機械学習を用いた円柱周り流れにおける低次元モードの抽出と時間発展予測

Extraction of low dimensional modes in a flow around a circular cylinder and prediction of their temporal evolutions using machine learning

○ 村田 高彬，慶大院, 横浜市港北区日吉 3-14-1
深見 開，慶大院, 横浜市港北区日吉 3-14-1
深潟 康二，慶大, 横浜市港北区日吉 3-14-1, E-mail： fukagata@mech.keio.ac.jp
Takaaki Murata, Keio University, Hiyoshi 3-14-1, Kohoku-ku, Yokohama 223-8522
Kai Fukami, Keio University, Hiyoshi 3-14-1, Kohoku-ku, Yokohama 223-8522
Koji Fukagata, Keio University, Hiyoshi 3-14-1, Kohoku-ku, Yokohama 223-8522

We present a new framework to extract low-dimensional modes and to predict their temporal evolutions. Autoencoder-type Convolutional Neural Network (CNN) which can learn nonlinearity of data is used to extract low-dimensional modes and Sparse Identification of Nonlinear Dynamics (SINDy) is performed to obtain ordinary differential equations with nonlinear terms of CNN encoded data. The proposed method is applied to a flow around a circular cylinder at $\mathrm{Re}_D = 100$. The CNN trained by fluctuation components of velocity vector u, v shows better results than the snapshot Proper Orthogonal Decomposition in terms of the energy reconstruction rate. The errors of CNN and SINDy are accumulated when we reproduce the time evolution of flow field. The error of CNN can be reduced by devising a better network structure; however, the error of SINDy depends on the waveform of encoded values.

1. 背景および目的

流体の流れは，複雑で，時間・空間的に多くのパラメータで表される現象である．しかし，この複雑な現象の中には，低次元モードと呼ばれる特徴的なパターンがいくつか含まれており，低次元モードの重ね合わせで流れの現象を再現できると言われている．

低次元モードを効率的に抽出する手法の一つとして，固有直交分解 (Proper Orthogonal Decomposition, POD)[1] がある．POD を用いると，流れ場の時系列データからエネルギーの高い順に低次元モードを抽出することができる．そのため，流れ現象に内在する重要な構造を抽出できる．また，時間的に変化する流れ場は，低次元モードの線形結合で表される．すなわち，低次元モードを介することによって，流れ場の時間変化するパラメータを減らすことができる．Bergmann et al.[2] は POD を介し，流れ場のパラメータを減らした最適制御を行い，計算コストの低減を確認した．

しかし，POD をはじめとした多くの低次元モードの抽出法は線形の理論に基づいており，非線形性の強い流れ現象に用いるには限界があると考えられている．Alfonsi & Primavera[3] は，$\mathrm{Re}_\tau = 180$ の乱流チャネルを POD で分解し，95%のエネルギーを再現するのに 7260 もの低次元モードが必要であると報告した．

そこで，本研究では非線形な回帰関数を学習できる機械学習に注目した．その中でも，データを低次元化する手法として畳み込みニューラルネットワーク (Convolutional Neural Network, CNN)[4] によるオートエンコーダ [5] がある．尾亦と白山 [6] は CNN オートエンコーダと POD を用いて翼周り流れ場の低次元化を行い，非定常流れ場の比較法を提案した．

CNN オートエンコーダはエンコーダとデコーダで構成されており，流れ場はエンコーダで圧縮され，デコーダで復元される．ここで，デコーダ部のみに注目すると，圧縮したデータの時間発展が得られれば，流れ場の時間発展が復元できることになる．そこで，本研究では圧縮されたデータの時間発展を得る方法として Sparse Identification of Nonlinear Dynamics (SINDy)[7] に注目した．SINDy では，LASSO[8] など疎な説明変数を得る重回帰分析を用いて，時系列データからその運動の支配方程式を得ることができる．

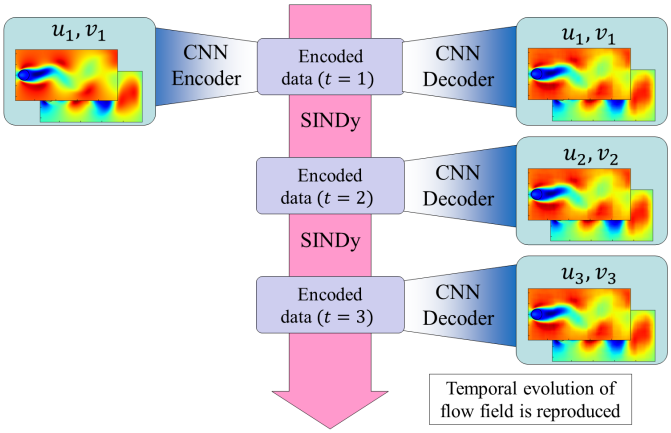

Fig. 1: Outline of this study.

