ハイブリッドシステムのモデル予測制御

FL06 - 26 -1

12/4/2006

山田 照樹

アウトライン

- ハイブリッドシステムの紹介
- ハイブリッドシステムの表現
- Multi-Parametric Toolbox (MPT)の紹介
- ハイブリッドシステムのモデル予測制御
- 適用例（車線変更システムの紹介）
- MPTで解くまでの流れ
- おわりに

ハイブリッドシステムとは

ハイブリッドシステム (Hybrid System)

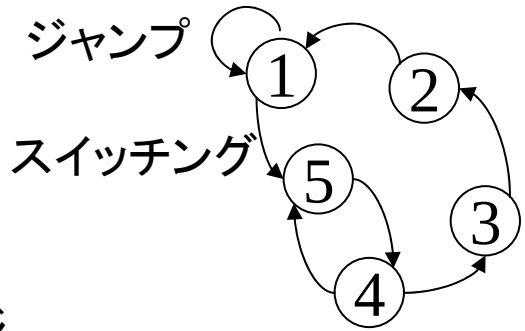
より広い範囲のシステムを扱うための表現

命題

If A Then B

命題が真・偽 ⟺ $\delta = \{0, 1\}$（論理変数）

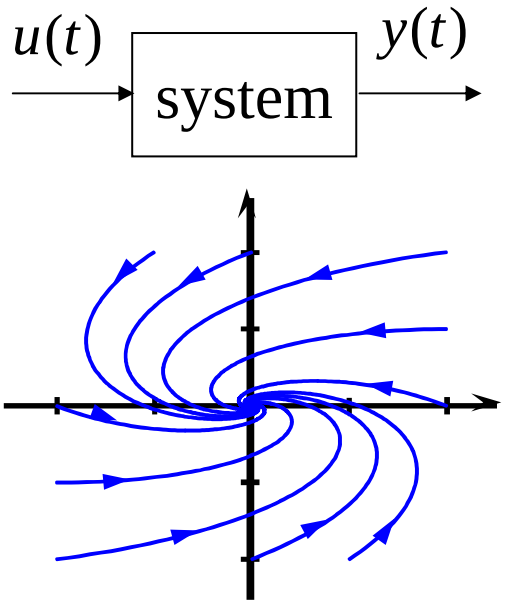
ギアチェンジ



表現

命題論理

オートマトン



スイッチの切り換え動作やif–thenルール（離散事象）

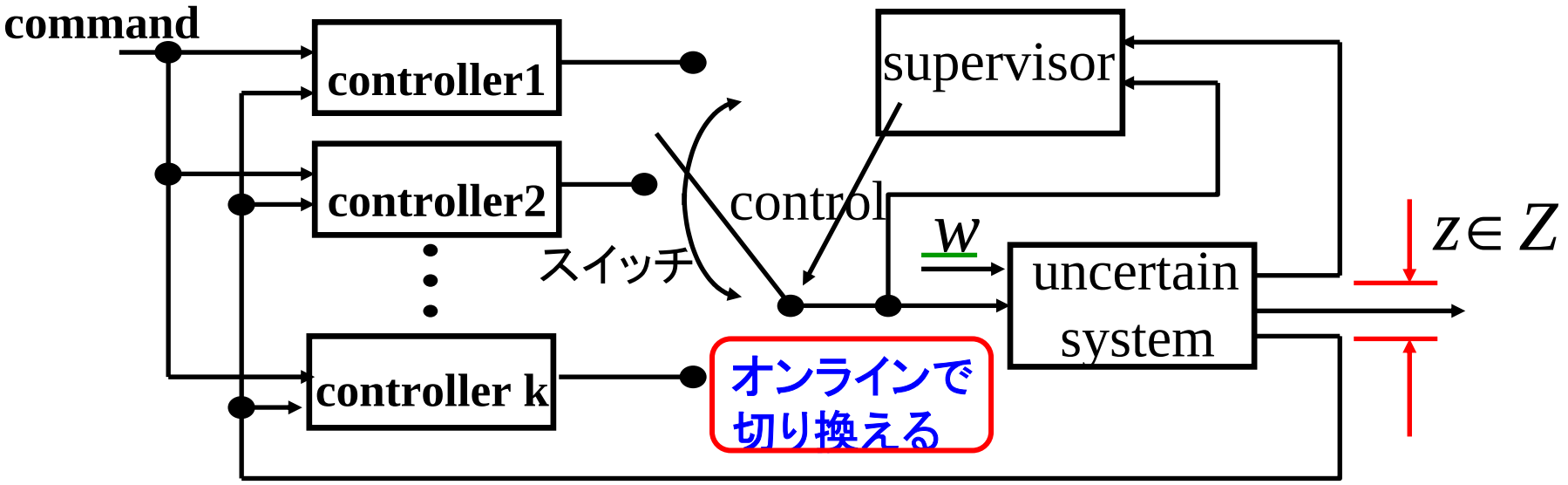
連続ダイナミクス（連続的事象）

離散的事象と連続的事象が相互作用する系

ハイブリッドシステムの例

スイッチング制御系

藤田, 平田, 計測と制御, 1999.



コントローラによって, 閉ループ系のシステムが変わる

$$x^i(t+1) = A^i x^i(t) + B^i w(t)$$

$$z^i(t) = C^i x^i(t) + D^i w(t) \qquad i = 1, \ldots, k$$

Robust control

モデルの不確かさや外乱に対する安定性・性能の保証

外乱に対して拘束条件を破らないコントローラの選定

$$z^i(t) \in Z \quad \forall w(t) \in W \quad \forall t$$

主なハイブリッドシステム表現

混合論理動的システム(MLD)　(Bemporad, Morari 99)

$$x_{t+1} = Ax_t + B_1 u_t + B_2 z_t + B_3 \delta_t \qquad \delta \in \{0,1\}^{m_1}$$

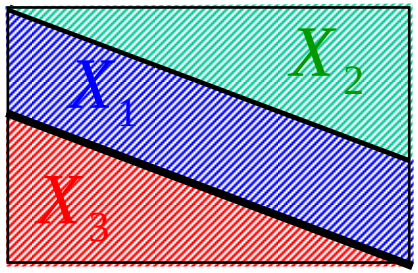
$$y_t = Cx_t + D_1 u_t + D_2 z_t + D_3 \delta_t \qquad z \in \mathcal{R}^{m_2}$$ 補助変数

$$E_1 x_t + E_2 u_t + E_3 z_t + E_4 \delta_t \le g_5$$

混合整数不等式

区分的アファインシステム(PWA)　(Sontag, 98)

$$x_{t+1} = A_i x_t + B_i u_t + f_i$$
$$y_t = C_i x_t + D_i u_t + g_i \quad \text{if} \quad \begin{bmatrix} x_t \\ u_t \end{bmatrix} \in X_i$$



動特性が領域によって切り換わるシステム

いずれも等価なシステム表現として変換できる.

混合整数計画問題

混合整数計画問題 ⟶ MLDシステムに対する最適化問題

ある制約条件下で，評価関数を最小化する問題

混合整数不等式　　モデル予測制御への応用

混合整数2次計画（Mixed Integer Linear programming）

評価関数　$\min_{x,\delta} \sum_{i=k}^{k+N} x^T(i)Qx(i)+v^T(i)Rv(i)$

制約条件　$x(i+1)=Ax(i)+Bv(i)$　$\delta=\{0,1\}$:integer

$Dx(i)+Ev(i)\le F$　$v(i)=[u(i)^T\ z(i)^T\ \delta(i)^T]^T$

混合整数不等式

制約条件の部分で MLD のシステム表現が用いられる.

