モデル予測制御のための高速時系列予測アルゴリズム

藤原　廉[†,††]　松原　靖子[†]　木村　輔[†]　櫻井　保志[†]

† 大阪大学産業科学研究所　〒 567–0047 大阪府茨木市
†† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市
E-mail: †{r-fujiwr88,yasuko,tasuku,yasushi}@sanken.osaka-u.ac.jp

あらまし　本論文では，大規模制御応答時系列データストリームにおける制御量予測手法である C-CAST について述べる．C-CAST は，制御量（**C**ontrolled sequence），動作信号，操作量の三要素で構成される制御応答時系列データから，制御量の時系列パターンを捉えることで，パターン間の遷移に基づく高速な制御量予測を実現する．より具体的には，動作信号および操作量を考慮できるように動的システムを拡張し，制御応答時系列データを適応型動的システムとしてモデル化することで，重要なパターンや複雑なパターンの遷移を柔軟に表現する．提案手法は，(a) 制御応答時系列データストリームから重要な特徴を発見し，刻々と変化していく潜在的なパターンやパターン遷移を高速かつ自動的に認識し，(b) 将来的な制御量予測を実現する．さらに，提案手法は (c) データストリームの長さに依存しない．実データを用いた実験では，提案手法が制御応答時系列データストリームの中から重要な時系列パターンを発見し，制御量予測を高精度に行うことを確認した．さらに，最新の既存手法と比較し大幅な精度向上を達成し，その計算速度はデータサイズに依存せず，高速に動作することを明らかにした．

キーワード　時系列データストリーム，制御量予測, ストリーム処理

1　ま え が き

宇宙ロケットの誘導制御，飛行機の姿勢制御，自動車の車両追従制御，工作機械・工業ロボットの自動制御など，時間発展する制御対象を動的システム（Dynamic System）としてモデル化するモデル予測制御（MPC：Model Predictive Control）[1,2] は，数多くの分野で導入され活躍している．この制御方法では，予測した制御量（$Controlled\ sequence$）に基づいて現在の操作量を調整することで，未来の制御量をシステムの目標と一致するように繰り返し最適化を行う．

しかし，MPC は制御対象に予測モデルが存在することを前提としており，モデル化されていない場合は MPC を使用できない．そのため，制御対象の内部挙動が複雑である場合や制御対象に任意に決定できる成分（動作内容や材料など）が含まれる場合，モデル化が困難もしくはモデル化にかかるコストが大きくなる．そのような場合において，制御対象のモデル化が不要である PID 制御 [3] が使用される．その簡易性から PID 制御の適用範囲は MPC よりも広い．一方で PID 制御は現在の出力をもとに操作量を決定するため，予測値を使用する場合と比べて制御の反映までに時間がかかり，結果としてその制御性能は MPC に劣る．このような制御対象において，データからの学習によって将来の制御量の予測が可能であれば，PID 制御ではなく制御性能の高い MPC が適用可能となる．そこで，近年，深層学習をはじめとした機械学習に基づくモデル予測制御の研究に期待が集まっている [4,5]．計算機の能力が大幅に向上したことを背景に，サンプル時間が短い工業装置等のシステムの制御，さらには高度な論理計算を要する制御問題への応用が模索されている [6–9]．

また，一般に工業装置から得られる制御応答時系列データは，装置の機差や稼働条件によって異なる時系列パターンを持つ．また，装置に任意に設定可能な成分 (動作内容や材料等) を含む場合，それらの時系列パターンに関する事前情報は利用できない．加えて，工業装置の作業内容の切り替わりに伴って動作信号や操作量で構成される外部信号が変化し，追うように制御量が急激に変化する．そのため，将来の制御量を正確に予測するためには，外部信号に基づいて時系列パターンの急激な遷移を捉える必要がある．そしてその際に事前情報を利用することはできない．したがって，現在のデータに応じて，制御応答時系列データから時系列パターンを柔軟かつ動的に推定し，推定した時系列パターンを外部信号に基づいて切り替えることで制御量を予測する必要がある．

そこで，本論文ではモデル化が困難である工業装置を対象とした制御量予測手法である C-CAST を提案する．C-CAST は動的システムの概念を拡張し，外部信号を陽に考慮可能な適応型動的システムを使用することで，制御応答データの時系列パターン（本研究では，“レジーム” と呼ぶ）やパターン遷移を柔軟に表現する．より具体的には，以下の問題を扱う．

制御量 $\boldsymbol{X} = \{\boldsymbol{x}(1), ..., \boldsymbol{x}(t_c)\}$ および外部信号 $\boldsymbol{Z} = \{\boldsymbol{z}(1), ..., \boldsymbol{z}(t_c), \boldsymbol{z}(t_c+1), ...\boldsymbol{z}(t_c+l_s)\}$ が与えられたとき，現時刻 t_c から l_s ステップ先の制御量 $\boldsymbol{x}(t_c+l_s)$ を予測する．このとき，$\boldsymbol{Z}$ は，動作信号 $\boldsymbol{Sig} = \{\boldsymbol{sig}(1), ..., \boldsymbol{sig}(t_c), \boldsymbol{sig}(t_c+1), ...\boldsymbol{sig}(t_c+l_s)\}$ および操作量 $\boldsymbol{U} = \{\boldsymbol{u}(1), ..., \boldsymbol{u}(t_c), \boldsymbol{u}(t_c+1), ...\boldsymbol{u}(t_c+l_s)\}$ から構成される．