本研究では，Fig. 1 のように CNN オートエンコーダと SINDy を組み合わせ，流れの縮約モデルの新しいフレームワークを構築することを目的とする．具体的には，$\mathrm{Re}_D = 100$ の円柱周り流れにおいて，CNN オートエンコーダを作成することにより低次元モードを抽出し，次に，エンコーダを用いて得られた低次元モードの時系列データに対し SINDy を行い常微分方程式を得ることにより圧縮場の時間発展を予測する．圧縮場の時間発展をデコーダに与えることで，流れ場を再現する．

2. 計算手法および訓練手法

2.1 訓練データの作成手法

CNN の訓練データとして，直接数値シミュレーション (DNS) を行って流れ場のデータを作成する．DNS の支配

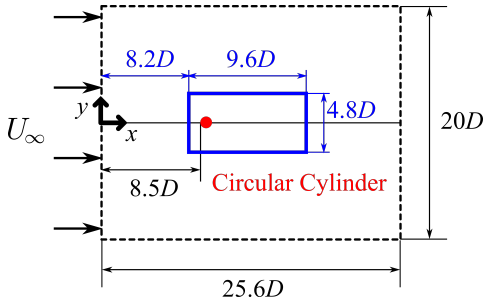

Fig. 2: Computational domain and the area of train data.

方程式は，連続の式と Navier-Stokes 方程式，

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\nabla \cdot (\boldsymbol{u}\boldsymbol{u}) - \nabla p + \frac{1}{\mathrm{Re}_D}\nabla^2 \boldsymbol{u} \tag{2}$$

である．円柱の直径 D を代表長さとしたレイノルズ数 Re_D は 100 とする．計算領域は Fig. 2 の点線で囲まれた領域で，計算の時間刻みは $\Delta t = 2.5 \times 10^{-3}$ とする．計算格子は幅 $\Delta x = \Delta y = 0.025$ の等間隔直交格子を用い，円柱表面の境界条件として，埋め込み境界法 [9] を用いる．

本研究では物体周りに注目するので，Fig. 2 の青実線で囲まれた $8.2 \leq x \leq 17.8$，$-2.4 \leq y \leq 2.4$ の領域における主流方向速度 u と主流垂直方向速度 v を訓練データとして用いる．また，訓練データの時間間隔は $100\Delta t = 0.25$ とする．DNS で計算されるストローハル数は約 0.1716 であるので，約 23 枚で 1 周期を構成している．瞬時場のデータを 10000 枚用意し，訓練データとして用いる．

2.2 CNN オートエンコーダ

CNN は局所的な特徴を抽出する畳み込み層と，特徴をまとめあげるプーリング層から構成されるニューラルネットワークの一つで，データの特徴を保ちながらデータを圧縮することができる．CNN オートエンコーダはエンコーダとデコーダで構成され，エンコーダ $\mathcal{F}_{\mathrm{enc}}$ を用いると，m 次元の入力データ $\boldsymbol{q}$ は l 次元 $(l \ll m)$ のデータ $\boldsymbol{r}$ に圧縮される．また，圧縮されたデータ $\boldsymbol{r}$ に対し，デコーダ $\mathcal{F}_{\mathrm{dec}}$ を用いると，m 次元の出力 $\tilde{\boldsymbol{q}}$ まで復元される．まとめると，

$$\boldsymbol{r} = \mathcal{F}_{\mathrm{enc}}(\boldsymbol{q}) \in \mathbb{R}^l \tag{3}$$

$$\tilde{\boldsymbol{q}} = \mathcal{F}_{\mathrm{dec}}(\boldsymbol{r}) \in \mathbb{R}^m \tag{4}$$

となる．CNN オートエンコーダは，入力 $\boldsymbol{q}$ と出力 $\tilde{\boldsymbol{q}}$ の差が小さくなるように訓練される．

本研究で用いた CNN の代表的なものを Table 1 に示す．Conv2D(h, w, n) 層では $h \times w$ のフィルターによる畳込みを n 回行う．Deconv2D(h, w, n) 層では，$h \times w$ のフィルターによる逆畳込みを n 回行うことにより，データの拡大を行う．出力層以外のすべての Conv2D 層，Deconv2D 層に対し活性化演算を行う．なお，* のついた Conv2D 層ではゼロパディングを行わない．また，Maxpooling 層では 2×2 の小領域に対して最大の値のみ残るようにデータの圧縮を行い，Upsampling 層では 1 つの値を 2×2 の小領域にコピーしてデータの拡大を行う．

この CNN において，入力層から 8th Conv2D 層までがエンコーダで，$(384, 192, 2)$ のサイズのデータを $(2, 1, 1)$ のサイズのデータまで圧縮する．1st Deconv2D 層から出力層までがデコーダで，$(2, 1, 1)$ のサイズに符号化されたデータを元のサイズのデータまで復元する．

Table 1 のような従来型 CNN オートエンコーダ (C-CNN) では，2 つの符号化された値に対し，モードごとの場の様子を見ることができない．そこで，本研究では Table 2 のような構造を持つモード分解型 CNN オートエンコーダ (MD-CNN) を作成する．まず，従来の CNN オートエンコーダと同様の構造のエンコーダ $\mathcal{F}_{\mathrm{enc}}$ で $(384, 192, 2)$ のデータ $\boldsymbol{q}$ を $(2, 1, 1)$ のサイズのデータ $\boldsymbol{r}$ まで圧縮する．次に，圧縮されたデータ $\boldsymbol{r}$ を 1 つ目の値 r_1 と 2 つ目の値 r_2 に分ける．1 つ目の値に対しては 1st Deconv2D 層から 14th Conv2D 層までのデコーダ $\mathcal{F}_{\mathrm{dec1}}$ を，2 つ目の値に対しては 4th Deconv2D 層から 20th Conv2D 層までのデコーダ $\mathcal{F}_{\mathrm{dec2}}$ をそれぞれ用い，それぞれもとのサイズのデータ $\tilde{\boldsymbol{q}}_1$，$\tilde{\boldsymbol{q}}_2$ まで拡大する．最後に，値 r_1 に対応する出力 $\tilde{\boldsymbol{q}}_1$ と値 r_2 に対応する出力 $\tilde{\boldsymbol{q}}_2$ を足して，CNN

