Modeling & Control of Hybrid Systems

Chapter 6 — Optimization-Based Control [1]

Manuel Mazo Jr.

Utrecht, July 6th, 2020

[1] Based on the original slides from Bart De Schutter

Outline

Outline

Optimization-based control

- Most often constrained to discrete-time (recast of original) dynamics;
- Are discrete-time hybrid systems really hybrid?
- Solutions rely often on mixed real-integer optimization (MILP, MIQP...)
- Finite horizon optimization problems.
- Receding horizon approaches (MPC) to extend to infinite horizon closed-loop control.

Outline

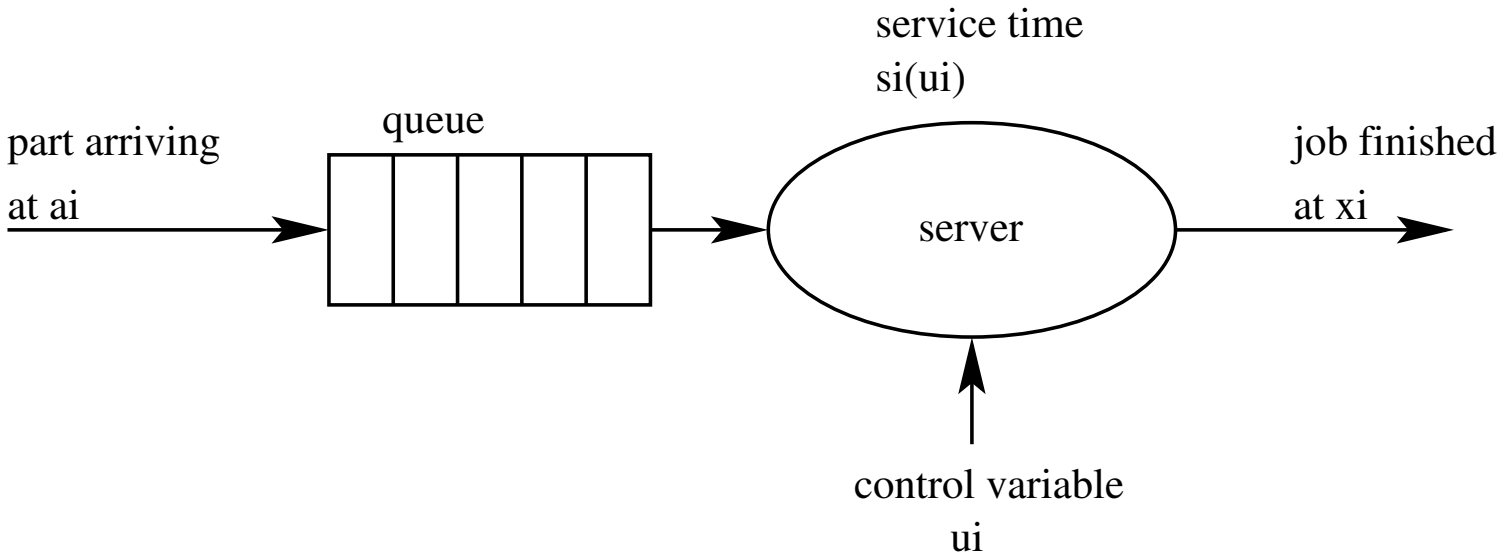
Hybrid manufacturing systems

- Manufacturing system: jobs move through network of work centers
- Jobs have
  - *temporal state* (event-driven): waiting time, departure time, ...
  - *physical state* (time-driven): temperature, size, weight, chemical composition, ...
- Trade-off between
  - temporal requirements on job completion times
  - physical requirements on quality of completed jobs

  assume higher quality $\rightarrow$ longer processing times

Cas01 C.G. Cassandras, D.L. Pepyne, and Y. Wardi. Optimal control of a class of hybrid systems. *IEEE Transactions on Automatic Control*, 46(3):398—415, March 2001

