Nonlinear reduced-order modeling for three-dimensional turbulent flow by large-scale machine learning

Ando, Kazuto

Onishi, Keiji

Bale, Rahul

Kuroda, Akiyoshi

Tsubokura, Makoto

Graphical Abstract

Nonlinear Reduced-Order Modeling for Three-Dimensional Turbulent Flow by Large-Scale Machine Learning

Kazuto Ando, Keiji Onishi, Rahul Bale, Akiyoshi Kuroda, Makoto Tsubokura

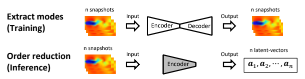
Nonlinear Reduced-Order Modeling for Three-Dimensional Turbulent Flow by Large-Scale Machine Learning

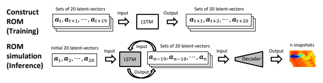
Distributed machine learning for flow field mode decomposition

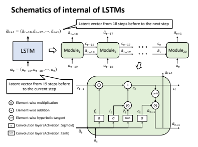
- Reduced-order modeling using 64 nonlinear modes obtained by the neural network reproduce three dimensional turbulent flow with Re=1000
- Implemented scalable distributed learning environment for the mode decomposition up to 25,250 CPUs (1.2M cores) on Fugaku
- Training loop indicates 7.8 PFLOPS and convolution kernel indicates 100 PFLOPS

Neural network-based nonlinear reduced-order model

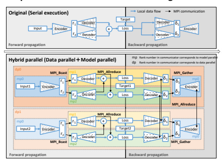
- Predict time-variation of latent vector using LSTM
- Neural network-based method shows higher reproduction accuracy than the method of combination of the proper orthogonal decomposition and Galerkin projection
- Reduced-order model reduces the number of calculation by 4 or 5 orders of magnitude relative to full-order model without significant precision loss



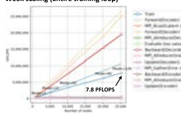

Construct ROM (Training)
Sets of 20 latent-vectors
$a_t, a_{t+1}, \cdots, a_{t+19}$
Input
LSTM
Output
Sets of 20 latent-vectors
$a_{t+1}, a_{t+2}, \cdots, a_{t+20}$
ROM simulation (Inference)
Initial 20 latent-vectors
$a_1, a_2, \cdots, a_{20}$
Input
LSTM
Output
Sets of 20 latent-vectors
$a_{n-19}, a_{n-18}, \cdots, a_n$
Input
Decoder
Output
n snapshots


Schematics of internal of LSTMs
Latent vector from 18 steps before to the next step
LSTM
Latent vector from 19 steps before to the current step
Element-wise multiplication
Element-wise addition
Element-wise hyperbolic tangent
Convolution layer (Activation: sigmoid)
Convolution layer (Activation: tanh)

Implementation of distributed learning

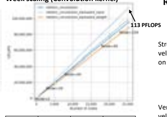


Week scaling (Entire training loop)



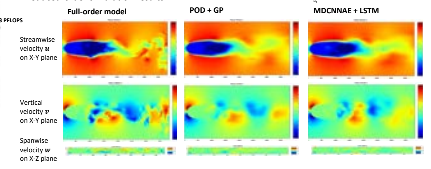
Mode	Node	PFLOPS	Scaling
2	750	0.31	100.0 %
20	5,250	1.81	81.5 %
40	10,250	3.43	79.0 %
80	20,250	6.28	73.3 %
100	25,250	7.80	72.9 %

Week scaling (Convolution kernel)



Mode	Node	PFLOPS	Scaling
2	750	3.35	100.0 %
20	5,250	23.62	100.7 %
40	10,250	46.14	100.7 %
80	20,250	91.14	100.7 %
100	25,250	113.75	100.8 %

Reduced-order simulation results



Highlights

Nonlinear Reduced-Order Modeling for Three-Dimensional Turbulent Flow by Large-Scale Machine Learning

Kazuto Ando, Keiji Onishi, Rahul Bale, Akiyoshi Kuroda, Makoto Tsubokura

- ROM using 64 nonlinear modes reproduce three dimensional turbulent flow with Re=1000
- Proposed method shows higher reproduction accuracy than combination of POD and GP
- Implemented distributed learning framework scales up to 25,250 CPUs on Fugaku
- ROM reduces the number of calculations by 4 or 5 orders of magnitude relative to FOM

Nonlinear Reduced-Order Modeling for Three-Dimensional Turbulent Flow by Large-Scale Machine Learning

Kazuto Ando[a,b,c], Keiji Onishi[b], Rahul Bale[b,c], Akiyoshi Kuroda[a], Makoto Tsubokura[b,c]

[a]*Operations and Computer Technologies Division, RIKEN Center for Computational Science, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, 650-0047, Hyogo, Japan*
[b]*Complex Phenomena Unified Simulation Research Team, RIKEN Center for Computational Science, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, 650-0047, Hyogo, Japan*
[c]*Department of Computational Science, Graduate School of System Informatics, Kobe University, 1-1, Rokkodai-cho, Nada-ku, Kobe, 657-8501, Hyogo, Japan*

Abstract

A large-scale machine learning-based nonlinear reduced-order modeling method was developed for a three-dimensional turbulent flow field (Re = 1000) using a neural-network with unsupervised learning. First, a mode decomposition method was applied to three-dimensional flow field data using a convolutional-autoencoder-like neural network. Then, a reduced-order model (ROM) was constructed using long short-term memory neural networks. Consequently, it was demonstrated that the time evolution of the turbulent three-dimensional flow field can be simulated at a significantly lower cost (approximately three orders of magnitude) without a major loss in accuracy. However, neural-network-based mode decomposition for the three-dimensional flow field requires a huge computational cost in terms of calculation and memory usage. Therefore, a distributed machine-learning method was implemented using a hybrid parallelism scheme tailored to the network structure. Thus, it was possible to decompose 1.7 million cells of the three-dimensional flow field data into 64 modes and reproduce those with sufficient accuracy. In this study, a uniform flow around a circular cylinder model was used as a test case. To validate the method, the reduction performance of the proposed mode decomposition method was compared with the proper orthogonal decomposition (POD) method. Furthermore, the target flow field was reproduced using ROM, and the reconstruction accuracy was evaluated in terms of various criteria compared with the accuracy based on POD in conjunction with Galerkin projection method.

Keywords: reduced-order model, turbulence, three-dimensional flow field, distributed machine learning, convolutional autoencoder (CAE), long short-term memory (LSTM) networks

1. Introduction

Numerical simulation, required in industrial applications, such as design optimization of automobile shapes and optimal control, must be executed repeatedly by changing the conditions, including the model shape and in-flow velocity. The cost of such a simulation is a major obstacle for industrial users considering the feasible size of the computational system and amount of time one user can exclusively devote for such a system. Therefore, it is necessary to construct a surrogate model that reduces the calculation cost while retaining the desired accuracy. There are two types of surrogate model. One involves the simplistic determination of the relation between the input (model shape, calculation condition, etc.) and output (drag, lift, etc.) using statistical methods or machine learning. The other is the reduced-order model (ROM), in which the dimensions of the data are reduced while retaining the physically important information, and the simulation is conducted in the reduced-order space. The former simply learns the relationship between input and output. The latter, however, learns the spatiotemporal features of the physical field; that is, the spatiotemporal relation, represented as the governing equations, is the learning target.

Various efforts have been made to construct ROM. First, the methods of dimension reduction are briefly introduced. The most commonly used method is proper orthogonal decomposition (POD). POD identifies a linear subspace in which most data are in the vicinity. A linear subspace is obtained by solving the eigenvalue problem. POD is discussed in detail in Section 2.1. Another classical method is multidimensional scaling. This method solves the eigenvalue problem, similar to POD; however, the target matrix is a more generalized distance matrix. If the matrix is constructed with Euclidean distance, this method is identical to POD. The distance matrix can be constructed from other sources than the Euclidean distance. If the kth neighboring point is used, the solution space can be extended to nonlinear space. This method is called “isomap” [1]. With this method, the distance between the most adjacent points is regarded as the Euclidean distance; however, the Riemannian manifold can be constructed by concatenating the local linear distant spaces.

Similar to isomap, several methods can address the nonlinear subspace. Locally linear embedding [2] is one such approach. In this method, the original data are first approximately reconstructed using the linear combination of the k-th neighboring points. In other words, the coefficients of the linear combination are evaluated to minimize the error between the original and reconstructed data. Next, while retaining the coefficients, the reduced-order data are approximated using the k-th neighboring points. Again, the minimization problem of the residual for the reduced data is solved. The Laplacian eigenmap [3] is a nonlinear method in which a weight matrix is constructed using the information of the neighboring points, and the dimensions are reduced while the information of the Laplacian matrix is preserved. Finally, another nonlinear method, t-SNE [4], which is an enhanced version of stochastic neighbor embedding (SNE) [5], symmetrizes the cost function and uses Student’s t-distribution when considering the distance in the reduced dimensional space.

Next, the methods for solving time marching in reduced-order space are introduced. The most typical method in conjunction with POD is Galerkin projection (GP) [6]. In this method, the solution is represented as a linear combination of the bases obtained by POD. The governing equation is projected into the space spanned by the bases, and the ROM is derived. Consequently, the time marching of the coefficient of linear combination of the bases is solved using ROM. The method of constructing ROM in this way using not only data but also information from the governing equations is called “intrusive reduced-order modeling.” This method is simple in terms of theory and methodology, thus is popular. However, this method does not provide sufficient prediction accuracy for an advection-dominant problem, that is, a case where nonlinearity appears strongly, such as the flow around a vehicle body. In other words, the solution space of such a problem exhibits a slowly decaying Kolmogorov n-width [7] with an increasing subspace dimensionality.

A method using a nonlinear manifold can be used to address this problem. The Galerkin projection can be extended to a nonlinear solution manifold. However, the computational cost of calculating the nonlinear term during ROM simulation is a challenge. To solve this problem, hyperreduction techniques, such as empirical interpolation method [8], discrete empirical interpolation method [9], Gauss–Newton with approximated tensors [10], and trajectory piecewise linearization [11], can be used. In addition, the spectral submanifold method has been attracting attention in recent years [12].

Because deep learning has attracted attention in the field of image recognition, dimension reduction and ROM methods using neural networks have been proposed as alternatives to the aforementioned methods. The dimension reduction method using an autoencoder can be considered as a nonlinear extension of POD [13]. The autoencoder consists of two parts: the encoder, which reduces the dimensions of the input data, and the decoder, which expands the dimensions of the input data. The weights of the encoder and decoder are updated to minimize the error between the input and output. The encoder and decoder are typically constructed as multiple layers of a convolutional neural network (CNN), which is called a convolutional neural network autoencoder (CNN-AE) [14]. Murata et al. proposed the mode-decomposing convolutional neural network autoencoder (MD-CNN-AE), which is a variant of CNN-AE in which the decoder section is branched for each mode, with the intention of visualizing the decomposed flow field, similar to POD [15]. In this study, this network was extended to a three-dimensional flow field. In recent years, there have also been examples of using a variational autoencoder (VAE) [16], where the network is trained such that the elements of the latent vector follow a normally distributed random variable [17, 18, 19, 20, 21].

However, there are many techniques of ROM prediction of time marching in the reduced-order space using a neural network. In most typical cases, long short-term memory (LSTM) networks are used [22]. This type of method is called "nonintrusive reduced-order modeling" [23] because it constructs the ROM only from data without using the information of the governing equations (e.g., Gaussian process regression [24] is a nonintrusive method that was widely used). Hasegawa et al. used a CNN-AE to reduce the dimensions and LSTM networks for the temporal prediction of latent vector (the variables represent the system state in the reduced-order space) to evaluate the robustness of the flow field prediction as the model shape changes [25]. In addition, Nakamura et al. proposed a method for arranging LSTMs with a different number of features in the hidden state in parallel to improve the accuracy of the time prediction [26]. Quilodrán et al. [27] proposed an LSTM-based method in conjunction with generative adversarial networks [28] to increase the prediction accuracy. Maulik et al. [29] solved the time integration with neural ordinary differential equations [30]. Kim et al. [31] and Lee et al.

[32] proposed a nonlinear alternative to the projection method, such as the Galerkin projection.

With regard to large-scale machine learning for mode decomposition, a distributed parallel machine-learning environment was implemented on the supercomputer Fugaku [33, 34]. The hybrid-parallel scheme, the combination of data parallelism that distributes the training data and model parallelism that distributes parts of the entire network structure were enhanced [35]. This implementation made it possible to decompose the three-dimensional flow field of 1.7 million cells into 100 modes using 25,250 nodes (1.2 million cores). In this study, the method was evaluated precisely in terms of the prediction accuracy for three-dimensional turbulent flow.

2. Methods

In this section, a high-precision flow simulation in which flow field snapshots are used as training data for machine learning is described. In addition, some mode decomposition methods to be evaluated in the following section are discussed.