オフライン計算で最適な経路を生成　CPLEX（ILOG社）

PWA表現とMLD表現の等価変換

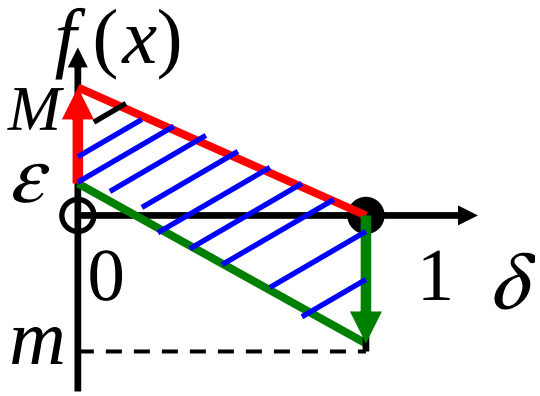
PWAとMLDは等価に変換することができる.

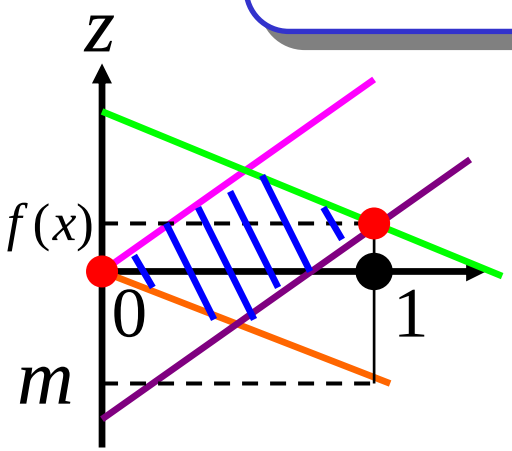
命題論理

$[f(x) \le 0] \leftrightarrow [\delta = 1]$ が真

混合整数不等式

$$f(x) \le M(1-\delta)$$
$$f(x) \ge \varepsilon + (m-\varepsilon)\delta$$





ε:小さい正の数

$f : \mathcal{R}^n \to \mathcal{R}$

$M = \max_{x \in X} f(x),$

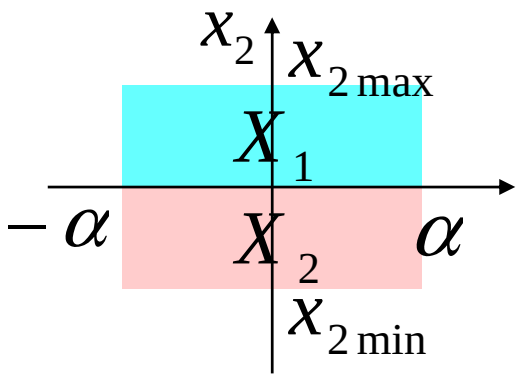
$m = \min_{x \in X} f(x)$

補助変数 $z := \delta f(x)$

$$[\delta = 0] \to [z = 0]$$
$$[\delta = 1] \to [z = f(x)]$$

$$m\delta \le z \le M\delta$$
$$f(x) - M(1-\delta) \le z \le f(x) - m(1-\delta)$$

PWA表現とMLD表現の変換の例



PWAシステム表現

$$x_1 \begin{cases} x_{t+1} = A_1 x_t + B u_t & x_t \in X_1 \\ x_{t+1} = A_2 x_t + B u_t & x_t \in X_2 \end{cases}$$

$$x_{t+1} = \delta_1 A_1 x_t + \delta_2 A_2 x_t + B u_t \quad ①$$

$$\delta_1 + \delta_2 = 1 \quad ②$$

$$X_1 = \{ x \in R^n \mid |x_1| \le \alpha,\ 0 \le x_2 \le x_{2\max} \}$$

$$X_2 = \{ x \in R^n \mid |x_1| \le \alpha,\ x_{2\min} \le x_2 < 0 \}$$

$$x_t \in X_1 \Leftrightarrow \delta_1 = 1$$

$$x_t \in X_2 \Leftrightarrow \delta_2 = 1$$

$[0 \le x_2 \le x_{2\max}] \leftrightarrow [\delta_1 = 1]$が真 $\Leftrightarrow$ $-x_{\min 2}\delta \le x_2 \le x_{\max 2}\delta - (1-\delta)\varepsilon$ ③

$$z_1 = \delta_1 x_1$$
$$z_2 = \delta_2 x_2$$

$\Leftrightarrow$

$$-\alpha\delta \le z_1 \le \alpha\delta \quad ④ \qquad x_1 - \alpha(1-\delta_1) \le z_1 \le x_1 + \alpha(1-\delta_1) \quad ⑤$$

$$x_{2\min}\delta_2 \le z_2 \le x_{2\max}\delta_2 \quad ⑥ \qquad x_2 - x_{2\max}(1-\delta_2) \le z_2 \le x_2 - x_{2\min}(1-\delta_2) \quad ⑦$$

