モデル予測制御方式によるスパース制御の実現

M2017SC003 岩田拓海
指導教員：大石泰章

1　はじめに

スパース制御（または hands-off control）が省エネルギーに役立つものとして研究が進められている [1, 2, 3, 4, 5]．スパースとは零の要素が多いことを意味し，入力がスパースとは制御を行う時間全体に対し入力値が零をとる時間が長いことを意味する．スパースな入力によりアクチュエータの動作が不要な区間ができ，電力や燃料等の消費を削減できるため green control とも呼ばれる [2]．

基本的なスパース制御は一度だけ最適化問題を解いて入力の時系列を得，これを印加する開ループ制御（以下，バッチ制御）である [2, 3]．しかし実システムでは外乱が入ることが予想され，開ループ制御はこれに対処できない．スパース制御を閉ループ系で行うため，スパース制御をモデル予測制御 [6]（以下，MPC）の形式で実行することを考える [4, 5]．本稿では MPC 方式によるスパース制御に対し，2 つの提案を行う．

1 つ目の提案は，時間的に後の入力ほど重たくなるような，時刻ペナルティを導入した目的関数を使うことである [7]．これにより MPC 方式を用いた場合でもバッチ制御と同様の入力が生成されるようになる．

2 つ目の提案は，制御実行時の計算負荷を抑えるため，事前にオフラインで最適解を計算する方法 [8] をスパース最適制御に適用することである [7, 9]．文献 [8] の方法は組込み系や高速な応答が求められる制御対象に有効である．スパース最適制御は入力のとる値が基本的に 0, ±1 の 3 値なので，あらかじめ入力に対応する 3 つの集合を求めておけばよい．実行時は各時刻で状態がどの集合に属すかを調べ，これに基づいて入力を決定する．3 つの集合を正しく表現できれば，各時刻で入力を適切に選択する制御器を得られる．しかし各集合は一般に凸ではなく，表現に工夫を要する．

本稿では次の表記を用いる．実ベクトル $\boldsymbol{u}_\mathrm{d} = \begin{bmatrix} u_\mathrm{d}[0] & u_\mathrm{d}[1] & \cdots & u_\mathrm{d}[N-1] \end{bmatrix}^\mathrm{T} \in \mathbb{R}^N$ に対し，ℓ^1 ノルムを $\|\boldsymbol{u}_\mathrm{d}\|_1 \triangleq \sum_{k=0}^{N-1} |u_\mathrm{d}[k]|$ と定義する．さらに ℓ^∞ ノルムを $\|\boldsymbol{u}_\mathrm{d}\|_\infty \triangleq \max\limits_{k=0,1,\ldots,N-1} |u_\mathrm{d}[k]|$ と定義する．

2　スパース最適制御と数値最適化

次の微分方程式で表される線形システムについて考える：

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + \boldsymbol{b}u(t), \quad t \in [0,\, T], \quad \boldsymbol{x}(0) = \boldsymbol{\xi}. \tag{1}$$

ただし状態を $\boldsymbol{x}(t) \in \mathbb{R}^d$，入力を $u(t) \in \mathbb{R}$ とし，$A \in \mathbb{R}^{d\times d}$，$\boldsymbol{b} \in \mathbb{R}^d$ とする．また $(A, \boldsymbol{b})$ は可到達であり，A は正則と仮定する．

このとき L^1 最適制御は，ほとんどすべての $t \in [0,\, T]$ で 0 もしくは ±1 の値だけをとる．しかし実際は数値計算を効率的に行うため，時間軸の離散化を行い ℓ^1 最適化問題を解いて最適制御を求めることが多い．連続時間システムにおける L^0 最適制御と L^1 最適制御の関係性は文献 [1, 10]，時間離散化による ℓ^1 最適化問題への帰着は文献 [2, 3, 10] を参照されたい．

時間区間を時間幅 h で N 個に分割し，

$$[0,\, T] = [0,\, h) \cup [h,\, 2h) \cup \cdots \cup [Nh - h,\, Nh] \tag{2}$$

とすると，線形システム (1) は次の差分方程式で記述される：

$$\boldsymbol{x}_\mathrm{d}[k+1] = A_\mathrm{d}\boldsymbol{x}_\mathrm{d}[k] + \boldsymbol{b}_\mathrm{d}u_\mathrm{d}[k], \quad k = 0,\, 1,\, \ldots,\, N-1, \qquad \boldsymbol{x}_\mathrm{d}[0] = \boldsymbol{\xi}. \tag{3}$$

