Neural Network-Based Parametric Model Reduction for Predicting Turbulent Flow for Different Vehicle Geometries

Kazuto Ando[1,2,3], **Rahul Bale**[2,3], **Akiyoshi Kuroda**[1], **Makoto Tsubokura**[2,3]

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

Keywords: Parametric model reduction, Neural network, Turbulent flow, Vehicle body

Abstract. Numerical simulations in industrial applications often require performing numerous high-precision computations parameterized by specific experimental conditions. For instance, in vehicle body design, aerodynamic simulations are essential for evaluating the aerodynamic characteristics of various proposed body geometries. However, computational resource constraints often become a bottleneck. Therefore, achieving the desired accuracy while minimizing computational cost is crucial. To address this challenge, model reduction methods have been developed to decrease the degrees of freedom by constraining the possible states of a physical system to a lower-dimensional subspace. In particular, reduction techniques that project the system onto a nonlinear subspace using neural networks have been actively studied. Our previous research developed a reduced-order model that integrates neural-network-based model reduction with a time-evolution method, implemented as a distributed parallel training framework to process high-resolution flow field data efficiently. In this study, we extend this reduction approach by incorporating a variational autoencoder to assess its robustness in high-Reynolds-number flows around multiple vehicle bodies with varying geometries. Specifically, we evaluate the reconstruction accuracy of vortex generation across different spatial and temporal scales using a compact latent representation, with a particular focus on the flow behavior near the rear end of the vehicle body.

1. Introduction

Motivation
Currently, the computational cost required for computational fluid dynamics (CFD) simulations continues to increase. One prominent example is large-scale computations performed on high-end supercomputer systems. Kato et al. [1] conducted large-scale simulations on the Supercomputer Fugaku [2,3] using 458,752 MPI processes across 114,688 compute nodes with the flow solver "FrontFlow/blue" to analyze the turbulent flow dynamics around a ship body, utilizing 32 billion finite elements. Similarly, Jansson et al. [4] employed the spectral-element-based direct numerical simulation (DNS) code "Neko" to perform high-precision simulations of Rayleigh-Bénard convection using 16,384 GPUs on LUMI [5].

At the same time, there is an increasing demand for conducting numerous medium-scale simulations, particularly in industrial applications. For instance, multi-objective shape optimization [6] requires running multiple simulations with different shapes to evaluate and optimize aerodynamic performance. However, performing a large number of simulations that fully resolve the smallest scales of the physical phenomena of interest—such as vortices generated by turbulence—entails a substantial computational cost. According to Kolmogorov's $-5/3$ power law, the smallest vortices generated around a vehicle body occur at approximately 100 Hz [7], making direct simulations infeasible within a practical computational budget.

Model reduction methods
Various methods have been proposed in the CFD community to mitigate the computational burden of such simulations. One widely used approach is model reduction, which restricts a physical system's states to a specific subspace and represents the system state using a small number of representative

variables (hereafter referred to as latent variables). Model reduction methods can be broadly categorized into linear and nonlinear approaches.

A representative method in the former category is proper orthogonal decomposition (POD) [8], which identifies an optimal low-dimensional basis, called modes, to approximate a given dataset—such as time-series flow field data.

On the other hand, nonlinear reduction methods, which are well-suited for capturing the nonlinear behavior of flow fields, have also been proposed. A typical approach in this category is the neural-network-based method, which extracts nonlinear modes from time-series instantaneous flow field data. Murata et al. proposed a neural network model called the mode-decomposing convolutional neural network autoencoder (MD-CNN-AE) to reduce the dimensionality of two-dimensional flow data around a circular cylinder [9]. They successfully represented the time evolution of the Kármán vortex street at Re = 100 using only two latent variables without significant reconstruction loss.

In our previous study, MD-CNN-AE was extended to three-dimensional flow fields at a higher Reynolds number (Re = 1,000) [10] using large-scale distributed machine learning on the Supercomputer Fugaku. A three-dimensional flow around a cylinder, simulated using 28 million computational cells, was reduced to 64 latent variables, and the time series of these latent variables were predicted using a long short-term memory (LSTM) network [11]. Our reduction method, implemented via distributed machine learning, demonstrated scalability up to 25,250 computational nodes (1,212,000 cores) on Fugaku, with the convolution routine achieving over 100 PFLOPS in single-precision floating-point arithmetic performance.