1.1　具　体　例

図 1 はある工業装置の制御応答時系列データにおける，C-CAST と既存手法の出力の様子を比較したものである．

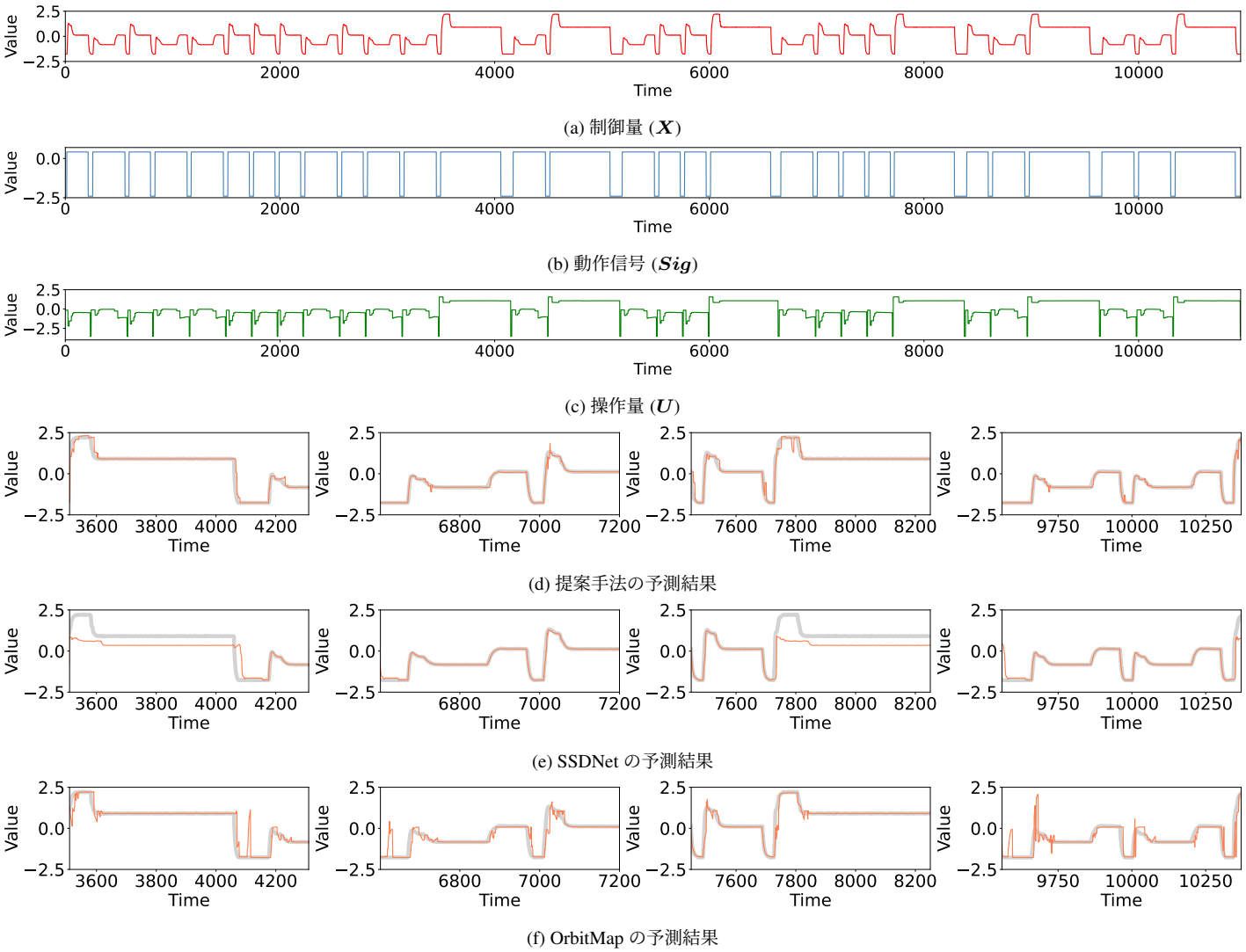


図 1: 実データにおける 5 ステップ先の予測結果：(a)–(c) オリジナルデータ，(d)–(f) 各時刻における提案手法および比較手法の予測結果.

図 1(a)–図 1(c) はそれぞれオリジナルデータの制御量（赤色の線），動作信号（青色の線），操作量（緑色の線）を示しており，図 1(d)–図 1(f) は，図 1(a) の時刻 3400 以降の 4 つの異なる時刻における，C-CAST および既存手法が予測した 5 ステップ先の制御量を示している.

ここで，橙色の線は各手法の予測制御量を，灰色の線はオリジナルデータの制御量を示す.

図 1(d) が示すように，提案手法は新たなパターンへ瞬時に適応し，外部信号を考慮してパターン遷移を予測することで，パターンの遷移に伴う制御値の急上昇および急降下を含む，将来の制御量を正確に予測できている．一方，図 1(e) より，SSDNet は，学習データに存在しない未知のパターン 3 の制御量を予測できず，精度が大きく低下していることが確認できる．同様に，図 1(f) より，OrbitMap は，パターン遷移のタイミングを誤って予測したため，予測結果に複数のスパイクが含まれることがわかる.

1.2 本論文の貢献

本研究では，大規模制御応答時系列データストリームのための制御量予測手法である C-CAST を提案する．C-CAST は次の特長を持つ.

（1） データ内に含まれる制御量の時系列パターン（レジーム）に関する事前知識を必要とせず，パターンの潜在的な振舞いを時間遷移と他の入力から把握する.

（2） 把握した時系列パターン（レジーム）を用いて最適な制御量予測を行う.

（3） モデル推定と予測に必要な計算コストはデータサイズに依存しない.

2 関 連 研 究

時系列データの解析および予測に関する研究は多岐にわたり，自己回帰モデル（AR ：autoregressive model），線形動的システム（LDS ：linear dynamical systems）およびカルマンフィルタ（KF ：Kalman filters） などの古典的時系列解析を活用した手法

や，近年増加している Deep Neural Network（DNN）に基づく手法など数多く提案されている．

古典的時系列解析を拡張した時系列予測手法として，AWSOM [10]，TBATS [11]，PLiF [12] をはじめ，数多く提案されている．RegimeCast [13] は大量に生成され続ける多次元センサデータから潜在的なパターンをリアルタイムに推定し，適応的に将来を予測し続ける．また OrbitMap [14] は時系列データの潜在状態および状態遷移をリアルタイムに推定し，長期的な予測を可能とする．しかしながら，これらの手法は外部信号を考慮した状態遷移やモデリングに対応しておらず，制御応答時系列の表現には適していない．また，大規模時系列におけるパターン発見のための手法も数多く提案されている [15–19]．これらの手法は時系列上の重要なパターンを発見することが可能であるが，一方で，将来予測を目的としていない．

外部入力を考慮した DNN に基づく時系列の解析に関する研究もさかんである [20–23]．たとえば，DeepAR [23] は RNN を用いて予測対象となる時系列データ以外の時系列データから特徴を抽出し，それらを用いて対象となる時系列データの将来の確率分布を予測している．また，SSDNet [20] は状態空間モデルと Transformer [24] を組み合わせた時系列予測手法である．過去の時系列データと予測先の時刻までの外部入力を使用することで，外的な要因によるトレンドや季節性を利用した時系列データの将来予測を可能としている．しかし，これらの DNN に基づく手法は，モデルの学習において膨大な計算量が必要とされ，刻々と取得されるデータからリアルタイムにモデルを更新し，最適な予測値を推定し続けることは困難である．

3 提案モデル

本章では提案手法のための基本的な概念と提案モデルについて述べる．

3.1 OrbitMap におけるレジーム

本節では大規模時系列予測の既存手法である OrbitMap [14] において導入されているレジーム θ_{base} について述べる．まず、レジーム θ_{base} は次の 2 種類の状態から構成される．

- $\boldsymbol{s}(t)$：時刻 t におけるレジーム θ_{base} の潜在値.
- $\boldsymbol{e}(t)$：時刻 t における制御量の推定値．潜在値 $\boldsymbol{s}(t)$ を用いて生成される．

