Modeling & Control of Hybrid Systems

Chapter 6 – Optimization-Based Control

Overview

1. Optimal control of hybrid systems
2. MPC for MLD and PWA systems
3. MPC for MMPS and continuous PWA systems
4. Game-theoretic approaches

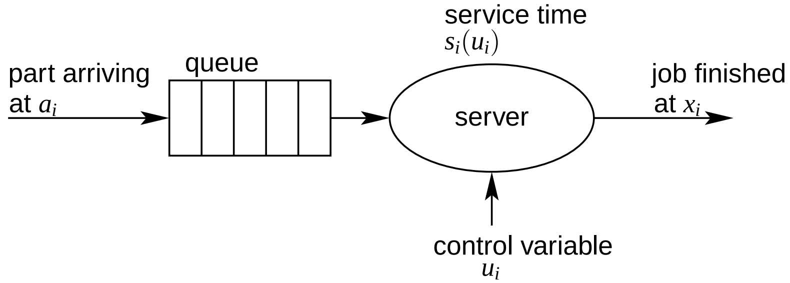
1. Optimal control of a class of hybrid systems

1. Optimal control for hybrid manufacturing systems
2. Example
3. Optimality conditions

1.1 Optimal control for 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

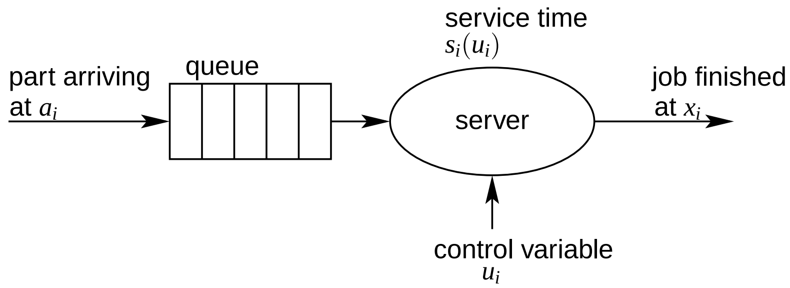
  assume higher quality $\rightarrow$ longer processing times



- 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



- 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

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

- *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 for hybrid manufacturing systems (cont.)

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

1.2 Example

- Steel heating/annealing manufacturing processes
- Involves slowly heating and cooling strips to some desired temperatures
- Higher level controller determines furnace reference temperature + amount of time strip is held in furnace
- Physical state z_i represents temperature and depends on *line speed* u_i and *furnace reference temperature* F_i:

$$\dot{z}_i(t) = -\frac{F_i - z_i(t_0)}{L} u_i + K_s\big(F_i^4 - z_i^4(t)\big) \qquad \text{for } t \geqslant t_0$$

- Constraint: $u_{\min} \leqslant u_i \leqslant u_{\max}$

1.2 Example (continued)

- Temporal state:
  x_i: time when job starts processing at furnace
  y_i: time when job completes processing

$$x_i = \max(a_i, x_{i-1}) + s_1(u_i) \quad \text{and} \quad y_i = x_i + s_2(u_i)$$

  with $s_1(u_i)$ elapsed time for whole body of strip to enter furnace (is dependent on length of strip),
  and $s_2(u_i)$ processing time for each point of strip to run through furnace (is dependent on length of furnace)
- Two control objectives:
  1. reduce temperature errors w.r.t. furnace reference temperature
  2. reduce entire processing time

1.2 Example (continued)

- Thus, optimal control problem is

$$\min_{u_1,\ldots,u_N} J = \sum_{i=1}^{N} \left(\theta(u_i) + \phi(y_i)\right)$$

subject to physical and temporal evolution equations

with

- $\phi(y_i)$ cost related to jobs departing at time y_i
  e.g., $\phi(y_i) = (y_i - d_i)^2$, with d_i due date
  $\rightarrow$ penalizes tardiness, and early completion (inventory cost)
- $\theta(u_i)$ penalizes deviation from reference temperature F_i:

$$\theta(u_i) = |F_i - z_i(L/u_i)|^2 + \beta \int_0^{L/u_i} (F_i - z_i(t))^2 dt$$

where L/u_i is time each point of strip stays in furnace

1.3 Optimality conditions

- Define augmented cost:

$$\bar{J}(x,\lambda,u) = \sum_{i=1}^{N} \left(L_i(x_i,u_i) + \lambda_i(\max(x_{i-1},a_i) + s_i(u_i) - x_i)\right)$$

where λ is co-state

- Assumption: costs L_i and s_i are continuously differentiable
- Ignoring non-differentiabilities associated with max, standard first-order necessary conditions for optimality require

$$\frac{\partial \bar{J}}{\partial u_i} = 0, \quad \frac{\partial \bar{J}}{\partial \lambda_i} = 0, \quad \frac{\partial \bar{J}}{\partial x_i} = 0 \quad \text{for } i = 1,\ldots,N\,.$$