Tab. 1: Network structure of conventional CNN autoencoder

Layer	Data size	Activation
Input	$(384, 192, 2)$	
1st Conv2D $(3, 3, 16)$	$(384, 192, 16)$	tanh
1st MaxPooling	$(192, 96, 16)$	
2nd Conv2D $(3, 3, 8)$	$(192, 96, 8)$	tanh
2nd MaxPooling	$(96, 48, 8)$	
3rd Conv2D $(3, 3, 8)$	$(96, 48, 8)$	tanh
3rd MaxPooling	$(48, 24, 8)$	
4th Conv2D $(3, 3, 8)$	$(48, 24, 8)$	tanh
4th MaxPooling	$(24, 12, 8)$	
5th Conv2D $(3, 3, 4)$	$(24, 12, 4)$	tanh
5th MaxPooling	$(12, 6, 4)$	
6th Conv2D $(3, 3, 4)$	$(12, 6, 4)$	tanh
6th MaxPooling	$(6, 3, 4)$	
7th Conv2D $(3, 2, 2)$*	$(4, 2, 2)$	tanh
8th Conv2D $(3, 2, 1)$*	$(2, 1, 1)$	tanh
(Encoded values)	$(2, 1, 1)$	
1st Deconv2D $(3, 2, 2)$	$(4, 2, 2)$	tanh
2nd Deconv2D $(3, 2, 2)$	$(6, 3, 4)$	tanh
1st Upsampling	$(12, 6, 4)$	
9th Conv2D $(3, 3, 4)$	$(12, 6, 4)$	tanh
2nd Upsampling	$(24, 12, 4)$	
10th Conv2D $(3, 3, 8)$	$(24, 12, 8)$	tanh
3rd Upsampling	$(48, 24, 8)$	
11th Conv2D $(3, 3, 8)$	$(48, 24, 8)$	tanh
4th Upsampling	$(96, 48, 8)$	
12th Conv2D $(3, 3, 8)$	$(96, 48, 8)$	tanh
5th Upsampling	$(192, 96, 8)$	
13th Conv2D $(3, 3, 16)$	$(192, 96, 16)$	tanh
6th Upsampling	$(384, 192, 16)$	
14th Conv2D $(3, 3, 2)$ /Output	$(384, 192, 2)$	

Tab. 2: Network structure of Mode Decomposing CNN autoencoder

Layer	Data size	Layer	Data size
Input	$(384, 192, 2)$		
(Same structure as Table 1)			
(Encoded values)	$(2, 1, 1)$		
Value 1	$(1, 1, 1)$	Value 2	$(1, 1, 1)$
1st Deconv2D $(2, 1, 1)$	$(2, 1, 1)$	4th Deconv2D $(2, 1, 1)$	$(2, 1, 1)$
2nd Deconv2D $(3, 2, 2)$	$(4, 2, 2)$	5th Deconv2D $(3, 2, 2)$	$(4, 2, 2)$
3rd Deconv2D $(3, 2, 4)$	$(6, 3, 4)$	6th Deconv2D $(3, 2, 4)$	$(6, 3, 4)$
1st Upsampling	$(12, 6, 4)$	7th Upsampling	$(12, 6, 4)$
9th Conv2D $(3, 3, 4)$	$(12, 6, 4)$	15th Conv2D $(3, 3, 4)$	$(12, 6, 4)$
2nd Upsampling	$(24, 12, 4)$	8th Upsampling	$(24, 12, 4)$
10th Conv2D $(3, 3, 8)$	$(24, 12, 8)$	16th Conv2D $(3, 3, 8)$	$(24, 12, 8)$
3rd Upsampling	$(48, 24, 8)$	9th Upsampling	$(48, 24, 8)$
11th Conv2D $(3, 3, 8)$	$(48, 24, 8)$	17th Conv2D $(3, 3, 8)$	$(48, 24, 8)$
4th Upsampling	$(96, 48, 8)$	10th Upsampling	$(96, 48, 8)$
12th Conv2D $(3, 3, 8)$	$(96, 48, 8)$	18th Conv2D $(3, 3, 8)$	$(96, 48, 8)$
5th Upsampling	$(192, 96, 8)$	11th Upsampling	$(192, 96, 8)$
13th Conv2D $(3, 3, 16)$	$(192, 96, 16)$	19th Conv2D $(3, 3, 16)$	$(192, 96, 16)$
6th Upsampling	$(384, 192, 16)$	12th Upsampling	$(384, 192, 16)$
14th Conv2D $(3, 3, 2)$ / Output 1	$(384, 192, 2)$	20th Conv2D $(3, 3, 2)$ / Output 2	$(384, 192, 2)$
Add (Output)	$(384, 192, 2)$		

の出力とする．まとめると，

$$\boldsymbol{r} = \mathcal{F}_{\mathrm{enc}}(\boldsymbol{q}) \in \mathbb{R}^2 \tag{5}$$

$$\tilde{\boldsymbol{q}}_1 = \mathcal{F}_{\mathrm{dec1}}(r_1) \in \mathbb{R}^m \tag{6}$$

$$\tilde{\boldsymbol{q}}_2 = \mathcal{F}_{\mathrm{dec2}}(r_2) \in \mathbb{R}^m \tag{7}$$

$$\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{q}}_1 + \tilde{\boldsymbol{q}}_2 \in \mathbb{R}^m \tag{8}$$

