HYSDEL Models and Controller Synthesis for Hybrid Systems

Alberto Bemporad

Dept. of Information Engineering
University of Siena, Italy
bemporad@dii.unisi.it

Automatic Control Laboratory
Swiss Federal Institute of Technology
bemporad@aut.ee.ethz.ch

Control Group in Siena

- 4 professors
- 4 postdocs
- 4 PhD students (+2 to come)

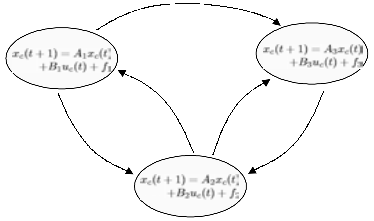
- Research activities:
  - robust control
  - identification
  - mobile robotics & dynamic vision
  - sequencing and scheduling
  - telelaboratory
  - hybrid systems
  - model predictive control

Summary of my talk

1. Models of hybrid systems
2. The HYSDEL language
3. Controller synthesis for hybrid systems
4. Safety analysis of hybrid systems
5. Examples
6. Ongoing research

Models of Hybrid Systems
(that can be handled by HYSDEL)

Hybrid Dynamics - Continuous Part



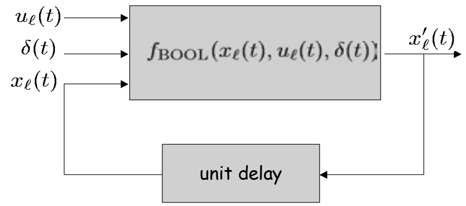
- Switched continuous dynamics

$$x_c'(t) = A_{i(t)} x_c(t) + B_{i(t)} u_c(t) + f_{i(t)}$$

$x_c \in \mathbb{R}^n$	continuous states
$u_c \in \mathbb{R}^m$	continuous inputs
$i \in \{0, 1, \ldots, s\}$	switching index

$$x_c'(t) = \begin{cases} \frac{dx_c}{dt}(t) & \text{derivatives} \\ x_c(t+T_s) & \text{discrete-time} \\ x_c(t+1) & \text{discrete-events} \end{cases}$$

Hybrid Dynamics - Discrete Part



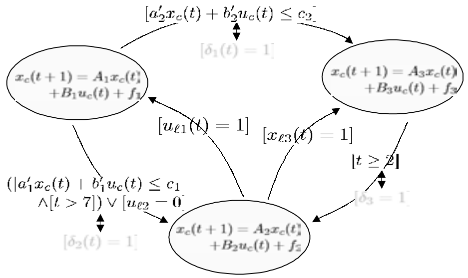
- Automata

$$x_\ell'(t) = f_{\text{BOOL}}(x_\ell(t), u_\ell(t), \delta(t))$$

$x_\ell \in \{0,1\}^{n_\ell}$	logic states
$u_\ell \in \{0,1\}^{m_\ell}$	logic inputs
$\delta \in \{0,1\}^{r_\ell}$	internal events

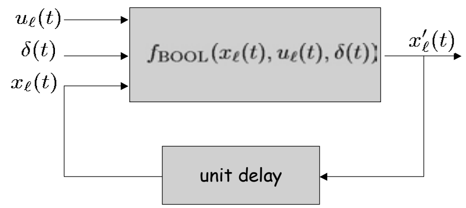
$$x_\ell'(t) = \begin{cases} x_\ell(t+T_s) & \text{discrete-time} \\ x_\ell(k+1) & \text{discrete-events} \end{cases}$$

Hybrid Dynamics



- Continuous dynamics change according to:
  1. Exogenous logic inputs $[u_\ell(t) = 1]$
  2. Threshold conditions $[a' x_c(t) \le b]$
  3. Time conditions $[t \ge 2]$
  4. Any logic combination of the former
- Reset conditions: possible (continuous time case more tricky)

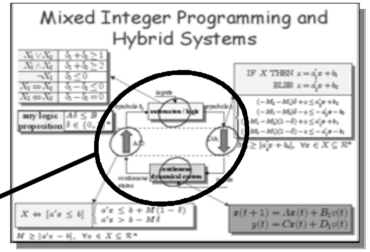
Hybrid Dynamics



- Discrete dynamics change according to a Boolean fnc of:
  1. Previous logic states $x_\ell(t)$
  2. Exogenous logic inputs $u_\ell(t)$
  3. Threshold conditions $\delta(t)$

Computational Models of Hybrid Systems

Mixed Logical Dynamical Systems



Mixed Logical Dynamical (MLD) form

$$
\begin{aligned}
x(t+1) &= Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) \\
y(t) &= Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) \\
E_2\delta(t) + E_3z(t) &\leq E_4x(t) + E_1u(t) + E_5
\end{aligned}
$$

(Bemporad, Morari, *Automatica*, 1999)

$x, y, u = \begin{bmatrix} \star_c \\ \star_\ell \end{bmatrix}, \star_c \in \mathbf{R}^{n_c}, \star_\ell \in \{0,1\}^{n_\ell}, \;\; z \in \mathbf{R}^{r_c}, \;\; \delta \in \{0,1\}^{r_\ell}$

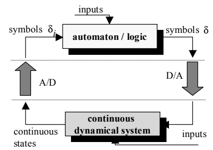
HYSDEL

HYSDEL

(HYbrid Systems DEscription Language)

- Describe *hybrid systems:*

- Automata
- Logic
- Lin. Dynamics
- Interfaces
- Constraints



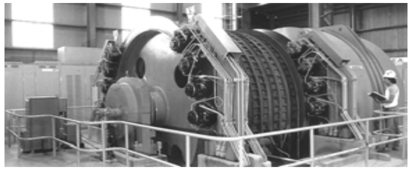
(Torrisi, Bemporad, Mignone, 2000)