Hybrid manufacturing systems

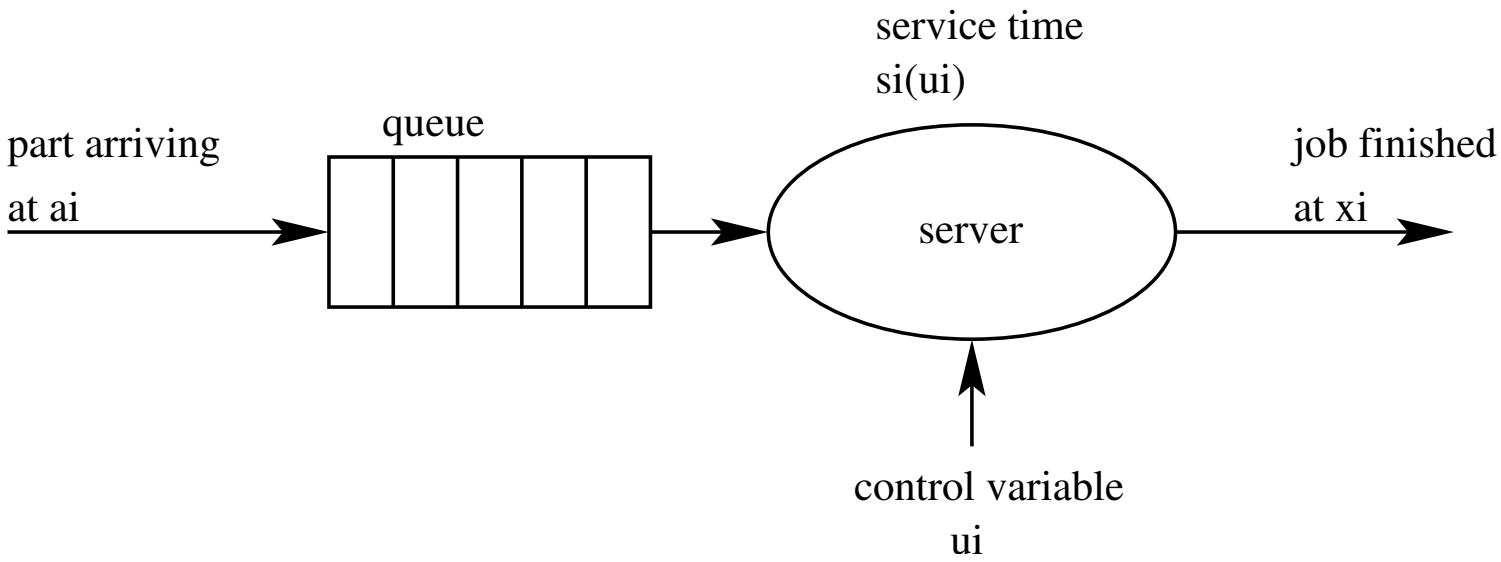


- Single-stage, single-server queueing system
- N jobs (each job corresponds to mode)
- Buffer with capacity $> N$
- As job i is processed, *physical state* z_i evolves according to

$$\dot{z}_i = g_i(z_i, u_i, t) \qquad \text{with } z_i(\tau_i) = \zeta_i$$

with τ_i time instant at which processing begins

Hybrid manufacturing systems



- *Control variable* u_i is used to attain final desired physical state:
  If $s_i(u_i)$ is *service time* and $\Gamma_i(u_i)$ is target quality set, then

$$s_i(u_i) = \min\{t \geq 0 \mid z_i(\tau_i + t) \in \Gamma_i(u_i)\}$$

- *Temporal state* x_i represents time when job is completed:
  If a_i is arrival time of job i, then

$$x_i = \max(x_{i-1}, a_i) + s_i(u_i) \qquad \text{(Lindley equation)}$$

Optimal control problem

Optimal control problem:

$$\min_{u_1,\ldots,u_N} J = \sum_{i=1}^{N} L_i(x_i, u_i)$$

subject to evolution equations for z_i and x_i

where $L(x_i, u_i)$ is cost function associated with job i

$\rightarrow$ classical discrete-time optimal control problems except for

- i does not count time steps
  $\rightarrow$ not really an issue
- max is non-differentiable for $a_i = x_{i-1}$
  $\rightarrow$ prevents use of standard gradient-based techniques
  $\rightarrow$ use non-differentiable calculus, generalized gradient

Optimality conditions

- Construct Lagrangian:

$$\mathcal{L}(x, \lambda, u) = \sum_{i=1}^{N} \big(L_i(x_i, u_i) + \lambda_i (\max(x_{i-1}, a_i) + s_i(u_i) - x_i) \big)$$

  where λ is co-state (lagrange multipliers)

- **Assumption:** costs L_i and s_i are continuously differentiable

- Ignoring non-differentiabilities associated with max, standard first-order necessary conditions for optimality require $\nabla \mathcal{L} = 0$, i.e.:

$$\frac{\partial \mathcal{L}}{\partial u_i} = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda_i} = 0, \quad \frac{\partial \mathcal{L}}{\partial x_i} = 0 \quad \text{for } i = 1, \ldots, N$$