2.1. Full-Order Model

To produce a reference flow field, a full-order model (FOM) simulation using CUBE [36] was conducted (that is, a high-precision three-dimensional flow simulation). CUBE is a unified simulation framework that is based on the building cube method (BCM) [37, 38] and immersed boundary method (IBM) [39, 40], encompassing an incompressible/compressible flow solver and a structural solver for computing fluid–solid interactions [41, 42]. In this study, an incompressible flow solver was used in which the governing equation is as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} + \mathbf{f}, \tag{1}$$

$$\rho\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where u, p, ρ, and μ are the flow velocity, pressure, density, and viscosity, respectively. The execution parameters of FOM are listed in Table 1. The computational domain is depicted the black-lined area in Figure 1, which consists of 28 million cells. In addition, the target region for machine learning is the area of 1.7 million cells enclosed by the red line in Figure. 1.

Table 1: Execution parameters of FOM.

Parameter	Value
Computational domain	$40D \times 20D \times 2.5D$[a]
Minimum cell size	$0.026D$[a]
Number of cells	$1,536 \times 768 \times 24$
Reynolds number	1,000
Time step size	5.0×10^{-3} s
Number of time steps	90,000
Output frequency	Every 5 steps (0.025 s)

[a] D(=1) is the circular cylinder diameter.

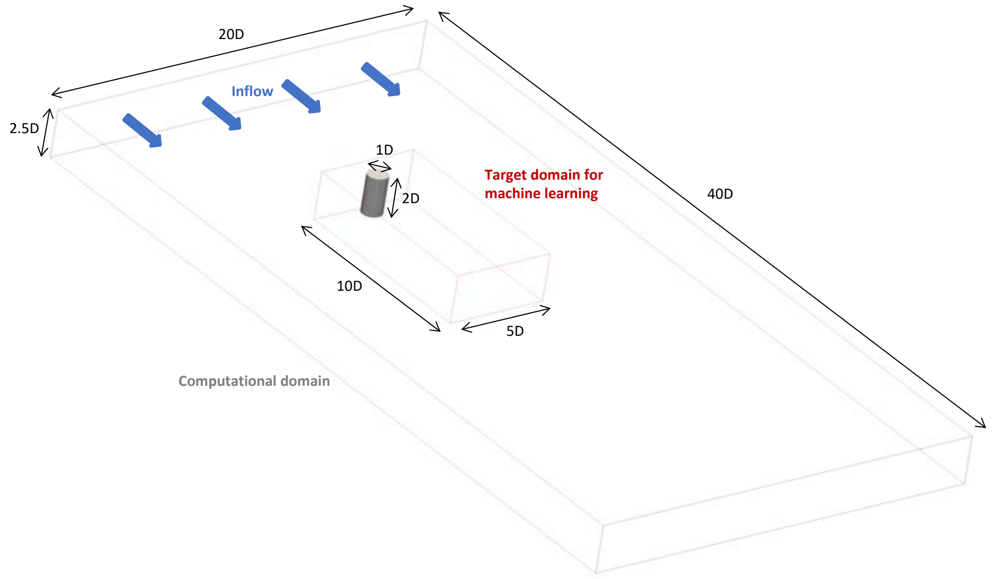


Figure 1: Computational domain of full-order model (Black lined box area) and target region of the machine learning (Red lined box area).

2.2. POD and Galerkin Projection

POD is a representative method for dimension reduction that assumes a linear solution manifold. The key idea of POD is to project high-dimensional data onto a lower-dimensional space. For example, in the case of three-dimensional data, the data are projected onto a two-dimensional plane (Fig. 2). In other words, the data dimensions are reduced from three to two.

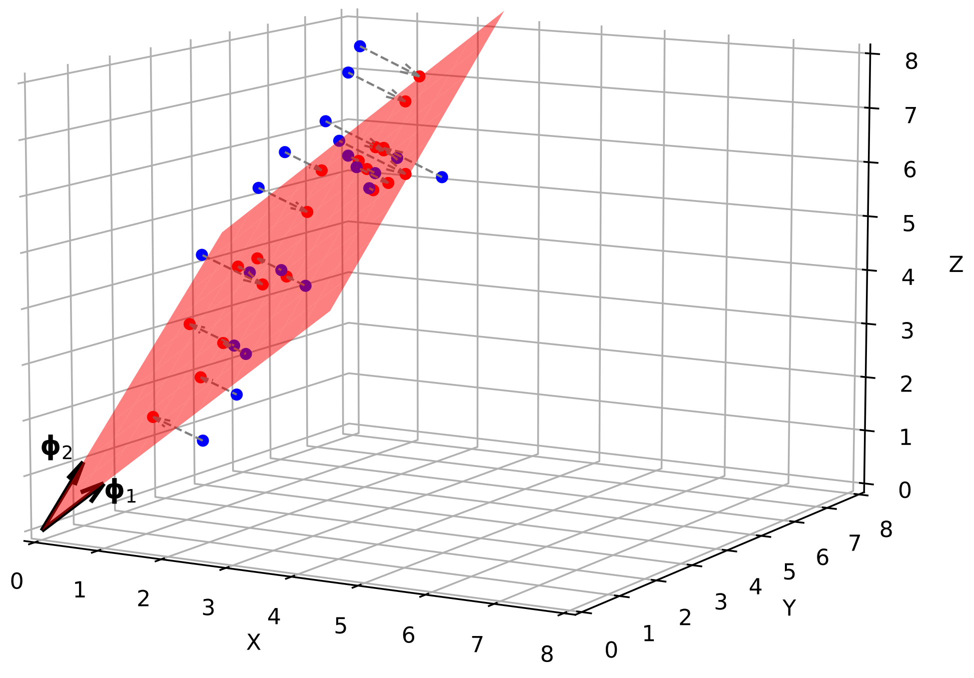


Figure 2: Dimension reduction into linear subspace. $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ represent POD bases. The red plane means the linear subspace spanned by $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$. The blue dots mean the original (three-dimensional) data and the red dots mean reduced-order (two-dimensional) data projected to the linear subspace.

In the following, POD is formulated (according to [43]) as

$$\mathbf{x}(t) = \mathbf{q}(\xi, t) - \bar{\mathbf{q}}(\xi) \in \mathbb{R}^n, \quad t = t_1, t_2, ..., t_m \tag{3}$$

where ξ, $\mathbf{q}(\xi, t)$, $\bar{\mathbf{q}}(\xi)$, and $\mathbf{x}(t)$ represent the spacial vector, the vector field, its time-averaged value, and the fluctuating value of the $\mathbf{q}(\xi, t)$, respectively.

Now, you can construct the matrix X with concatenating the time series of the $\mathbf{x}(t)$.

$$X = [\mathbf{x}(t_1)\ \mathbf{x}(t_2)\ ...\ \mathbf{x}(t_m)] \in \mathbb{R}^{n\times m} \tag{4}$$

Using the method of Lagrange multipliers, the k-th POD mode $\boldsymbol{\phi}_j$ is evaluated such that variance $\sum_{i=1}^{m}(\mathbf{x}(t_i)\cdot\boldsymbol{\phi}_j)^2$ is maximized subject to constraint to $\boldsymbol{\phi}_j^T\boldsymbol{\phi}_j = 1$.

$$\begin{aligned}
\boldsymbol{\phi}_j &= \underset{\boldsymbol{\phi}_j}{\arg\max}\, L(\boldsymbol{\phi}_j) && (5)\\
&= \underset{\boldsymbol{\phi}_j}{\arg\max} \sum_{i=1}^{m}(\mathbf{x}(t_i)\cdot\boldsymbol{\phi}_j)^2 - \lambda_j(\boldsymbol{\phi}_j^T\boldsymbol{\phi}_j) && (6)\\
&= \underset{\boldsymbol{\phi}_j}{\arg\max} \sum_{i=1}^{m}(\boldsymbol{\phi}_j^T\mathbf{x}(t_i)\mathbf{x}^T(t_i)\boldsymbol{\phi}_j) - \lambda_j(\boldsymbol{\phi}_j^T\boldsymbol{\phi}_j). && (7)
\end{aligned}$$

That is, $\boldsymbol{\phi}_j$ is found when the first derivative of $L(\boldsymbol{\phi}_j)$ is 0.

$$\begin{aligned}
\frac{\partial L(\boldsymbol{\phi}_j)}{\partial\boldsymbol{\phi}_j} &= 2\big(\sum_{i=1}^{m}\mathbf{x}(t_i)\mathbf{x}^T(t_i)\boldsymbol{\phi}_j - \lambda_j\boldsymbol{\phi}_j\big) && (8)\\
&= 2\big(XX^T\boldsymbol{\phi}_j - \lambda_j\boldsymbol{\phi}_j\big) && (9)\\
&= 0 && (10)
\end{aligned}$$

Finally, the following eigenvalue problem is derived:

$$XX^T\boldsymbol{\phi}_j = \lambda_j\boldsymbol{\phi}_j, \quad \boldsymbol{\phi}_j \in \mathbb{R}^n. \tag{11}$$

However, solving this problem requires considerable computational time; therefore, the size of the matrix in the following form is reduced in practice (called “snapshot POD”)

$$X^TX\boldsymbol{\psi}_j = \lambda_j\boldsymbol{\psi}_j, \quad \boldsymbol{\psi}_j \in \mathbb{R}^m \tag{12}$$

$$\boldsymbol{\phi}_j = X\boldsymbol{\psi}_j\frac{1}{\sqrt{\lambda_j}}, \quad \boldsymbol{\phi}_j \in \mathbb{R}^n \tag{13}$$

Once POD mode $\boldsymbol{\phi}_j$ is obtained, the ROM can be constructed using Galerkin projection. The flow field is represented using $\{\boldsymbol{\phi}_j\}_{j=1}^{r}$ as follows:

$$\mathbf{u}(\mathbf{x},t) = \sum_{j=0}^{r} a_j(t)\boldsymbol{\phi}_j(\mathbf{x}). \tag{14}$$

The incompressible Navier-Stokes equation is

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u}. \tag{15}$$

Substituting Eq. (14) into Eq. (15) and taking the dot product with $\boldsymbol{\phi}_i$ on both sides yields

$$\sum_{j=0}^{r}\frac{da}{dt}\langle\boldsymbol{\phi}_i,\boldsymbol{\phi}_j\rangle + \sum_{j=0}^{r}\sum_{k=0}^{r} a_j a_k\langle\boldsymbol{\phi}_i,(\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{\phi}_k\rangle = -\langle\boldsymbol{\phi}_i,\nabla p\rangle + \frac{1}{Re}\sum_{j=0}^{r} a_j\langle\boldsymbol{\phi}_i,\nabla^2\boldsymbol{\phi}_j\rangle. \tag{16}$$

From $\langle\boldsymbol{\phi}_i,\boldsymbol{\phi}_j\rangle = \delta_{ij}$ and $\langle\boldsymbol{\phi}_i,\nabla p\rangle = 0$, Eq. (16) can be written as

$$\sum_{j=0}^{r}\frac{da_i}{dt} = \sum_{j=0}^{r}\sum_{k=0}^{r} F_{ijk}a_j a_k + \sum_{j=0}^{r} G_{ij}a_j, \tag{17}$$

$$F_{jk} = -\langle\boldsymbol{\phi}_i,(\boldsymbol{\phi}_j\cdot\nabla)\boldsymbol{\phi}_k\rangle. \tag{18}$$

$$G_{ij} = \frac{1}{Re}\langle\boldsymbol{\phi}_i,\nabla^2\boldsymbol{\phi}_j\rangle. \tag{19}$$

Now, ROM of the incompressible Navier-Stokes equation by the Galerkin projection is derived. Here, F_{ijk} and G_{ij} are time-invariant, i.e., they can be calculated prior to the start of the simulation. Finally, the time evolution of the flow field can be simulated by solving r sets of ordinary differential equations.

2.3. Neural-Network-Based ROM

Next, the details of the neural network for mode decomposition, MD-CNN-AE, are introduced. Figure 3 shows a schematic of the network structure. The first half of the network, the encoder, inputs the high-precision flow field snapshots and gradually reduces the dimensions as the data pass through the layers, and it outputs the vector containing the reduced variables, the latent vector. The second half of the network, the decoder, branches for each mode for decomposition. Each branched network inputs each element of the latent vector. Then, the data dimension is expanded as it passes through the layers. Each branched network outputs a decomposed flow field for each mode. Finally, these decomposed flow fields are combined, and the flow field that reproduces the original flow field is output. These networks are trained to minimize the errors between the original flow field $\mathbf{x}(t)$ and reconstructed flow field $\sum_{j=1}^{r} \mathcal{F}_{dec,j}([\mathcal{F}_{enc}(\mathbf{x}(\mathbf{t}))]_j)$. This optimization problem is formulated as

$$\{\boldsymbol{\phi}_j\}_{j=1}^{r} = \underset{\{\tilde{\boldsymbol{\phi}}_j\}_{j=1}^{r}}{\arg\min} \int_{t_{min}}^{t_{max}} \|(\mathbf{x}(t) - \sum_{j=1}^{r} \mathcal{F}_{dec,j}([\mathcal{F}_{enc}(\mathbf{x}(t))]_j)\|^2 dt, \tag{20}$$

where $\mathcal{F}_{enc}$ is the encoder; $\mathcal{F}_{dec,j}$ is the decoder corresponding to the j-th mode, and $\{\boldsymbol{\phi}_j\}_{j=1}^{r}$ is the trained network weights. The detailed network structure is shown in Appendix A.1.