①～⑦をまとめることで，MLD表現に変換できる．

HYSDEL & hys2pwa

ハイブリッドシステム

線形ダイナミクス, オートマトン, If-then-else 規則, 命題論理

HYbrid System DEscription Language (HYSDEL) ver.2.0.5

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

Mixed Logical Dynamical (MLD) システム

$$x_{t+1} = Ax_t + B_1 u_t + B_2 z_t + B_3 \delta_t$$

$$E_1 x_t + E_2 u_t + E_3 z_t + E_4 z_t \leq g_5$$

Piecewise Affine (PWA) Plugin: hys2pwa 1.1.3

hys2pwa http://control.ee.ethz.ch/~hybrid/hysdel/hysdel.msql

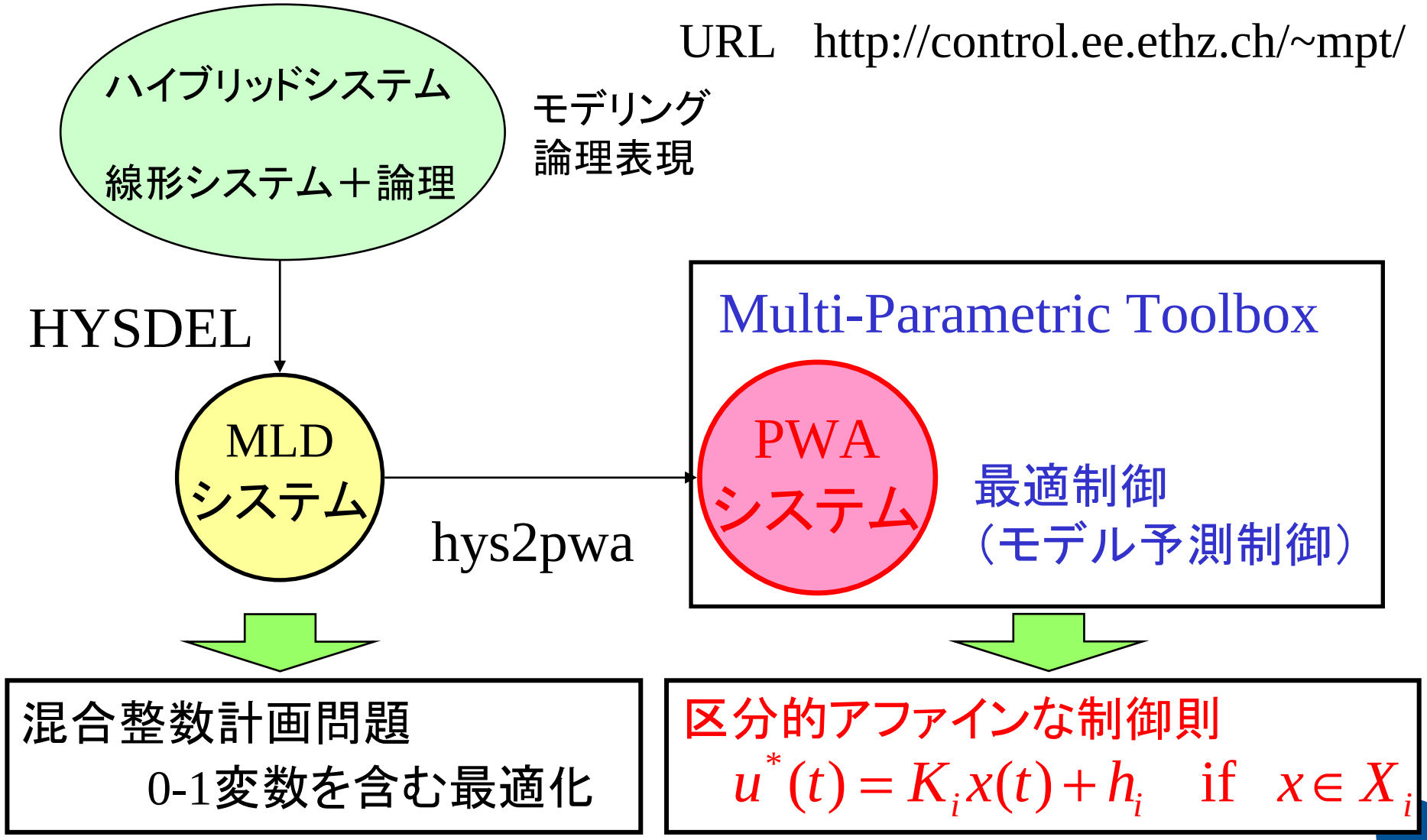
Piecewise Affine (PWA)システム

$$x_{t+1} = A_i x_t + B_i u_t + f_i \qquad \text{if} \quad \begin{bmatrix} x_t \\ u_t \end{bmatrix} \in X_i$$

Multi-Parametric Toolbox

Multi-Parametric Toolbox　スイス連邦工科大 (ETH) IfA

URL　http://control.ee.ethz.ch/~mpt/



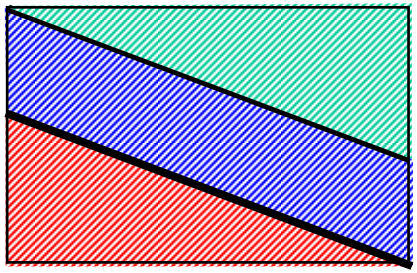
ハイブリッドシステムのモデル予測制御

区分的アファイン（線形）システム

$$x_{t+1} = A_i x_t + B_i u_t + f_i \quad \text{if} \quad \begin{bmatrix} x_t \\ u_t \end{bmatrix} \in X_i, \quad i \in I$$

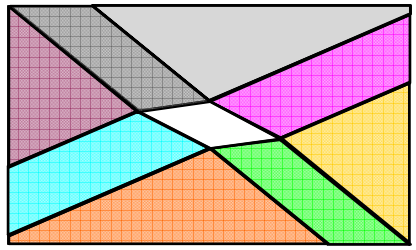
拘束条件

$$\begin{cases} x_k \in X, & \forall k \in \{1, \ldots, N\} \\ u_k \in U, & \forall k \in \{1, \ldots, N-1\} \\ x_N \in X_{set} \end{cases}$$



有限時間最適制御問題（モデル予測制御問題）

$$J_N^*(x(0)) = \min_{u_0, \cdots, u_{N-1}} \sum_{k=0}^{N-1} \left(\|Qx(k)\|_l + \|Ru(k)\|_l \right) + \|Q_f x(N)\|_l$$



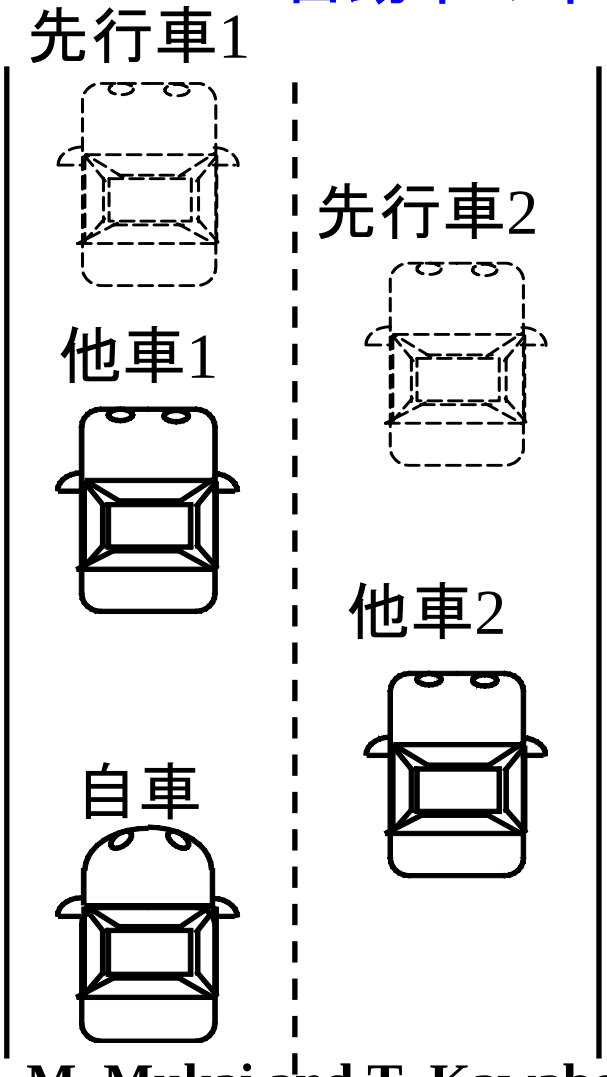
状態 $x(t)$ をパラメータとする陽な形の(Explicit) 制御則

$$u(t) = K_i x(t) + h_i \quad \text{if} \quad x \in X_i$$

A. Bemporad, F. Borrelli and M. Morari, (CDC, 2000)
F. Borrelli, M. Baotic, A. Bemporad and M. Morari (ACC, 2003)
M. Baotic, F.J. Christophersen and M. Morari (ECC, 2003)

ハイブリッドシステムのモデル予測制御の適用例

自動車の車線変更をおこなうシステム



M. Mukai and T. Kawabe
(SICE-ICASE, 2006)

- 先行車との間隔が詰まったら車線変更をする.
- 先行車との間隔に応じて速度を調整する.

周囲の状況によって走行モードを切り換える.