Solution

- Results in
  - Stationarity condition: $\frac{\partial L_i(x_i, u_i)}{\partial u_i} + \lambda_i \frac{ds_i(u_i)}{du_i} = 0$
  - Temporal state equation: $x_i = \max(x_{i-1}, a_i) + s_i(u_i)$ with $x_0 = -\infty$
  - Co-state equation: $\lambda_i = \frac{\partial L_i(x_i, u_i)}{\partial x_i} + \lambda_{i+1} \frac{d \max(x_i, a_{i+1})}{dx_i}$ with final boundary condition

$$\lambda_N = \frac{\partial L_N(x_N, u_N)}{\partial x_N}$$

- Defines *two-point boundary-value problem* (TPBVP)

How to deal with non-differentiability

- max is Lipschitz continuous + differentiable except for $x_i = a_{i+1}$:

$$\frac{d \max(x_i, a_{i+1})}{dx_i} = \begin{cases} 0 & \text{if } x_i < a_{i+1} \\ 1 & \text{if } x_i > a_{i+1} \end{cases}$$

- Use *generalized gradient*:

Definition (Generalized gradient)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz continuous, and let $S(u)$ denote set of all sequences $\{u_m\}_{m=1}^{\infty}$ that satisfy

- $u_m \to u$ as $m \to \infty$
- gradient $\nabla f(u_m)$ exists for all m
- $\lim_{m\to\infty} \nabla f(u_m) = \phi$ exists

Then the *generalized gradient* $\partial f(u)$ is defined as convex hull of all limits ϕ corresponding to some sequence $\{u_m\}_{m=1}^{\infty}$ in $S(u)$

Generalized gradient properties

1. $\partial f(u)$ non-empty, compact, convex set in $\mathbb{R}^n$;
2. $\partial f(u) = \{\nabla f(u)\}$ iff f is continuously differentiable in some open set containing u;
3. if u is local minimum, then $0 \in \partial f(u)$
   $\rightarrow$ this becomes first-order optimality condition in non-smooth optimization

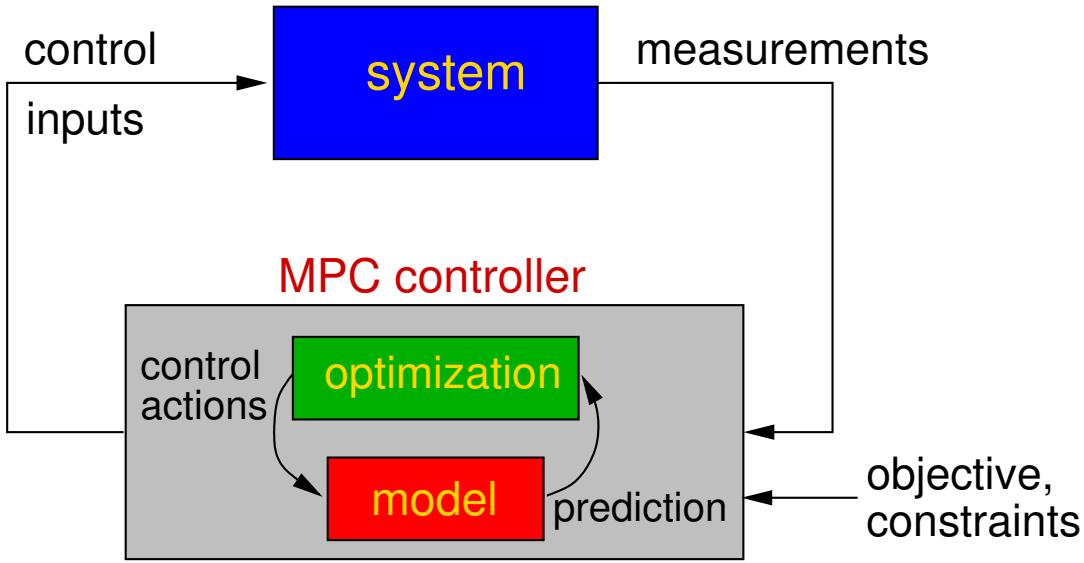
See lecture notes[2] for computation of $\partial \mathcal{L}$

[2] N.B.:In Lecture notes $\mathcal{L}$ is $\bar{J}$

Outline

Model predictive control (MPC)

- Very popular in process industry
- Model-based
- Easy to tune
- Multi-input multi-output (MIMO)
- Allows constraints on inputs and outputs
- Adaptive / receding horizon
- Uses on-line optimization



→ Simple extension to MLD, PWA, and MMPS systems

MPC (continued)

Discrete time approach, at sample step k:

- Use model to predict system output over prediction interval $[kT, (k+N_\mathrm{p})T]$ for given input sequence $u(k), \ldots, u(k+N_\mathrm{p}-1)$