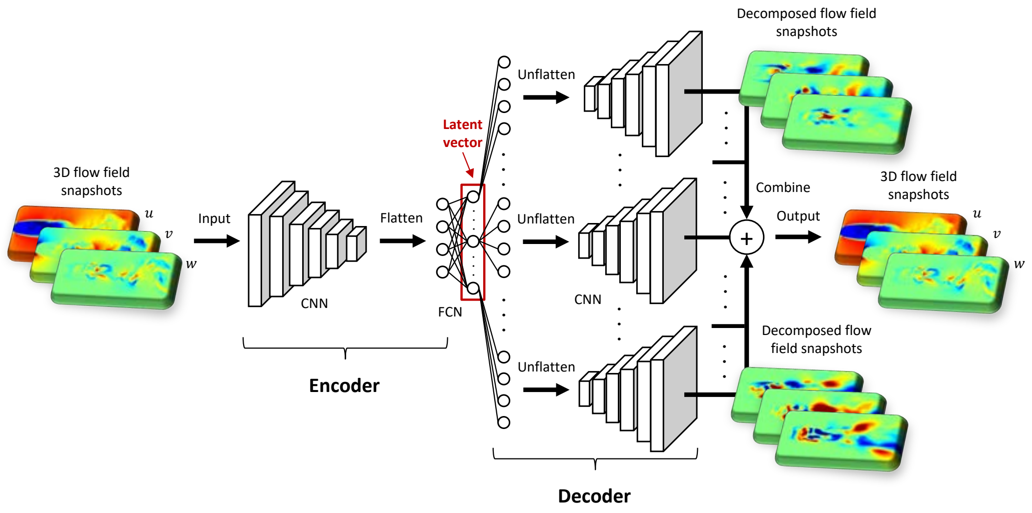


Figure 3: Schematics of network for three-dimensional mode decomposition.

In addition, a hybrid-parallel scheme was implemented, that is, the training data and network structure are both distributed to the other MPI processes, which do not share memory space, to execute large-scale machine learning for mode decomposition (Figure 4). For the latter parallelization, the encoder and multiple decoders are assigned to a single MPI process. This implementation can overcome the memory barrier and increase the number of decompositions to the limit of the number of nodes provided by the system. To conduct large-scale machine learning using this implementation, as many as 25,250 nodes of the Fugaku supercomputer (1.2 million cores) were used. The number of MPI processes to be used for data parallelization is fixed at 384 MPI processes, and the total number of processes is increased by a constant factor of 384 with a number of decompositions.

The number of data used for training is 10,000 with a time step size equal to 0.025 seconds. Of those 7,000 are for training and 3000 are for evaluation. Besides, the number of test data is 2,000. The global batch size is 768. Therefore, dividing by the number of 384 MPI processes regarding to data parallelism gives a local batch size of 2. This network trained 11,000 epochs using the Adam optimizer [44]. A more detailed explanation of hyperparameters for the network is shown in Appendix B.1.

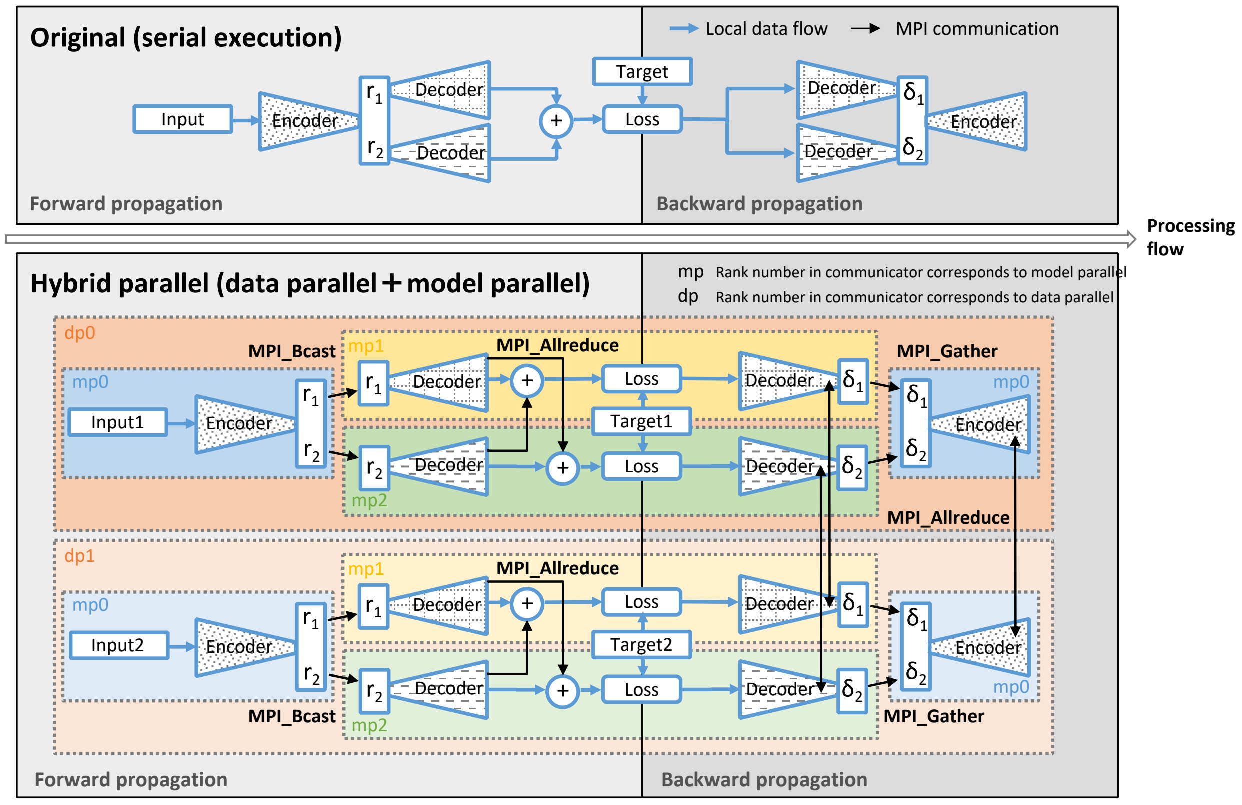
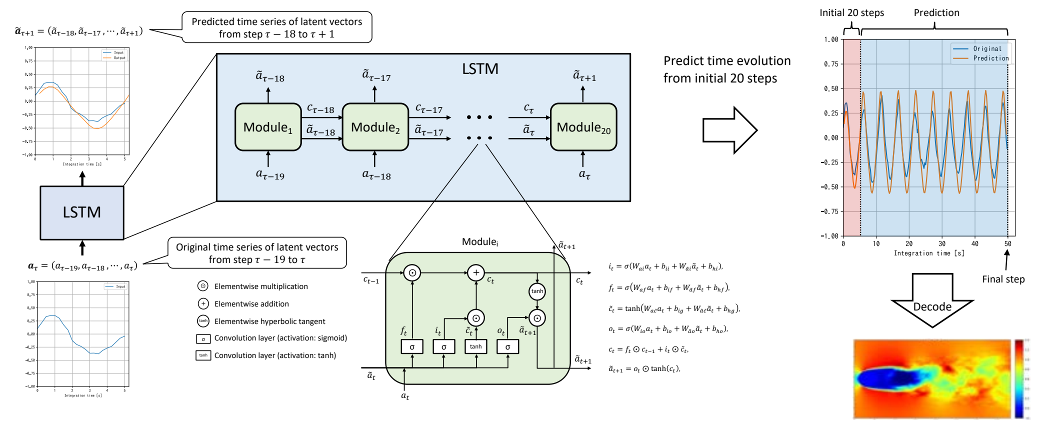


Figure 4: Processing flow of the hybrid parallelization scheme for mode decomposition. The top is the original sequential execution, and the bottom is the hybrid parallel execution. The blue and black line represent the local data flow and the MPI communication, respectively.

Once the machine learning for mode decomposition has been completed, the dimensions of the original flow field snapshots can be reduced. Therefore, the time series of latent vectors can be obtained. Next, another neural network, the LSTM network, is harnessed for learning the time series. LSTM network is a type of recursive neural network (RNN) that specializes in learning long-term dependency. Figure 5 shows the internal structure of the LSTM. LSTMs take the 20-time steps of the latent vectors to step τ and predict the unknown latent vector of step τ+1. Once the LSTM is trained, using only the initial 20-time steps of the latent vectors, it is possible to predict the following time steps up to the last step. Finally, the trained decoder network can decode the time series of latent vectors to the flow field snapshots. The corresponding procedure is formulated as follows:



Predicted time series of latent vectors from step τ − 18 to τ + 1
Original time series of latent vectors from step τ − 19 to τ
LSTM
Module₁
Module₂
Module₂₀
Moduleᵢ
Elementwise multiplication
Elementwise addition
Elementwise hyperbolic tangent
Convolution layer (activation: sigmoid)
Convolution layer (activation: tanh)
Predict time evolution from initial 20 steps
Initial 20 steps
Prediction
Original
Prediction
Integration time [s]
Final step
Decode

Figure 5: Schematics of network for learning time series of latent vector.

$$i_t = \sigma(W_{ai}a_t + b_{ii} + W_{\tilde{a}i}\tilde{a}_t + b_{hi}), \tag{21}$$

$$f_t = \sigma(W_{af}a_t + b_{if} + W_{\tilde{a}f}\tilde{a}_t + b_{hf}), \tag{22}$$

$$\tilde{c}_t = \tanh(W_{ac}a_t + b_{ig} + W_{\tilde{a}\tilde{c}}\tilde{a}_t + b_{hg}), \tag{23}$$

$$o_t = \sigma(W_{io}a_t + b_{io} + W_{\tilde{a}o}\tilde{a}_t + b_{ho}), \tag{24}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{25}$$

$$\tilde{a}_{t+1} = o_t \odot \tanh(c_t), \tag{26}$$

where a_t, W, b, i_t, f_t, $\tilde{c}_t$, o_t, c_t, and $\tilde{a}_{t+1}$ represent the latent vector of the current time-step, weight, bias, input gate, forget gate, candidate of the new cell state, output gate, new cell state, and predicted latent vector of the next time-step, respectively. The network inputs the latent vector for the past 20 steps and outputs that for the past 19 steps and the subsequent step. The entire network consists of a parallel arrangement of LSTM networks with a different numbers of features in the hidden state, as reported by Nakamura et al. [26]. The detailed network structure is shown in Appendix A.2.

The number of data used for training is 10,000 with a time step size equal to 0.25 seconds. Of those 7,000 are for training and 3000 are for evaluation. Besides, the number of test data is 2,000. This network trained 2,000 epochs using the Adam optimizer. A more detailed explanation of hyperparameters for the networks is shown in Appendix B.2.

3. Results and Discussion

In this section, the results of ROM simulations using the conventional (combining POD and GP) and proposed (combining MD-CNN-AE and LSTM) methods are presented, and the reproducibilities of both for the FOM flow field are compared.

3.1. Reduction Performance of Flow Field Structure

In this subsection, to evaluate the flow field dimension-reduction performance (excluding the time evolution part), the reduction performance of the only mode decomposition part, MD-CNN-AE, is evaluated.

Figure 6 shows the history of the validation error when training MD-CNN-AE. In this case, the maximum number of mode decompositions is up to 64 due to the restriction of computational resources (it requires 6,244 computational nodes $\times$ 150 hours). We conducted two modes of decomposition for the minimum case. However, there is no reproduction performance. Thus, we set 16 modes as the baseline number of decompositions. Before 6000 epochs, the error of the 64-mode training is larger than that of 16-modes training. This means that a greater number of training iterations is required when the number of mode decompositions increases.

