT {
         BOOL uclose, uopen, ustop; }
   } /* end of interface */

   IMPLEMENTATION {
      AUTOMATA {
         closing = (uclose & closing) | (uclose & stop);
         stop    = ustop | (uopen & closing) | (uclose & opening);
         opening = (uopen & stop) | (uopen & opening);   }
      MUST {
         ~(uclose & uopen);
         ~(uclose & ustop);
         ~(uopen  & ustop); }
   } /* end implementation */
} /* end system */
```

Constraints

$$0 \le h \le h_{max}$$

```
SYSTEM watertank {
   INTERFACE {
      STATE {
         REAL h; }
      INPUT {
         REAL Q; }
      PARAMETER {
         REAL hmax = 0.3;
         REAL k    = 1; }
   } /* end interface */


   IMPLEMENTATION {
      CONTINUOUS {
         h = h + k*Q; }
      MUST {
          h - hmax <= 0;
         -h        <= 0; }
   } /* end implementation */
} /* end system */
```

HYSDEL

- Generates MLD mathematical description out of user-provided source file
- Translates arbitrary logic conditions into appropriate mixed-integer constraints
- Automatically calculates lower/upper bounds of linear expressions
- Allows to simulate MLD systems in MATLAB & Simulink
- GPL-based tool
- http://control.ee.ethz.ch/~hybrid/hysdel/

HYSDEL 3.0

Michal Kvasnica, Martin Herceg

Automatic Control Laboratory, ETH Zürich

www.control.ethz.ch

ABB

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ABB Success Stories

Jura Cement and ABB Switzerland achieved the first known **successful application of a MLD system on a cement mill.**

The outcome has been that the **mill can be run for maximum production** and also ensuring energy inputs and additives are used efficiently and effectively.

ABB Success Stories

ABB

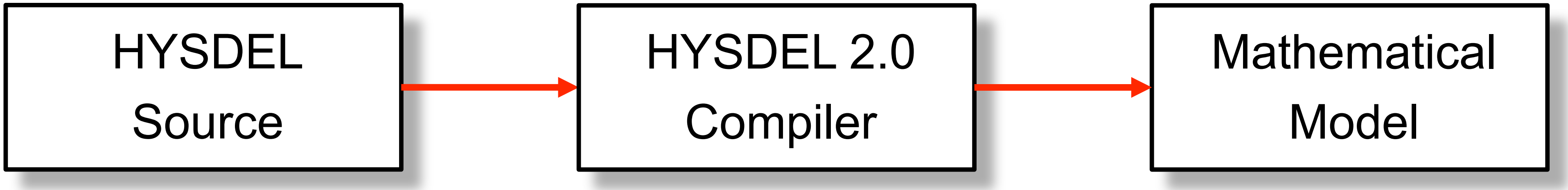
ABB technology wins 2008 Global Fuels Award for energy efficiency

February 19, 2008 – **ABB's Expert Optimizer software was honored with the "Most innovative technology for electrical energy efficiency" award** at the second annual Global Fuels conference in London earlier this month. Part of ABB's Collaborative Production Management portfolio, **Expert Optimizer helps cement plants to significantly reduce their energy consumption and energy costs**. Pro Publications International Ltd. organized the conference; over 100 cement industry delegates from 27 countries attended the 2008 event.

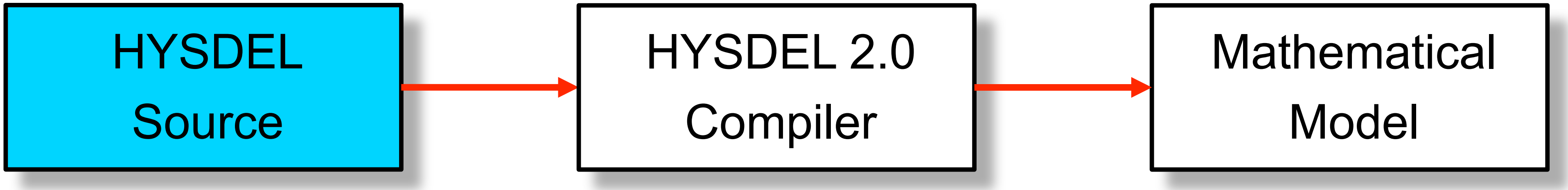
HYSDEL

- HYSDEL = Hybrid Systems Description Language
- HYSDEL is a framework for modeling of hybrid systems
  – uses simple natural language statements to model complex relations
  – generates mathematical models suitable for plant optimization
- Two versions are available:
  – HYSDEL 2.0 – the official version
  – HYSDEL 3.0 – currently under development

Operation Principle of HYSDEL 2.0

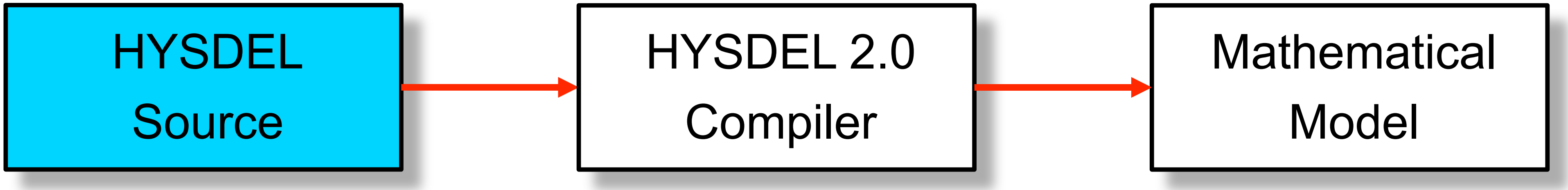


Main HYSDEL 2.0 Language Features



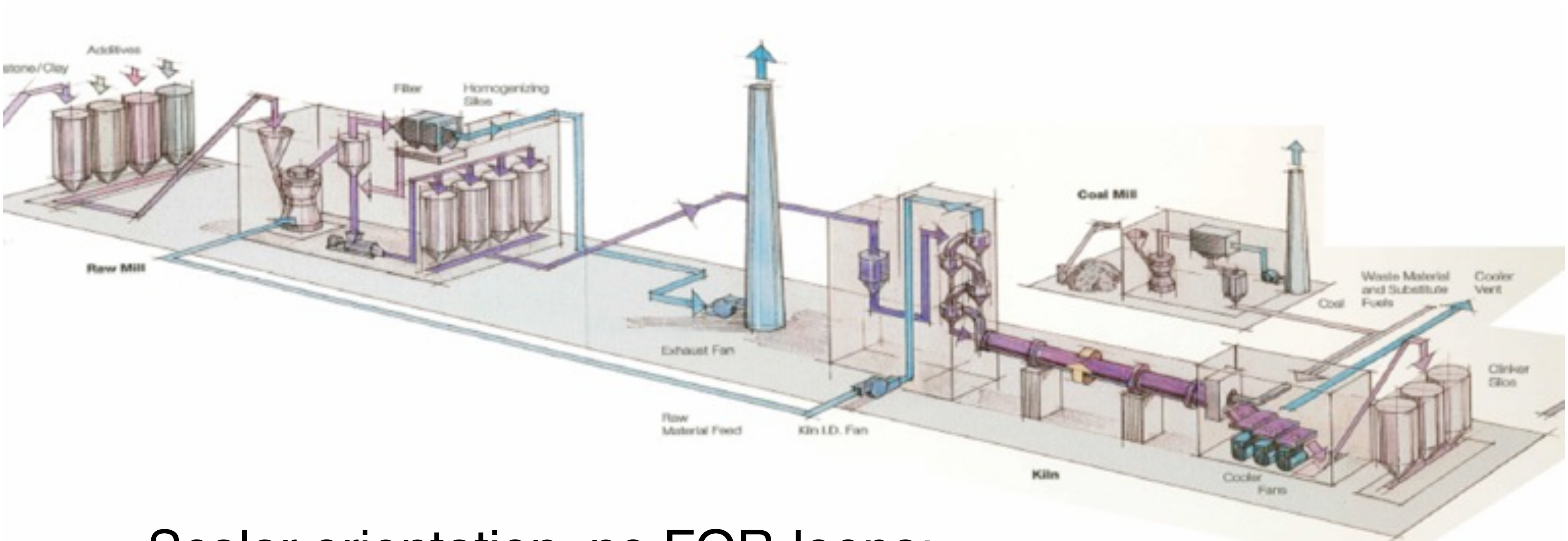
- Hybrid systems modeling can be based on:
  - difference equations
  - on/off switches
  - IF-THEN-ELSE rules
  - finite state automata
- Variables can be marked as binary or real
- Constraints can be defined

HYSDEL 2.0 Language



- PROS:
  - easy to understand syntax similar to C/C++
  - allows rapid prototyping of hybrid systems
- CONS:
  - only allows scalar variables to be defined
  - doesn't allow FOR loops to be used
  - compositions of multiple models not allowed

Cons Illustrated

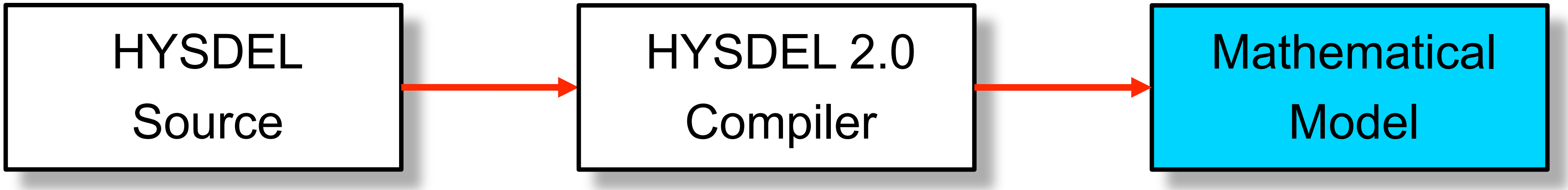


- Scalar orientation, no FOR-loops:
  - creation of models is tedious
- No support for compositions of multiple models:
  - one single model has to describe the whole plant

HYSDEL 2.0 Compiler

- PRO: written in C++
  - very fast processing of source files
- CONS: written in C++
  - maintenance difficult
  - poorly extendible
  - requires compilation for different OS platforms
  - no access to optimization packages that may required to get higher quality models

Mathematical Model

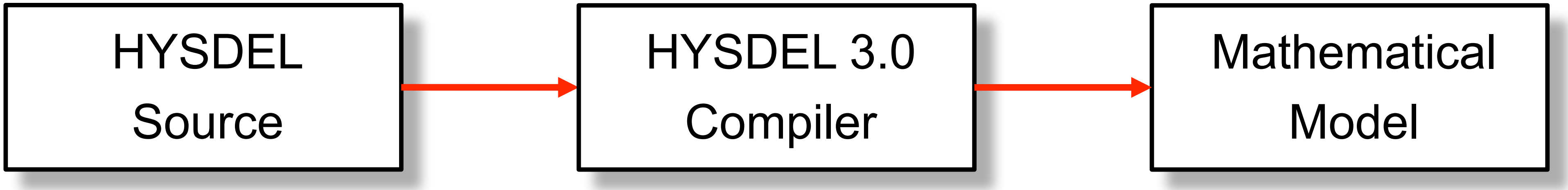


- Represents a mathematical equivalent of the natural language model
- Serves to predict the evolution of the plant
- Can be directly used for plant optimization and simulation
- Question: can different model be obtained that reduces optimization time?

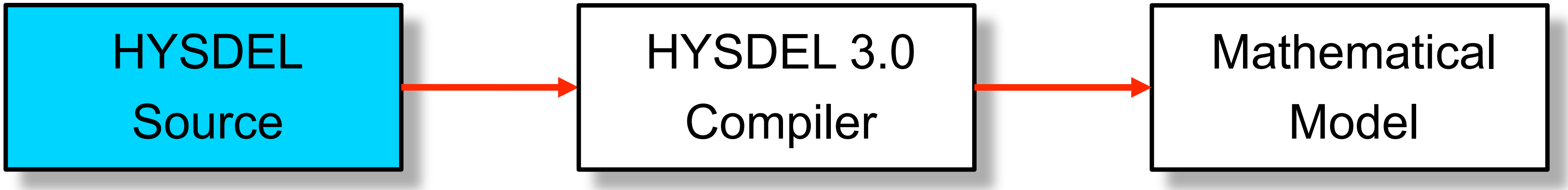
HYSDEL 3.0

- Main goal: address all shortcomings of HYSDEL 2.0
- Particular goals:
  – extend the HYSDEL 2.0 syntax
  – allow compositions of hybrid systems
  – rewrite the compiler
  – generate “faster” models (in terms of optimization time)

Operation Principle of HYSDEL 3.0



HYSDEL 3.0 Language Extentions



- Variables can be in form of vectors and matrices
- Access to individual components of vectors by means of indexing
- Nested FOR loops are allowed
- Hybrid systems consisting of subsystems can be defined

Examples

Vectors, matrices

```
PARAMETER { REAL A = [1, 2; 3, 4]; }
STATE {
  REAL x(nx*N, 2) [lb, ub];
}
```

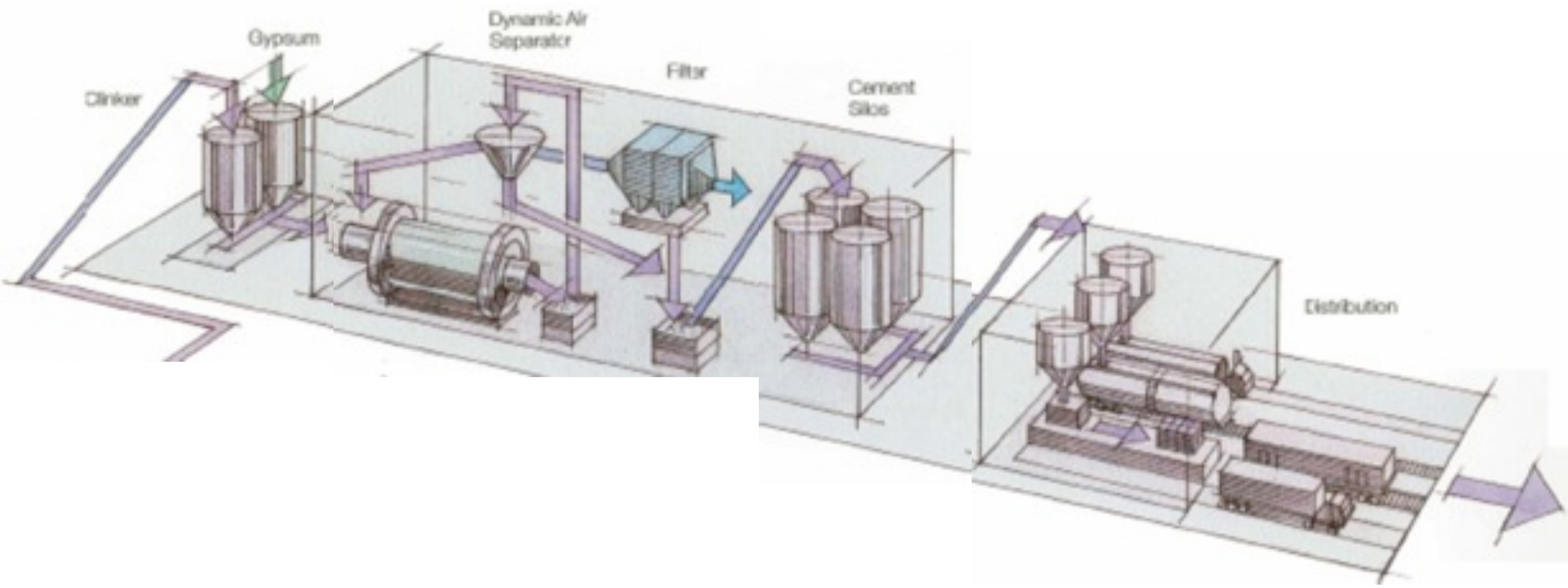
Indexing

```
PARAMETER { REAL N(2); }
CONTINUOUS {
  x = x(N(1:2), 1:3) + u(2*N);
}
```

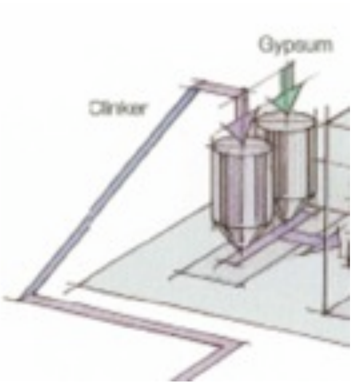
FOR loops

```
FOR (i = 1:N) {
  x(i) = 2*x(N-i+1);
}
```

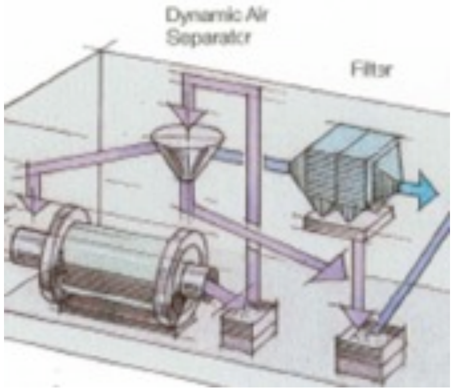
Compositions of Multiple Models



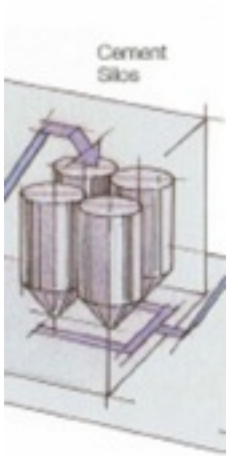
Step 1: Divide the Plant into Subsystems



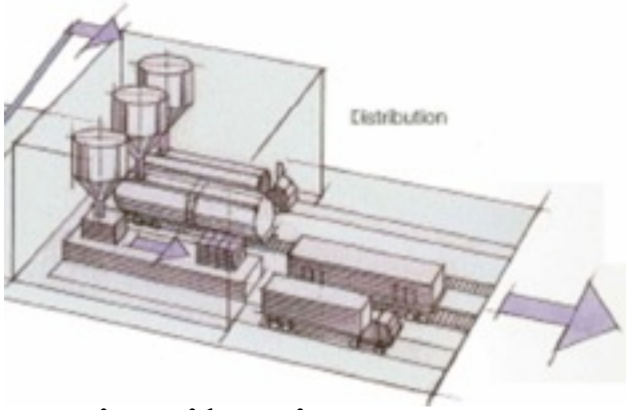
Feeder



Separator

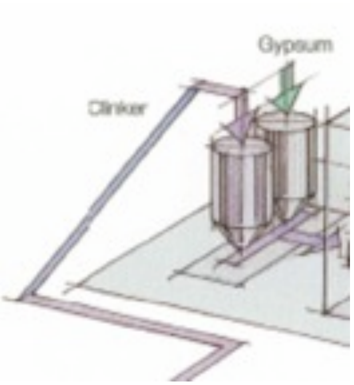


Silos



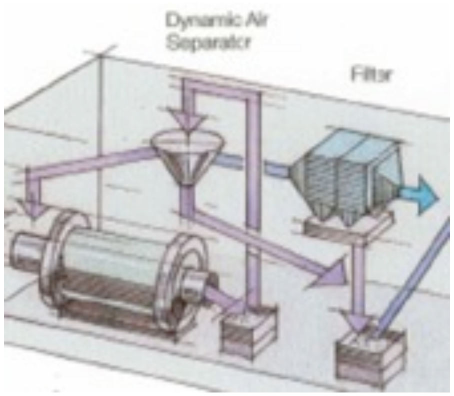
Distribution

Step 2: Create a Model of each Subsystem



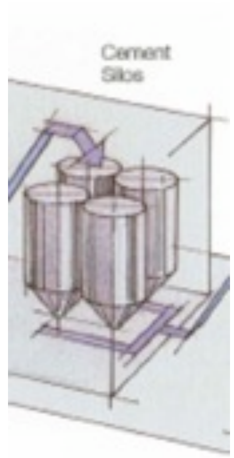
Feeder

feeder.hys



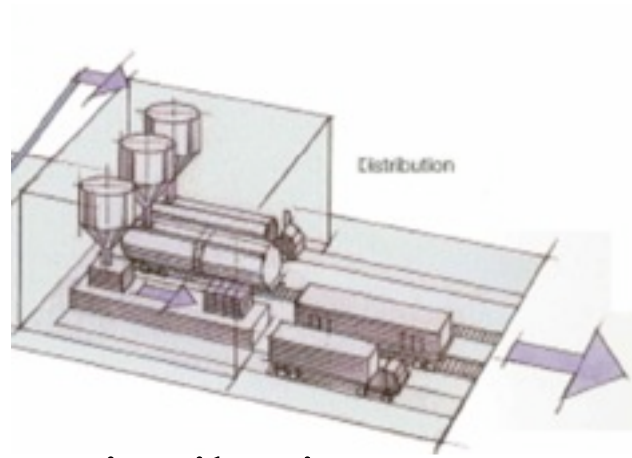
Separator

separator.hys



Silos

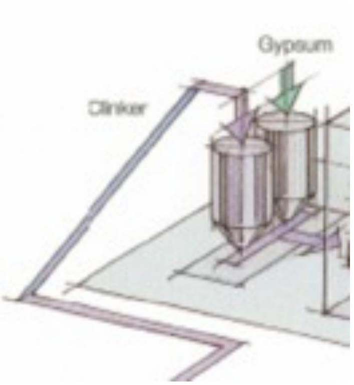
silos.hys



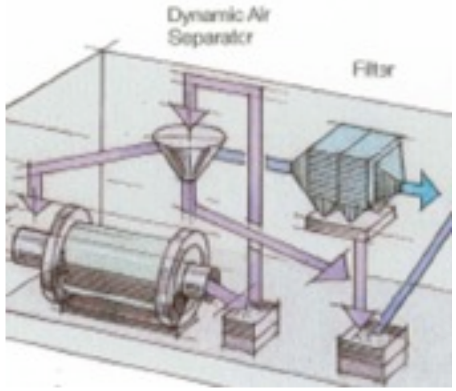
Distribution

distribution.hys

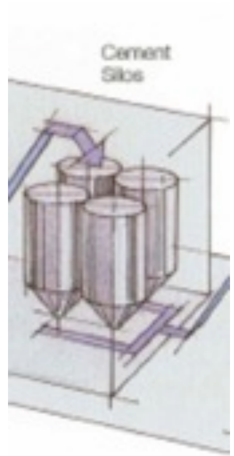
Step 3: Define Interconnections



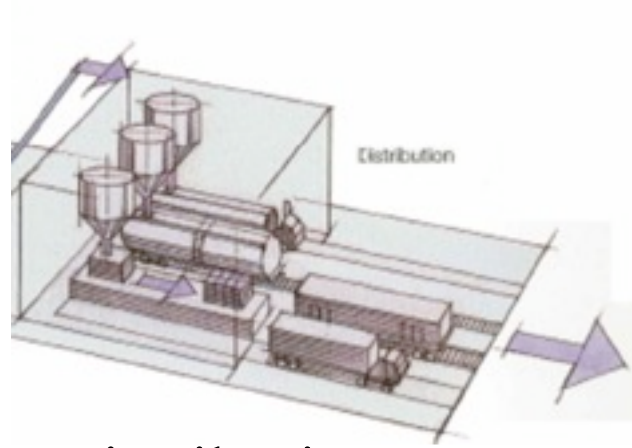
Feeder



Separator



Silos

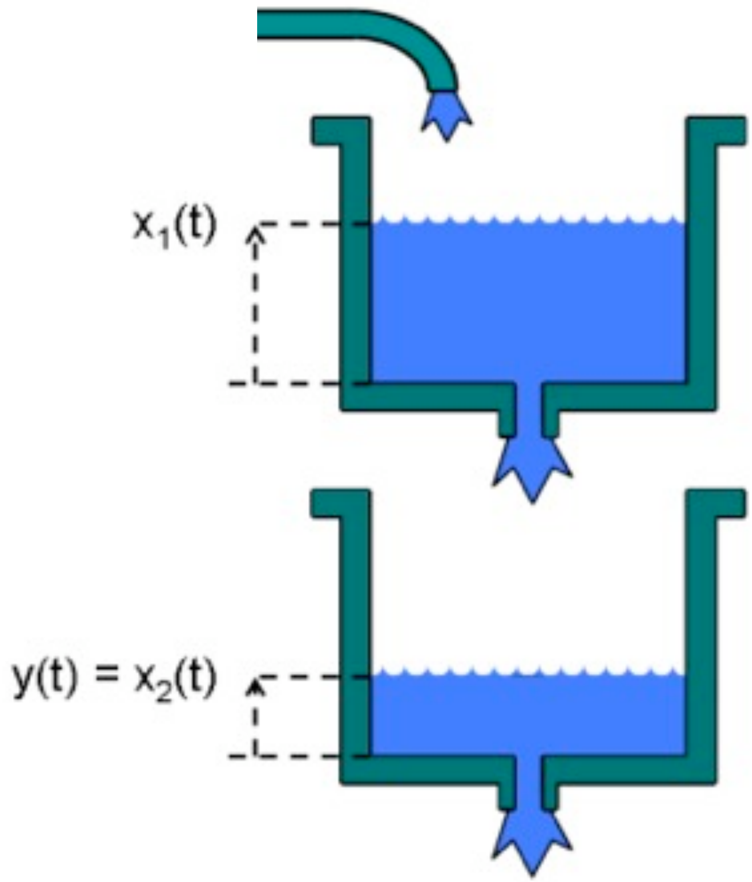


Distribution

```
feeder.output    = separator.input
separator.output = silos.input
silos.output     = distribution.input
```

Example – Two Tank System

```
SYSTEM single_tank {
 INTERFACE {
  STATE { REAL x; }
  INPUT { REAL inflow; }
  OUTPUT { REAL outflow; }
  PARAMETER { REAL k = 0.5;}
 }
 IMPLEMENTATION {
  CONTINUOUS {
   x = inflow - k*x + x;
  }
  OUTPUT {
   outflow = k*x;
  }
 }
}
```



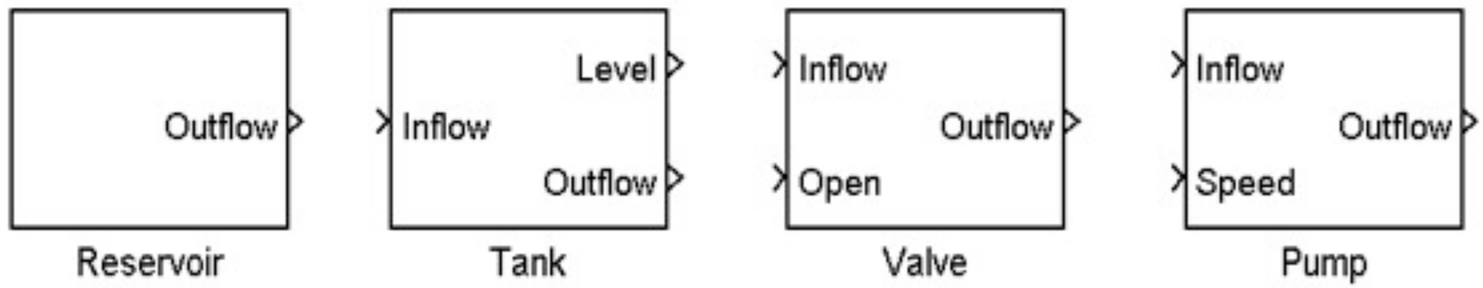
Example – Two Tank System

```
SYSTEM single_tank {
 INTERFACE {
  STATE { REAL x; }
  INPUT { REAL inflow; }
  OUTPUT { REAL outflow; }
  PARAMETER { REAL k = 0.5;}
 }
 IMPLEMENTATION {
  CONTINUOUS {
    x = inflow - k*x + x;
   }
   OUTPUT {
    outflow = k*x;
   }
 }
}
```

```
SYSTEM two_tanks_master {
 INTERFACE {
  MODULE {
    single_tank T1, T2;
   }
   INPUT { REAL inflow; }
 }
 IMPLEMENTATION {
  LINEAR {
    T1.inflow = inflow;
    T1.outflow = T2.inflow;
   }
 }
}
```

Graphical Modeling

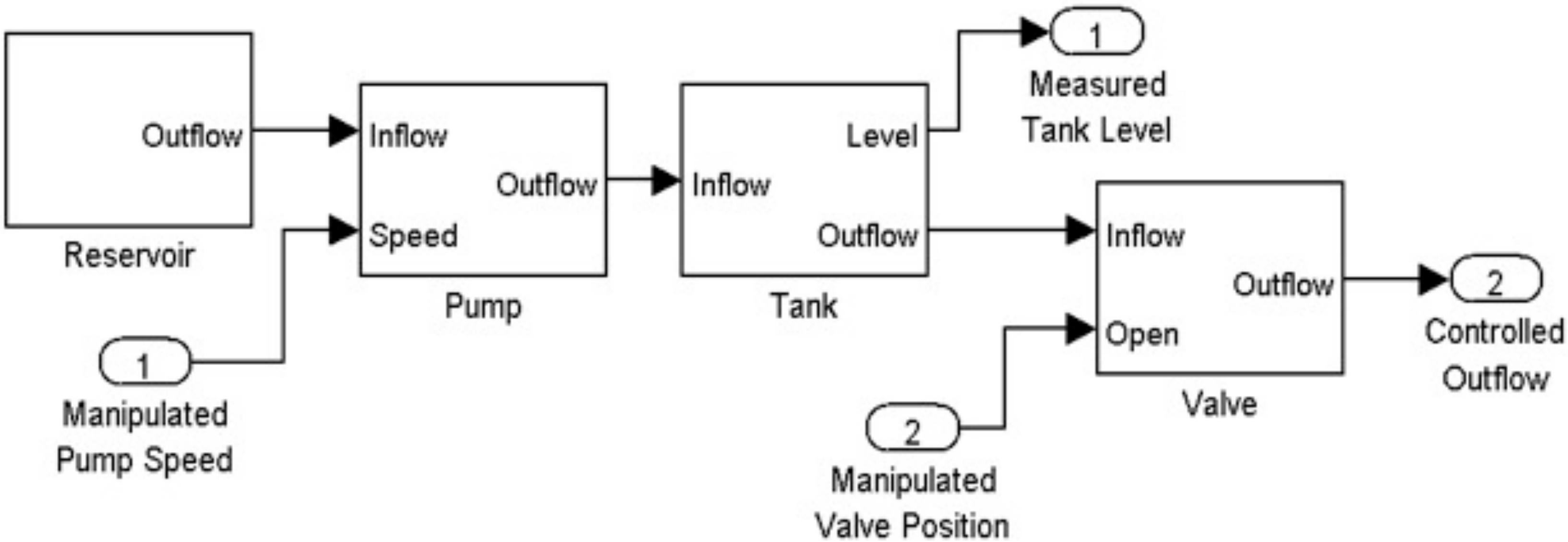
- Library of standard units:



- Dynamical behavior of each block is described by a separate HYSDEL source file