Meanwhile, ongoing research focuses on parametric model reduction, which enables reduced-order simulations under different execution settings (e.g., object shape, Reynolds number) than those used during training. One example of this approach is the study by Koo et al. [12], which predicts the time evolution of flow velocity in a reservoir pipe-valve (RPV) system using a POD-based reduced-order model. This model is parameterized with respect to two parameters: the height at the upstream end and the Reynolds number. POD extracts basis functions from instantaneous flow fields generated for nine different parameter combinations using high-precision simulations. Finally, the accuracy of the reduced-order model was evaluated for parameter values not included in these nine training cases. A detailed classification of parametric model reduction methods is provided by Benner et al. [13], where techniques such as rational interpolation, balanced truncation, and POD are introduced, along with application examples including thermal modeling of electric motors, optimization of batch chromatography, and control of rod movement in nuclear reactor cores.

In this study, we implement a neural-network-based parametric model reduction method for predicting unknown turbulent flow fields and evaluate its robustness in the context of varying vehicle body shapes.

Related works

Hasegawa et al. [14] proposed a neural-network-based model reduction technique that combines a convolutional neural network autoencoder with a long short-term memory (LSTM) network. They evaluated the robustness of their model for two-dimensional flow around various randomly generated bluff body shapes. As a result, flow fields for approximately 20 bluff body shapes were successfully reproduced using a neural network trained on 80 shapes.

Hasegawa et al. [15] further assessed the robustness of their method for two-dimensional circular cylinder flow at varying Reynolds numbers. In this study, Reynolds number information was fed into the LSTM network, allowing the model to successfully predict the flow field around a circular cylinder at Reynolds numbers not included in the training data.

Higashida et al. [16] investigated the robustness of our previously developed model reduction method [10] for two-dimensional flow around two distant square cylinders with varying inter-cylinder distances. A comprehensive comparison with POD across different inter-cylinder distances demonstrated that the proposed method outperforms POD in reconstructing flow fields at untrained distances.

Contributions
This study proposes a neural-network-based parametric model reduction method tailored for three-dimensional high-Reynolds-number turbulent flow. The proposed method predicts the time evolution of turbulent flow ($Re \sim 8.4 \times 10^6$) in the rear end region of a vehicle body with varying body shapes.

Additionally, a variational autoencoder (VAE) is introduced for dimensionality reduction. This approach is robust to parameterization as it ensures continuity in the latent space. The applicability of the proposed method is demonstrated using eleven different realistic vehicle models.

The paper is structured as follows: Section 2 introduces the proposed model reduction method, including training data generation, network architecture, and parallelization strategies. Section 3 presents the evaluation results of the proposed model reduction method for turbulent flow with varying vehicle body shapes. Finally, Section 4 concludes the paper with a summary.

2. Methods

This section describes the method used to generate training data for this study, specifically the target flow field data for model reduction. It also introduces our neural-network-based parametric nonlinear reduction method, providing a detailed explanation of the neural network architecture and distributed parallel training approach.

2.1. Training data

This section explains how the training data used in this study was generated. High-precision three-dimensional turbulent flow simulations were conducted using the simulation framework "CUBE" [17], developed by the Complex Phenomena Research Team at the RIKEN Center for Computational Science. CUBE is a unified simulation framework based on the Building Cube Method (BCM) [18,19] and the Immersed Boundary Method (IBM) [20,21].

Governing equations
The governing equations for the flow simulations conducted in this study are defined as follows. The incompressible Navier-Stokes equations are given by

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}\right) = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{f}, \tag{1}$$

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

where $\boldsymbol{u}$ represents the flow velocity vector, p is the pressure, ρ is the density, μ is the viscocity, and $\boldsymbol{f}$ is the force vector exerted by the immersed geometries in the fluid (the vehicle body in this study).

Problem settings
This section describes the problem settings in this study. The computational domain extends over a large area surrounding the vehicle body, with a constant inflow entering in the positive X-direction (Figure 1(a)).

