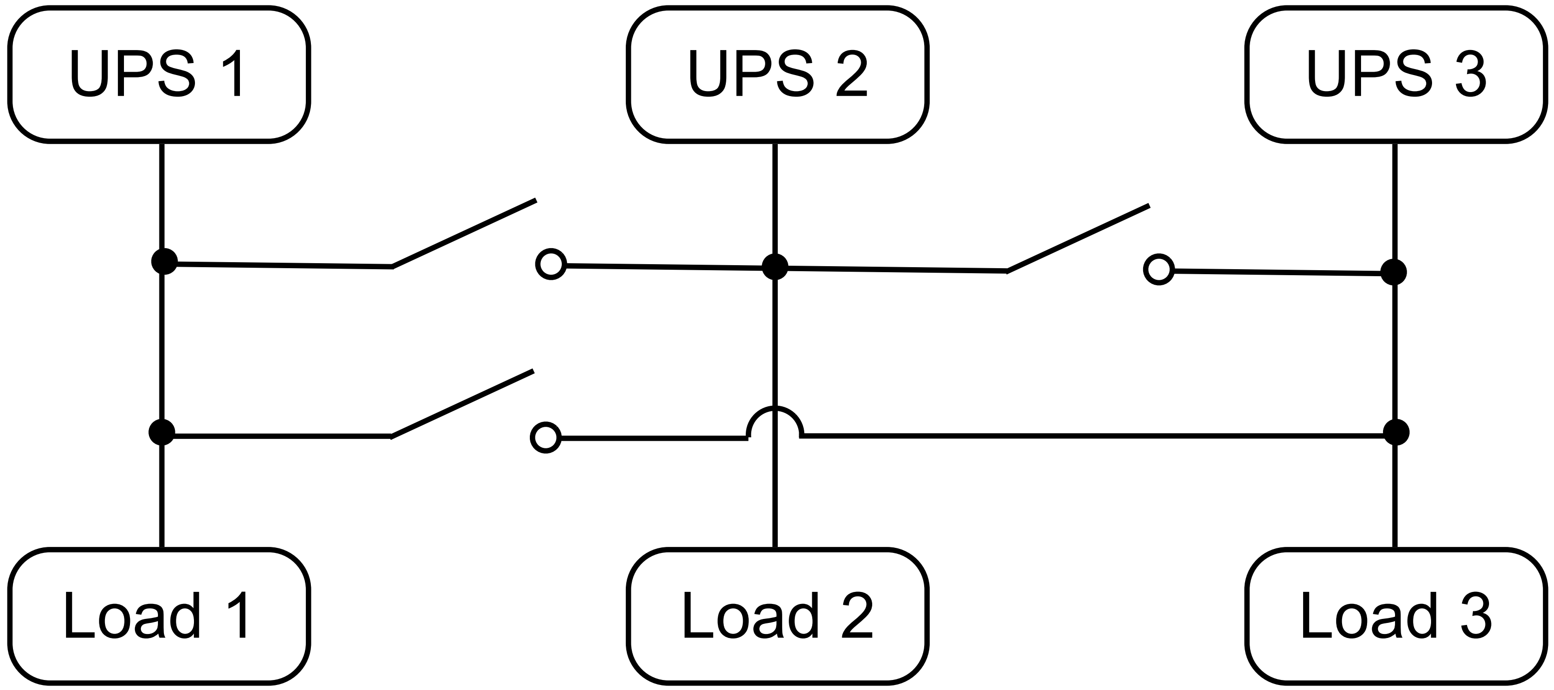
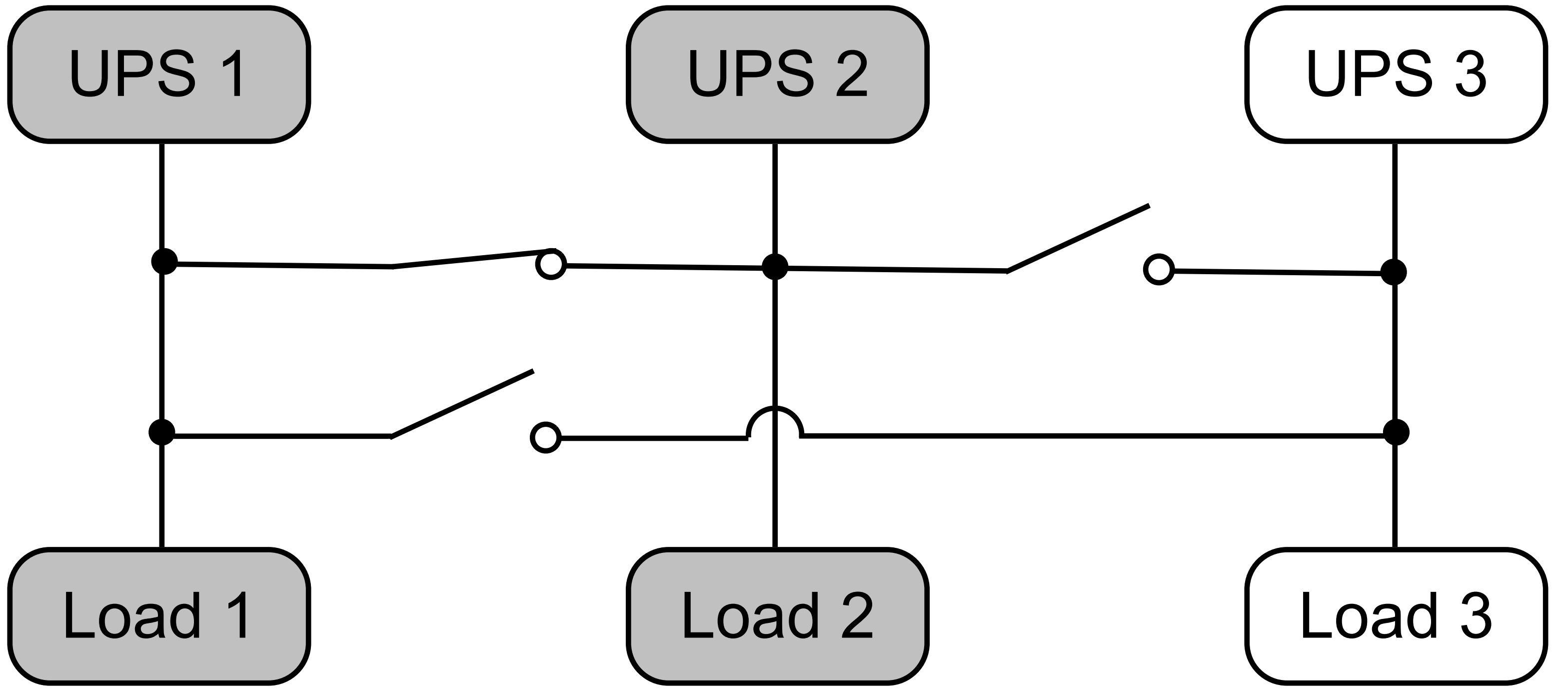
Hybrid Systems

Michal Kvasnica
Slovak University of Technology in Bratislava

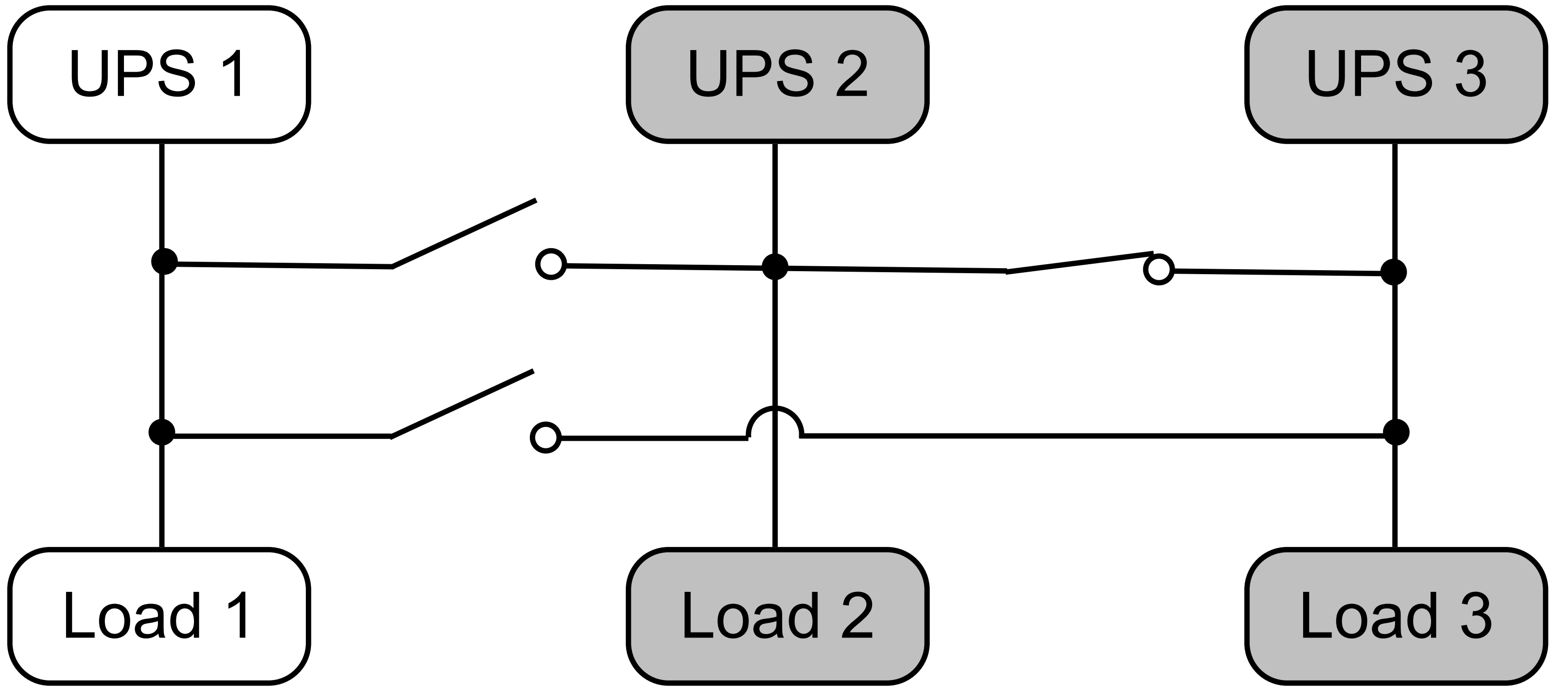
UPS Optimization Example



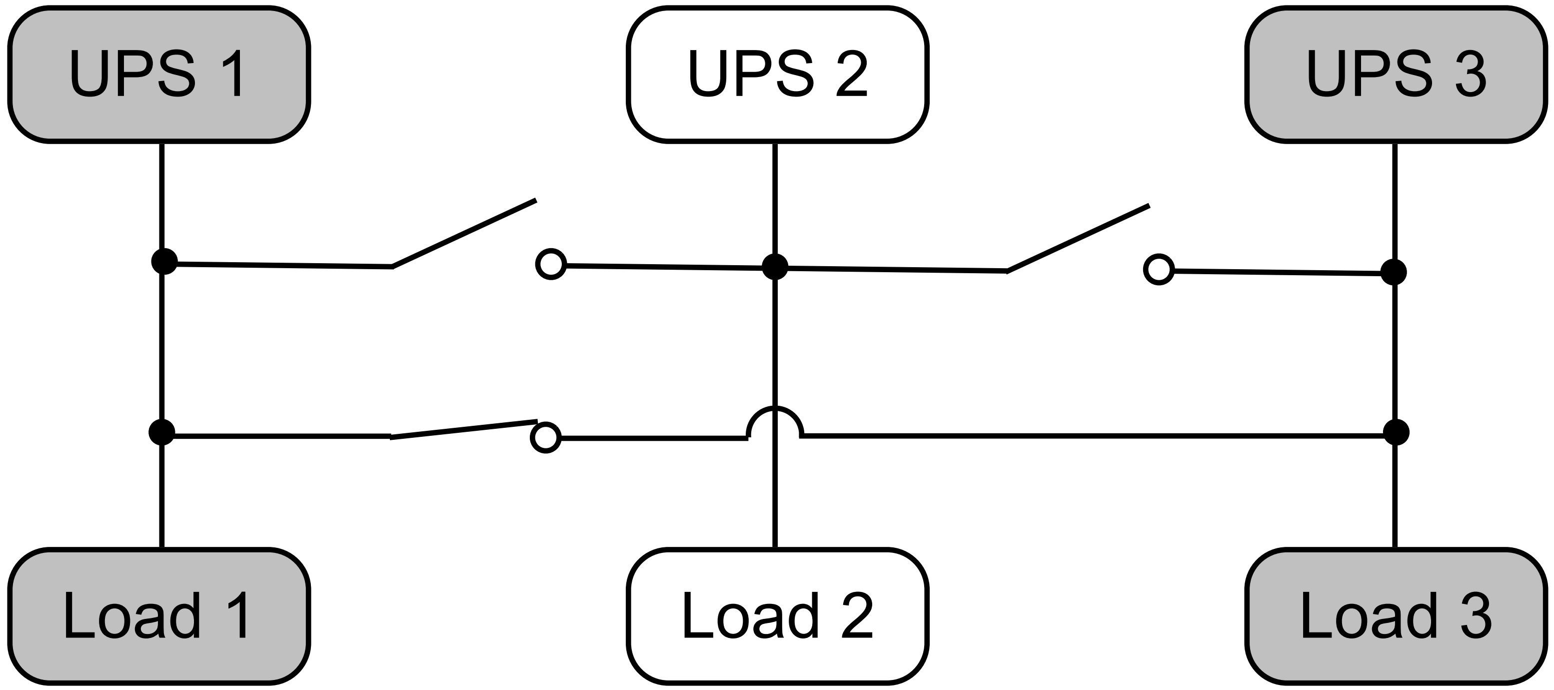
UPS Optimization Example



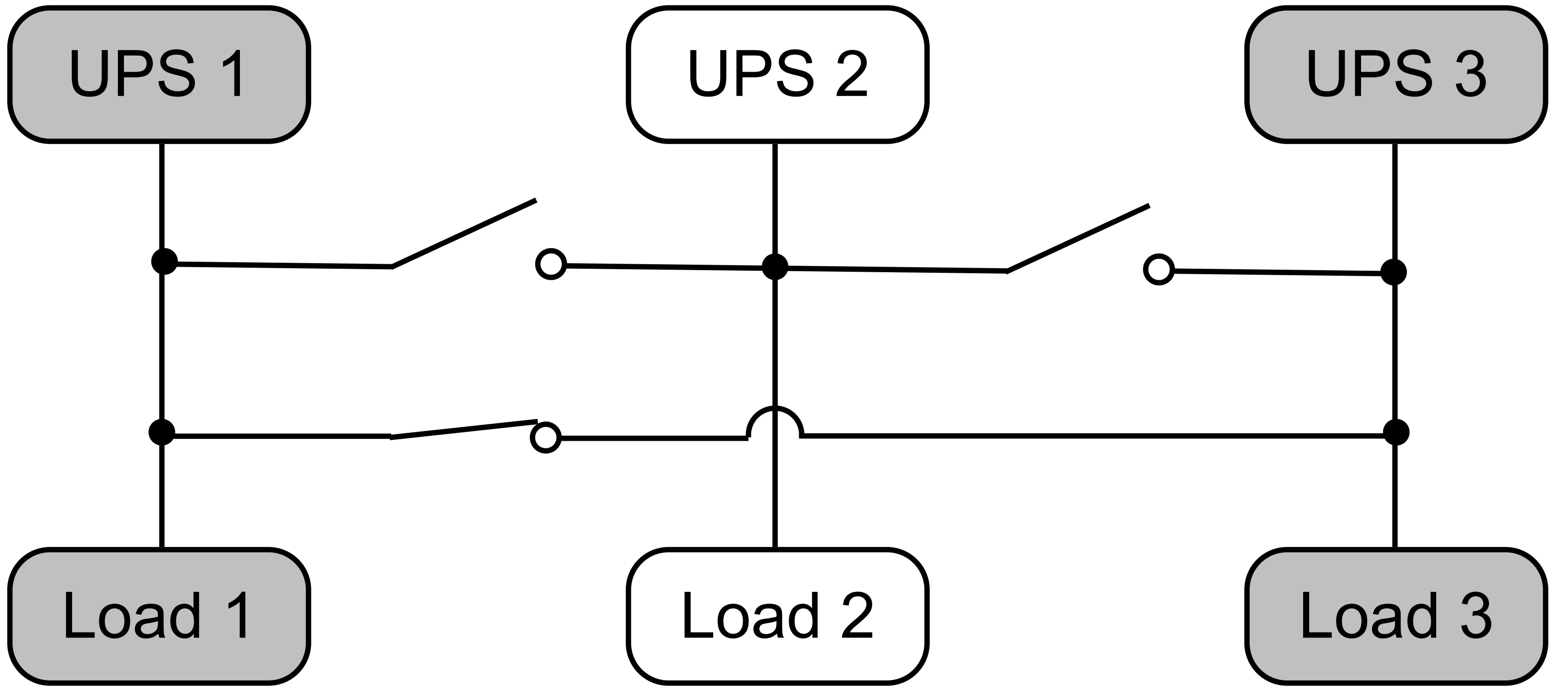
UPS Optimization Example



UPS Optimization Example

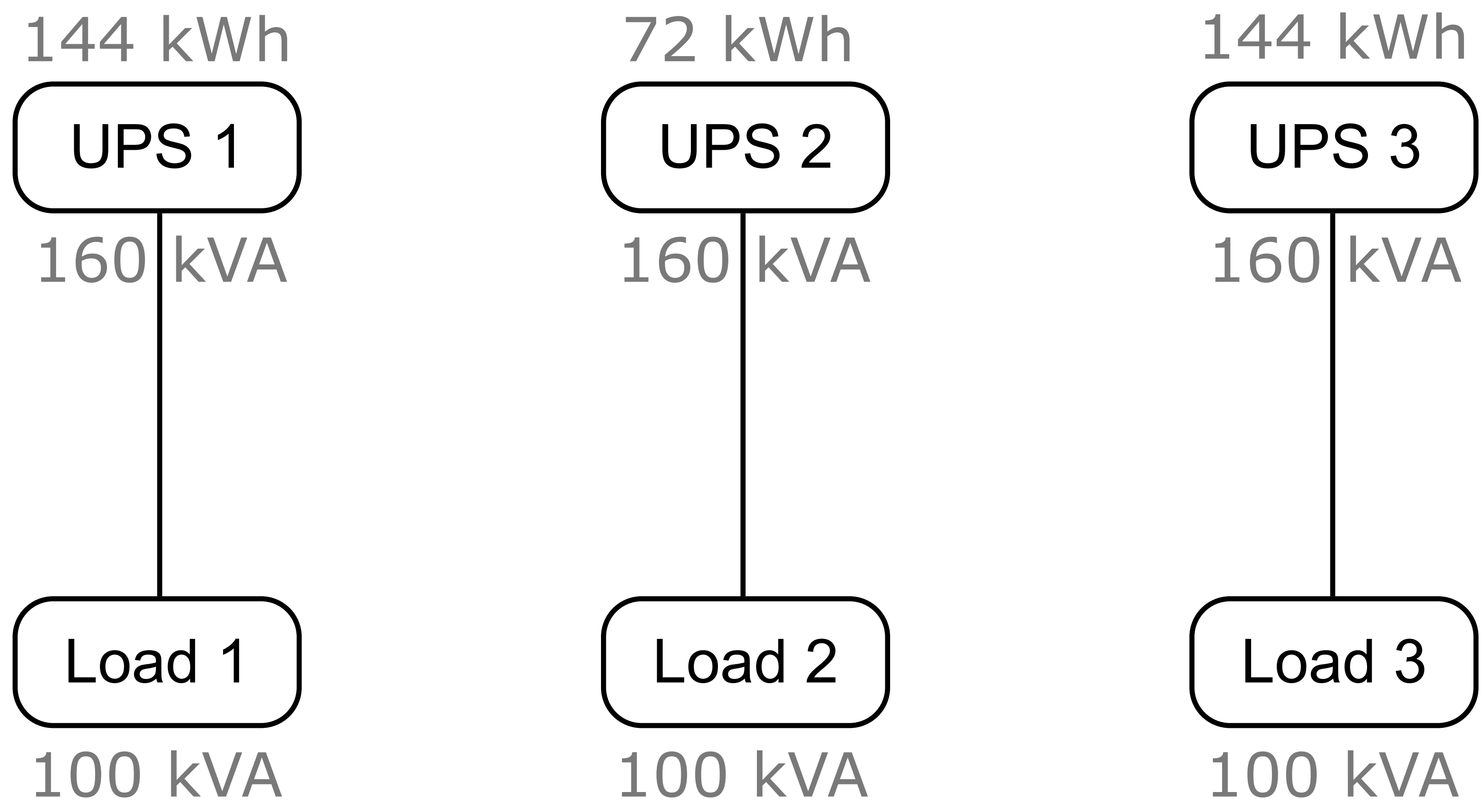


UPS Optimization Example



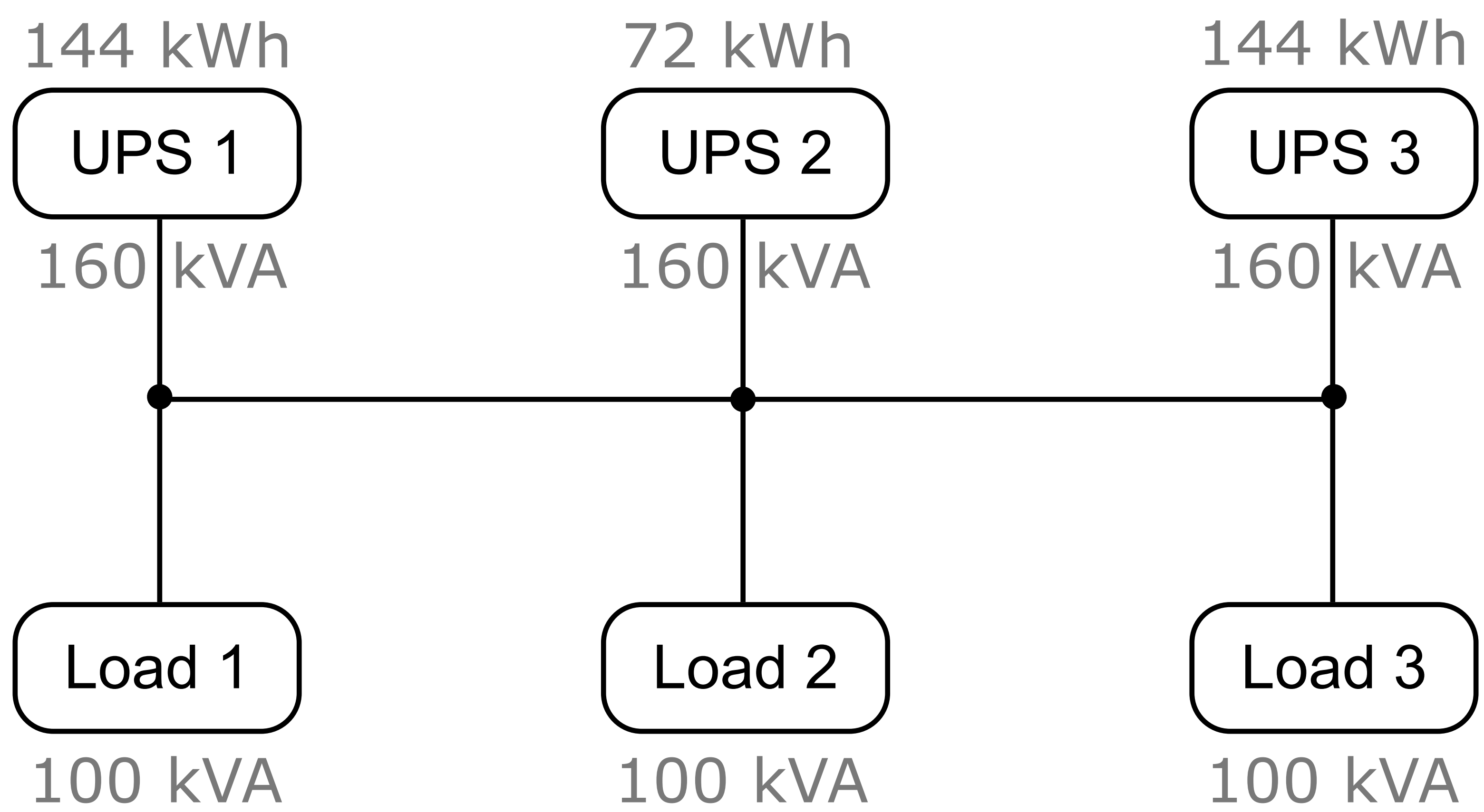
Maximize	**minimal uptime**
Considering	**current UPS capacities**
	UPS discharge model
	UPS & loads constraints
	only 1 switch at a time

UPS Optimization Example



Uptime without switching: **43** minutes

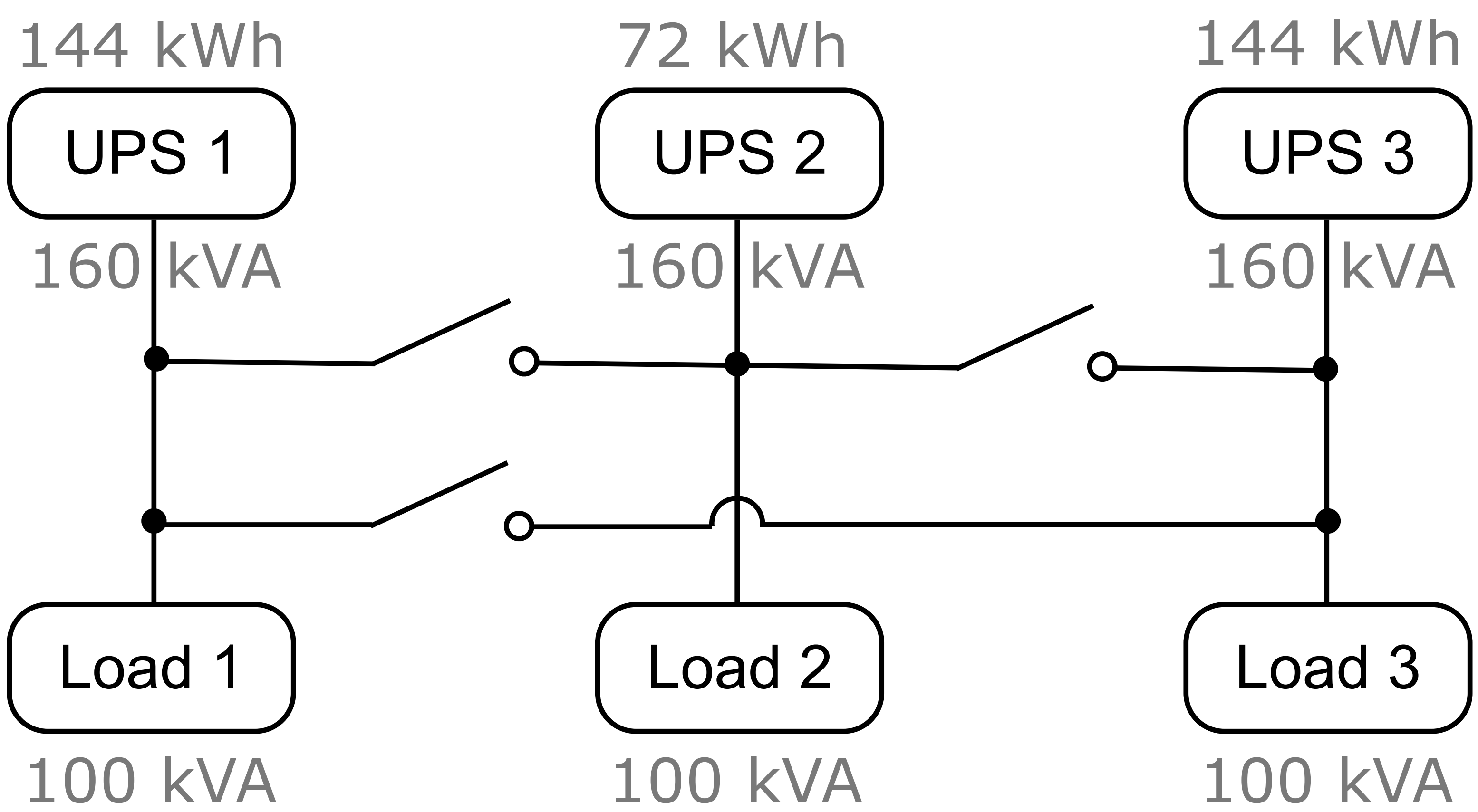
UPS Optimization Example



Uptime without switching: **43** minutes

Maximal theoretical uptime: **72** minutes

UPS Optimization Example

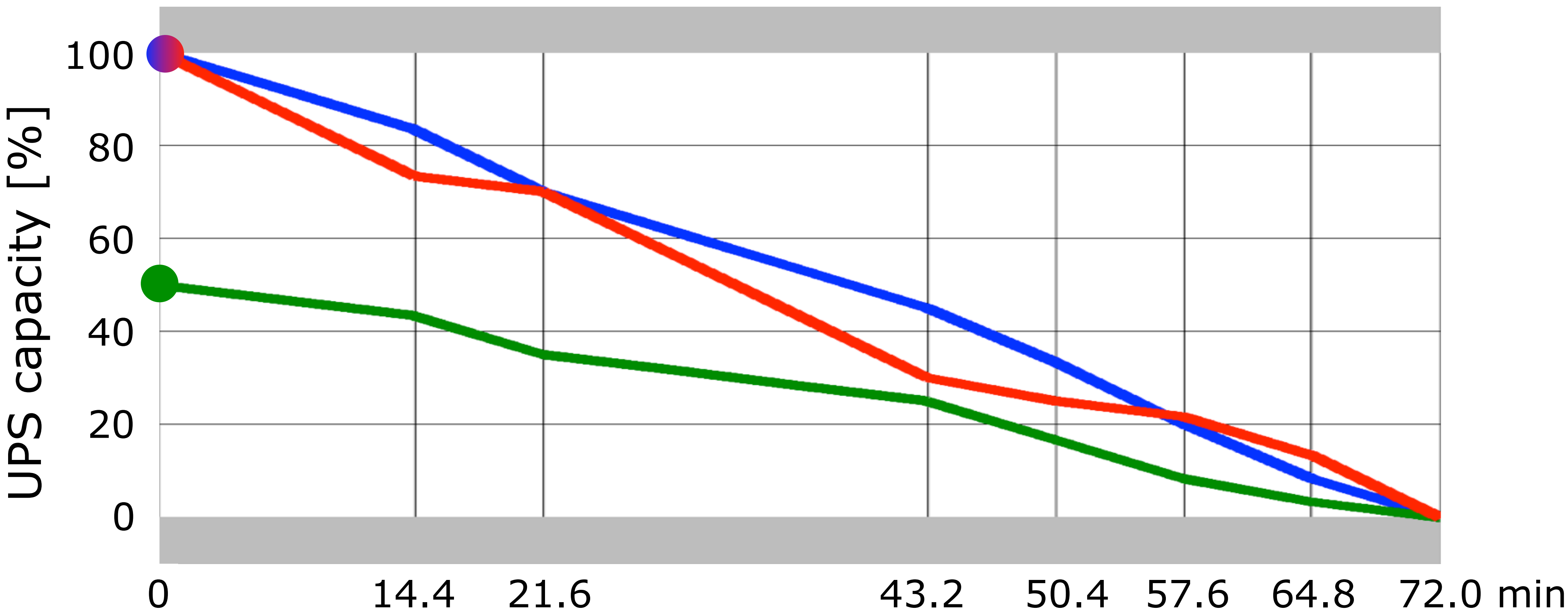


Uptime without switching: **43** minutes

Maximal theoretical uptime: **72** minutes

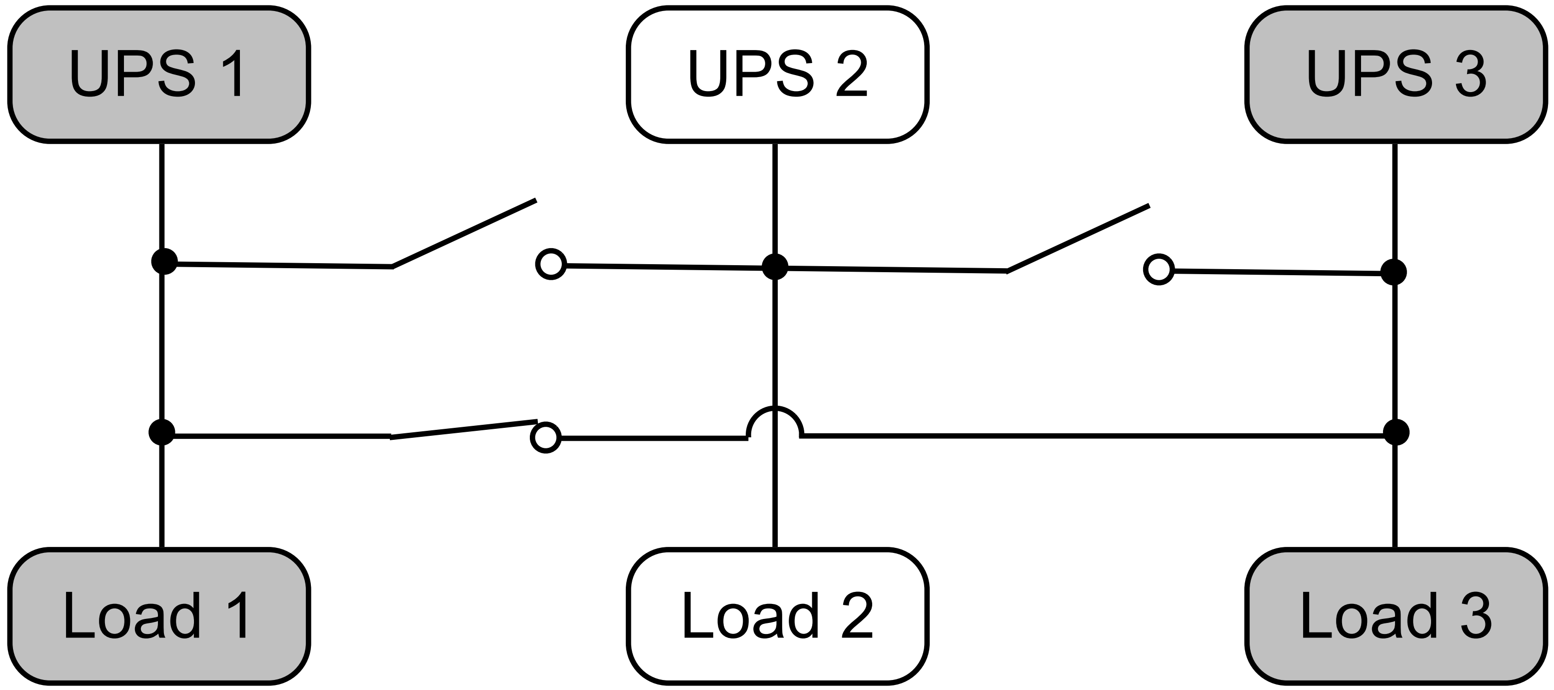
Uptime with switching: **72** minutes

UPS Optimization Example



UPS output [kVA]								
	#1	100	160	100	140	160	140	100
	#2	40	100	40	100	100	60	40
	#3	160	40	160	60	40	100	160