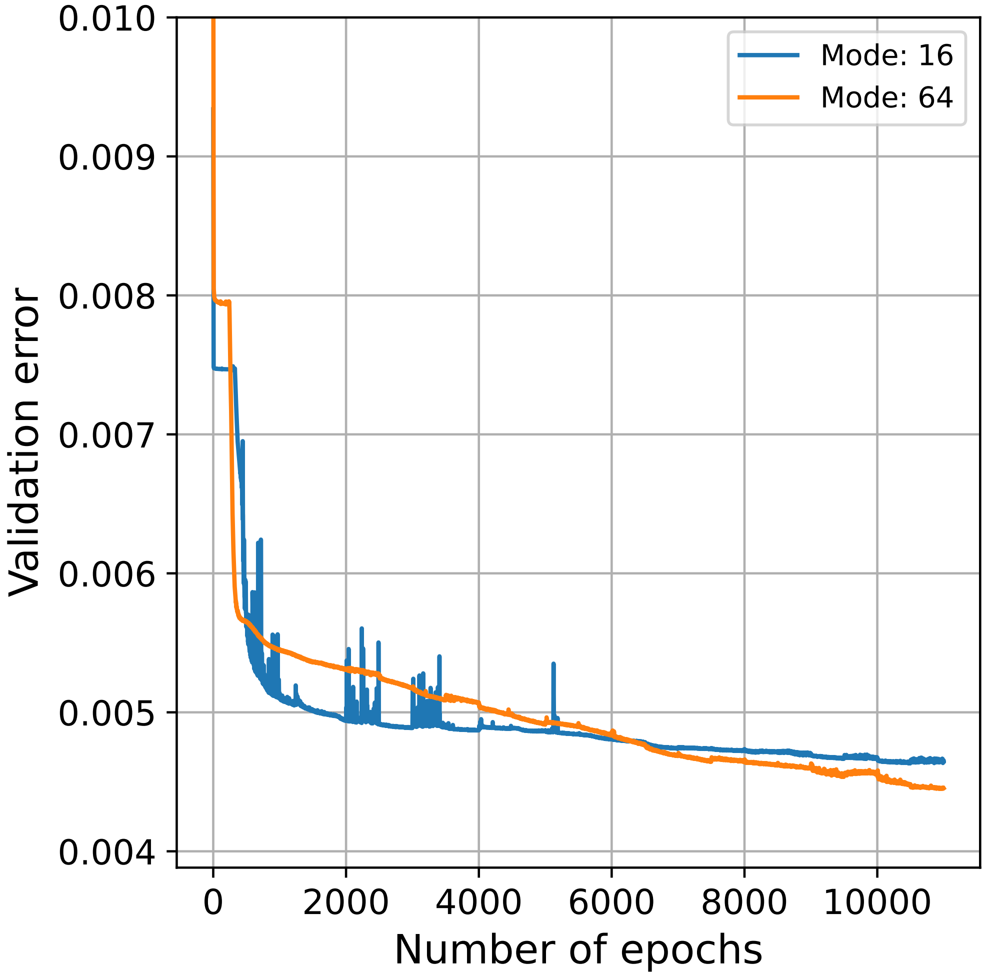


Figure 6: History of validation error of MD-CNN-AE training: the blue and orange lines indicate the decomposition into 16 and 64 modes.

To evaluate the reduction performance of the instantaneous flow field, POD decomposition was applied to the original flow field (full-order model) and reconstructed flow field with the MD-CNN-AE, and the energy distribution of each POD mode was evaluated (Figure 7). When using 16 modes, if the number of epochs is larger, the gradient of the energy retained by each POD mode is more moderate. The same trend can be seen with the 64 modes, but the gradient is gentler than that with the 16 modes.

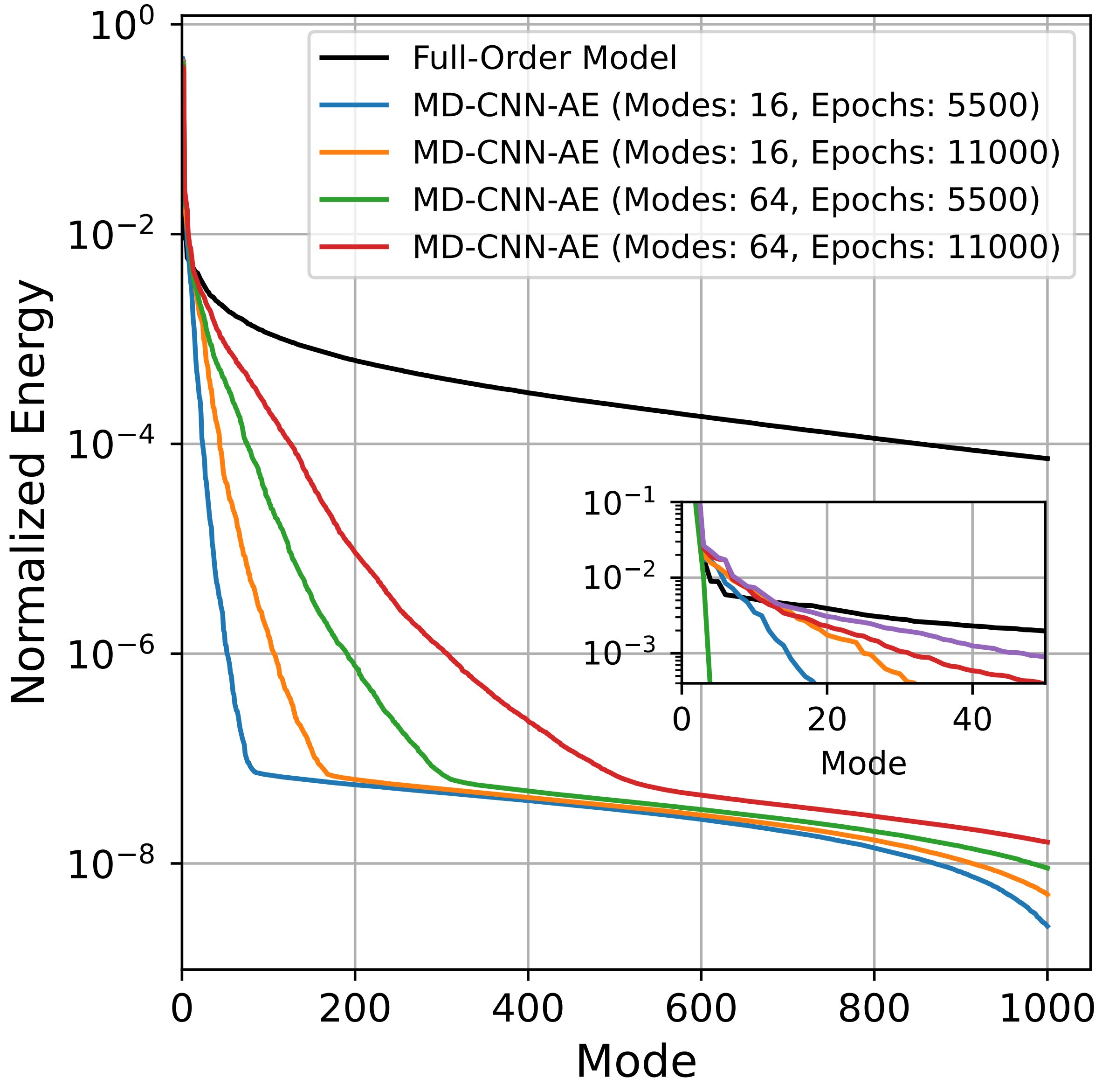


Figure 7: Energy distribution corresponding to each POD mode: the black line indicates the Full-order model. The blue and orange lines indicate the 16-modes decomposition using MD-CNN-AE trained with 5,500 and 11,000 epochs. The green and red lines indicate the 64-modes decomposition using MD-CNN-AE trained with 5,500 and 11,000 epochs.

The time series of the latent vector elements with the 16-mode decomposition are shown in Figure 8 (only four elements out of 16 are extracted). It can be seen that the value of some elements can change significantly as the number of epochs increases.

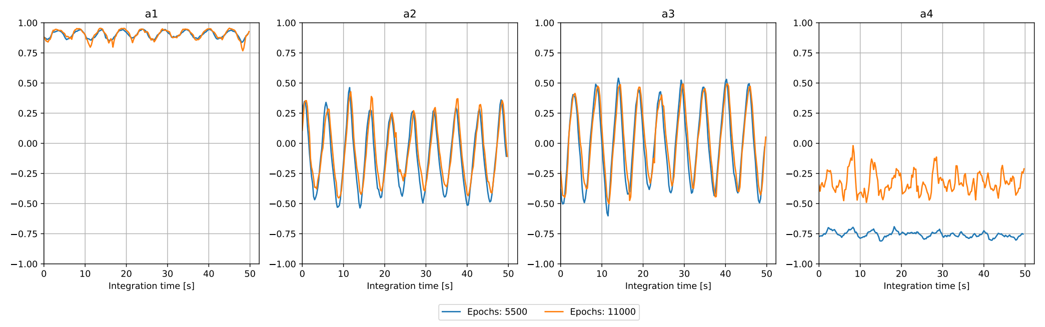


Figure 8: Time series of latent vector elements: the blue and orange lines indicate the results of MD-CNN-AE trained with 5,500 and 11,000 epochs.

Next, the decomposed flow field using POD and MD-CNN-AE is shown. The original flow field created with the FOM simulation was decomposed using POD (Figure 9(a)). Each decomposed flow field had a symmetric structure in which the two modes were paired. However, the MD-CNN-AE decomposition applied to the original flow field (Figure 9(b)) showed a more complicated flow structure in each mode, which was not symmetrical. Furthermore, only mode 3 of the MD-CNN-AE decomposition was decomposed by POD (Figure 10). It can be seen that one MD-CNN-AE mode contains two or more POD modes.

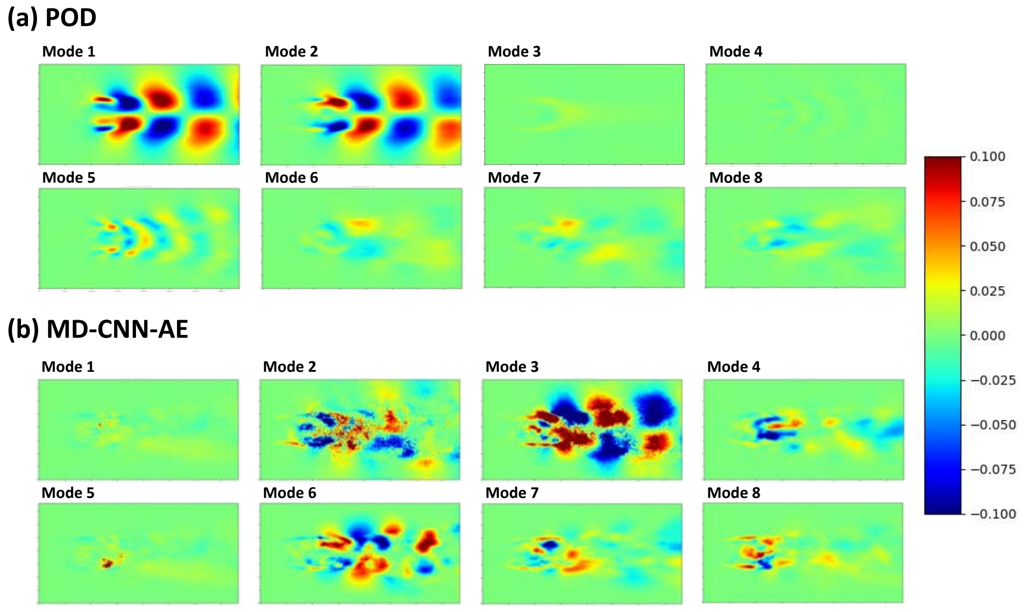


Figure 9: Decomposed flow field by applying (a)POD and (b)MD-CNN-AE to FOM simulation result.

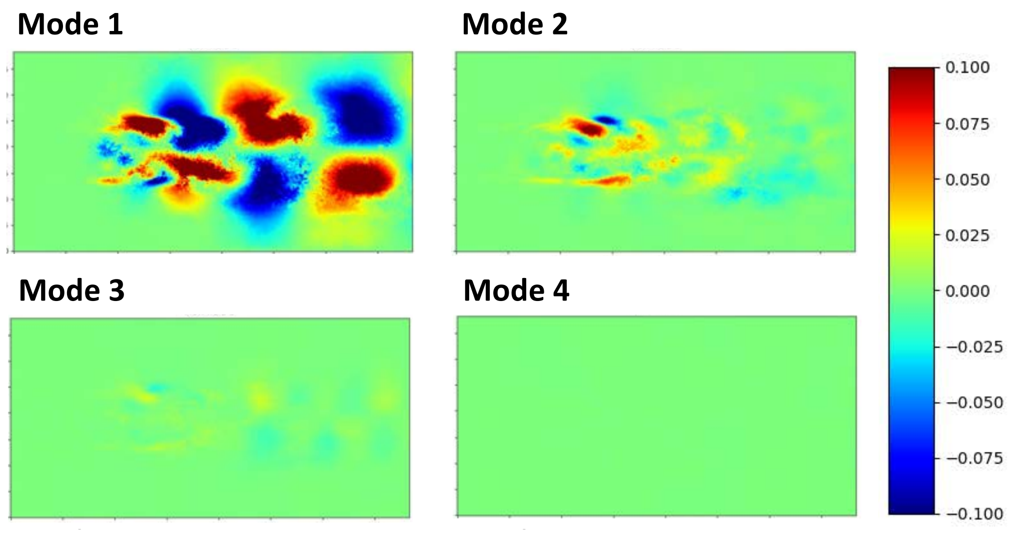


Figure 10: Decomposed flow field by applying POD to mode 3 of MD-CNN-AE result.

3.2. Prediction by ROM Simulation

The prediction performance of ROM simulation was evaluated. The time variation of the element of the latent vector of the training data and ROM simulation results using LSTM are shown in Figure 11. The LSTM predicts all the time steps from only the initial 20 steps. The oscillation phase of the ROM prediction almost matched with that of the test data. In addition, the amplitude of the prediction roughly matched in each mode (See Appendix C for all of the latent vector elements by 16-modes and 64-modes decomposition). The time series of latent vector elements with low-frequency oscillation shows deviations from FOM results. Although the hyper-parameter optimization for LSTM is necessary to resolve low-frequency oscillations, this topic lies outside the scope of this paper.