- Automatically generate MLD models in Matlab

- MLD model is not unique in terms of the number of auxiliary variables → optimize model (minimize # binary variables !)

http://control.ethz.ch/~hybrid/hysdel

AD and DA



Nonlinear amplification unit

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

```
SYSTEM motor {
   INTERFACE {
      STATE {
         REAL ucomp; }
      INPUT {
         REAL u [0,10]; }
      PARAMETER {
         REAL ut   = 1;
         REAL e    = 1e-6;}
   } /* end interface */
```

```
IMPLEMENTATION {
      AUX {
         REAL unl;
         BOOL th; }
      AD {
         th = ut - u <= 0; }
      DA {
         unl = { IF th THEN 2.3*u - 1.3*ut
                      ELSE u };  }
      CONTINUOUS {
         ucomp = unl; }
   } /* end implementation */
} /* end system */
```

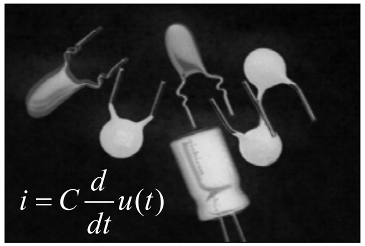
LOGIC

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

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

```
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 */
```

CONTINUOUS

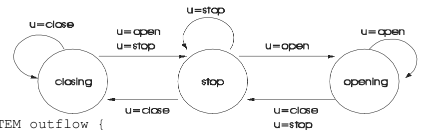


Apply forward difference rule:

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

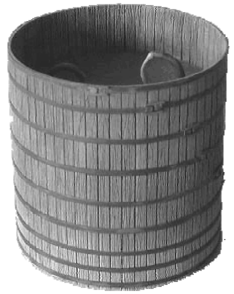
```
SYSTEM capacitorD {
   INTERFACE {
      STATE {
         REAL u; }
      PARAMETER {
         REAL R = 1e4;
         REAL C = 1e-4;
         REAL T = 1e-1; }
   } /* end interface */

   IMPLEMENTATION {

      CONTINUOUS {
        u = u - T/C/R*i;  }

   } /* end implementation */
} /* end system */
```

AUTOMATA



```
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 */
```

MUST



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

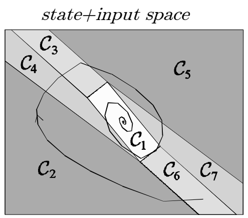
```
SYSTEM watertank {
   INTERFACE {
      STATE {
         REAL h; }
      INPUT {
         REAL Q; }
      PARAMETER {
         REAL hmax = 0.3;
         REAL k    = 1; }
   } /* end interface */

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

Realization and Transformation

(other state-space hybrid models)

Existing Hybrid Models

• Piecewise affine (PWA) systems (Sontag, 1981, 1996)



• Polyhedral partition of state+input space

$$\mathcal{X}_i = \{ \begin{bmatrix} x \\ u \end{bmatrix} : \; H_i \begin{bmatrix} x \\ u \end{bmatrix} \le K_i \}, \;\; i = 1, \ldots, s$$

• Affine dynamics in each region

$$x(t+1) = A_{i(t)} x(t) + B_{i(t)} u(t) + f_{i(t)} \quad \text{if} \quad \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \mathcal{X}_{i(t)}$$

• Can approximate nonlinear dynamics arbitrarily well

Existing Hybrid Models

• Linear complementarity (LC) systems (Heemels, 1999)

$$\begin{aligned} x(t+1) &= Ax(t) + B_1 u(t) + B_2 w(t) \\ y(t) &= Cx(t) + D_1 u(t) + D_2 w(t) \\ v(t) &= E_1 x(t) + E_2 u(t) + E_3 w(t) + e_4 \\ 0 &\le v(t) \perp w(t) \ge 0 \end{aligned}$$

Ex: mechanical systems
circuits with diodes etc.

• Extended linear complementarity (ELC) systems (De Schutter, De Moor, 2000)
Generalization of LC systems

• Min-max-plus-scaling (MMPS) systems (De Schutter, Van Den Boom, 2000)

$$\begin{aligned} x(t+1) &= M_x(x(t), u(t), d(t)) \\ y(t) &= M_y(x(t), u(t), d(t)) \\ 0 &\ge M_c(x(t), u(t), d(t)) \end{aligned}$$

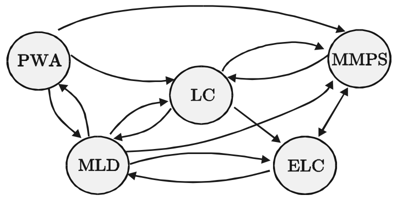
MMPS function: defined by the grammar

$M := x_i | \alpha | \max(M_1, M_2) | \min(M_1, M_2) | M_1 + M_2 | \beta M_1$

Example: $x(t+1) = 2\max(x(t), 0) + \min(-\frac{1}{2}u(t), 1)$

Used for modeling discrete-event systems (t=event counter)

Equivalence Results

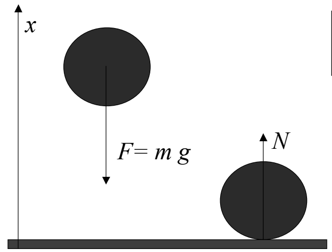


Theorem 1 *All the above five classes of discrete-time hybrid models are equivalent (possibly under additional assumptions, like boundedness of input and state variables)*

(Heemels, De Schutter, Bemporad, *Automatica,* 2001 + CDC2001)

Theoretical properties and analysis/synthesis tools can be transferred from one class to another !

Example: Bouncing Ball



$$\ddot{x} = -g$$
$$x \leq 0 \Rightarrow \dot{x}(t^+) = -(1-\alpha)\dot{x}(t^-)$$

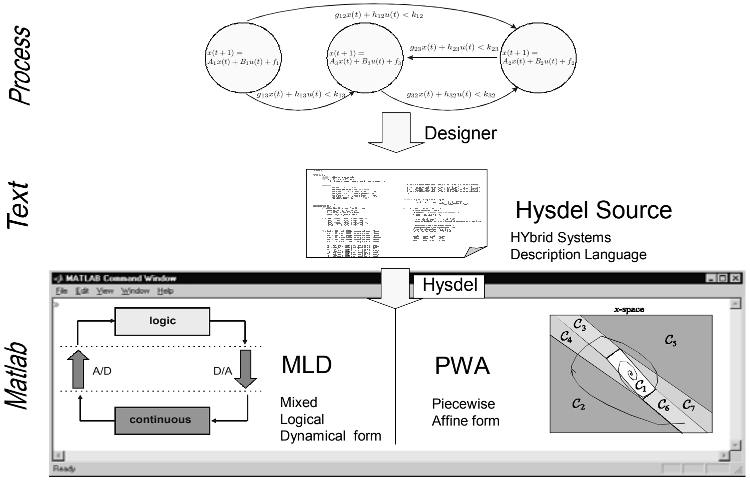
$$\alpha \in [0, 1]$$

How to model this system in MLD form?

HYSDEL - Bouncing Ball