- Parameters of the blocks (e.g. the cross-sectional area of a tank or the volume of the reservoir) can be changed for each block separately

Graphical Modeling

- Blocks are then interconnected:



- HYSDEL 3.0 automatically generates the “master” model which defines:
  – dynamical behavior of each “slave” model
  – interconnections between different “slave” models

HYSDEL 3.0 Compiler

- Written in Matlab
  - cheap to maintain
  - easy to extend
  - OS platform independent
- Uses optimization packages to improve “quality” of the generated models

Importance of Model Quality

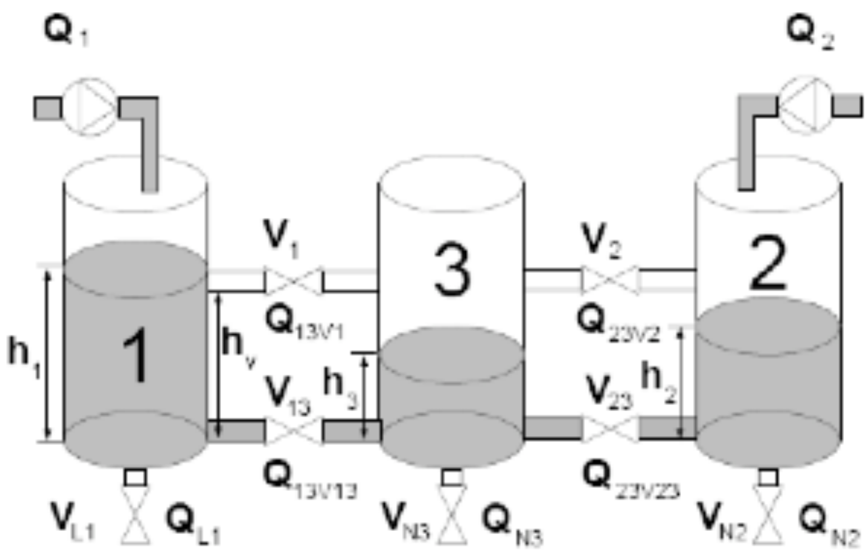
- Model "quality" is related to logic statements:

$$[f(x) \leq 0] \iff [\delta = 1] \text{ iff } \begin{cases} f(x) \leq M(1-\delta) \\ f(x) \geq \epsilon + (m-\epsilon)\delta \end{cases}$$

- Tighter value of M leads problems which can be solved more quickly:

Horizon	M = 50	M = 25	M = 10
7	1 sec	1 sec	1 sec
8	6 secs	5 secs	3 secs
9	265 secs	52 secs	31 secs

Importance of Problem Formulation



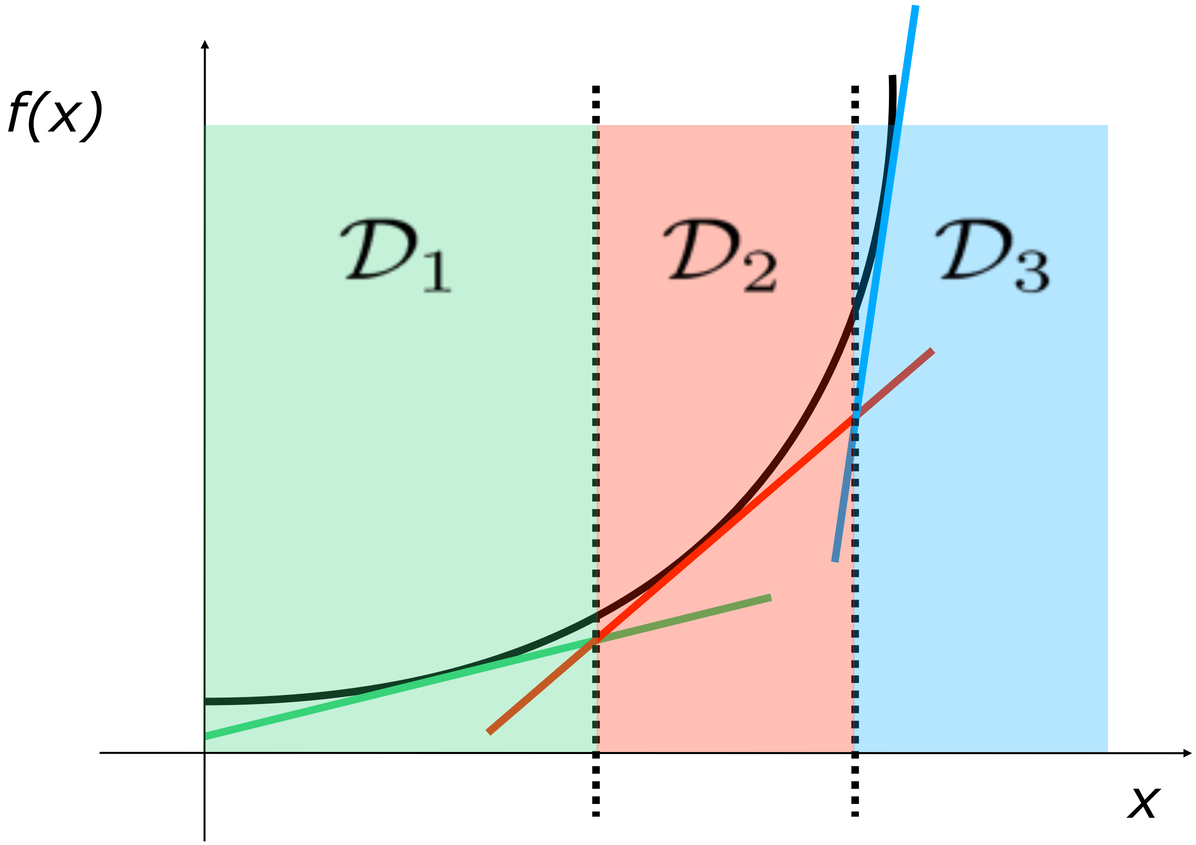
- Drive levels in tanks to desired locations
- Valves can only be open/closed
- Problem formulated as an MILP
- Solved by CPLEX 9.0

Prediction horizon	HYSDEL3 runtime	HYSDEL2 runtime
7	0.1 secs	1 sec
8	1 sec	3 secs
9	9 secs	31 secs
10	50 secs	252 secs

Hybrid Systems Seminar
Part 4: Piecewise Affine Systems

Michal Kvasnica, Alexander Szücs

Piecewise Affine Systems



- Another popular framework for modeling of hybrid systems
- IF-THEN rules translate into an mixed-integer model
- arbitrary precision can be achieved by adding more linearizations

$$x_{k+1} = A_i x_k + B_i u_k + f_i \quad \text{IF} \quad x_k \in \mathcal{D}_i$$

PWA vs MLD Models

- MLD: natural for systems including finite state automata and logic expressions
- PWA: ideal for approximating nonlinear functions
- Main message: under mild assumptions one can convert from MLD to PWA representation and vice versa
- MPT includes MLD-to-PWA and PWA-to-MLD translations

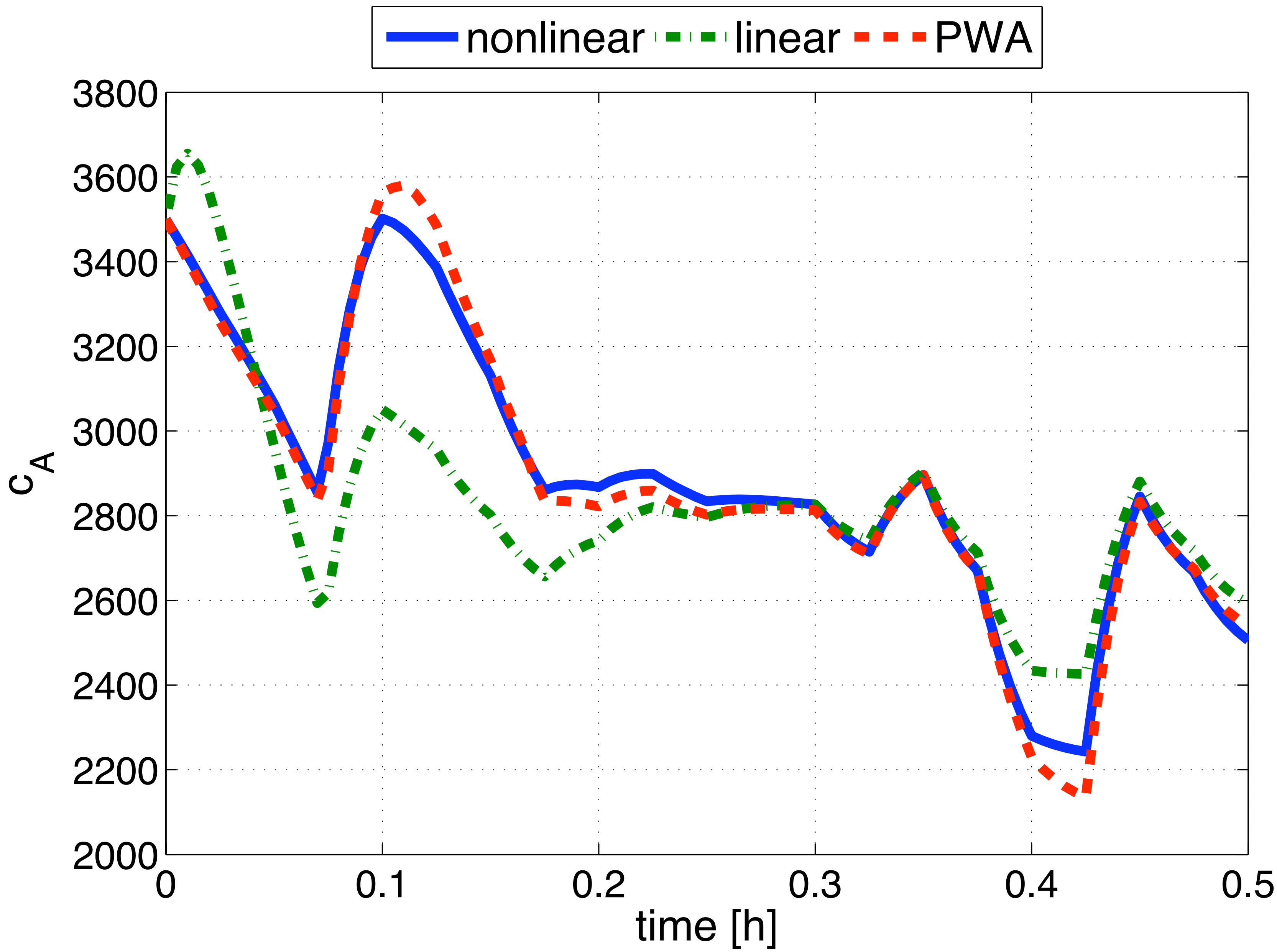
Case Study: CSTR

- Nasty nonlinear dynamics

$$\dot{x} = \begin{bmatrix} -k_1(T)c_A - k_2(T)c_A^2 + (c_{in} - c_A)u_1 \\ k_1(T)(c_A - c_B) - c_B u_1 \\ h(c_A, c_B, T) + (T_c - T)\alpha + (T_{in} - T)u_1 \\ (T - T_c)\beta + \gamma u_2 \end{bmatrix}$$

- Constraints on states and inputs
- Approximated by a PWA system with 32 local linearizations

Case Study: CSTR



Mathematical Formulation

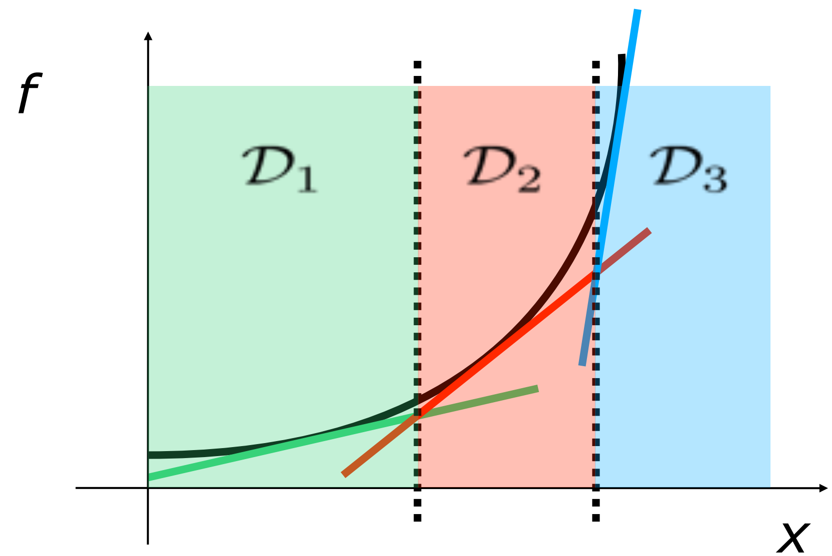
$$\boxed{x_{k+1} = A_i x_k + B_i u_k + f_i \quad \text{IF} \quad x_k \in \mathcal{D}_i}$$

- Key assumptions:
  - each dynamics is valid over a polytopic region $\mathcal{D}_i = \{x_k \mid D_i^x x_k \leq D_i^0\}$
  - the regions do not overlap, i.e. $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$
- Associate one binary selector per one region: $(\delta_i = 1) \Leftrightarrow (x_k \in \mathcal{D}_i)$
- Conversion to mixed-integer inequalities: $D_i^x x_k - D_i^0 \leq M(1 - \delta_i)$
- Add an exclusive-or condition: $\sum \delta_i = 1$
- Finally add:

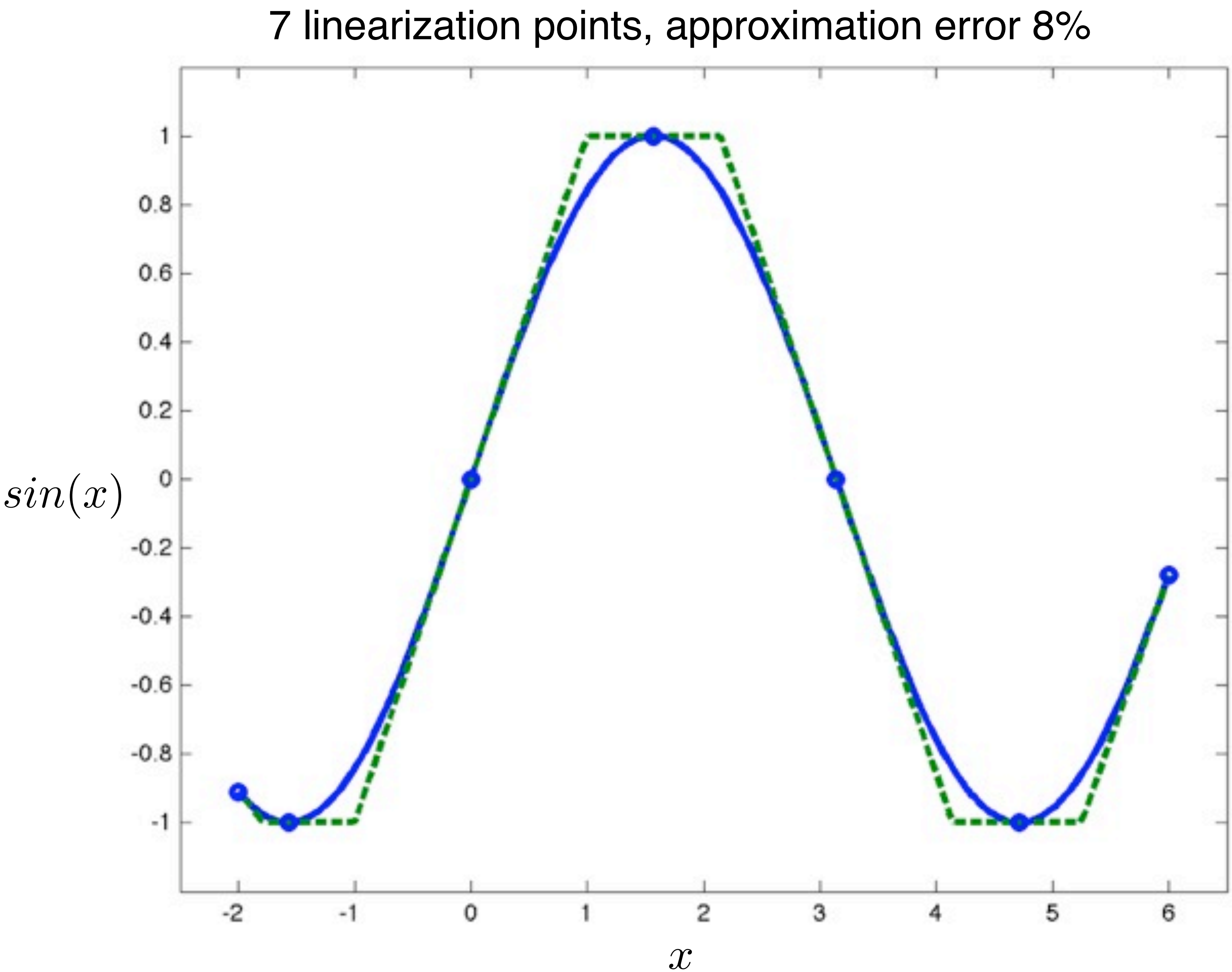
$$x_{k+1} \leq M(1 - \delta_i) + (A_i x_k + B_i u_k + f_i)$$
$$x_{k+1} \geq m(1 - \delta_i) + (A_i x_k + B_i u_k + f_i)$$

Obtaining PWA Models

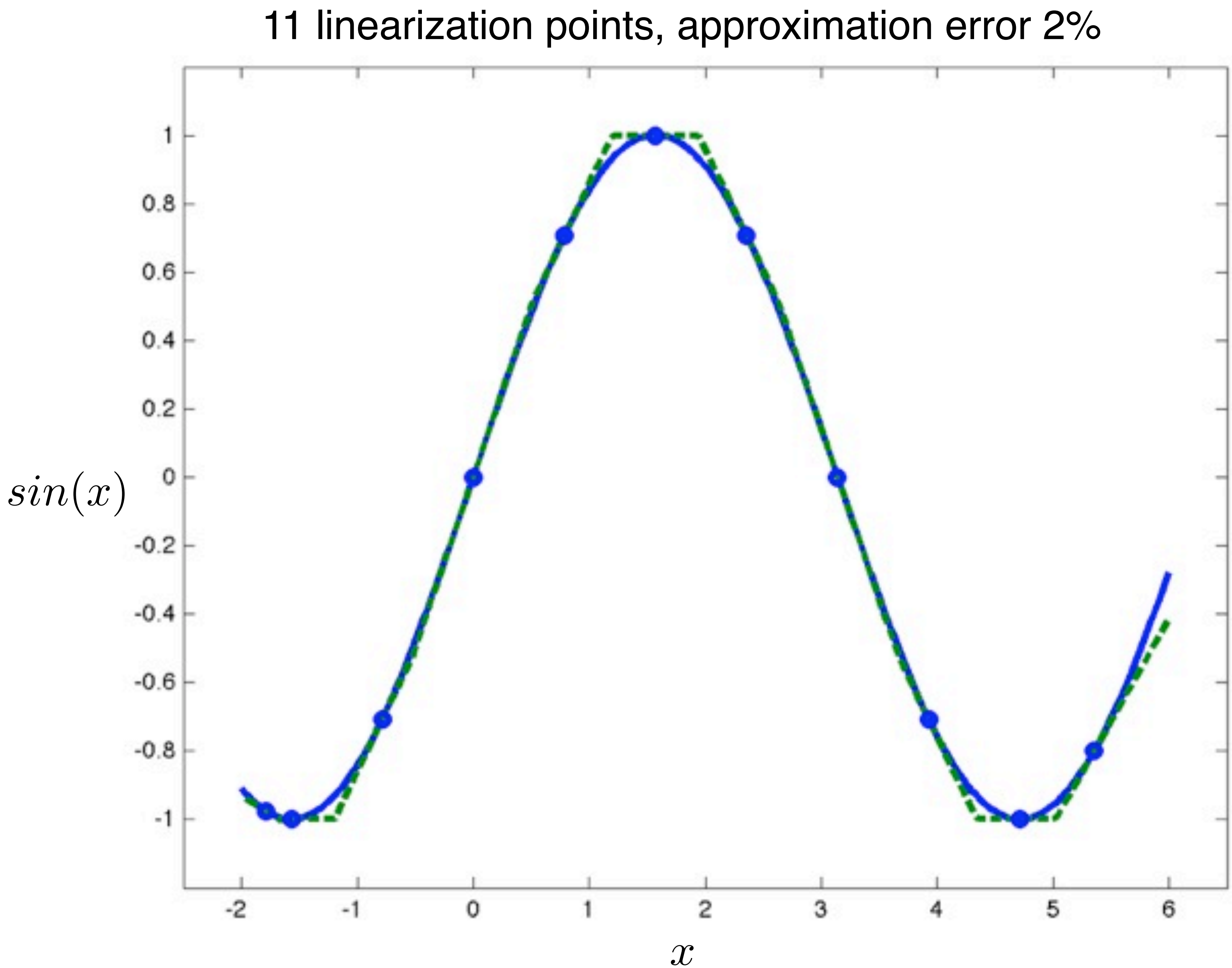
- The process of obtaining a PWA approximation of a nonlinear function includes:
  - selection of suitable linearization points
  - calculation of corresponding local linearization
  - determination of regions of validity
- Bottom line: easy to do hand in 1D, difficult in 2D, impossible in higher dimensions
- Question: can the process be automated?