ただし時間離散化された状態 $\boldsymbol{x}_\mathrm{d}[k]$，入力 $u_\mathrm{d}[k]$ は

$$\boldsymbol{x}_\mathrm{d}[k] \triangleq \boldsymbol{x}(kh), \quad u_\mathrm{d}[k] \triangleq u(kh) \tag{4}$$

とおく．また

$$A_\mathrm{d} \triangleq \mathrm{e}^{Ah}, \quad \boldsymbol{b}_\mathrm{d} \triangleq \int_0^h \mathrm{e}^{At}\boldsymbol{b}\,\mathrm{d}t \tag{5}$$

である．ここで，$(A_\mathrm{d}, \boldsymbol{b}_\mathrm{d})$ は可到達と仮定する．

このとき，次の ℓ^1 最適化問題を考える：

$$\begin{aligned} \underset{\boldsymbol{u}_\mathrm{d}\in\mathbb{R}^N}{\text{minimize}} \quad & \|\boldsymbol{u}_\mathrm{d}\|_1 \\ \text{subject to} \quad & \|\boldsymbol{u}_\mathrm{d}\|_\infty \le 1, \\ & A_\mathrm{d}^N\boldsymbol{\xi} + \boldsymbol{\Phi}\boldsymbol{u}_\mathrm{d} = \boldsymbol{0}. \end{aligned} \tag{6}$$

ただし

$$\boldsymbol{\Phi} \triangleq \begin{bmatrix} A_\mathrm{d}^{N-1}\boldsymbol{b}_\mathrm{d} & A_\mathrm{d}^{N-2}\boldsymbol{b}_\mathrm{d} & \cdots & A_\mathrm{d}\boldsymbol{b}_\mathrm{d} & \boldsymbol{b}_\mathrm{d} \end{bmatrix} \tag{7}$$

と定義する．問題 (6) を解いて得られる入力の時系列 $\boldsymbol{u}_\mathrm{d} = \begin{bmatrix} u_\mathrm{d}[0] & u_\mathrm{d}[1] & \cdots & u_\mathrm{d}[N-1] \end{bmatrix}^\mathrm{T}$ は時刻 $k = 0,\, 1,\, \ldots,\, N-1$ で $|u_\mathrm{d}[k]| \le 1$ を満たしつつ，状態 $\boldsymbol{x}_\mathrm{d}[k]$ を $\boldsymbol{x}_\mathrm{d}[0] = \boldsymbol{\xi}$ から N ステップで原点，すなわち $\boldsymbol{x}_\mathrm{d}[N] = \boldsymbol{0}$ に遷移させる入力のうち，スパースな（零の要素が多い）ものである．問題 (6) は cvx 等のソフトウェア [11] を使うことで簡単かつ効率的に解くことができる．

3　時刻ペナルティを用いたモデル予測制御

文献 [2, 3] では問題 (6) を解いて入力の時系列を得，そのまま適用するという開ループのスパース制御を考えている．しかし実システムでの運用を考えると外乱の影響を考慮することが求められ，スパース制御を閉ループで実行する必要がある．そのため，スパース制御を MPC 方式で行うことを考える．ここで問題なのはバッチ制御の入力と，MPC の入力が一致するとは限らないことである．

バッチ制御では，ℓ^1 最適化で得た入力の時系列

$$\boldsymbol{u}_\mathrm{d}^{(0)} = \begin{bmatrix} u_\mathrm{d}^{(0)}[0] & u_\mathrm{d}^{(0)}[1] & \cdots & u_\mathrm{d}^{(0)}[N-1] \end{bmatrix}^\mathrm{T} \tag{8}$$

を各時刻 $k = 0, 1, \ldots, N-1$ で印加する．

一方，MPC はまず時刻 $k = 0$ で $\boldsymbol{u}_{\mathrm{d}}^{(0)}$ の最初の値 $u_{\mathrm{d}}^{(0)}[0]$ のみを印加する．次の時刻 $k = 1$ では状態 $\boldsymbol{x}_{\mathrm{d}}[1]$ を観測し，これを $\boldsymbol{\xi}$ と置き換えて問題 (6) を解き，得られた入力の時系列

$$\boldsymbol{u}_{\mathrm{d}}^{(1)} = \begin{bmatrix} u_{\mathrm{d}}^{(1)}[1] & u_{\mathrm{d}}^{(1)}[2] & \cdots & u_{\mathrm{d}}^{(1)}[N] \end{bmatrix}^{\mathrm{T}} \tag{9}$$