```
SYSTEM ball {
INTERFACE {
/* Description of variables and constants */
        STATE { REAL height    [-10,10];
                REAL velocity  [-100,100];
        }
        PARAMETER {
                REAL g=9.8;
                REAL dissipation=.4;  /* 0=elastic, 1=completely anelastic */
                REAL e=1e-6;
                REAL Ts=.05;
        }
}
IMPLEMENTATION {
        AUX {   REAL hnext;
                REAL vnext;
                BOOL negative;
        }
        AD {
                negative = height <= 0 [hmax,hmin,e];
        }
        DA {    hnext = {  IF negative THEN height-Ts*velocity
                           ELSE height+Ts*velocity-Ts*Ts*g                  };

                vnext = {  IF negative THEN -(1-dissipation)*velocity
                           ELSE velocity-Ts*g                               };
        }
        CONTINUOUS {
                height   = hnext;
                velocity = vnext;
        }

}
}
```

Modeling Flow



System Theory for Hybrid Systems

- Analysis
  - Realization & Transformation
  - Well-posedness
  - Stability
  - Reachability (=Verification)
  - Observability
- Synthesis
  - Control
  - State estimation
  - Identification
  - Modeling language

Controller Synthesis

Optimal Control of Hybrid Systems

- Finite-time optimal control problem:

$$\min_{\xi} J(\xi, x(0), r) \triangleq \sum_{k=0}^{T-1} \|Q(y(k) - r)\| + \|R(u(k) - u_r)\| + \sigma\left(\|\delta(k) - \delta_r\| + \|z(k) - z_r\| + \|x(k) - x_r\|\right)$$

$$\text{subj. to} \begin{cases} \text{MLD model} \\ x(t|t) = x(t) \\ x(t+T|t) = x_r \end{cases}$$

$$\xi = [u(0), \ldots, u(T-1), \delta(0), \ldots, \delta(T-1), z(0), \ldots, z(T-1)]$$

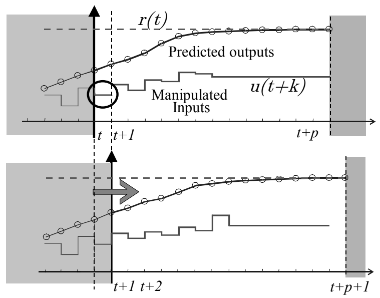
- Solution
  - Squared 2-norm: Mixed-Integer Quadratic Program (MIQP)
  - ∞ or 1-norm: Mixed Integer Linear Program (MILP)

Model Predictive Control

- At time t :

  Solve an optimal control problem over a finite future horizon p :

  - minimize performance
  - subject to constraints



- Only apply the first optimal move $u^*(t)$
- Get new measurements, and repeat the optimization at time $t+1$

Advantage of on-line optimization: ***FEEDBACK!***

Closed-Loop Stability

Theorem 1 *Let* (x_r, u_r) *be the equilibrium pair for the set point* r. *Assume that the optimization problem is feasible at time* $t = 0$. *Then* $\forall Q, R \succ 0$, $\sigma > 0$, *the predictive controller stabilizes the MLD system*

$$\lim_{t\to\infty} y(t) = r \qquad \lim_{t\to\infty} u(t) = u_r$$

$\lim_{t\to\infty} x(t) = x_r$, $\lim_{t\to\infty} z(t) = z_r$, $\lim_{t\to\infty} \delta(t) = \delta_r$, *and all the constraints are fulfilled.*

(Bemporad, Morari, *Automatica,* 1999)

Proof: use optimal value function as a Lyapunov function

Mixed-Integer Program Solvers

- Mixed-Integer Programming is *NP*-hard

BUT

- General purpose Branch & Bound/Branch & Cut solvers available for MILP (CPLEX) and MIQP (Fletcher-Leyffer, Sahinidis, XPRESS-MP)
  Free Matlab MILP/MIQP solver (Bemporad, Mignone, 1999)

  More solvers and benchmarks: `http://plato.la.asu.edu/bench.html`

- No need to reach global optimum (see proof of the theorem), although performance deteriorates

On-Line vs. Off-Line Optimization

$$\min_U J(U, x(t)) \triangleq \sum_{k=0}^{T-1} \|Qy(t+k+1|t)\|_\infty + \|Ru(t+k)\|_\infty$$

$$\text{subj. to} \begin{cases} \text{MLD model} \\ x(t|t) = x(t) \\ x(t+T|t) = 0 \end{cases}$$

- **On-line optimization**: given $x(t)$, solve the problem at each time step t

  Mixed-Integer Linear Program (MILP)

- Good for large sampling times (e.g., 1 h) / expensive hardware …

  … but not for fast sampling (e.g. 10 ms) / cheap hardware !

- **Off-line optimization**: get the explicit solution of the MPC controller by solving the MILP **for all** $x(t)$

$$\min_\xi J(\xi, x(t)) \triangleq f'\xi$$
$$\text{s.t.}\, G\xi \le W + Fx(t)$$

multi-parametric Mixed Integer Linear Program (mp-MILP)

Explicit Form of Model Predictive Control

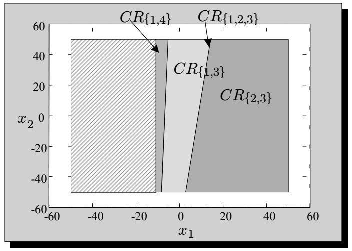
via Multiparametric Programming

Example of Multiparametric Solution

Multiparametric LP ($\xi \in \mathbf{R}^2$)

$$\min_{\xi} \; -3\xi_1 - 8\xi_2$$

$$\text{s.t.} \begin{cases} \xi_1 + \xi_2 & \leq 13 + x_1 \\ 5\xi_1 - 4\xi_2 & \leq 20 \\ -8\xi_1 + 22\xi_2 & \leq 121 + x_2 \\ -4\xi_1 - \xi_2 & \leq -8 \\ -\xi_1 & \leq 0 \\ -\xi_2 & \leq 0 \end{cases}$$