OrbitMap において，複数の潜在的パターンを含んでいる時系列イベントストリームを次のような潜在的非線形微分方程式として表現する．

$$\frac{d\boldsymbol{s}(t)}{dt} = p + Q\boldsymbol{s}(t) + A\boldsymbol{S}(t) \tag{1}$$

$$e(t) = u + V\boldsymbol{s}(t) \tag{2}$$

ここで，初期条件を $s(0) = s_0$，$d\boldsymbol{s}(t)/dt$ を時刻 t の導関数とし，$\boldsymbol{S}(t)$ を $\boldsymbol{s}(t)$ の 2 次形式の行列とする．また，それぞれの潜在的パターンを式 1，式 2 のように表現し，単一の潜在的な非線形動的システムにおけるパラメータ集合 $\{s_0, p, Q, A, u, V\}$ をレジーム θ_{base} と定義する．

3.2 動的システム

動的システムとは，現在の出力が同時刻の操作量や過去の操作量，および，内部状態に依存するシステムのことである．動的システムは，さまざまな分野，たとえば，宇宙工学，自動車，化学プラント，製薬，あるいは近年では，経済分野などにおいて応用されている．また，操作量を陽に考慮したモデルを使用することで，モデル化された現象そのものを制御できる．

動的システムにおいて，現在の操作量を $\boldsymbol{u}(t)$，現在の内部状態を $\boldsymbol{s}(t)$ とすると，これらは**式 (3)** のように表現される．このとき過去の入力は現在の内部状態に内包されている． また，センサーなどによって現時刻の状態が観測されるとき，センサーによる雑音を $\boldsymbol{d}$ とすると，観測値 $\boldsymbol{e}(t)$ は**式 (4)** のように表される．まとめると，動的システムは**図 2** のように表現される．

$$\frac{d\boldsymbol{s}(t)}{dt} = A\boldsymbol{s}(t) + B\boldsymbol{u}(t) \tag{3}$$

$$\boldsymbol{e}(t) = C\boldsymbol{s}(t) + \boldsymbol{d} \tag{4}$$

動的システムにおいて制御量に直接影響を与えるのは操作量とされている．したがって動的システムは操作量を考慮することは可能だが，動作信号についてはその限りではない．そのため，次節では，動的システムの概念を操作量だけでなく，動作信号も考慮できるように拡張した提案手法である C-CAST について述べる．

3.3 C-CAST

本研究の目的は，制御応答データストリームが与えられたとき，その中から重要かつ潜在的な時系列パターンを発見し，現在の潜在的なパターンおよび外部信号から将来の制御量を予測することである．そこで本研究では，3.1 節で述べたレジームを拡張し，制御応答データストリームにおけるレジームを定義する．そして外部信号に基づいてレジームの遷移を捉えることで制御量予測を実現する．このような条件下において，提案手法は次の 2 つの特長を持つ必要がある．

(P1) 動的システムで表現されるレジーム

(P2) 外部信号を考慮したレジーム遷移

3.3.1 動的システムで表現されるレジーム (P1)

まず，本節では単一のレジームの表現法について述べる．つまり，ここではレジームの遷移は存在しないものとする．また，3.1 節で定義したレジーム θ_{base} は外部からの影響を考慮していないため，3.2 節にて述べた動的システムに基づくレジーム θ を定義する．

レジーム θ は次の 3 種類の状態から構成される．

- $\boldsymbol{s}(t)$：時刻 t におけるレジーム θ の潜在値.
- $\boldsymbol{u}(t)$：時刻 t における操作量.

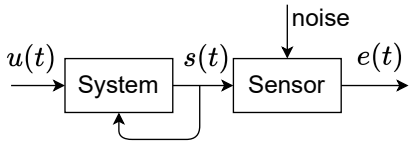


図 2: 動的システムにおける入出力の様子

- $\boldsymbol{e}(t)$：時刻 t における制御量の推定値．潜在値 $\boldsymbol{s}(t)$ および操作量 $\boldsymbol{u}(t)$ を用いて生成される．

ここで，レジーム制御部（Controller）を導入する．レジーム制御部は時刻 t における外部信号 $\boldsymbol{z}(t)$ および潜在値 $\boldsymbol{s}(t)$ を入力とし，操作量 $\boldsymbol{u}(t)$ および帰還入力値 $\boldsymbol{s}_{FB}(t)$ を出力する．レジーム制御部および帰還入力値 $\boldsymbol{s}_{FB}(t)$ の詳細は次節にて述べる．

単一のレジームおよびレジーム制御部は**図 3**(a) のように表される．レジームはレジーム制御部から時刻 t の操作量 $u(t)$ および帰還入力値 $\boldsymbol{s}_{FB}(t)$ を与えられることで，次の時刻の潜在値 $\boldsymbol{s}(t+1)$ および推定値 $\boldsymbol{e}(t+1)$ を生成する．つまり，式 (3) を次の**式 (5)** のように拡張し，式 (5) によって潜在値 $\boldsymbol{s}(t+1)$ を生成し，式 (4) で $\boldsymbol{e}(t+1)$ を生成する．

$$\frac{d\boldsymbol{s}(t)}{dt} = A\boldsymbol{s}_{FB}(t) + B\boldsymbol{u}(t) \tag{5}$$

まとめると，以下の定義を得る．

［定義 1］（レジーム θ） θ を単一のレジームにおけるパラメータ集合とする：$\theta = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{d}\}$.

3.3.2 外部信号を考慮したレジームの遷移 (P2)

次に本研究では，レジームの遷移方法として C-SWITCHER を提案する．C-SWITCHER において，レジーム制御部は状態遷移ベクトル $(\boldsymbol{s}_i^{out}, \boldsymbol{z}_i^{out}, \boldsymbol{s}_i^{in})$ を持ち，図 3(b) で示すように，レジーム制御部が現時刻の潜在値 $\boldsymbol{s}(t)$ および外部信号 $\boldsymbol{z}(t)$ を元に，適切なレジームに帰還入力値 $\boldsymbol{s}_{FB}(t)$ および操作量 $\boldsymbol{u}(t)$ を入力することで各時刻の推定値 $\boldsymbol{e}(t)$ を生成する．そして，入力を受けたレジームは潜在値 $\boldsymbol{s}(t+1)$ をレジーム制御部にフィードバックする．そして，フィードバックされた潜在値 $\boldsymbol{s}(t+1)$ および外部入力が遷移条件を満たしたとき，図 3(c) に示すようにレジーム制御部は入力経路を遷移先のレジームに切り替える．このように入力先を外部信号とフィードバックされた潜在値 $\boldsymbol{s}(t+1)$ を使用して切り替えることで，外部信号を考慮したレジームの動的な遷移を実現する．すなわち，C-SWITCHER では次のような規則に従ってレジーム遷移が発生する．