となる．この構造の CNN では，圧縮された値に対して $\tilde{\boldsymbol{q}}_1$, $\tilde{\boldsymbol{q}}_2$ を可視化することで，それぞれのモードを解析することができる．

CNN の訓練の際，損失関数は平均二乗誤差とし，学習率の決定には Adam[10] を用いる．また，ネットワークの過学習を防ぐために，Early Stopping[11] を導入する．訓練データのうち，ランダムに選んだ 70%のデータを訓練データとして，残り 30%を検証用データとして用いる．

2.3 Sparse Identification of Nonlinear Dynamical systems (SINDy)

SINDy[7] は，時系列データからその運動を支配する非線形方程式を得る手法である．一般的な動的システムは

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{9}$$

という常微分方程式で表される．ここで，$\boldsymbol{x}(t)$ は時間 t における系の状態を表す．以下 $\boldsymbol{x}(t) = (x(t), y(t))$ と仮定する．$\boldsymbol{x}$ についての時間履歴を集め，

$$\boldsymbol{X} = \begin{pmatrix} x(t_1) & y(t_1) \\ x(t_2) & y(t_2) \\ \vdots & \vdots \\ x(t_m) & y(t_m) \end{pmatrix} \tag{10}$$

で表される行列 $\boldsymbol{X}$ を構成する．また，同様に $x(t)$ を時間微分した値 $\dot{x}(t)$ を集め，

$$\dot{\boldsymbol{X}} = \begin{pmatrix} \dot{x}(t_1) & \dot{y}(t_1) \\ \dot{x}(t_2) & \dot{y}(t_2) \\ \vdots & \vdots \\ \dot{x}(t_m) & \dot{y}(t_m) \end{pmatrix} \tag{11}$$

で表される行列 $\dot{\boldsymbol{X}}$ を計算する．次に，$\boldsymbol{X}$ の非線形関数で構成される Library 行列 Θ

$$\Theta(\boldsymbol{X}) = \begin{pmatrix} | & | & | & | & | & | & & | \\ 1 & \boldsymbol{x} & \boldsymbol{y} & \boldsymbol{x^2} & \boldsymbol{xy} & \boldsymbol{y^2} & \cdots & \boldsymbol{y^5} \\ | & | & | & | & | & | & & | \end{pmatrix} \tag{12}$$

を構成する．この $\Theta(\boldsymbol{X})$ を用いて，

$$\dot{\boldsymbol{X}}(t) = \Theta(\boldsymbol{X})\Xi \tag{13}$$

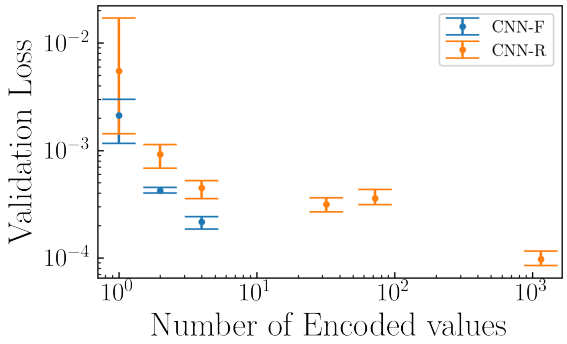

Fig. 3: Relationship between number of encoded values and validation loss.

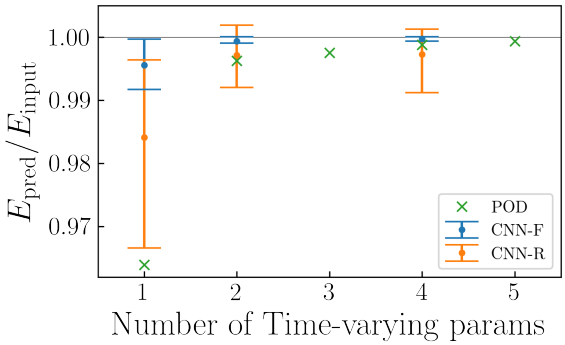

Fig. 4: Relationship between Number of time-varying values and Energy reconstruction rate.

で表される係数ベクトル Ξ,

$$\Xi = \begin{pmatrix} \xi_{(x,\,1)} & \xi_{(y,\,1)} \\ \xi_{(x,\,2)} & \xi_{(y,\,2)} \\ \vdots & \vdots \\ \xi_{(x,\,l)} & \xi_{(y,\,l)} \end{pmatrix} \tag{14}$$

を回帰分析により決定する．ここで，l は Library 行列の列の数である．この係数ベクトルが求まれば，Eq. (13) より，常微分方程式は

$$\frac{dx}{dt} = \Theta(x)\xi_x \tag{15}$$

$$\frac{dy}{dt} = \Theta(y)\xi_y \tag{16}$$

と求まる．この Ξ を求める際，SINDy では疎な係数行列を得られる回帰分析を用いることにより，常微分方程式に寄与度の大きい成分を特定する．

本研究では，CNN エンコーダで圧縮された 2 値に対し，SINDy を行いその常微分方程式を得た．次に，常微分方程式を数値積分して 2 値の時間発展を計算した．さらに，これらを CNN デコーダに与え，流れ場の時系列データを得た．