↓

ハイブリッドシステム表現

安全のための交通状況や他車の挙動の予測

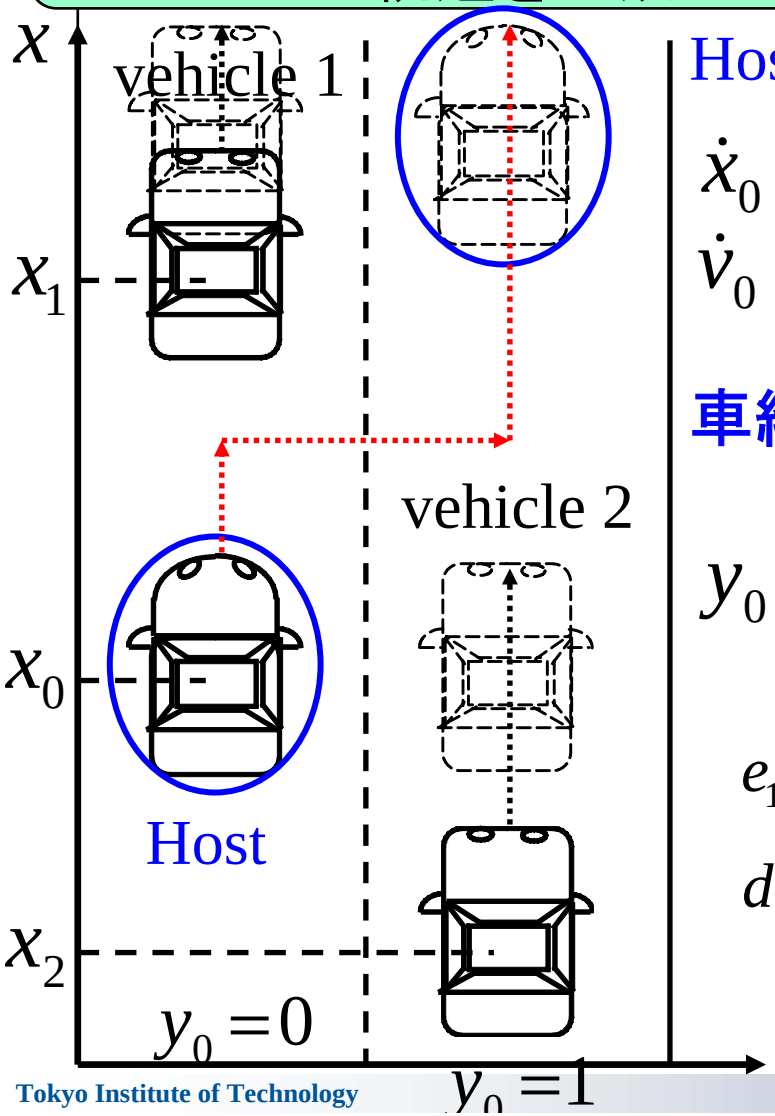
↓

モデル予測による最適な軌道

運転手の安全な運転の支援

車線変更システムの問題の定式化（1）

制御目的：周辺の車両との相対距離をもとに，安全，かつ最適な軌道を生成して先行車を追い抜く．



Host vehicleのシステム（自車のモデル）

$$\dot{x}_0 = v_0$$

$$\dot{v}_0 = u_x^0 \quad \text{(加速度入力)}$$

車線変更の表現

$$y_0 = \begin{cases} 1 & \text{if } [e_1 < d_{\min 1}] \text{ and } [|e_2| < d_{\min 2}] \text{ 右車線} \\ 0 & \text{if others} \quad \text{左車線} \end{cases}$$

e_1, e_2: Host からみた vehicle1,2の相対位置

$d_{\min 1}, d_{\min 2}$:車線変更のためのパラメータ

車線変更システムの問題の定式化(2)

Control objectives for safe and smooth driving of Host vehicle

1. Host vehicle の加速度の大きさを大きくしない.
2. 車線変更は頻繁におこなわない.
3. Host vehicle の目標速度をできるだけ維持.
4. Host vehicle はできるだけ他の車両と衝突が起こらない
   ないような位置にいる.

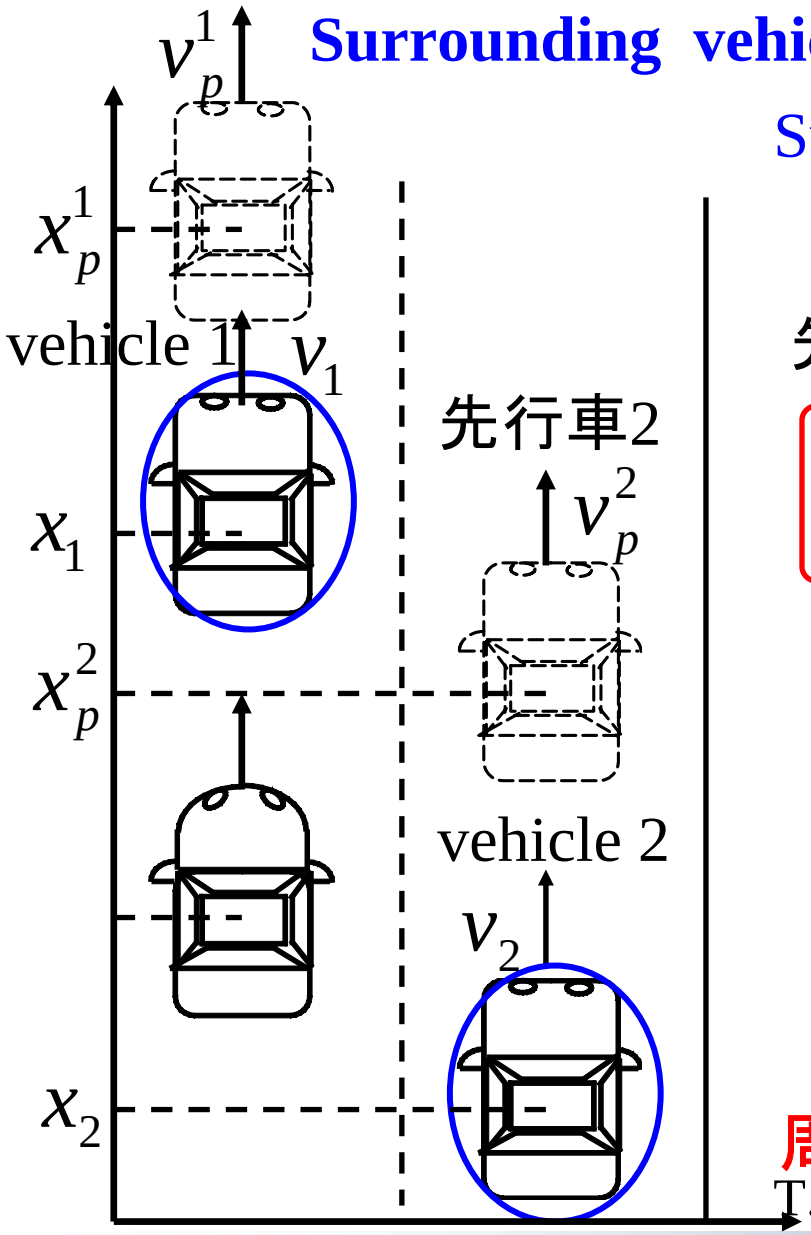
Performance index（モデル予測制御の評価関数）

$$J=\sum_{t=0}^{N-1}(\underbrace{w_u\,|u_x|}_{1}+\underbrace{w_y\,|y_0|}_{2}+\underbrace{w_v\,|v_0-v_0^d|}_{3})$$

v_0^d : Host vehicle の目標速度

N : horizon , w_u, w_y, w_v :入力, レーンチェンジ, 速度の重み

車線変更システムの問題の定式化(3)



Surrounding vehicle model (Host 以外のvehicle)

Surrounding vehicle iのシステム

$$\dot{x}_i = v_i \quad \dot{v}_i = z^i$$

先行車と vehicle i の関係を表わす変数

$$e_r^i = x_p^i - x_i - h_i v_i$$ 望ましい車間距離までの偏差

$$e_{rv}^i = v_p^i - v_i$$ 相対速度

$$e_v^i = v_i^d - v_i$$ vehicle i の目標速度偏差

z^i (加速度)の特性 (h_i : 望ましい車頭時間)

$$z^i = \begin{cases} k_v (v_i^d - v_i) & \text{if } e_r^i > e_{\min}^i \\ k_1 (x_p^i - x_i - h_i v_i) + k_2 (v_p^i - v_i) & \end{cases}$$

周囲の状況に応じて走行モードを変える.