- レジーム θ_i は各時刻 t においてレジーム制御部から帰還入力値 $\boldsymbol{s}_{FB}(t)$ および操作量 $\boldsymbol{u}(t)$ を受け取り $\boldsymbol{s}(t+1)$ を式 (5) に従って生成し，レジーム制御部に入力する．レジーム遷移が発生するまでこの操作は繰り返される．
- レジーム制御部が時刻 t_s における外部信号 $\boldsymbol{z}(t_s)$，および潜在値 $\boldsymbol{s}_i(t_s)$ を受け取ったとき，$\boldsymbol{z}(t_s)$ が $\boldsymbol{z}_i^{out}$ に十分に近い，かつ $\boldsymbol{s}(t_s)$ が $\boldsymbol{s}_i^{out}$ に十分に近い場合，レジーム θ_j への遷移が発生する．具体的にはレジーム制御部が $\boldsymbol{s}_j(t_s) = \boldsymbol{v}_j^{in}$ および，操作量 $\boldsymbol{u}(t_s)$ をレジーム θ_j に入力し，時刻 t_s 以降の推定値をレジーム θ_j を使用して推定する．

ここでは，レジーム θ_i，レジーム θ_j 間の遷移は，潜在値 $\boldsymbol{s}_i(t)$ または外部信号 $\boldsymbol{z}(t)$ が遷移ベクトルに十分に接近した時のみ発生する（i.e., $D_s(t) = ||\boldsymbol{s}_i(t) - \boldsymbol{s}_i^{out}|| < \rho$ かつ $D_z(t) = ||\boldsymbol{z}(t) - \boldsymbol{z}_i^{out}|| < \rho_z$）．ここで，$\rho$ は遷移強度，ρ_z は制御強度とする．

まとめると，以下のように定式化される．

C-SWITCHER. $\boldsymbol{s}_{FB}(t)$ を時刻 t にレジーム制御部が出力する帰還入力値とする．このとき，レジーム θ_i から θ_j への遷移は次のように $\boldsymbol{s}_{FB}(t)$ が切り替わることによって表現される．

$$\boldsymbol{s}_{FB}(t) = \begin{cases} \boldsymbol{s}_i(t) & (1 \leqq t < t_s) \\ \boldsymbol{s}_j^{in} & (t = t_s) \\ \boldsymbol{s}_j(t) & (t_s < t) \end{cases} \tag{6}$$

ここで，$D_s(t) = ||\boldsymbol{s}_i(t) - \boldsymbol{s}_i^{out}||$，$D_z(t) = ||\boldsymbol{z}(t) - \boldsymbol{z}_i^{out}||$ としたとき，$t_s = \underset{\{t | D_s(t) \leqq \rho, D_z(t) \leqq \rho_z\}}{\arg\min} D_s(t)$ となる．

ここでの $\boldsymbol{s}_i(t)$ はレジーム θ_i が時刻 t において生成した潜在値であり，また，$\{\boldsymbol{s}_i^{out}, \boldsymbol{z}_i^{out}, \boldsymbol{s}_j^{in}\}$ はレジーム θ_i とレジーム θ_j における遷移ベクトルである．

まとめると，提案モデルは次の要素で構成される．

［定義 2］（C-CAST の全パラメータ集合 M） 全レジームの集合を $\Theta = \theta_1, ...\theta_r$，$V$ を全遷移ベクトル集合とするとき（i.e., $\{\boldsymbol{s}^{out}, \boldsymbol{z}^{out}, \boldsymbol{s}^{in}\} \in V$），提案モデルの全パラメータ集合 M は $\{\Theta, V\} \in M$ となる．

4 アルゴリズム

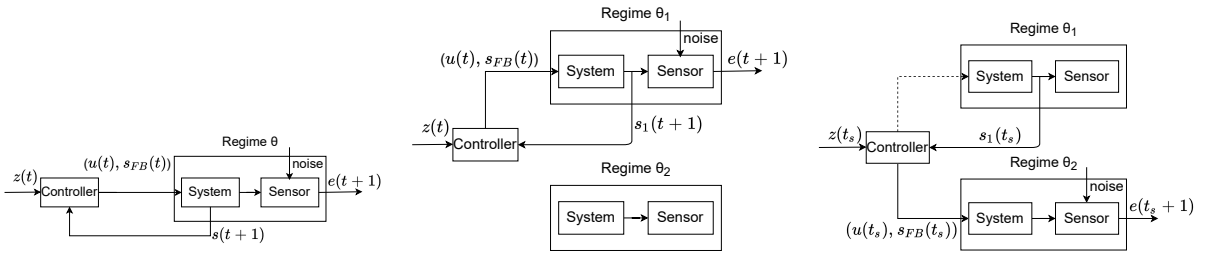
本章では，制御応答時系列の予測手法である C-CAST のアルゴリズムについて述べる．

4.1 問題定義

ここで本手法で必要な概念について定義を行う．また**表 1** に主な記号と定義を示す．

［定義 3］（制御量データストリーム $\boldsymbol{X}$） $\boldsymbol{X}$ を制御量から構成されるデータストリーム $\boldsymbol{X} = \{\boldsymbol{x}(1), ..., \boldsymbol{x}(t_c)\}$ とし，現時刻を t_c とする．

［定義 4］（外部信号データストリーム $\boldsymbol{Z}$） $\boldsymbol{z}(t)$ を時刻 t における動作信号と制御量の組 $\boldsymbol{z}(t) = \{\boldsymbol{u}(t), \boldsymbol{sig}(t)\}$ とするとき，$\boldsymbol{Z}$ を動作信号および操作信号から構成される外部信号データストリーム $\boldsymbol{Z} = \{\boldsymbol{z}(1), ..., \boldsymbol{z}(t_c), \boldsymbol{z}(t_c+1), ...\boldsymbol{z}(t_c+l_s)\}$ とする．このとき，現時刻を t_c とし，l_s ステップ先の予測を行うものとする．



(a) 単一レジームとレジーム制御部　(b) 時刻 t における制御量推定　(c) レジーム遷移

図 3: C-CAST 概要：(a) 単一のレジームはレジーム制御部から入力 $(u(t), s_{FB}(t))$ を受け取ることで推定値 $e(t+1)$ を出力する．(b) 各時刻においてレジーム制御部は 1 つのレジームのみに入力を与え，推定値を出力する．(c) レジーム遷移条件を満たしたとき，レジーム制御部は入力を与えるレジームを切り替える．

表 1: 主な記号と定義

記号	定義
t_c	現在の時刻
$\boldsymbol{X}$	制御量のデータストリーム
$\boldsymbol{x}(t)$	時刻 t における制御量
$\boldsymbol{U}$	操作量量のデータストリーム
$\boldsymbol{u}(t)$	時刻 t における操作量
$\boldsymbol{sig}$	動作信号のデータストリーム
$\boldsymbol{sig}(t)$	時刻 t における動作信号
$\boldsymbol{Z}$	外部信号のデータストリーム：$\{\boldsymbol{U}, \boldsymbol{Sig}\} \in \boldsymbol{Z}$
$\boldsymbol{z}(t)$	時刻 t における外部信号：$\boldsymbol{z}(t) = \{\boldsymbol{u}(t), \boldsymbol{sig}(t)\}$
$\boldsymbol{s}(t)$	時刻 t における潜在値
$\boldsymbol{e}(t)$	時刻 t における推定値
$\boldsymbol{X}_c$	カレントウィンドウ：$\boldsymbol{X}_c = \boldsymbol{X}[t_m : t_c]$
$\boldsymbol{Z}_{ref}$	リファレンスウィンドウ：$\boldsymbol{Z}_{ref} = \boldsymbol{Z}[t_m : t_c + l_s]$
Θ	全レジームのパラメータ集合：$\Theta = \{\theta_1, ...\theta_r\}$
V	全遷移ベクトルの集合：$\{\boldsymbol{s}^{out}, \boldsymbol{z}^{out}, \boldsymbol{s}^{in}\} \in V$
M	C-CAST の全パラメータ集合：$\{\Theta, V\} \in M$