$$\xi(x) = \begin{cases} \begin{bmatrix} 0.00 & 0.05 \\ 0 & 0.06 \end{bmatrix} x + \begin{bmatrix} 11.85 \\ 9.80 \end{bmatrix} & \text{if } \begin{bmatrix} 0.02 & 0.00 \\ 0.00 & 0.02 \\ 0.00 & -0.02 \\ -0.12 & 0.01 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \\ -1.00 \end{bmatrix} & \text{CR}_{\{2,3\}} \\ \begin{bmatrix} 0.73 & -0.03 \\ 0.27 & 0.03 \end{bmatrix} x + \begin{bmatrix} 5.50 \\ 7.50 \end{bmatrix} & \text{if } \begin{bmatrix} 0.00 & 0.02 \\ 0.00 & -0.02 \\ 0.12 & -0.01 \\ -0.15 & 0.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \end{bmatrix} & \text{CR}_{\{1,3\}} \\ \begin{bmatrix} -0.33 & 0.00 \\ 1.33 & 0 \end{bmatrix} x + \begin{bmatrix} -1.67 \\ 14.67 \end{bmatrix} & \text{if } \begin{bmatrix} 0.00 & 0.02 \\ 0.00 & -0.02 \\ 0.15 & -0.00 \\ -0.09 & 0.00 \end{bmatrix} x \leq \begin{bmatrix} 1.00 \\ 1.00 \\ -1.00 \\ 1.00 \end{bmatrix} & \text{CR}_{\{1,4\}} \end{cases}$$

Multiparametric MILP

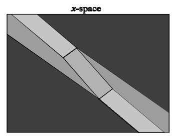
$$\min_{\xi=\{\xi_c,\xi_d\}} \; f'\xi_c + d'\xi_d \qquad \xi_c \in \mathbf{R}^n$$
$$\text{s.t. } G\xi_c + E\xi_d \leq W + Fx \qquad \xi_d \in \{0,1\}^m$$

- mp-MILP can be solved (by alternating MILPs and mp-LPs)

(Dua, Pistikopoulos, 1999)

- **Theorem:** The multiparametric solution $\xi^*(x)$ is piecewise affine
- **Corollary: The MPC controller is piecewise affine in** x

$$u(x) = \begin{cases} F_1 x + G_1 & \text{if } H_1 x \leq K_1 \\ \vdots & \vdots \\ F_N x + G_N & \text{if } H_N x \leq K_N \end{cases}$$

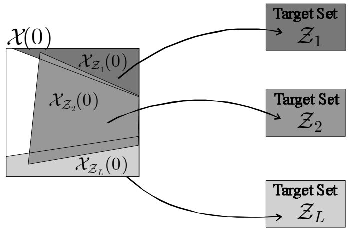


- Remarks on explicit MPC law:
  - **Automatic partitioning** of state-space (no gridding!)
  - **Stability** guarantee (value function=PWL Lyapunov function)

Reachability Analysis (Verification)

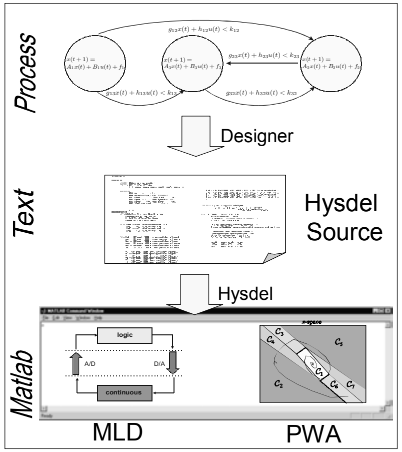
Reachability Analysis/Verification

(Bemporad, Torrisi, Morari, HSCC 2000)



- Efficient algorithms for reachability analysis of MLD/PWA systems were developed during the VHS project
- Software available in Matlab (requires `fabio.dll`)

Hysdel for Verification



- Inputs (e.g.: disturbances, set-points)
- Finite-time reachability analysis
- Logic-based, threshold-based, and time-based verification queries
- Reachability-based optimization

Examples

Traction Control System

(F. Borrelli, A. Bemporad, M. Fodor, D. Hrovat)

ETH Eidgenössische Technische Hochschule Zürich

Vehicle Traction Control

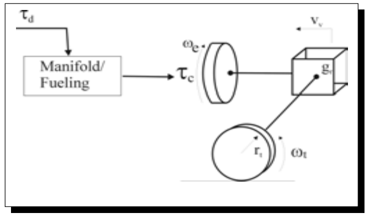
Improve driver's ability to control a vehicle
under adverse external conditions (wet or icy roads)

Model
nonlinear, uncertain, constraints

Controller
suitable for real-time implementation

MLD hybrid framework + optimization-based control strategy

Simple Traction Model

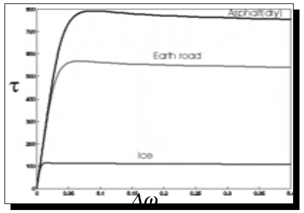


- Mechanical system

$$\dot{\omega}_e = \frac{1}{J_e}\left(\tau_c - b_e\omega_e - \frac{\tau_t}{g_r}\right)$$
$$\dot{v}_v = \frac{\tau_t}{m_v r_t}$$

- Manifold/fueling dynamics

$$\tau_c = b_i \tau_d (t - \tau_f)$$

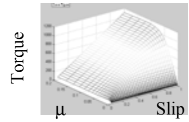


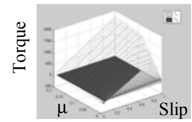
- Tire torque τ_t is a function of slip $\Delta\omega$ and road surface adhesion μ

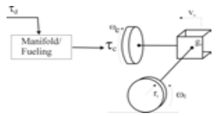
$$\Delta\omega = \frac{\omega_e}{g_r} - \frac{v_v}{r_t}$$

Wheel slip

Hybrid Model







Nonlinear tire torque $\tau_t = f(\Delta\omega, \mu)$

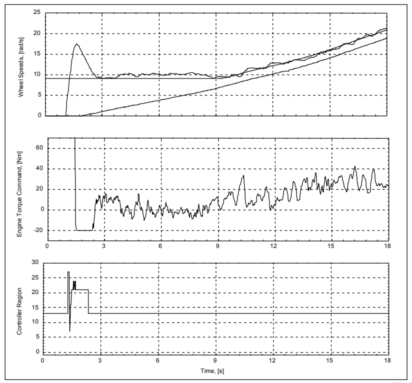
PWA Approximation

(PWL Toolbox, Julian, 1999) (Ferrari et al., HSCC'01)

HYSDEL
(Hybrid Systems Description Language)

→ Mixed-Logical Dynamical (MLD) Hybrid Model (discrete time)

EXPERIMENTAL RESULTS



Experiment

- >500 regions
- 20ms sampling time
- Pentium 266Mhz + Labview

Comments from Ford:

- Performance of the hybrid MPC controller is quite good given the limited development time, oversimplified plant model used, and minimal development iterations.
- The hybrid MPC controller requires much less supervision by logical constructs than controllers developed with traditional techniques.

Ford Motor Company

Hybrid Control Example: Cruise Control System

(A. Bemporad, F.D. Torrisi)

Hybrid Control Problem

Renault Clio 1.9 DTI RXE

GOAL:

command gear ratio, gas pedal, and brakes to **track** a desired speed and minimize consumption

Hysdel Model

(Bemporad, Torrisi, 2000)