1.3 Optimality conditions (continued)

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

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

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

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

How to deal with non-differentiability (continued)

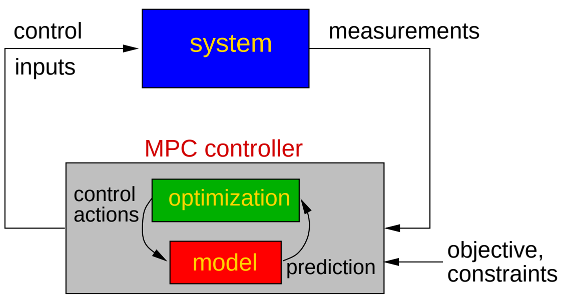
- Properties of generalized gradient:
  - if f is continuously differentiable in some open set containing u, then $\partial f(u) = \{\nabla f(u)\}$
  - if u is local minimum, then $0 \in \partial f(u)$
    $\longrightarrow$ this becomes first-order optimality condition in non-smooth optimization
- See lecture notes for computation of $\partial \bar{J}$
- Note: presence of idle period results in decoupling

2. MPC for MLD systems

1. Model predictive control (MPC)
2. MPC for MLD and PWA systems

2.1 Model predictive control (MPC)

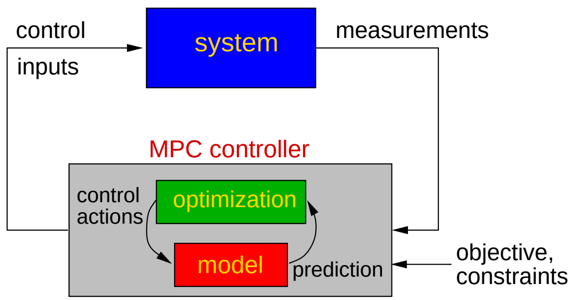
- very popular in process industry
- model-based
- easy to tune
- multi-input multi-output (MIMO)
- allows constraints on inputs and outputs
- adaptive / receding horizon
- uses on-line optimization



$\longrightarrow$ apply to MLD, PWA and MMPS systems while keeping advantages

MPC (continued)

At sample step k:



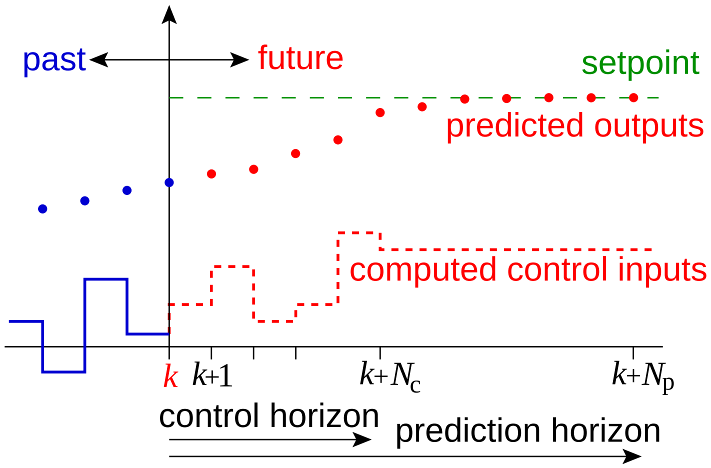
- Use model to predict system output over prediction period $[k, k+N_\mathrm{p}]$ for given input sequence $u(k), \ldots, u(k+N_\mathrm{p}-1)$

  N_p: prediction horizon

  $\tilde{u}(k) = [\, u^\mathsf{T}(k) \ \ldots \ u^\mathsf{T}(k+N_\mathrm{p}-1) \,]^\mathsf{T}$
- Define performance criterion $J(k)$ over $[k, k+N_\mathrm{p}]$, e.g.,
  $J(k) =$ tracking error $+ \lambda$input effort/energy
- Constraints on u, x, y