Automatic Multiple Linearization of 1D Functions

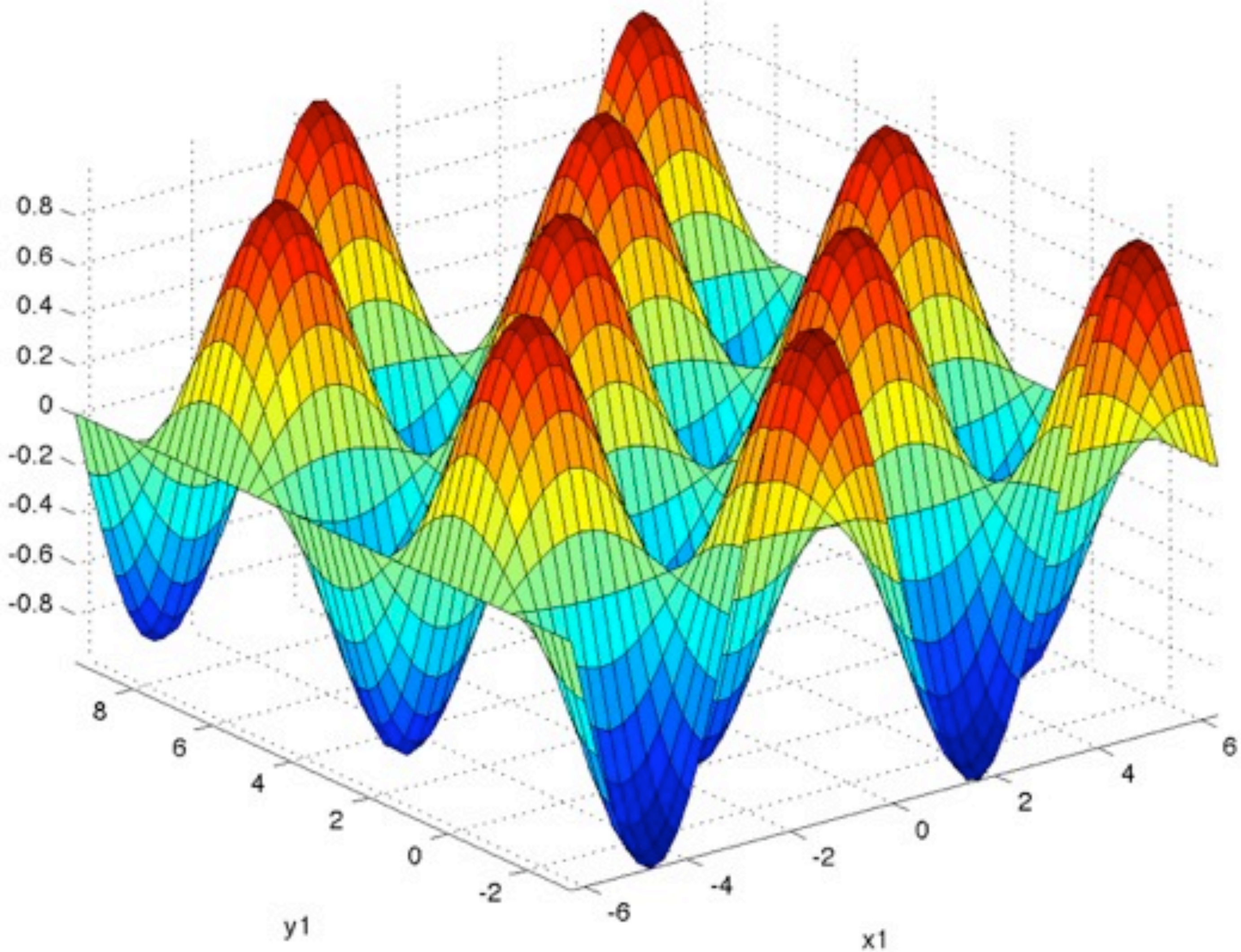


Automatic Multiple Linearization of 1D Functions



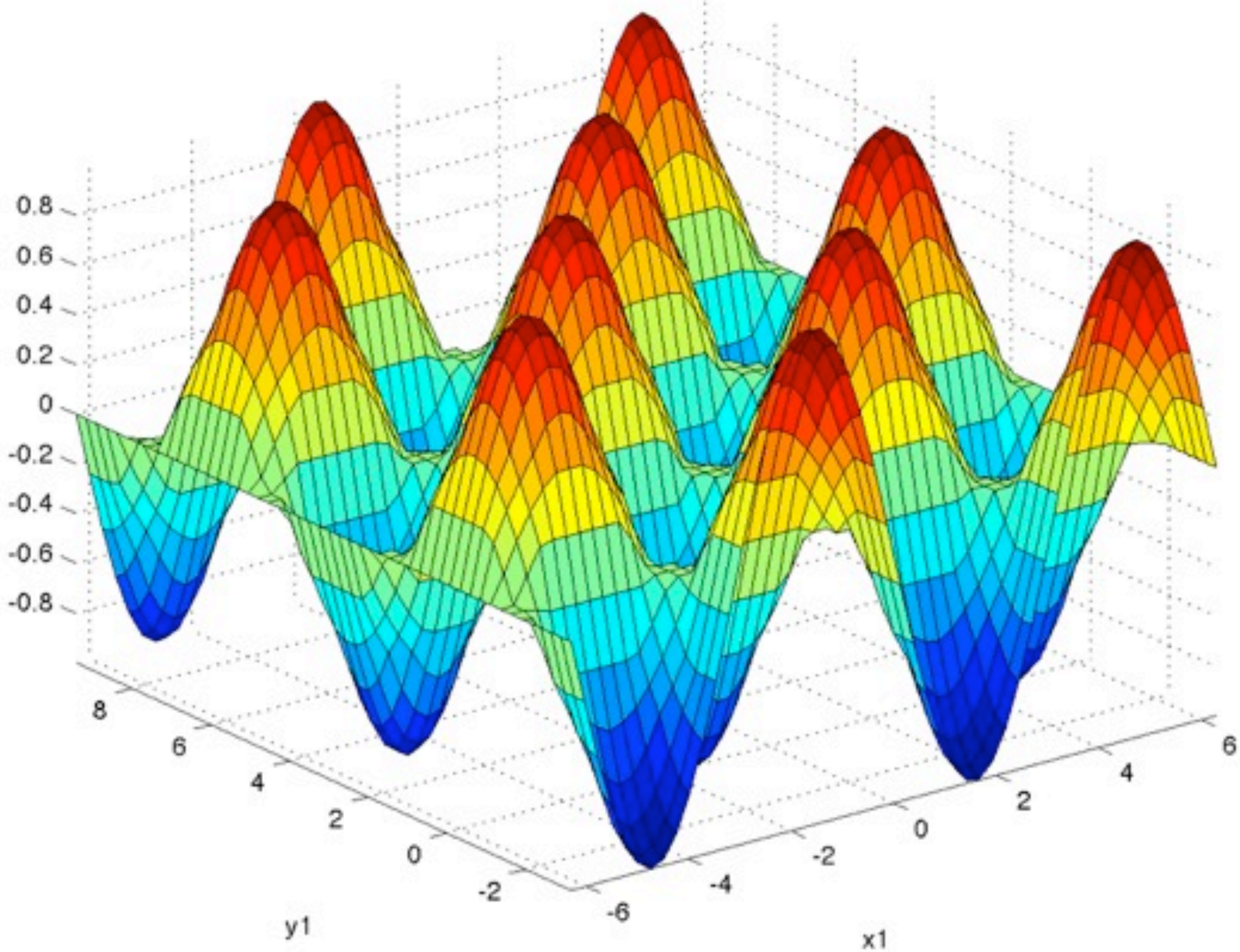
Automatic Multiple Linearization of 2D Functions

$$f(x_1, x_2) = \sin(x_1)\cos(x_2)$$



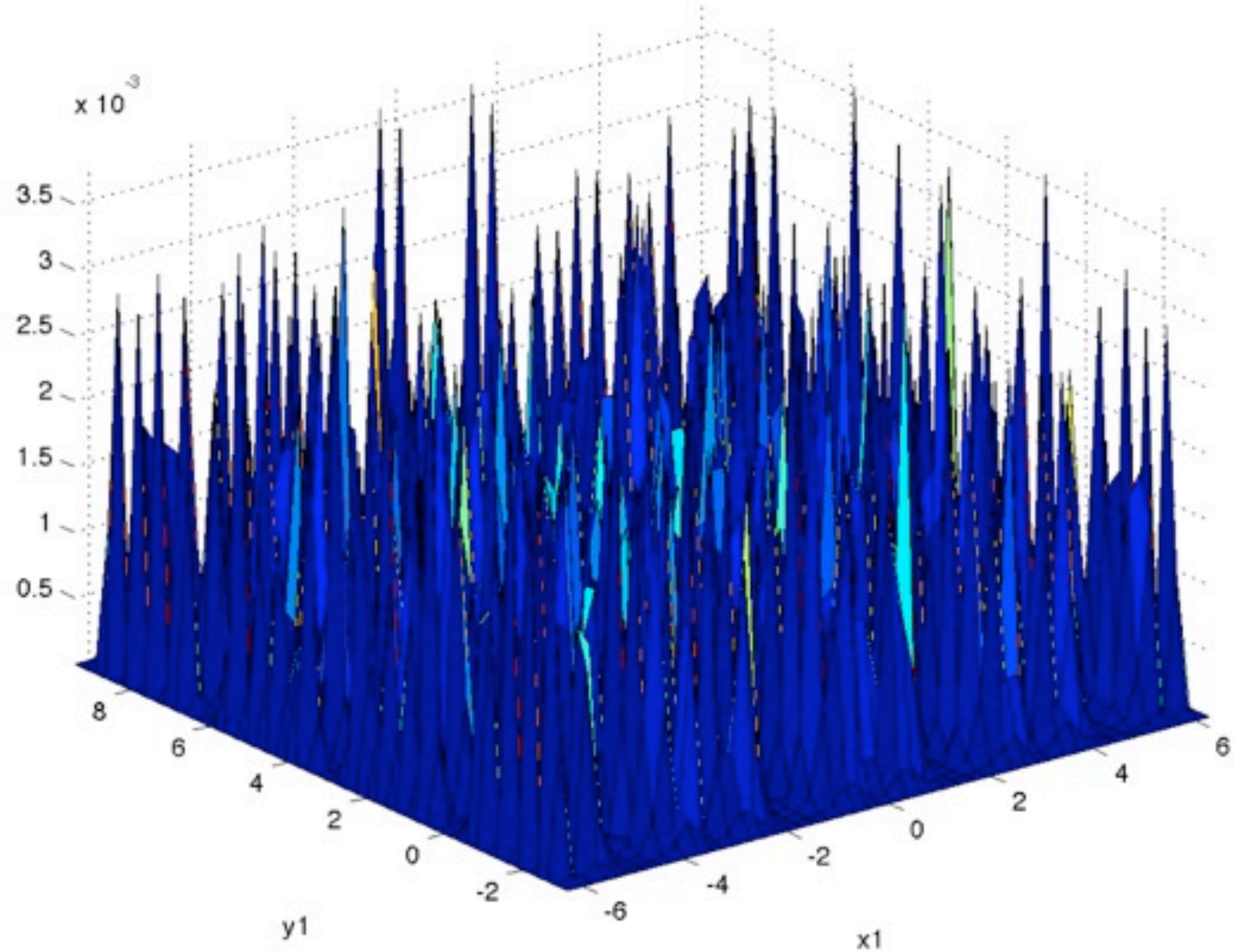
Automatic Multiple Linearization of 2D Functions

PWA approximation using 10 linearizations



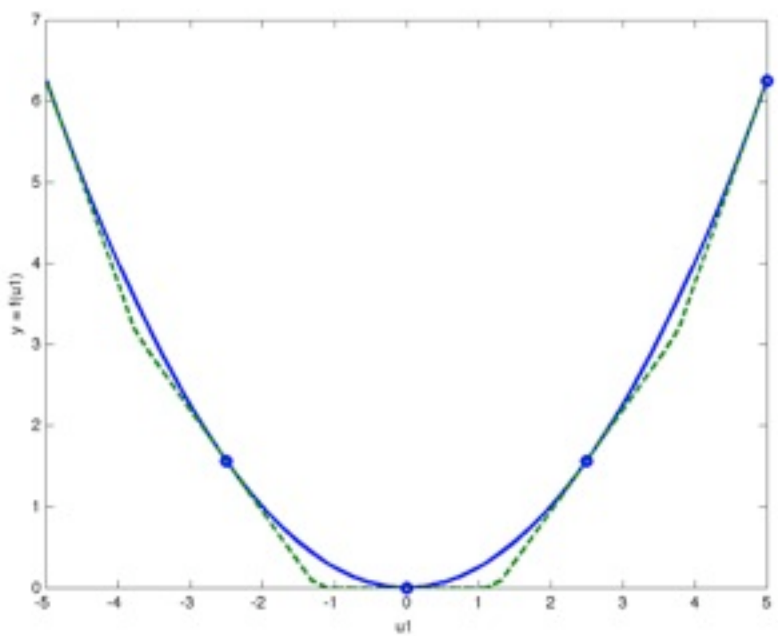
Automatic Multiple Linearization of 2D Functions

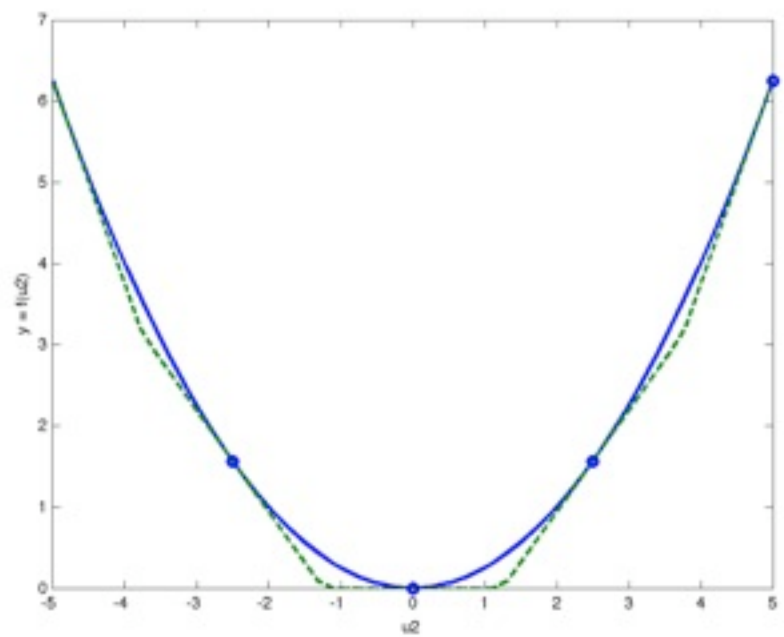
Approximation error < 0.1 %



The Theory Behind

- Consider a product of two variables $f = x_1 x_2$
- Define two auxiliary variables $u_1 = (x_1 + x_2),\ u_2 = (x_1 - x_2)$
- Observe the equivalence: $f = \frac{1}{4}(u_1^2 - u_2^2)$
- Now we have a difference of two nonlinear 1D functions, hence we are back to the 1D scenario

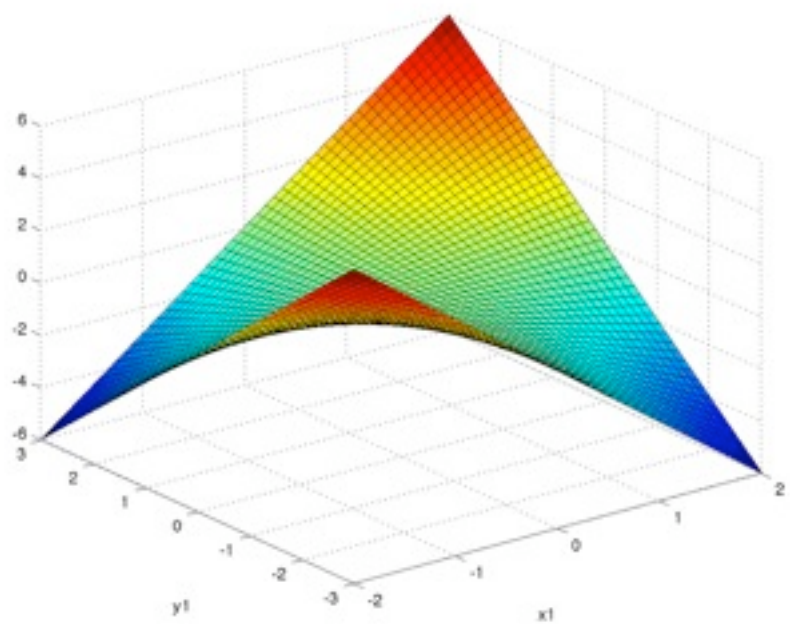


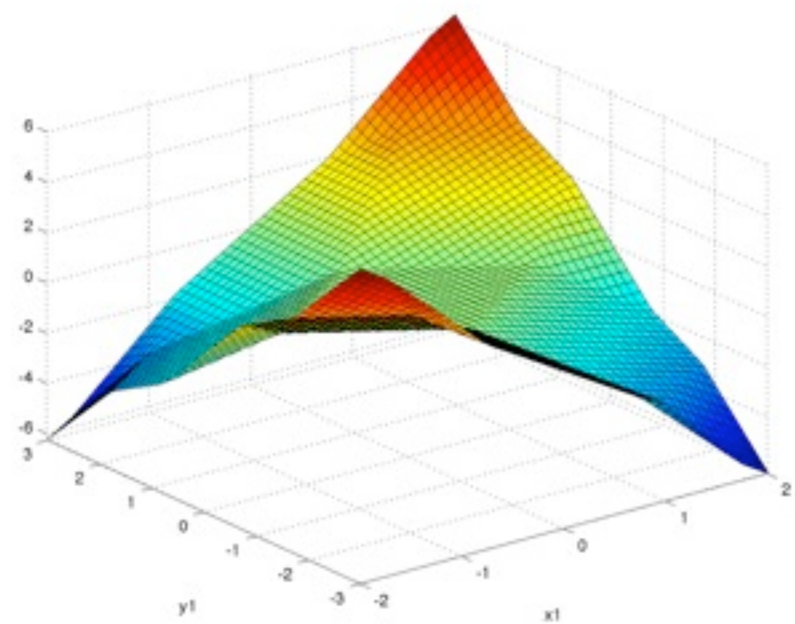


Williams: *Model Building in Mathematical Programming*, Wiley, 1993

The Theory Behind

- Consider a product of two variables $f = x_1 x_2$
- Define two auxiliary variables $u_1 = (x_1 + x_2),\ u_2 = (x_1 - x_2)$ (1)
- Observe the equivalence: $f = \frac{1}{4}(u_1^2 - u_2^2)$ (2)
- Now we have a difference of two nonlinear 1D functions, hence we are back to the 1D scenario (3)
- The overall model is composed of (1), (2) and (3)





PWA Approximation Toolbox

- Based on the Symbolic Toolbox
- Inputs:
  - symbolic representation of an arbitrary nonlinear function, e.g.