  N_p: prediction horizon

  $\tilde{u}(k) = [\, u^\mathrm{T}(k) \;\; \ldots \;\; u^\mathrm{T}(k+N_\mathrm{p}-1) \,]^\mathrm{T}$

- Define performance criterion $J(k)$ over $[kT, (k+N_\mathrm{p})T]$, e.g.,
  $J(k) = \text{tracking error} + \lambda \cdot \text{input effort/energy}$
- Specify constraints on u, x, y
- Find the input sequence $\tilde{u}(k)$ that minimizes $J(k)$ subject to system equations + constraints

MPC (continued)

Receding horizon principle:

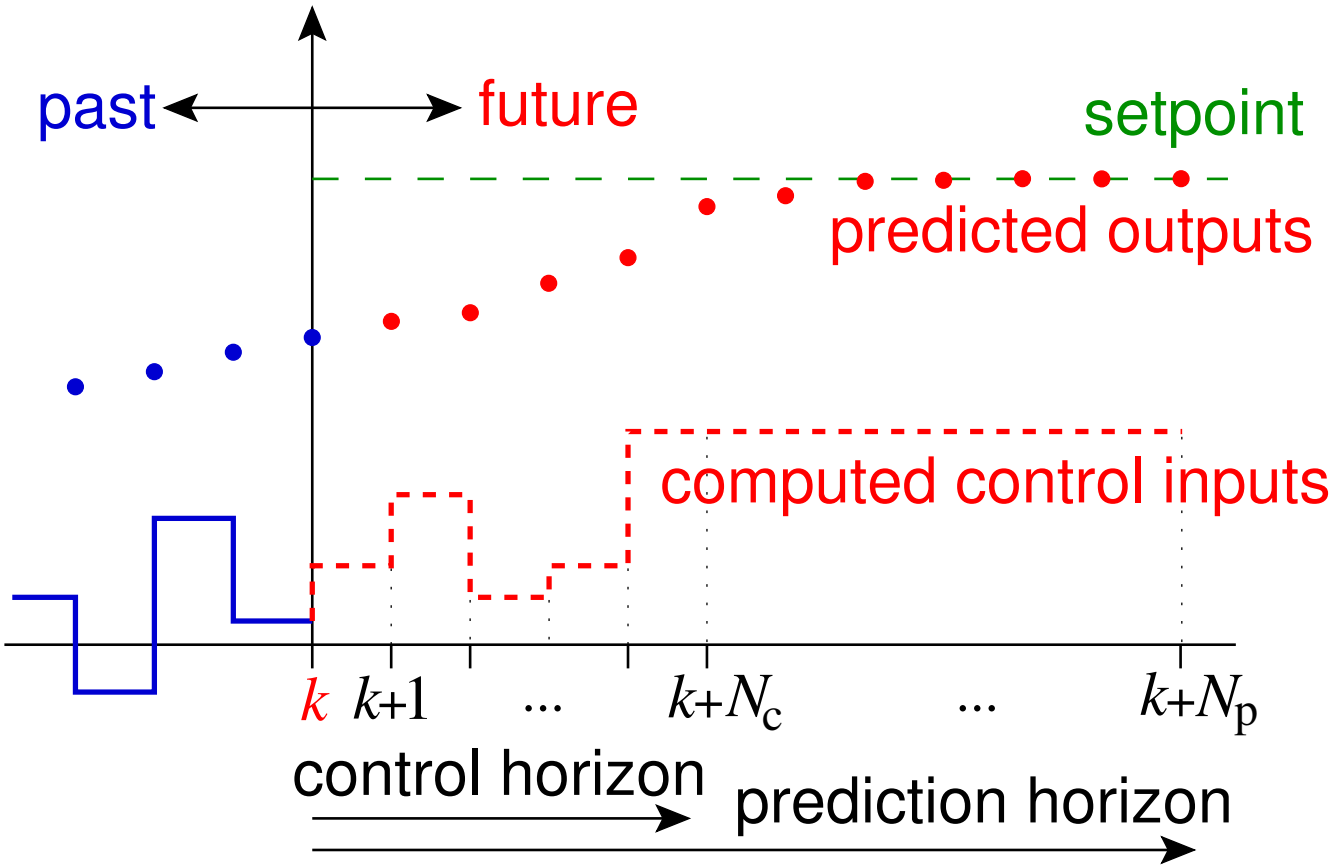
- Compute optimal input sequence $\tilde{u}(k)$
- Implement only first sample $u(k)$
- Update model & shift interval
- Restart optimization