の最初の値 $u_{\mathrm{d}}^{(1)}[1]$ を印加する．このように MPC では，時刻 k で最適化問題を解き，得られた時系列 $\boldsymbol{u}_{\mathrm{d}}^{(k)}$ の最初の値 $u_{\mathrm{d}}^{(k)}[k]$ の印加を繰り返す．

$\boldsymbol{u}_{\mathrm{d}}^{(0)}$ は時刻 N で $\boldsymbol{x}_{\mathrm{d}}[N] = \boldsymbol{0}$ を導く入力であり，$\boldsymbol{u}_{\mathrm{d}}^{(1)}$ は時刻 $N+1$ で $\boldsymbol{x}_{\mathrm{d}}[N+1] = \boldsymbol{0}$ を導く入力であるから，基本的に MPC の方が制御が完了する時刻が遅れる．また $u_{\mathrm{d}}^{(0)}[1] = u_{\mathrm{d}}^{(1)}[1]$ となる保証はない．つまり，バッチ制御と MPC の入力は基本的に異なることがわかる．

そこで早い時刻での制御を奨励するために，問題 (6) を次のように変形する：

$$\begin{aligned} \underset{\boldsymbol{u}_{\mathrm{d}} \in \mathbb{R}^N}{\text{minimize}} \quad & \sum_{k=0}^{N-1} |u_{\mathrm{d}}[k]| \alpha^k \\ \text{subject to} \quad & \|\boldsymbol{u}_{\mathrm{d}}\|_\infty \leq 1, \\ & A_{\mathrm{d}}^N \boldsymbol{\xi} + \boldsymbol{\Phi} \boldsymbol{u}_{\mathrm{d}} = \boldsymbol{0}. \end{aligned} \tag{10}$$

ただし，$\alpha > 1$ であり，α^k は時刻が進むに伴い指数関数的に大きくなる．α^k を時刻ペナルティと呼ぶことにする．時刻ペナルティの意味は遅い時刻での入力を重たく評価することで早い時刻の制御を奨励することである．

例として，次の制御対象を考える：

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t). \tag{11}$$

$h = 0.001$ および $N = 2000$ として差分方程式 (3) を考え，問題 (10) を解く．$\boldsymbol{x}(0) = \begin{bmatrix} 1 & -1 \end{bmatrix}^{\mathrm{T}}$ の場合のバッチ制御と MPC の入力を図 1 に示す．なお $\alpha = 1.0015$ とした．

この結果からバッチ制御と MPC の入力が一致することがわかる．なお $\alpha = 1$ のときは一致しない．提案法は外乱がなければバッチ制御に一致し，外乱があればこれに対処できる．以上から提案法が有効であると言える．

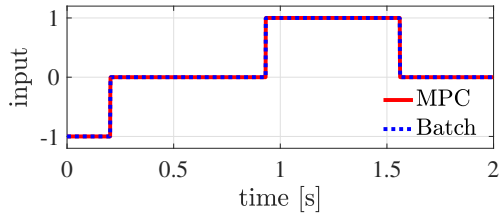

図 1 入力：MPC の場合 (実線) とバッチ制御の場合 (点線)

4 事前計算を用いたモデル予測制御

4.1 動機

MPC を行う際，各時刻にオンラインで最適化を行う方法では実装が困難なケースが考えられる．例えば組込み系や，高速な応答が求められる制御対象の場合である．

そこで本節では，Bemporad らの方法 [8] に基づいてスパース最適制御を行う．この方法では，事前にオフラインで状態に対する最適な入力の系列を計算しておく．特に入力の系列の最初の値に基づいて，状態空間を分割することを考える．分割により得られる集合がどのような集合になるのかということも興味の対象である．