$$\sin(x_1^2 + \exp(1/x_2))(x_3 - \cos(|x_4|))$$

  - lower/upper bounds on variables
  - number of linearization points
- Outputs:
  - individual linearizations
  - regions of validity
  - direct export to HYSDEL is work in progress

Hybrid Systems Seminar
Part 5: MPC for Hybrid Systems

Michal Kvasnica

Mixed Logical Dynamical (MLD) Models

- Compact mathematical representation of hybrid systems

$$\boxed{\begin{aligned} &x(t+1) = Ax(t) + B_u u(t) + B_\delta \delta(t) + B_z z(t) \\ &y(t) = Cx(t) + D_u u(t) + D_\delta \delta(t) + D_z z(t) \\ &E_x x(t) + E_u u(t) + E_\delta \delta(t) + E_z z(t) \leq E_0 \end{aligned}}$$

- Involves continuous and binary states, inputs, outputs
- Auxiliary variables:
  - binary selectors $\delta(t)$
  - continuous variables $z(t)$
- Mixed-integer linear constraints:
  - include physical constraints on state, inputs, outputs
  - capture events, FSM, mode selection

MPC Formulation for MLD Models

$$
\begin{aligned}
\min \quad & \sum_{k=0}^{N-1} \left(\|Q_x x_{t+k}\|_p + \|Q_u u_{t+k}\|_p\right) \\
\text{s.t.} \quad & x_{t+k+1} = A x_{t+k} + B_u u_{t+k} + B_\delta \delta_{t+k} + B_z z_{t+k} \\
& E_x x_{t+k} + E_u u_{t+k} + E_\delta \delta_{t+k} + E_z z_{t+k} \le E_0 \\
& x_{t+k} \in \mathcal{X} \\
& u_{t+k} \in \mathcal{U} \\
& x_t = x(t) \\
& \delta_{t+k} \in \{0,1\}^{n_\delta}, \;\; z_{t+k} \in \mathbb{R}^{n_z}
\end{aligned}
$$

- The optimization problem is no longer convex!
  - mixed-integer QP for $p = 2$
  - mixed-integer LP for $p = \{1, \infty\}$
- Can still be solved in “reasonable” time (CPLEX, GLPK)

Piecewise Affine (PWA) Models

$$\boxed{x_{k+1} = A_i x_k + B_i u_k + f_i \quad \text{IF} \quad x_k \in \mathcal{D}_i}$$

- Key assumptions:
  - each dynamics is valid over a polytopic region $\mathcal{D}_i = \{x_k \mid D_i^x x_k \leq D_i^0\}$
  - the regions do not overlap, i.e. $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$
- Associate one binary selector per one region: $(\delta_i = 1) \Leftrightarrow (x_k \in \mathcal{D}_i)$
- Conversion to mixed-integer inequalities: $D_i^x x_k - D_i^0 \leq M(1 - \delta_i)$
- Add an exclusive-or condition: $\sum \delta_i = 1$
- Finally add:

$$x_{k+1} \leq M(1 - \delta_i) + (A_i x_k + B_i u_k + f_i)$$
$$x_{k+1} \geq m(1 - \delta_i) + (A_i x_k + B_i u_k + f_i)$$

MPC Formulation for PWA Models

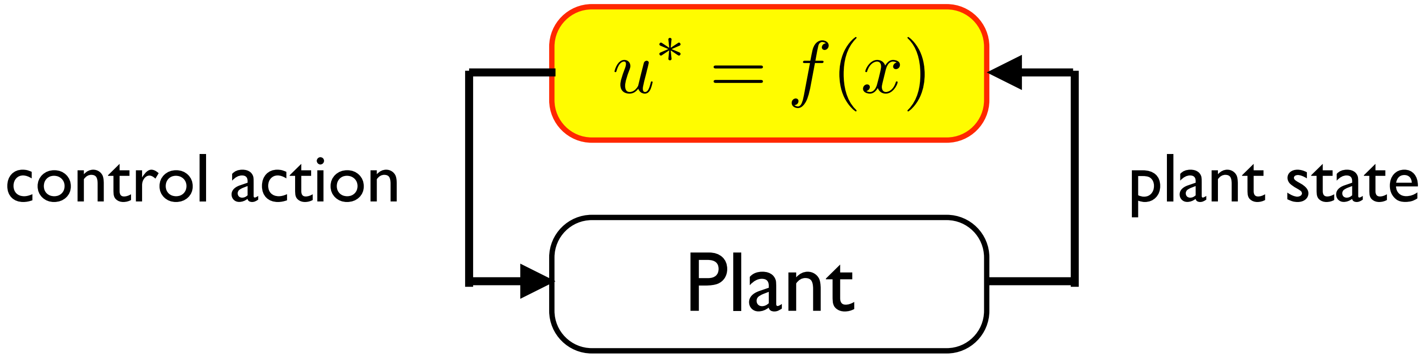
$$
\begin{aligned}
\min \quad & \sum_{k=0}^{N-1} \left(\|Q_x x_{t+k}\|_p + \|Q_u u_{t+k}\|_p \right) \\
\text{s.t.} \quad & x_{t+k+1} \leq M(1 - \delta_{t+k,i}) + (A_i x_{t+k} + B_i u_{t+k} + f_i) \\
& x_{t+k+1} \geq M(1 - \delta_{t+k,i}) + (A_i x_{t+k} + B_i u_{t+k} + f_i) \\
& D_i^x x_{t+k} - D_i^0 \leq M(1 - \delta_{t+k,i}) \\
& \sum \delta_{t+k,i} = 1 \\