Extra condition to reduce computational complexity:
control horizon N_c

$$u(k+j) = u(k+N_c-1) \quad \text{for } j = N_c, \ldots, N_p - 1$$

$\rightarrow$ smoother controller signal & stabilizing effect

MPC illustration



MPC for MLD systems

- Consider MLD system:

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k)$$
$$y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k)$$
$$E_1 x(k) + E_2 u(k) + E_3 \delta(k) + E_4 z(k) \leqslant g_5,$$

- $x(k) = [\, x_{\mathrm{r}}{}^{\mathrm{T}}(k)\ x_{\mathrm{b}}{}^{\mathrm{T}}(k)\,]^{\mathrm{T}}$ with $x_{\mathrm{r}}(k)$ real-valued, $x_{\mathrm{b}}(k)$ boolean
  $z(k)$: real-valued auxiliary variables
  $\delta(k)$: boolean auxiliary variables
- Consider equilibrium state/input/output $(x_{\mathrm{eq}}, u_{\mathrm{eq}}, y_{\mathrm{eq}}) \rightarrow (\delta_{\mathrm{eq}}, z_{\mathrm{eq}})$
- $\hat{x}(k+j|k)$: estimate of x at sample step $k+j$ based on information available at sample step k

Typical optimization problem

- Stabilize system to equilibrium state:

$$J(k) = \sum_{j=1}^{N_\mathrm{p}} \|\hat{x}(k+j|k) - x_\mathrm{eq}\|_{Q_x}^2 + \|u(k+j-1) - u_\mathrm{eq}\|_{Q_u}^2 + \|\hat{y}(k+j|k) - y_\mathrm{eq}\|_{Q_y}^2 + \|\hat{\delta}(k+j-1|k) - \delta_\mathrm{eq}\|_{Q_\delta}^2 + \|\hat{z}(k+j-1|k) - z_\mathrm{eq}\|_{Q_z}^2$$

  with $Q_\cdot \underset{(-)}{\geq} 0$

- End-point condition: $\hat{x}(k+N_\mathrm{p}|k) = x_\mathrm{eq}$
- Control horizon constraint: $u(k+j) = u(k+N_\mathrm{c}-1)$ for $j = N_\mathrm{c}, \ldots, N_\mathrm{p}-1$
- **Property**: If feasible solution exists for $x(0)$, then MPC input stabilizes system, i.e.,

$$\lim_{k\to\infty} x(k) = x_\mathrm{eq} \qquad \lim_{k\to\infty} \|y(k) - y_\mathrm{eq}\|_{Q_y} = 0 \qquad \lim_{k\to\infty} \|z(k) - z_\mathrm{eq}\|_{Q_z} = 0$$

$$\lim_{k\to\infty} u(k) = u_\mathrm{eq} \qquad \lim_{k\to\infty} \|\delta(k) - \delta_\mathrm{eq}\|_{Q_\delta} = 0$$

Algorithms for MLD-MPC

$\rightarrow$ mixed-integer quadratic programming (MIQP)

- Successive substitution of system equations:
  $\rightarrow \hat{x}(k+j|k)$ is linear function of $x(k)$, $\tilde{u}$, $\tilde{\delta}$ and $\tilde{z}$
  Also holds for $\hat{y}(k+j|k)$
- Define $\tilde{V}(k) = \left[\tilde{u}^{\mathrm{T}}(k) \;\; \tilde{\delta}^{\mathrm{T}}(k) \;\; \tilde{z}^{\mathrm{T}}(k)\right]^{\mathrm{T}}$
  $\rightarrow$ contains both real-valued and integer-valued components
- Results in

$$\min_{\tilde{V}(k)} \;\; \tilde{V}^{\mathrm{T}}(k) S_1 \tilde{V}(k) + 2(S_2 + x^{\mathrm{T}}(k) S_3) \tilde{V}(k) \tag{3.1}$$

$$\text{subject to} \;\; F_1 \tilde{V}(k) \leqslant F_2 + F_3 x(k) \;\; , \tag{3.2}$$

$=$ **MIQP problem**

Bem99 A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407—427, March 1999.