ここで，毎時刻において新たなエントリー $x(t_c)$ が発生し，t_c が増加するものとする．そこで，最新の時刻において観測された制御量の集合のうち，現在のレジーム θ_c の開始時点 t_s から現時刻までの部分集合をカレントウィンドウと呼び，次のように定義する．

［定義 5］（カレントウィンドウ $\boldsymbol{X}_c$） $\boldsymbol{X}_c = \boldsymbol{X}[t_s : t_c]$ を長さ l_c のカレントウィンドウとする．ここで，$\boldsymbol{X}_c$ は制御量データストリーム $\boldsymbol{X}$ の時刻 t_s から時刻 t_c までの部分シーケンスを表す．

カレントウィンドウ $\boldsymbol{X}_c$ が与えられたとき，次の目標は，パラメータ集合 Θ から最適なレジームを発見し，式 (5)，式 (4) に基づき l_s ステップ先の 制御量を推定することである．

また，l_s ステップ先の予測をするにあたり，時刻 $t_c + l_s$ までの動作信号，操作量は既知とする．そこで，現時刻で既知の動作信号および操作信号のうち，現在のレジーム θ_c の開始時点 t_s から時刻 $t_c + l_s$ までの部分集合をリファレンスウィンドウと呼び，次のように定義する．

［定義 6］（リファレンスウィンドウ $\boldsymbol{Z}_{ref}$） $\boldsymbol{Z}_{ref} = \boldsymbol{Z}[t_s : t_c + l_c]$ を長さ $l_c + l_s$ のリファレンスウィンドウとする．ここで，$\boldsymbol{Z}_{ref}$ は外部信号データストリーム $\boldsymbol{Z}$ の時刻 t_s から時刻 $t_c + l_c$ までの部分シーケンスを表す．

Algorithm 1 C-CAST

1: **Input:** (a) New value $\boldsymbol{x}(t_c)$ at time points t_c
(b) New value $\boldsymbol{z}(t_c + l_s)$ at time points $t_s + l_s$
(c) Parameter set $M = \{\Theta, V\}$
(d) Candidate $C = \{\theta_c, \boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\}$
2: **Output:** (a) Updated parameter set M'
(b) Updated candidate C'
(c) Estimated variables $\boldsymbol{e}(t_c + l_s)$
3: /* Estimate C and Update M */
4: $\{M', C'\} =$ C-ESTIMATOR$(x(t_c), z(t_c + l_s), M, C)$
5: /* Forecast l_s step ahead estimated value */
6: $\boldsymbol{e}(t_c + l_s) =$ C-GENERATOR$(z(t_c + l_s), M', C')$
7: **return** $\{M', C', \boldsymbol{e}(t_c + l_s)\}$

まとめると，本論文で取り組む問題を以下のように定義する．

［問題 1］ 制御量データストリーム $\boldsymbol{X}$ および外部信号データストリーム $\boldsymbol{Z}$ が与えられたとき，l_s ステップ先の制御量を出力し続ける．より具体的には，各時刻 t_c において，

(a) カレントウィンドウ $\boldsymbol{X}_c$ およびリファレンスウィンドウ $\boldsymbol{Z}_{ref}$ を用いてパラメータ集合 M を更新する．
(b) 更新後のパラメータ集合 M およびリファレンスウィンドウ $\boldsymbol{Z}_{ref}$ を用いて l_s ステップ先の制御量 $\boldsymbol{x}(t_c + l_s)$ を予測する．

これらを逐次的，かつ高速に行うことが求められる．ここで，現時刻 t_c におけるレジームを, θ_c ，θ_c の直前に存在したレジームを θ_p とする．このとき，(a) では，現時刻のレジーム θ_c，レジーム間の遷移ベクトル $\{\boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\}$ を更新している．そこで，現在のレジーム，遷移ベクトルをまとめて，候補モデル $C = \{\theta_c, \boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\}$ として保持する．

4.2 提案アルゴリズム

C-CAST は次のアルゴリズムで構成される (Algorithm 1).

- C-ESTIMATOR ：カレントウィンドウ $\boldsymbol{X}_c$，リファレンスウィンドウ $\boldsymbol{Z}_{ref}$，パラメータ集合 M，および候補モデル

Algorithm 2 C-ESTIMATOR

Input: (a) New value $\boldsymbol{x}(t_c)$ at time points t_c
(b) New value $\boldsymbol{z}(t_c + l_s)$ at time points $t_s + l_s$
(c) Parameter set $M = \{\Theta, V\}$
(d) Candidate $C = \{\theta_c, \boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\}$
Output: (a) Updated parameter set M'
(b) Updated candidate C'
$\boldsymbol{X}_c = \boldsymbol{X}[t_s : t_c]$; $\boldsymbol{Z}_{ref} = \boldsymbol{Z}[t_s : t_c + l_s]$
/* Update current regime */
$\{\theta_c, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\} = \underset{\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}}{\arg\min} f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref})$
if $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref}) > \rho$ **then**
  /* Find a better regime in Θ */
  $\{\theta_c, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\} = \underset{\theta \in \Theta, \boldsymbol{s}^{out}, \boldsymbol{s}^{in}}{\arg\min} f(\theta_c, \boldsymbol{s}^{out}, \boldsymbol{s}^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref})$
  if $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref}) > \rho$ **then**
    /* Create new regime */
    $\{\theta_c, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\} = RegimeCreation(\boldsymbol{X}_c, \boldsymbol{Z}_{ref})$
    $\Theta = \Theta \cup \theta_c$
  end if
end if
/* Detect segment X_c end and Insert new transition vector */
if $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref}) > \rho$ **then**
  $V = V \cup \{\boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}, \boldsymbol{s}_c^{in}\}$
  $(\boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}) = (\boldsymbol{s}_c^{out}, \boldsymbol{z}(t_c))$
  $\theta_c = \boldsymbol{s}_c^{in} = \boldsymbol{s}_c^{out} = \emptyset$
end if
$M' = \{\Theta, V\}$
$C' = \{\theta_c, \boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\}$
return $\{M', C'\}$

C を与えたとき，パラメータ集合 M，および候補モデル C を更新する (Algorithm 2).