& x_{t+k} \in \mathcal{X} \\
& u_{t+k} \in \mathcal{U} \\
& x_t = x(t) \\
& \delta_{t+k,i} \in \{0, 1\}
\end{aligned}
$$

- Also non-convex, leads to MILP or MIQP problems

Hybrid Systems Seminar
Part 6: Explicit Model Predictive Control

Michal Kvasnica

Model Predictive Control



Given a performance index $J_N = \sum_{k=0}^{N-1} u_k^T R u_k + x_k^T Q x_k$

$\boxed{\text{Compute control action } u^* = f(x) \text{ in acceptable time}}$

$$\begin{aligned} u^* &= \arg\min J_N \\ &\quad \text{Plant model} \\ &\quad \text{Constraints} \end{aligned}$$

MPC Formulation

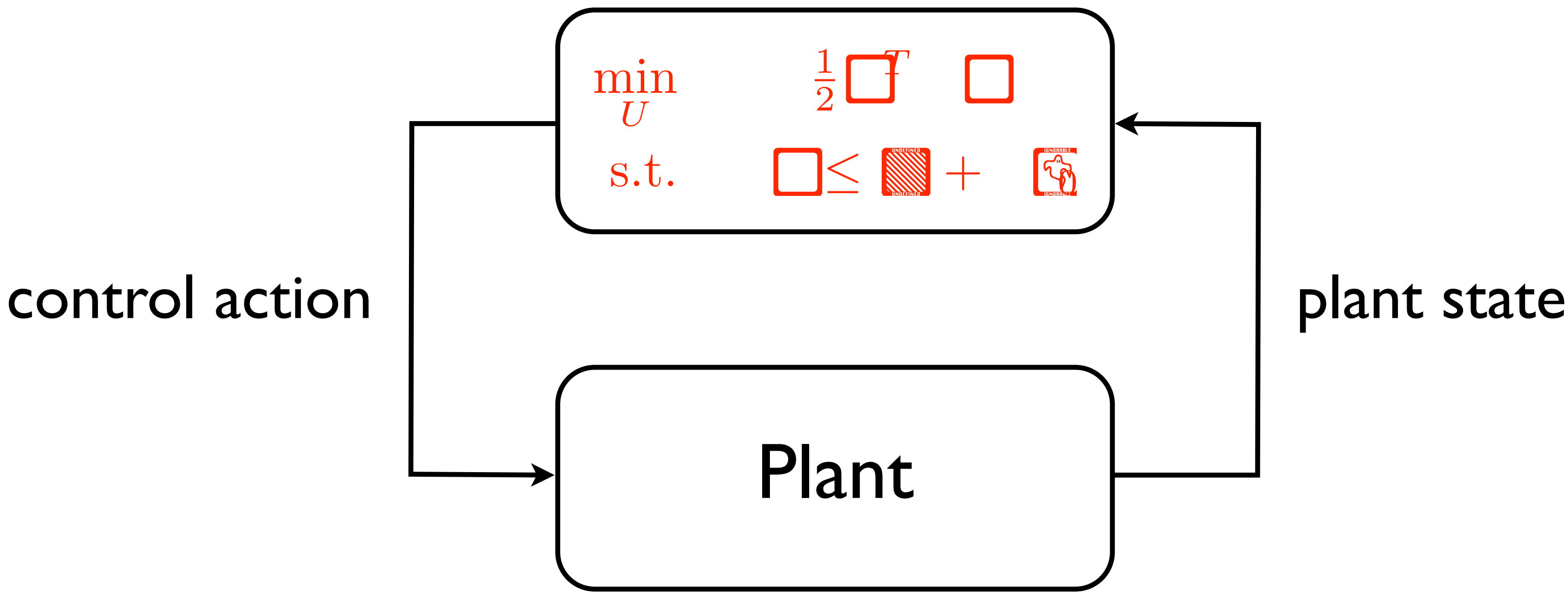
$$\begin{aligned} \min_{U=[u_0,\ldots,u_{N-1}]} \quad & \sum_{k=0}^{N-1} u_k^T R u_k + x_k^T Q x_k \\ \text{s.t.} \quad & x_k \in \mathcal{X} \\ & u_k \in \mathcal{U} \\ & x_{k+1} = f(x_k, u_k) \end{aligned}$$

$$\begin{aligned} x_1 &= Ax_0 + Bu_0 \\ x_2 &= Ax_1 + Bu_1 \\ &= A^2 x_0 + ABu_0 + Bu_1 \\ x_3 &= Ax_2 + Bu_2 \\ &= A^3 x_0 + A^2 Bu_0 + ABu_1 + Bu_2 \\ &\vdots \end{aligned}$$

$$\boxed{\begin{aligned} \min_{U} \quad & \tfrac{1}{2} U^T H U \\ \text{s.t.} \quad & GU \leq W + Sx_0 \end{aligned}}$$

Parameters (initial condition)

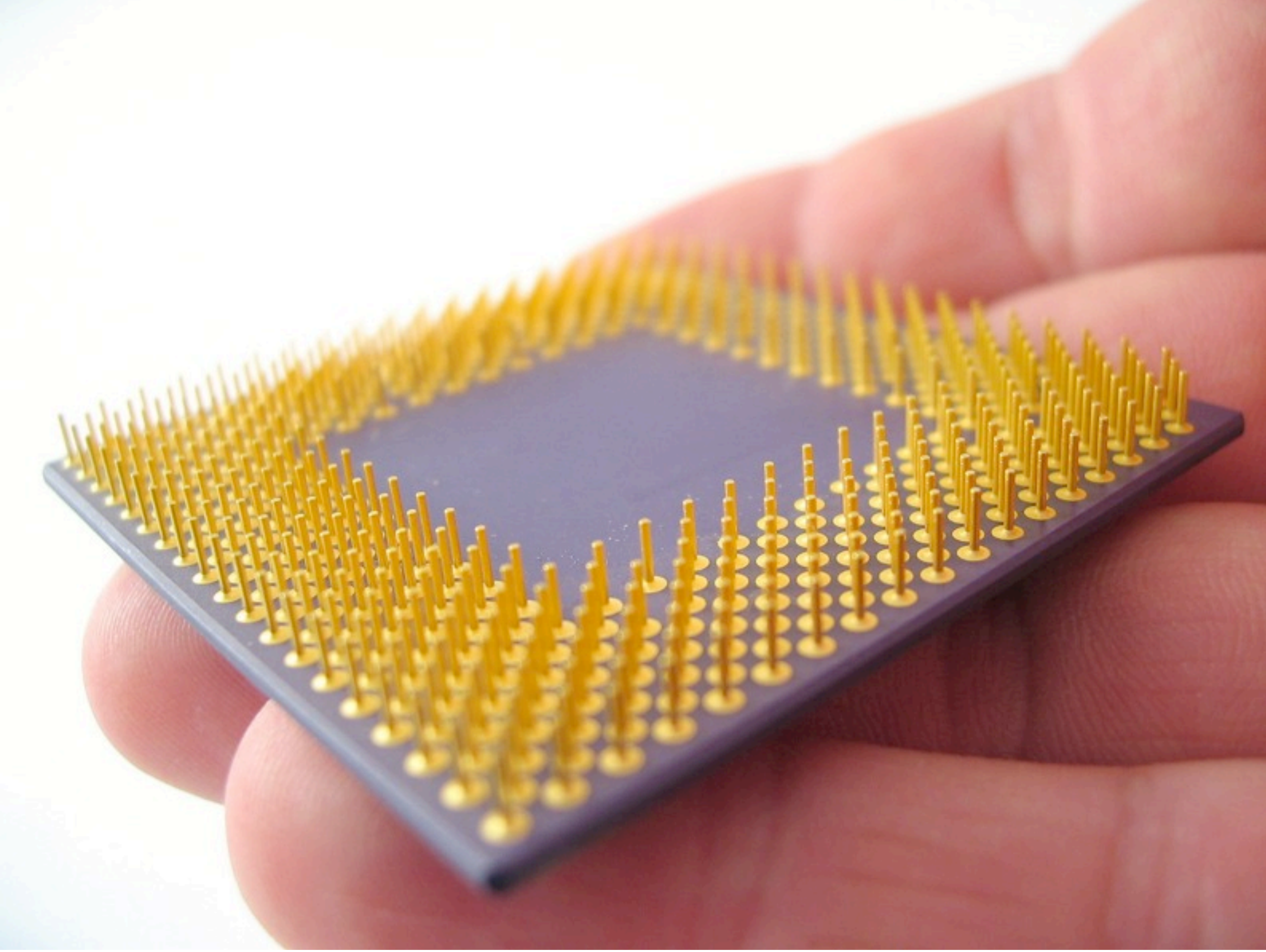
On-Line MPC

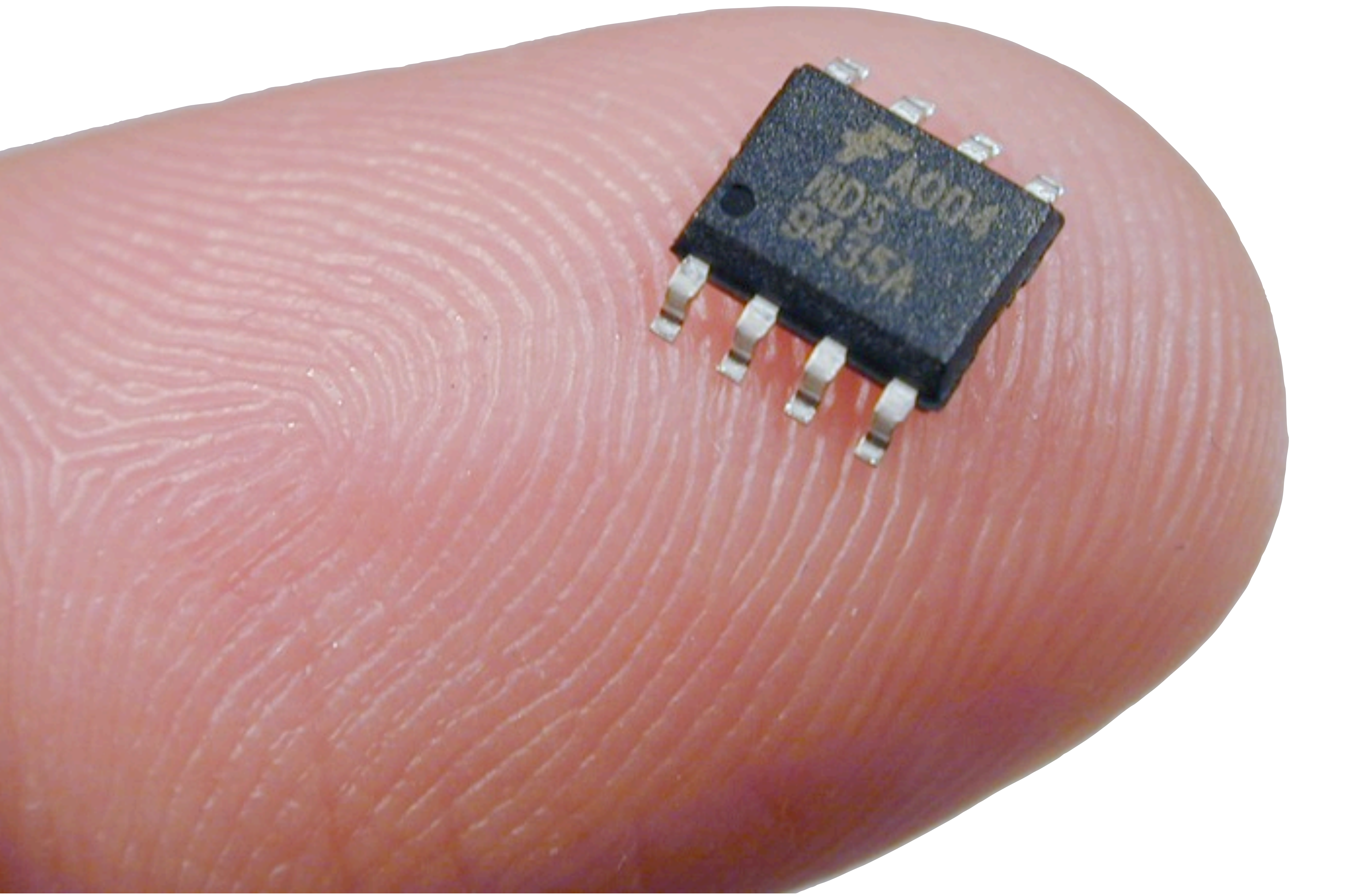


On-Line MPC: Properties

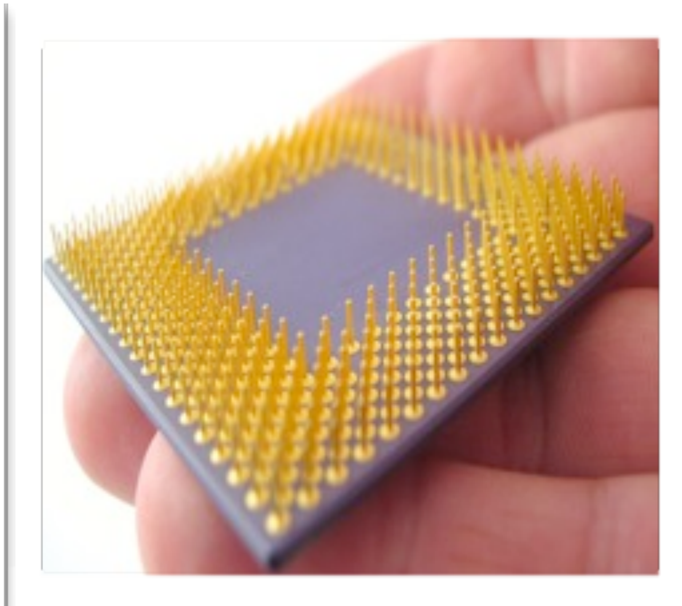
Constraints	✓
Optimal performance	✓
Fast implementation	✘

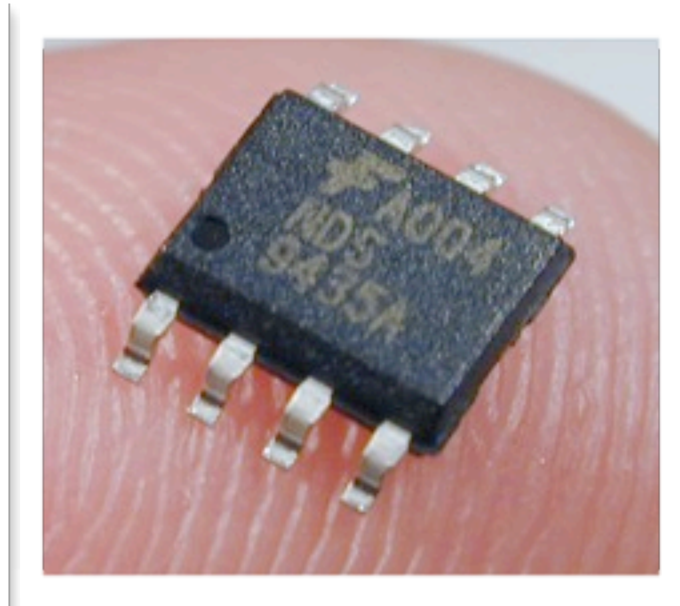

ZENITH
ZENITH




A004
9435A

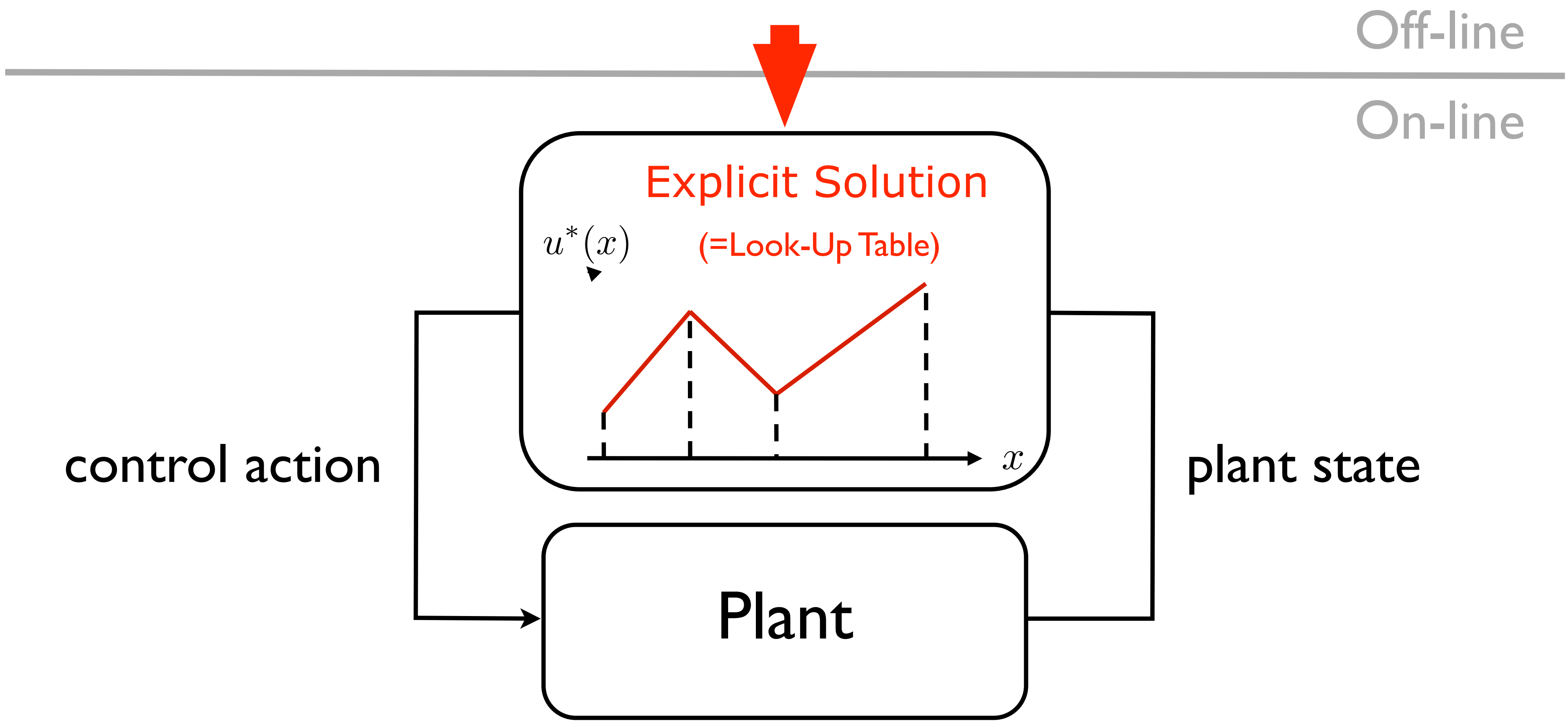
Where is the Problem?





10 000 MFLOPS/sec	100 MFLOPS/sec	1 MFLOPS/sec
more than 2 GB	more than 128 MB	less than 8 kB

Off-Line MPC


Off-line
On-line
Explicit Solution
(=Look-Up Table)
$u^*(x)$
x
control action
plant state
Plant

Multi-Parametric Programming

$$\boxed{\begin{array}{cc} \min\limits_{U} & \frac{1}{2}U^T H U \\ \text{s.t.} & GU \leq W + Sx_0 \end{array}}$$

Karush-Kuhn-Tucker (KKT) optimality conditions

$$\begin{aligned} HU^* + G^T\lambda^* &= 0 \\ \lambda_i^*(G_i U^* - W_i - S_i x_0) &= 0 \\ \lambda_i^* &\geq 0 \end{aligned}$$

Active constraints: $G_i U^* - W_i - S_i x_0 = 0, \quad \lambda_i^* > 0$

Inactive constraints: $G_i U^* - W_i - S_i x_0 < 0, \quad \lambda_i^* = 0$

Multi-Parametric Programming

1. Find local expression for $U^*(x_0)$

$$\begin{array}{ll} \min\limits_{U} & \frac{1}{2}U^T H U \\ \text{s.t.} & GU \leq W + Sx_0 \end{array}$$

- Pick some feasible x_0
- Solve the QP to find $U^*, \ \lambda^*$
- KKT conditions for active constraints:

$$\begin{aligned} HU^* + \widehat{G}^T \widehat{\lambda}^* &= 0 \qquad (1) \\ \widehat{G}U^* - \widehat{W} - \widehat{S}x_0 &= 0 \qquad (2) \end{aligned}$$

From (1): $U^* = -H^{-1}\widehat{G}^T\widehat{\lambda}^*$