3. 結果および考察

3.1 CNN オートエンコーダによる流れ場の再現

まず，CNN の圧縮率の決定のために，様々な圧縮率の CNN を作成し，評価を行った．従来型の CNN オートエンコーダの構造で，圧縮するサイズを $(24, 12, 4)$，$(6, 3, 4)$，$(4, 2, 4)$，$(2, 1, 2)$，$(2, 1, 1)$，$(1, 1, 1)$ と変え，入出力を速度場の値とした CNN (C-CNN-R) を作成した．また，圧縮するサイズを $(2, 1, 2)$，$(2, 1, 1)$，$(1, 1, 1)$ とし，入出力を速度場の変動成分のみとした CNN (C-CNN-F) も作成した．以上のケースに対し，5 回同一構成の CNN を作成した．

CNN の圧縮された値の数と Validation Loss の関係を Fig. 3 に示す．図より，圧縮された値の数が 4 つ以上では Loss はほぼ変わらず，場を 2 値に圧縮する場合でも Loss はほぼ 1×10^{-4} のオーダになることが確認された．入出力を変動成分にしたときは，主流方向速度 u の値の平均値が小さくなるので Loss が小さくなっている．

POD との比較のため，CNN のエネルギー再現率の平均値を計算した．これは，CNN において，入力した流れ場の持つエネルギー E_{input} のうち復元された流れ場のエネルギー E_{pred} の割合を示している．なおエネルギーは入力データの領域 V に対し

$$E = \int_V (u^2 + v^2) dV \tag{17}$$

より計算した．CNN と POD のエネルギー再現率 $E_{\mathrm{pred}}/E_{\mathrm{input}}$ は Fig. 4 のようになった．なお，図において，時間的に変化する値の数とは，POD においては展開係数の数，CNN においては符号化された値の数である．時間変化値の数が 1 のときは，CNN は POD よりエネルギーを再現できることが確認された．同様に 2，4 のときは入出力を変動成分にした CNN のみ POD よりよい結果を示すことが確認された．また，2 つの CNN で比べると，入出力を変動にしたときのほうがエネルギー再現率が高い．これは，エネルギー全体のうち平均成分が 92%以上を占めているからである．さらに，CNN において時間変化値の数が 2，4 で大きな差がないことも確認された．このことから，以降符号化された値を $(2, 1, 1)$ と固定して CNN の作成を行った．

符号化された値を $(2, 1, 1)$ として Table 2 のような MD-CNN について，訓練データが流れ場の生の値の CNN (MD-CNN-R) と変動成分の CNN をそれぞれ 5 回作成した．時間的に変化する値の数が 2 つの場合について，POD と CNN のエネルギー再現率を比較すると Fig. 5 のようになった．C-CNN と比べ，MD-CNN はネットワークの構造が複雑で，訓練されるパラメータの数が多いため，エネルギー再現率が若干高くなっている．訓練データが生の値の場合，エネルギー再現率のばらつきが大きくなっている．C-CNN-F や MD-CNN-F ではエネルギー再現率は 99.8%以上を超えており，POD より高い割合でエネルギーを再現することができている．

CNN-F および MD-CNN-F に流れ場を入力して出力された結果を Figs. 6-7 に示す．図より，どちらも入力が再現されている様子が確認できる．CNN を通して出力される場は，入力と比べると場が粗くなっている様子が確認できる．この原因として，CNN において Upsampling でデータを拡大していることと，損失関数に平均二乗誤差を用いていることが挙げられる．

MD-CNN では，モードごとの場を見ることができる．Figures 6-7 の (a) を入力したときに出力される場を Fig. 8 に示す．同様に，POD でのそれぞれの場を Fig. 9 に示す．MD-CNN のモードと POD のモードは似たような傾向を示しているが，MD-CNN の方が流れの細かい構造を捉えている様子が確認できる．

3.2 モード時間発展方程式の導出

3.2.1 C-CNN-F と SINDy を組み合わせた結果

C-CNN-F を用いて，流れ場の時系列データを圧縮したとき，場は 2 値で表される．この 2 値を X，Y とすると，その時間履歴は Fig. 10 のようになった．さらに，二次中心差分を用いて X，Y の時間微分値 $\dot{X}$，$\dot{Y}$ を計算した．

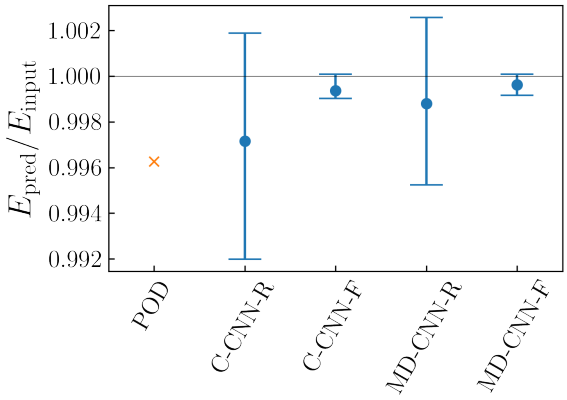

Fig. 5: Energy reconstruction rate for four types of CNN and POD.