In contrast, the region used for training the machine learning model is restricted to the area behind the vehicle body (shown as the gray area in Figure 1(a)). The computational grid becomes increasingly finer

as it approaches the vehicle body's surface, and within the training domain, the grid spacing is uniform and at its finest resolution (Figure 1(b)).

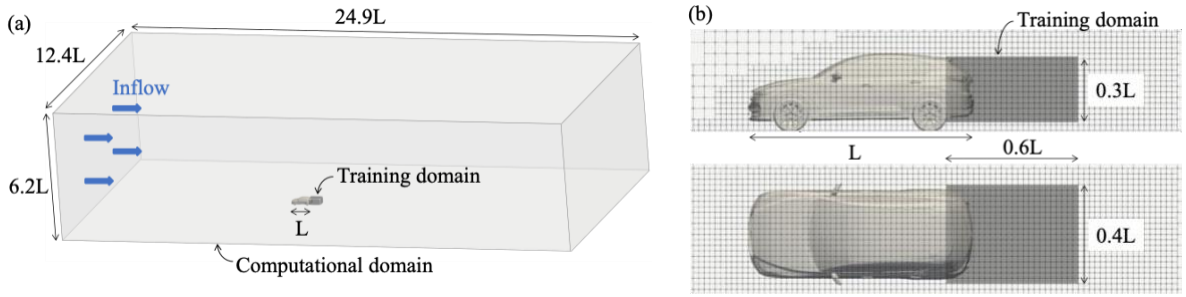

Figure 1. (a) Computational domain with inflow direction. (b) Training domain with uniform fine grid spacing.

In this study, three-dimensional flow velocity fields were obtained from turbulent simulations performed under the same execution conditions (see Table 1) for 11 vehicle body models provided by Professor Takuji Nakashima from Hiroshima University (Figure 2(a)).

These models were generated through principal component analysis (PCA) based on data from 124 commercially available SUVs, each characterized by 22 side-view design variables. The first principal component was varied from -1.5 to 1.5 in increments of 0.3, resulting in 11 distinct models.

These variations in shape lead to significant differences in aerodynamic performance. For instance, the maximum deformation in the X-direction at the rearmost position of the vehicle body is approximately 12 cm (Figure 2(b)).

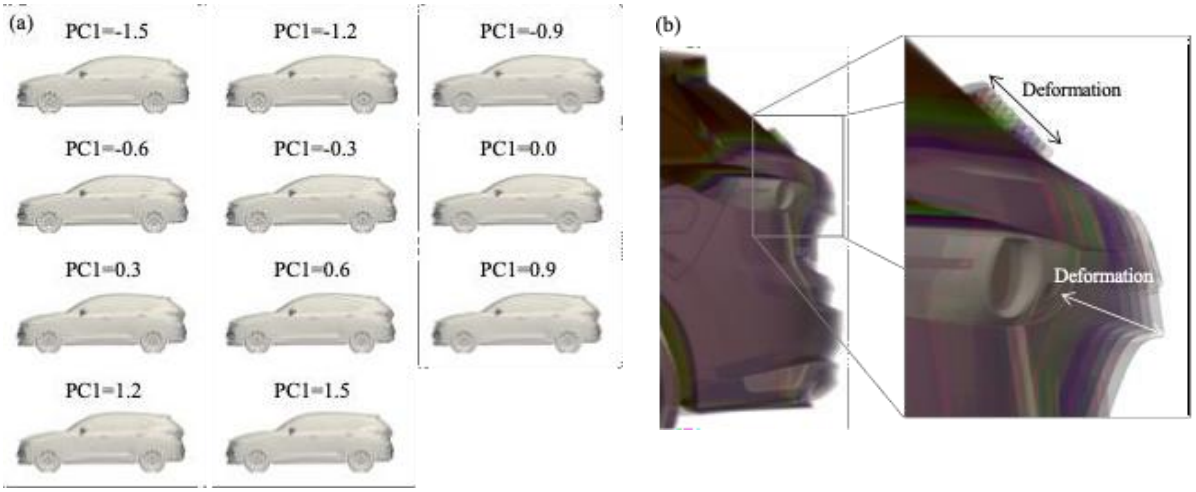

Figure 2. (a) Vehicle body models generated through principal component analysis (PCA). (b) Shape variations at the rearmost position of the vehicle body.