```
SYSTEM car {

INTERFACE {
        STATE { REAL position, speed; }
        INPUT { REAL torque, F_brake;
                BOOL gear1, gear2, gear3, gear4, gear5, gearR;  }

        PARAMETER{
                REAL mass = 1020; /* kg */
                REAL beta_friction = 25; /* W/m*s */
                REAL Rgear1 = 3.7271; REAL Rgear2 = 2.048;
                REAL Rgear3 = 1.321; REAL Rgear4 = 0.971;
                REAL Rgear5 = 0.756; REAL RgearR = -3.545;
                REAL wheel_rim = 14;        /* in */
                ...

                }  }
IMPLEMENTATION {
        AUX {REAL Fe1, Fe2, Fe3, Fe4, Fe5, FeR;
             REAL w1, w2, w3, w4, w5, wR;
             BOOL dPWL1,dPWL2,dPWL3,dPWL4;
             REAL DCe1,DCe2,DCe3,DCe4; }

        AD { dPWL1 = wPWL1-(w1+w2+w3+w4+w5+wR)<=0;
             dPWL2 = wPWL2-(w1+w2+w3+w4+w5+wR)<=0;
             dPWL3 = wPWL3-(w1+w2+w3+w4+w5+wR)<=0;
             dPWL4 = wPWL4-(w1+w2+w3+w4+w5+wR)<=0; }