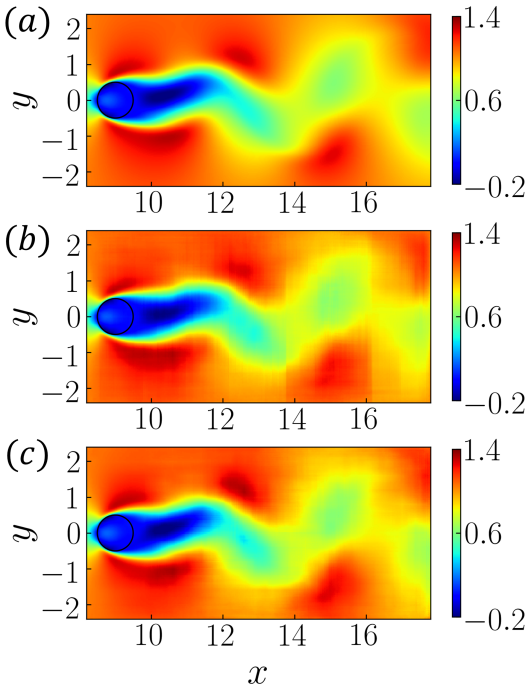

Fig. 6: The color contour of stream-wise velocity u; (a) Input (DNS), (b) Decoded field by C-CNN-F, (c) Decoded field by MD-CNN-F

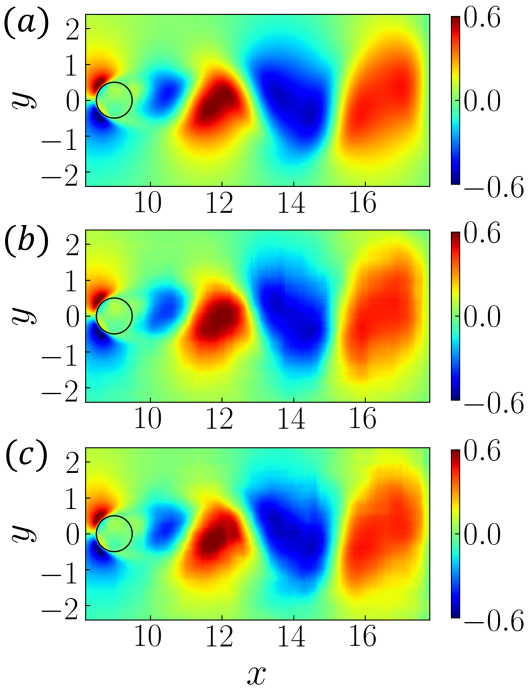

Fig. 7: The color contour of wall-normal velocity v; (a) Input (DNS), (b) Decoded field by C-CNN-F, (c) Decoded field by MD-CNN-F

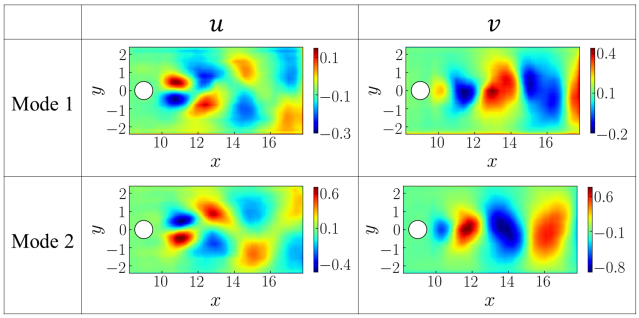

Fig. 8: Each modes of MD-CNN-F

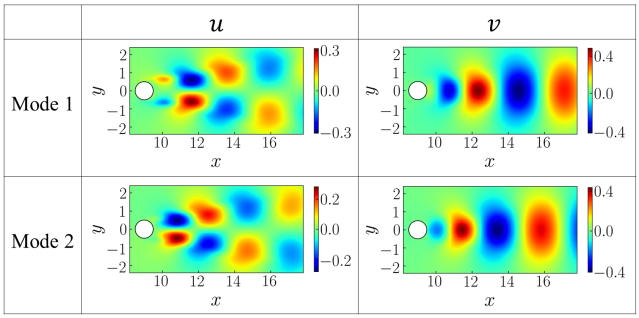

Fig. 9: Each modes of POD

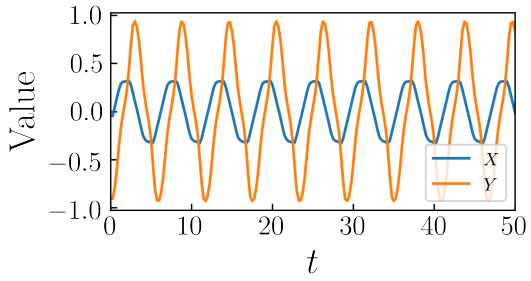

Fig. 10: C-CNN-F encoded values.

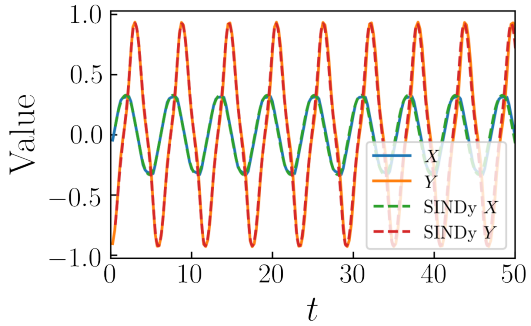

Fig. 11: Numerical integration of Eqs. (18)-(19).