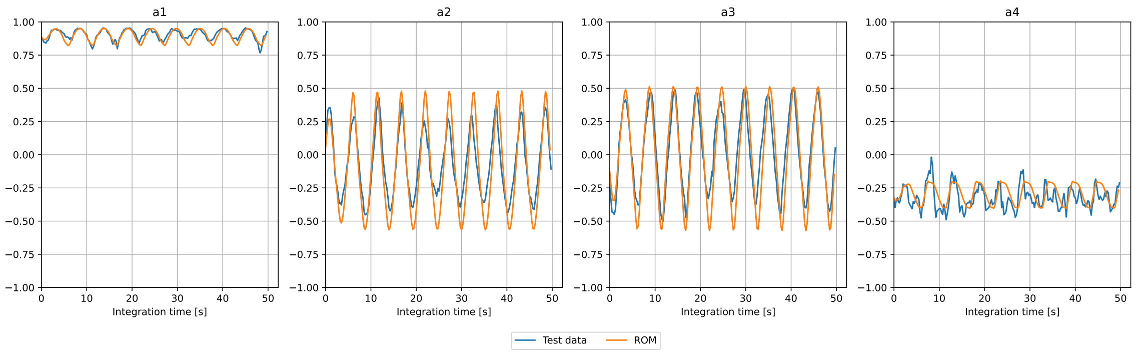


Figure 11: Time series of latent vector elements (only the first four modes are extracted): the blue and orange lines indicate, respectively the test data and the prediction with LSTM derived from first 20 time steps.

Figures 12–14 show comparisons of the flow fields between the FOM results and ROM predictions with 16-mode decomposition. The proposed method, which enhanced the MD-CNN-AE and LSTM (MD-CNN-AE + LSTM), reproduced a more complex flow field than the method combining POD and GP (POD + GP). Moreover, the oscillation phase predicted by MD-CNN-AE + LSTM is better matched than that predicted by POD + GP. The reason MD-CNN-AE + LSTM works better than POD + GP is that the decomposed mode extracted by MD-CNN-AE corresponds to the multiple POD-decomposed modes [15]. Therefore, when the number of decomposition modes is fixed, the former shows higher reproducibility.

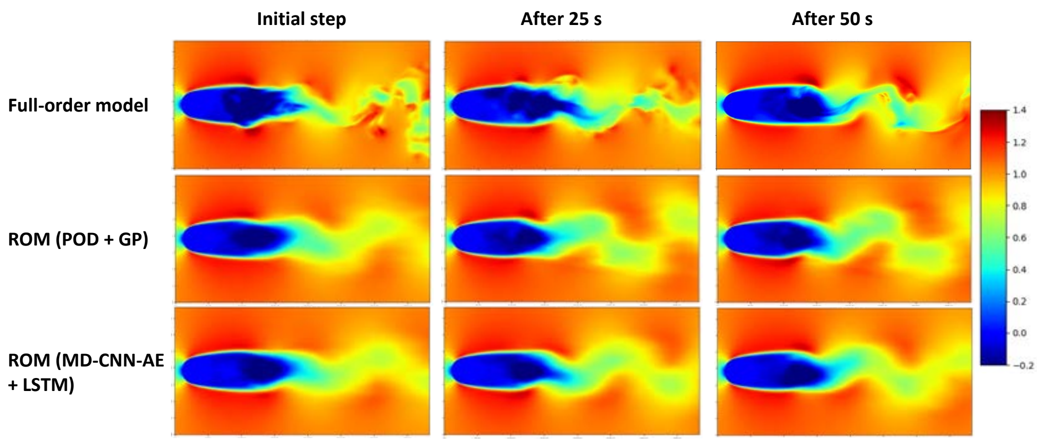


Figure 12: Comparison of streamwise velocity between FOM and ROM predictions with 16 modes.

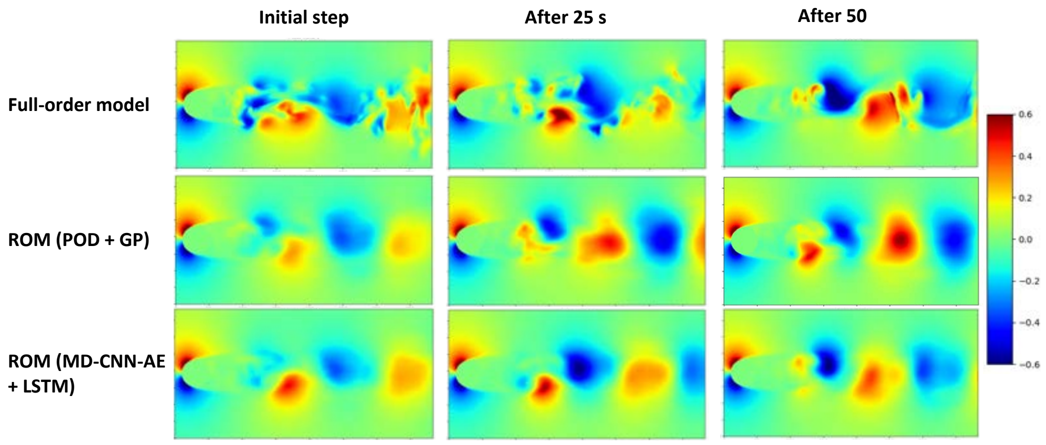


Figure 13: Comparison of vertical velocity between FOM and ROM predictions with 16 modes.

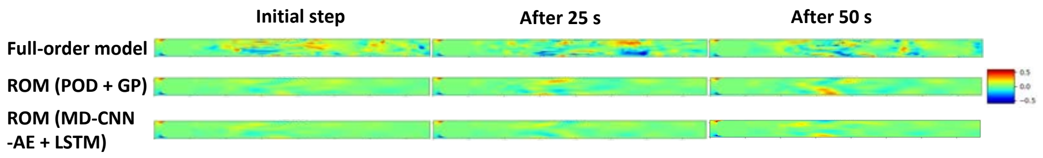


Figure 14: Comparison of spanwise velocity between FOM and ROM predictions with 16 modes.

Figures 15–17 are comparisons of the flow fields between the FOM results and ROM predictions with 64-mode decomposition. Using MD-CNN-AE + LSTM, The reproducibility of the complex flow field structure contained in the original flow field was better than that of the 16 modes. Specifically, the amplitude of the flow was reproduced well by the proposed method in the spanwise velocity.

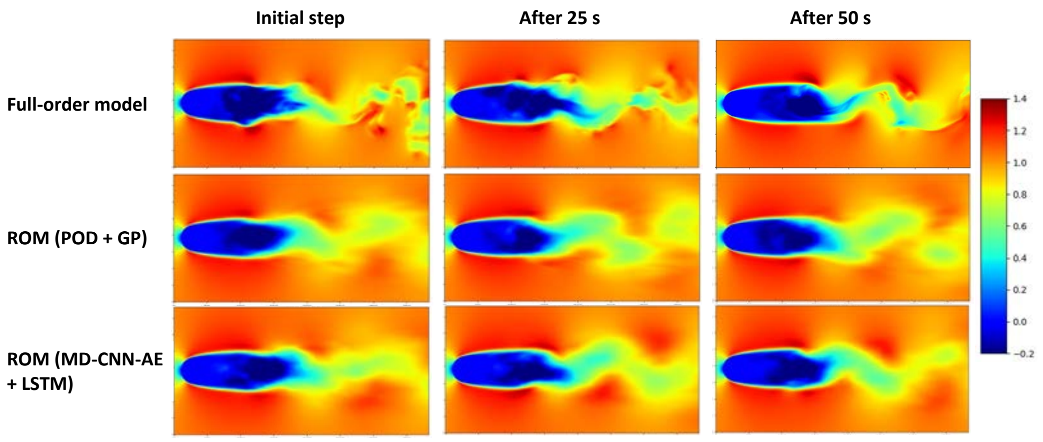


Figure 15: Comparison of streamwise velocity between FOM and ROM predictions with 64 modes.

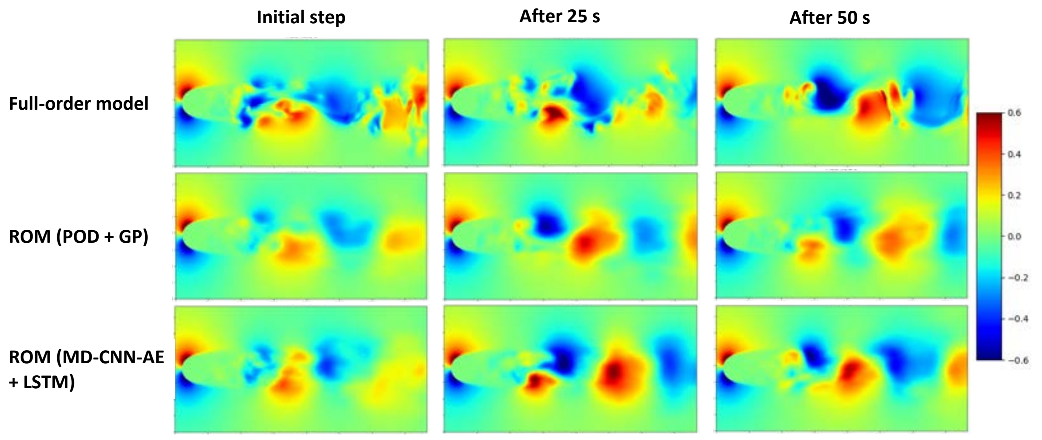


Figure 16: Comparison of vertical velocity between FOM and ROM predictions with 64 modes.

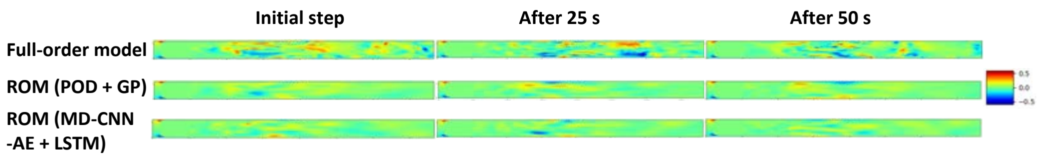


Figure 17: Comparison of spanwise velocity between FOM and ROM predictions with 64 modes.

Figure 18 shows the time-averaged value of the L2 error between the FOM and ROM predictions with 16-mode decomposition. In MD-CNN-AE + LSTM method, the area with a large error value (black region) is smaller than that of POD + GP.

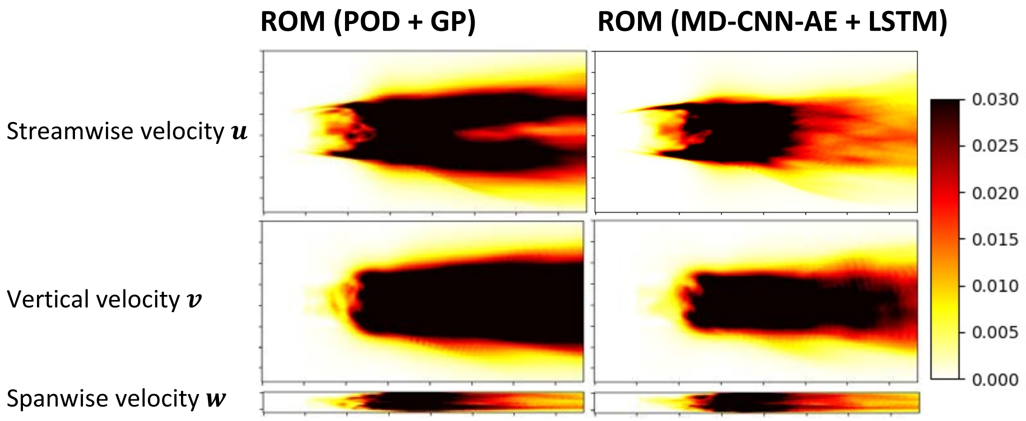


Figure 18: Comparison of time-averaged L2 error between FOM and ROM predictions with 16 modes: the upper, center, and bottom figure represents, respectively the stream, vertical, and spanwise velocity.

Figure 19 shows the time-averaged value of the L2 error between FOM and ROM predictions with 64 modes. The reproduction accuracy of MD-CNN-AE + LSTM is still higher than that of POD + DP; however, the difference is moderate because the property of the time-averaged field is considered to be mostly determined in 64 POD modes.