UPS Optimization Example



Maximize **minimal uptime**

Considering **current UPS capacities**

UPS discharge model

UPS & loads constraints

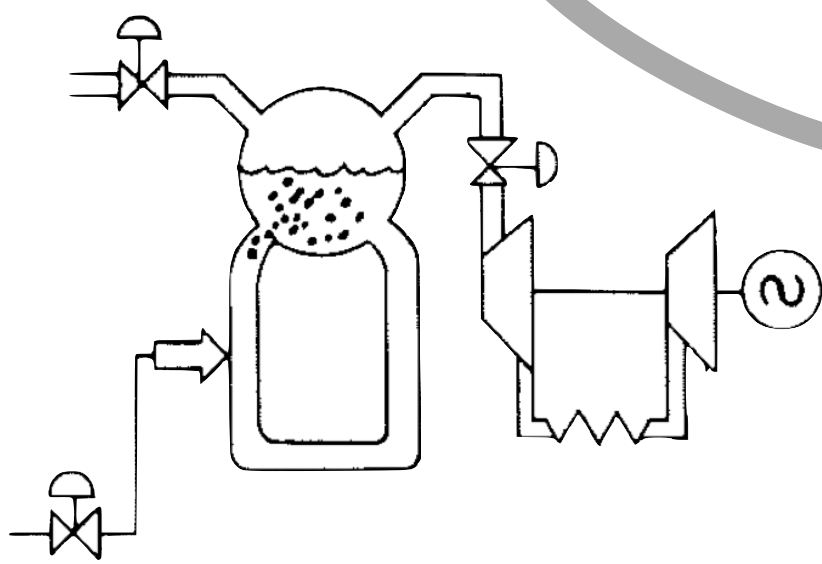
only 1 switch at a time

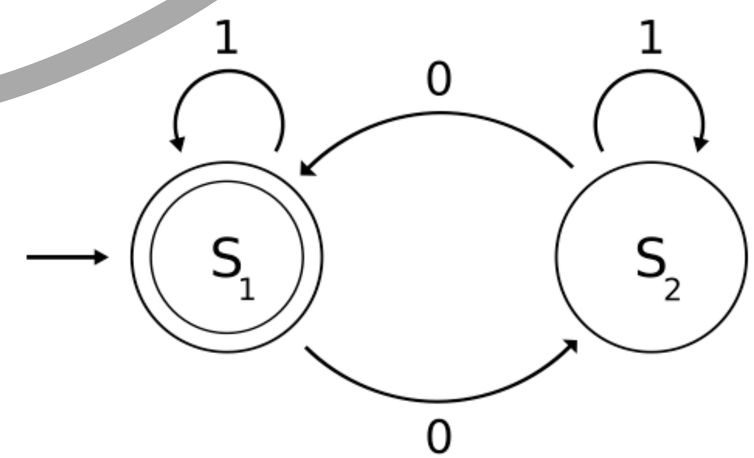
If-then rules based on switches

Control theory

Computer science

Hybrid systems





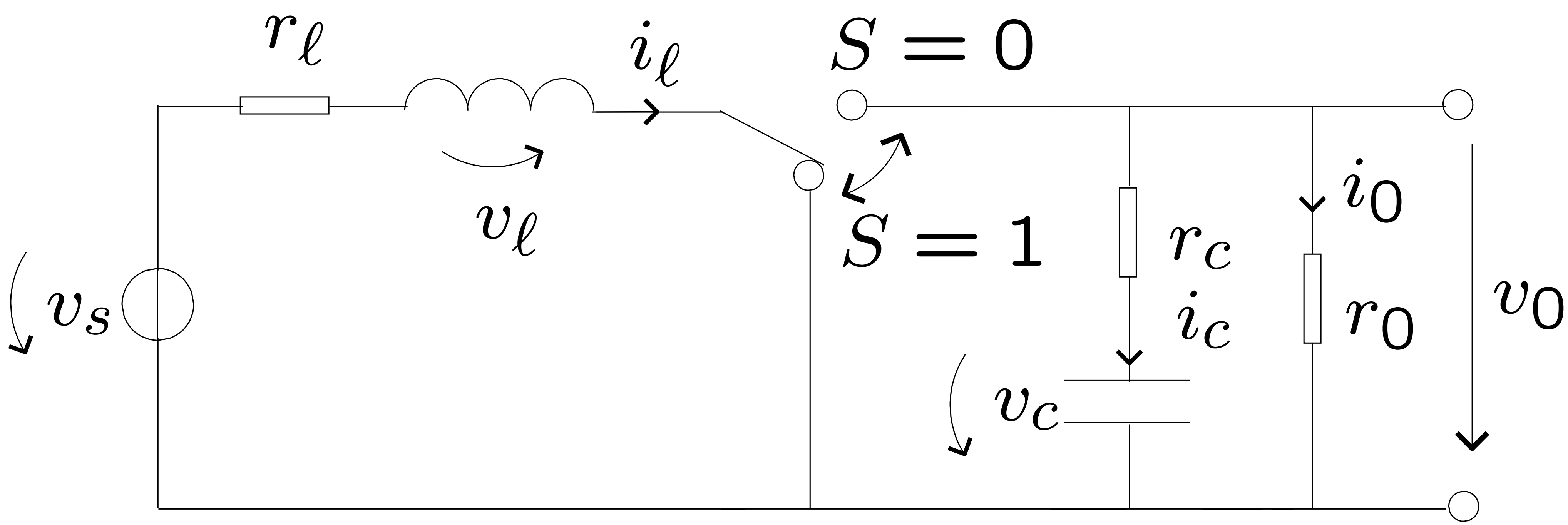
Real-valued signals
Differential equations

Binary signals
If-then-else rules
Finite-state machines

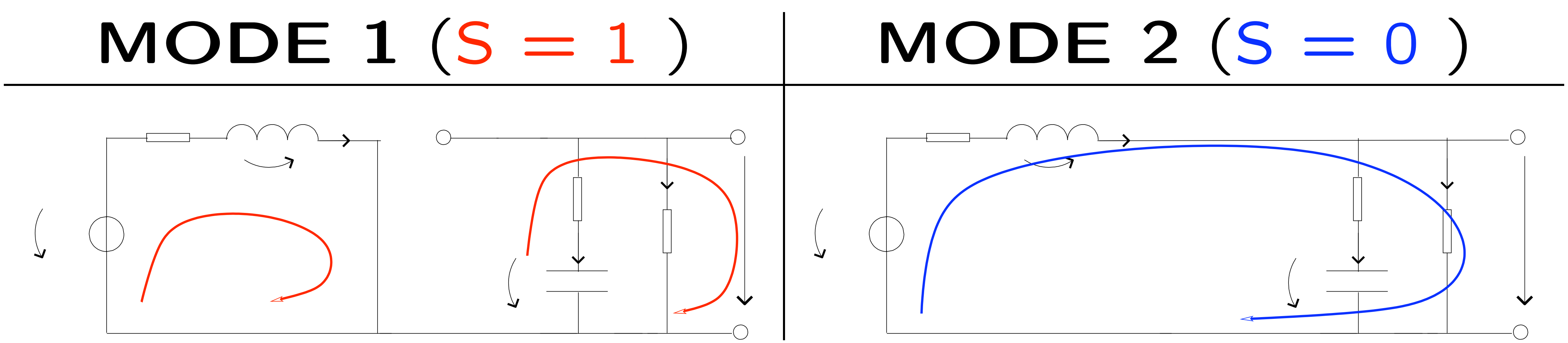
Hybrid Systems in Practice

- Plants with binary controls (e.g. turbine on/off)
- Logic constraints (e.g. when unit 1 is on, unit 2 must be off)
- Multi-stage control (e.g. startup, normal operation, shutdown)
- Systems with nonlinearities (e.g. hysteresis or dead zone)

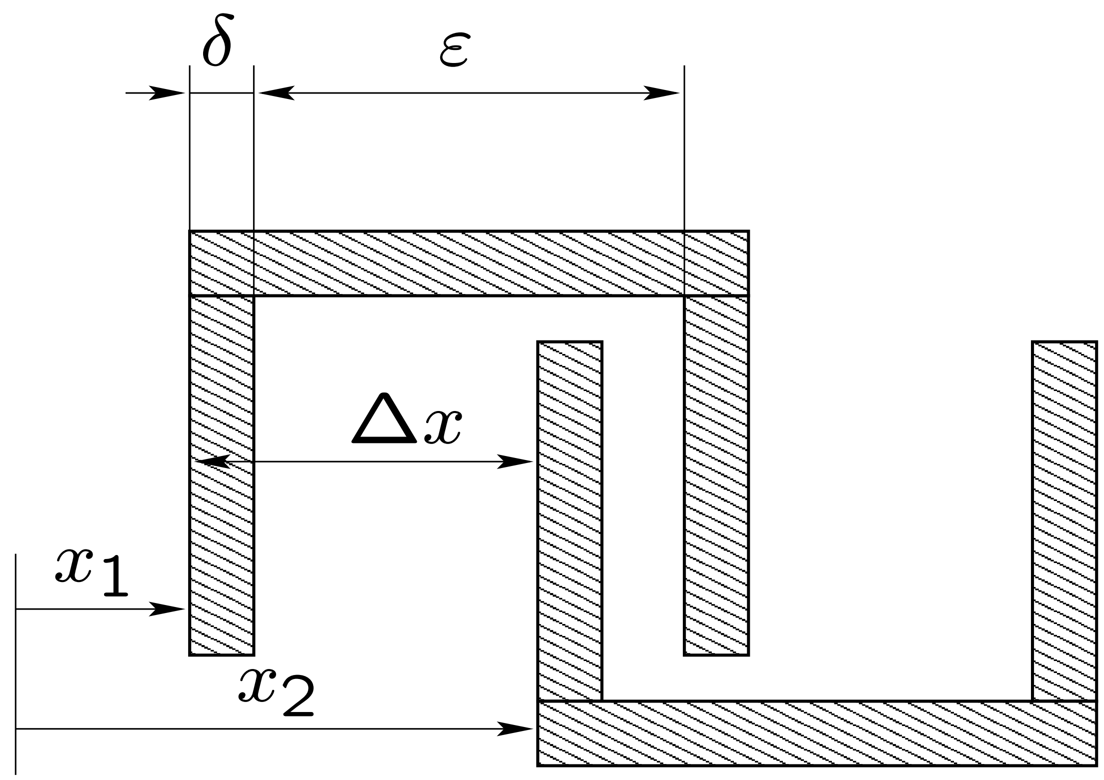
DC-DC Converter



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

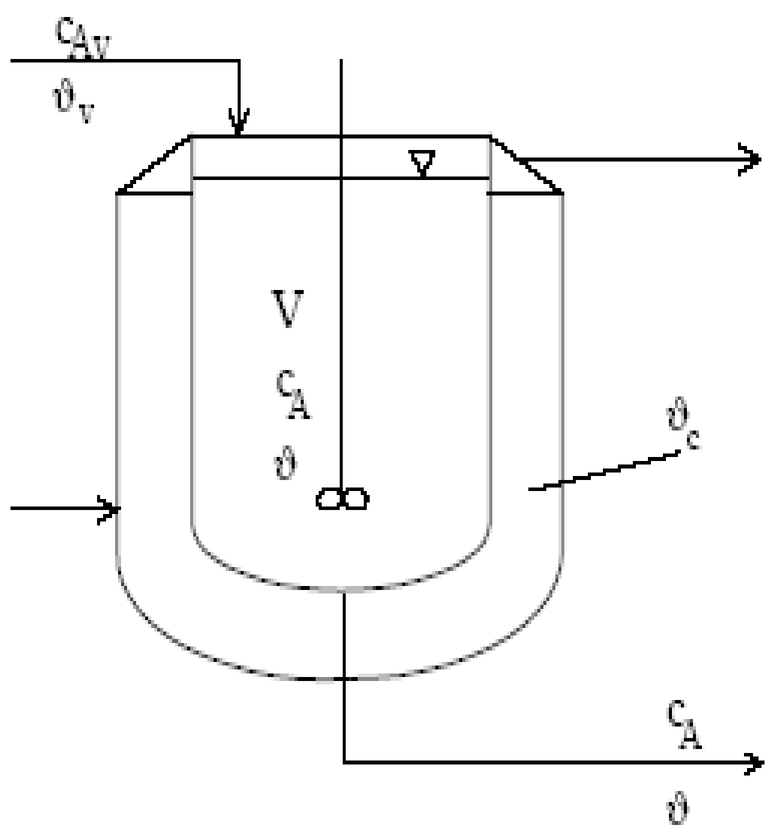


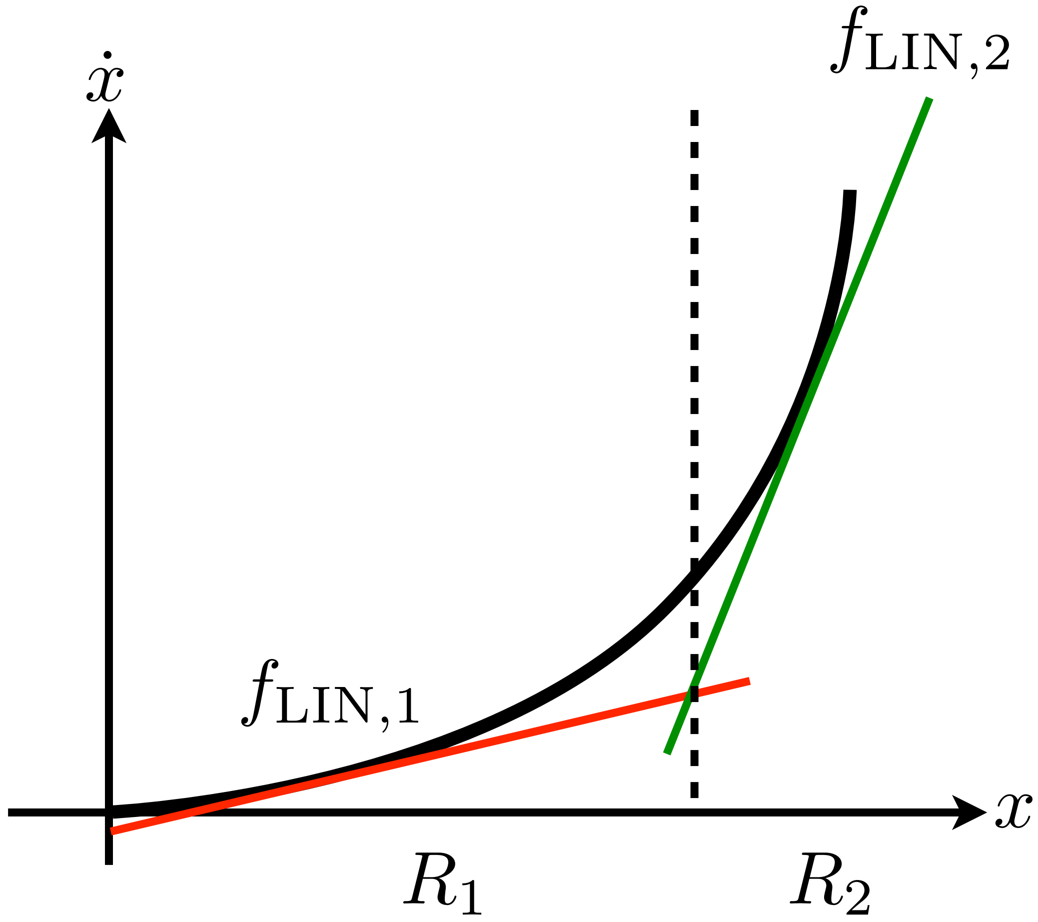
Mechanical System with Backlash



- 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

1. Discrete Hybrid Automata

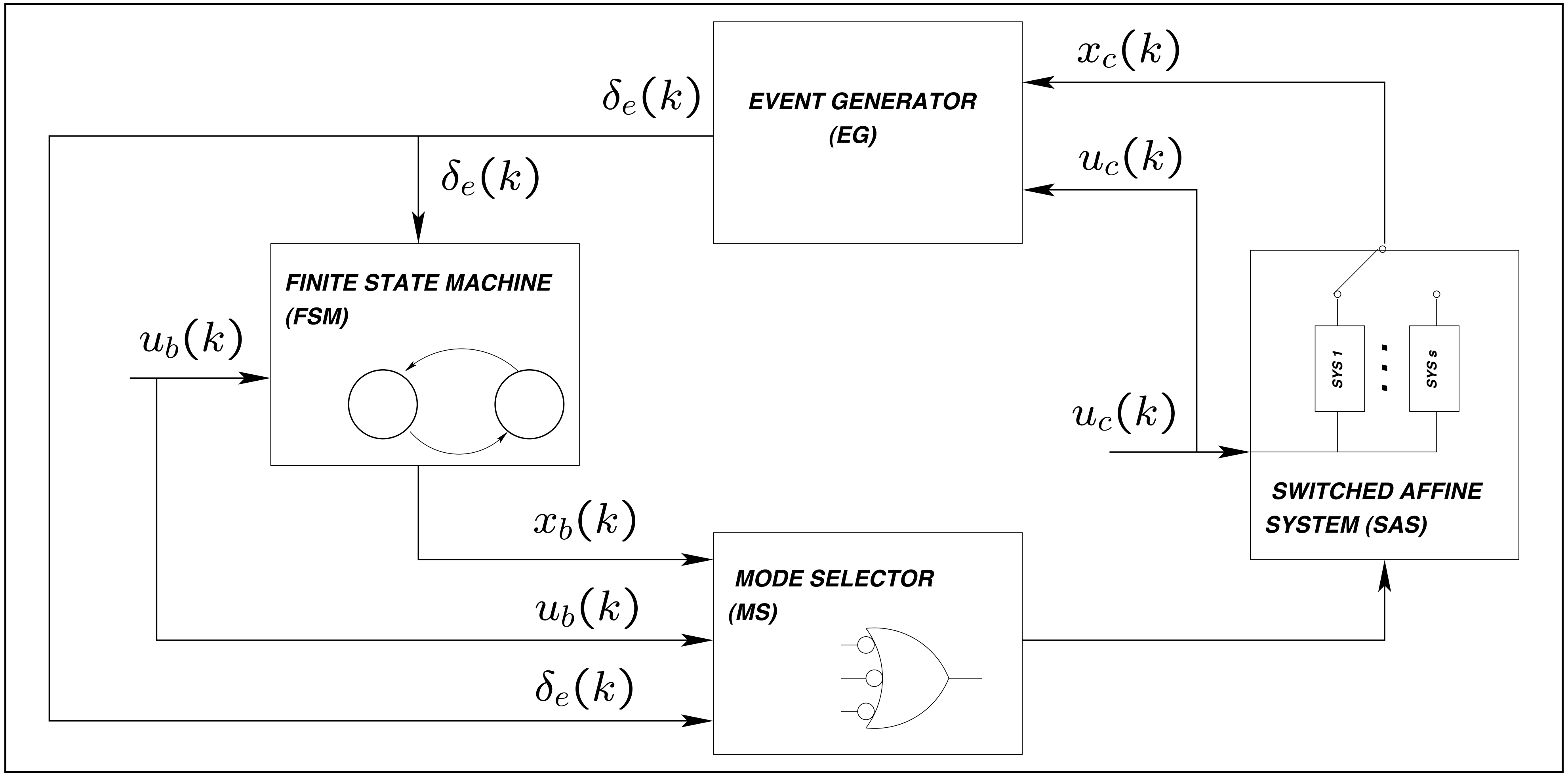
2. HYSDEL

3. Piecewise Affine Models

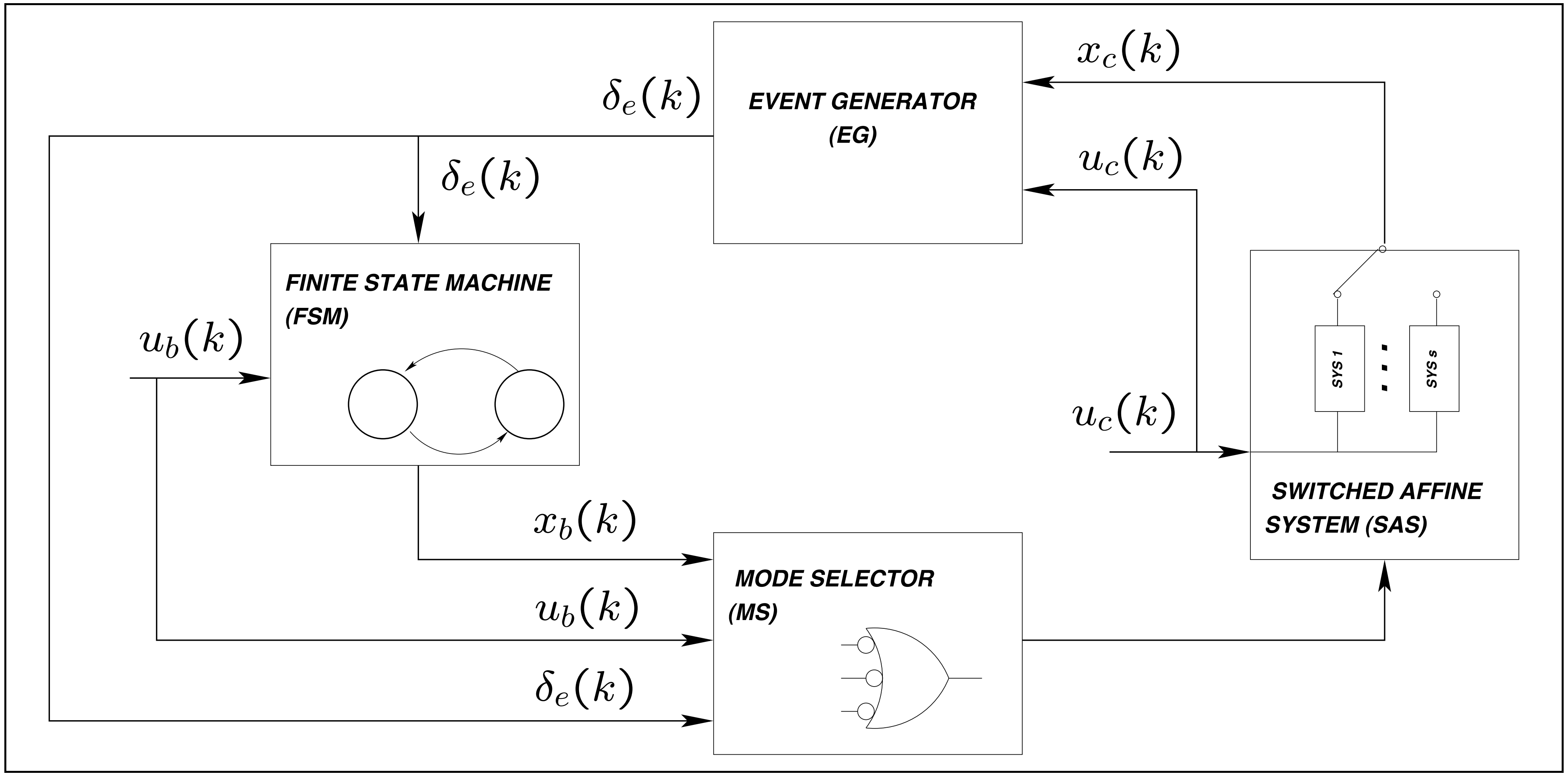
4. MPC for Hybrid Systems

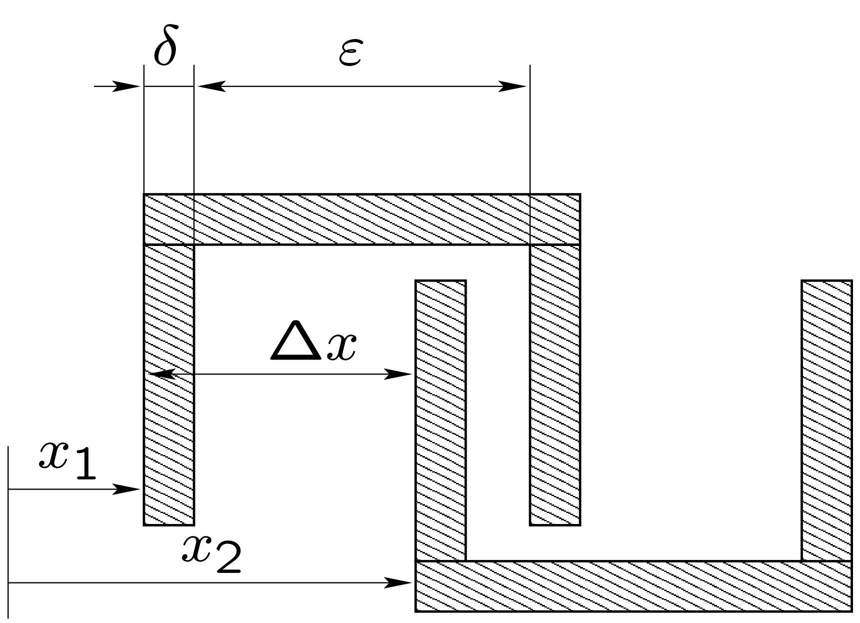
5. Closing Remarks

Discrete Hybrid Automata



Discrete Hybrid Automata



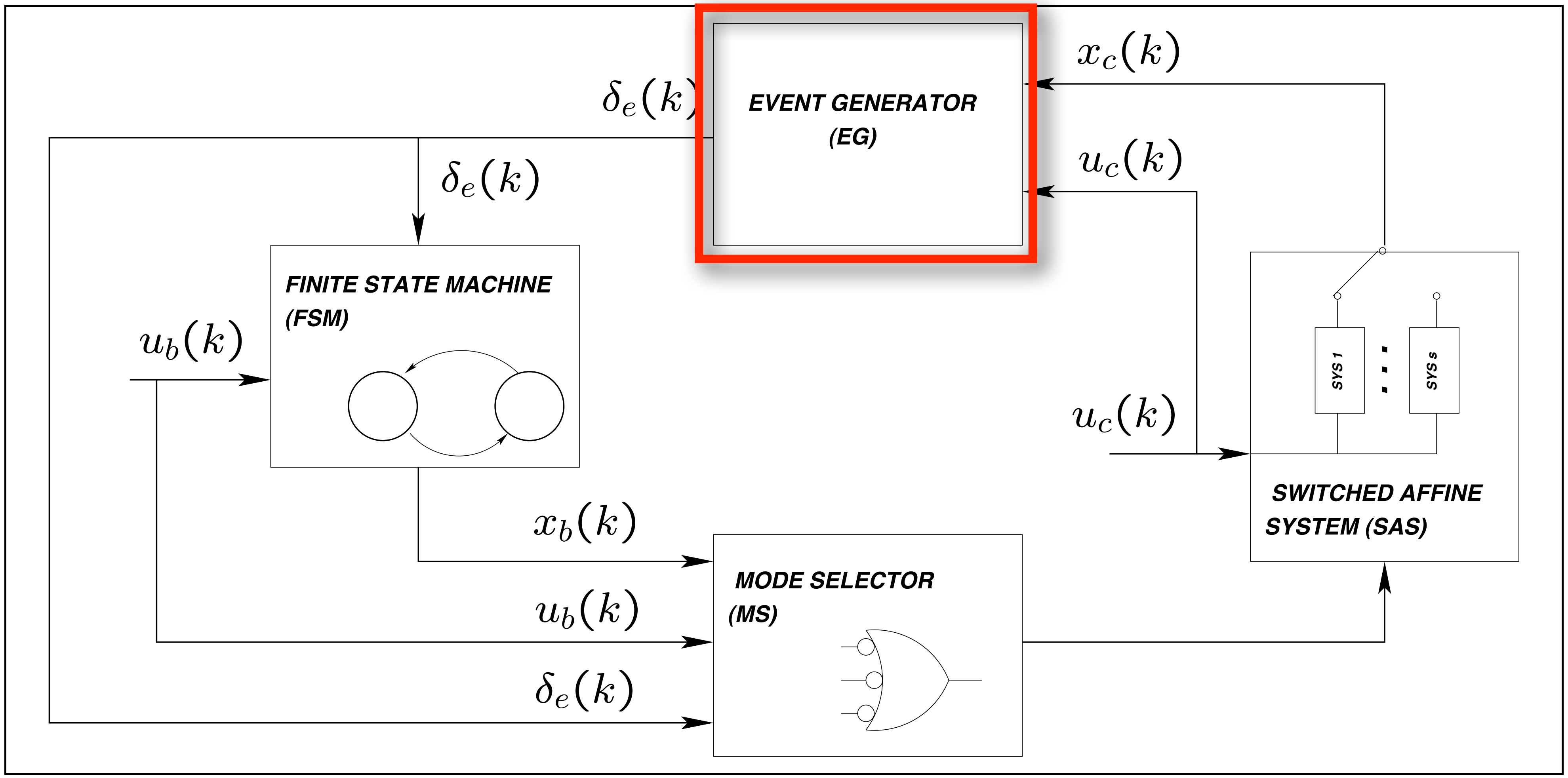


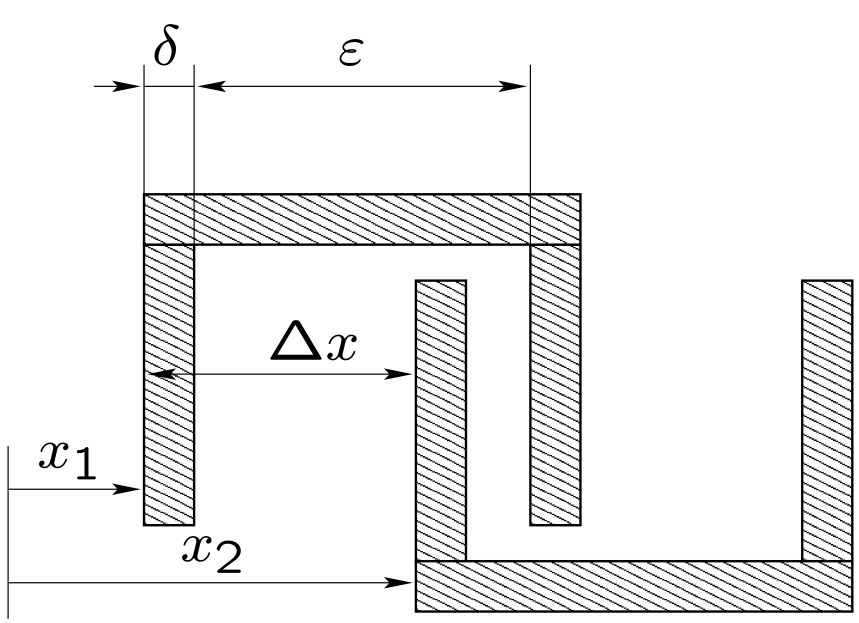
- Contact mode:

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

- Backlash mode

Discrete Hybrid Automata



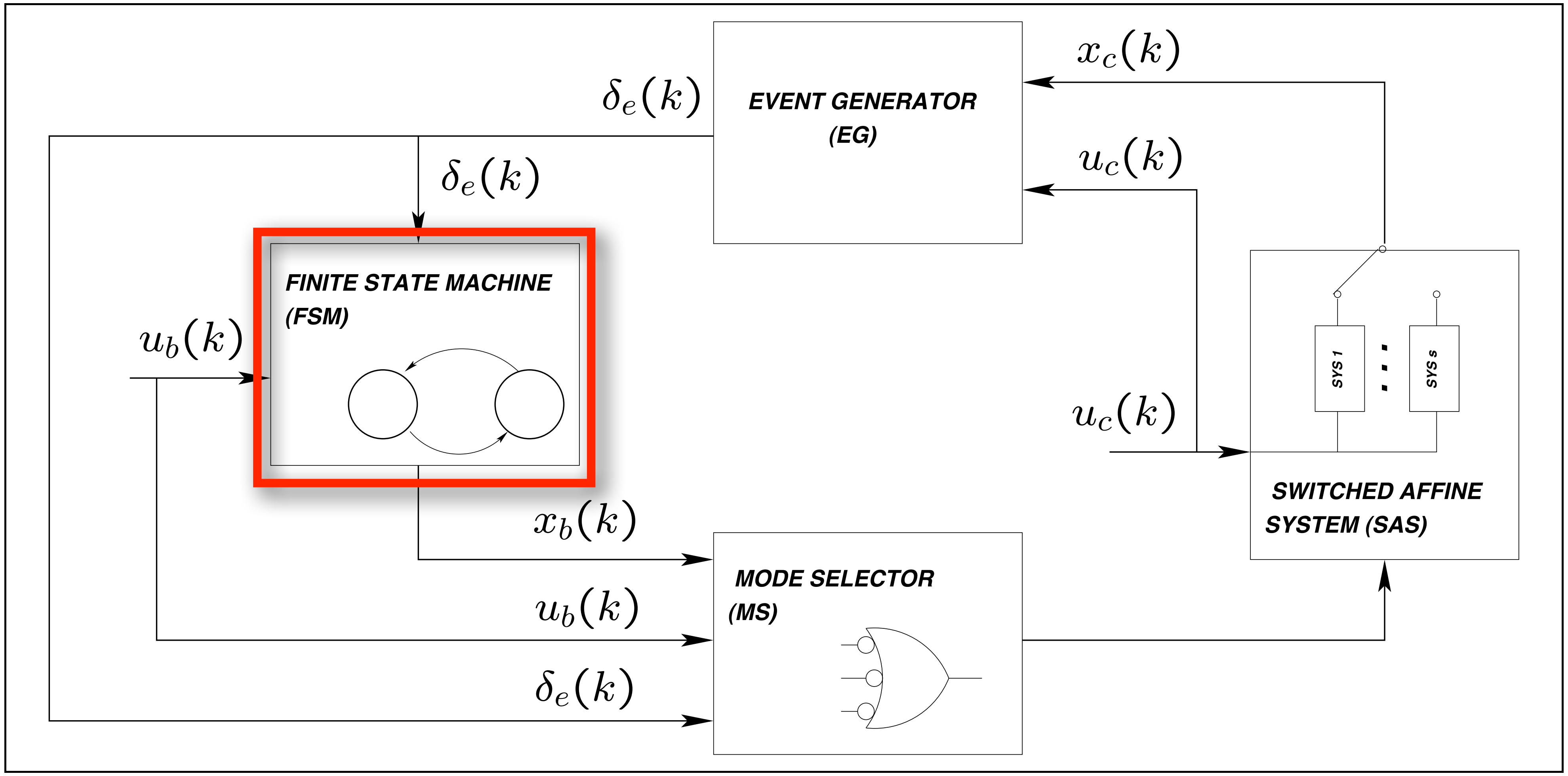


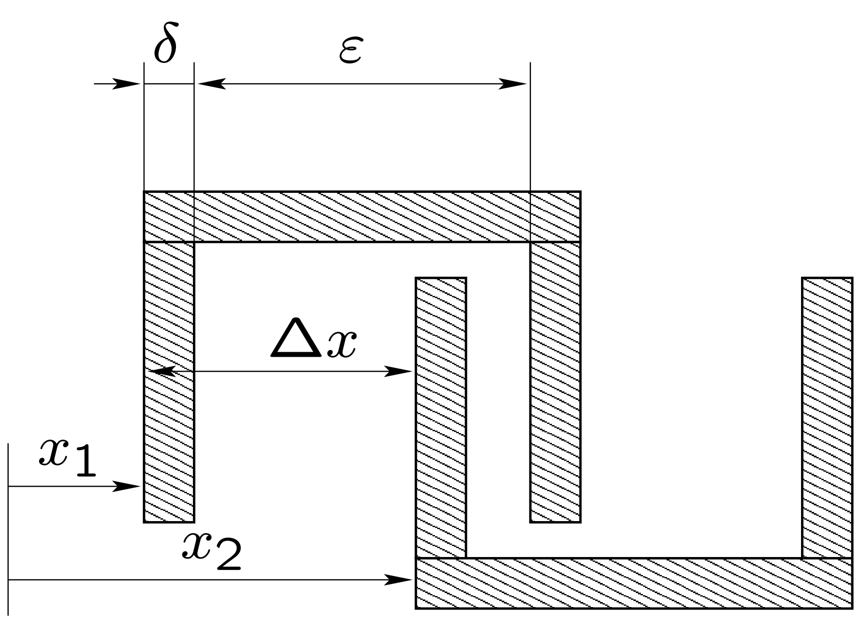
- Contact mode:

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

- Backlash mode

Discrete Hybrid Automata



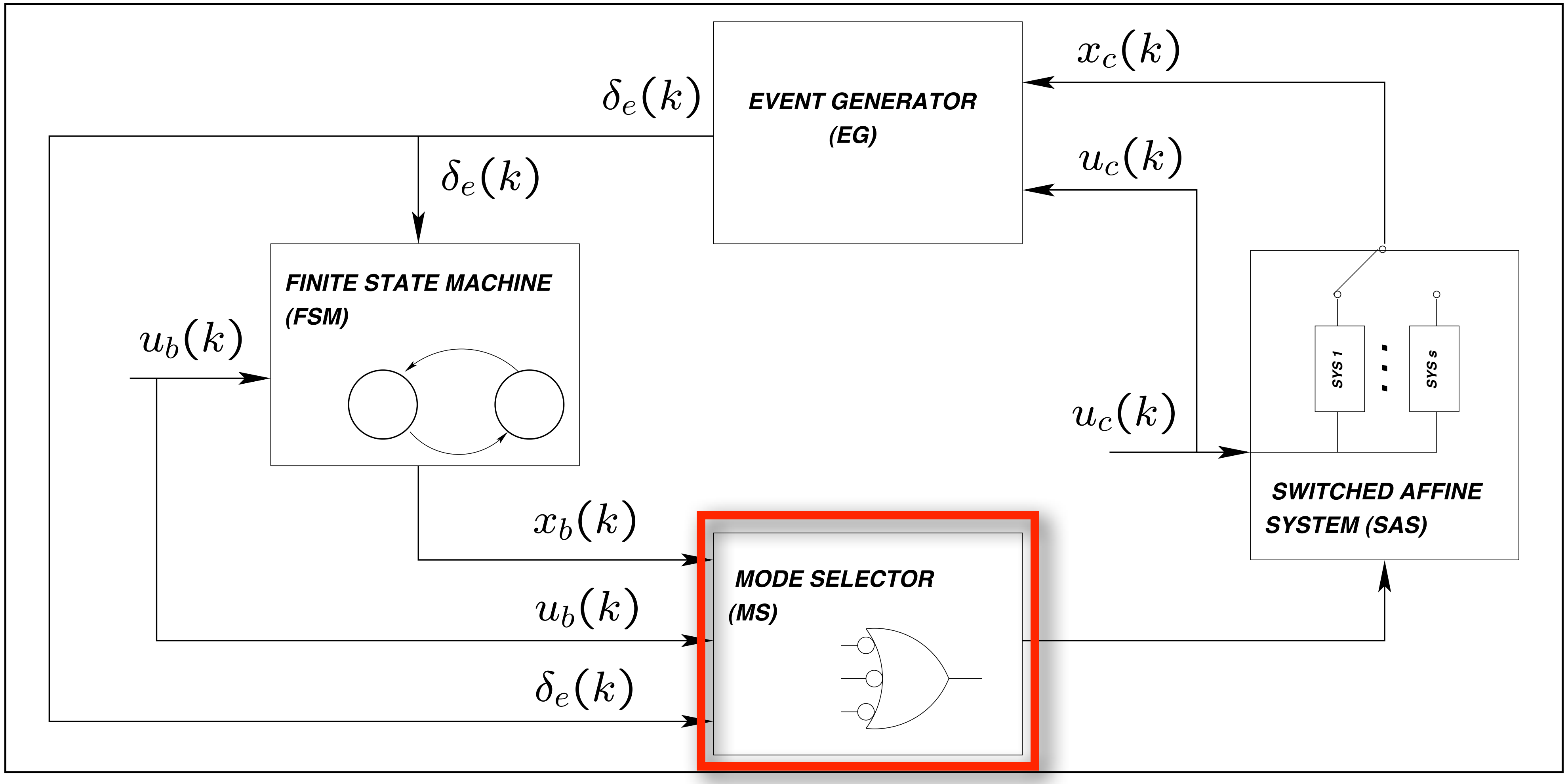


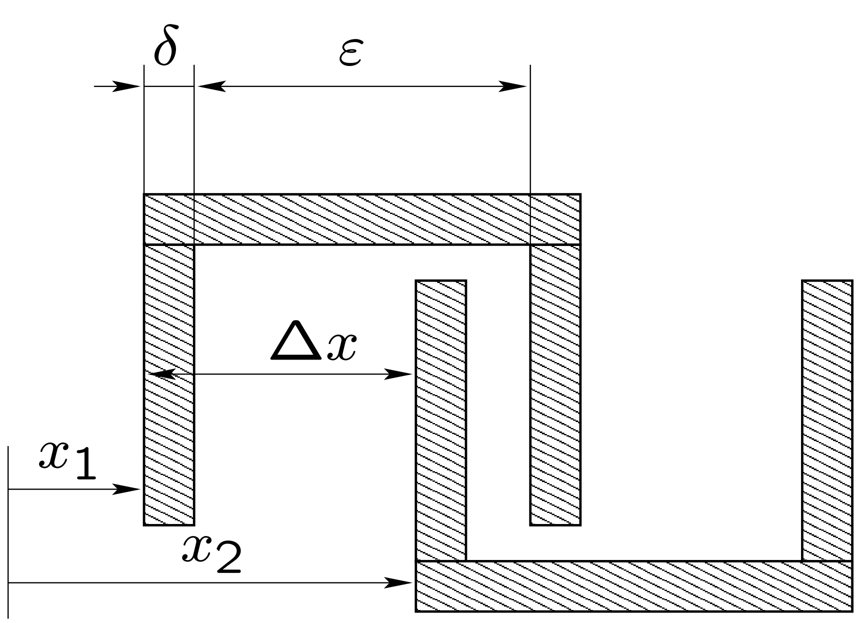
- Contact mode:

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

- Backlash mode

Discrete Hybrid Automata



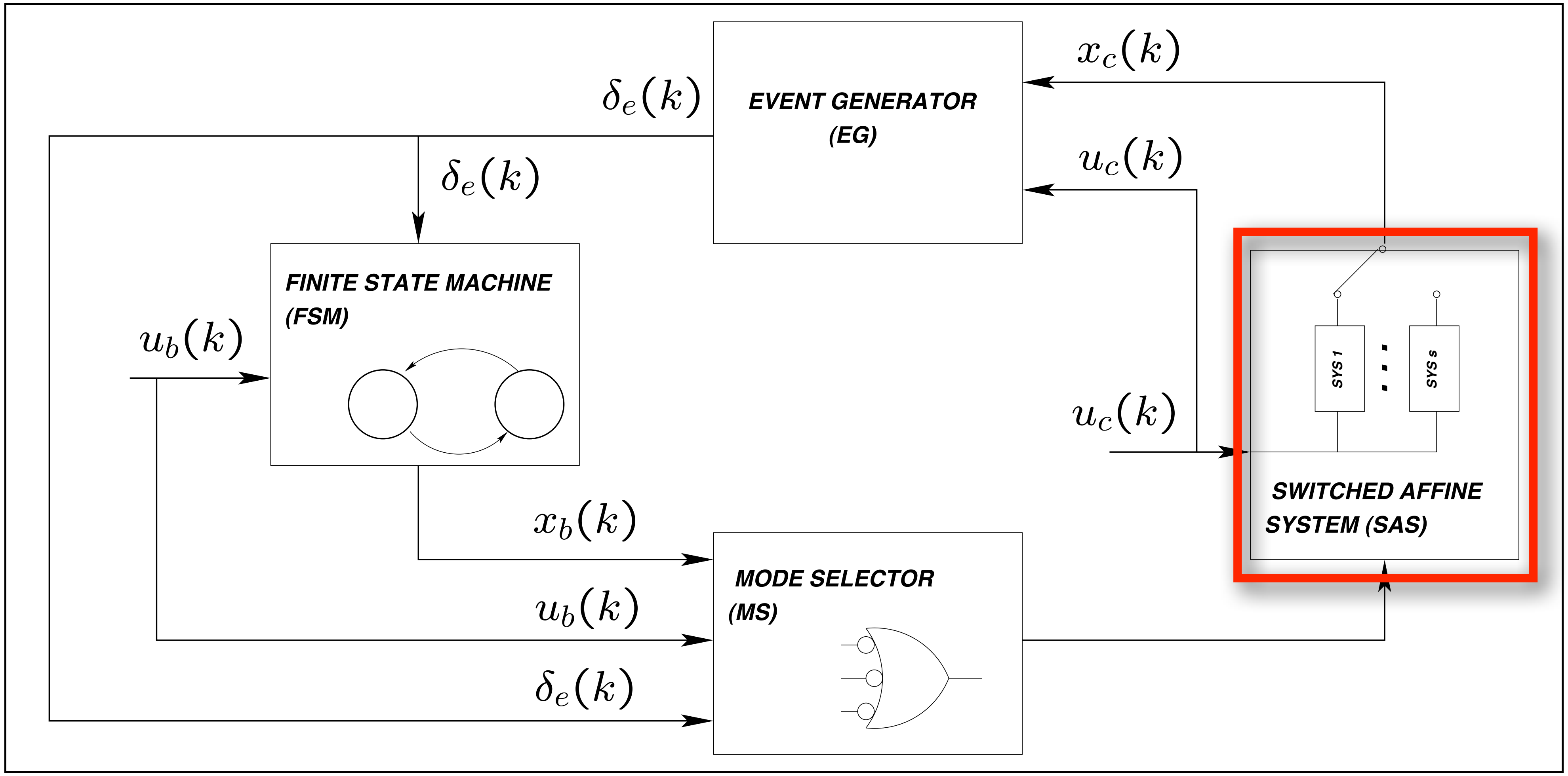


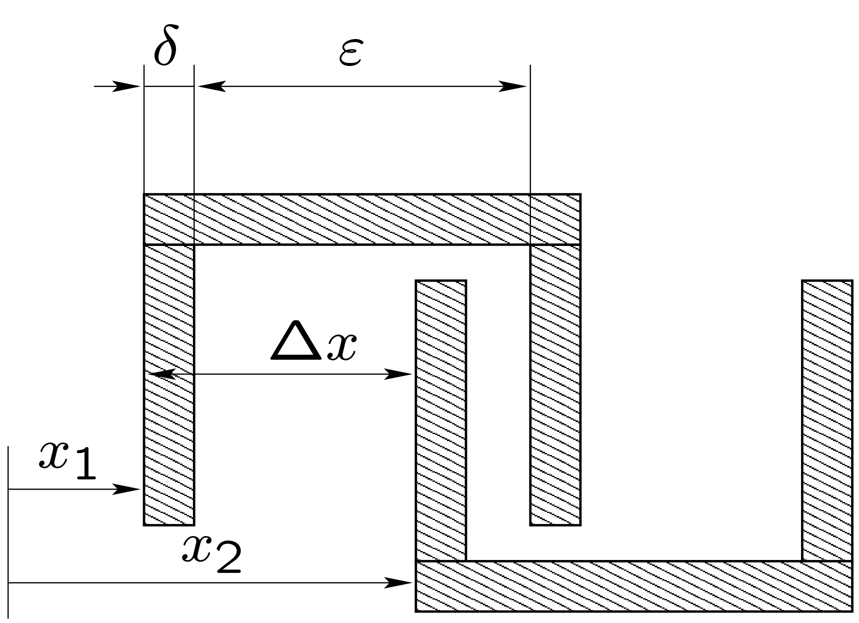
- Contact mode:

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

- 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:
  - convert logic expressions into algebraic expressions
- Examples:

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

Mathematical Modeling of DHAs

- More complex example:

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

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

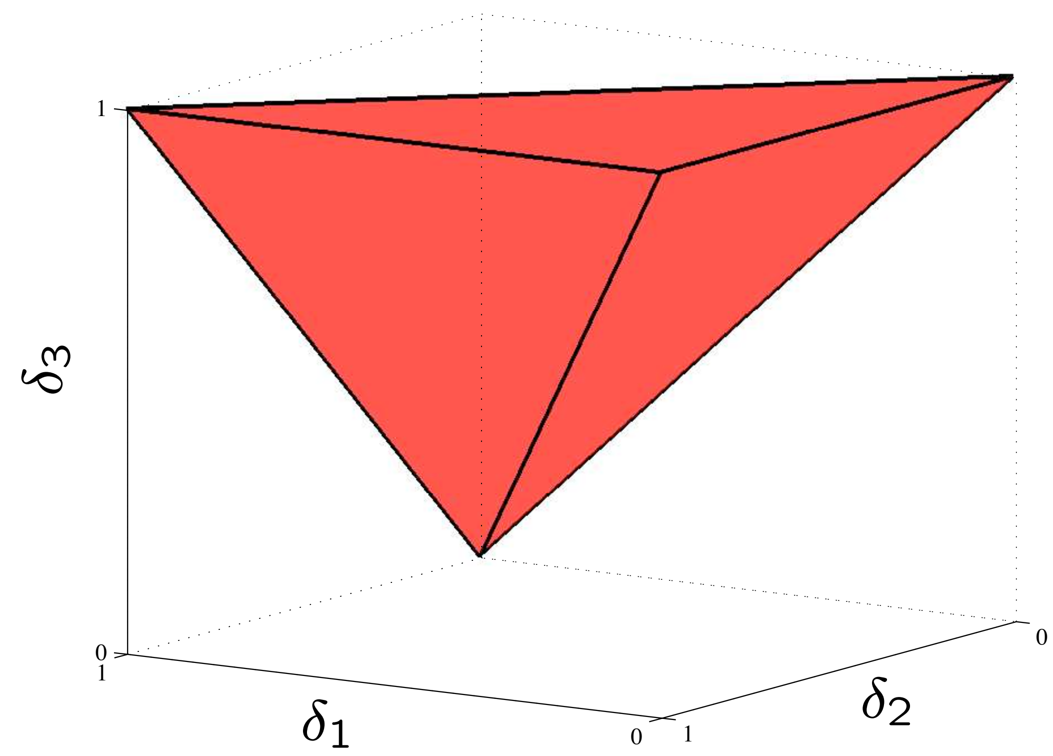
$$\delta_a = (\delta_1 \wedge \delta_2) \;\; \Leftrightarrow \;\; \begin{cases} \delta_a \leq \delta_1 \\ \delta_a \leq \delta_2 \\ \delta_1 + \delta_2 \leq 1 + \delta_a \end{cases}$$

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

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 $z_i = f_i(x)\delta_i$

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:

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

- Associate $(\delta(t) = 1) \Leftrightarrow (x(t) \leq 0)$
- 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:

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

1. Discrete Hybrid Automata

2. HYSDEL

3. Piecewise Affine Models

4. MPC for Hybrid Systems

5. Closing Remarks

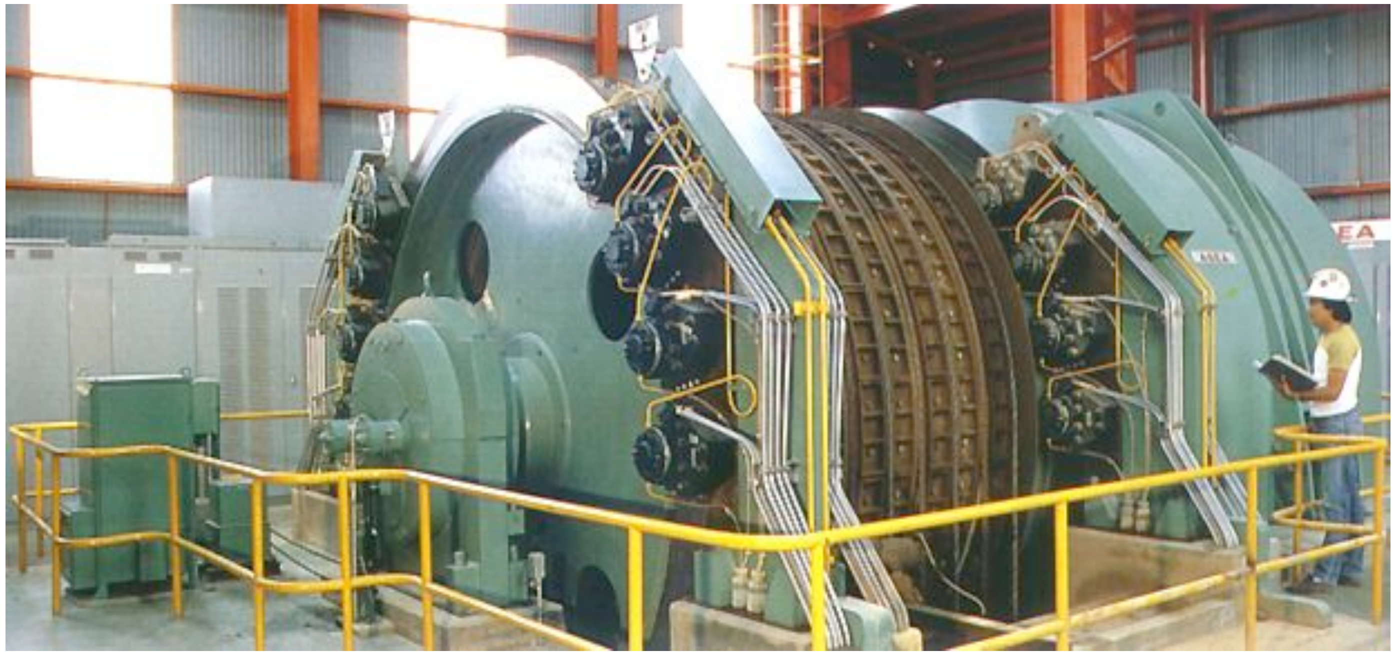
HYbrid Systems DEscription Language (HYSDEL)


```
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 */
```

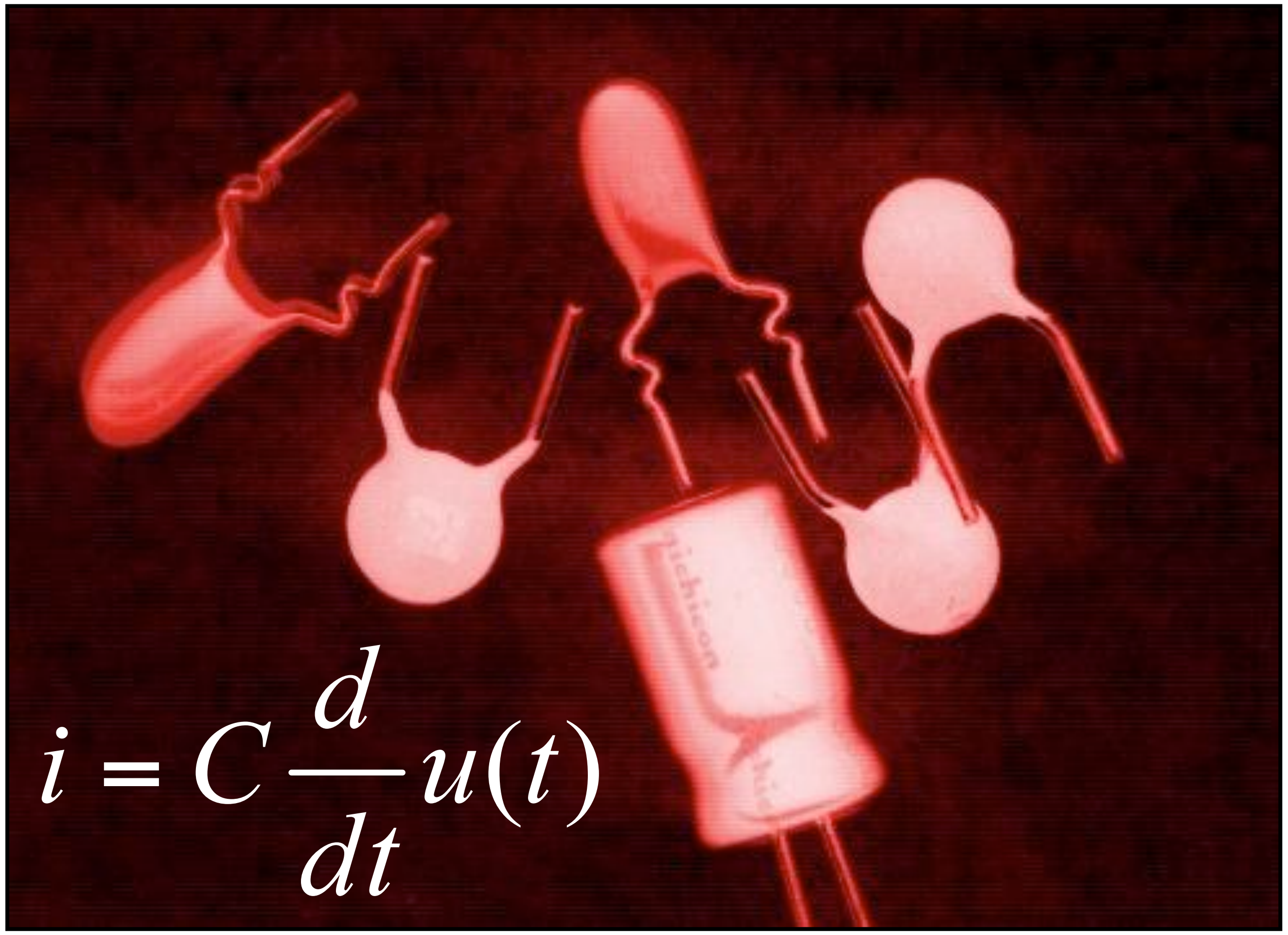
Logic Expressions

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

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

```
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

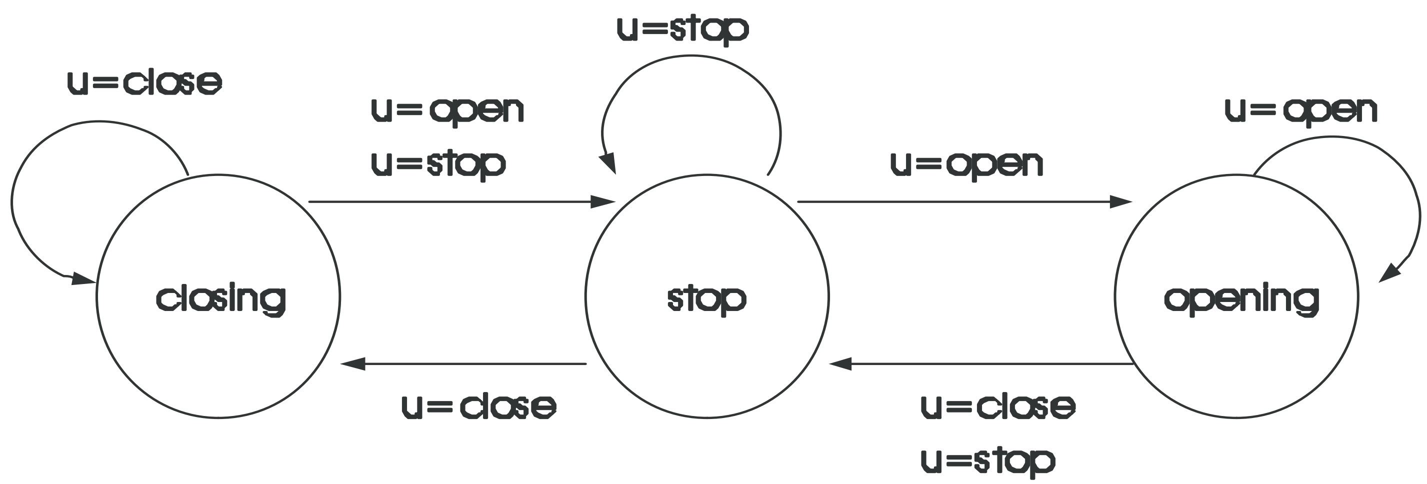


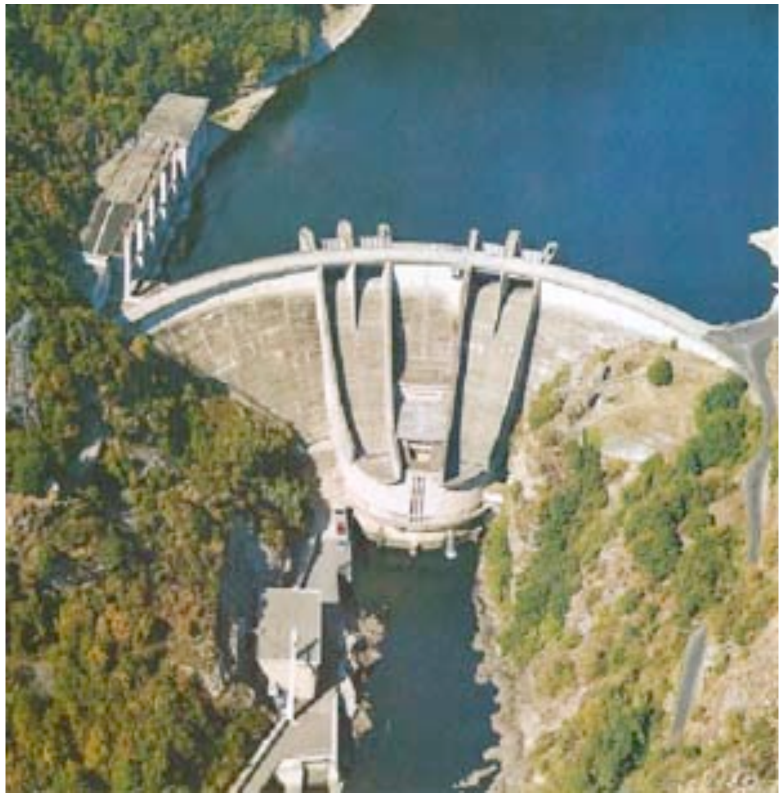