T. Kawabe, H. Nishira and T. Ohtsuka, CCA, 2004.

モデル予測制御を解くまでの流れ（1）

HYSDELのファイルに記述する内容

状態変数

$$x_t = [x_0, x_i, v_0, v_i]^T$$

入力

u_x　Host の加速度

出力

$$y_t = [x_0, y_0]^T$$

システムの記述

$$\dot{x}_0 = v_0$$
$$\dot{x}_i = v_i$$
$$\dot{v}_0 = u_x^0$$
$$\dot{v}_i = z^i$$

If-then-else 規則, 命題論理

If-thenルールできまる補助変数 z^i

If-thenルールできまるレーン番号 y_0

y_0: 0-1変数

HYSDEL コンパイラ

混合論理動的システムへの変換

$$x_{t+1} = Ax_t + B_1 u_t + B_2 z_t + B_3 \delta_t$$
$$E_1 x_t + E_2 u_t + E_3 z_t + E_4 z_t \le g_5$$

モデル予測制御を解くまでの流れ(2)


```
/* host vehicle1 先行車(等速)*/
SYSTEM LaneChange{
      INTERFACE  {
/* Description of variables and
      STATE {
        REAL x0 [0,1000];
        REAL x1 [-1000,1000];
        REAL x2 [0,1000];
        REAL v0 [0,60];
        REAL v1 [0,40];
        }
    INPUT {
        REAL u1;
    }
    OUTPUT {
        REAL y1 ;
        BOOL y2 ;
    }
    PARAMETER {
        REAL dmin1=100;
        REAL emin=120;
        REAL h0=3;
        REAL k1=0.05;
        REAL k2=0.2;
        REAL kv=0.5;
        REAL vd1=30;
        REAL dt=0.1;
    }
}
```

```
IMPLEMENTATION {
    AUX {
        REAL z0;
        BOOL d0, d1;
    }
    AD {
        /* PWA Domain */
        d0 = emin-(x2-x1-h0*v1) <= 0;        /*ダミービークルの加速度切換え条件*/
        d1 = x1-x0-dmin1 <= 0;               /*レーンチェンジの切換え条件*/
        }
    DA {
        z0 = {IF d0 THEN kv*(vd1-v1)
                    ELSE k1*(x2-x1-h0*v1)+k2*(20-v1)};  /*ダミービークルの位置x2,等速20*/
        }
    CONTINUOUS {
        x0=x0+dt*v0;
        x1=x1+dt*v1;
        x2=x2+dt*20;
        v0=v0+dt*u1;
        v1=v1+dt*z0;
                }
    OUTPUT {
        y1 = x0 ;
        y2 = d1 ;
    }
    MUST {
        u1 <= 60;
        -u1 <= 60;
        z0 <= 60;
        -z0 <=60;
```

HYSDELファイルの記述

状態変数，入力，出力

補助変数と{0,1}変数

If-then ルールの記述

状態方程式の記述

拘束条件の記述

モデル予測制御を解くまでの流れ（3）

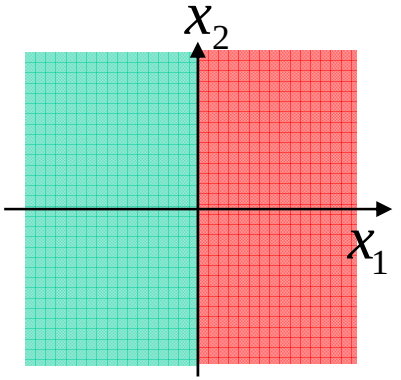
Matlabのコマンド

sysStruct = mpt('~. Hys', Ts) → MLDシステム表現 / PWAシステム表現

HYSDELを使わないでも，システムが既知であるならば，
直接PWAシステムを記述できる．

PWAシステム

$$x(t+1)=\begin{cases}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}x(t)+\begin{bmatrix}0\\ 1\end{bmatrix}u(t) & \text{if } \begin{bmatrix}1 & 0\end{bmatrix}x(t)\le 0\\ \begin{bmatrix}0.5 & 0.2\\ 0 & 1\end{bmatrix}x(t)+\begin{bmatrix}0\\ 1\end{bmatrix}u(t)+\begin{bmatrix}0.5\\ 0\end{bmatrix} & \text{if } \begin{bmatrix}-1 & 0\end{bmatrix}x(t)\le 0\end{cases}$$



$-1 \le u(t) \le 1$

```
sysStruct.umin=-1;
sysStruct.umax=1;
```

```
sysStruct.A{1}=eye(2);  sysStruct.A{2}=[0.5 0.2;0 1];
sysStruct.B{1}=[0;1];   sysStruct.B{2}=[0;1];
sysStruct.f{1}=[0;0];   sysStruct.f{2}=[0.5;0];
```

```
sysStruct.guardX{1}=[1 0];
sysStruct.guardC{1}=[0];
```

モデル予測制御を解くまでの流れ（4）

PWAシステムに対するモデル予測制御の評価関数の設定

$$J_N^*(x(0)) = \min_{u_0,\cdots,u_{N-1}} \sum_{k=0}^{N-1} \left(\|Qx(k)\|_l + \|Ru(k)\|_l \right) + \|Q_f x(N)\|_l$$

stage cost　　terminal cost

Matlabコマンド

probStruct.N Horizon　　**probStruct.Q** weights on the states

probStruct.R weight on the input　　**probStruct.PN** weights on the terminal state

probStruct.norm norm（1, 2, ∞）

probStruct.subopt_lev

0：cost-optimal solution (1, ∞ノルムのとき)

1：Time-optimal solution

2：low-complexity solution (one-step controller)

probStruct.Tconstraint

0：no terminal constraint

1：use LQR terminal set

2：user-provided terminal set

モデル予測制御を解くまでの流れ（5）

モデル予測制御を解くコマンド

ctrlStruct = mpt_control(sysStruct, probStruct)

Explicit な形の
コントローラ

システムの構造体　**評価関数の構造体**

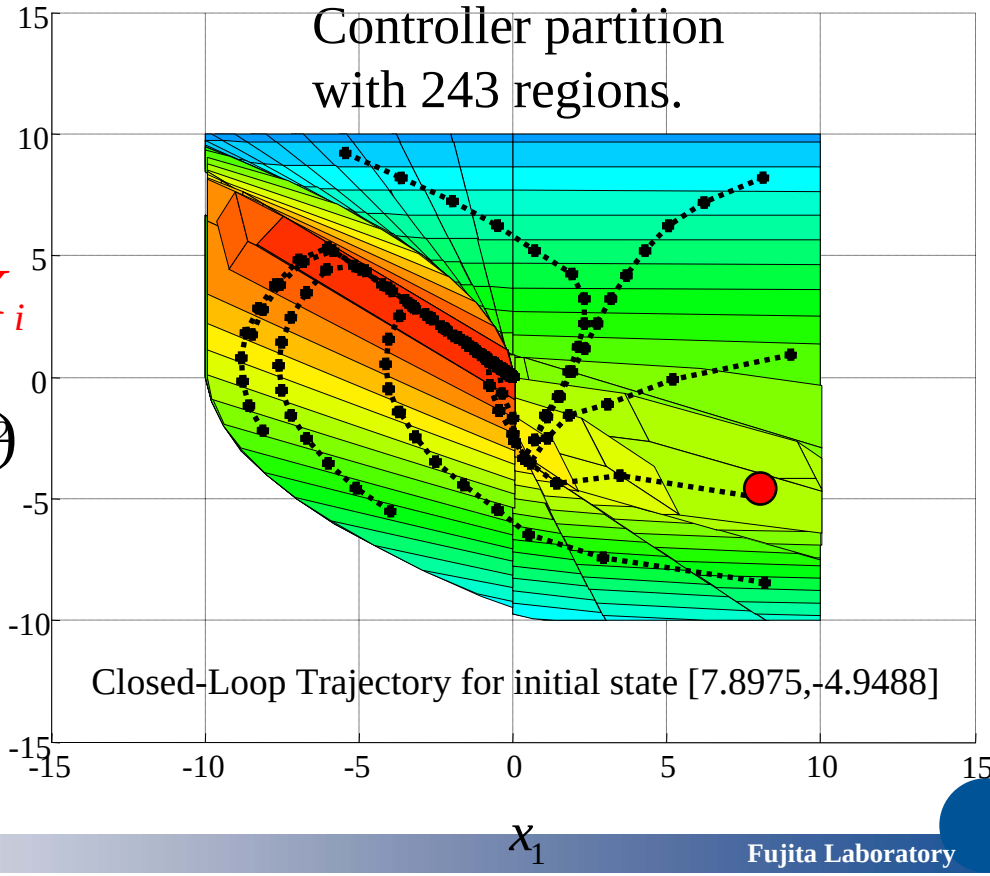
mpt_plotPartition(ctrlStruct)