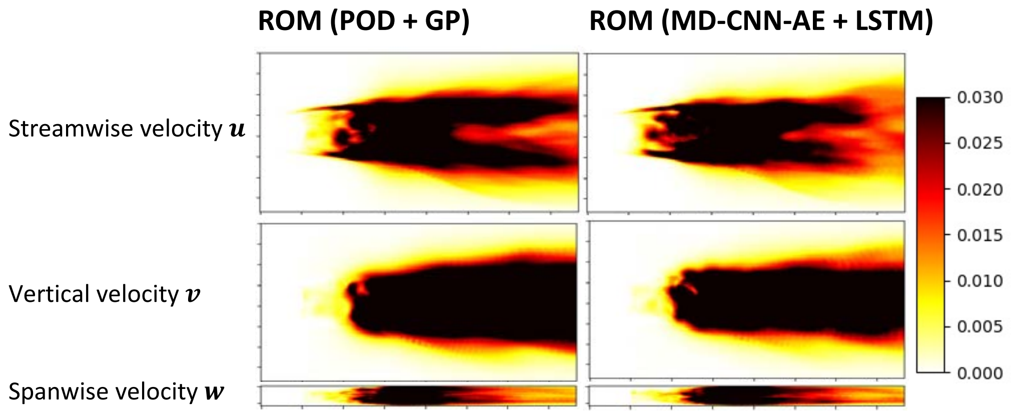


Figure 19: Comparison of time-averaged L2 error between FOM and ROM predictions with 64 modes.

Figure 20 shows the time variation of the space-averaged L2 error between the FOM and ROM predictions with 16 and 64 modes. While both the MD-CNN-AE + LSTM and POD + GP methods show a trend of increasing error as the number of modes increases, the error value of the MD-CNN-AE + LSTM is smaller than that of the POD + GP method for most of the period. However, the error of the POD + GP may oscillate periodically; thus, in the last 10 s, the error of the POD + GP method is lower than that of MD-CNN-AE + LSTM method.

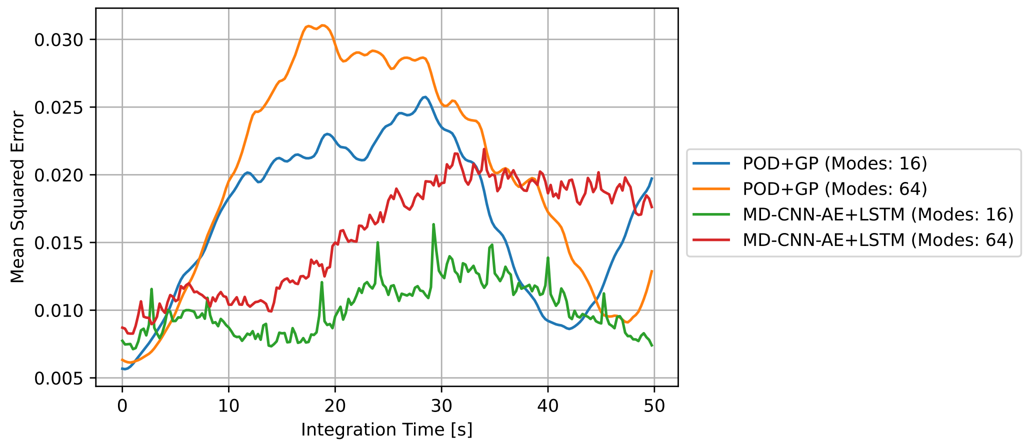


Figure 20: Comparison of time variation of the space-averaged L2 error between FOM and ROM predictions with 16 and 64 modes.

Figure 21 shows the time-averaged value of the flow field on the center line between the FOM and ROM predictions with 16 and 64 modes. In this case, MD-CNN-AE + LSTM method with 16 modes is the best fitted to the FOM results.

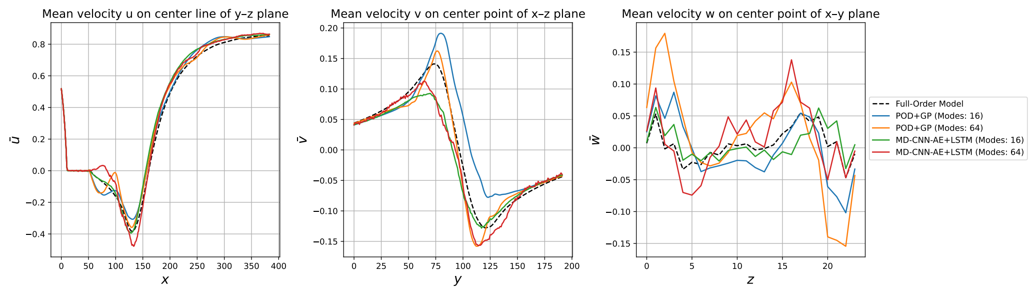


Figure 21: Comparison of the time-averaged value of the flow field on the center line between FOM and ROM predictions with 16 and 64 modes. From left to right are the time-averaged value of the stream-wise, vertical, and span-wise velocity, respectively.

The results so far in this chapter show that as the number of decomposing modes increases, finer vortex structures can be resolved (Fig. 12–17) while the reproducibility of average values in time or space decreases (Fig. 18–21). This is because the temporal or spatial average of the flow field is determined by only a few dominant modes. Using more than a sufficient number of modes to reproduce the mean field is considered to introduce noise into the mean field reconstruction.

Figure 22 shows the isosurface of the second invariant of the velocity gradient tensor between the FOM and ROM predictions with 16 modes. The reproduction of the POD + GP method is not precise in the wake region. In contrast, MD-CNN-AE + LSTM can reproduce the complex flow field of the FOM.

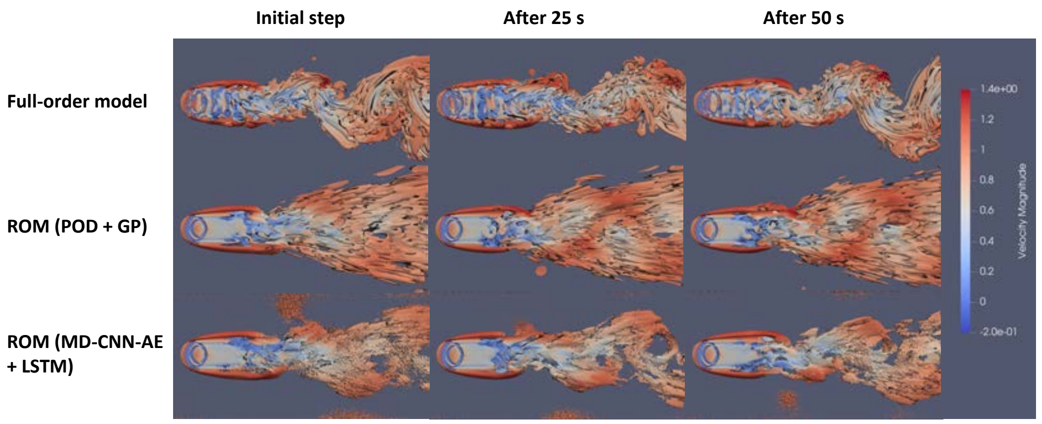


Figure 22: Comparison of the iso-surface of the second invariant of the velocity-gradient tensor between FOM and ROM predictions with 16 modes.

Figure 23 shows the isosurface of the second invariant of the velocity gradient tensor between the FOM and ROM predictions with 64 modes. Although, the reproduced flow field of the AI method is slightly noisy owing to the insufficient number of learning epochs, it can reproduce a more complex flow field than the one with 16 modes. Thus, it is expected that the greater the number of modes to decompose in MD-CNN-AE + LSTM, the greater is the reproduction accuracy for the complex flow field.

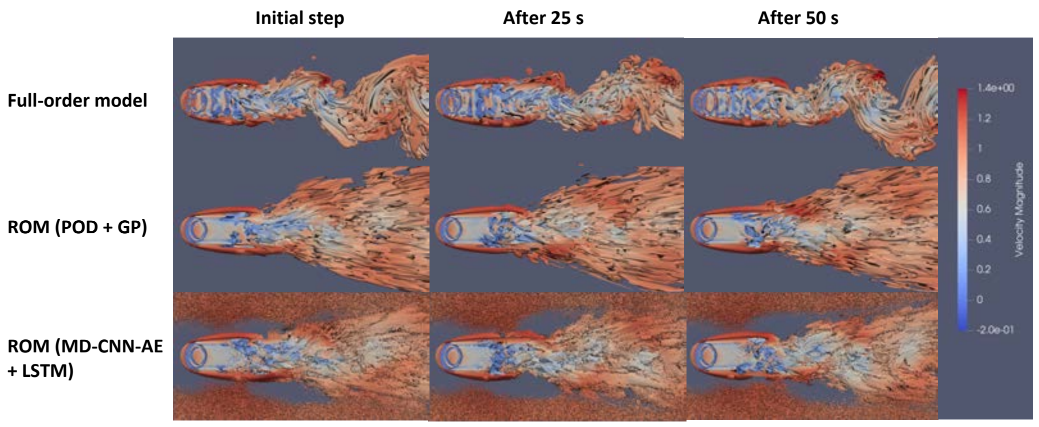


Figure 23: Comparison of the iso-surface of the second invariant of the velocity-gradient tensor between FOM and ROM predictions with 64 modes.

3.3. Computational Performance of Training for Mode Decomposition

Table 2 lists the computational performances of the training for mode decomposition using one of the four computational units called core memory groups (CMGs) in A64FX CPU installed in Fugaku node. The entire training loop, which involves I/O and communications, indicates 129 GFLOPS, which corresponds to 7.76 % of the single-precision floating-point arithmetic peak performance. Moreover, the convolution kernel indicates 1.1 TFLOPS, which corresponds to 65.39 % of the peak performance. This kernel calls convolution routine in the Intel oneDNN library ported to Fugaku by Fujitsu [45]. Figure 24 shows the CPU cycle counter result, which indicates whether the core works efficiently in each CPU cycle in the convolution routine. The light blue bar indicates the amount of time while the instructions are committed most efficiently — that is, this kernel is almost perfectly optimized to A64FX.

Table 2: Computational performance of the training for mode decomposition[a].

	Total	Convolution
FLOPS	129.50 GFLOPS (7.67%)	1103.09 GFLOPS (65.39%)
Mem. throughput	10.92 GB/s (4.26%)	38.16 GB/s (14.91%)
L1D cache miss rate	2.39%	0.34%
L2 cache miss rate	0.67%	0.25%

[a] 1 CMG running at 2.2GHz is used.

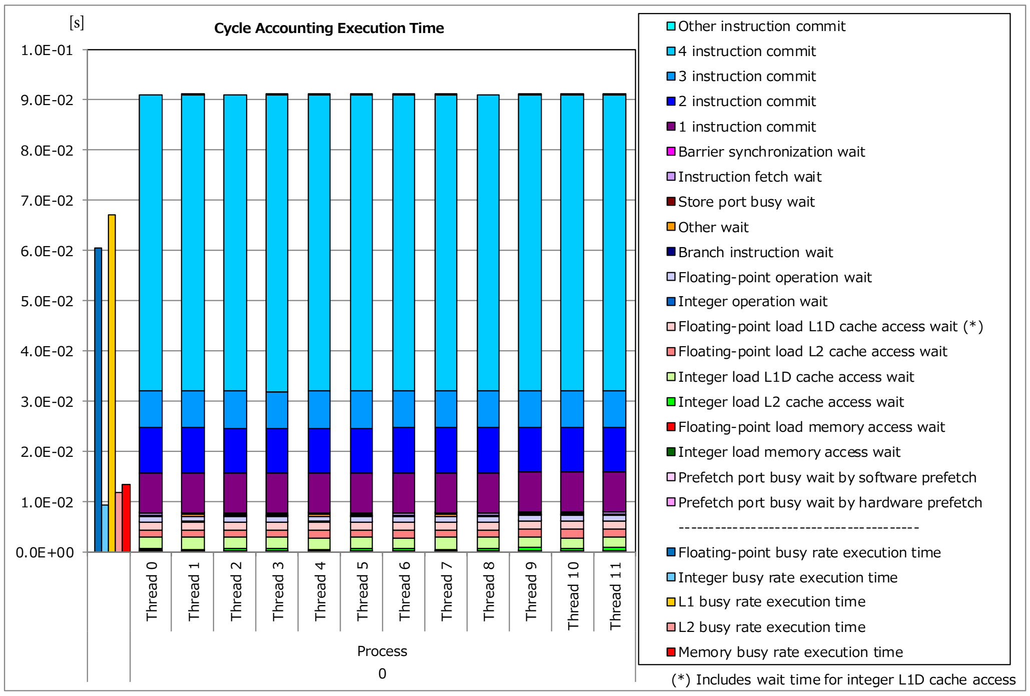


Figure 24: Cycle counter result of the convolution kernel .

Figure 25 show the results of the weak scaling test — that is, the number of nodes increased while maintaining the computational cost per node. In this case, the number of modes was increased; thus, the number of branches in the encoder increased as the number of nodes increased. The entire training process indicates 7.80 PFLOPS when decomposing into 100 modes using 25,250 nodes and 72.9% of weak scaling performance relative to that of 750 nodes. Furthermore, forward-propagation and backward-propagation indicates 25.1 PFLOPS and 19.4 PFLOPS, respectively.