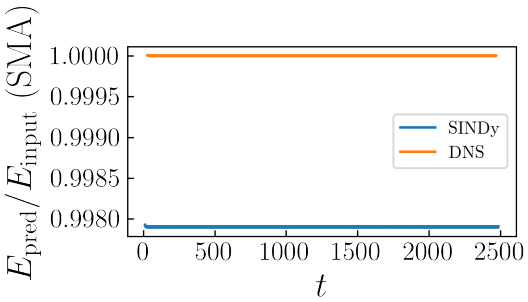

Fig. 12: Energy reconstruction rate of SINDy & C-C-NN-F decoded field (Simple Moving Average).

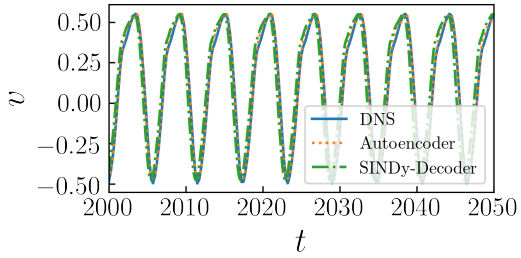

Fig. 13: Time history of wall-normal velocity on one point in the wake (SINDy & C-CNN-F).

次に，Eq. (12) のようなライブラリ行列を計算し，Eq. (13) で表される係数行列 Ξ を回帰的に求めた．なお，係数行列を求める際は LASSO を用い，解のスパース度を支配する L1 正規化項の係数 α は $\alpha = 1 \times 10^{-4}$ とした．係数行列から，圧縮場の支配方程式は，

$$\frac{dx}{dt} = -0.0056 + 0.2848x - 0.5101y + 0.5281xy^2 - 0.0120y^3 + 0.0112y^4 - 0.8732xy^4 + 0.1740y^5 \tag{18}$$

$$\frac{dy}{dt} = -0.0038 + 1.8696x - 0.2617y - 0.0130y^2 + 9.2229xy^2 - 0.6550y^3 + 0.4130xy^3 - 0.0011y^4 - 7.0571xy^4 + 0.3785y^5 \tag{19}$$

となった．この微分方程式を 4 次のルンゲ・クッタ法を用いて積分すると，Fig. 11 のようになり，良好な一致を確認できた．

Figure 12 に，復元された流れ場に関するエネルギー再現率 $E_{\mathrm{pred}}/E_{\mathrm{input}}$ を示す．図より，SINDy と CNN デコーダを組み合わせて復元された流れ場はエネルギー的に定常な状態になることが確認された．この CNN 自体のエネルギー再現率は 0.9990，SINDy と CNN デコーダを組み合わせたエネルギー再現率は 0.9979 なので，SINDy 自体にもエネルギー的誤差が生じ，その再現率は 0.9989 であることがわかった．しかし，これらの値は，いずれも POD のエネルギー再現率 0.9963 より高くなっており，エネルギーの点からは POD より良く流れ場が再現できていると言える．

後流の $(x, y) = (12, -0.5)$ の点における主流垂直方向速度 v は Fig. 13 のようになり，時間が十分経過した後でも，流れ場をよく再現していることが確認された．この点の主流垂直方向速度 v の時間履歴から求めたストローハル数は 0.1721 となり，DNS のストローハル数 0.1716 とよく一致した．

3.2.2 MD-CNN-F と SINDy を組み合わせた結果

前節同様に MD-CNN-F のエンコーダで圧縮された 2 値を X，Y とすると，その時間履歴は Fig. 14 のようになった．

次に，二次中心差分から $\dot{X}$，$\dot{Y}$，さらにライブラリ行列を計算し，係数行列 Ξ を求めた．なお，LASSO におけるパラメータ $\alpha = 7.5 \times 10^{-4}$ とした．係数行列から，圧縮場の支配方程式は，

$$\frac{dx}{dt} = 0.0011 - 1.8340x - 1.0763y - 0.0138x^2 - 0.1276x^3 - 0.2938y^3 - 0.0086x^4 + 3.0670x^3y^2 - 1.8249x^2y^3 \tag{20}$$

$$\frac{dy}{dt} = -0.0008 + 2.0368x + 1.8343y - 0.0074y^4 - 0.4967x^5 + 1.8045xy^4 - 0.4307y^5 \tag{21}$$

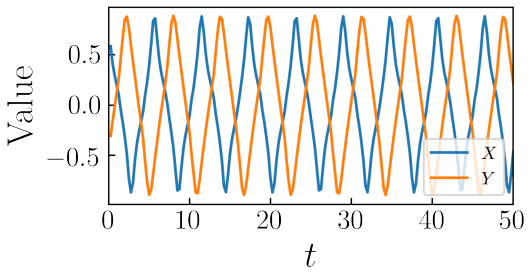

Fig. 14: MD-CNN-F encoded values.

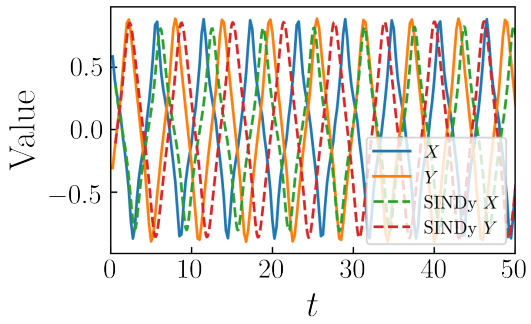

Fig. 15: Numerical integration of Eqs. (20)-(21).