コントローラの領域の分割

$$u(t) = K_i x(t) + h_i \quad \text{if} \quad x \in X_i$$

mpt_plotTrajectory(ctrlStruct)

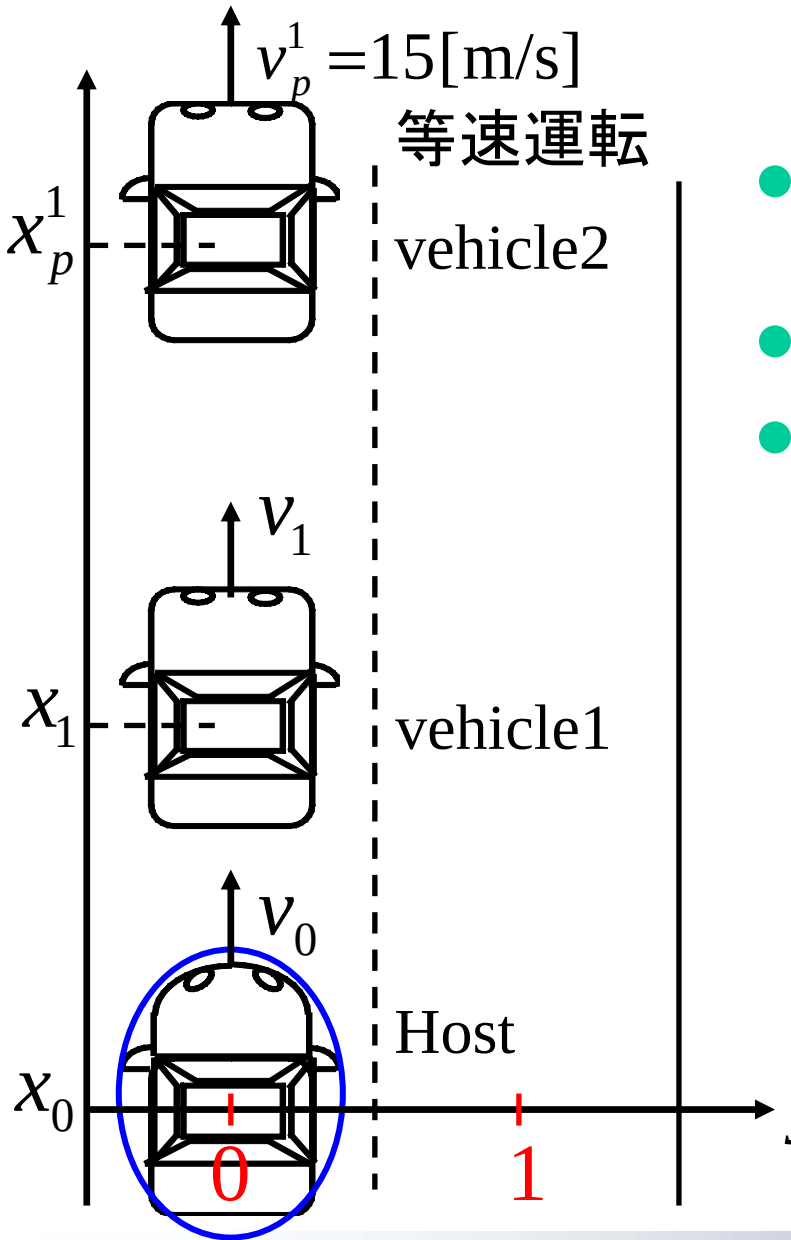
初期状態を指定して
閉ループ系の解軌道を描く



モデル予測制御を解くまでの流れ(6)


```
clear
close all

% HYSDELからPWAシステムの式を求める(シミュレーションのサンプリング100ms)
sysStruct = mpt_sys('MukaiLanchange.hys',0.1);

sysStruct.umax=5;
sysStruct.umin=-5;

sysStruct.ymax=[700 1];
sysStruct.ymin=[0 0];

probStruct.norm=1;
probStruct.N=3;
probStruct.subopt_lev=0;
probStruct.R=0.00001;
probStruct.Qy=10*[0 1];
probStruct.Qyref=10*[0 0];
probStruct.Q=[0 0 0 1 0];
probStruct.xref=30*[0 0 0 1 0];                %ビークル0の目標速度20m/s%

ctrlStruct=mpt_control(sysStruct, probStruct) %オフラインでexplicit controllerを求める%
region=mpt_plotPartition(ctrlStruct)           %controllerの領域の計算%

%開ループ系の軌道計算%
[X,U,Y,D,cost,trajectory,feasible]=mpt_computeTrajectory(ctrlStruct,[0;60;180;25;21],200);

```

車線変更のシミュレーション（1）



シミュレーションの目的

- ハイブリッドシステムで表現された車線変更のシステムの理解
- 安全な車線変更，軌道生成の実現
- MPTの使い方に慣れる.