MPC problem

- Find at sample step k input sequence $\tilde{u}(k)$ that minimizes $J(k)$ subject to system equations + constraints



MPC problem (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

2.2 MPC for MLD systems

- Consider MLD system:

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

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

2.2 MPC for MLD systems (continued)

- Stabilize system to equilibrium state:

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

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

- End-point condition: $\hat{x}(k+N_p|k) = x_{eq}$
- Control horizon constraint:
  $u(k+j) = u(k+N_c-1)$ for $j = N_c, \ldots, N_p - 1$

2.2 MPC for MLD systems (continued)

- **Property**:
  If feasible solution exists for $x(0)$, then MPC input stabilizes system, i.e.,

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

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

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 $\tilde{V}(k) = \left[\tilde{u}^{\mathsf{T}}(k) \;\; \tilde{\delta}^{\mathsf{T}}(k) \;\; \tilde{z}^{\mathsf{T}}(k)\right]^{\mathsf{T}}$
  $\rightarrow$ contains both real-valued and integer-valued components
- Results in

$$\min_{\tilde{V}(k)} \tilde{V}^{\mathsf{T}}(k) S_1 \tilde{V}(k) + 2(S_2 + x^{\mathsf{T}}(k) S_3) \tilde{V}(k) \tag{1}$$

$$\text{subject to } F_1 \tilde{V}(k) \leqslant F_2 + F_3 x(k) \;\;, \tag{2}$$

= MIQP problem

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)
- Software:
  - Multi-Parametric Toolbox (MPT) : http://control.ee.ethz.ch/~mpt/
  - Hybrid toolbox : http://www.dii.unisi.it/hybrid/toolbox/
  - TOMLAB, CPLEX, Xpress
  - NAG, Matlab NAG Toolbox

3. MPC for continuous PWA systems

1. Equivalence of continuous PWA and MMPS systems
2. Canonical forms of MMPS functions
3. Model predictive control for MMPS systems
4. Algorithms for MMPS-MPC
5. Example

3.1 Equivalence of continuous PWA and MMPS systems

PWA systems

- Continuous PWA function $f:\mathbb{R}^n \to \mathbb{R}$:
  - domain space divided into polyhedral regions $R_{(1)},\dots,R_{(N)}$
  - 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:

$$\begin{aligned} x(k) &= \mathscr{P}_x(x(k-1),u(k)) \\ y(k) &= \mathscr{P}_y(x(k),u(k)) \end{aligned}$$

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

PWA systems (cont.)

- Note: continuous PWA model can be used as approximation of general nonlinear continuous state space model

$$\begin{aligned} x(k) &= \mathscr{N}_x(x(k-1),u(k)) \\ y(k) &= \mathscr{N}_y(x(k),u(k)) \end{aligned}$$

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
- Examples:
  - $5x_1 - \max(x_2 + x_3, 5x_1 - 2x_2)$
  - $\max\big(x_1, \min(x_2, x_3)\big) + \max\big(x_2 - 8x_3 + \min(x_1, 5x_2), -7x_1$
- Note: MMPS function is continuous
- MMPS system:

$$\begin{aligned} x(k) &= \mathscr{M}_x(x(k-1),u(k)) \\ y(k) &= \mathscr{M}_y(x(k),u(k)) \end{aligned}$$

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

Equivalence of continuous PWA and MMPS systems

- Previous result: (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

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

$\rightarrow$ f is also MMPS function
- So classes of continuous PWA functions and MMPS functions coincide

Equivalence of continuous PWA and MMPS systems (cont.)

- Continuous PWA systems and MMPS systems are equivalent:
  - $\longrightarrow$ for given continuous PWA model there exists MMPS model (and vice versa) such that input-output behaviors coincide
  - $\Rightarrow$ use properties & techniques from PWA systems for MMPS systems and vice versa