Algorithms for MLD-MPC (continued)

- MIQP = NP-hard
- For small-sized problems: cutting plane methods, decomposition methods, logic-based methods, *branch-and-bound* methods (tree search)
- If cost is 1 or ∞ norms $\rightarrow$ MILP
- Multi-parametric approach $\rightarrow$ explicit PWA solutions to MILP
- Software:
  - Multi-Parametric Toolbox (MPT) : http://control.ee.ethz.ch/~mpt/
  - Hybrid toolbox : http://www.ing.unitn.it/ bemporad/hybrid/toolbox/
  - TOMLAB, CPLEX, Xpress
  - NAG, Matlab NAG Toolbox

Continuous PWA systems

- Continuous PWA function $f : \mathbb{R}^n \to \mathbb{R}$:
  - domain space divided into polyhedral regions $R_{(1)}, \ldots, R_{(N)}$
  - in each region $R_{(i)}$ f can be expressed as

$$f(x) = \alpha_{(i)}^T x + \beta_{(i)}$$

  - f is continuous over border of any two regions
- Continuous PWA system:

$$x(k) = \mathcal{P}_x(x(k-1), u(k))$$
$$y(k) = \mathcal{P}_y(x(k), u(k))$$

with $\mathcal{P}_x$, $\mathcal{P}_y$ vector-valued continuous PWA functions

Max-min-plus-scaling (MMPS) systems

- MMPS function f is constructed recursively:

$$f := x_i \,|\, \alpha \,|\, \max(f_k, f_l) \,|\, \min(f_k, f_l) \,|\, f_k + f_l \,|\, \beta f_k$$

  with f_k, f_l again MMPS functions

- MMPS function is continuous
- MMPS system:

$$x(k) = \mathcal{M}_x(x(k-1), u(k))$$
$$y(k) = \mathcal{M}_y(x(k), u(k))$$

  with $\mathcal{M}_x$, $\mathcal{M}_y$ vector-valued MMPS functions

Equivalence of continuous PWA and MMPS systems

Previous result: (Chapter 2)
(General) PWA systems are equivalent to *constrained* MMPS systems

- Any MMPS function is also continuous PWA
- A continuous PWA function f can be rewritten as MMPS:

$$f = \max_j \min_i \left(\alpha_i^T x + \beta_i\right)$$

$\Rightarrow$ Continuous PWA functions $\equiv$ MMPS functions

$\Rightarrow$ Continuous PWA systems $\equiv$ MMPS systems:

$\Rightarrow$ use properties & techniques from continuous PWA systems for MMPS systems and vice versa

Canonical forms of MMPS functions

- Any MMPS function $f : \mathbb{R}^n \to \mathbb{R}$ can be rewritten into min-max canonical form

$$f = \min_i \max_j (\alpha_{(i,j)}^T x + \beta_{(i,j)})$$

or into max-min canonical form

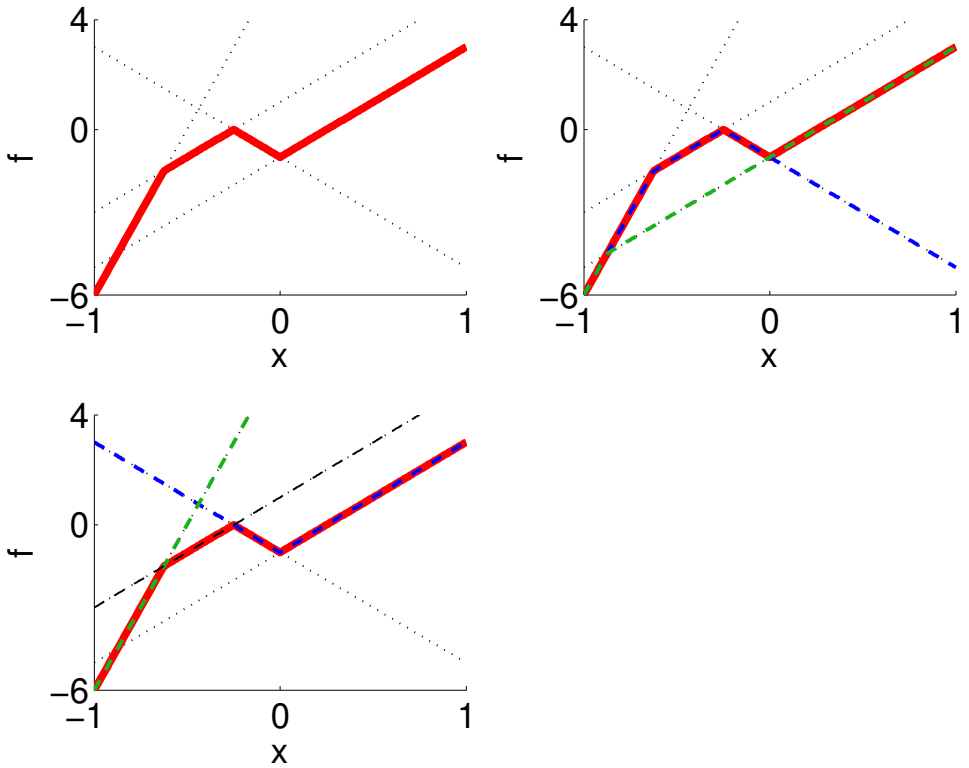
$$f = \max_i \min_j (\gamma_{(i,j)}^T x + \delta_{(i,j)})$$

Proof sketch: Use the properties:

- distributive properties of max and min w.r.t. $+$ and positive scalars;
- distributive properties of max and min w.r.t. each other.;
- $min(s) = -max(-s)$ and viceversa.