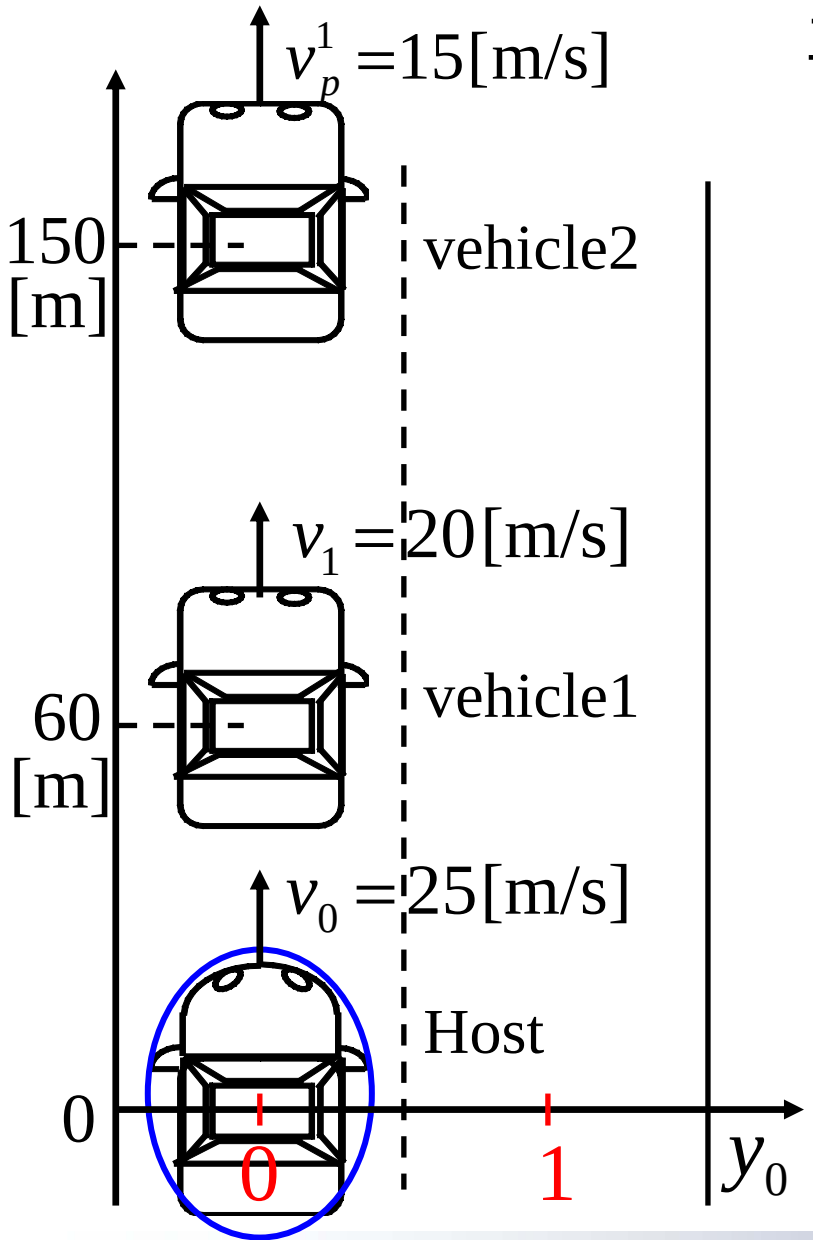
シミュレーションの状況設定

$$x=[x_0, x_1, x_p^1, v_0, v_1]^T$$

$$\begin{cases} \dot{x}_0 = v_0 \\ \dot{x}_1 = v_1 \\ \dot{x}_p^1 = v_p^1 \\ \dot{v}_0 = u_x^0 \\ \dot{v}_1 = z^1 \end{cases} \qquad y_0 = \begin{cases} 1 \text{ if } [e_1 < d_{\min 1}] \\ 0 \text{ if others} \end{cases}$$

$$z^1 = \begin{cases} k_v (v_1^d - v_1) & \text{if } e_r^1 > e_{\min}^1 \\ k_1 (x_p^1 - x_1 - h_1 v_1) + k_2 (v_p^1 - v_1) \end{cases}$$

車線変更のシミュレーション（2）



車線変更のための閾値　$d_{\min 1} = 35[\mathrm{m}]$

vehicle1の走行モードのための閾値　$e_{\min}^1 = 20[\mathrm{m}]$

望ましい車頭時間　$h_1 = 3[\mathrm{s}]$

vehicle1の目標速度　$v_1^d = 20[\mathrm{m/s}]$

Host, vehicle1の加速度制約

$|u_x^0| \le 5 \quad |z^1| \le 5$

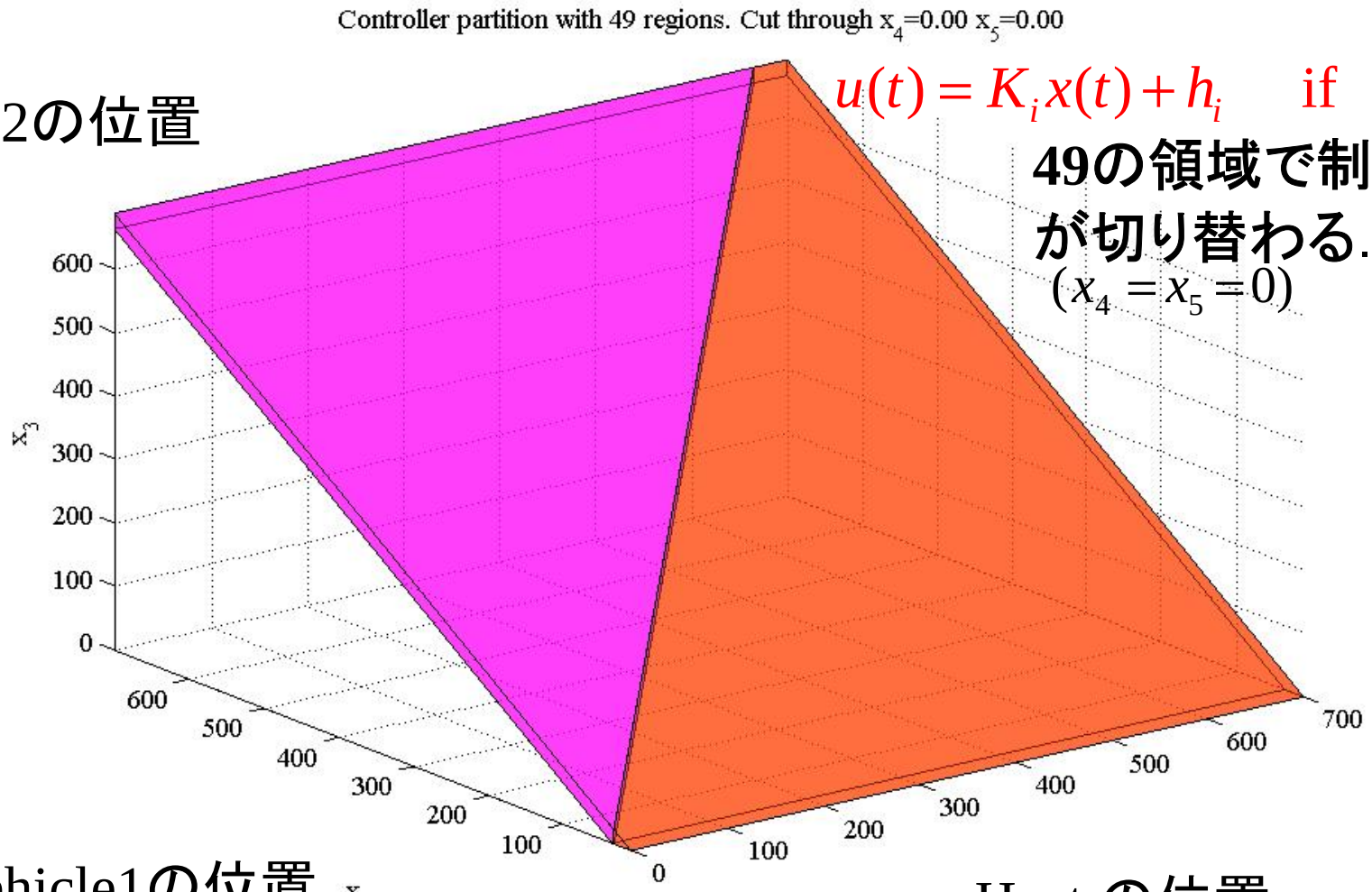
評価関数

$$J = \sum_{t=0}^{N-1} (w_u |u_x| + w_y |y_0| + w_v |v_0 - v_0^d|)$$

$w_u = 10^{-5} \quad w_y = 10 \quad w_v = 1 \quad N = 3$

車線変更のシミュレーション（3）

mpt_plotPartition(ctrlStruct) の実行結果



$$u(t) = K_i x(t) + h_i \quad \text{if} \quad x \in X_i$$

49の領域で制御則が切り替わる.