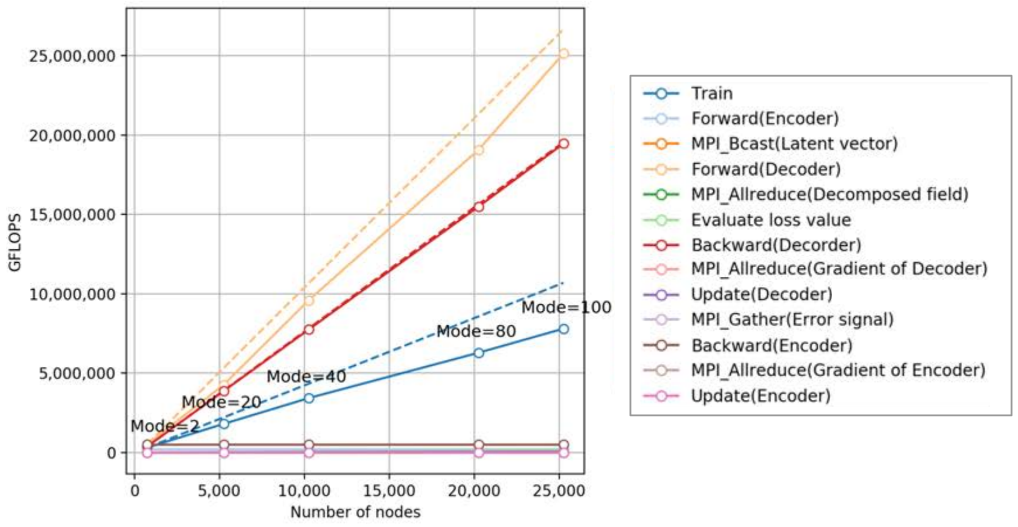


Figure 25: Weak scaling performance of the training for mode decomposition: the blue, orange, and red lines indicate, respectively the performance of the entire training process, forward-propagation, and back-propagation in the decoder. The solid line corresponds to measured values and the dashed line corresponds to the ideal scaling of that.

Figure 26 shows the result of the weak scaling test of the convolution routines. Its scaling is almost perfect and reaches around 100 PFLOPS using 25,250 nodes.

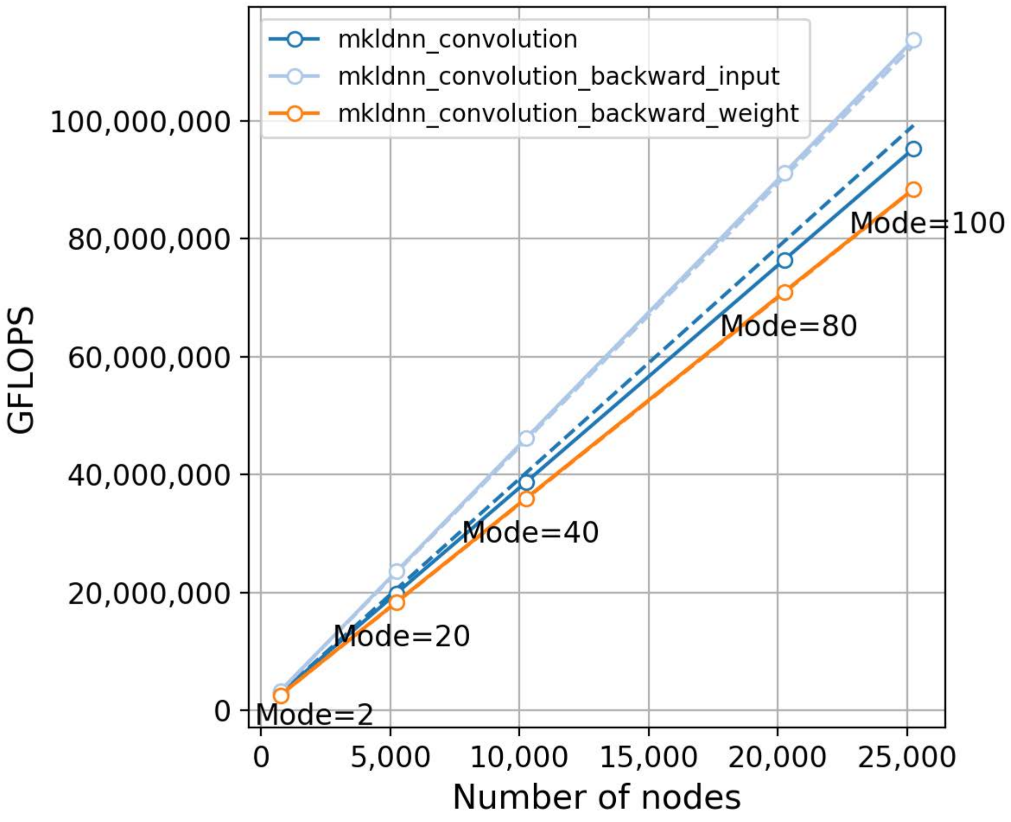


Figure 26: Weak scaling performance of the convolution routines. The blue, the cyan, and the orange line means the routine for forward propagation, the back-propagation to calculate error signal, and the back-propagation to calculate the amount of the weight update. The solid line corresponds to measured values and the dashed line corresponds to the ideal scaling of that.

3.4. Computational Cost of FOM and ROM

Table 3 lists the computational costs of the FOM and ROM using the MD-CNN-AE + LSTM method. Compared to the FOM, if you decompose into 20 modes, the execution time of the ROM is reduced by 4 orders of magnitude, and the number of operations is reduced by 5 orders of magnitude. Even if you use 400 modes, the execution time and the number of operations are estimated to reduce by 3 and 4 orders of magnitude, respectively.

Table 3: Comparison of the computational costs of FOM and ROM using AI method.

	Cells/ modes	CPU	Clock [GHz]	Total CPUs	Total cores	Exec. time per step	Operations per step[a]
FOM	28M	Intel Xeon Gold 6148	2.4	32	384	1.74 s	85.5 Tflop
ROM	2	Fujitsu A64FX	2.0	1	12	572 µs	439 Mflop
	20					7.37 µs	566 Mflop
	400					4.22 ms[b]	3.24 Gflop[b]

[a] The number of operations is an estimated value calculated by (Execution time)×(Peak floating point calculation performance)

[b] Estimated value calculated from the measurements of two modes and 20 modes.

4. Conclusions

A large-scale machine-learning-based nonlinear reduced-order modeling method for a three-dimensional turbulent flow field (Re = 1000) using unsupervised neural-network learning was proposed. The mode decomposition method for the three-dimensional flow field data using a CNN-AE-like neural network is implemented in the form of distributed machine learning using a hybrid parallelism scheme tailored for the proposed network structure. To validate this method, the prediction performance of the mode decomposition method was evaluated and compared with the POD method. Using this implementation, it is possible to decompose 1.7 million cells of the three-dimensional flow field data into 64 modes, which results in a sufficient reduction accuracy. Then, a ROM was constructed using LSTM neural networks. It was demonstrated that the time evolution of a turbulent three-dimensional flow field can be simulated at a significantly low cost (approximately three orders of magnitude) without a major loss of accuracy. In this study, a uniform flow around a circular cylinder model was used as a test case. The reconstruction accuracy was estimated in terms of various criteria compared with the model based on POD in conjunction with the GP method, and the superiority of the proposed method was demonstrated.

In future, it is planned to apply the method to more-complex flow fields, such as the flow around a vehicle body. In addition, time-random phenomena, such as the temporal pressure variation in the fluctuating flow around a 3D triangular cylinder, should be reproduced. Furthermore, the reduction performance of the VAE version of MD-CNN-AE must be evaluated.

Acknowledgement

We sincerely appreciate the advice and support of Prof. Koji Fukagata, Mr. Kai Fukami, and Mr. Takaaki Murata of the Keio University. Computational resources of Fugaku were provided through the HPCI System Research Project (Project ID: hp220180).

Appendix A. Network Structure

In this section, the network structure we used are detailed.

Appendix A.1. MD-CNN-AE

The detailed MD-CNN-AE network structure is presented in Table A.4.

Table A.4: Network structure of MD-CNN-AE.

Encoder	Data size	Decoder	Data size
Input	(384, 192, 24, 3)	1st Value	(1, 1, 1, 1)
1st 3D convolution	(384, 192, 24, 16)	Fully connected	(6, 3, 3, 4)
1st 3D maxpooling	(192, 96, 12, 16)	1st 3D upsampling	(12, 6, 3, 4)
2nd 3D convolution	(192, 96, 12, 8)	1st 3D convolution	(12, 6, 3, 4)
2nd 3D maxpooling	(96, 48, 6, 8)	2nd 3D upsampling	(24, 12, 3, 4)
3rd 3D convolution	(96, 48, 6, 8)	2nd 3D convolution	(24, 12, 3, 8)
3rd 3D maxpooling	(48, 24, 3, 8)	3rd 3D upsampling	(48, 24, 3, 8)
4th 3D convolution	(48, 24, 3, 8)	3rd 3D convolution	(48, 24, 3, 8)
4th 3D maxpooling	(24, 12, 3, 8)	4th 3D upsampling	(96, 48, 6, 8)
5th 3D convolution	(24, 12, 3, 4)	4th 3D convolution	(96, 48, 6, 8)
5th 3D maxpooling	(12, 6, 3, 4)	5th 3D upsampling	(192, 96, 12, 8)
6th 3D convolution	(12, 6, 3, 4)	5th 3D convolution	(192, 96, 12, 16)
6th 3D maxpooling	(6, 3, 3, 4)	6th 3D upsampling	(384, 192, 24, 16)
Fully connected	(n,[a] 1, 1, 1)	6th 3D convolution	(384, 192, 24, 3)

[a] n is the number of the mode decompositions.

Appendix A.2. LSTM

The detailed LSTM network structure is shown in Table A.5.

Table A.5: Network structure of LSTM.

Order	Layer	Activation	Output size
1	Input		(20, n)[a]
2-a	LSTM	tanh	(20, n)
	LSTM	linear	(20, n)
2-b	LSTM	tanh	(20, 1.5n)
	LSTM	linear	(20, n)
2-c	LSTM	tanh	(20, 2n)
	LSTM	linear	(20, n)
2-d	LSTM	tanh	(20, 2.5n)
	LSTM	linear	(20, n)
2-e	LSTM	tanh	(20, 3n)
	LSTM	linear	(20, n)
2-f	LSTM	tanh	(20, 3.5n)
	LSTM	linear	(20, n)
2-g	LSTM	tanh	(20, 4n)
	LSTM	linear	(20, n)
3	Add 2-a–2-g	tanh	(20, n)

[a] n is the number of the mode decompositions.

Appendix B. Hyperparameters

In this section, the hyperparameters we used are detailed.

Appendix B.1. MD-CNN-AE

The hyperparameters of MD-CNN-AE we used is shown in the Table B.6.

Table B.6: Hyperparameters of MD-CNN-AE.

Name	Value	Name	Value
Filter size	$3 \times 3 \times 3$	Batch size	768
Pooling size	$2 \times 2 \times 2$	Number of epochs	11,000
Weight initializer	Xavier uniform	Number of training data	9984
Optimizer	Adam	Ratio of validation data	30%
Learning rate	0.001	Number of test data	2,000
Time step size	0.025 s		

Appendix B.2. LSTM

The hyperparameters of MD-CNN-AE we used is shown in the Table B.7.

Table B.7: Hyperparameters of LSTM.

Name	Value	Name	Value
Sequence length	20	Batch size	1,000
Num. of LSTM layers	3	Number of epochs	2,000
Weight initializer	Xavier uniform	Number of training data	10,000
Optimizer	Adam	Ratio of validation data	30%
Learning rate	0.001	Number of test data	2,000
Time step size	0.25 s		

Appendix C. Time series prediction with LSTM

In this section, the time series of the latent vector elements predicted by LSTM are shown.

Figure C.27 shows all the latent vector elements in the case decomposed in 16 modes.