```
SYSTEM capacitorD {
   INTERFACE {
      STATE {
         REAL u; }
      PARAMETER {
         REAL R = 1e4;
         REAL C = 1e-4;
         REAL T = 1e-1; }
   } /* end interface */

   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

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

```
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/

1. Discrete Hybrid Automata

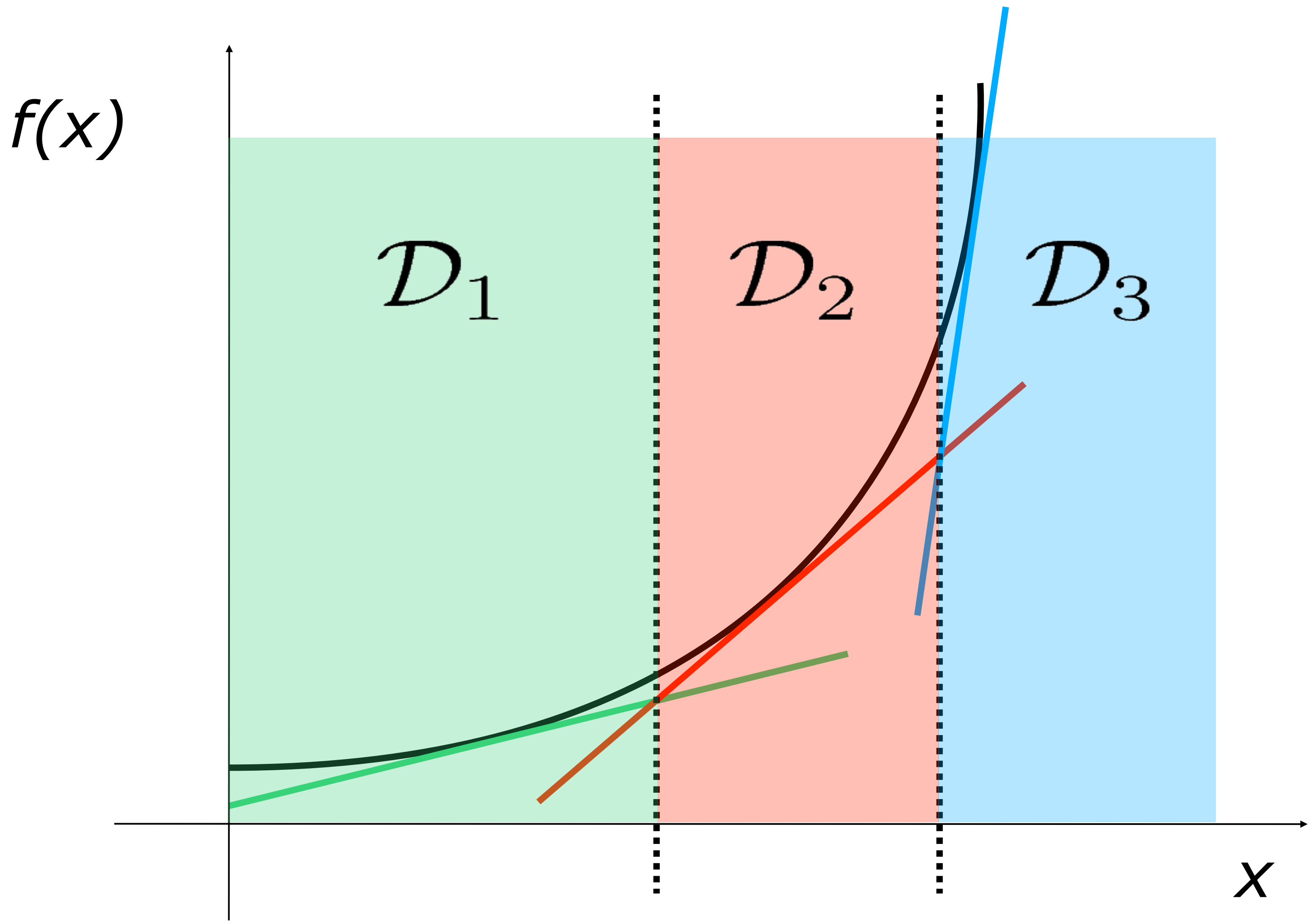
2. HYSDEL

3. Piecewise Affine Models

4. MPC for Hybrid Systems

5. Closing Remarks

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
- Under mild assumptions one can convert from MLD to PWA representation and vice versa

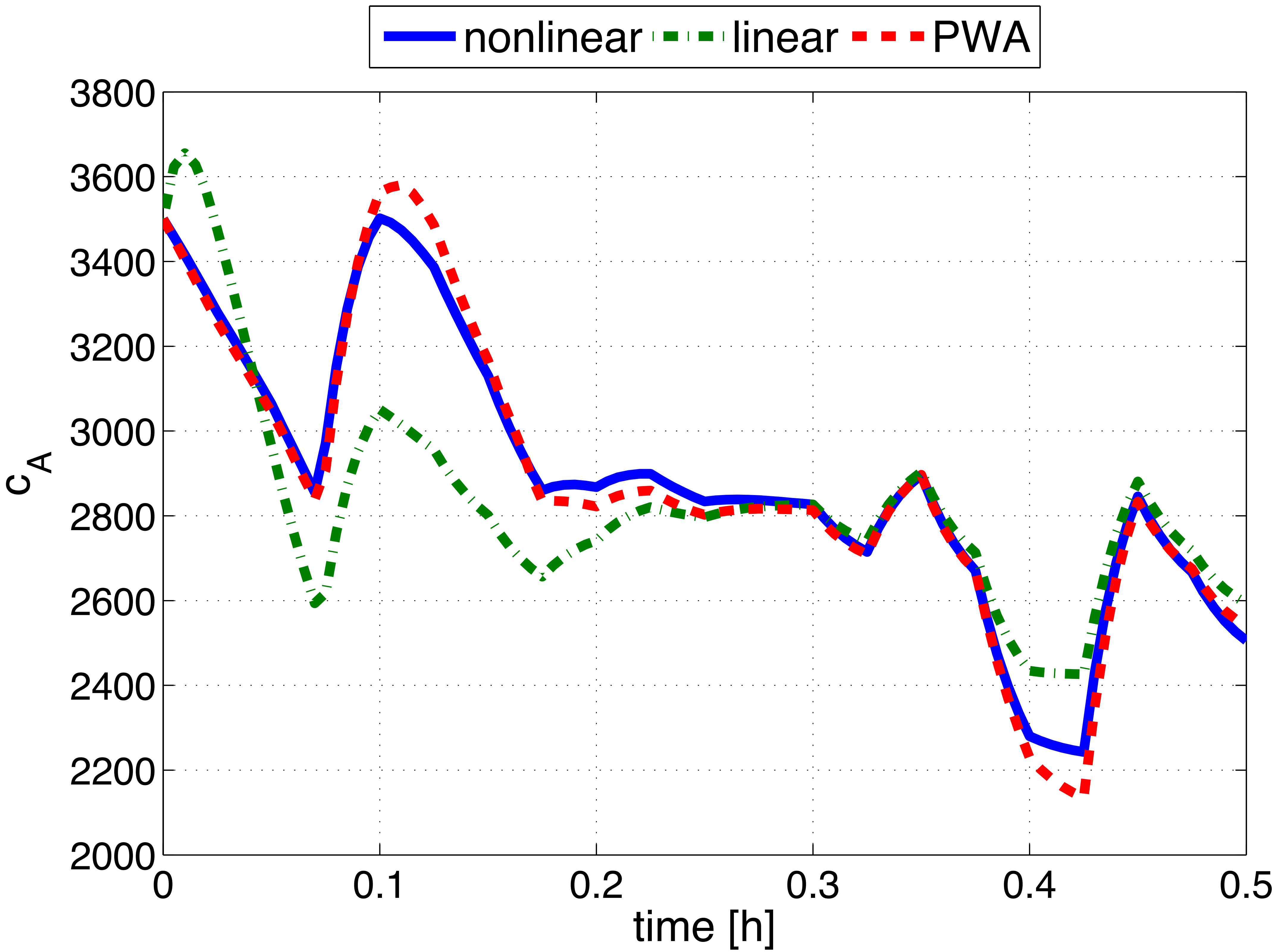
Case Study: CSTR

- Nasty nonlinear dynamics

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

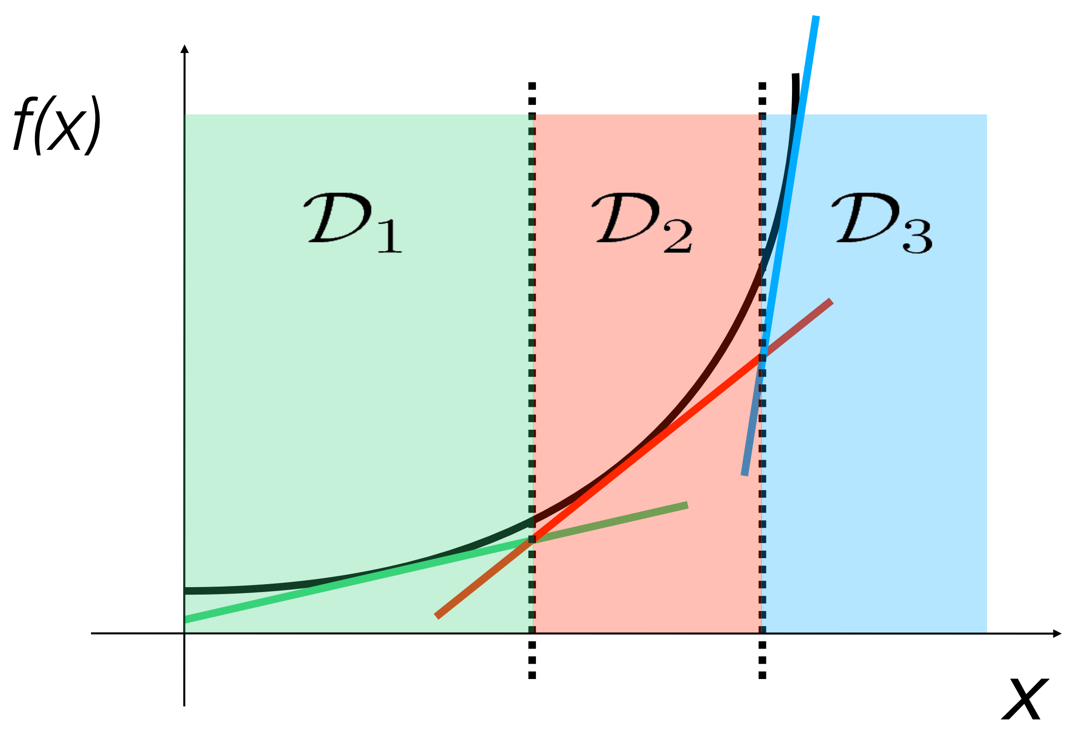
- Constraints on states and inputs
- Approximated by a PWA system with 32 local linearizations

Case Study: CSTR

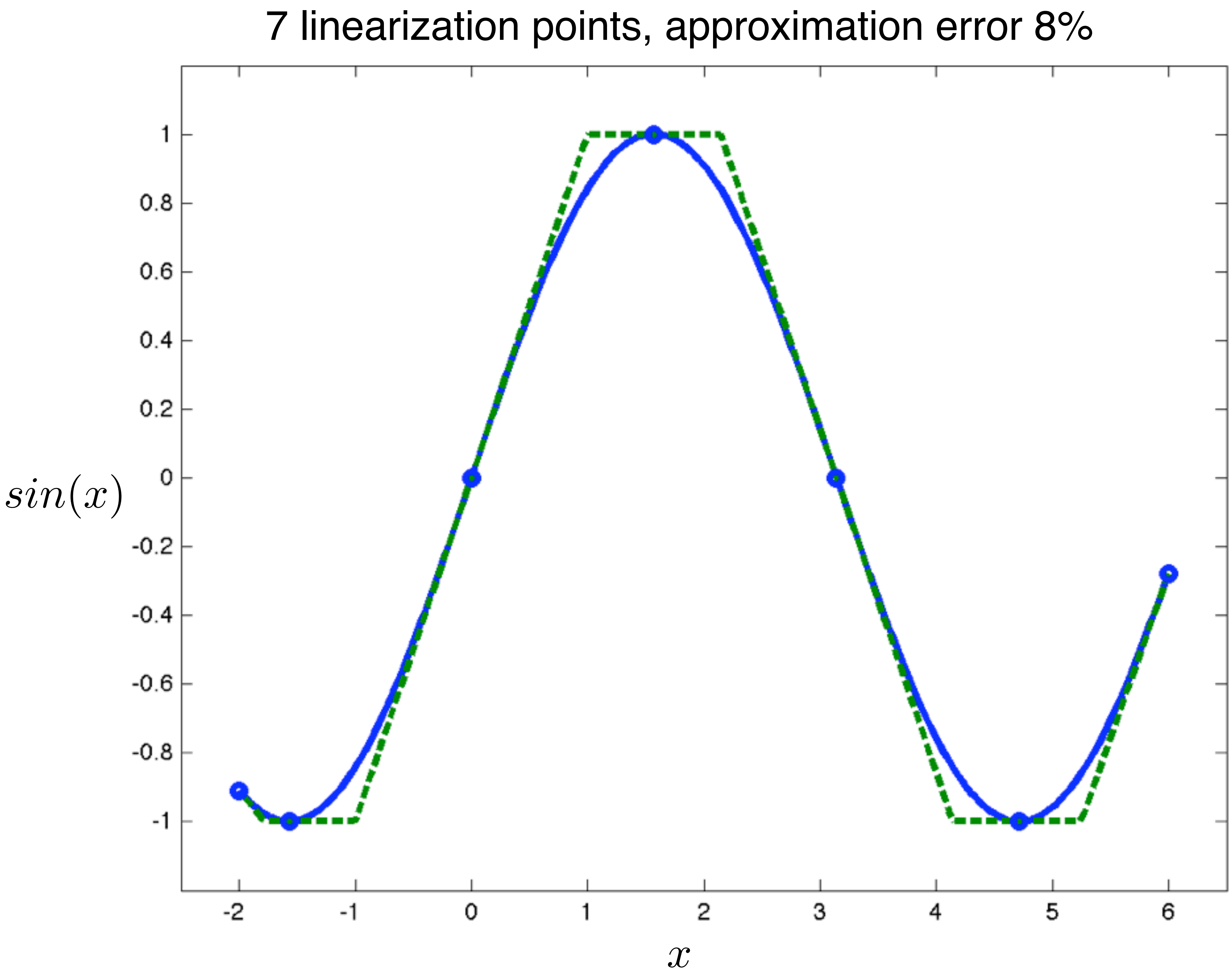


Obtaining PWA Models

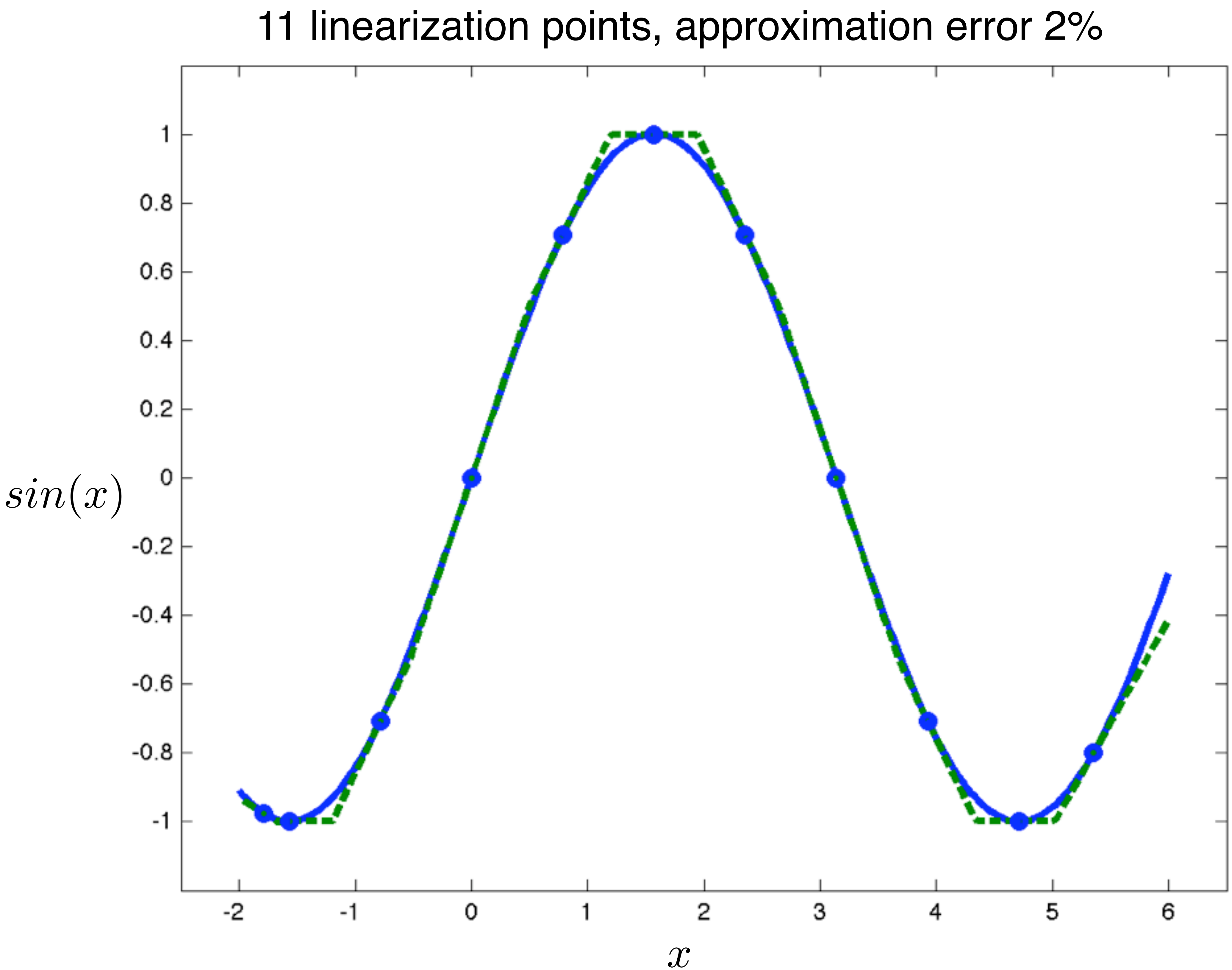
- The process of obtaining a PWA approximation of a nonlinear function includes:
  - selection of suitable linearization points
  - calculation of corresponding local linearization
  - determination of regions of validity
- Bottom line: easy to do in 1D, difficult in 2D, impossible in higher dimensions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



Automatic Multiple Linearization of 1D Functions

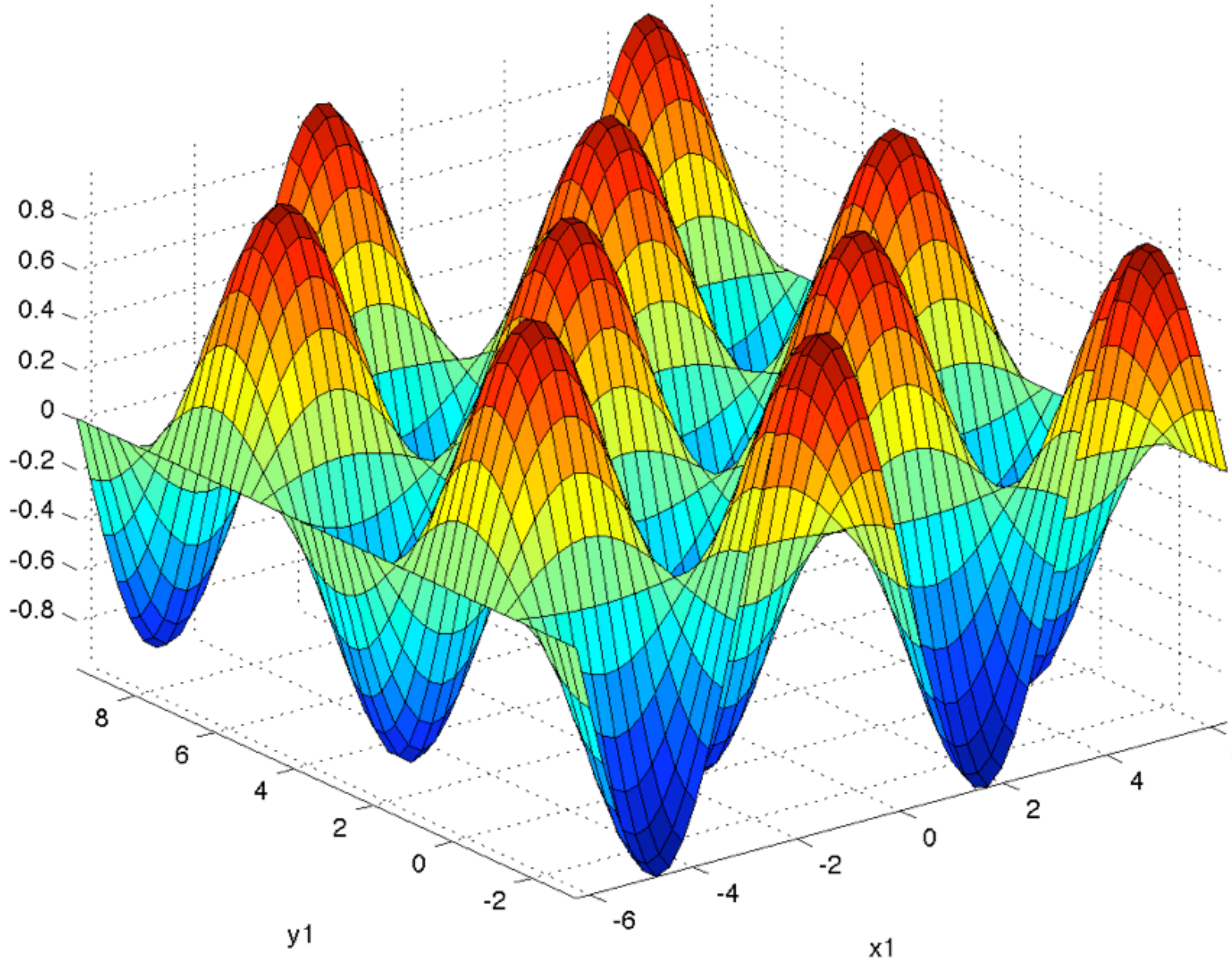


Automatic Multiple Linearization of 1D Functions



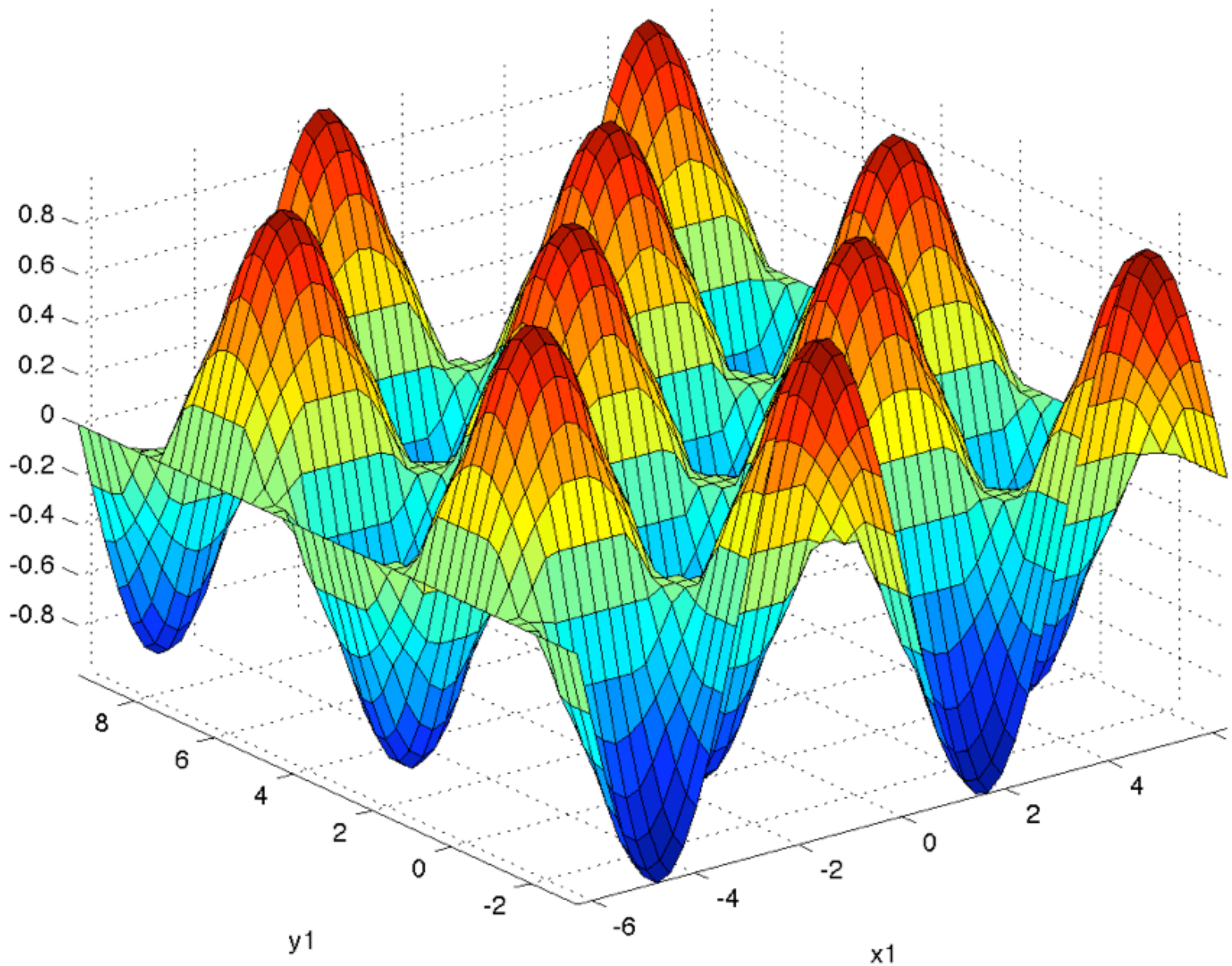
Automatic Multiple Linearization of 2D Functions

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



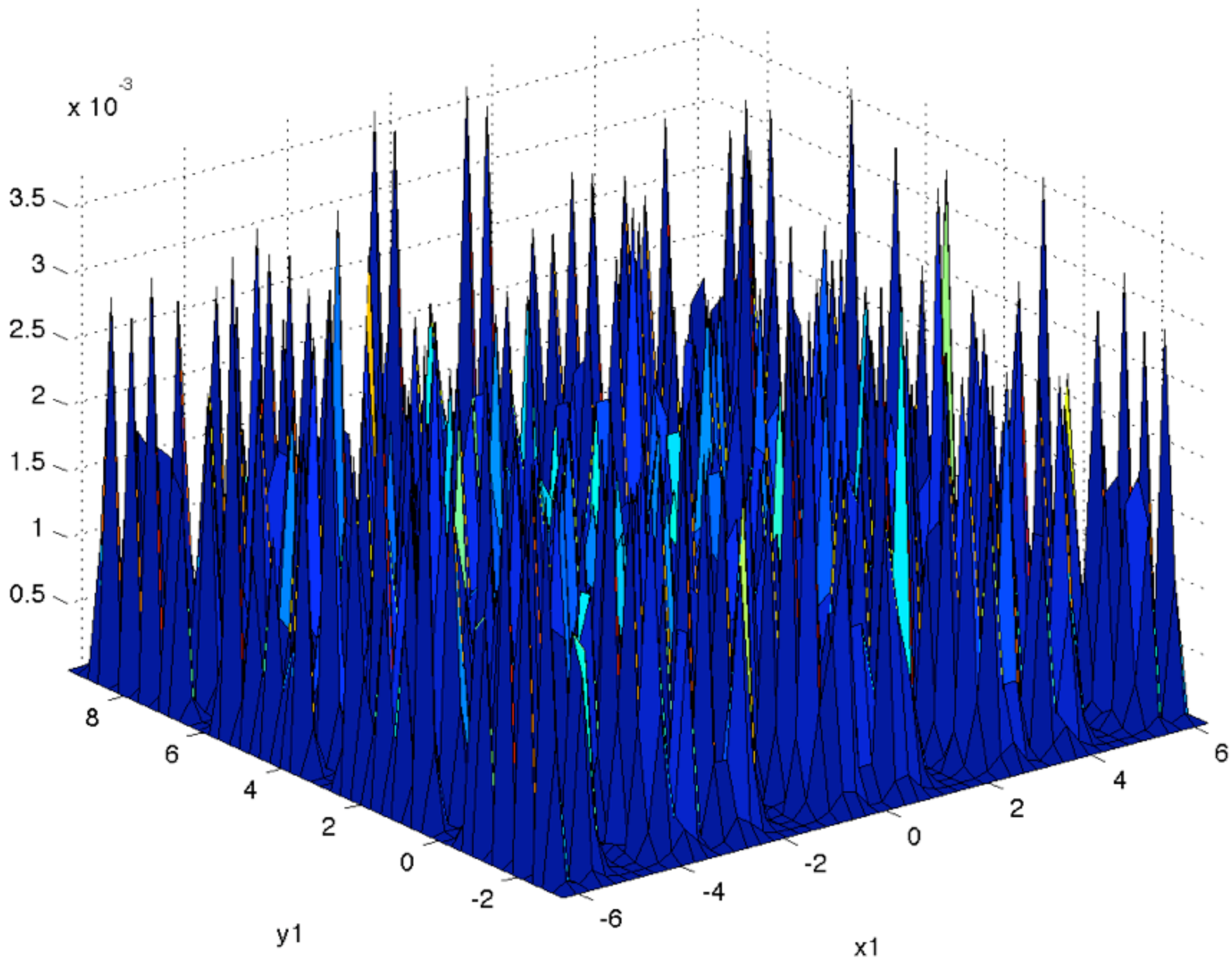
Automatic Multiple Linearization of 2D Functions

PWA approximation using 10 linearizations



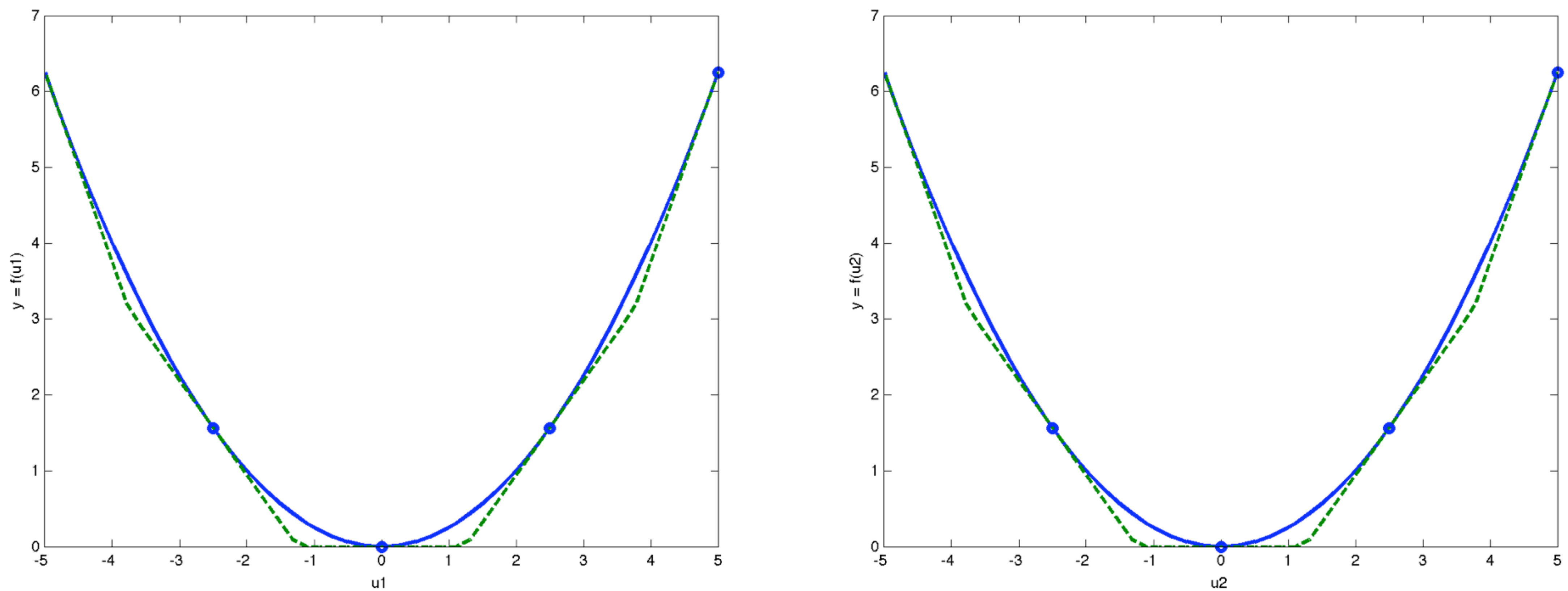
Automatic Multiple Linearization of 2D Functions

Approximation error < 0.1 %



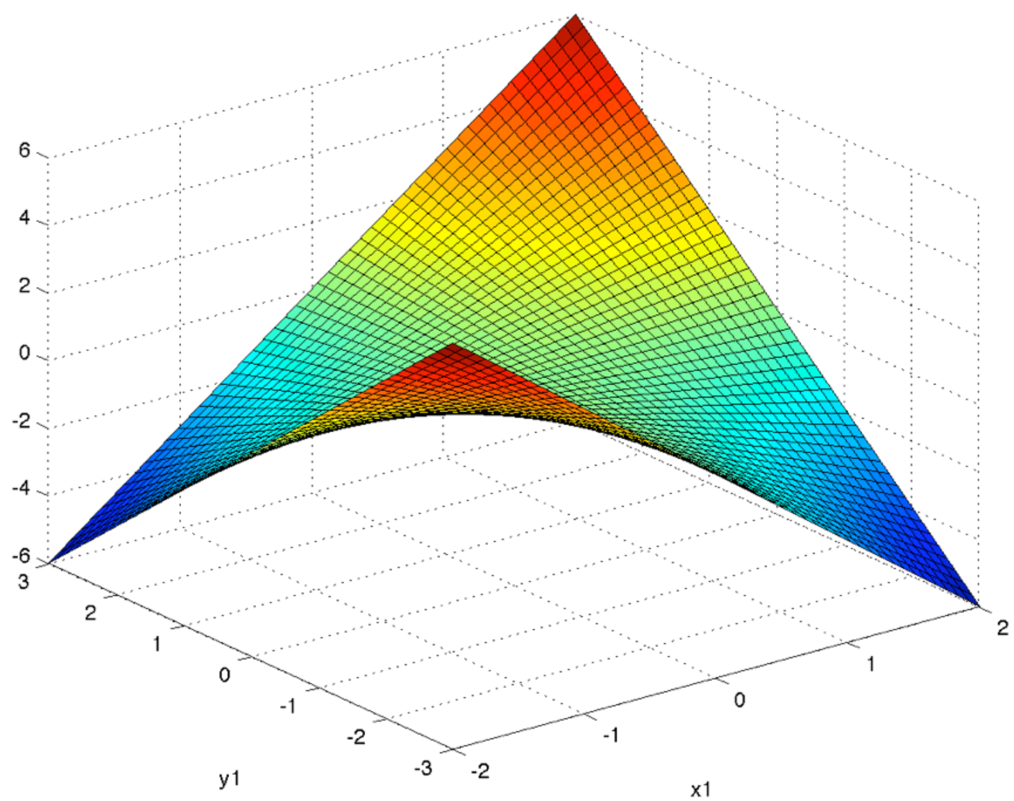
The Theory Behind

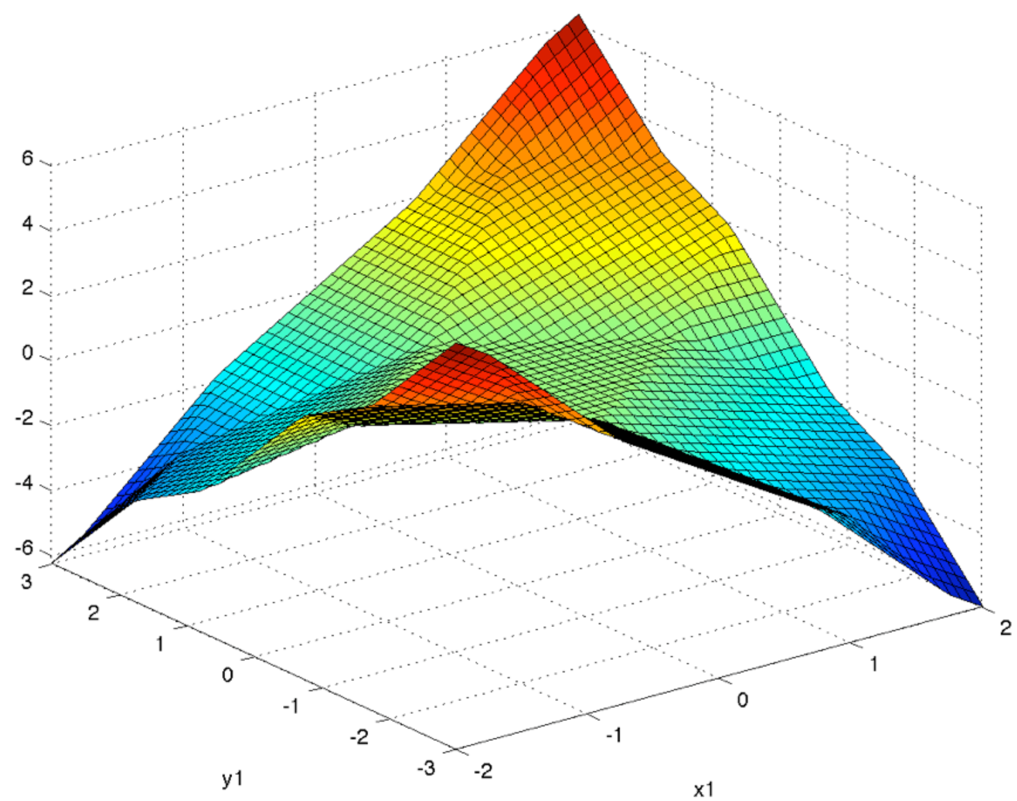
- Consider a product of two variables $f = x_1 x_2$
- Define two auxiliary variables $u_1 = (x_1 + x_2),\ u_2 = (x_1 - x_2)$