        DA { Fe1 = {IF gear1  THEN torque/speed_factor*Rgear1;
             Fe2 = {IF gear2  THEN torque/speed_factor*Rgear2;
             Fe3 = {IF gear3  THEN torque/speed_factor*Rgear3;
             Fe4 = {IF gear4  THEN torque/speed_factor*Rgear4;
             Fe5 = {IF gear5  THEN torque/speed_factor*Rgear5;
             FeR = {IF gearR  THEN torque/speed_factor*RgearR;

             w1 = {IF gear1  THEN speed/speed_factor*Rgear1;
             w2 = {IF gear2  THEN speed/speed_factor*Rgear2;
             w3 = {IF gear3  THEN speed/speed_factor*Rgear3;
             w4 = {IF gear4  THEN speed/speed_factor*Rgear4;
             w5 = {IF gear5  THEN speed/speed_factor*Rgear5;
             wR = {IF gearR  THEN speed/speed_factor*RgearR;

             DCe1 = {IF dPWL1 THEN (aPWL2-aPWL1)+(bPWL2-bPWL1)*(w1+w2+w3+w4+w5+wR);
             DCe2 = {IF dPWL2 THEN (aPWL3-aPWL2)+(bPWL3-bPWL2)*(w1+w2+w3+w4+w5+wR);
             DCe3 = {IF dPWL3 THEN (aPWL4-aPWL3)+(bPWL4-bPWL3)*(w1+w2+w3+w4+w5+wR);
             DCe4 = {IF dPWL4 THEN (aPWL5-aPWL4)+(bPWL5-bPWL4)*(w1+w2+w3+w4+w5+wR);
             }

        CONTINUOUS { position = position+Ts*speed;
                     speed = speed+Ts/mass*(Fe1+Fe2+Fe3+Fe4+Fe5+FeR-
                                      F_brake-beta_friction*speed);

        MUST    {  wemin <= w1+w2+w3+w4+w5+wR;
                   w1+w2+w3+w4+w5+wR <= wemax;
                   -F_brake <=0; /* brakes cannot accelerate ! */
                   F_brake <= max_brake_force;

                   -torque-(alpha1+beta1*(w1+w2+w3+w4+w5+wR)) <=0;
                   torque-(aPWL1+bPWL1*(w1+w2+w3+w4+w5+wR)+DCe1+DCe2+DCe3+DCe4)-1<=0;

                   -(gear1+gear2+gear3+gear4+gear5+gearR)<=-1;
                   (gear1+gear2+gear3+gear4+gear5+gearR)<=1;
                   Fe1+Fe2+Fe3+Fe4+Fe5+FeR <= max_force;
                   -Fe1-Fe2-Fe3-Fe4-Fe5-FeR <= -max_force;

                   dPWL4 -> dPWL3; dPWL4 -> dPWL2;
                   dPWL4 -> dPWL1; dPWL3 -> dPWL2;
                   dPWL3 -> dPWL1; dPWL2 -> dPWL1;  }
```

http://control.ethz.ch/~hybrid/hysdel

Hybrid Controller

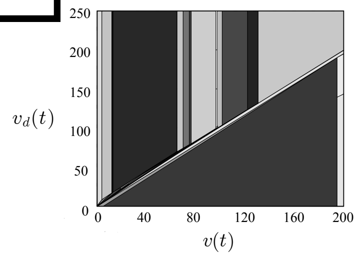
- Smoother tracking controller

$$\min_{u_t} J(u_t, x(t)) \triangleq |v(t+1|t) - v_d(t)| + \rho|\omega|$$

$$\text{subj. to} \begin{cases} |v(t+1|t) - v(t)| < T_s a_{\max} \\ \text{MLD model} \\ x(t|t) = x(t) \end{cases}$$

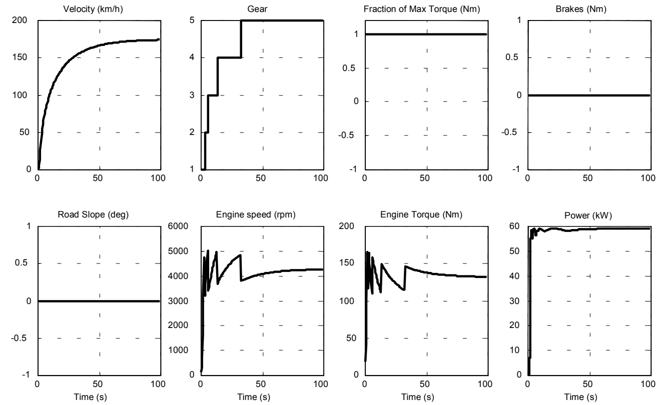
MILP optimization problem

Linear constraints	100
Continuous variables	19
Binary variables	10
Parameters	2
Time to solve mp-MILP (Sun Ultra 10)	28 m
Number of regions	**54**



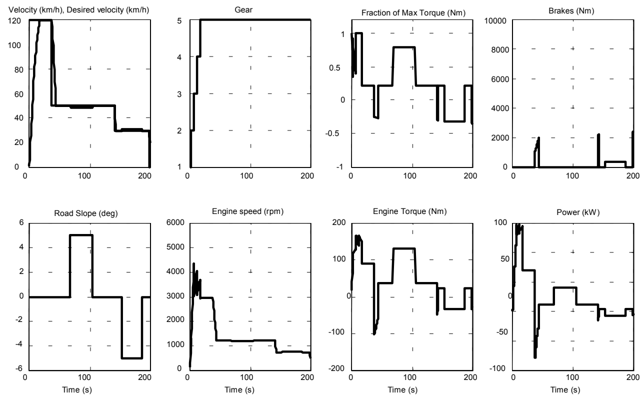
Hybrid Controller

- Max-speed controller



Hybrid Controller

- Smoother tracking controller

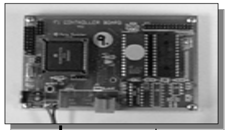


Verification of a Cruise Control System

(F.D. Torrisi, A. Bemporad)

Cruise Control System

Renault Clio 1.9 DTI RXE



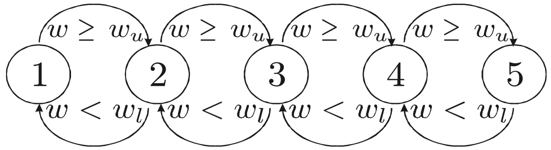
GOAL:

Verify if a given switching controller satisfies certain specifications

(Torrisi, Bemporad, 2001)

Cruise Control System

Gear selector:



Speed controller:

$$e(t+1) = e(t) + T_s(v_r(t) - v(t))$$ + saturation

$$u_t(t) = \begin{cases} k_t(v_r(t) - v(t)) + i_t e(t) & \text{if } v(t) < v_r(t) + 1 \\ 0 & \text{otherwise} \end{cases}$$

$$u_b(t) = \begin{cases} k_b(v_r(t) - v(t)) & \text{if } v(t) \geq v_r(t) + 1 \\ 0 & \text{otherwise} \end{cases}$$

Hysdel Model (HYbrid Systems DEscription Language)