- C-GENERATOR：候補モデル C，およびパラメータ集合 M，リファレンスウィンドウ $\boldsymbol{Z}_{ref}$ を用いて，l_s ステップ先の推定値 $\boldsymbol{e}(t_c + l_s)$ 予測する (Algorithm 3).

4.2.1 C-ESTIMATOR

カレントウィンドウ $\boldsymbol{X}_c$，リファレンスウィンドウ $\boldsymbol{Z}_{ref}$，パラメータ集合 M，候補モデル C を与えたときを考える．ここで，損失関数を $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref})$ と定義する．$f(\cdot)$ は推定値とオリジナルデータのフィッテング精度を示す（i.e., $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref}) = \sum_{t=t_s}^{t_c} ||\boldsymbol{x}(t) - \boldsymbol{e}(t)||$）．

関数 C-ESTIMATOR は次の手順で行われる．

(I) $(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in})$ を $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref})$ が最小になるように更新する．

(II) $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref}) > \rho$ となるとき，パラメータ集合 M から，$f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref})$ が最小になるパラメータを探索する．

(III) $f(\theta_c, \boldsymbol{s}_c^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{X}_c, \boldsymbol{Z}_{ref}) > \rho$ となるとき，新たなレジームを生成し，パラメータ集合 Θ に追加する．

(IV) 遷移ベクトルの生成（もし存在すれば）：これまでの手順のなかで，遷移ベクトル集合 V に存在しない遷移が確認されたとき，新たな遷移ベクトルを生成し，遷移ベクトル集合 V に追加する．

Algorithm 3 C-GENERATOR

Input: (a) New value $\boldsymbol{z}(t_c + l_s)$ at time points $t_s + l_s$
(b) Parameter set $M = \{\Theta, V\}$
(c) Candidate $C = \{\theta_c, \boldsymbol{s}_p^{out}, \boldsymbol{z}_p^{out}, \boldsymbol{s}_c^{in}, \boldsymbol{s}_c^{out}\}$
Output: (a) Estimated variables $\boldsymbol{e}(t_c + l_s)$
/* Initialize */
$s(t_c) = \boldsymbol{s}_c^{in}$; $t_s = t_c$; $\boldsymbol{Z}_{ref} = \boldsymbol{Z}[t_s : t_c + l_s]$
while $t_s < t_c + l_s$ **do**
  /* Generate latent variables and estimated variables */
  $(\boldsymbol{s}(t_s), ..., \boldsymbol{s}(t_c + l_s), \boldsymbol{e}(t_c + l_s)) = generate(\theta_c, \boldsymbol{Z}_{ref}, \boldsymbol{s}(t_s))$
  /* Search transition */
  $t_{best} = t_c + l_s$
  for $i = 1$ to r **do**
    /* Find any transition vector that is to close trajectory */
    if $(\boldsymbol{s}_c^{out}, \boldsymbol{z}_c^{out}, \boldsymbol{s}_i^{in}) \in V$ **then**
      $t = \underset{\substack{||\boldsymbol{s}(t) - \boldsymbol{s}_c^{out}|| < \rho, \\ t_s < t < t_c + l_s}}{\arg\min} ||\boldsymbol{s}(t) - \boldsymbol{s}_c^{out}||$
      if $||\boldsymbol{z}(t) - \boldsymbol{z}_c^{out}|| < \rho_z$ and $t < t_{best}$ **then**
        $t_{best} = t$; $\theta_f = \theta_i$ // Select shortest transition
      end if
    end if
  end for
  /* Transition $\theta_c \to \theta_f$ */
  $\theta_c = \theta_f$; $t_s = t_{best}$; $\boldsymbol{s}(t_s) = \boldsymbol{s}_f^{in}$
end while
return $\boldsymbol{e}(t_c + l_s)$

Regime Creation. カレントウィンドウ X_c に未知の時系列パターンが現れたとき，新たなレジーム $\theta = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{d}\}$ を推定する必要がある．OrbitMap [14] においてパラメータ推定アルゴリズムに expectation–maximization（EM）アルゴリズムを採用しているが，このアルゴリズムにおいて外部入力は白色ノイズとして扱われるため，今回の提案手法には適さない．そこで，本研究では Adjoint Sensitivity Method [25,26] を拡張したパラメータ推定アルゴリズムを提案する．具体的には，Adjoint Sensitivity Method を使用し，損失関数に対するパラメータ $\{\boldsymbol{A}, \boldsymbol{B}\}$ の勾配を求め，それを用いた勾配法で $\{\boldsymbol{A}, \boldsymbol{B}\}$ を推定し，パラメータ $\{\boldsymbol{C}, \boldsymbol{d}\}$ を Levenberg-Marquardt（LM）アルゴリズム [27] を用いて推定する．これを誤差が収束するまで繰り返す．

4.2.2 C-GENERATOR

次にモデル候補 C，パラメータ集合 M およびリファレンスウィンドウ Z_{ref} が与えられたときを考える．関数 C-GENERATOR は次の手順で行われる．

(I) $\boldsymbol{s}(t_c)$ を初期値として，時刻 $t_c + l_s$ までの推定値をレジーム θ_c を使用して生成する．

(II) 生成された潜在値 $\{\boldsymbol{s}(t_c), ..., \boldsymbol{s}(t_c + l_s)\}$ および，外部信号 $\{\boldsymbol{z}(t_c), ...\boldsymbol{z}(t_c + l_s)\}$ について，$D_s(t_s) = ||\boldsymbol{s}_i(t_s) - \boldsymbol{s}_i^{out}|| < \rho$ かつ $D_z(t_s) = ||\boldsymbol{z}(t_s) - \boldsymbol{z}_c^{out}|| < \rho_z$ となる時刻 t_s が存在すれば，潜在値を $\{\boldsymbol{s}(t_c), ..., \boldsymbol{s}(t_s - 1), \boldsymbol{s}_f^{in}, ..., \boldsymbol{s}_f(t_c + l_s)\}$ と更新する．このとき，$\{\boldsymbol{s}_f^{in}, ..., s_f(t_c + l_s)\}$ は，$\boldsymbol{s}_f^{in}$ を初期値として，時刻 $t_c + l_s$ までの推定値をレジーム θ_f を使用して生成する．

(III) もし $t_s < t_c + l_s$ ならば，$t_c = t_s$ として手順 (II) に戻る．

そうでないなら，推定値を出力して終了する．

5 評価実験

本論文では C-CAST の有効性を検証するため，実データを用いた実験を行った．本章では以下の項目について検証する．

Q1　制御量予測に対する提案手法の有効性
Q2　リアルタイム予測に対する提案手法の精度の検証
Q3　データストリームに対する計算時間の検証

また，実験は 512GB のメモリ，AMD EPYC 7502 2.5GHz の 32 コア CPU および NVIDIA RTX A6000 48GB GPU を搭載した Linux マシン上で実施した．