- Observe the equivalence: $f = \frac{1}{4}(u_1^2 - u_2^2)$
- Now we have a difference of two nonlinear 1D functions, hence we are back to the 1D scenario



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



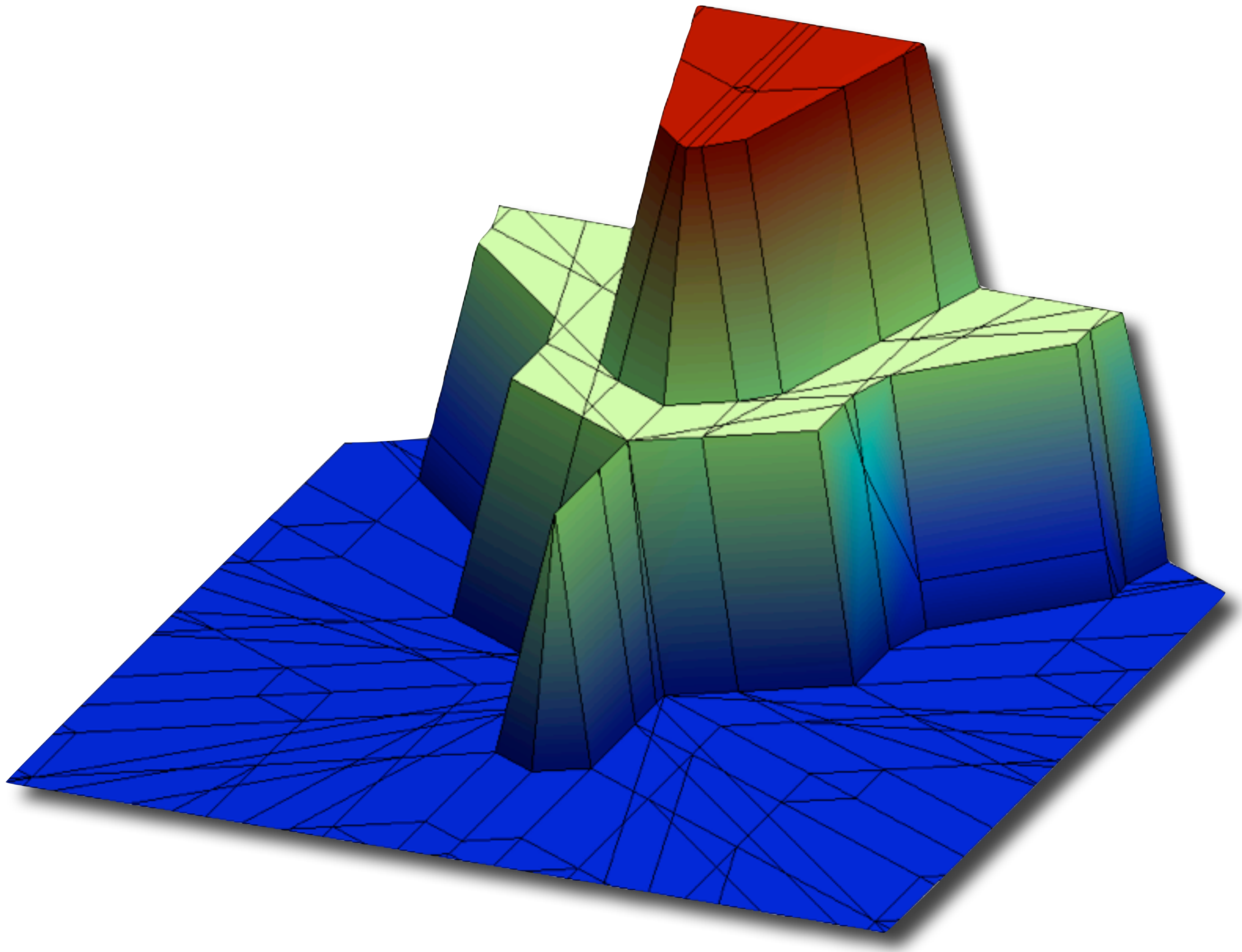


AUTOPROX: Automatic PWA Approximation Toolbox

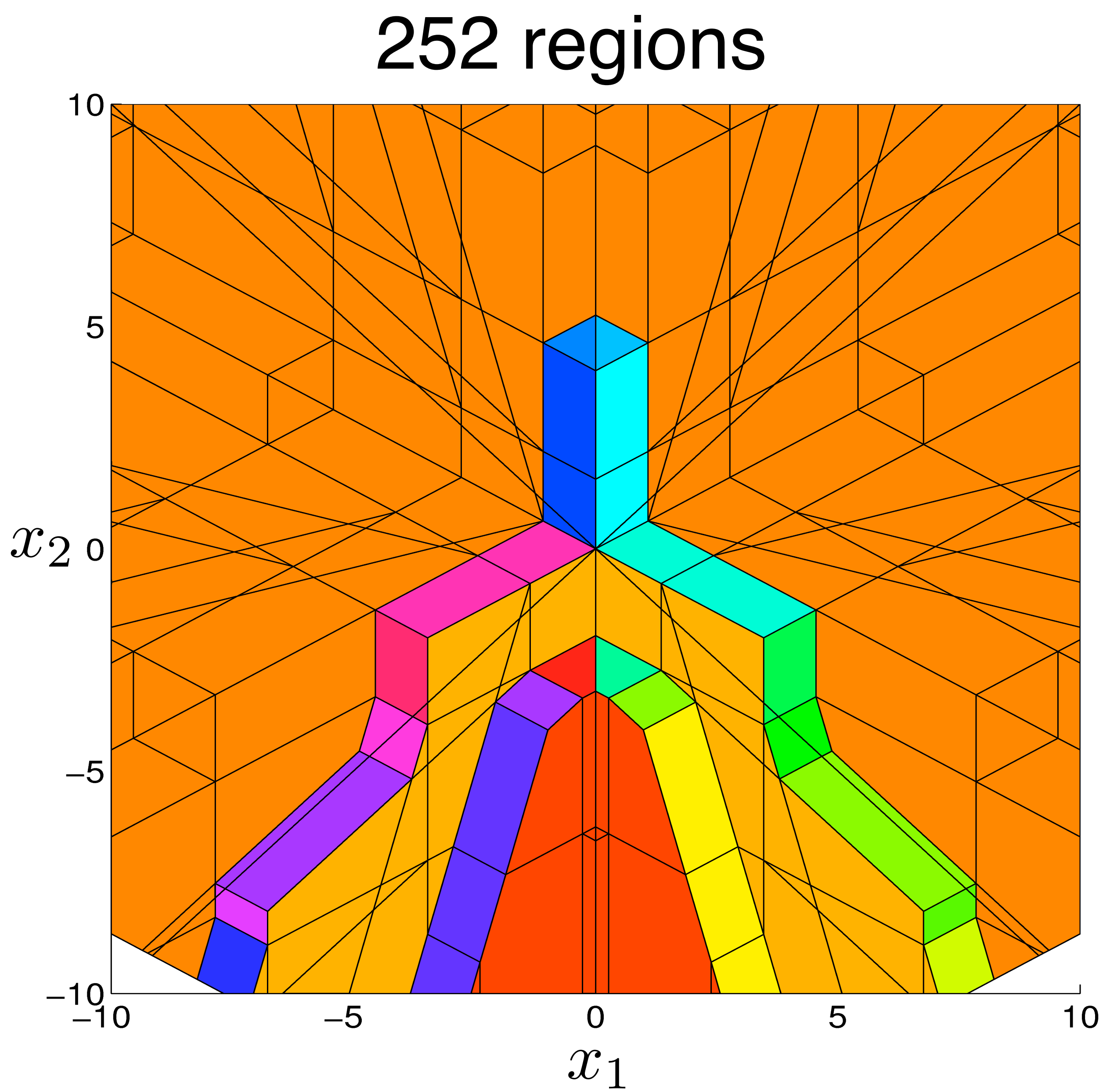
- http://www.kirp.chtf.stuba.sk/~sw/
- Inputs:
  - symbolic representation of an arbitrary nonlinear function, e.g.

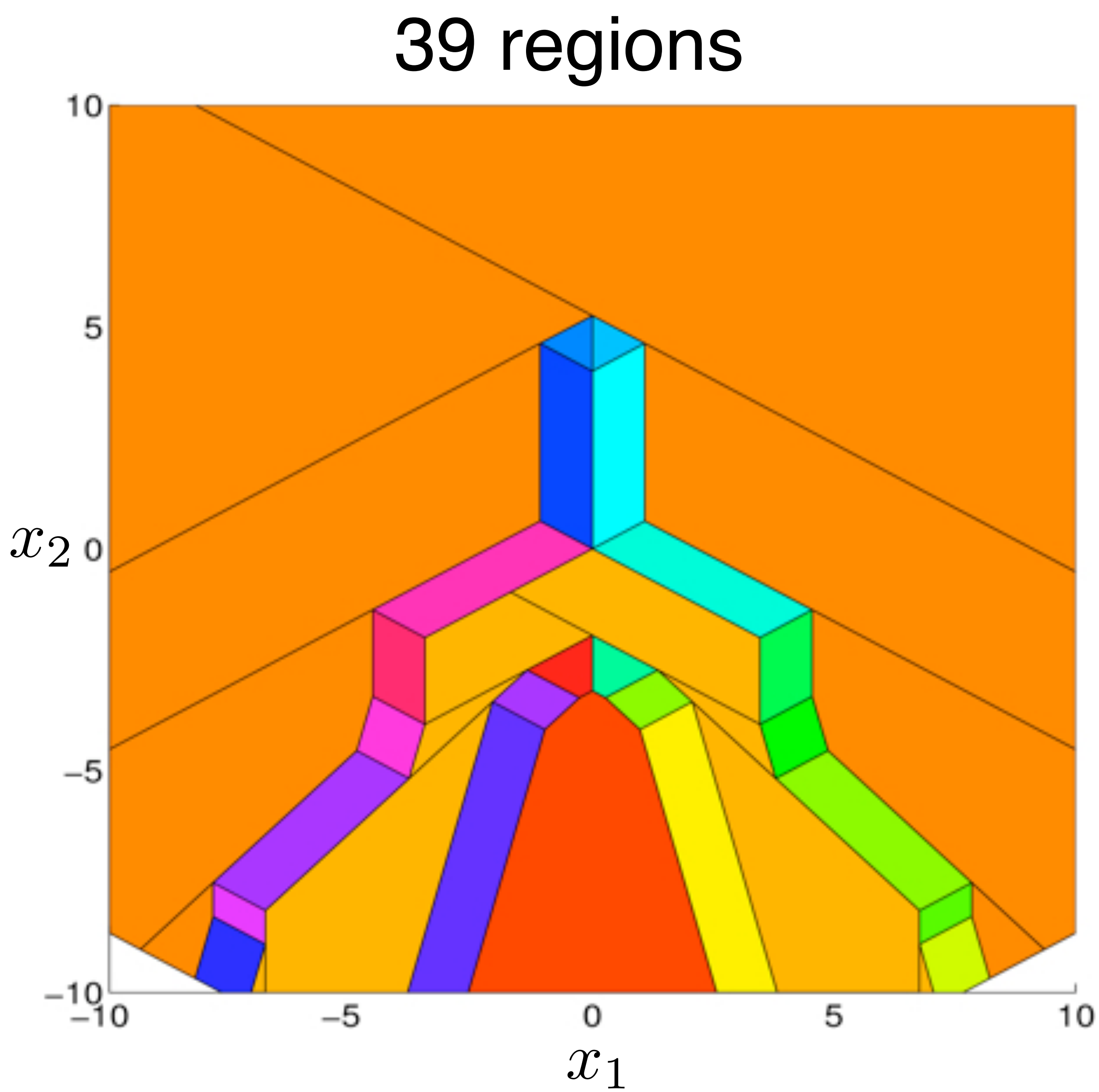
$$\sin(x_1^2 + \exp(1/x_2))(x_3 - \cos(|x_4|))$$

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

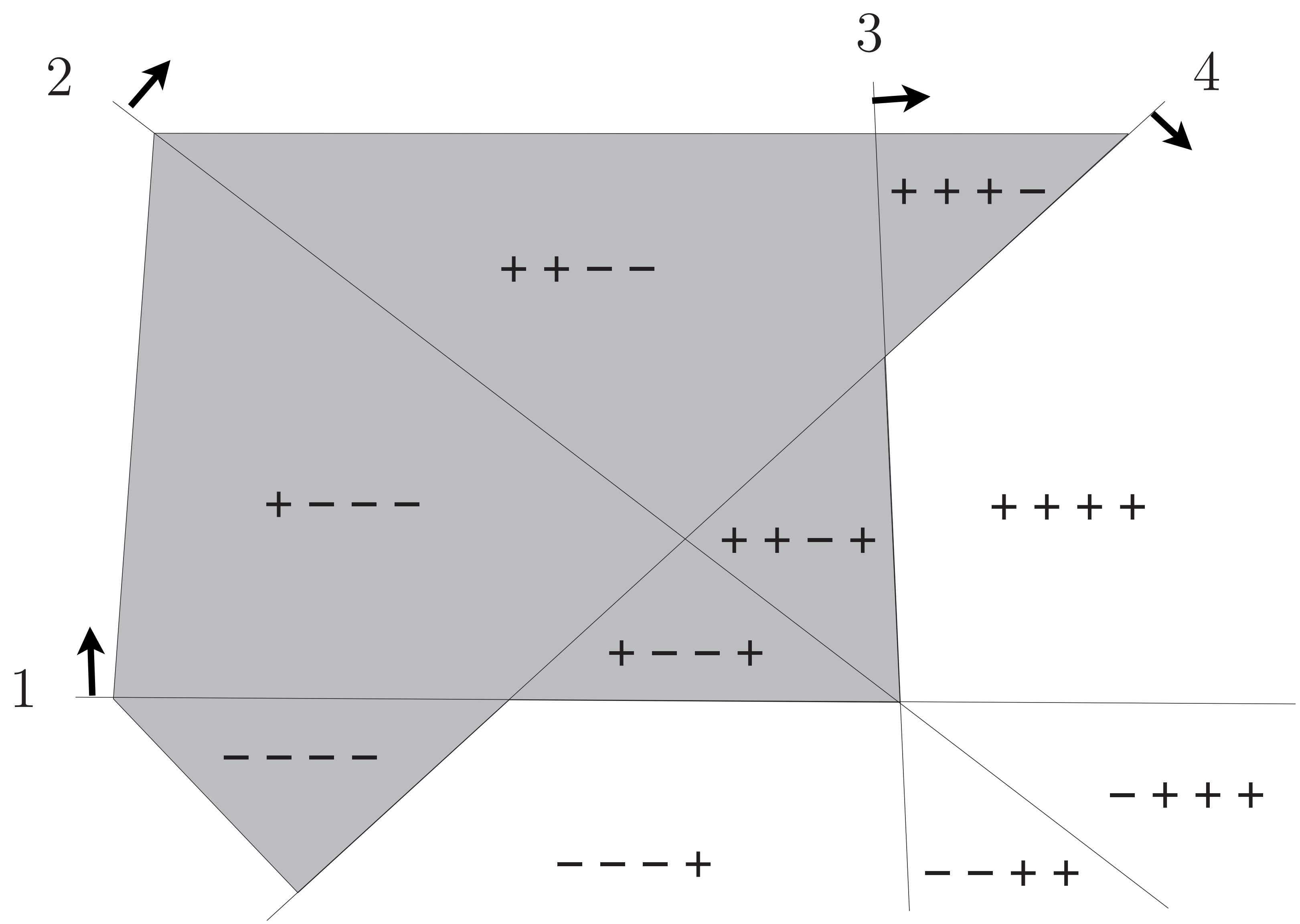


Simplification of PWA Functions

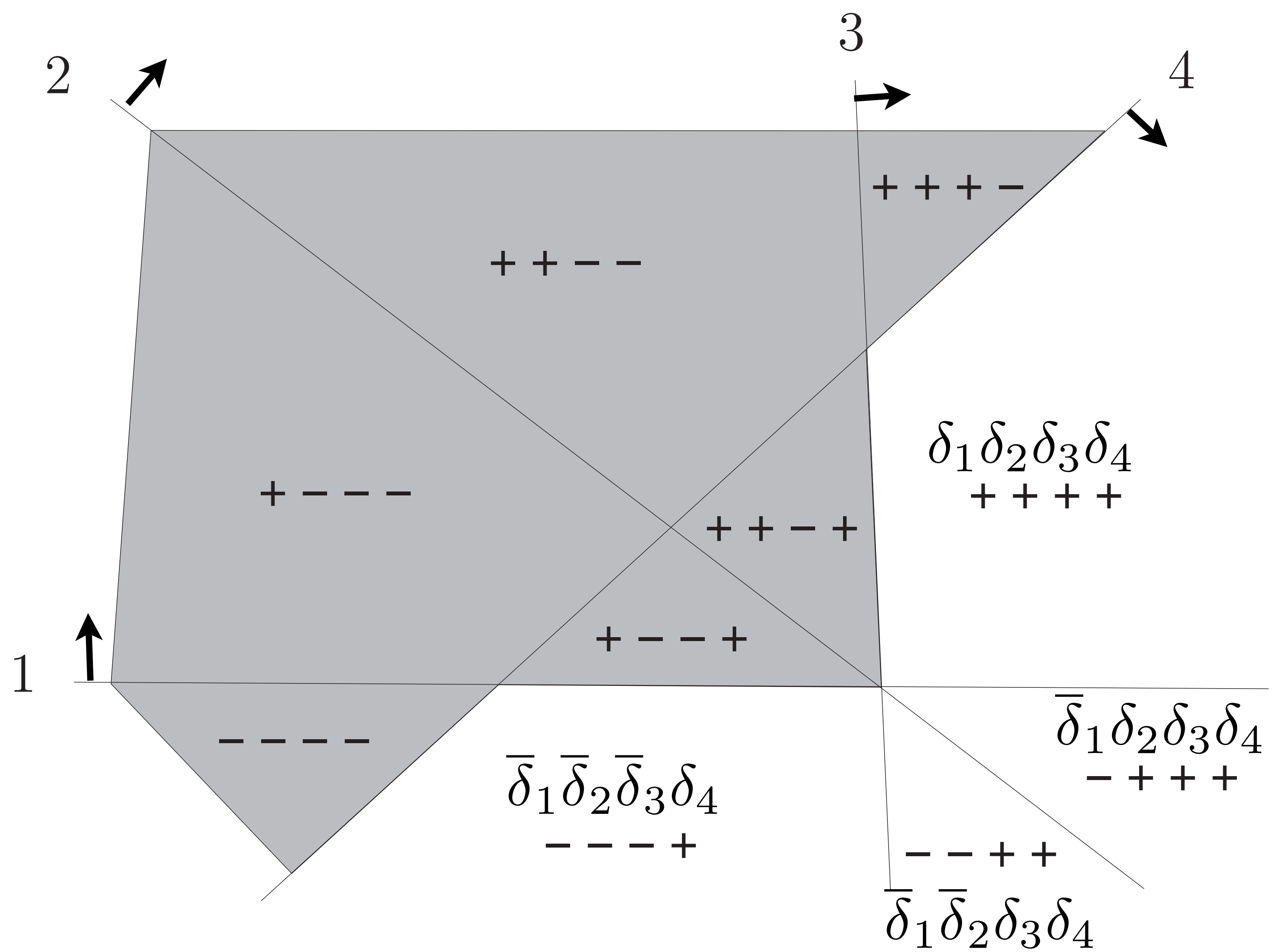




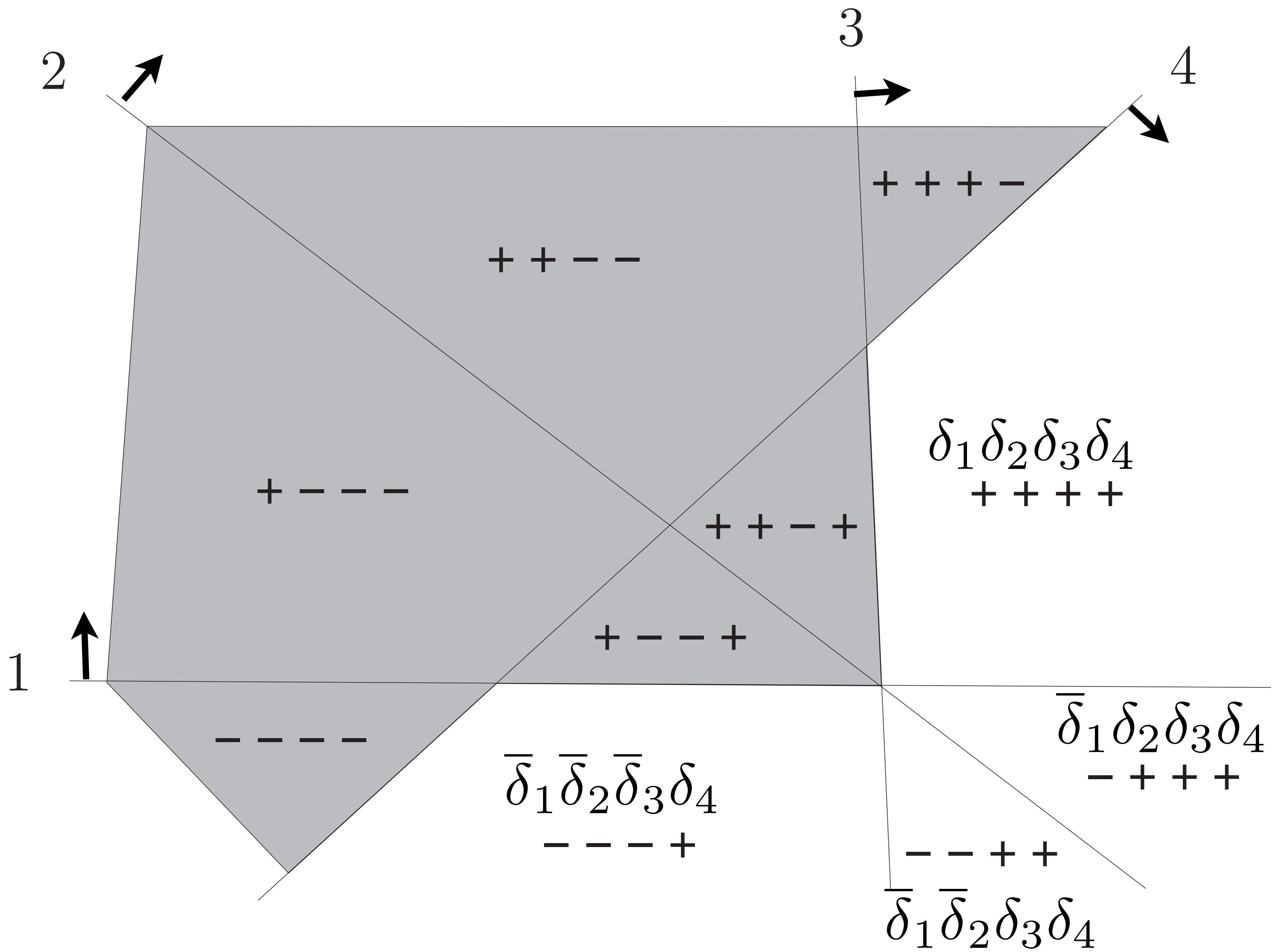
Step 1: Hyperplane Arrangement



Step 2: Associate Boolean Literals

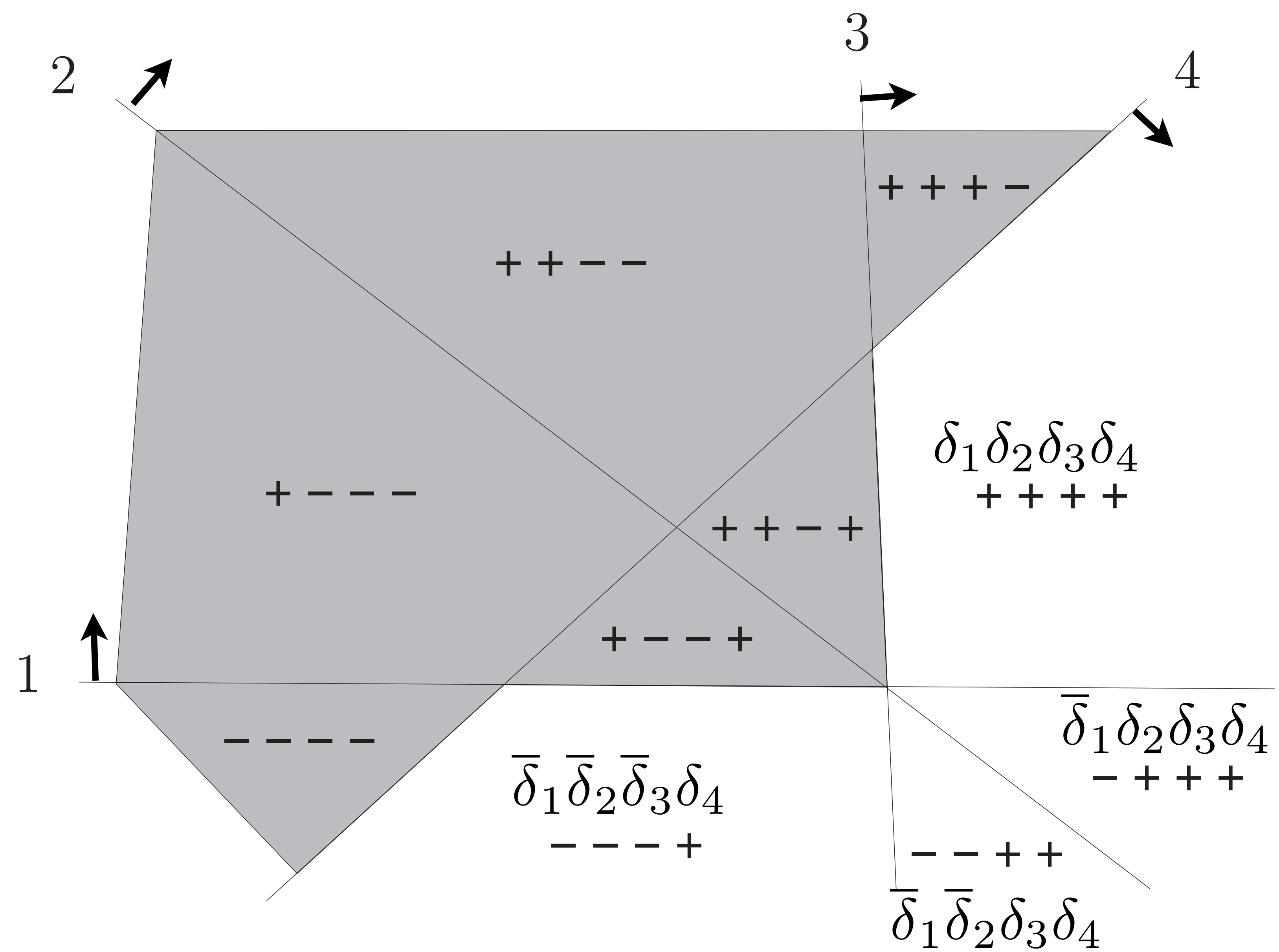


Step 3: Represent Regions to Merge by Logic Functions



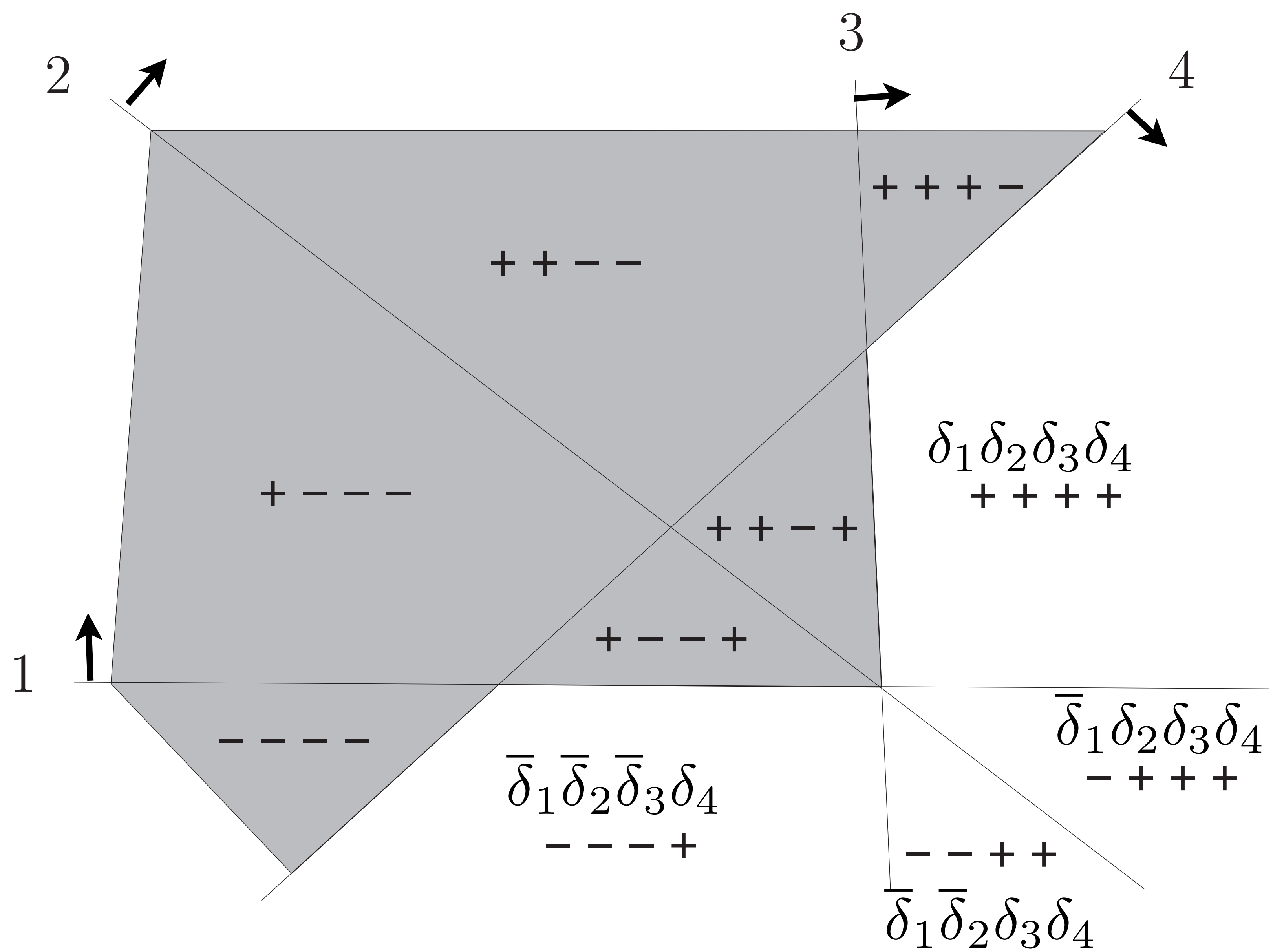
$$\text{White regions} = \overline{\delta}_1\overline{\delta}_2\overline{\delta}_3\delta_4 + \overline{\delta}_1\overline{\delta}_2\delta_3\delta_4 + \overline{\delta}_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4$$

Step 4: Simplify the Function



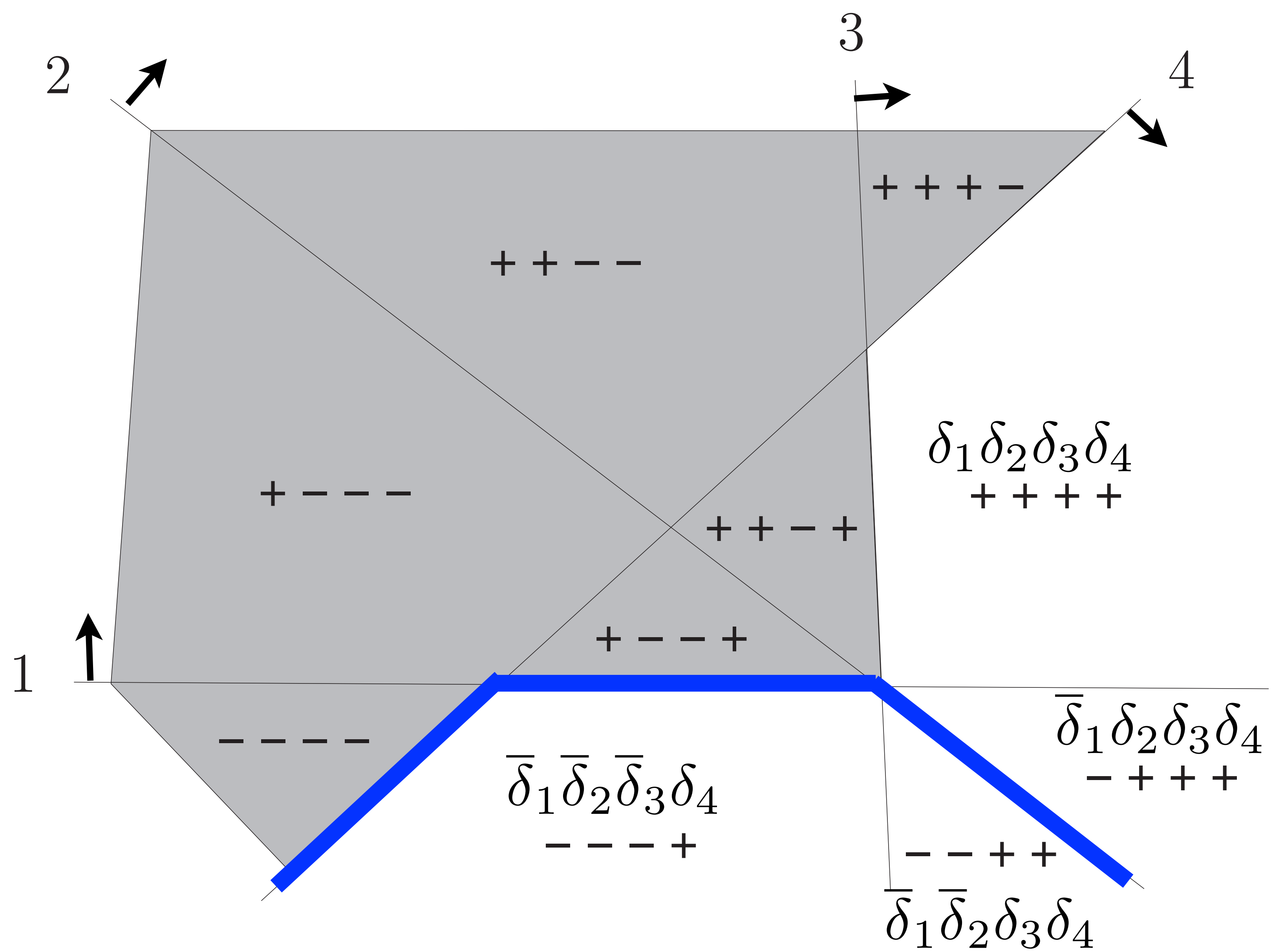
$$
\begin{aligned}
\text{White regions} &= \overline{\delta}_1\overline{\delta}_2\overline{\delta}_3\delta_4 + \overline{\delta}_1\overline{\delta}_2\delta_3\delta_4 + \overline{\delta}_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4 \\
&= \overline{\delta}_1\overline{\delta}_2\delta_4(\delta_3 + \overline{\delta}_3) + \delta_2\delta_3\delta_4(\delta_1 + \overline{\delta}_1)
\end{aligned}
$$

Step 4: Simplify the Function



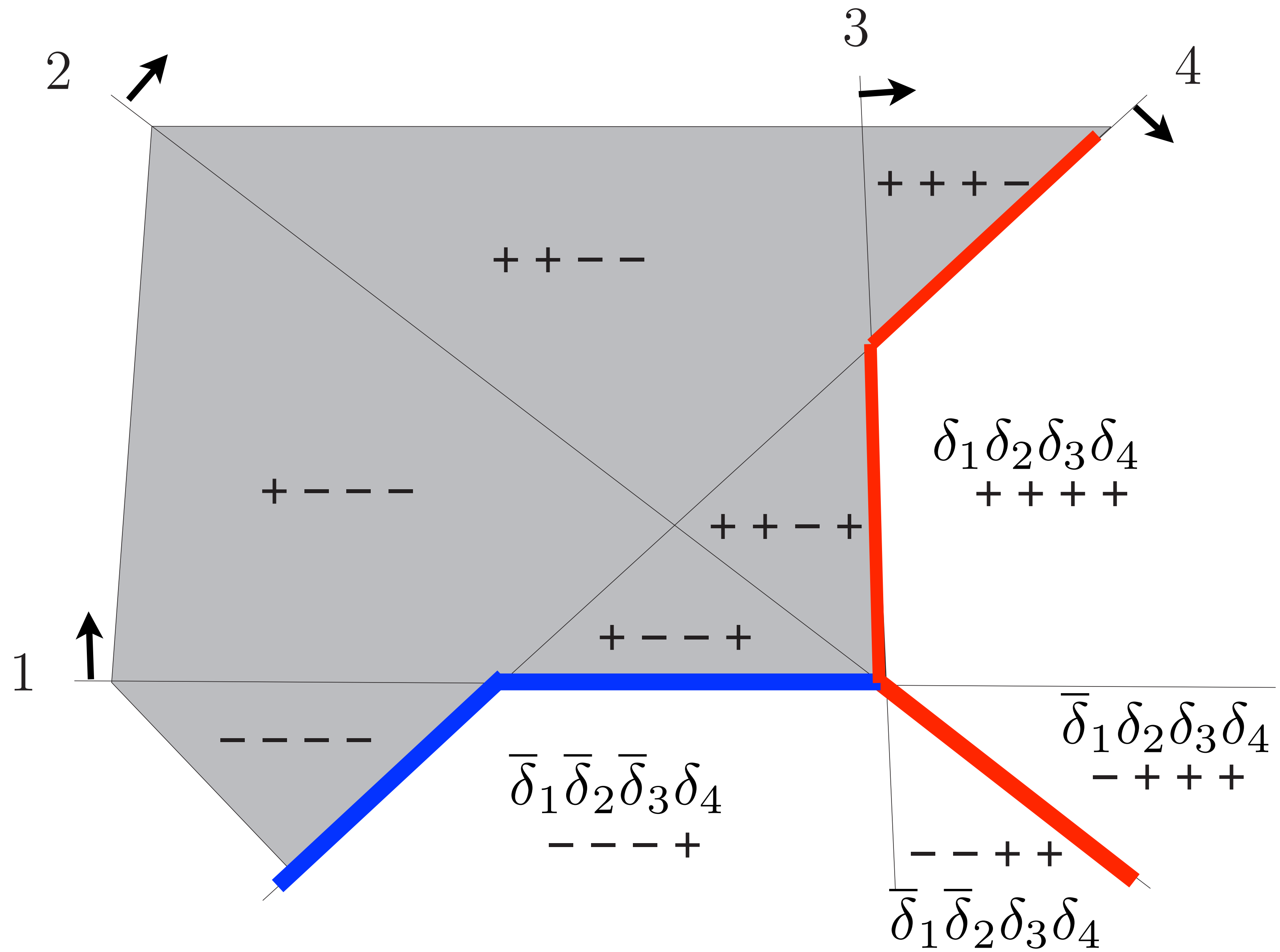
$$
\begin{aligned}
\text{White regions} &= \overline{\delta}_1\overline{\delta}_2\overline{\delta}_3\delta_4 + \overline{\delta}_1\overline{\delta}_2\delta_3\delta_4 + \overline{\delta}_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4 \\
&= \overline{\delta}_1\overline{\delta}_2\delta_4(\delta_3 + \overline{\delta}_3) + \delta_2\delta_3\delta_4(\delta_1 + \overline{\delta}_1) \\