```
SYSTEM car {
INTERFACE {
    STATE { REAL speed, err, vr; BOOL gear1, gear2, gear3, gear4, gear5; }
    PARAMETER {…}}
IMPLEMENTATION {
    AUX {
        REAL Fe1, Fe2, Fe3, Fe4, Fe5, w1, w2, w3, w4, w5, DCe1, DCe2, DCe3, DCe4,zut, zub, ierr, torque, F_brake;
        BOOL dPWL1, dPWL2, dPWL3, dPWL4, sd, su, verr, sat_torque, sat_F_brake, no_sat;}
    LOGIC { no_sat = ~(sat_torque | sat_F_brake) & verr; }
    AD {dPWL1 = wPWL1 - (w1 + w2 + w3 + w4 + w5) <= 0; dPWL2 = wPWL2 - (w1 + w2 + w3 + w4 + w5) <= 0;
        dPWL3 = wPWL3 - (w1 + w2 + w3 + w4 + w5) <= 0; dPWL4 = wPWL4 - (w1 + w2 + w3 + w4 + w5) <= 0;

        sd = (w1 + w2 + w3 + w4 + w5) - wl <= 0; su = wu - (w1 + w2 + w3 + w4 + w5) <= 0; verr = speed - vr - 2 <= 0;
        sat_torque = - zut + (DCe1 + DCe2 + DCe3 + DCe4) + 1 <= 0; sat_F_brake = - zub + max_brake_force <= 0;          }
    DA {Fe1 = {IF gear1 THEN torque / speed_factor * Rgear1}; Fe2 = {IF gear2 THEN torque / speed_factor * Rgear2};
        Fe3 = {IF gear3 THEN torque / speed_factor * Rgear3}; Fe4 = {IF gear4 THEN torque / speed_factor * Rgear4};
        Fe5 = {IF gear5 THEN torque / speed_factor * Rgear5};

        w1 = {IF gear1 THEN speed / speed_factor * Rgear1}; w2 = {IF gear2 THEN speed / speed_factor * Rgear2};
        w3 = {IF gear3 THEN speed / speed_factor * Rgear3}; w4 = {IF gear4 THEN speed / speed_factor * Rgear4};
        w5 = {IF gear5 THEN speed / speed_factor * Rgear5};

        DCe1 = {IF dPWL1 THEN (aPWL2) + (bPWL2) * (w1 + w2 + w3 + w4 + w5) ELSE (aPWL1) + (bPWL1) * (w1 + w2 + w3 + w4 + w5)};
        DCe2 = {IF dPWL2 THEN (aPWL3 - aPWL2) + (bPWL3 - bPWL2) * (w1 + w2 + w3 + w4 + w5)};
        DCe3 = {IF dPWL3 THEN (aPWL4 - aPWL3) + (bPWL4 - bPWL3) * (w1 + w2 + w3 + w4 + w5)};
        DCe4 = {IF dPWL4 THEN (aPWL5 - aPWL4) + (bPWL5 - bPWL4) * (w1 + w2 + w3 + w4 + w5)};