Table 1 provides a detailed description of the simulation settings used in this study. The computational domain is discretized into 200 million cells, with a Reynolds number of approximately 8.4×10^6. The sampling frequency for the training data is set to 0.5 ms to capture high-frequency fluctuations in the wake region.

Table 1. Simulation parameters and conditions.

Flow solver type	Incompressible flow solver

Computational domain size	114 m × 57 m × 28 m
Vehicle model size	4.6 m × 2.0 m × 1.5 m
Number of cubes	48,980
Cells per cube	16 × 16 × 16 = 4,096
Total number of cells	48,980 × 4,096 = 200,622,080
Minimum cell size	7.0 mm
Time step size	2.5×10^{-5} s
Sampling frequency	0.5 ms (per 20 steps)
Integration time	1.75 s (70,000 steps)
Inflow velocity	27.78 m/s (100 km/h)
Reynolds number	~ 8.4×10^{6}
Time integration scheme	Crank-Nicolson method
Pressure Poisson solver	V-cycle multigrid
Viscous term discretization	Second-order central difference
Convection term discretization	QUICK scheme

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

2.1. Training strategy

This section provides details on network training, including the network architecture, loss function, and distributed parallel training method.

Network architecture

The schematic of the entire network architecture used in this study is shown in Figure 3.

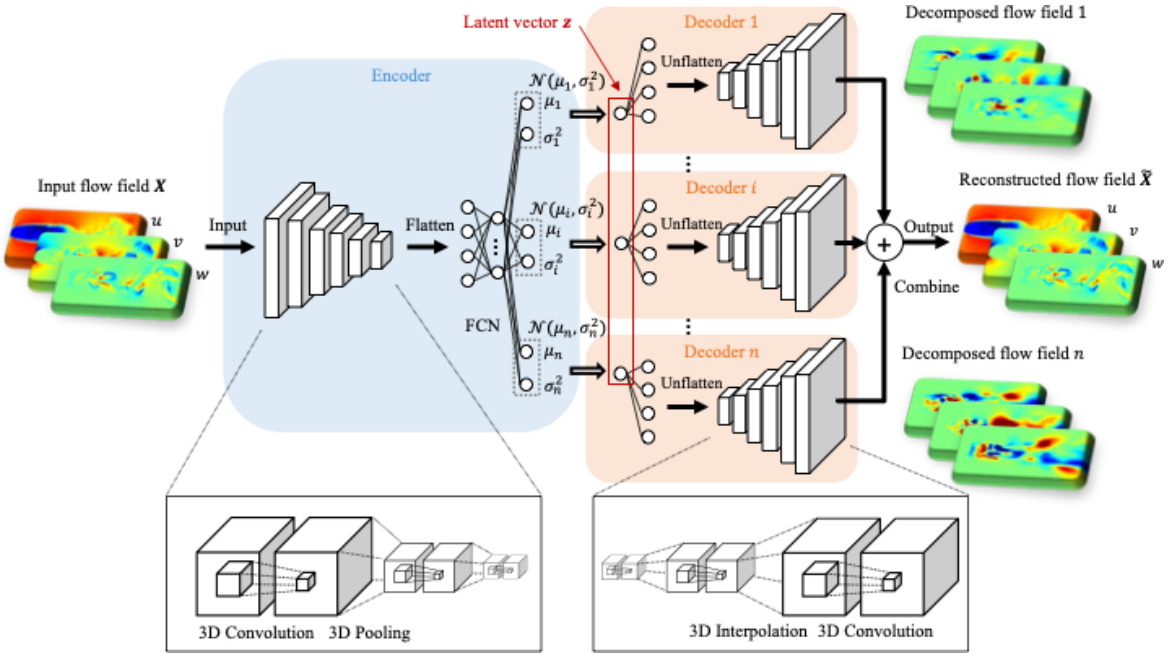

Figure 3. Overview of the neural network architecture, including the encoder and branched decoder.

The first half of the network, called the encoder, reduces the high-dimensional flow field data and outputs a low-dimensional representation called the latent vector. The encoder consists of multiple layers of blocks composed of three-dimensional convolutional layers and pooling layers. Additionally, to stabilize training, layer normalization is applied to the output of each layer.