&= \overline{\delta}_1\overline{\delta}_2\delta_4 + \delta_2\delta_3\delta_4
\end{aligned}
$$

Step 5: Recover Regions



$$\begin{aligned}\text{White regions} &= \bar\delta_1\bar\delta_2\bar\delta_3\delta_4 + \bar\delta_1\bar\delta_2\delta_3\delta_4 + \bar\delta_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4\\ &= \bar\delta_1\bar\delta_2\delta_4(\delta_3 + \bar\delta_3) + \delta_2\delta_3\delta_4(\delta_1 + \bar\delta_1)\\ &= \boxed{\bar\delta_1\bar\delta_2\delta_4} + \delta_2\delta_3\delta_4\end{aligned}$$

Step 5: Recover Regions



$$
\begin{aligned}
\text{White regions} &= \bar{\delta}_1\bar{\delta}_2\bar{\delta}_3\delta_4 + \bar{\delta}_1\bar{\delta}_2\delta_3\delta_4 + \bar{\delta}_1\delta_2\delta_3\delta_4 + \delta_1\delta_2\delta_3\delta_4 \\
&= \bar{\delta}_1\bar{\delta}_2\delta_4(\delta_3 + \bar{\delta}_3) + \delta_2\delta_3\delta_4(\delta_1 + \bar{\delta}_1) \\
&= \bar{\delta}_1\bar{\delta}_2\delta_4 + \boxed{\delta_2\delta_3\delta_4}
\end{aligned}
$$

1. Discrete Hybrid Automata

2. HYSDEL

3. Piecewise Affine Models

4. MPC for Hybrid Systems

5. Closing Remarks

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

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

- 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 (GUROBI, CPLEX)

MPC for PWA Models

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

- 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 for PWA Models

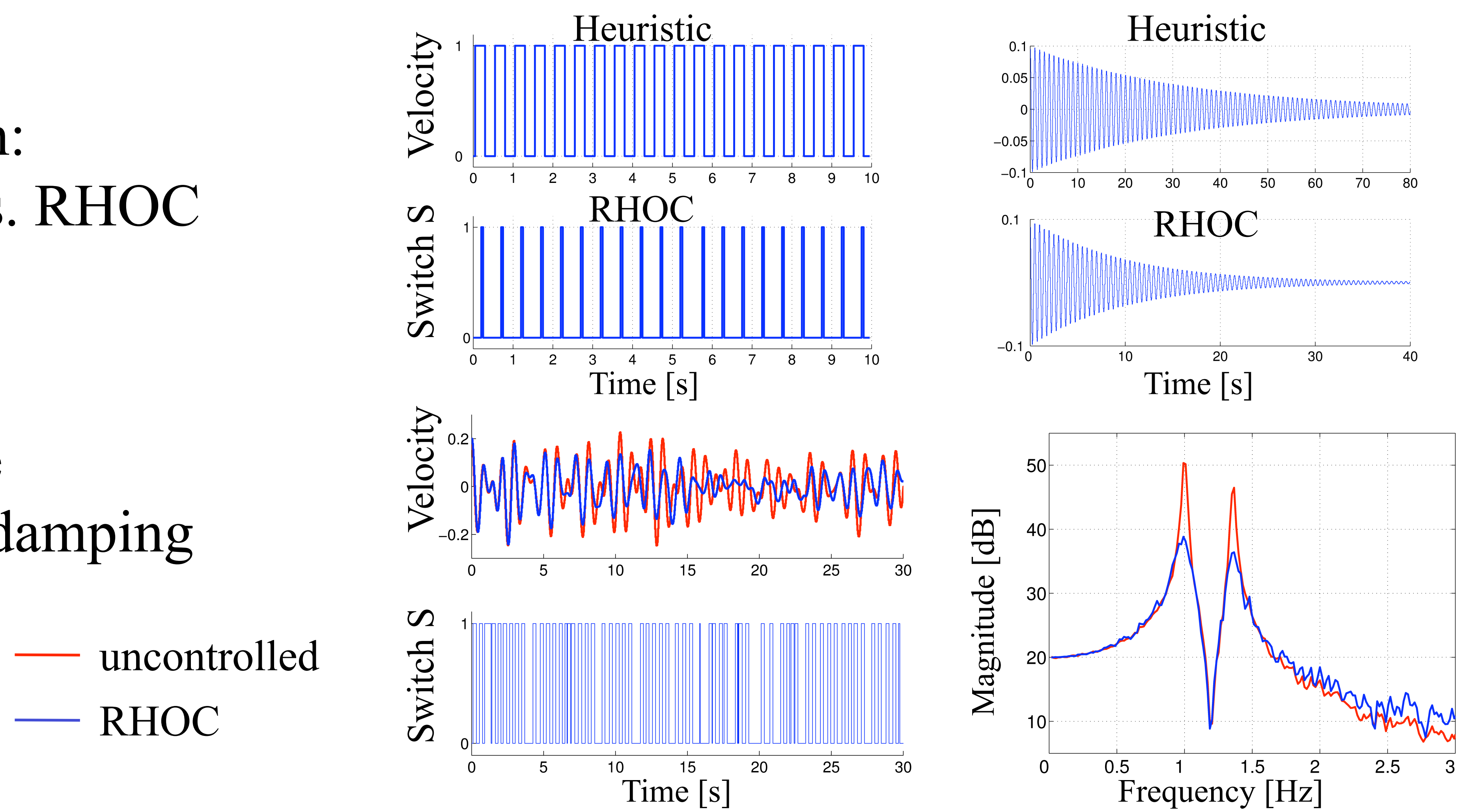
$$
\begin{aligned}
\min \quad & \sum_{k=0}^{N-1} \left(\|Q_x x_{t+k}\|_p + \|Q_u u_{t+k}\|_p \right) \\
\text{s.t.} \quad & x_{t+k+1} \leq M(1-\delta_{t+k,i}) + (A_i x_{t+k} + B_i u_{t+k} + f_i) \\
& x_{t+k+1} \geq M(1-\delta_{t+k,i}) + (A_i x_{t+k} + B_i u_{t+k} + f_i) \\
& D_i^x x_{t+k} - D_i^0 \leq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

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

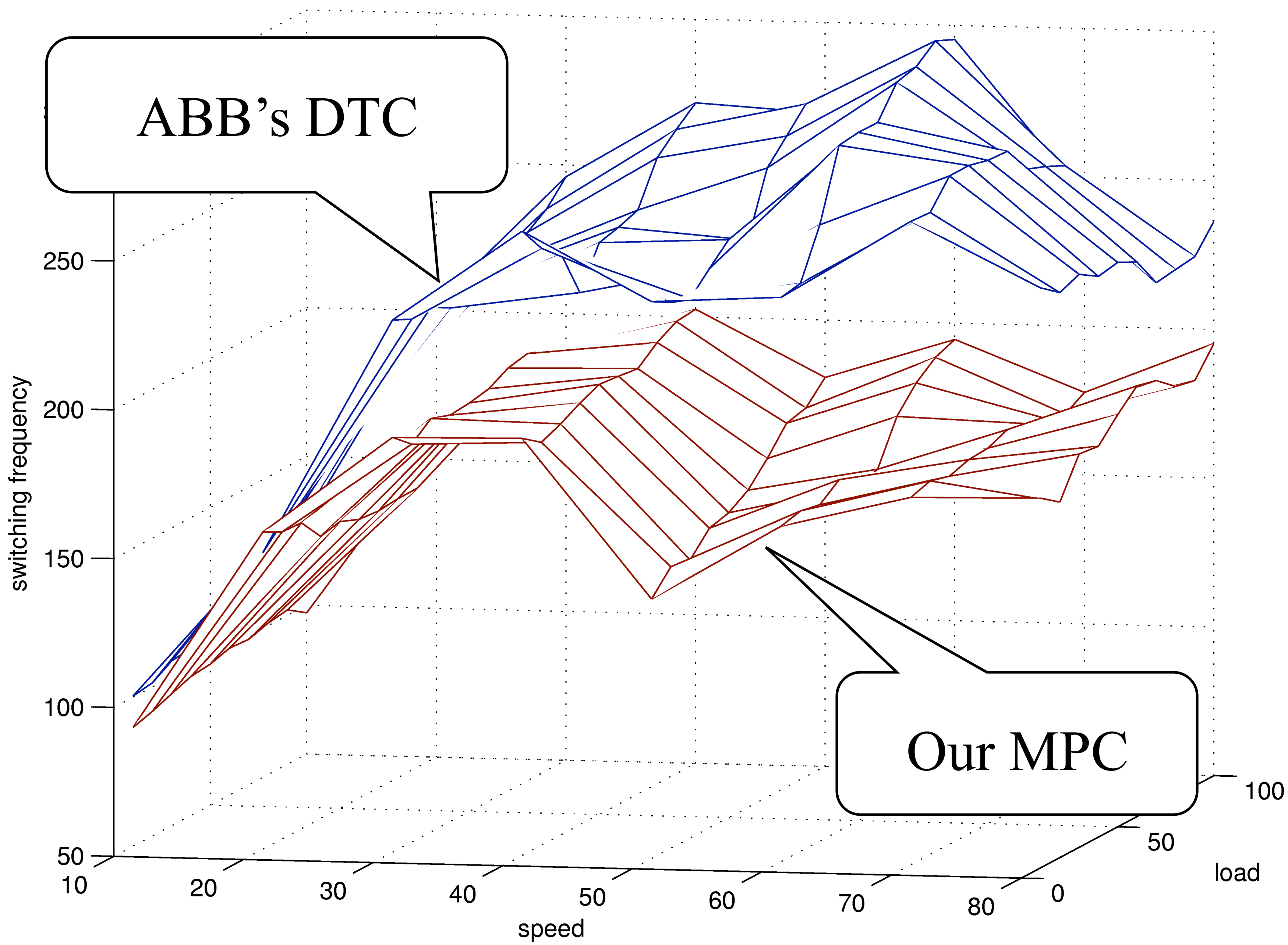
Smart Damping Materials

- **Control Design using Receding Horizon Optimal Control**
  - Receding Horizon Optimal Control (RHOC) solving MIQP *online*

- **Simulations**
  - Comparison:
    Heuristic vs. RHOC
  - Multi-mode
    broadband damping