4.2 手順

この方法の手順について説明する．まず，時間幅 h およびホライズン N を設定して様々な初期状態 $\boldsymbol{\xi}$ について問題 (6) を解き，最適な入力の系列 $\boldsymbol{u}_{\mathrm{d}}$ を求める．このとき実行可能となる初期状態 $\boldsymbol{\xi}$ の集合は，N ステップで原点に到達できる初期状態の集合，すなわち可到達集合である．ここでは可到達集合を $\mathcal{R}$ と記述する．また初期状態 $\boldsymbol{\xi}$ に対し ℓ^1 最適化問題 (6) を解いて得られた入力の系列 $\boldsymbol{u}_{\mathrm{d}}$ のうち，最初の値 $u_{\mathrm{d}}[0]$ を $u_0(\boldsymbol{\xi})$ と書く．スパース最適制御では基本的に入力は $0, \pm 1$ の 3 値であり，$u_0(\boldsymbol{\xi})$ の値も基本的に $0, \pm 1$ の 3 値である．厳密には $0, \pm 1$ 以外の値もとり得るが，ここでは入力の値を丸めて考える．

ここで，可到達集合 $\mathcal{R}$ は $u_0(\boldsymbol{\xi})$ のとる値に基づいて 3 つの集合 $\mathbf{R}_-$, $\mathbf{R}_0$, $\mathbf{R}_+$ に分けることができる．ここで $u_0(\boldsymbol{\xi}) = -1$ となる初期状態 $\boldsymbol{\xi}$ の集合を $\mathbf{R}_-$ とし，同様に $\mathbf{R}_0$，$\mathbf{R}_+$ を定める．ホライズン N が大きいとき，各集合は一般に多面体ではなく，凸でもない．このため，集合を表現するのに工夫を要する．

4.3 節では集合 $\mathbf{R}_-$, $\mathbf{R}_0$, $\mathbf{R}_+$ をそれぞれ近似して $\mathbf{P}_-$, $\mathbf{P}_0$, $\mathbf{P}_+$ を得る方法を述べる．制御実行時はこれらの近似集合に基づいて入力を得る．システム (3) において時刻 k の入力を

$$u_{\mathrm{d}}[k] = \begin{cases} -1 & \text{if } \boldsymbol{x}_{\mathrm{d}}[k] \in \mathbf{P}_-, \\ 0 & \text{if } \boldsymbol{x}_{\mathrm{d}}[k] \in \mathbf{P}_0, \\ 1 & \text{if } \boldsymbol{x}_{\mathrm{d}}[k] \in \mathbf{P}_+ \end{cases} \tag{12}$$

のように決定する．高い精度の近似ができれば，各時刻で適切な入力を与える制御器が得られる．

4.3 集合の近似

集合 $\mathbf{P}_-$, $\mathbf{P}_0$, $\mathbf{P}_+$ による近似的表現は Dabbene–Henrion の方法 [12] とサポートベクターマシン [13](以下，SVM) の併用により行う．Dabbene–Henrion の方法によって $\mathbf{P}_0$ を求め，SVM を用いて $\mathbf{P}_-, \mathbf{P}_+$ を分類するための関数 $f(\boldsymbol{x})$ を得る．

Dabbene–Henrion の方法は集合を近似的に表現する多項式を求める方法であり，非凸な集合に対しても有効である．ここでは状態を原点に遷移させることを目的としているため，原点付近で近似精度が高いことが重要である．文献 [7] では状態空間中の点の座標 $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathrm{T}}$ を極座標 $\boldsymbol{x}_{\mathrm{p}} = \begin{bmatrix} r & \theta \end{bmatrix}^{\mathrm{T}}$ に変換し，その上で Dabbene–Henrion の方法を適用した．これにより原点付近の近似精度が高い近似集合を求めた．

極座標 $\boldsymbol{x}_{\mathrm{p}} = \begin{bmatrix} r & \theta \end{bmatrix}^{\mathrm{T}}$ を用いる場合において，$\mathbf{R}_0$ を近似して $\mathbf{P}_0$ を求める方法を記述する．r と θ の単項式を並べたベクトル $\boldsymbol{\pi}(\boldsymbol{x}_{\mathrm{p}})^{\mathrm{T}} = \begin{bmatrix} 1 & r & \theta & r\theta & r^2\theta & \cdots \end{bmatrix}$ に