Example

$$
\begin{aligned}
f(x) &= \min(8x+6, 1) - 2\max\left(\min(2x+1, 1-2x), -2x\right) \\
&= \max\left(\min(12x+6, 4x+1, -4x-1), \min(12x+6, 4x-1)\right) \\
&= \min\left(\max(4x-1, -4x-1), 12x+6, 4x+1\right)
\end{aligned}
$$



MPC for MMPS systems

- Use MMPS model

$$x(k) = \mathcal{M}_x(x(k-1), u(k))$$
$$y(k) = \mathcal{M}_y(x(k), u(k))$$

  as
  - model of MMPS system
  - equivalent model of continuous PWA system
  - approximation of general smooth nonlinear system

- Prediction horizon: N_p
- Estimate $\hat{\mathrm{y}}(k+j|k)$ of output at sample step $k+j$:

$$\hat{\mathrm{y}}(k+j|k) = \mathrm{F}_j(\mathrm{x}(k-1), \mathrm{u}(k), \ldots, \mathrm{u}(k+j))$$

  $\rightarrow$ F_j is MMPS function!

MMPS Cost and constraint functions

- Reference signal: r
- Cost criterion J: reference tracking (J_{out}) vs control effort (J_{in}):

$$J(k) = J_{\text{out}}(k) + \lambda J_{\text{in}}(k) \qquad \text{with } \lambda > 0$$

- Some possible cost functions:

$$J_{\text{out},1}(k) = \|\tilde{\mathrm{y}}(k) - \tilde{\mathrm{r}}(k)\|_1 \qquad J_{\text{out},\infty}(k) = \|\tilde{\mathrm{y}}(k) - \tilde{\mathrm{r}}(k)\|_\infty$$
$$J_{\text{in},1}(k) = \|\tilde{\mathrm{u}}(k)\|_1 \qquad J_{\text{in},\infty}(k) = \|\tilde{\mathrm{u}}(k)\|_\infty$$

with

$$\tilde{\mathrm{u}}(k) = \begin{bmatrix} \mathrm{u}^T(k) & \dots & \mathrm{u}^T(k+N_{\mathrm{p}}-1) \end{bmatrix}^T$$
$$\tilde{\mathrm{y}}(k) = \begin{bmatrix} \hat{\mathrm{y}}^{\mathrm{T}}(k|k) & \dots & \hat{\mathrm{y}}^{\mathrm{T}}(k+N_{\mathrm{p}}-1|k) \end{bmatrix}^T$$
$$\tilde{\mathrm{r}}(k) = \begin{bmatrix} \mathrm{r}^T(k) & \dots & \mathrm{r}^T(k+N_{\mathrm{p}}-1) \end{bmatrix}^T$$

- Constraints on input and output signals:

$$C_{\mathrm{c}}(k, \mathrm{x}(k-1), \tilde{\mathrm{u}}(k), \tilde{\mathrm{y}}(k)) \geqslant 0$$

Algorithms for MMPS-MPC

- Nonlinear optimization (SQP, ELCP):
  - $\rightarrow$ local minima, excessive computation time
- MPC for mixed logical-dynamical (MLD) systems [Bemporad, Morari]:
  - $\rightarrow$ mixed real-integer quadratic programming problems
- Approach based on canonical forms:
  - $\rightarrow$ collection of linear programming problems

LP-based algorithm

Assume: linear (or convex) constraint in $\tilde{u}(k)$

$$P(k)\tilde{u}(k) + q(k) \geqslant 0$$

Recall: $J(k)$ is MMPS function

$$\Rightarrow J(k) = \max_i \left(\min_j (\gamma_{(i,j)}^T \tilde{u} + \delta_{(i,j)}) \right)$$

$$= \min_i \left(\max_j (\alpha_{(i,j)}^T \tilde{u} + \beta_{(i,j)}) \right)$$

$$\Rightarrow \min_{\tilde{u}} \; J(k) = \min_{\tilde{u}} \min_i \left(\max_j (\alpha_{(i,j)}^T \tilde{u} + \beta_{(i,j)}) \right)$$

$$= \min_i \underbrace{\min_{\tilde{u}} \left(\max_j (\alpha_{(i,j)}^T \tilde{u} + \beta_{(i,j)}) \right)}$$

$$\rightarrow \text{LP!}$$

LP-based algorithm (cont.)