Direct Torque Control



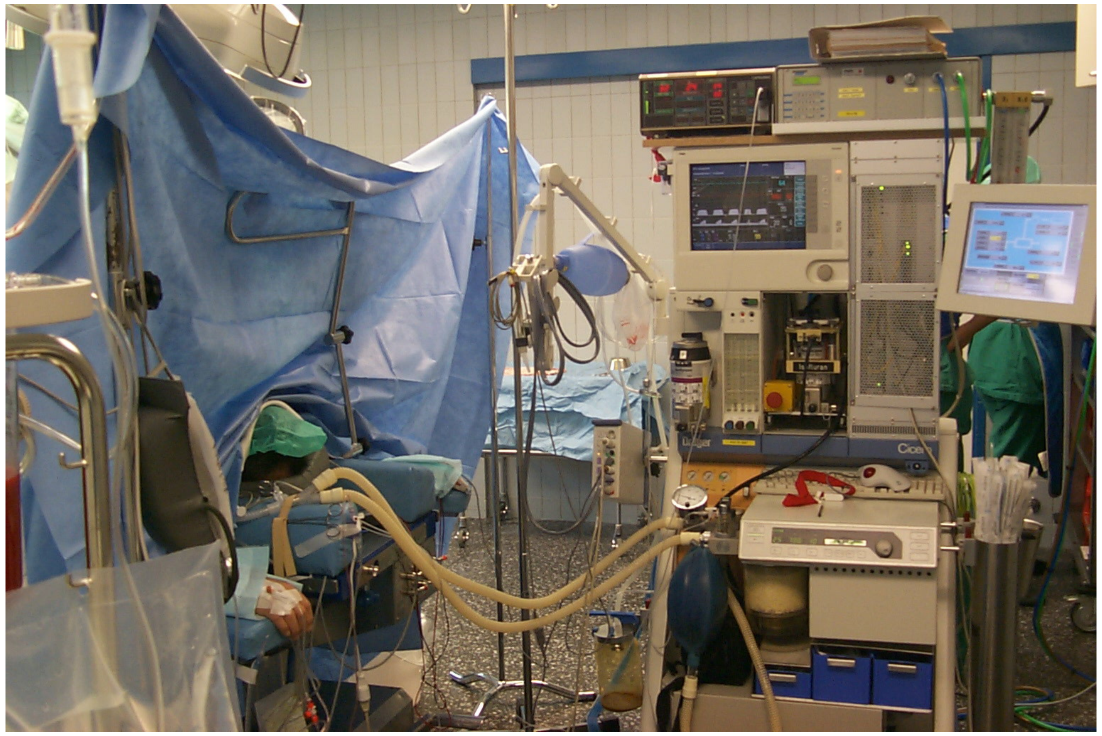
Reduction of switching frequency by up to 45 % (on average 25 %)

Control of Anesthesia

INSELSPITAL
UNIVERSITÄTSSPITAL BERN

Physical Setup:

- Patient undergoing surgery
- Analgesic infusion pump

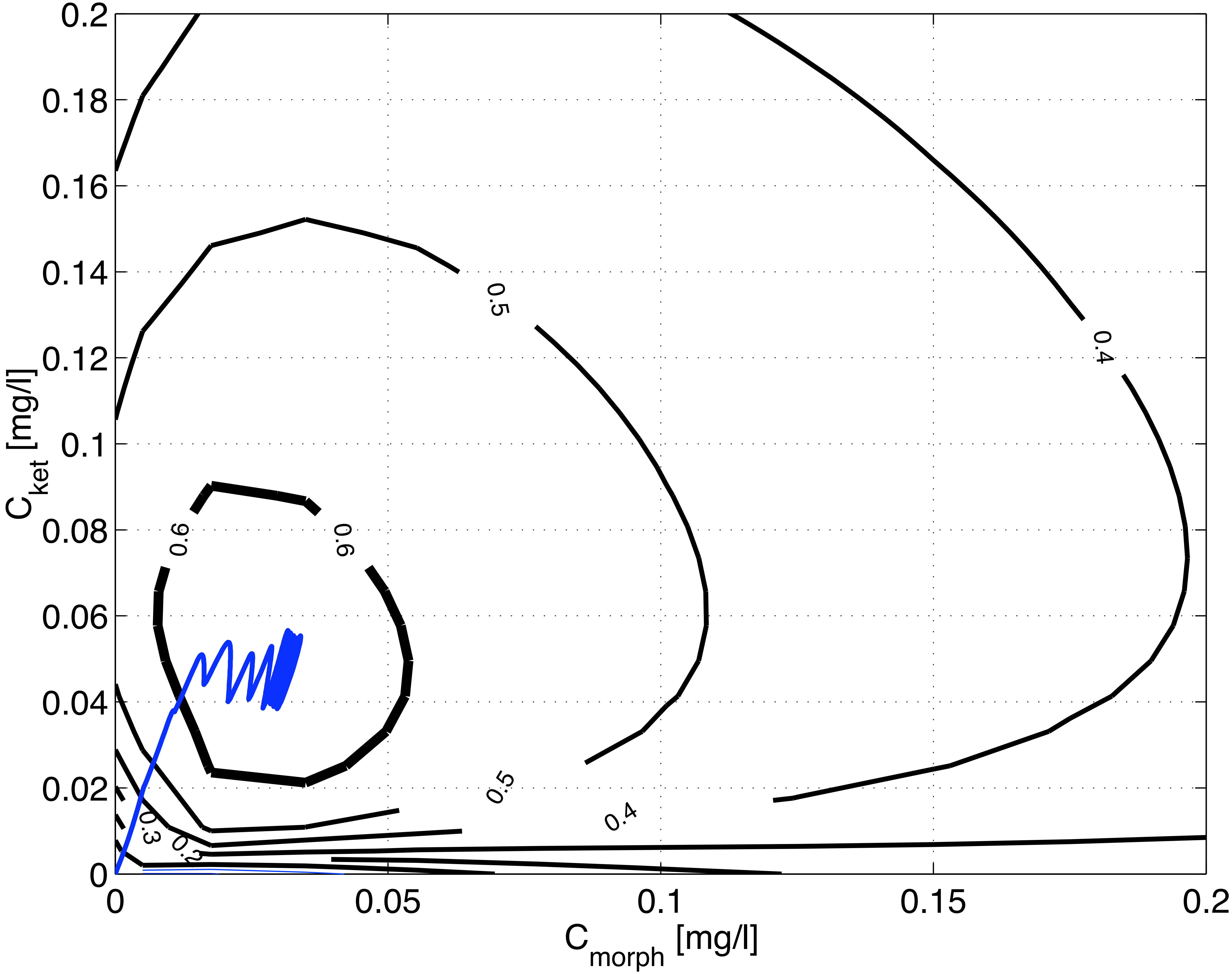


Control Objectives:

- Minimize stress reaction to surgical stimulation (by controlling mean arterial pressure)
- Minimize drug consumption

Excellent performance of administration scheme,
mean arterial pressure variations kept within bounds

Control of Anesthesia



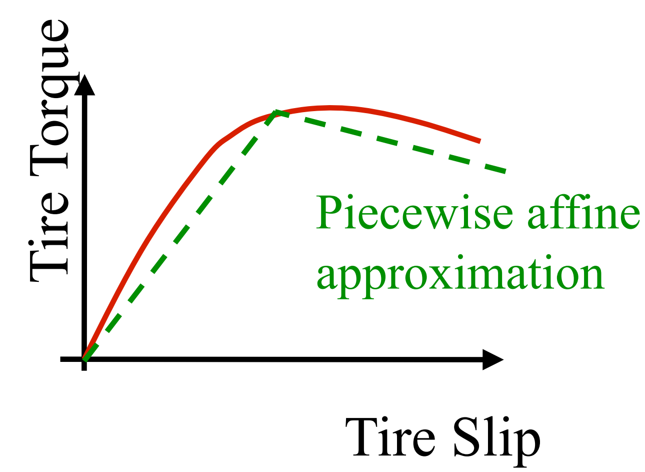
Traction Control

Physical Setup:

- Improve driver's ability to control vehicle under ad conditions (wet or icy roads)
- Tire torque is nonlinear function of slip
- Uncertainties and constraints

Control Objectives:

- Maximize tire torque by keeping the tire slip close to the desired value

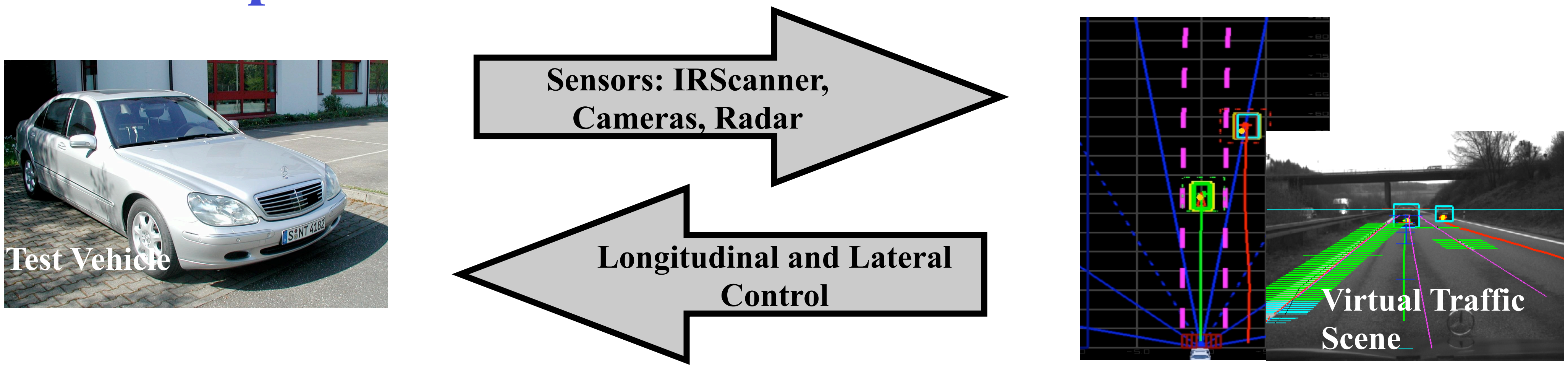


Experimental results: 2000 Ford Focus on a Polished Ice Surface;
Receding Horizon controller with 20 ms sampling time

Adaptive Cruise Control

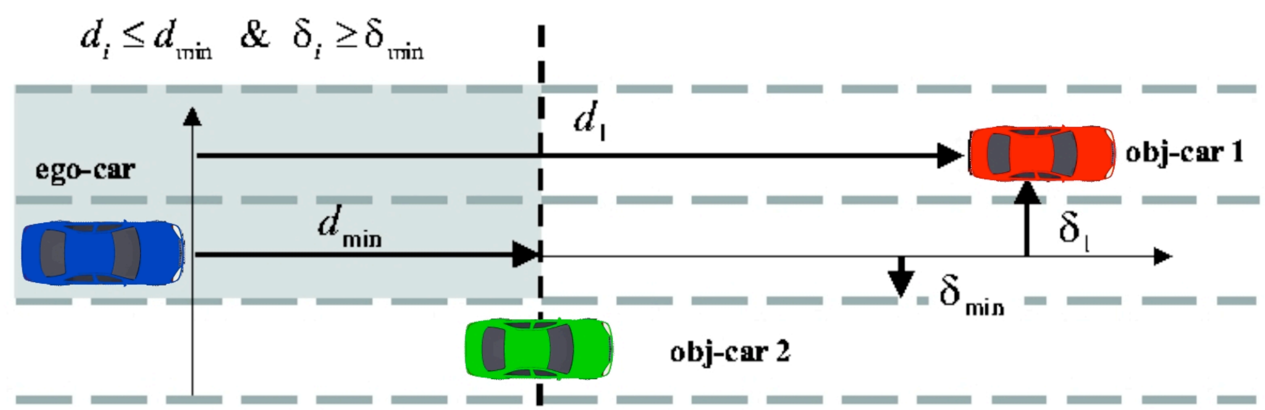
DAIMLERCHRYSLER

Physical Setup:



Control Objectives:

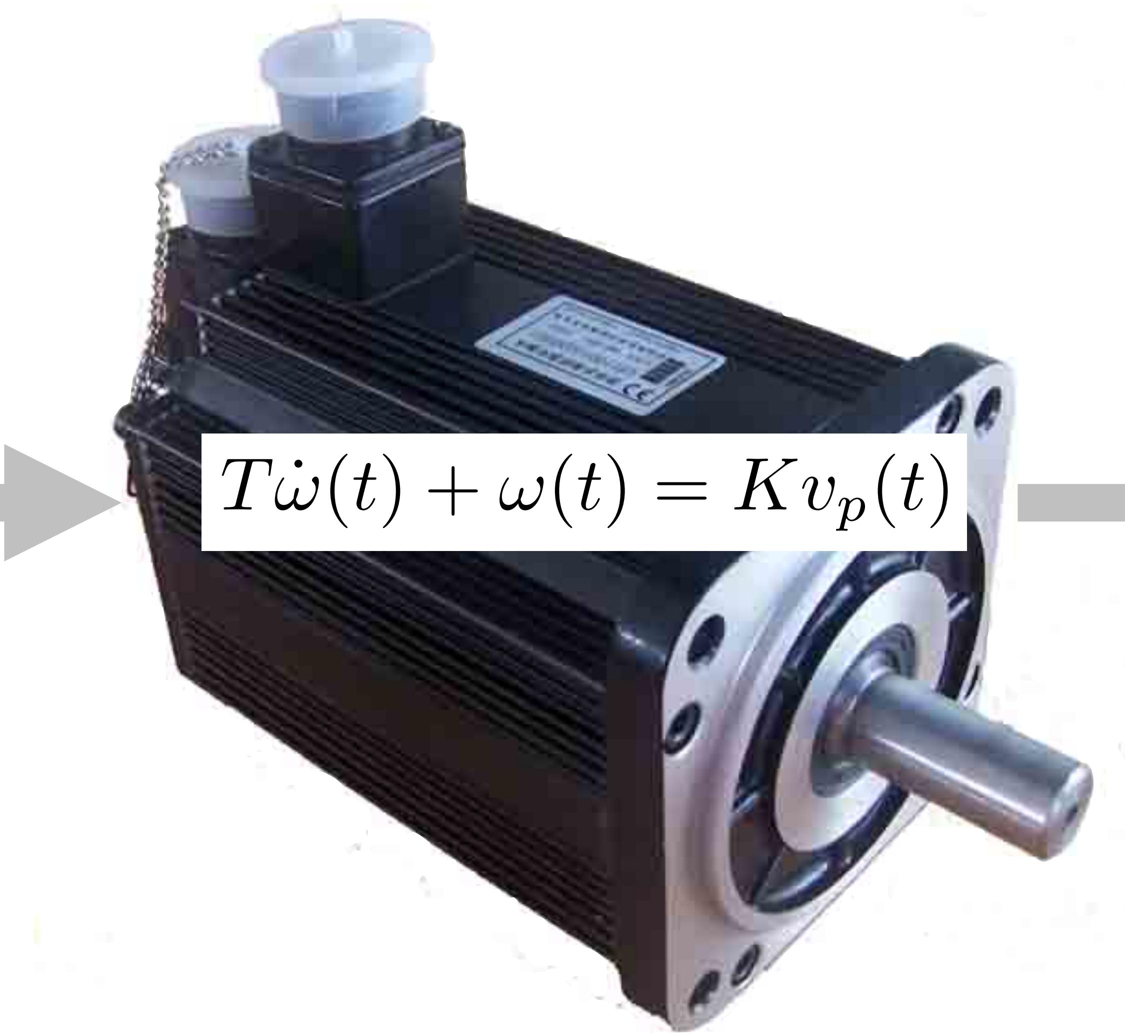
- Track reference speed
- Respect traffic rules
- Consider all objects on all lanes



Optimal state-feedback control law successfully implemented and tested on a research car Mercedes E430 with 80ms sampling time

Servo Motor Example

voltage

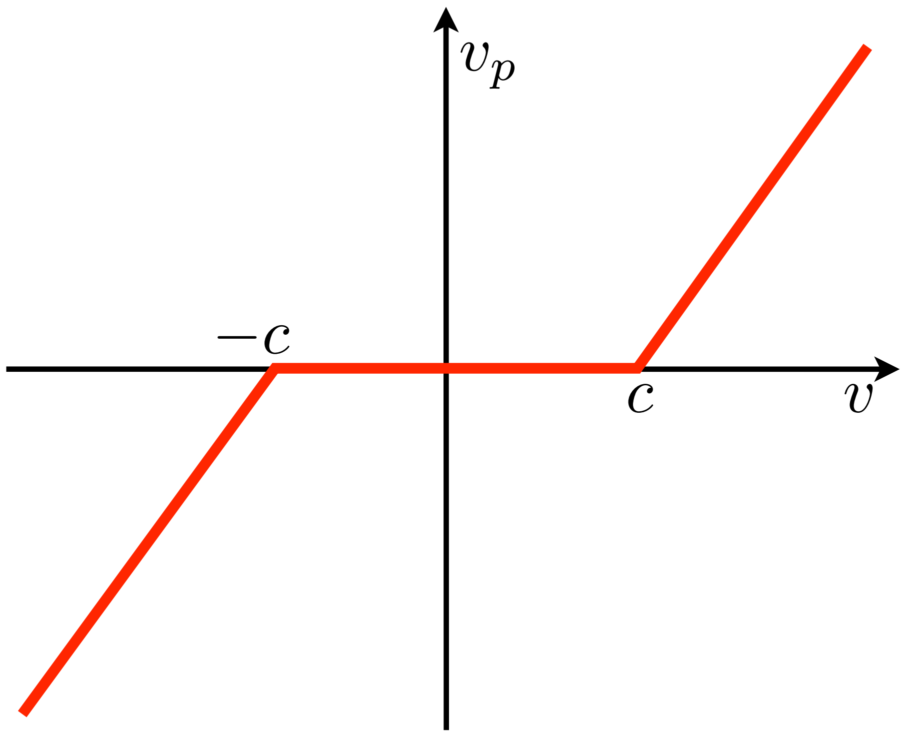


$$T\dot{\omega}(t) + \omega(t) = K v_p(t)$$

angle

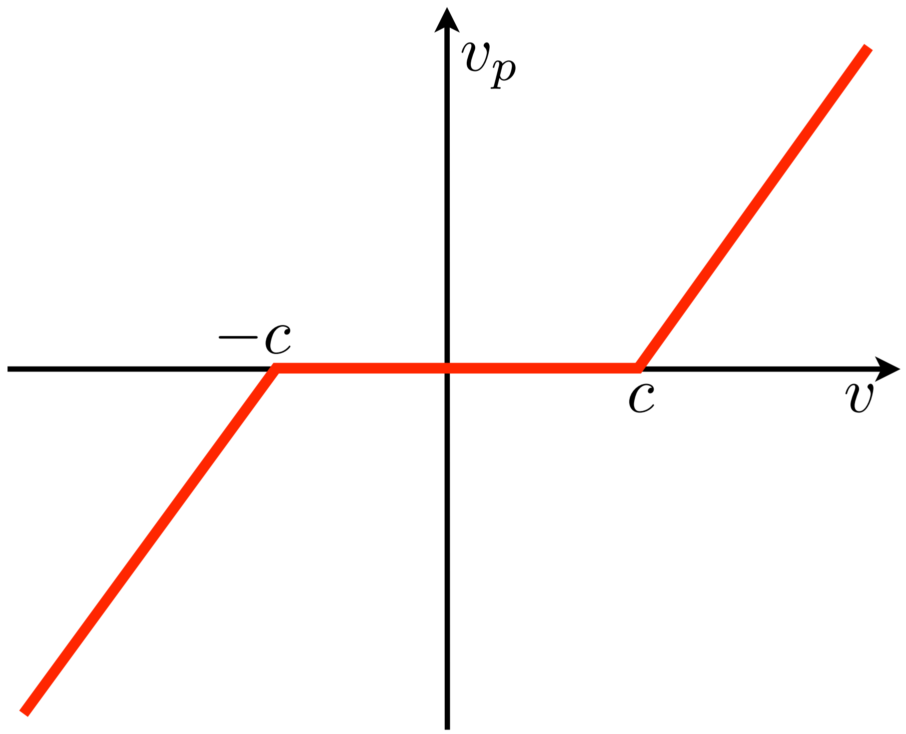
angular velocity

Servo Motor Example: Deadzone

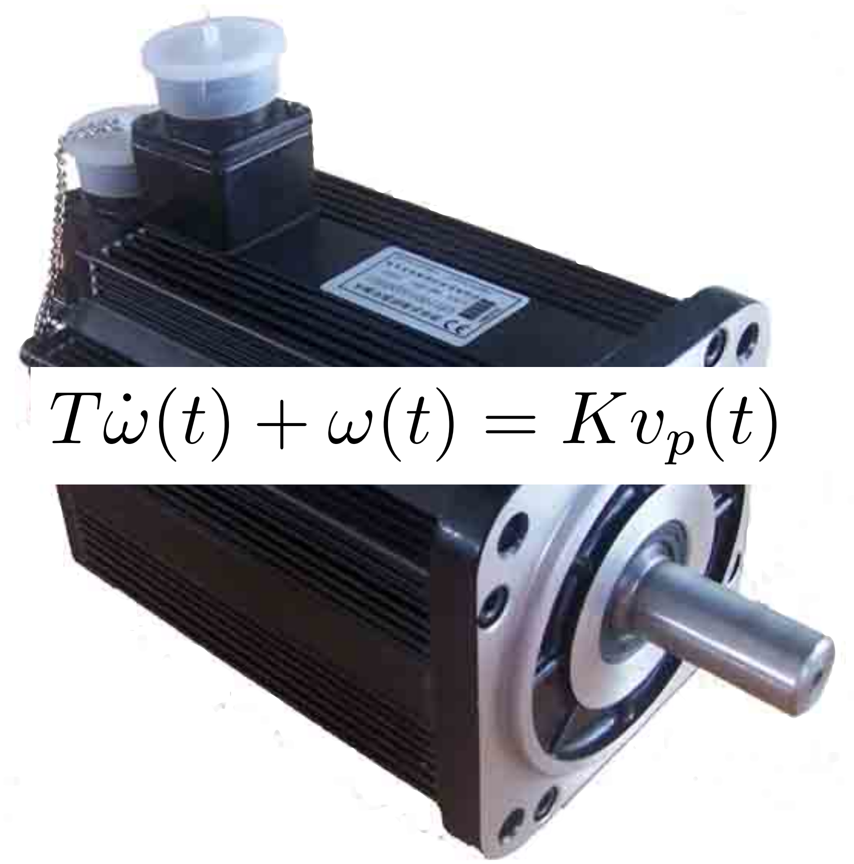


$$v_p = \begin{cases} m(v-c) & \text{if } v > c \\ 0 & \text{if } -c \leq v \leq c \\ m(v+c) & \text{if } v < -c \end{cases}$$

Servo Motor Example



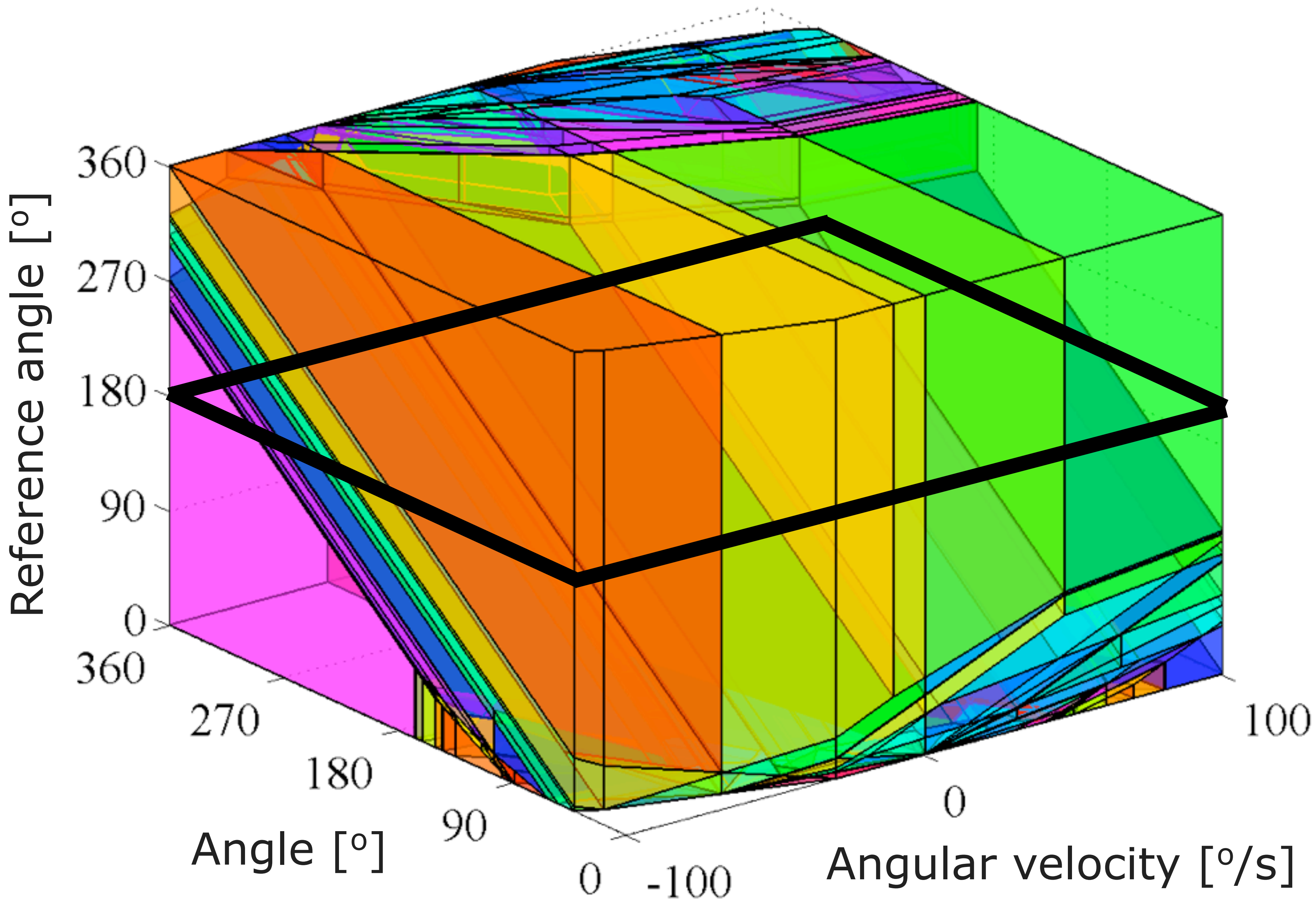
+



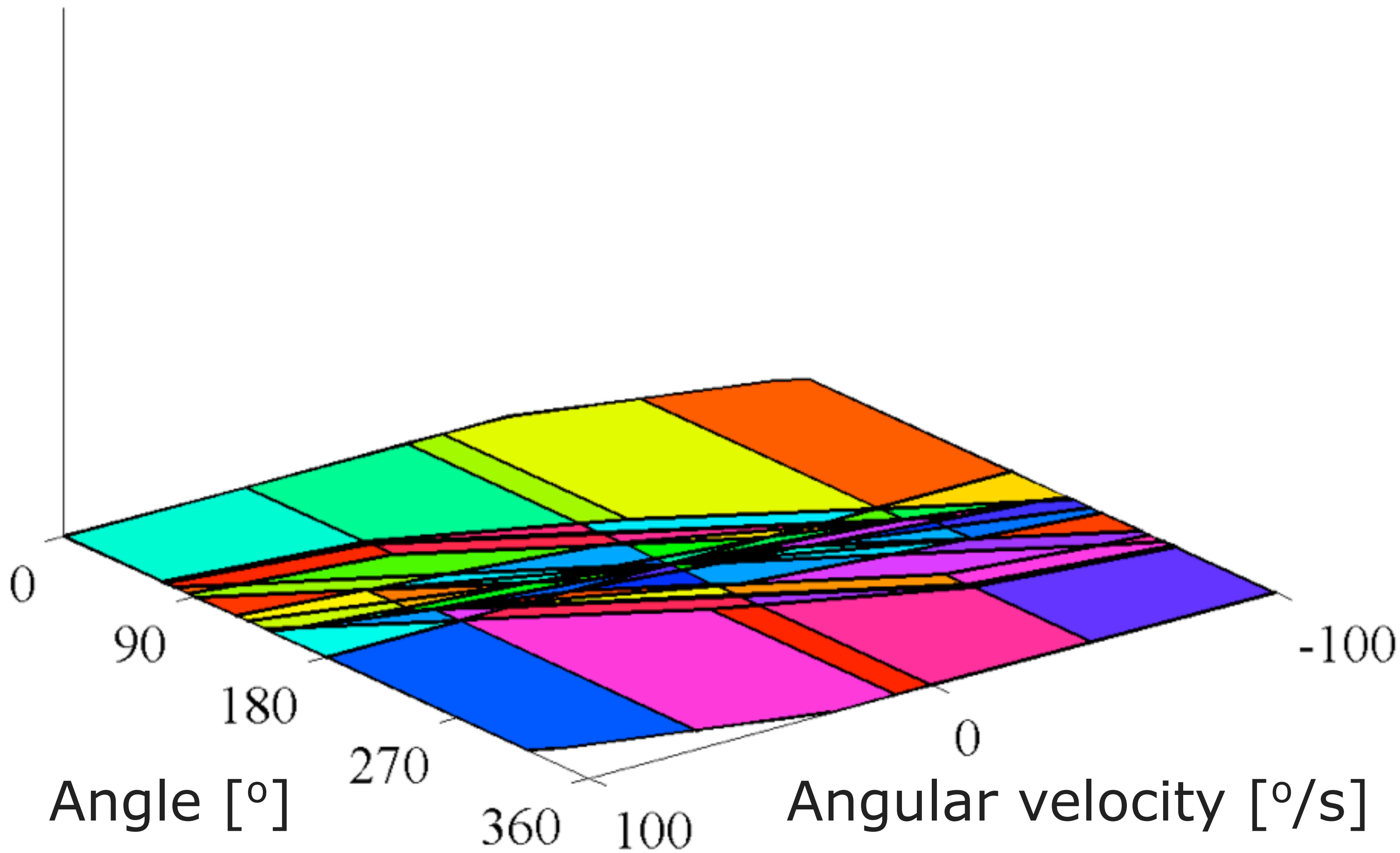
$$T\dot{\omega}(t) + \omega(t) = K v_p(t)$$

Minimize	**tracking error**
Considering	**measurements**
	linear servo model
	deadzone nonlinearity
	constraints

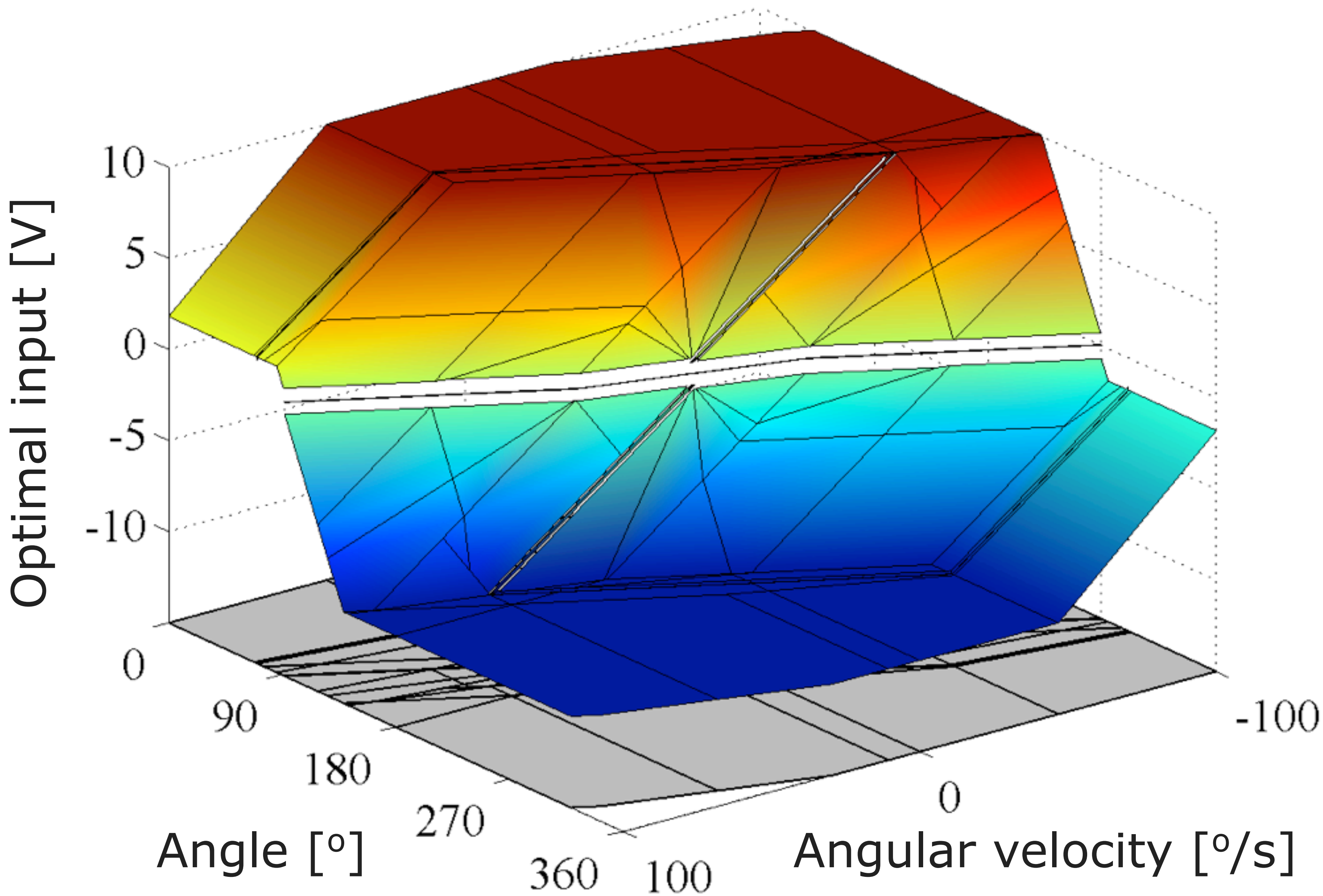
Explicit MPC with 627 Regions