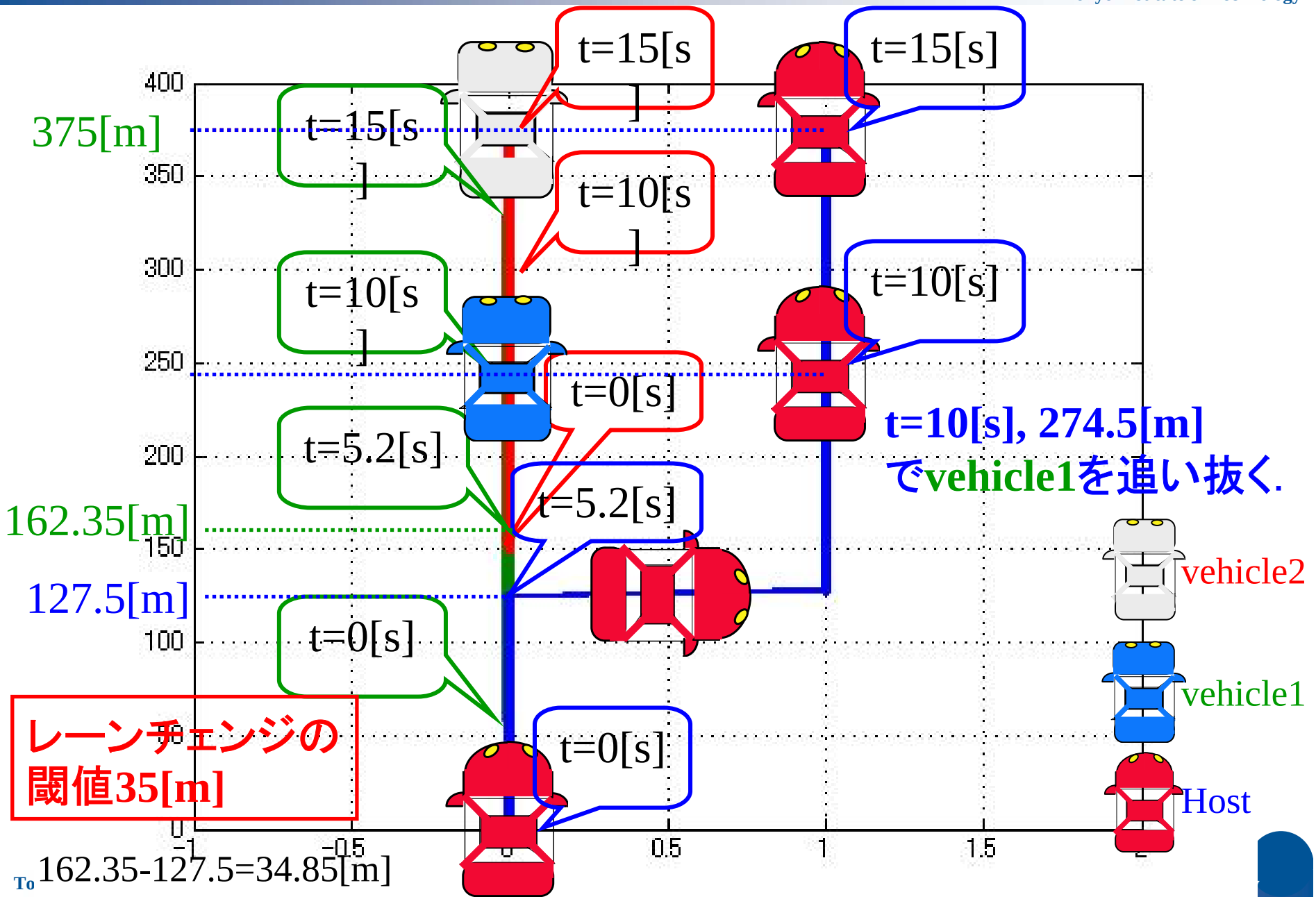
$(x_4 = x_5 = 0)$

vehicle2の位置

vehicle1の位置

Host の位置

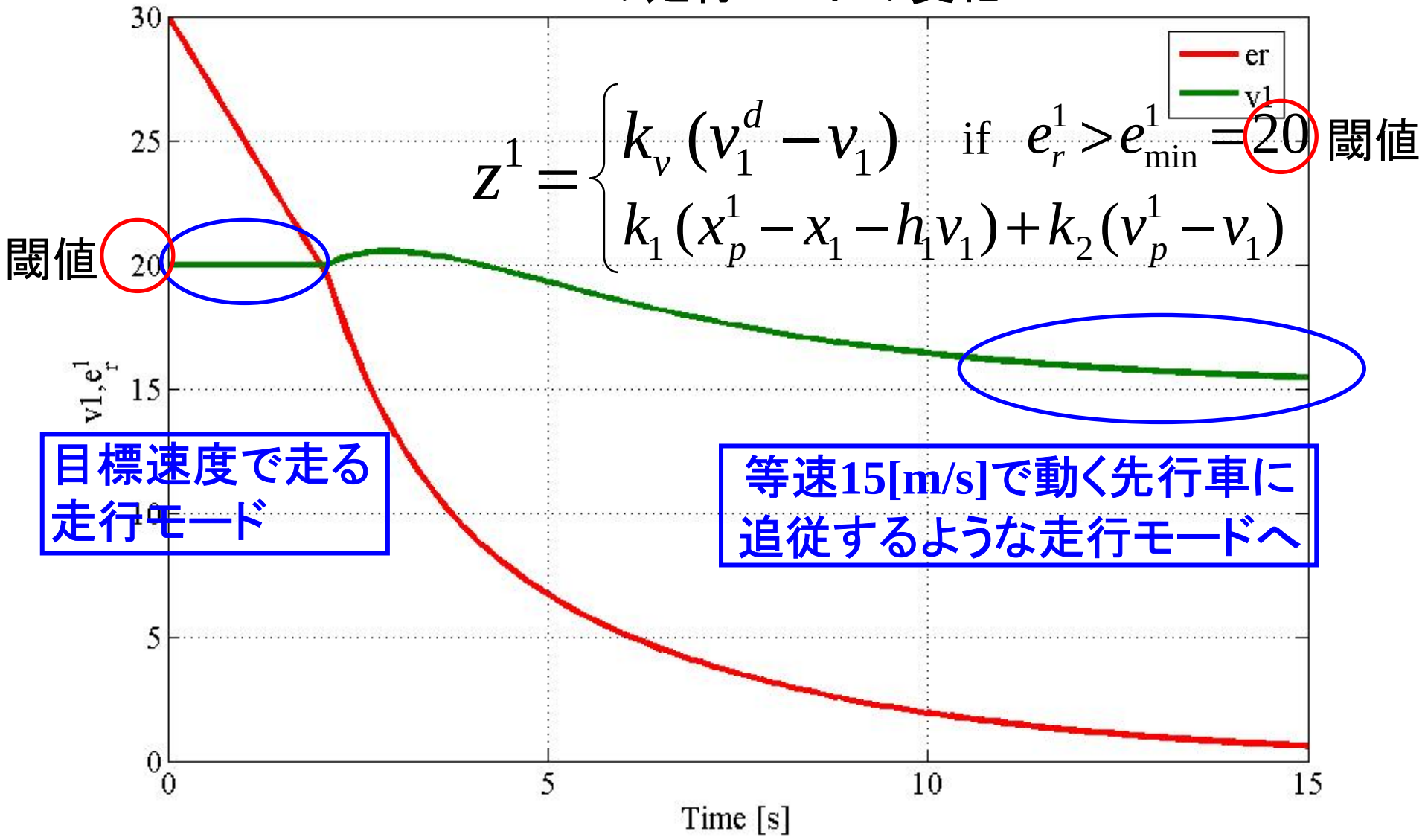
車線変更のシミュレーション（4）



To 162.35-127.5=34.85[m]

車線変更のシミュレーション（5）

vehicle1の走行モードの変化



シミュレーションのまとめ

HYSDELで線形システム，拘束条件，If－thenルールを記述

sysStruct = mpt('~. Hys', Ts)

車線変更
走行モード
の切り替え規則

PWAシステムの構造体(sysStruct), 評価関数の構造体(probStruct)

ctrlStruct = mpt_control(sysStruct, probStruct)

状態量$x(t)$をパラメータとした陽な形のコントローラが得られる

$$u(t)=K_i x(t)+h_i \quad \text{if} \quad x\in X_i$$

ハイブリッドシステム

Host vehicleによる車線変更

周囲のvehicleによる走行モードの切り替え

→ 安全でスムーズな運転モデル

ハイブリッドモデル予測制御

おわりに

本日話した内容

ハイブリッドシステムとその表現

MPTの紹介とその大まかな使い方

MPTをつかったハイブリッドモデル予測制御の適用例

今後の予定

最大出力許容集合を考えることで，
ロバストなモデル予測制御への展開

MPTの応用的な使い方（ロバスト安定など）

スイッチング制御の応用の検討