        zut = {IF verr THEN kt * (vr - speed) + it * err}; zub = {IF ~verr THEN - kb * (vr - speed) - ib * err};
        torque ={IF sat_torque THEN (DCe1 + DCe2 + DCe3 + DCe4) + 1 ELSE zut}; F_brake = {IF sat_F_brake THEN max_brake_force ELSE zub};
        ierr = {IF no_sat THEN err + Ts * (vr - speed)};}
    CONTINUOUS {
        speed = speed + Ts / mass * (Fe1 + Fe2 + Fe3 + Fe4 + Fe5 - F_brake - beta_friction * speed); err = ierr; vr = vr;}
    AUTOMATA {
        gear1 =                 (gear2 & sd) | (gear1 &       ~su); gear2 = (gear1 & su) | (gear3 & sd) | (gear2 & ~sd & ~su);
        gear3 = (gear2 & su) | (gear4 & sd) | (gear3 & ~sd & ~su); gear4 = (gear3 & su) | (gear5 & sd) | (gear4 & ~sd & ~su);
        gear5 = (gear4 & su)                | (gear5 & ~sd );}
    MUST {
        -w1 <= -wemin; w1 <= wemax; -w2 <= -wemin; w2 <= wemax; -w3 <= -wemin; w3 <= wemax; -w4 <= -wemin; w4 <= wemax; -w5 <= -wemin;
        w5 <= wemax; -F_brake <= 0; F_brake <= max_brake_force; torque - (DCe1 + DCe2 + DCe3 + DCe4) - 1 <= 0;
        -((REAL gear1) + (REAL gear2) + (REAL gear3) + (REAL gear4) + (REAL gear5)) <= -0.9999;
        (REAL gear1) + (REAL gear2) + (REAL gear3) + (REAL gear4) + (REAL gear5)  <= 1.0001;
        dPWL4 -> dPWL3; dPWL4 -> dPWL2; dPWL4 -> dPWL1; dPWL3 -> dPWL2; dPWL3 -> dPWL1; dPWL2 -> dPWL1;}}
```

http://control.ethz.ch/~hybrid/hysdel (Torrisi, Bemporad, Mignone, 2000)

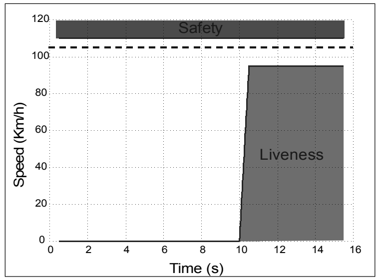
Hybrid Model

- MLD model

$$\begin{aligned} x(t+1) &= Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) \\ y(t) &= Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) \\ E_2\delta(t) + E_3z(t) &\leq E_4x(t) + E_1u(t) + E_5 \end{aligned}$$

- 3 continuous states: v, v_r, e (vehicle speed, reference and tracking error)
- 5 binary states: g_1, g_2, g_3, g_4, g_5 (gears)
- 19 auxiliary continuous vars: (5 traction force, 5 engine speed, 5 reset/saturation, 4 PWL max engine torque)
- 15 auxiliary binary vars: (4 PWL max torque breakpoints, 4 saturations 5 logic updates, 2 gear switching conditions)
- 173 mixed-integer inequalities

Verification

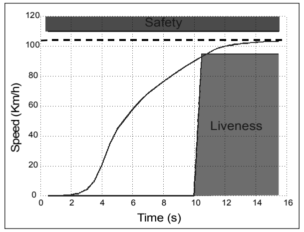


Verify that the cruise control reaches the desired speed reference ($\mathcal{Z}_2 = \{v, t : v < v_r - 2r_{\text{toll}}, t > 10/T_s\}$) and without driving above the limit ($\mathcal{Z}_1 = \{v : v > v_r + r_{\text{toll}}\}$)

$r_{\text{toll}} = 5 \text{ km/h}$

Verification Results

- For all $v_r \in [30, 70]$ km/h the controller satisfies both liveness & safety properties
- CPU time: ~2.5h on Matlab5.3, PC650MHz.



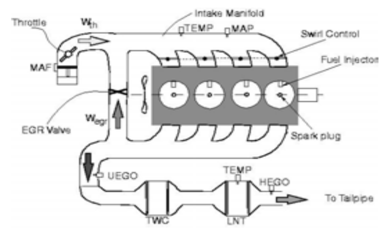
- For $v_r \in [30, 120]$ km/h the verification algorithm finds the first counterexample after ~7m

Hybrid Modeling and Control of a Direct Injection Stratified Charge (DISC) Engine

A. Bemporad, N. Giorgetti, I. Kolmanovsky, D. Hrovat

Nonlinear Hybrid Model

- States/Controlled outputs:
  1. Intake manifold pressure
  2. Air-to-fuel ratio
  3. Engine brake torque
- Inputs (continuous):
  1. Mass flow rate through throttle
  2. Mass flow rate of fuel
  3. Spark timing

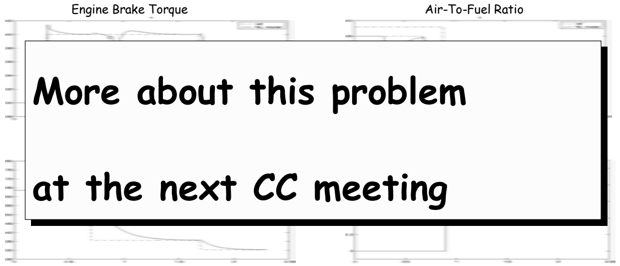


- Inputs (binary):
  1. ρ = Regime of combustion (stratified/homogeneous)
- Constraints on:
  1. Air-to-fuel ratio (due to engine roughness, misfiring, smoke emiss.)
  2. Spark timing (to avoid excessive engine roughness)
  3. Mass flow rate of throttle

⇒ Dynamic equations are nonlinear
Dynamics and constraints depend on ρ !

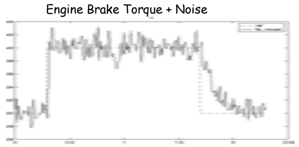
Hybrid MPC Control

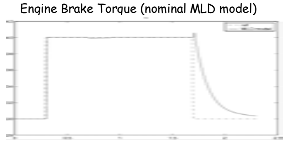
Closed-loop: Nonlinear hybrid model + Hybrid MPC controller



More about this problem

at the next CC meeting





The End

Announcements

1. CC project web site:

   http://www.dii.unisi.it/~hybrid/cc

Please contribute !!! (mailto: hybrid@dii.unisi.it)

2. IEEE Technical Committee on Hybrid Systems web site: http://www.ieeecss.org/TAB/

   http://www.dii.unisi.it/~hybrid/ieee

Please contribute !!! (mailto: hybrid@dii.unisi.it)