From (2): $\widehat{\lambda}^*(x_0) = -(\widehat{G}H^{-1}\widehat{G}^T)^{-1}(\widehat{W} + \widehat{S}x_0)$

$$U^*(x_0) = H^{-1}\widehat{G}^T(\widehat{G}H^{-1}\widehat{G}^T)^{-1}(\widehat{W} + \widehat{S}x_0)$$

Multi-Parametric Programming

1. Find local expression for $U^*(x_0)$

$$\begin{array}{ll} \min\limits_{U} & \frac{1}{2}U^T H U \\ \text{s.t.} & GU \leq W + Sx_0 \end{array}$$

$$\begin{aligned} U^*(x_0) &= \boxed{H^{-1}\widehat{G}^T(\widehat{G}H^{-1}\widehat{G}^T)^{-1}}(\boxed{\widehat{W}} + \boxed{\widehat{S}x_0}) \\ &= Kx_0 + L \\ \widehat{\lambda}^*(x_0) &= -(\widehat{G}H^{-1}\widehat{G}^T)^{-1}(\widehat{W} + \widehat{S}x_0) \\ &= Mx_0 + N \end{aligned}$$

In some neighborhood of x_0, the optimizer is an affine function of the initial condition

Multi-Parametric Programming

2. Find the region of validity

$$\begin{array}{ll} \min\limits_{U} & \frac{1}{2}U^T H U \\ \text{s.t.} & GU \le W + Sx_0 \end{array}$$

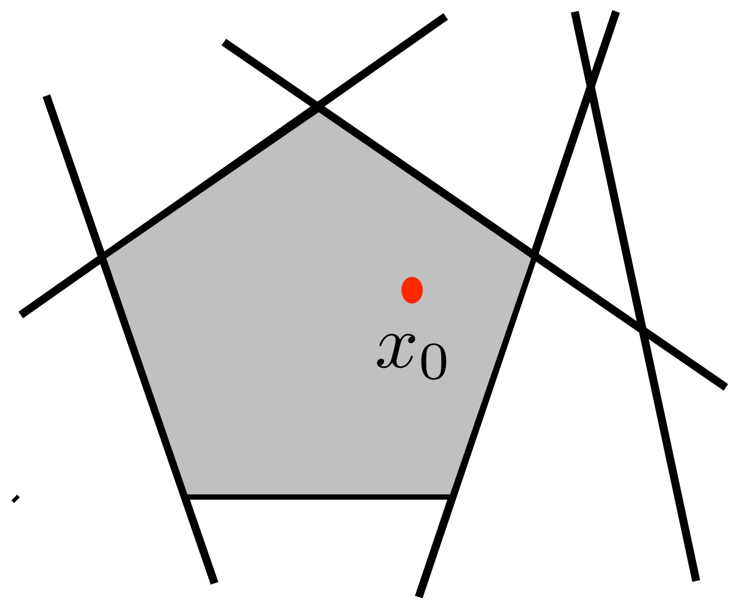
Substitute $U^*(x_0)$ and $\lambda^*(x_0)$ into

$$G\,U \le W + Sx_0$$

$$\lambda \ge 0$$

Polytopic critical region

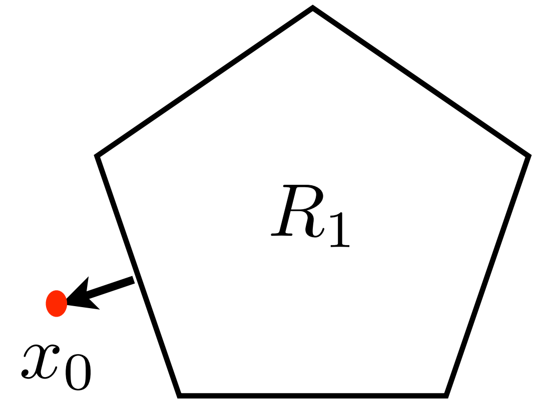
$$R = \{x_0 \mid Ax_0 \le b\}$$



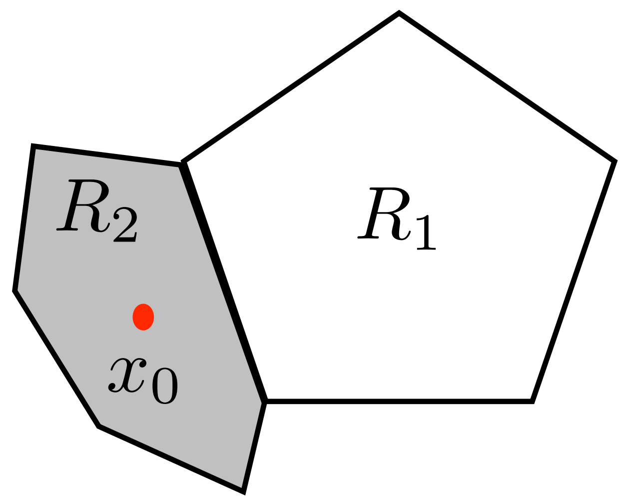
Multi-Parametric Programming

3. Proceed iteratively

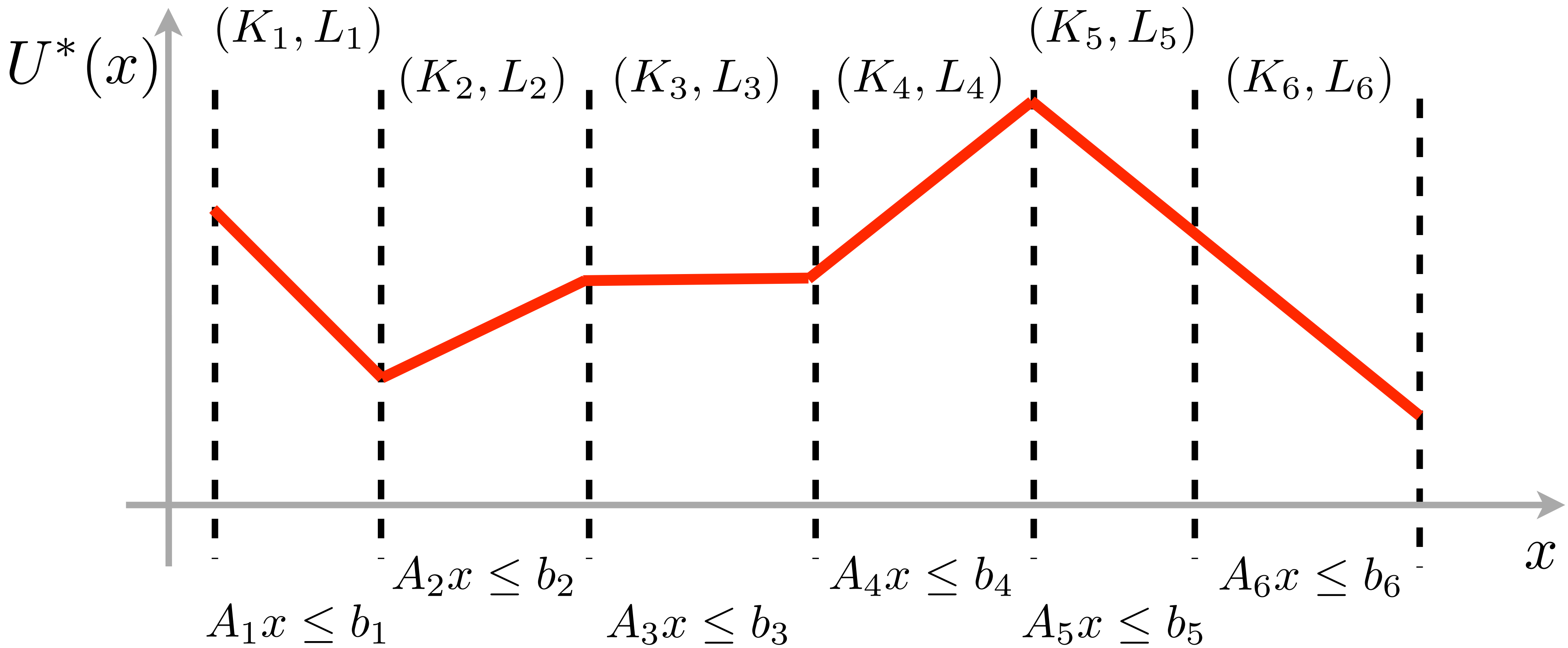
- Pick a new initial condition



- Solve the QP again, obtain explicit representation of the optimizer and form a new region

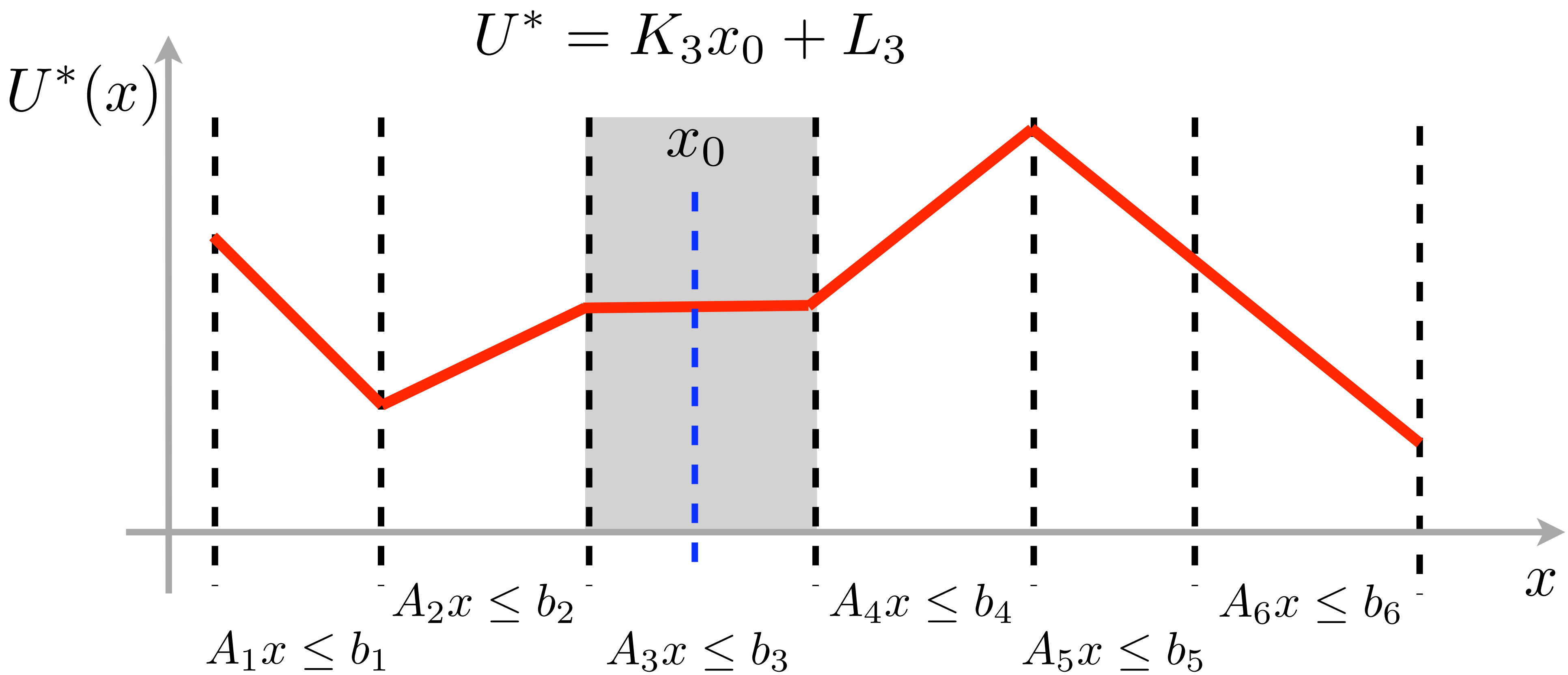


Solution Properties



- State space is divided into polytopic regions
- Control law is affine in each region

Implementation



- Identify region which contains current state
- Evaluate the feedback law

Pros & Cons

PROs:

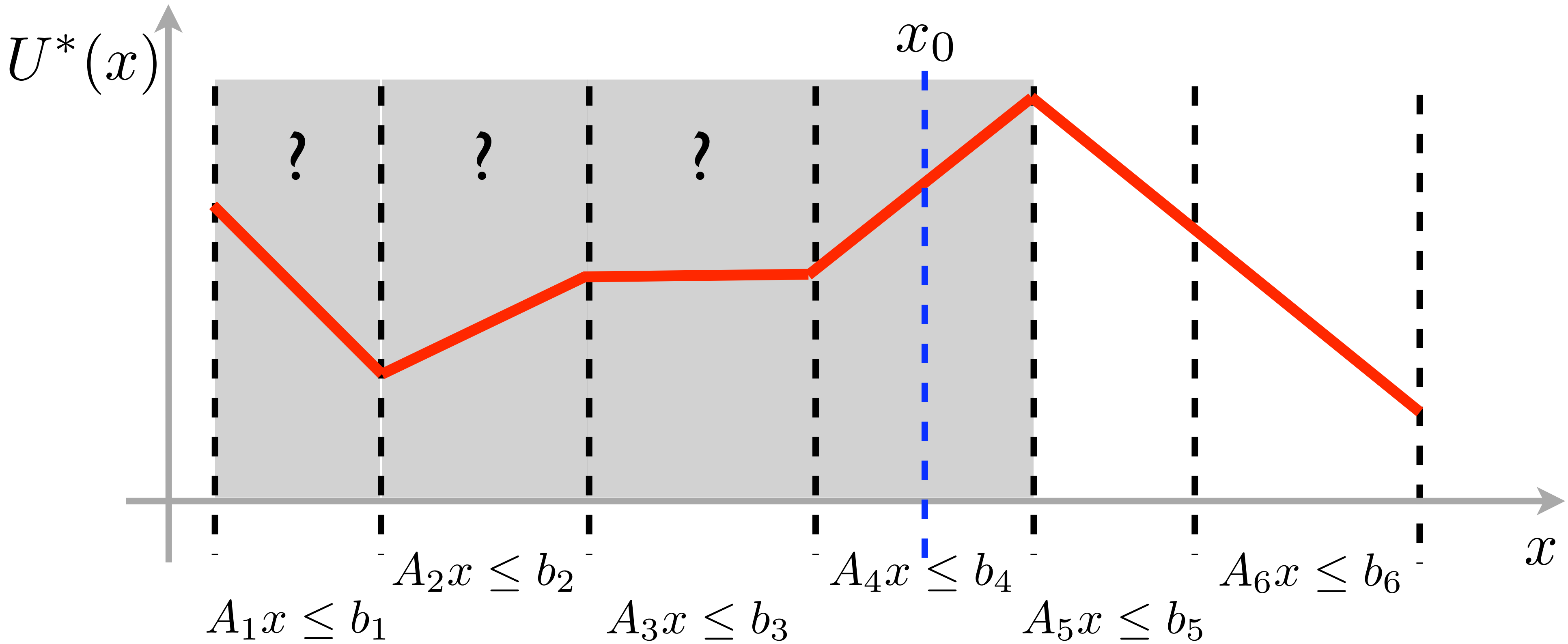
- easy to implement
- “fast” on-line evaluation
- analysis of implementation issues possible

CONs:

- number of controller regions can be large
- no control over the construction of the solution
- computation scales badly

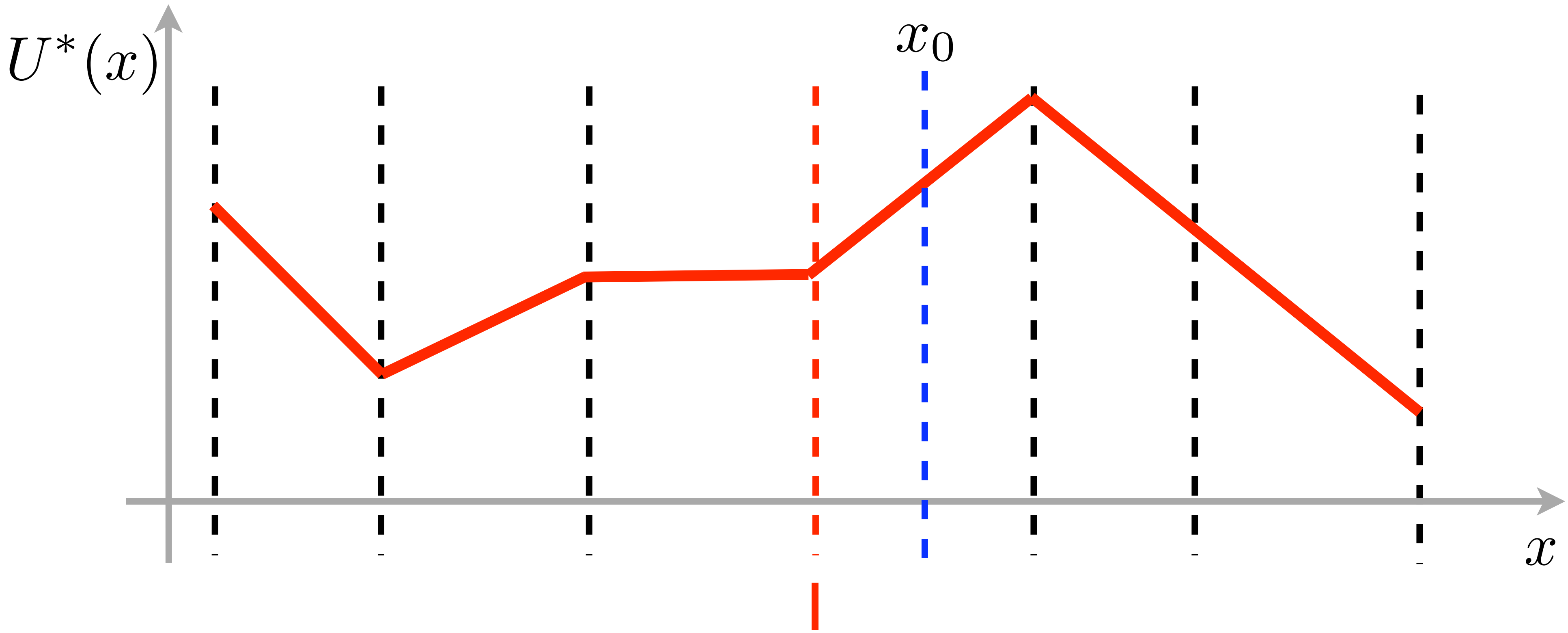
Controller complexity is the crucial issue!

Sequential Search



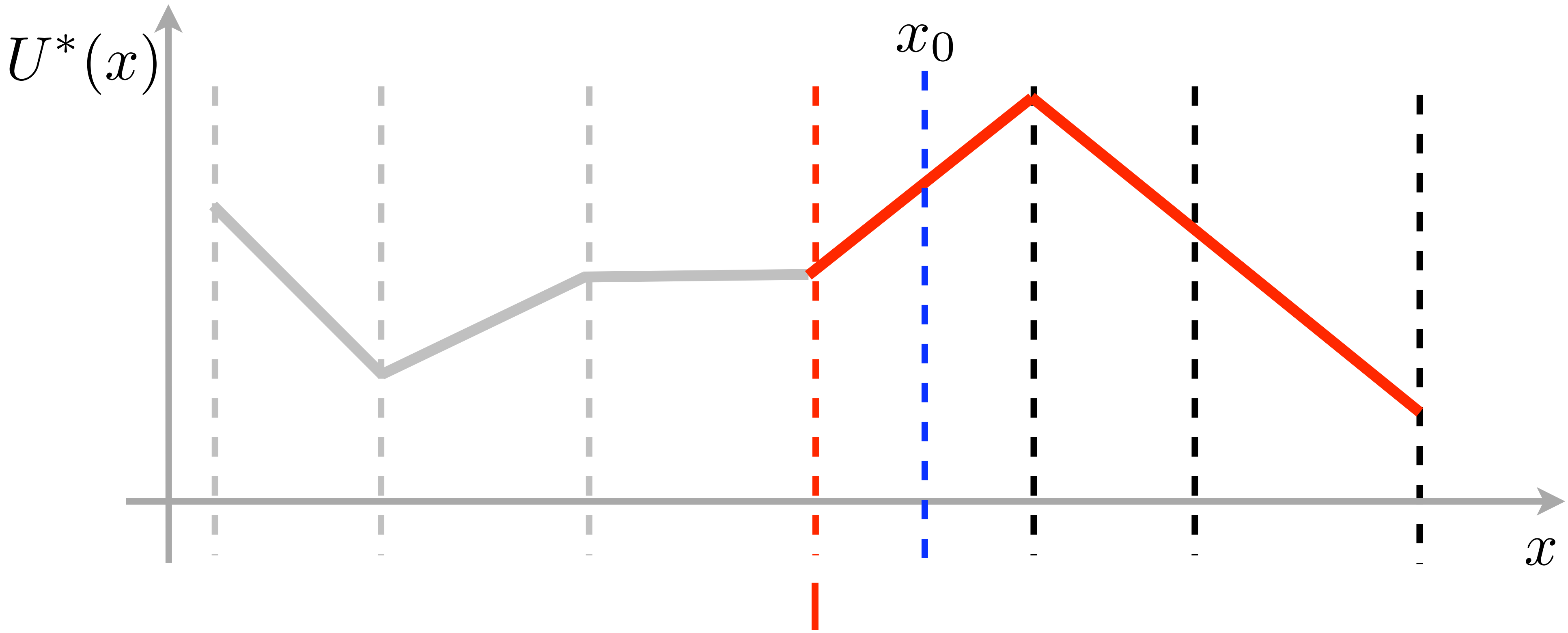
- Region identification (CPU): $\mathcal{O}(N_R)$
- Region storage (memory): $\mathcal{O}(N_R)$

Binary Search Tree



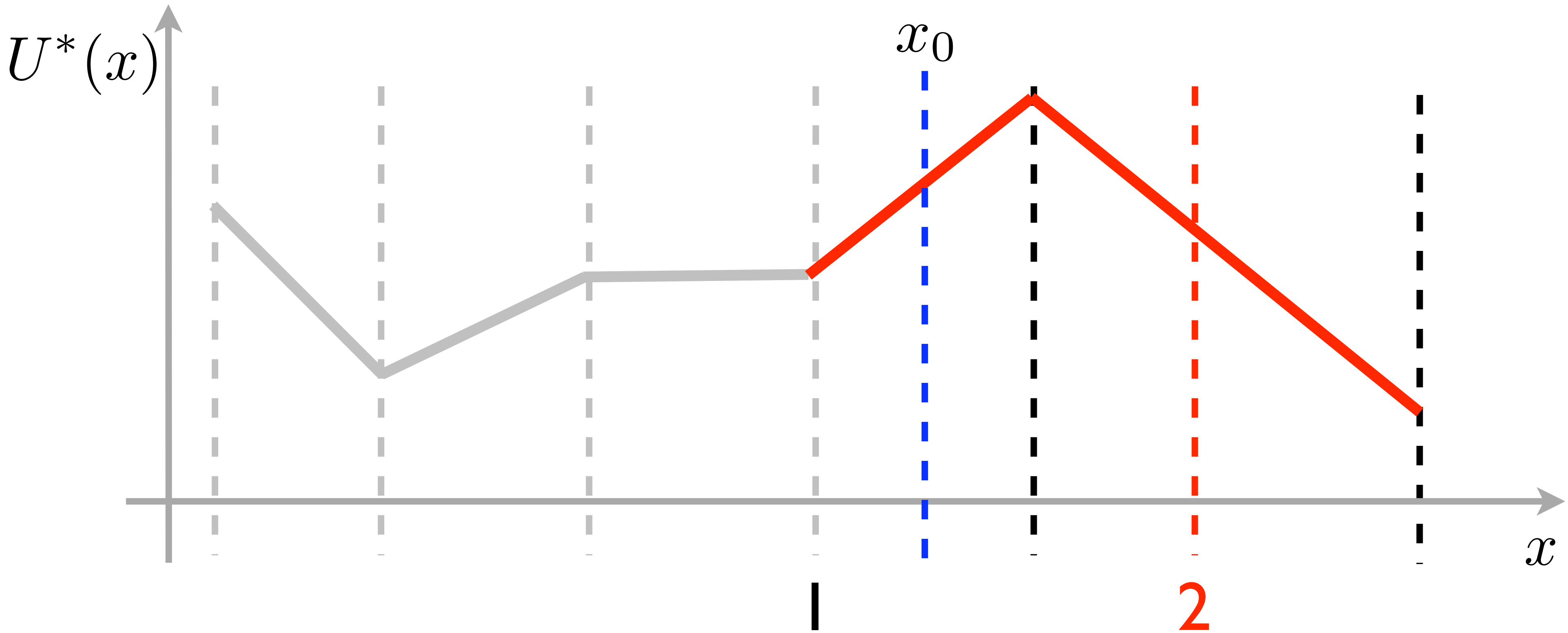
Tondel et al., Automatica 2003

Binary Search Tree



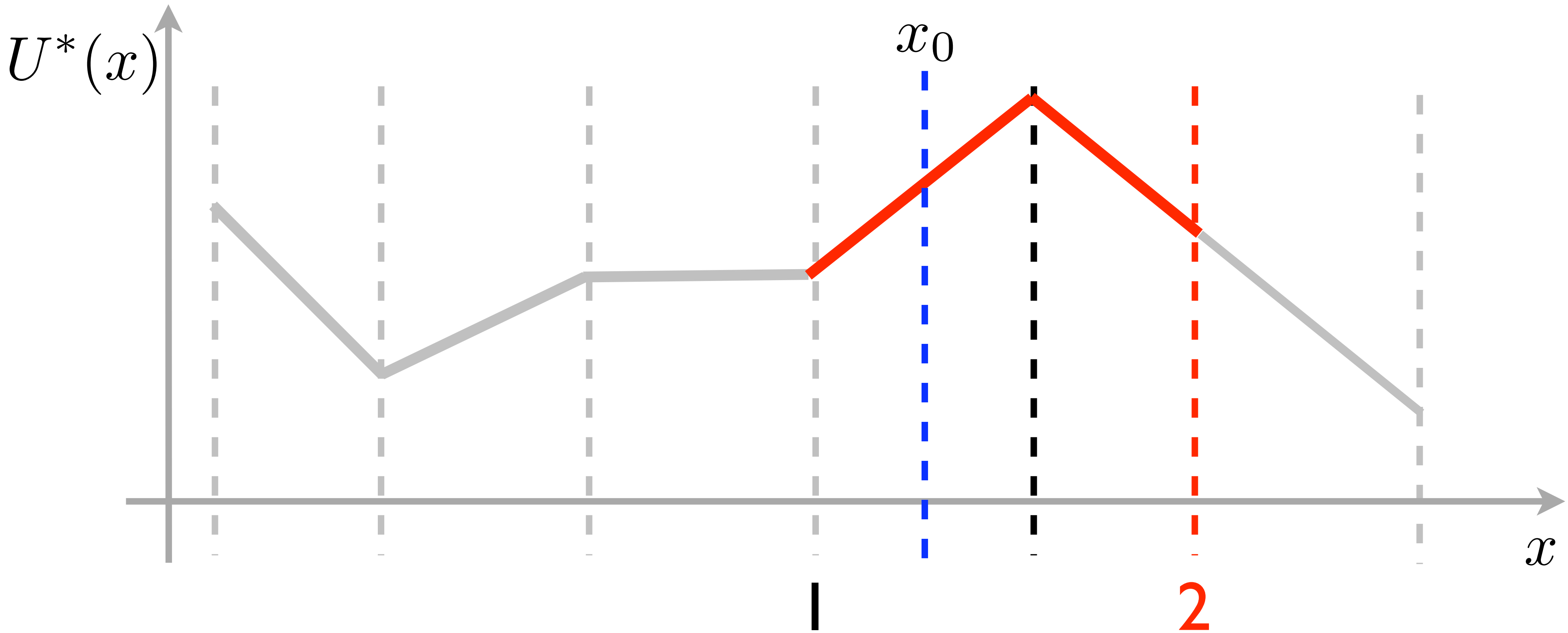
Tondel et al., Automatica 2003

Binary Search Tree



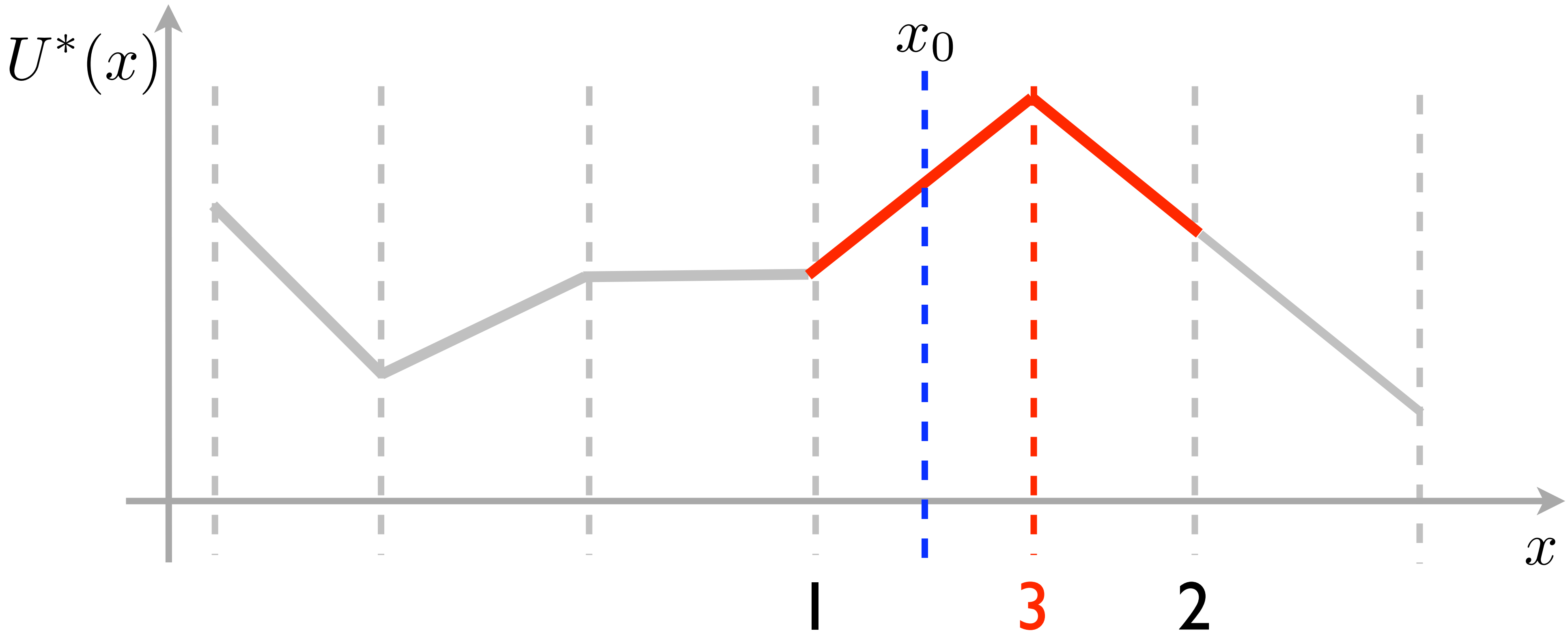
Tondel et al., Automatica 2003

Binary Search Tree



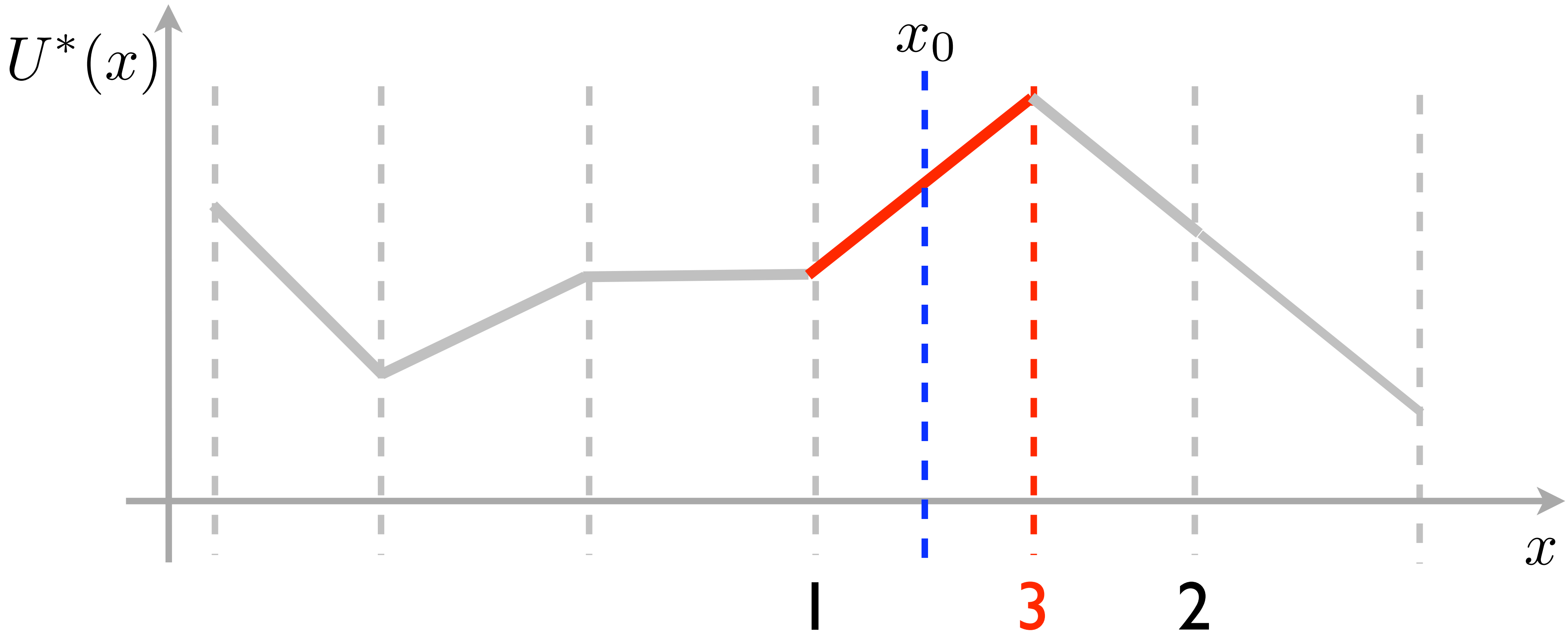
Tondel et al., Automatica 2003

Binary Search Tree



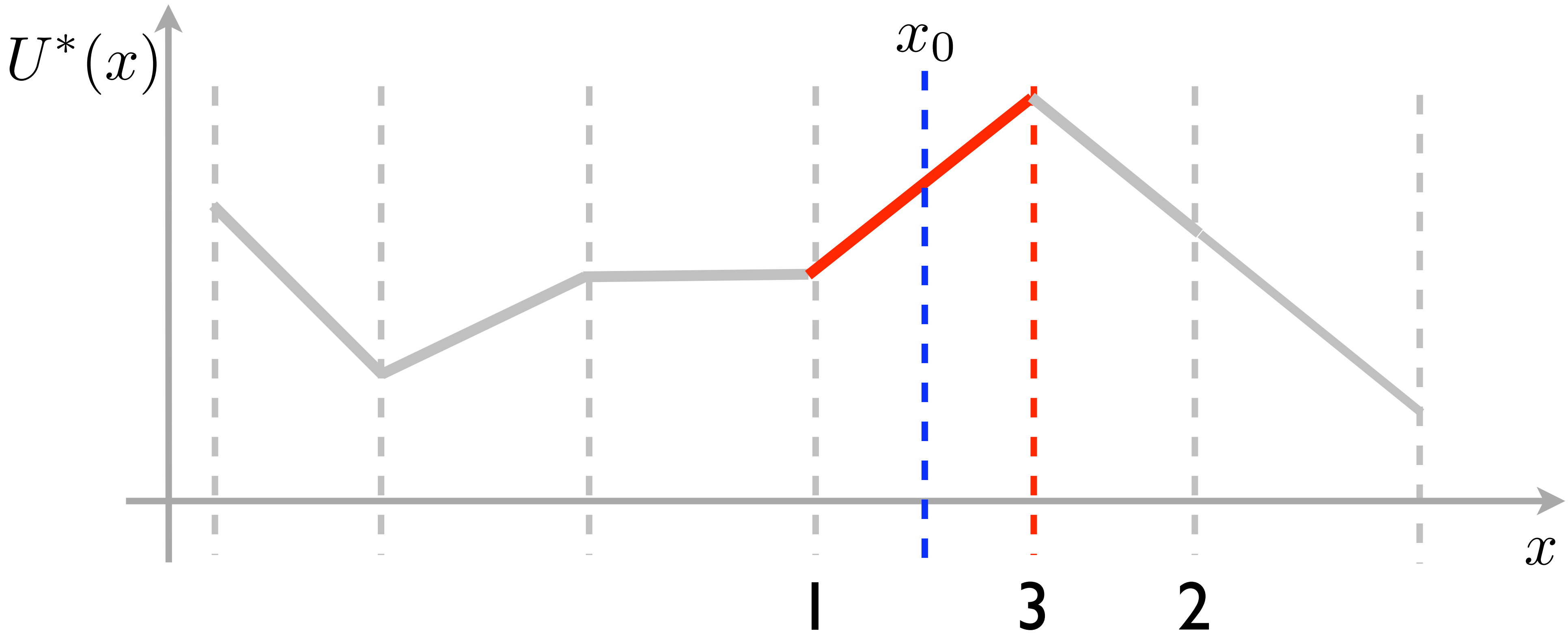
Tondel et al., Automatica 2003

Binary Search Tree