基づく多項式 $\boldsymbol{\pi}(\boldsymbol{x}_\mathrm{p})^\mathrm{T}\boldsymbol{p}$ を考える．ここでの目的は，集合 $\mathbf{P}_0 = \{\boldsymbol{x}_\mathrm{p} \,|\, \boldsymbol{\pi}(\boldsymbol{x}_\mathrm{p})^\mathrm{T}\boldsymbol{p} \geq 1\}$ が $\mathbf{R}_0$ を含み，かつその面積ができるだけ小さくなるようにすることである．そのために集合 $\mathbf{R}_0$ を含むような箱型の集合 $\mathbf{B} = [r_\mathrm{min}, r_\mathrm{max}] \times [\theta_\mathrm{min}, \theta_\mathrm{max}]$ をとり，$\mathbf{B}$ の中に点 $\boldsymbol{x}_\mathrm{p}^{(j)} = (r^{(j)}, \theta^{(j)})$ (ただし，$j = 1, 2, ..., M$ であり，M は十分大きい）を与える．このとき，次の問題を考える：

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{minimize}} \quad & \int_{\theta_\mathrm{min}}^{\theta_\mathrm{max}} \int_{r_\mathrm{min}}^{r_\mathrm{max}} \boldsymbol{\pi}(\boldsymbol{x}_\mathrm{p})^\mathrm{T}\boldsymbol{p} \,\mathrm{d}r\mathrm{d}\theta \\ \text{subject to} \quad & \boldsymbol{\pi}(\boldsymbol{x}_\mathrm{p}^{(j)})^\mathrm{T}\boldsymbol{p} \geq 1 \;\; \text{if } \boldsymbol{x}_\mathrm{p}^{(j)} \in \mathbf{R}_0, \\ & \boldsymbol{\pi}(\boldsymbol{x}_\mathrm{p}^{(j)})^\mathrm{T}\boldsymbol{p} \geq 0 \;\; \text{otherwise.} \end{aligned} \tag{13}$$

問題 (13) の目的関数は，積分の際に Jacobian r を使わないことに注意する．これによって原点付近が強調され，原点付近の近似精度が高い近似集合が得られる．この問題 (13) を解いて，集合 $\mathbf{R}_0$ の近似集合 $\mathbf{P}_0 = \{\boldsymbol{x}_\mathrm{p} \,|\, \boldsymbol{\pi}(\boldsymbol{x}_\mathrm{p})^\mathrm{T}\boldsymbol{p} \geq 1\}$ が得られる．

ここで問題となるのは状態の次元数が増加し，特に 4 次元以上になると極座標によるアプローチが困難になることである．そこで次元数に関わらず適用可能な座標変換を考える．

簡単のため，状態が 2 次元の場合を説明する．文献 [7] のように極座標を用いず，原点から状態空間中の点 $\boldsymbol{x}$ に対する角度情報を与える必要がある．そこで，図 2 の赤点 $\boldsymbol{x}$ を単位円に射影した青点の座標 $(\bar{x}_1, \bar{x}_2)$ を使い，角度 θ の代わりとして用いる．このとき状態空間中の座標を $\bar{\boldsymbol{x}} = \begin{bmatrix} r & \bar{x}_1 & \bar{x}_2 \end{bmatrix}^\mathrm{T}$ で表現する．

状態空間中の点 $\boldsymbol{x}$ が r，$\bar{x}_1$，$\bar{x}_2$ を用いて表されているとき，$\mathbf{R}_0$ を $\mathbf{P}_0$ で近似する方法を説明する．図 2 の緑点は赤点を直線 $x_1 = 1$ に射影したもので，この x_2 座標を X_2 と書く．このとき r，$\bar{x}_1$，$\bar{x}_2$ の単項式を並べて得られるベクトル $\boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T} = \begin{bmatrix} 1 & r & \bar{x}_1 & \bar{x}_2 & \cdots \end{bmatrix}$ に基づく多項式 $\boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T}\boldsymbol{p}$ を考える．$\bar{x}_1$，$\bar{x}_2$ について，X_2 を使うことで

$$\bar{x}_1 = \frac{1}{\sqrt{1 + X_2^2}}, \quad \bar{x}_2 = \frac{X_2}{\sqrt{1 + X_2^2}} \tag{14}$$

と表わせる．また，

$$\mathrm{d}\theta = \frac{1}{1 + X_2^2} \,\mathrm{d}X_2 \tag{15}$$

である．さらに θ の積分範囲 $[\theta_\mathrm{min}, \theta_\mathrm{max}]$ に対応する X_2 の範囲は $[-\infty, \infty]$ である．問題 (13) の場合と同様に $\mathbf{R}_0$ を含むような箱型集合 $\mathbf{B}$ をとり，$\mathbf{B}$ の中に点 $\bar{\boldsymbol{x}}^{(j)} = (r^{(j)}, \bar{x}_1^{(j)}, \bar{x}_2^{(j)})$ (ただし，$j = 1, 2, ..., M$ であり，M は十分大きい）を与える．問題 (13) について式 (14), (15) を用いて変換することで，解くべき問題は次のようになる:

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{minimize}} \quad & \int_{-\infty}^{\infty} \int_{r_\mathrm{min}}^{r_\mathrm{max}} \frac{\boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T}\boldsymbol{p}}{1 + X_2^2} \,\mathrm{d}r\mathrm{d}X_2 \\ \text{subject to} \quad & \boldsymbol{\pi}(\bar{\boldsymbol{x}}^{(j)})^\mathrm{T}\boldsymbol{p} \geq 1 \;\; \text{if } \bar{\boldsymbol{x}}^{(j)} \in \mathbf{R}_0, \\ & \boldsymbol{\pi}(\bar{\boldsymbol{x}}^{(j)})^\mathrm{T}\boldsymbol{p} \geq 0 \;\; \text{otherwise.} \end{aligned} \tag{16}$$

問題 (16) を解いて，$\mathbf{P}_0 = \{\bar{\boldsymbol{x}} \,|\, \boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T}\boldsymbol{p} \geq 1\}$ が得られる．

図 2 で示した座標変換の考え方は，状態の次元数が変わったとしても単位球 $r = 1$ に射影した座標，超平面 $x_1 = 1$ に射影した座標を考えればよい．そのため，次元数に関わらず適用できる．

例として状態が 3 次元の場合を考える．状態空間中の座標を原点からの距離 r と単位球 $r = 1$ に射影した点の座標 $(\bar{x}_1, \bar{x}_2, \bar{x}_3)$ を使って $\bar{\boldsymbol{x}} = \begin{bmatrix} r & \bar{x}_1 & \bar{x}_2 & \bar{x}_3 \end{bmatrix}^\mathrm{T}$ のように表現する．このとき，ベクトル $\boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T} = \begin{bmatrix} 1 & r & \bar{x}_1 & \bar{x}_2 & \bar{x}_3 & \cdots \end{bmatrix}$ に基づく多項式 $\boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T}\boldsymbol{p}$ を考える．3 次元の場合，$\mathbf{P}_0$ は次の問題を解くことで得られる：

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{minimize}} \quad & \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{r_\mathrm{min}}^{r_\mathrm{max}} \frac{\boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T}\boldsymbol{p}}{(1 + X_2^2 + X_3^2)^{\frac{3}{2}}} \,\mathrm{d}r\mathrm{d}X_2\mathrm{d}X_3 \\ \text{subject to} \quad & \boldsymbol{\pi}(\bar{\boldsymbol{x}}^{(j)})^\mathrm{T}\boldsymbol{p} \geq 1 \;\; \text{if } \bar{\boldsymbol{x}}^{(j)} \in \mathbf{R}_0, \\ & \boldsymbol{\pi}(\bar{\boldsymbol{x}}^{(j)})^\mathrm{T}\boldsymbol{p} \geq 0 \;\; \text{otherwise.} \end{aligned} \tag{17}$$

ただし X_2, X_3 は初めの点 $\boldsymbol{x}$ を超平面 $x_1 = 1$ に射影した点の座標である．問題 (17) を解いて，$\mathbf{P}_0 = \{\bar{\boldsymbol{x}} \,|\, \boldsymbol{\pi}(\bar{\boldsymbol{x}})^\mathrm{T}\boldsymbol{p} \geq 1\}$ が得られる．

$\mathbf{P}_-, \mathbf{P}_+$ はこれらを分類する関数 $f(\boldsymbol{x})$ を SVM により導出することで得る．このとき $f(\boldsymbol{x}) = 0$ が $\mathbf{P}_-, \mathbf{P}_+$ の境界となる．$\mathbf{P}_-, \mathbf{P}_+$ は次のように定める：

$$\mathbf{P}_- = \{\boldsymbol{x} \,|\, \boldsymbol{x} \in \mathcal{R} \text{ かつ } \boldsymbol{x} \notin \mathbf{P}_0 \text{ かつ } f(\boldsymbol{x}) < 0\}, \tag{18}$$

$$\mathbf{P}_+ = \{\boldsymbol{x} \,|\, \boldsymbol{x} \in \mathcal{R} \text{ かつ } \boldsymbol{x} \notin \mathbf{P}_0 \text{ かつ } f(\boldsymbol{x}) > 0\} \tag{19}$$