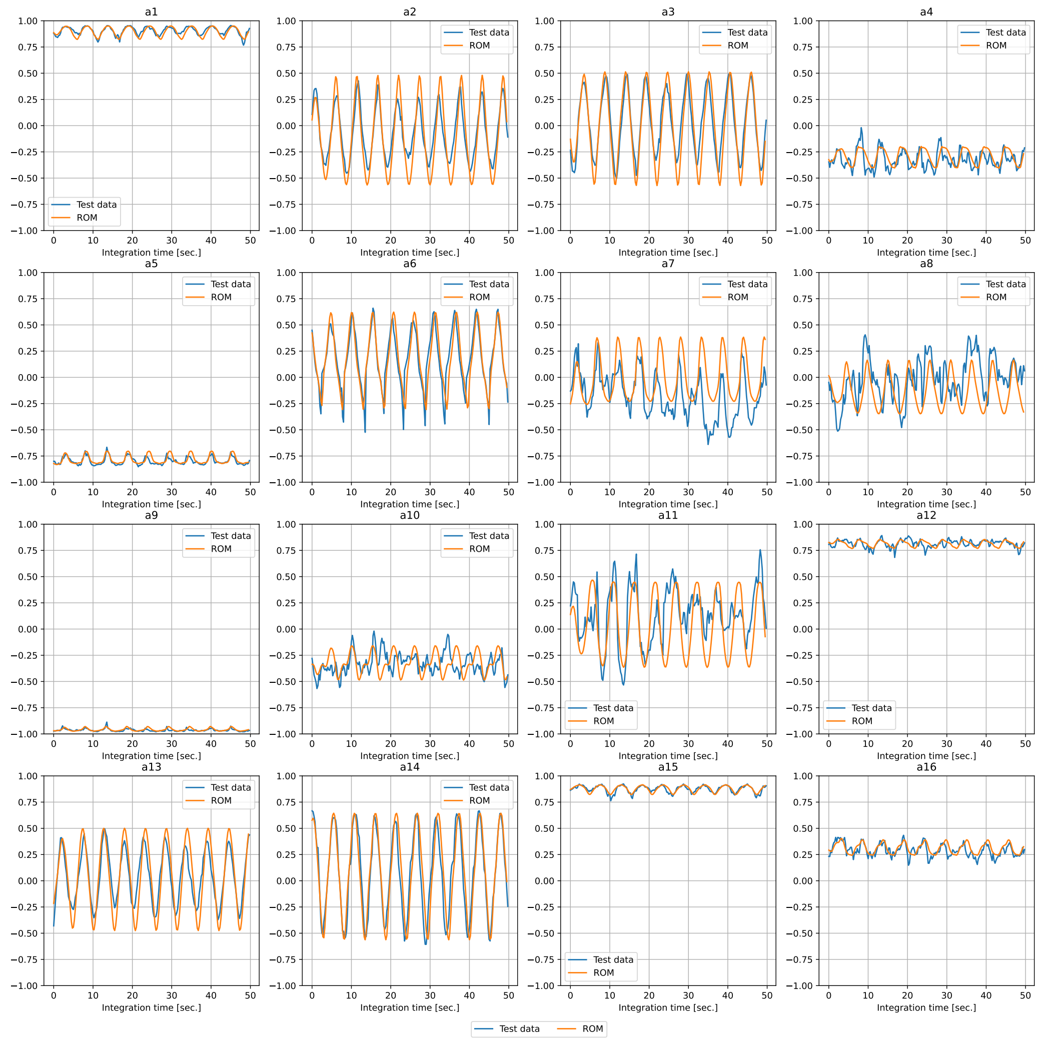


Figure C.27: Time series of latent vector elements predicted by LSTM with 16 modes-decomposition: the blue and orange lines indicate the test data and the prediction with LSTM derived from first 20 time steps.

Figure C.28 shows all the latent vector elements in the case decomposed in 64 modes.

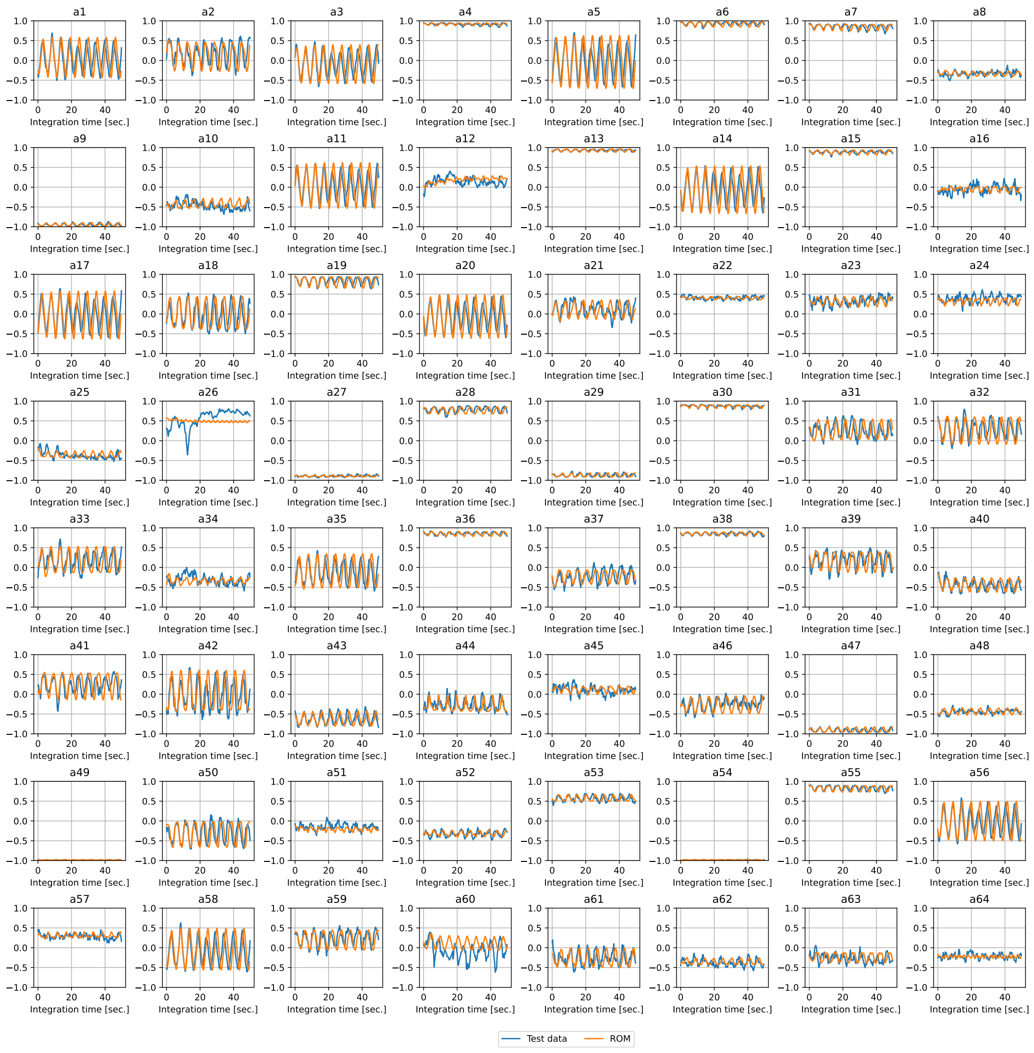


Figure C.28: Time series of latent vector elements predicted by LSTM with 64 modes-decomposition: the blue and orange lines indicate the test data and the prediction with LSTM derived from first 20 time steps.

References

[1] J. B. Tenenbaum, V. de Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.

[2] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[3] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: T. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, Vol. 14, MIT Press, 2001.
URL `https://proceedings.neurips.cc/paper/2001/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf`

[4] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of Machine Learning Research 9 (11) (2008).

[5] G. E. Hinton, S. Roweis, Stochastic neighbor embedding, Advances in Neural Information Processing Systems 15 (2002).

[6] B. Galerkin, On electrical circuits for the approximate solution of the Laplace equation, Vestnik Inzh 19 (1915) 897–908.

[7] A. Kolmogoroff, Uber die beste annaherung von funktionen einer gegebenen funktionenklasse, Ann. Math. 37 (1) (1936) 107–110.

[8] M. Barrault, Y. Maday, N. C. Nguyen, A. T. Patera, An ‘empirical interpolation’ method: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Mathematique 339 (9) (2004) 667–672. doi:https://doi.org/10.1016/j.crma.2004.08.006.
URL `https://www.sciencedirect.com/science/article/pii/S1631073X04004248`

[9] S. Chaturantabut, D. C. Sorensen, Discrete empirical interpolation for nonlinear model reduction, in: Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, 2009, pp. 4316–4321.

[10] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations, Int. J. Numer. Methods Eng. 86 (2) (2011) 155–181.

[11] M. Rewienski, J. White, A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 22 (2) (2003) 155–170.

[12] G. Haller, S. Ponsioen, Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction, Nonlinear Dyn. 86 (3) (2016) 1493–1534.

[13] G. E. Hinton, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[14] N. Omata, S. Shirayama, A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder, AIP Adv. 9 (1) (2019) 015006.

[15] T. Murata, K. Fukami, K. Fukagata, Nonlinear mode decomposition with convolutional neural networks for fluid dynamics, Journal of Fluid Mechanics 882 (2020) A13. doi:10.1017/jfm.2019.822.

[16] D. P. Kingma, M. Welling, Auto-Encoding variational bayes (Dec. 2013). arXiv:1312.6114v10.

[17] A. Plumerault, H. Le Borgne, C. Hudelot, AVAE: Adversarial variational auto encoder (Dec. 2020). arXiv:2012.11551.

[18] R. Maulik, T. Botsas, N. Ramachandra, L. R. Mason, I. Pan, Latent-space time evolution of non-intrusive reduced-order models using Gaussian process emulation (Jul. 2020). arXiv:2007.12167.

[19] T. Botsas, I. Pan, L. R. Mason, O. K. Matar, Multiphase flow applications of nonintrusive reduced-order models with gaussian process emulation, Data-Centric Engineering 3 (2022).

[20] J. M. Graving, I. D. Couzin, VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering (Jul. 2020).

[21] S. Lee, K. Jang, H. Cho, H. Kim, S. Shin, Parametric non-intrusive model order reduction for flow-fields using unsupervised machine learning, Comput. Methods Appl. Mech. Eng. 384 (2021) 113999.

[22] S. Hochreiter, J. Schmidhuber, Long Short-Term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[23] J. Yu, C. Yan, M. Guo, Non-intrusive reduced-order modeling for fluid problems: A brief review, Proc. Inst. Mech. Eng. G J. Aerosp. Eng. 233 (16) (2019) 5896–5912.

[24] C. K. Williams, C. E. Rasmussen, Gaussian processes for machine learning, Vol. 2, MIT press Cambridge, MA, 2006.

[25] K. Hasegawa, K. Fukami, T. Murata, K. Fukagata, Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes, Theor. Comput. Fluid Dyn. 34 (4) (2020) 367–383.

[26] T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, K. Fukagata, Extension of CNN-LSTM based reduced order surrogate for minimal turbulent channel flow, undefined (2020).

[27] C. Quilodrán-Casas, R. Arcucci, C. Pain, Y. Guo, Adversarially trained LSTMs on reduced order models of urban air pollution simulations (Jan. 2021). arXiv:2101.01568.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 27, Curran Associates, Inc., 2014.
URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[29] R. Maulik, A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, D. Livescu, Time-series learning of latent-space dynamics for reduced-order model closure, Physica D 405 (2020) 132368.

[30] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018, pp. 6571–6583.

[31] Y. Kim, Y. Choi, D. Widemann, T. Zohdi, A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder (Sep. 2020). arXiv:2009.11990.

[32] K. Lee, K. T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, J. Comput. Phys. 404 (2020) 108973.

[33] T. Yoshida, Fujitsu high performance CPU for the Post-K computer, in: Hot Chips, Vol. 30, 2018.

[34] R-CCS, About Fugaku, https://www.r-ccs.riken.jp/en/fugaku/about/, (Accessed on 10/07/2021) (2021).

[35] K. Ando, K. Onishi, R. Bale, M. Tsubokura, A. Kuroda, K. Minami, Nonlinear mode decomposition and Reduced-Order modeling for Three-Dimensional cylinder flow by distributed learning on Fugaku, in: High Performance Computing, Springer International Publishing, 2021, pp. 122–137.

[36] N. Jansson, R. Bale, K. Onishi, M. Tsubokura, CUBE: A scalable framework for large-scale industrial simulations, Int. J. High Perform. Comput. Appl. 33 (4) (2019) 678–698.

[37] K. Nakahashi, Building-Cube method for flow problems with broadband characteristic length, in: Computational Fluid Dynamics 2002, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 77–81.

[38] K. Onishi, M. Tsubokura, Topology-free immersed boundary method for incompressible turbulence flows: An aerodynamic simulation for "dirty" CAD geometry, Comput. Methods Appl. Mech. Eng. 378 (2021) 113734.

[39] C. S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.

[40] A. P. S. Bhalla, R. Bale, B. E. Griffith, N. A. Patankar, A unified mathematical framework and an adaptive numerical method for fluid-structure interaction with rigid, deforming, and elastic bodies, J. Comput. Phys. 250 (2013) 446–476.

[41] K. Nishiguchi, R. Bale, S. Okazawa, M. Tsubokura, Full eulerian deformable solid-fluid interaction scheme based on building-cube method for large-scale parallel computing, International Journal for Numerical Methods in Engineering 117 (2) (2019) 221–248. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.5954, doi:https://doi.org/10.1002/nme.5954.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.5954`

[42] T. Shimada, K. Nishiguchi, R. Bale, S. Okazawa, M. Tsubokura, Eulerian finite volume formulation using lagrangian marker particles for incompressible fluid–structure interaction problems, International Journal for Numerical Methods in Engineering 123 (5) (2022) 1294–1328. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.6896, doi:https://doi.org/10.1002/nme.6896.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6896`

[43] K. Taira, S. L. Brunton, S. T. M. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, L. S. Ukeiley, Modal analysis of fluid flows: An overview, AIAA Journal 55 (12) (2017) 4013–4041.

[44] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (Dec. 2014). arXiv:1412.6980.

[45] Fujitsu, Deep Neural Network Library for AArch64, https://github.com/fujitsu/dnnl_aarch64 (2021).