Tondel et al., Automatica 2003

Binary Search Tree



- Region identification (CPU): $\mathcal{O}(\log_2(N_R))$
- Region storage (memory): $\mathcal{O}(N_R)$

Complexity Comparison

# of regions	CPU FLOPS	Max sampling rate	Memory (B)
25	50	20 kHz	1 600
110	60	16 kHz	4 400
240	80	12 kHz	7 600

Assumed is a CPU with 1 MFLOPS

Hybrid Systems Seminar
Part 7: Closing Remarks

Michal Kvasnica

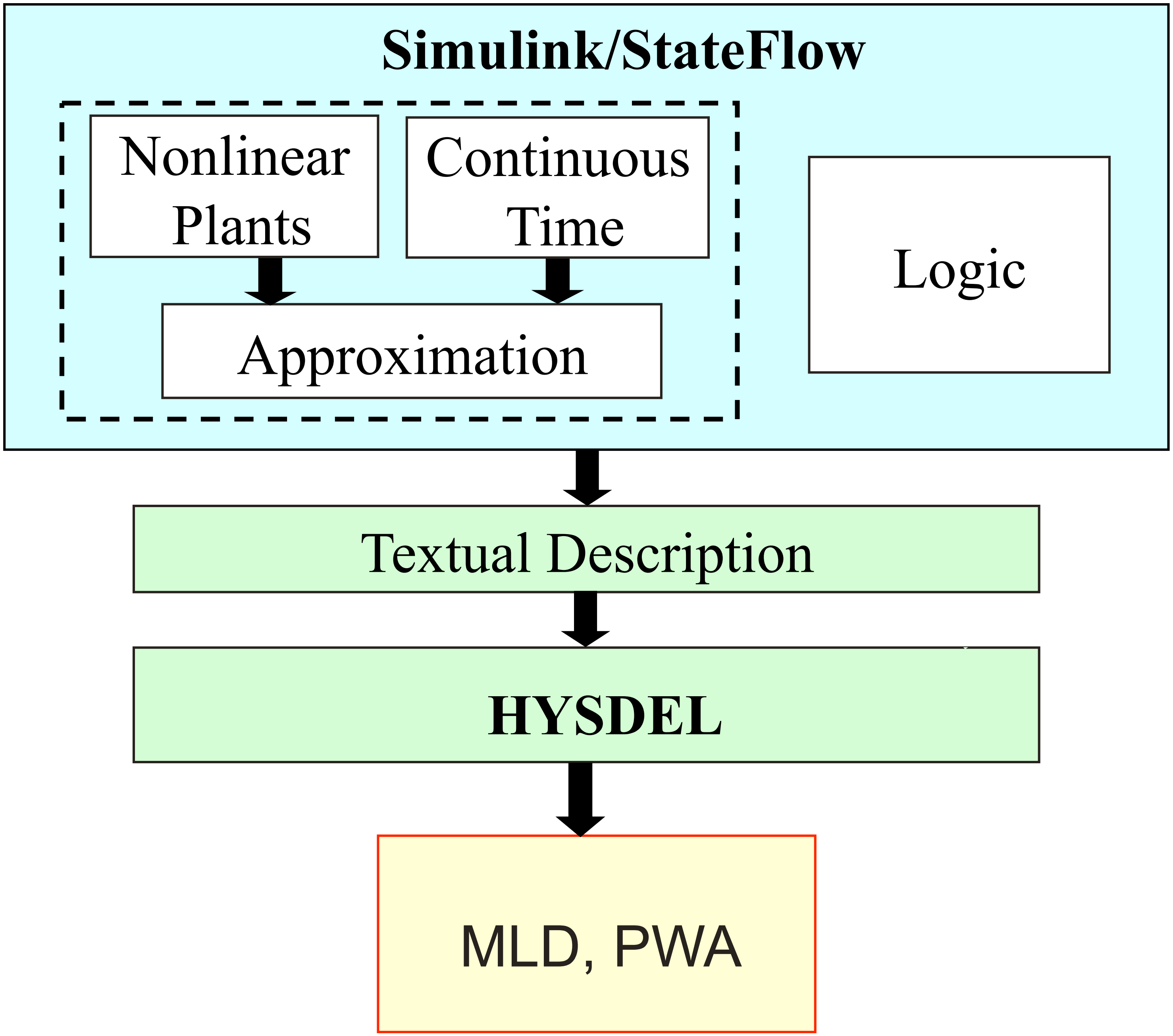
Hybrid Systems

- Successful in practice (cf. the ABB story)
- Main claimed benefits:
  - systematic approach to modeling, simulation and control
  - good compromise between quality and complexity of the models when hybrid model is used as an approximator of a nonlinear system
  - many systems are naturally hybrid (e.g. electrical devices)
- Main criticism:
  - creating a good hybrid model requires lots of expertise
  - not 100% clear how to optimize model quality
  - mixed-integer MPC problems are difficult to solve (but still easier compared to full nonlinear optimization)

Open Challenges

- Modeling
  - Can a fully automated PWA-based modeling tool be achieved?
  - Investigate behavior of mixed-integer solvers, figure out how to tune the model such that optimization runs significantly faster
- On-Line MPC:
  - All mixed-integer solvers are exponential in the worst case. Can we get a better bound on the runtime?
  - Conditioning, ordering of constraints influences the runtime by 10x. Can we figure out what the optimal pre-processing should be?
- Explicit MPC:
  - Complexity of explicit solutions is decisive. How to reduce the number of regions and/or speed up the region search?

Our Vision of Automated Hybrid Modeling



Software for Hybrid Systems

- Multi-Parametric Toolbox (includes HYSDEL2, YALMIP, HIT)
  - http://control.ee.ethz.ch/~mpt/
- HYSDEL 2.0
  - http://control.ee.ethz.ch/~hybrid/hysdel/
- HYSDEL 3.0
  - http://kirp.chtf.stuba.sk/~kvasnica/
- YALMIP
  - http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php
- Hybrid Identification Toolbox (HIT)
  - http://www-rocq.inria.fr/who/Giancarlo.Ferrari-Trecate/HIT_toolbox.html

Interesting References

- Main paper on MLD systems & MPC
  - Bemporad & Morari: *Control of Integrating Logic, Dynamics, and Constraints*, Automatica 1999
- Main book on mathematical modeling of systems with logic
  - Williams: *Model Building in Mathematical Programming*, Wiley, 1993
- Book on hybrid systems
  - Lunze: *Handbook of Hybrid Systems Control*, Cambridge Press, 2009
- Books on explicit MPC & hybrid systems
  - Borrelli: *Constrained Optimal Control of Linear and Hybrid Systems*, Springer, 2003
  - Kvasnica: *Real-Time Model Predictive Control via Multi-Parametric Programming*, VDM Verlag, 2009