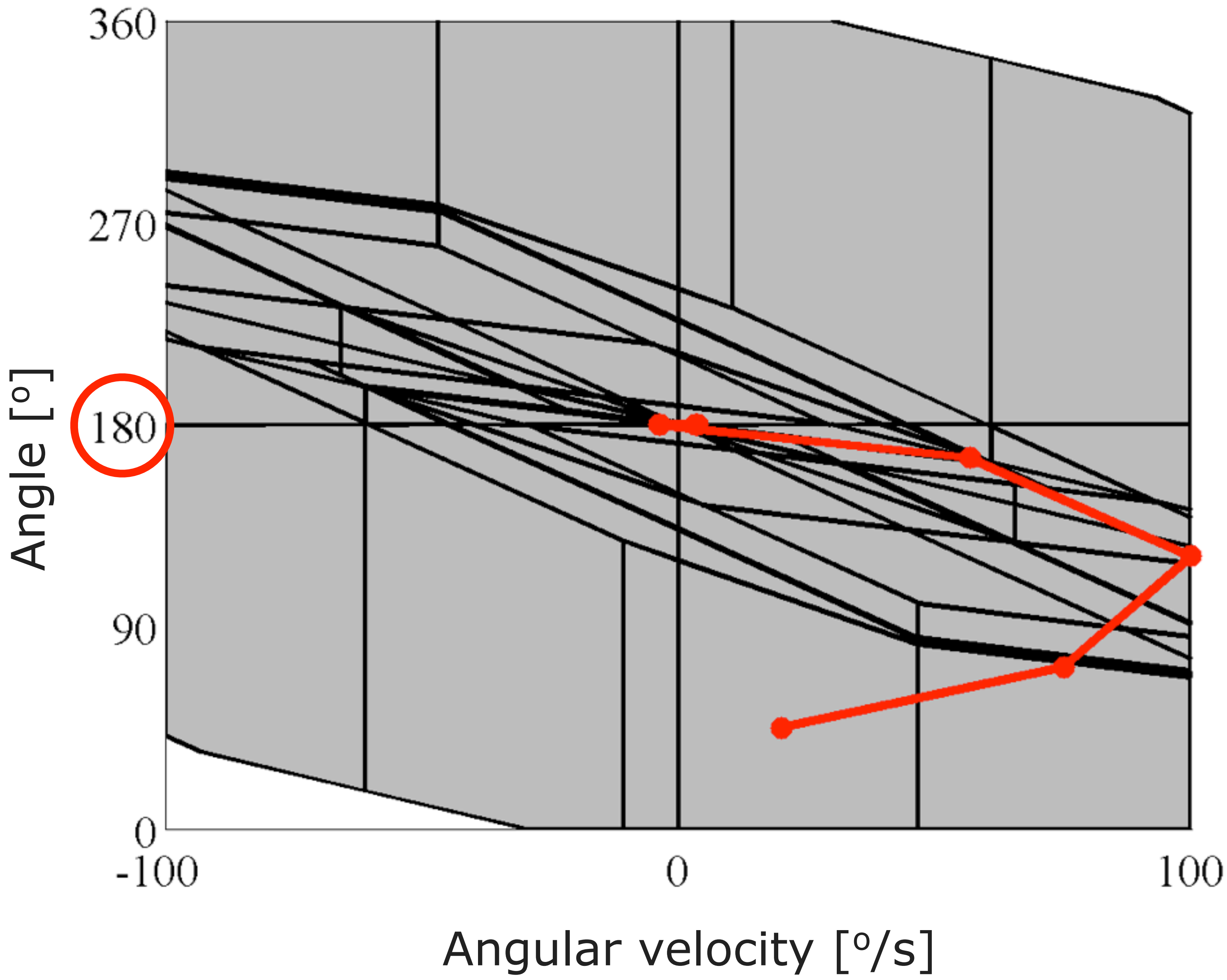
Regions



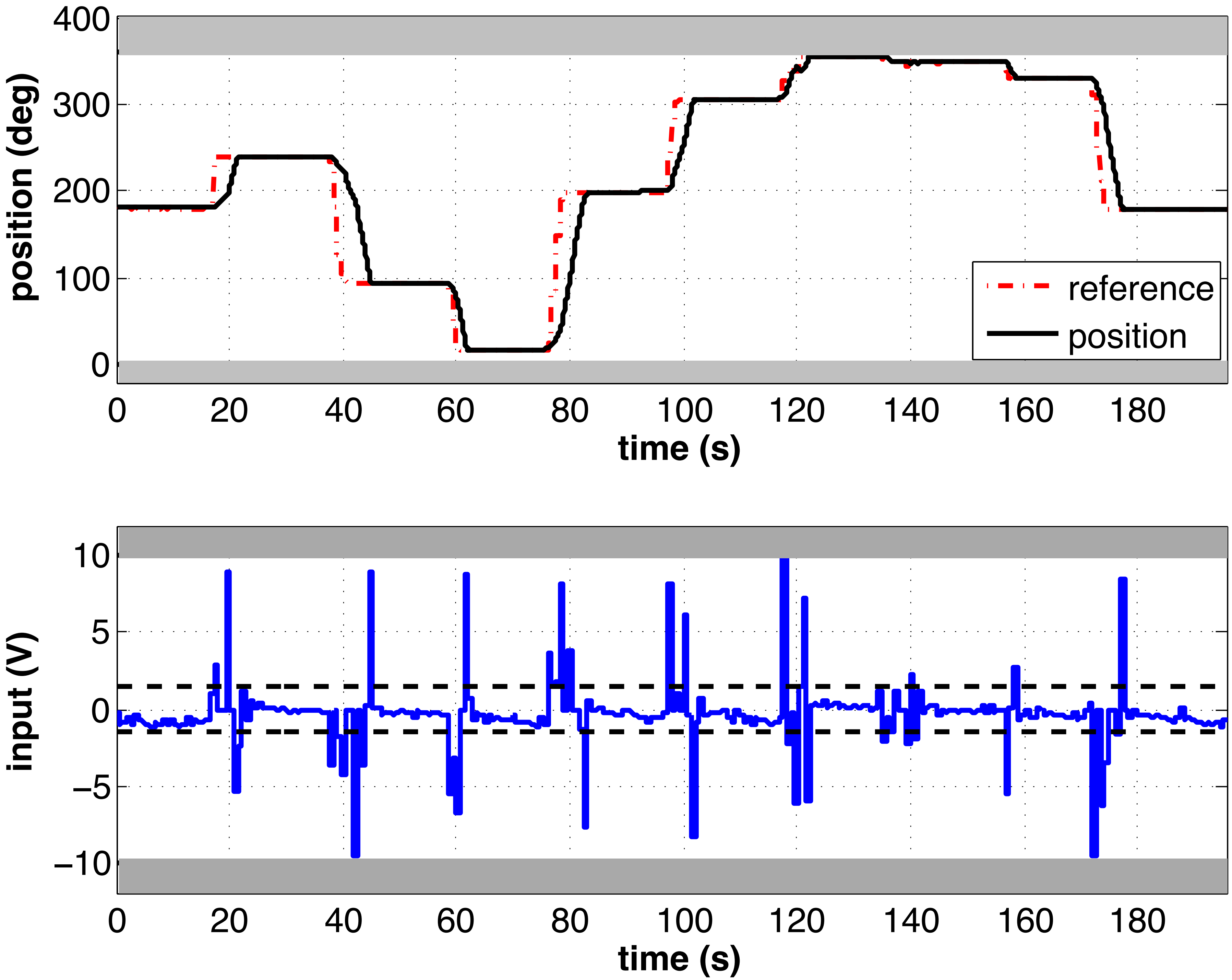
Optimal Control Law



Implementation



Real Measurements



Magnetic Manipulator

$$\dot{p} = v$$
$$\dot{v} = i\frac{\alpha\Delta}{(\Delta^2 + \beta)^3}$$

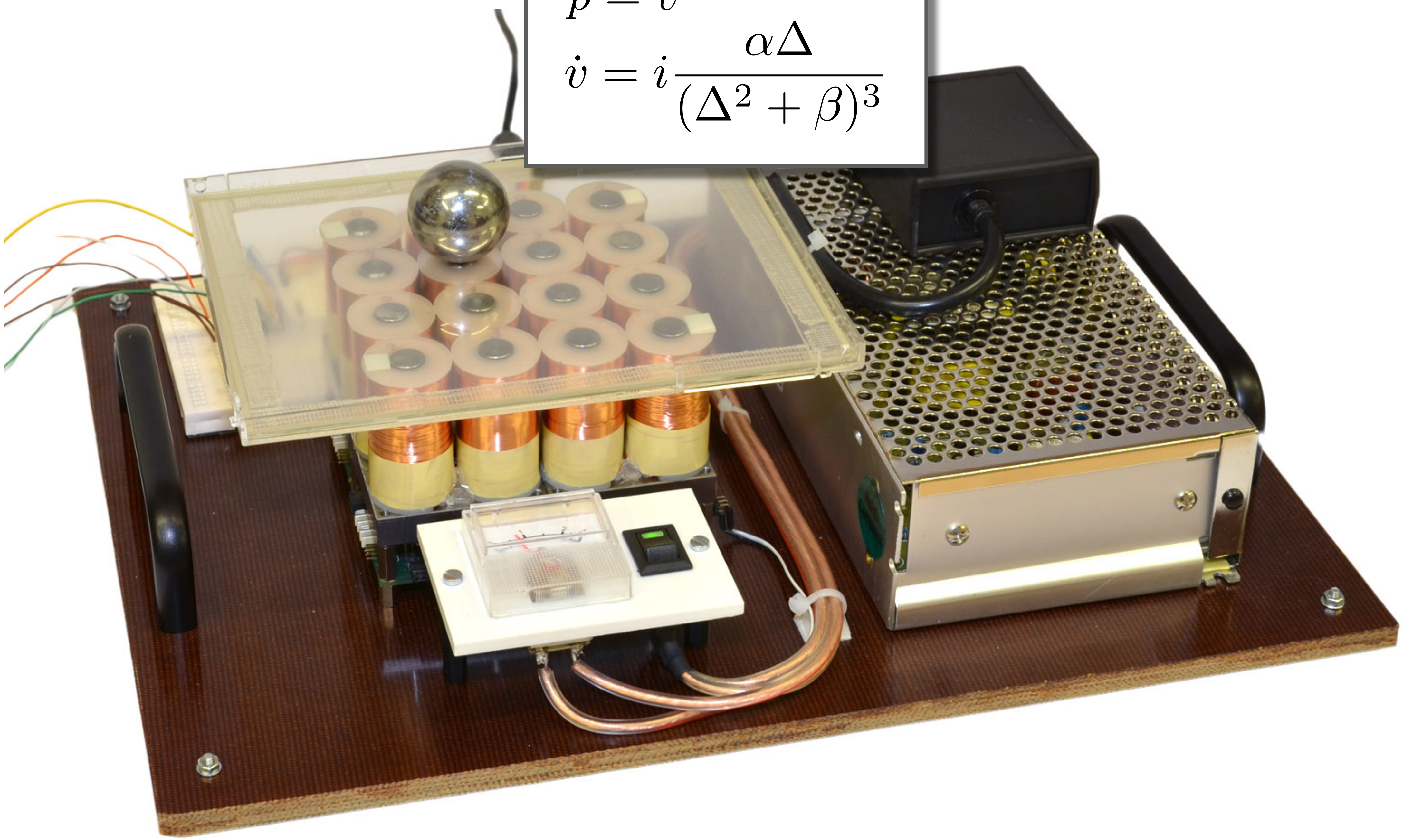
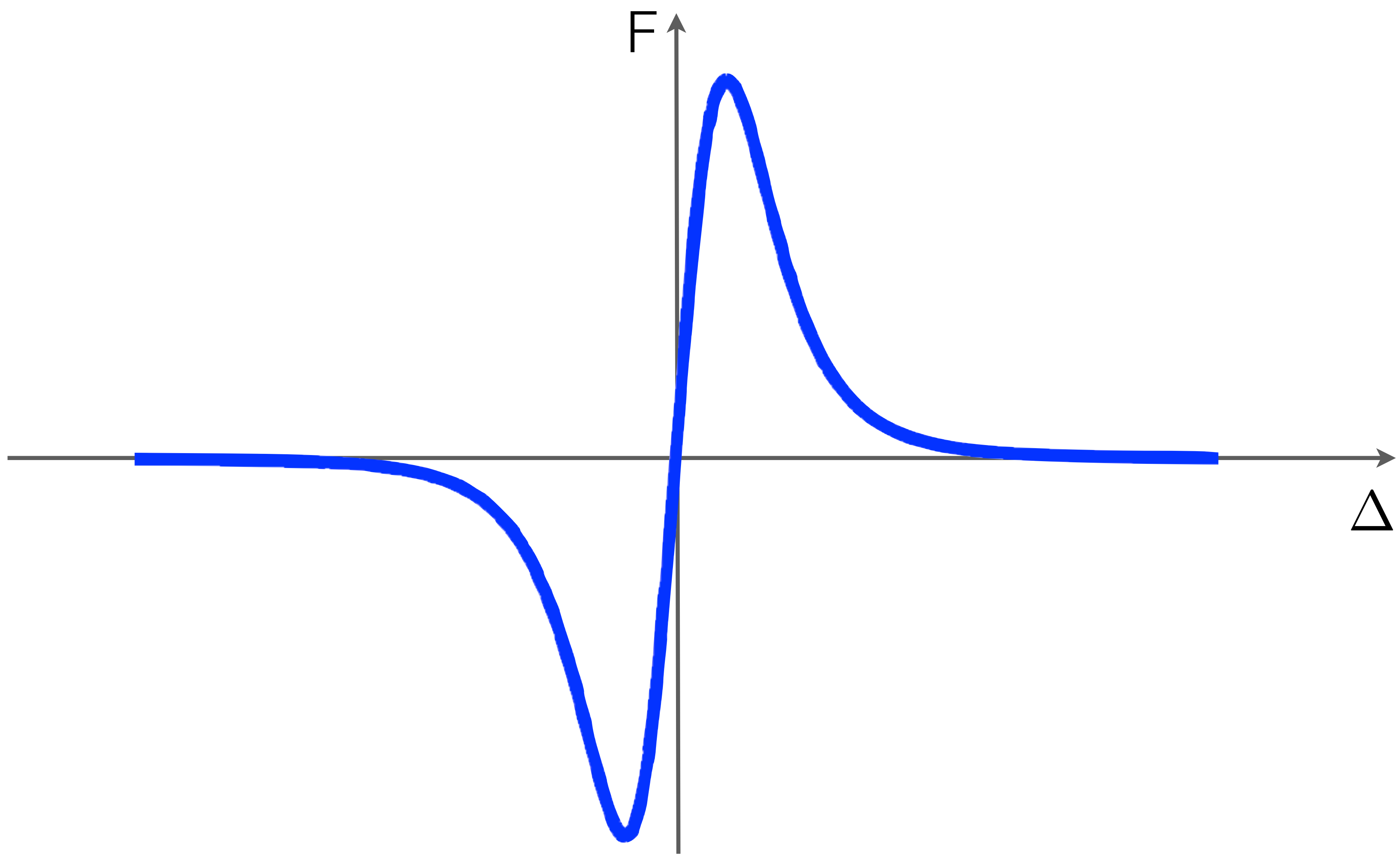


Photo & data courtesy of Z. Hurák, CVUT Prague

Magnetic Manipulator

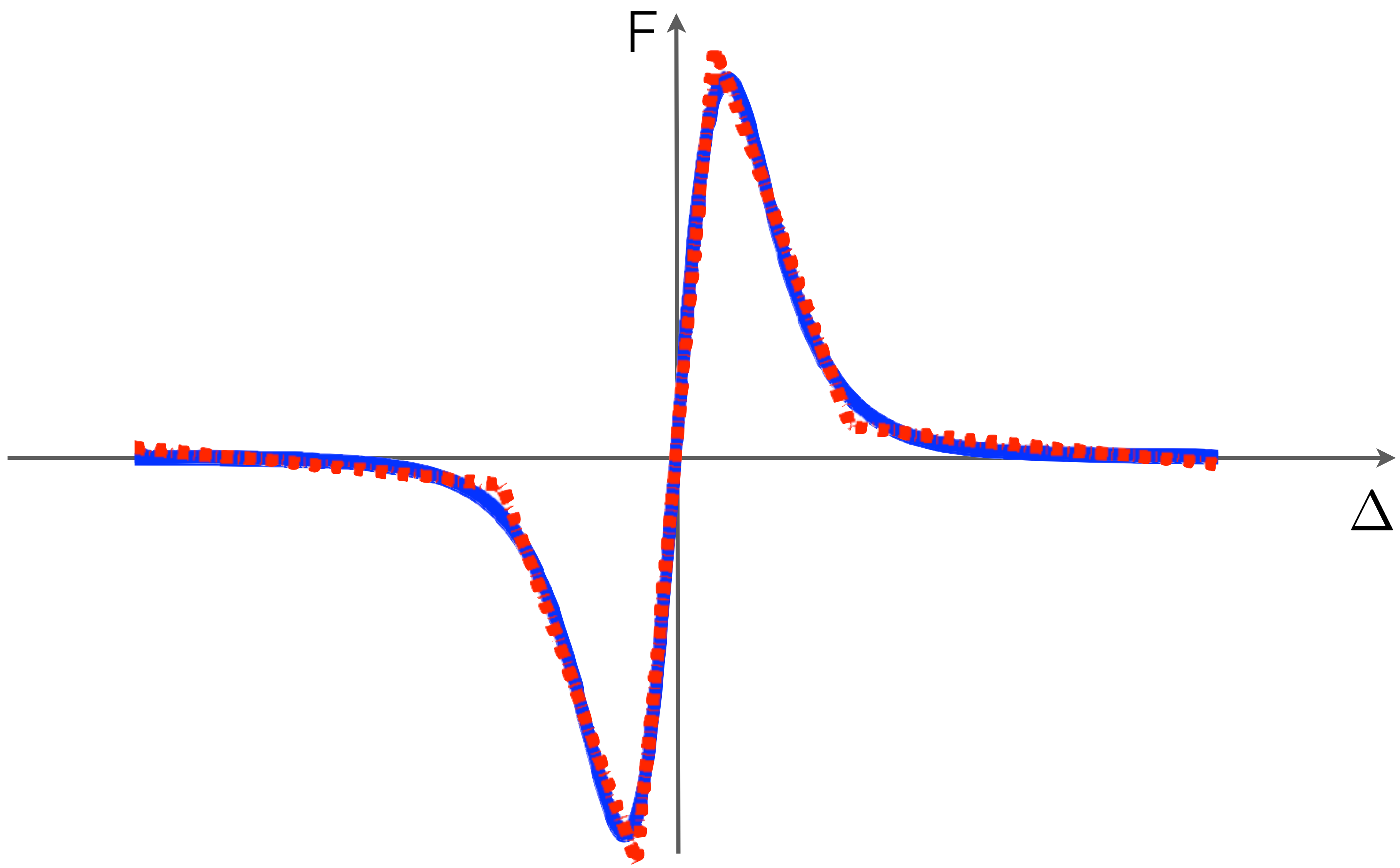
$$\dot{p} = v$$
$$\dot{v} = i\boxed{\frac{\alpha\Delta}{(\Delta^2+\beta)^3}}$$



Magnetic Manipulator

$$\dot{p} = v$$

$$\dot{v} = i\boxed{\frac{\alpha\Delta}{(\Delta^2 + \beta)^3}}$$



Magnetic Manipulator

PWA model

5 binaries for $F = {}^{\alpha\Delta}/_{(\Delta^2+\beta)^3}$

2 binaries for iF

Discretization at 0.1 seconds

MPC

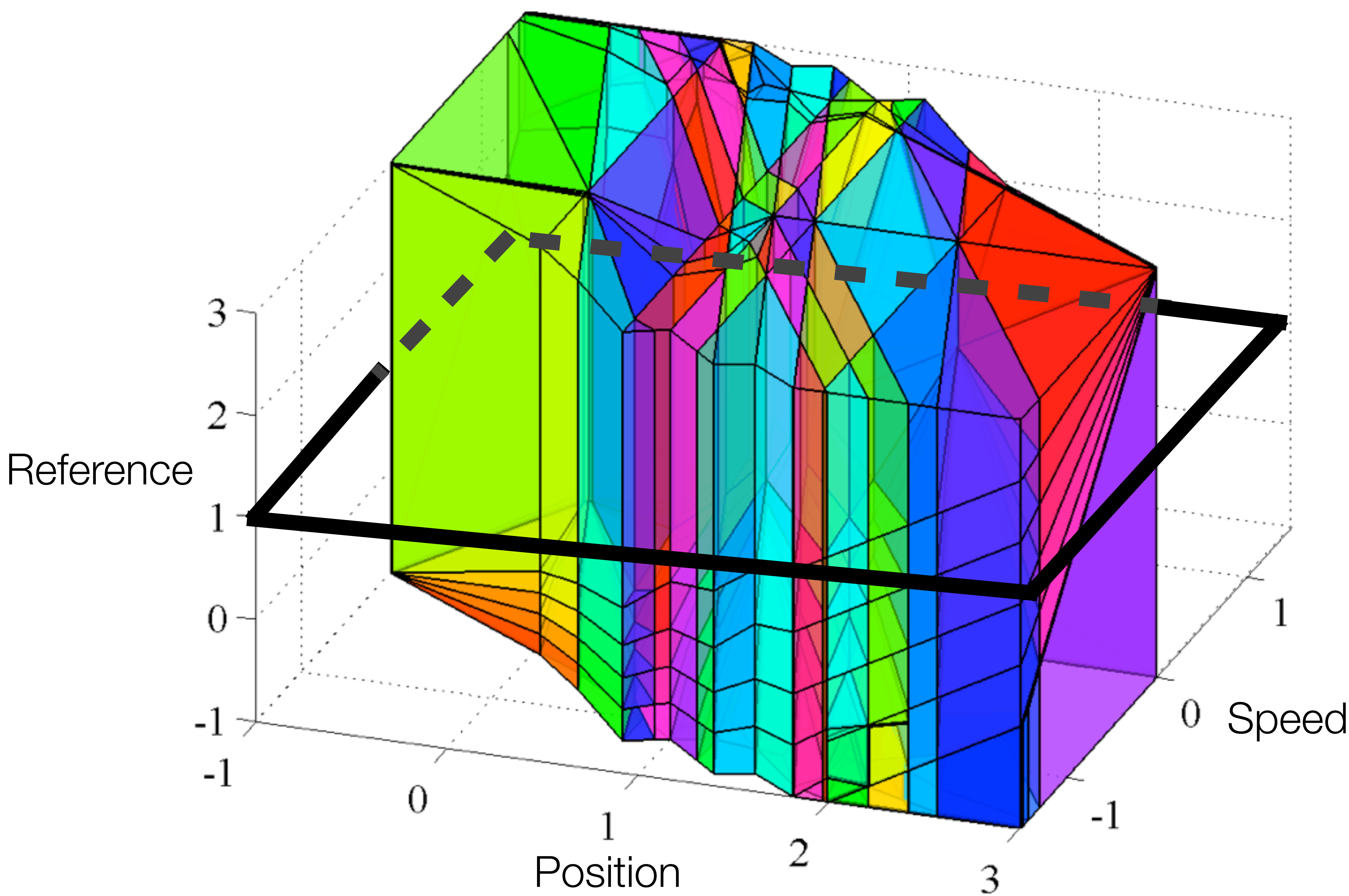
Min/max constraints on position, speed, current

PWL cost function $\sum_{k=0}^{N-1} |p_{k+1} - r| + |v_{k+1}| + |i_k|$

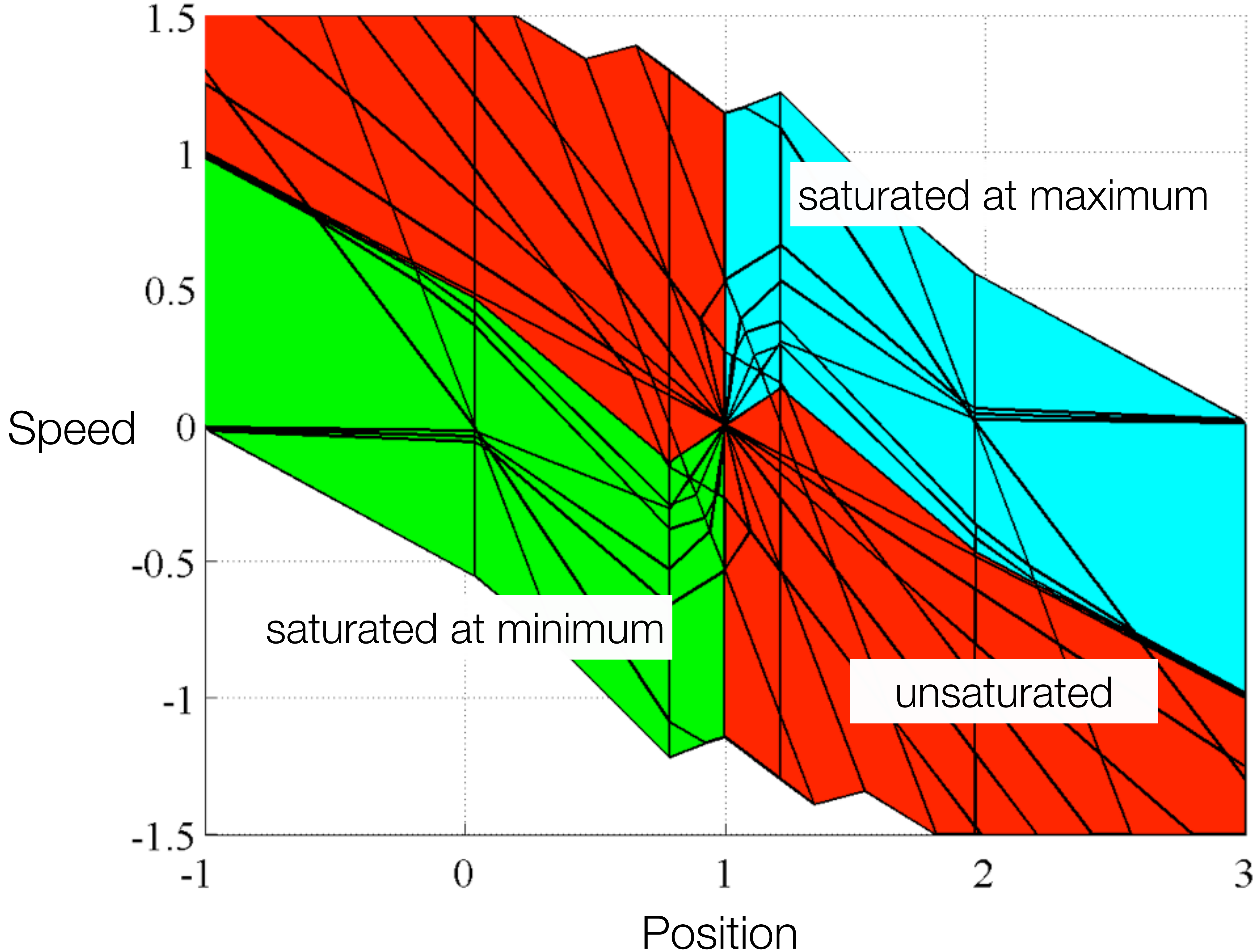
Prediction horizon 5

Solved as a parametric MILP with 3 parameters

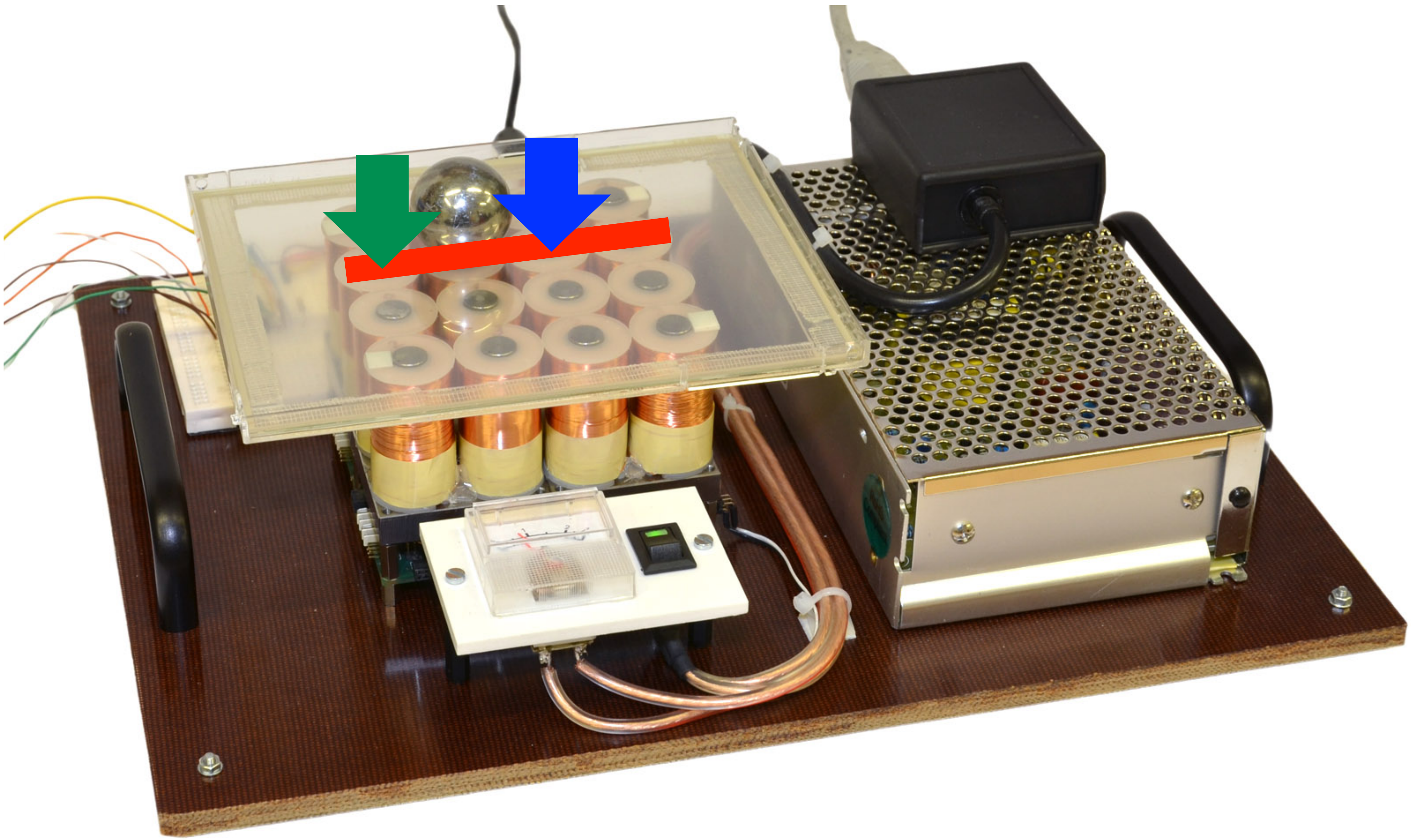
Explicit MPC with 917 Regions



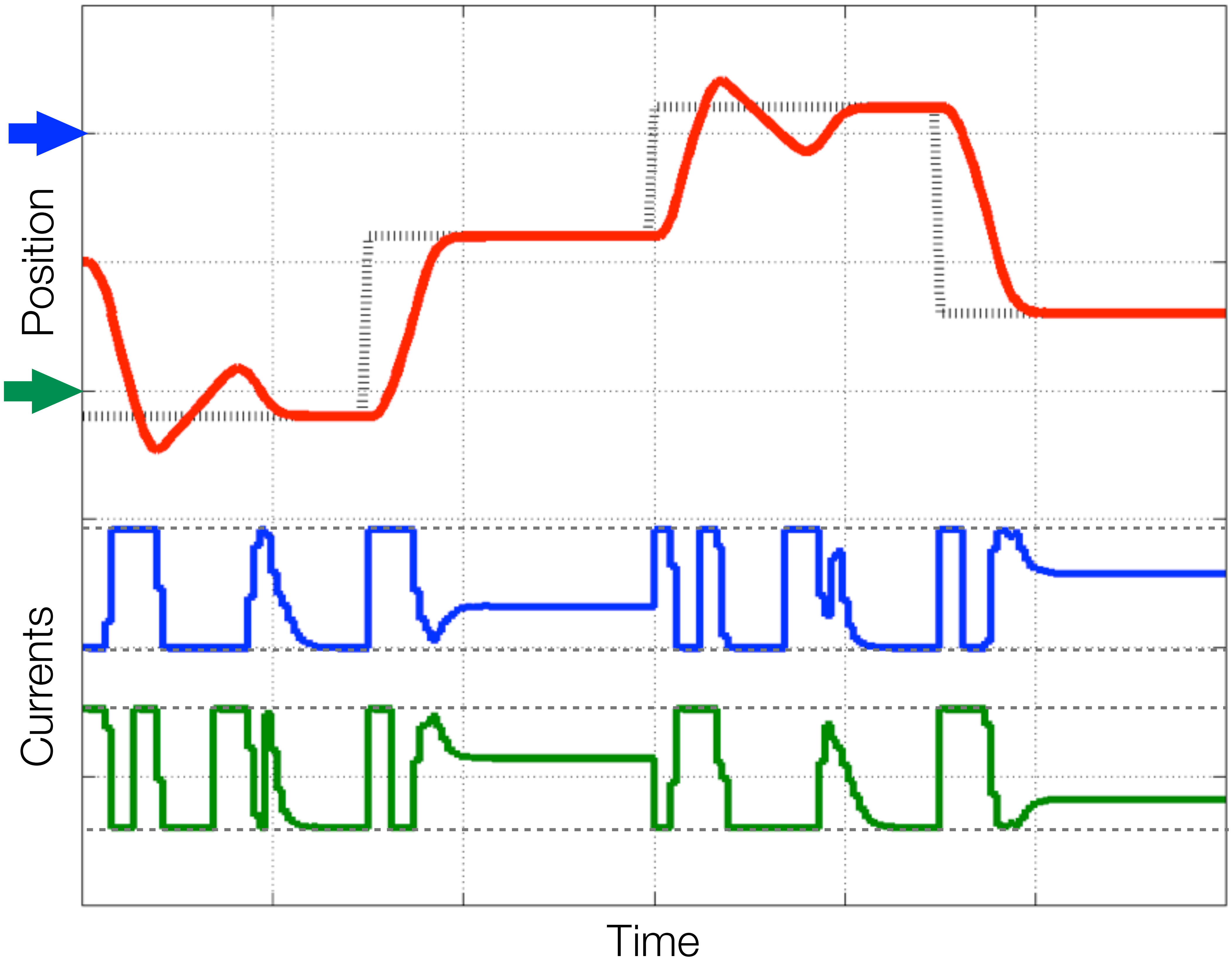
Cross-Section Through Reference=1.0



Simulation Scenario



Simulation Results



1. Discrete Hybrid Automata

2. HYSDEL

3. Piecewise Affine Models

4. MPC for Hybrid Systems

5. Closing Remarks

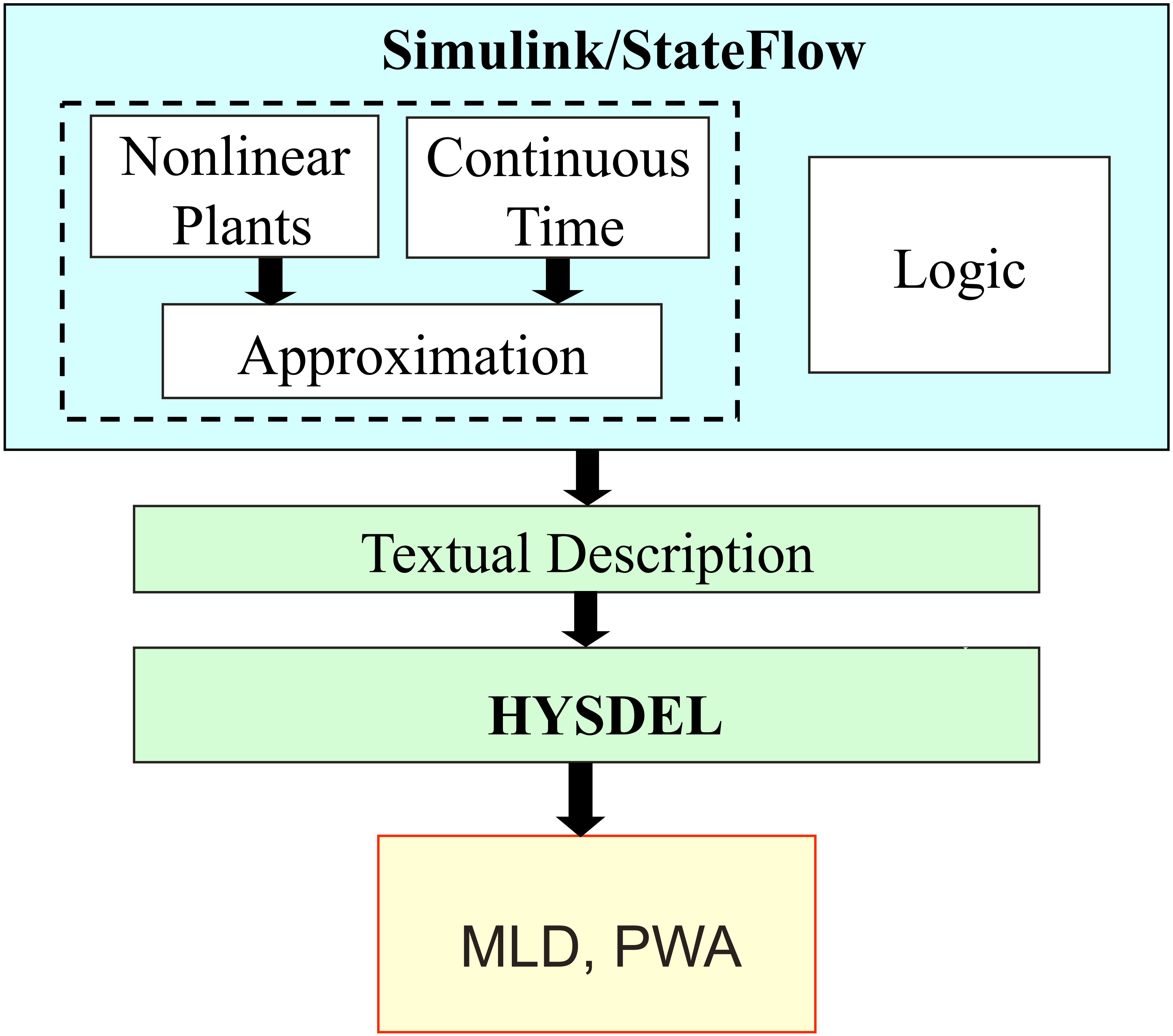
Hybrid Systems

- 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
- Control:
  - 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?

Our Vision of Automated Hybrid Modeling



Software for Hybrid Systems

- Multi-Parametric Toolbox (includes HYSDEL2, YALMIP, HIT)
  - http://control.ee.ethz.ch/~mpt/
- HYSDEL
  - http://control.ee.ethz.ch/~hybrid/hysdel/
- YALMIP
  - http://users.isy.liu.se/johanl/yalmip/
- 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
- Book on hybrid systems
  - Lunze: *Handbook of Hybrid Systems Control*, Cambridge Press, 2009
- Book on explicit MPC for hybrid systems
  - Borrelli, Bemporad, Morari: *Predictive Control for Linear and Hybrid Systems*

    http://www.mpc.berkeley.edu/mpc-course-material