LP i:

$$\min_{\tilde{u}} \; t$$

$$\text{s.t.} \begin{cases} t \geqslant \alpha_{(i,j)}^T \tilde{u} + \beta_{(i,j)} & \text{for all } j \\ P\tilde{u} + q \geqslant 0 \end{cases}$$

$\Rightarrow$ set of linear programming problems!

Example

PWA model:

$$y(k) = x(k) = \begin{cases} 0.5\,x(k-1) + 4\,u(k) - 1 & \text{if } 0.5\,x(k-1) + 3.8\,u(k) \leqslant 2 \\ 0.2\,u(k) + 1 & \text{if } 0.5\,x(k-1) + 3.8\,u(k) > 2 \end{cases}$$

Equivalent MMPS model:

$$y(k) = x(k) = \min(0.5\,x(k-1) + 4\,u(k) - 1,\ 0.2\,u(k) + 1)$$

Constraints:

$$-0.2 \leqslant \Delta u(k) \leqslant 0.2 \quad \text{and} \quad u(k) \geqslant 0 \quad \text{for all } k$$

Let $N_c = N_p = 2$ and $J(k) = J_{\text{out},\infty}(k) + \lambda J_{\text{in},1}(k)$

$$= \|\tilde{y}(k) - \tilde{r}(k)\|_\infty + \lambda \|\tilde{u}(k)\|_1$$

Reformulation as LPs

After substitution:

$$J(k) = \max\big(\min(t_1, t_2), s_1, s_2, \min(t_3, t_4, t_5), s_3, s_4, s_5\big)$$

with t_i, s_i affine functions of $x_1(k-1), u(k), u(k+1), r(k)$

Min-max canonical form:

$$\begin{aligned} J(k) = \min\big(&\max(t_1, t_3, s_1, s_2, s_3, s_4, s_5), \max(t_1, t_4, s_1, s_2, s_3, s_4, s_5), \\ &\max(t_1, t_5, s_1, s_2, s_3, s_4, s_5), \max(t_2, t_3, s_1, s_2, s_3, s_4, s_5), \\ &\max(t_2, t_4, s_1, s_2, s_3, s_4, s_5), \max(t_2, t_5, s_1, s_2, s_3, s_4, s_5)\big) \end{aligned}$$

$\rightarrow$ solve 6 LPs

Timing results

CPU time for closed-loop MPC over period $[1, 15]$:

Method	CPU time (s)
LP	0.55
SQP	4.90
MLD	2.74
ELCP	198.82

Outline

Game-theoretic approaches

- Safety-critical applications such as collision avoidance in free flight or automated highways
  - → guarantee safety even in case intentions of other aircraft/vehicle are not known (non-cooperative game)
    if (partial) communication possible → cooperative game
- Consider continuous-time system

$$\dot{x} = f(x, u, d)$$

with u control inputs (corresponding to 1st player), and d disturbance inputs (corresponding to 2nd player/adversary)

- Assume safety constraints can be represented by set

$$F = \{x \in X \mid S(x) \geq 0\}$$

Game-theoretic approach

- Let $t_0 \leq t_{\text{end}}$ and consider cost function

$$J : X \times \mathcal{U} \times \mathcal{D} \times [t_0, t_{\text{end}}] \to \mathbb{R} : (x, u(\cdot), d(\cdot), t) \mapsto S(x(t_{\text{end}}))$$

  where $\mathcal{U}$ and $\mathcal{D}$ denote admissible control and disturbance functions
- Cost is function of final state $x(t_{\text{end}})$ only!

$\to$ J is cost associated with trajectory starting at x at time $t \in [t_0, t_{\text{end}}]$ with inputs $u(\cdot)$ and $d(\cdot)$, and ending at the final state $x(t_{\text{end}})$

- Define value function

$$J^\star(x, t) = \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} J(x, u, d, t)$$

Game-theoretic approach (cont.)

- The set

$$\{x \in X \mid \min_{\tau \in [t, t_{\text{end}}]} J^\star(x, \tau) \geq 0\}$$

  contains all states for which system can be forced by control u to remain in safe set F for at least $|t_{\text{end}} - t|$ time units, irrespective of disturbance function d
- Value function $J^\star$ can be computed using Hamilton-Jacobi equations
  - (numerical) solution of Hamilton-Jacobi equations is tremendous task
  + approach provides systematic way to check safety properties for continuous-time systems and certain classes of hybrid systems

Outline

Summary

- Optimal control of hybrid systems
- MPC for MLD and PWA systems
- MPC for MMPS and continuous PWA systems
- Game-theoretic approaches