3.2 Canonical forms of MMPS functions

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

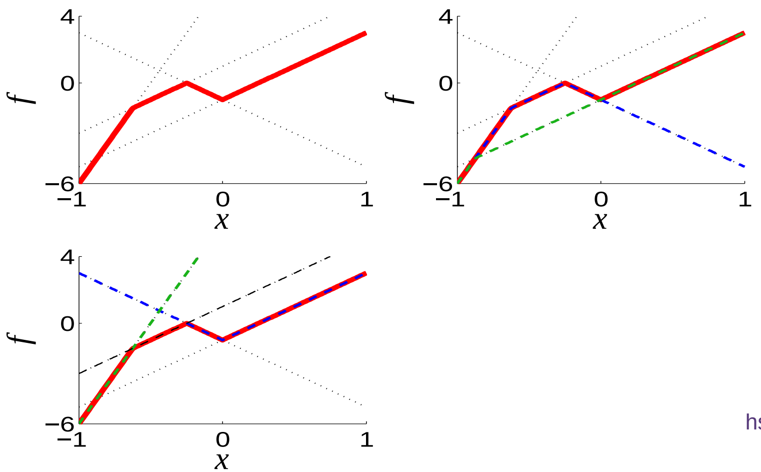
$$f = \min_i \max_j (\alpha_{(i,j)}^T x + \beta_{(i,j)})$$

or into max-min canonical form

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

Example

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



3.3 MPC for MMPS systems

- Use MMPS model

$$\begin{aligned} x(k) &= \mathscr{M}_x(x(k-1), u(k)) \\ y(k) &= \mathscr{M}_y(x(k), u(k)) \end{aligned}$$

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

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

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

$\longrightarrow$ F_j is MMPS function!

3.3 MPC for MMPS systems (continued)

- 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:

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

with

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

Note: $|x| = \max(x, -x) \to$ cost functions are MMPS functions

3.3 MPC for MMPS systems (continued)

- Constraints on input and output signals:

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

3.4 Algorithms for MMPS-MPC

- Nonlinear optimization (SQP, ELCP):

$\to$ local minima, excessive computation time

- MPC for mixed logical-dynamical (MLD) systems [Bemporad, Morari]:

$\to$ mixed real-integer quadratic programming problems

- New approach based on canonical forms:

$\to$ set 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

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

$$\Rightarrow \min_{\tilde{u}} J(k) = \min_{\tilde{u}} \min_i \left(\max_j (\alpha_{(i,j)}^T \tilde{u} + \beta_{(i,j)}) \right) = \min_i \underbrace{\min_{\tilde{u}} \left(\max_j (\alpha_{(i,j)}^T \tilde{u} + \beta_{(i,j)}) \right)}_{\to \text{ LP!}}$$

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!*

3.5 Example

PWA model:

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

Equivalent MMPS model:

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

Constraints:

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

Let $N_\mathrm{c} = N_\mathrm{p} = 2$ and $J(k) = J_{\mathrm{out},\infty}(k) + \lambda J_{\mathrm{in},1}(k)$
$= \|\tilde{y}(k) - \tilde{r}(k)\|_\infty + \lambda \|\tilde{u}(k)\|_1$

3.5 Example (continued)

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

3.5 Example (continued)

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

4. Game-theoretic approaches

- Safety-critical applications such as collision avoidance in free flight or automated highways

$\rightarrow$ guarantee safety even in case intentions of other aircraft/vehicle are not known (non-cooperative game)
if (partial) communication possible $\rightarrow$ cooperative game

- Consider continuous-time system

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

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

- Assume safety constraints can be represented by set

$$F = \{x \in X \mid k(x) \geqslant 0\}$$

Game-theoretic approach

- Let $t \leqslant 0$ and consider cost function

$$J : X \times \mathscr{U} \times \mathscr{D} \times \mathbb{R}^- \rightarrow \mathbb{R} : (x, u(\cdot), d(\cdot), t) \mapsto k(x(0))$$

where $\mathscr{U}$ and $\mathscr{D}$ denote admissible control and disturbance functions

- Cost is function of final state $x(0)$ only!

$\rightarrow$ J is cost associated with trajectory starting at x at time $t \leqslant 0$ with inputs $u(\cdot)$ and $d(\cdot)$, and ending at time $t = 0$ at the final state $x(0)$

- Define value function

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

Game-theoretic approach (cont.)

- The set

$$\{x \in X \mid \min_{t' \in [t, 0]} J^\star(x, t') \geqslant 0\}$$

contains all states for which system can be forced by control u to remain in safe set F for at least $|t|$ 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 formidable task

  + approach provides systematic way to check safety properties for continuous-time systems, and certain classes of hybrid systems