比較手法. 有効性を検証するために，以下の最新の既存手法と比較を行った．

- OrbitMap [14]：大規模時系列イベントストリームのリアルタイム予測手法．モデル推定時の閾値は学習時に最もフィッテング性能が良い値を使用した．
- SSDNet [20]：Transformer に基づく深層モデルを用いた時系列予測手法であり，将来の外部信号を考慮して予測を行う．Transformer のエンコーダ層，デコーダー層のユニット数はそれぞれ 3 とした．また，最適化アルゴリズムとして Adam [28] を使用し，300 エポック学習した中で最もフィッテング性能が良いモデルを使用した．

データセット. ある工業装置の制御応答時系列データストリームを使用した．データセットは平均値と分散値で正規化 (z-normalization) して使用した．また，データストリームの時刻 0 から時刻 3400 までを学習データとして使用した．具体的には，SSDNet では学習データをモデル学習のみに使用し，提案手法および OrbitMap は初期パラメータの推定およびモデル学習に用いた．また，SSDNet のウィンドウサイズについて，10 から 50 まで 10 刻みでそれぞれ実験を行い，学習データにおいて誤差が最小となったウィンドウサイズを用いた．

5.1 Q1：提案手法の有効性

本節では，制御応答時系列データに対する C-CAST の予測能力を検証する．図 1 および**図 4** は実際の制御応答時系列データに対する提案手法および比較手法の予測結果である．そして，図 4 では各パターン（レジーム A-C）の開始時における予測結果を比較する．すでに 1 章の図 1 においても示したように，提案手法は外部信号を利用することで動作信号に基づいてパターン遷移を予測可能であるため，遷移の開始時に発生する制御量の急激な変動を予測できている．一方，図 4(b) および図 4(c) より，比較手法は制御量の急激な変動に対応できていないことが分かる．特に，学習データに含まれないレジーム C の変化において，制御量の立ち上がりが遅れていることを確認できる．SSDNet は，C-CAST と同様に外部信号を考慮できるため，部分的に急激な変動に対応できているが提案手法ほど精度は高くない．一方，OrbitMap は，外部信号を扱えず動作信号に基づくパターン遷移を予測できていないため，制御量の急激な変動を予測できていない．

5.2 Q2：提案手法の精度

本節では，C-CAST の予測精度を検証するため，既存手法である SSDNet および OrbitMap と精度比較を行った．

図 5 は，工業制御応答データストリームにおける C-CAST の予測結果の精度を示している．具体的には，オリジナルデータの制御量と 5 ステップ先の予測制御量の推定値の二乗平均誤差（RMSE：root mean square error）および平均絶対誤差（MAE：mean absolute error）を示している．図に示す通り，提案手法は最新の既存手法である SSDNet および OrbitMap と比較し，2 つの評価指標のどちらにおいても高い予測精度を持つ．SSDNet はモデルをオンラインに更新できず，テスト区間に含まれる未知のパターンに対応できないため精度が低下している．また，OrbitMap は外部信号を考慮した時系列パターンの表現やパターン遷移の予測ができないため，適切に制御量を予測することができない．

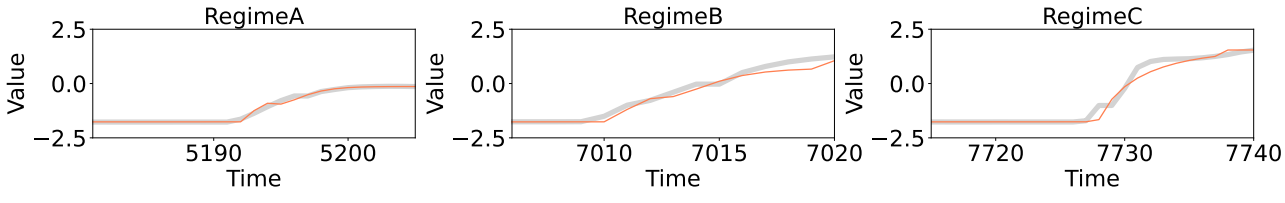
また，状態の遷移がさらに複雑になった場合，初めて現れる遷移に関しては提案手法を含め全手法で予測精度が低下する．しかし，その後に再び同じ状態遷移が現れた際，提案手法はそれに適応可能である．一方で，SSDNet はモデルをオンラインに更新できないため，初めて現れた時と同様に予測精度が低下する．そして OrbitMap は状態遷移を適切に予測できないため，初めて現れた時と同様に予測精度が低下する．つまり，状態遷移が複雑になるほど各手法の予測精度は低下するが，提案手法は比較手法と比較して精度の低下が小さいと考えられる．

5.3 Q3：提案手法の計算時間

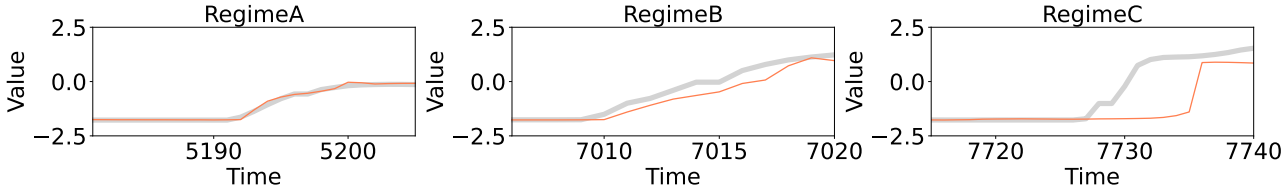
続いて，提案手法の計算コストについて検証する．**図 6**(a) は，各時刻 t_c における計算コストを，提案手法の C-CAST と既存手法である SSDNet および OrbitMap と比較したものである．また，図 6(b) では，提案手法のデータストリーム全体の計算時間の分布を示している．モデルの実行において，提案手法および OrbitMap は CPU のみを用い，深層学習である SSDNet は CPU および GPU を併用した．なお，図中の y 軸は対数スケールで示している．

図 6(a) に示す通り，大規模時系列の高速予測手法である OrbitMap と同様に, C-CAST はデータストリームの長さに依存せず，高速に動作することが確認できる．さらに，提案手法は，深層学習である SSDNet と比較して大幅な性能向上を達成した．具体的には，SSDNet と比較し 136,700 倍の高速化を実現している．

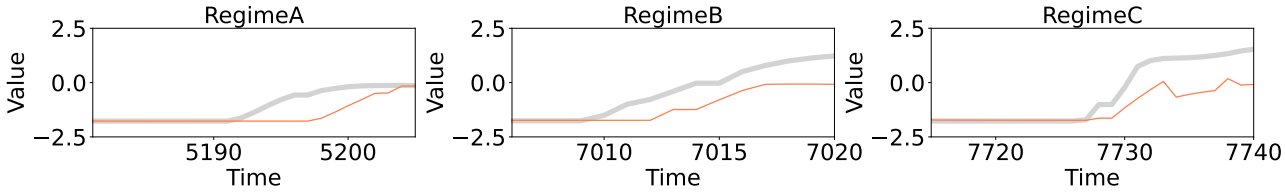
また，本実験で使用した装置において MCMPC [29] を適用する場合，予測値を求めた後に操作量を最適化する必要がある．そのため，データサイズの大きさに対する計算時間だけでなく，計算時間の分布も非常に重要である．図 6(b) は，データセットの各時刻の処理にかかる時間の度数分布をプロットしたものである．本実験で使用したデータセットのサンプリングレートは $100ms$ であり，これはモデル予測制御のためには少