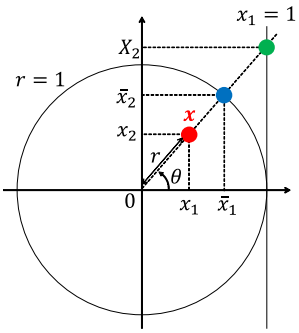

図 2 座標変換法：直交座標でとった点（赤点）から単位円 $r = 1$ への射影（青点）と直線 $x_1 = 1$ への射影（緑点）

4.4 数値例

例として，次の制御対象について考える：

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.5 & -1.5 & 1.5 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t), \quad t \in [0, T]. \tag{20}$$

$T = 2$ と設定する．時間区間 $[0, T]$ を時間幅 $h = 0.001$ で $N = 2000$ ステップに分割する．時間離散化された差分方程式 (3) を考え，様々な初期状態 $\boldsymbol{\xi}$ に対する ℓ^1 最適

化問題 (6) を解くことで，可到達集合 $\mathcal{R}$ を求める．$\mathcal{R}$ とそれを分割した $\mathbf{R}_-, \mathbf{R}_0, \mathbf{R}_+$ に属する初期状態 $\boldsymbol{\xi}$ の点を図 3 の左図に示す．$\mathbf{R}_-$ と $\mathbf{R}_+$ は原点に対して点対称である．さらに $\mathbf{R}_0$ は $\mathbf{R}_-$ と $\mathbf{R}_+$ の間に挟まれるように存在する．$\mathbf{R}_0$ に属す点のみを図 3 の右図に示す．

4.3 節に記述した方法で $\mathbf{P}_-, \mathbf{P}_0, \mathbf{P}_+$ を求める．まず問題（17）を考え，$\mathbf{R}_0$ の点に基づいて $\mathbf{P}_0$ を得る．得られた $\mathbf{P}_0$ を図 4 に示す．さらに SVM により関数 $f(\boldsymbol{x})$ を導出することで，$\mathbf{P}_-$ と $\mathbf{P}_+$ を定める．

得られた $\mathbf{P}_-, \mathbf{P}_0, \mathbf{P}_+$ に基づいて制御器を設計し，数値実験を行う．初期状態を $\boldsymbol{x}(0) = \begin{bmatrix} 0.4 & -0.4 & 0.2 \end{bmatrix}^{\mathrm{T}}$ と設定する．入力，状態の推移を図 5，図 6 に示す．MPC の場合とバッチ制御で一度だけ ℓ^1 最適化した場合を比較すると，ほぼ同じ入力と状態の推移が得られることがわかる．開ループ方式は外乱に対処できないが，MPC の方式で行った提案法はこれに対処した入力を生成できるため，提案法が有効だと言える．