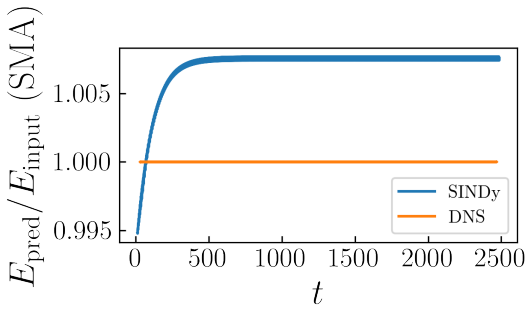

Fig. 16: Energy reconstruction rate of SINDy & MD-CNN-F decoded field (Simple Moving Average).

となった．この微分方程式を積分すると，Fig. 15 のようになった．図より，もとの圧縮場と比べると周期がずれることが確認された．圧縮場の様子が三角波のようになっているため，SINDy では誤差の大きい方程式しか得られないからであると考えられる．

Figure 16 に，復元された流れ場に関するエネルギー再現率 $E_{\mathrm{pred}}/E_{\mathrm{input}}$ を示す．図より，流れ場のエネルギーは最初上昇し，しばらくすると定常になることが確認された．この CNN 自体のエネルギー再現率は 0.9996，SINDy と CNN デコーダを組み合わせたエネルギー再現率は 1.0075 なので，SINDy 自体の再現率は 1.0079 であった．

後流の $(x, y) = (12, -0.5)$ の点における主流垂直方向速度 v は Fig. 17 のようになり，位相はずれているが振幅は減衰せず，流れ場をよく再現していることが確認された．また，ストローハル数は 0.1781 となり，DNS のストローハル数 0.1716 より若干大きい値となった．

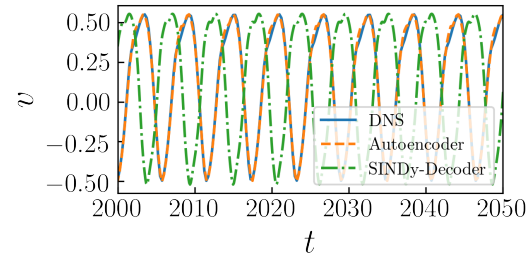

Fig. 17: Time history of wall-normal velocity on one point in the wake (SINDy & MD-CNN-F decoded field).

4. 結論

CNN オートエンコーダと SINDy を組み合わせ，流れの縮約モデルの新しいフレームワークを提案した．$\mathrm{Re}_D = 100$ の円柱周り流れに関して，流れ場を CNN を用いて圧縮すると，2 値まで圧縮しても流れのエネルギーの 99.9%以上を再現できることが確認された．これは POD の 99.6%より大きい値であり，CNN を用いることにより，高い再現率を実現できることが確認された．また，CNN オートエンコーダを通した流れ場は，元のものに比べると粗くなることが確認された．

次に，CNN エンコーダを用いて得られた圧縮場に関して SINDy を行い，微分方程式を得た．微分方程式を積分した値を CNN デコーダに与えると，流れが再現できることが確認された．積分した値の誤差により，エネルギー再現率に誤差が生じたり，復元された流れ場の周期が元のものと異なることがあった．CNN オートエンコーダと SINDy のそれぞれのエネルギー再現率は，ケースごとに異なるが，CNN に関しては構造や入出力を工夫することでその誤差を小さくできる．そのため，SINDy による誤差をいかに小さくするかが重要となる．

SINDy を行って得られた方程式の解釈や，誤差が小さくなるように SINDy を行う方法を解明すること，また他の流れ場にこの手法を適用することなどが今後の課題として挙げられる．

参考文献

(1) 平邦彦, “固有直交分解による流体解析: 1. 基礎,” ながれ, 30 (2011), pp. 115-127.

(2) Bergmann, M., Cordier, L., and Brancher, J. P., “Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model,” *Phys. Fluids*, 17 (2005), 097101.

(3) Alfonsi, G., and Primavera, L., “The structure of turbulent boundary layers in the wall region of plane channel flow,” *Proc. R. Soc. London Ser. A*, 463 (2007), 2078

(4) Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” *Adv. Neural Inf. Process. Syst.*, 25 (2012), pp. 1097-1105.

(5) Hinton, G. E., and Salakhutdinov, R. R., “Reducing the dimensionality of data with neural networks,” *Science*, 313 (2006), pp. 504-507.

(6) 尾亦範泰, 白山晋, “深層学習を用いた低次元表現による非定常流れ場の比較法,” ながれ, 36 (2017), pp. 363-369.

(7) Brunton, S. L., Proctor, J. L., and Kutz, J. N., “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” Proc. Natl. Acad. Sci. U. S. A., 113 (2016), pp. 3932-3937.

(8) Tibshirani, R., “Regression shrinkage and selection via the lasso,” *J. R. Stat. Soc. Ser. B*, 58 (1996), pp. 267-288.

(9) Kor, H., Badri Ghomizad, M., and Fukagata, K., “A unified interpolation stencil for ghost-cell immersed boundary method for flow around complex geometries,” *J. Fluid Sci. Technol.*, 12 (2017), JFST0011.

(10) Kingma, D. P., and Ba, J., “Adam: a method for stochastic optimization,” *arXiv preprint*, (2014), 1412.6980.

(11) Prechelt, L., “Automatic early stopping using cross validation: quantifying the criteria.,” *Neural Netw.*, 11 (2014), pp.761-767.