(a) 提案手法の予測結果

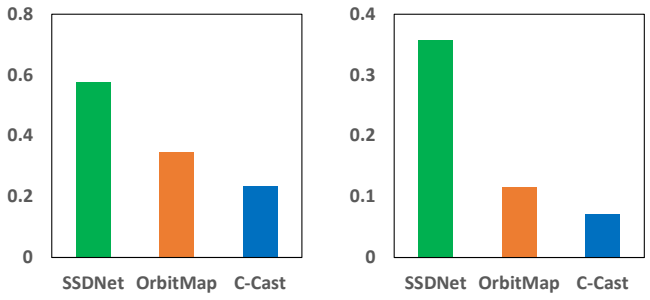


(b) SSDNet の予測結果



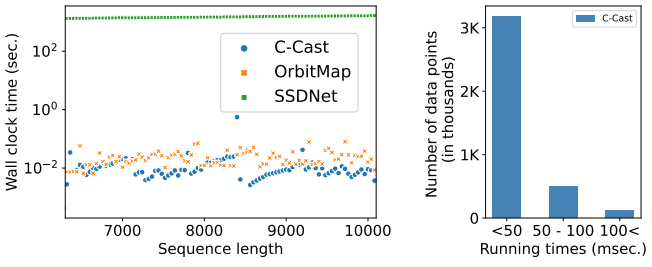
(c) OrbitMap の予測結果

図 4: 各パターン (レジーム) 開始時における予測精度：(a) 提案手法および (b)–(c) 比較手法の予測結果



(a) RMSE　　(b) MAE

図 5: 工業制御応答ストリームにおける制御量の予測精度



(a) 各時刻における計算速度　　(b) 平均計算速度

図 6: 各時刻 t_c における計算コスト（a）と平均値（b）

なくとも $100ms$ 以内に計算を完了させる必要があることを意味する．$3.7K$ データポイント（全データポイントの約 97%）では $100ms$ 以内に，$3.1K$ データポイント（全データポイントの約 83%）では $50ms$ 以内に計算が完了する．このように，C-CAST は制御部分に操作量を最適化するための十分な時間を与えることができる。

6 むすび

本論文では，制御応答時系列データストリームにおける制御量予測手法である C-CAST について述べた．C-CAST は，外部信号である動作信号および操作量を考慮できるように動的システムを拡張し，制御応答時系列データを適応型動的システムとしてモデル化することで，制御量の重要なパターンや外部信号によるパターンの遷移を柔軟に表現し，最適な制御量予測を実現する．

実データを用いた実験では，C-CAST が制御応答時系列データに対し，制御量の時系列パターンやパターン遷移を高速かつ継続的に発見し，制御量予測を高精度に行うことを確認した．今後の課題として，様々な制御応答時系列データをより柔軟に表現するための高度なモデル学習や，予測モデルを利用し，最適な操作量を決定する手法について検討していく予定である．

謝辞　本研究の一部は JSPS 科研費，JP20H00585, JP21H03446, JP22K17896, 国立研究開発法人情報通信研究機構委託研究 NICT 03501, 総務省 SCOPE JP192107004, JST-AIP 加速課題 JPMJCR21U4, ERCA 環境研究総合推進費 JPMEERF20201R02, の助成を受けたものです．

文　献

[1] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[2] David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Autom.*, 36(6):789–814, 2000.

[3] Karl Johan Åström and Tore Hägglund. *PID Controllers: Theory, Design, and Tuning*. ISA - The Instrumentation, Systems and Automation Society, 1995.

[4] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N

Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[5] Benjamin Karg and Sergio Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Trans. Cybern.*, 50(9):3866–3878, 2020.

[6] D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, 1990.

[7] Toshiyuki Ohtsuka. A continuation/gmres method for fast computation of nonlinear receding horizon control. *Autom.*, 40(4):563–574, 2004.

[8] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In Danica Kragic, Antonio Bicchi, and Alessandro De Luca, editors, *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pages 1433–1440. IEEE, 2016.

[9] Peter Ortner and Luigi del Re. Predictive control of a diesel engine air path. *IEEE Trans. Control. Syst. Technol.*, 15(3):449–456, 2007.

[10] Spiros Papadimitriou, Anthony Brockwell, and Christos Faloutsos. Adaptive, hands-off stream mining. In Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer, editors, *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*, pages 560–571. Morgan Kaufmann, 2003.

[11] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[12] Lei Li, B. Aditya Prakash, and Christos Faloutsos. Parsimonious linear fingerprinting for time series. *Proc. VLDB Endow.*, 3(1):385–396, 2010.

[13] Yasuko Matsubara and Yasushi Sakurai. Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1045–1054. ACM, 2016.

[14] Yasuko Matsubara and Yasushi Sakurai. Dynamic modeling and forecasting of time-evolving data streams. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 458–468. ACM, 2019.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[16] David Hallac, Sagar Vare, Stephen P. Boyd, and Jure Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 215–223. ACM, 2017.

[17] Takato Honda, Yasuko Matsubara, Ryo Neyama, Mutsumi Abe, and Yasushi Sakurai. Multi-aspect mining of complex sensor sequences. In Jianyong Wang, Kyuseok Shim, and Xindong Wu, editors, *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pages 299–308. IEEE, 2019.

[18] Kouki Kawabata, Yasuko Matsubara, and Yasushi Sakurai. Automatic sequential pattern mining in data streams. In Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu, editors, *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 1733–1742. ACM, 2019.

[19] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. Autoplait: automatic mining of co-evolving time sequences. In Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu, editors, *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 193–204. ACM, 2014.

[20] Yang Lin, Irena Koprinska, and Mashud Rana. Ssdnet: State space decomposition neural network for time series forecasting. In James Bailey, Pauli Miettinen, Yun Sing Koh, Dacheng Tao, and Xindong Wu, editors, *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*, pages 370–378. IEEE, 2021.

[21] Yuan Xue, Denny Zhou, Nan Du, Andrew M. Dai, Zhen Xu, Kun Zhang, and Claire Cui. Deep state-space generative model for correlated time-to-event predictions. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1552–1562. ACM, 2020.

[22] Yuyang Wang, Alex Smola, Danielle C. Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6607–6617. PMLR, 2019.

[23] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[25] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. 1987.

[26] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583, 2018.

[27] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In *Numerical Analysis*, pages 105–116, 1978.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[29] Nikolas Kantas, JM Maciejowski, and A Lecchini-Visintini. Sequential monte carlo for model predictive control. In *Nonlinear model predictive control*, pages 263–273. Springer, 2009.