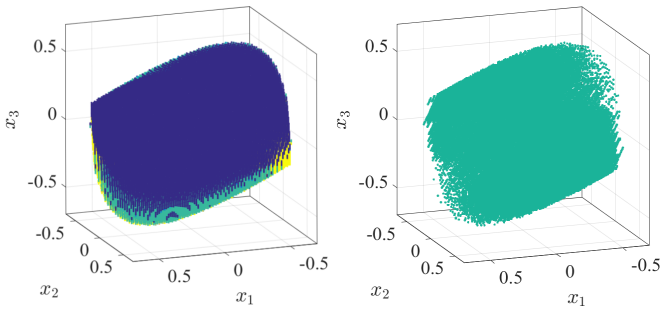

図 3 $\mathcal{R}$ に属す点：$\mathbf{R}_-$（青点），$\mathbf{R}_0$（緑点），$\mathbf{R}_+$（黄点）

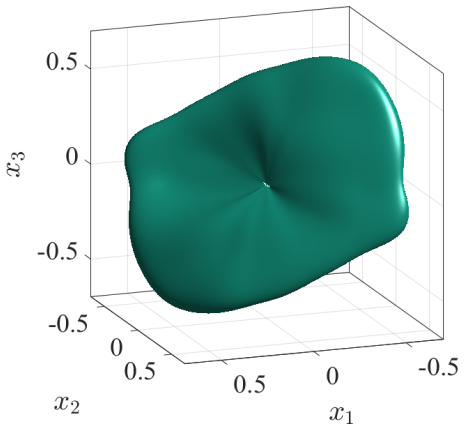

図 4 $\mathbf{R}_0$ の近似集合 $\mathbf{P}_0$

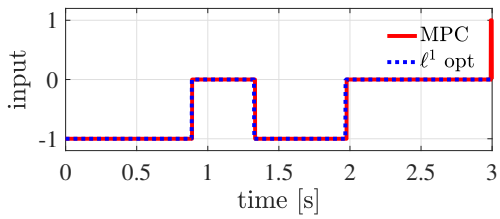

図 5 入力：提案法の場合（実線）と従来法の場合（点線）

5 おわりに

本稿ではモデル予測制御方式によるスパース制御について 2 つの提案を行った．1 つ目は MPC とバッチ制御で同じ入力を生成するため，目的関数に時刻ペナルティを導入したことである．2 つ目は事前計算を用いる方法 [8] によるスパース制御を提案し，実装法を与えたことである．

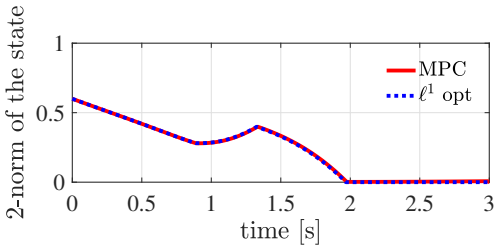

図 6 状態の 2 ノルム：提案法の場合（実線）と従来法の場合（点線）

今後の課題は 2 つ目の提案に対し，離散値制御 [14] にも適用し得るよう方法を拡張することである．

参考文献

[1] M. Nagahara, D. E. Quevedo, and D. Nešić: Maximum hands-off control: A paradigm of control effort; *IEEE Transactions on Automatic Control*, Vol. 61, No. 3 (2016)

[2] M. Nagahara, D. E. Quevedo, and D. Nešić: Hands-off control as green control, *SICE Control Division Multi Symposium* (2014) `http://arxiv.org/abs/1407.2377`

[3] N. Challapalli, M. Nagahara, M. Vidyasagar: Continuous hands-off control by CLOT norm minimization; *IFAC-PapersOnLine*, Vol. 50, No. 1, pp. 14454–14459 (2017) `http://arxiv.org/abs/1611.02071`

[4] T. Ikeda and M. Nagahara: Value function in maximum hands-off control for linear systems, *Automatica*, Vol. 64, pp. 190–195 (2016)

[5] N. Nagahara, J. Østergaard, and D. E. Quevedo: Discrete-time hands-off control by sparse optimization, *EURASIP Journal on Advances in Signal Pricessing*, Vol. 2016, No. 1, pp. 1–8 (2016)

[6] 児島晃, 大塚敏之: モデル予測制御の考え方, 計測と制御, Vol. 42, No. 4, pp. 310–312 (2003)

[7] 岩田拓海, 大石泰章, 永原正章： モデル予測制御によるスパース制御の実現, 第 61 回自動制御連合講演会予稿集, pp. 1059–1064 (2018)

[8] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos: The explicit linear quadratic regulator for constrained systems; *Automatica*, Vol. 38, No. 1, pp. 3–20 (2002)

[9] 岩田拓海，大石泰章，永原正章： 事前計算を用いたモデル予測制御によるスパース制御, 第 6 回計測自動制御学会制御部門マルチシンポジウム, 発表予定 (2019)

[10] 永原正章: スパースモデリング—基礎から動的システムへの応用—, コロナ社 (2017)

[11] M. Grant and S. Boyd: CVX: Matlab software for disciplined convex programming, version 2.1 (2014) `http://cvxr.com/cvx`

[12] F. Dabbene and D. Henrion: Set approximation via minimum-volume polynomial sublevel sets; in *Proceedings of the European Control Conference*, pp. 1114–1119 (2013)

[13] 竹内一郎, 鳥山昌幸: サポートベクトルマシン, 講談社 (2015)

[14] T. Ikeda, M. Nagahara, S. Ono: Discrete-valued control of linear time-invariant systems by sum-of-absolute-values optimization; *IEEE Transactions on Automatic Control*, Vol. 62, No. 6, pp. 2750–2763 (2017)