In contrast, the latter half of the network, called the decoder, is branched, with each branch having its own parameters. Each branch expands the dimension from a single element of the latent vector to a decomposed flow field, which has the same dimensions as the original high-dimensional flow field data. The decoder consists of multiple layers of blocks composed of convolutional layers and interpolation layers. Finally, the decomposed flow fields are combined, producing the reconstructed flow field. Training is performed to minimize the error between the original and reconstructed flow fields. Once training has sufficiently progressed, the network can effectively map high-dimensional flow field data to a low-dimensional representation.

This procedure corresponds to the decomposition of the flow field into multiple nonlinear modes, with mode information preserved as network parameters. The number of decompositions in the decoder is a user-defined hyperparameter, which is equal to the dimensionality of the latent vector. The smaller this value, the higher the reduction (compression) efficiency. However, there is a trade-off: reducing the latent dimension decreases reconstruction accuracy. It should be noted that a large number of branches (modes) is required to efficiently reduce highly nonlinear flows, such as high-Reynolds-number turbulent flows. The detailed network structure is provided in Table A.1 and A.2 in Appendix A.

The optimization problem solved during training is formulated as follows:

$$\boldsymbol{\phi}, \{\boldsymbol{\theta}_i\}_{i=1}^{n} = \underset{\boldsymbol{\phi}, \{\boldsymbol{\theta}_i\}_{i=1}^{n}}{\operatorname{argmin}} \sum_{t_{min}}^{t_{max}} L(\boldsymbol{X}(t), \tilde{\boldsymbol{X}}(t), \boldsymbol{\phi}, \{\boldsymbol{\theta}_i\}_{i=1}^{n}), \tag{3}$$

$$\tilde{\boldsymbol{X}}(t) = \sum_{i=1}^{n} \boldsymbol{\mathcal{F}}_{dec}(z_i(t); \boldsymbol{\theta}_i), \tag{4}$$

$$\boldsymbol{z}(t) \sim \mathcal{N}(\boldsymbol{z}(t); \boldsymbol{\mu}(t), \boldsymbol{\sigma}(t)^2), \tag{5}$$

$$\boldsymbol{\mu}(t), \boldsymbol{\sigma}(t)^2 = \boldsymbol{\mathcal{F}}_{enc}(\boldsymbol{X}(t); \boldsymbol{\phi}), \tag{6}$$

where $\boldsymbol{\phi}$ represents the network parameters of the encoder, $\boldsymbol{\theta}_i$ is the network parameters of the decoder corresponding to the i-th mode, and n is the number of modes (i.e., the number of decoders or the number of latent variables). t_{min} and t_{max} denote the start and end points of the training time range. $\boldsymbol{X}(t)$ is the high-precision simulation result of the flow field (i.e., the input data of the network) at time t, while $\tilde{\boldsymbol{X}}(t)$ is the reconstructed flow field (i.e., the output data of the network) at time t. $\boldsymbol{\mathcal{F}}_{dec}$ represents the decoder neural network, and $z_i(t)$ is the latent variable corresponding to the i-th mode at time t. $\mathcal{N}(\boldsymbol{z}(t);\ \boldsymbol{\mu}(t),\ \boldsymbol{\sigma}(t)^2)$ is the normal distribution with mean $\boldsymbol{\mu}(t)$ and variance $\boldsymbol{\sigma}(t)^2$, which represents the probability distribution of the latent vector $\boldsymbol{z}(t)$. $\boldsymbol{\mathcal{F}}_{enc}$ represents the encoder neural network.

The loss function is formulated as follows:

$$L(\boldsymbol{X}(t), \tilde{\boldsymbol{X}}(t), \boldsymbol{\phi}, \{\boldsymbol{\theta}_i\}_{i=1}^{n}) = \frac{\|\boldsymbol{X}(t) - \tilde{\boldsymbol{X}}(t)\|_2^2}{N} + D_{KL}(q_\phi(\boldsymbol{z}(t)|\boldsymbol{X}(t)) \| p(\boldsymbol{z}(t))), \tag{5}$$

$$D_{KL}(q_\phi(\boldsymbol{z}(t)|\boldsymbol{X}(t)) \| p(\boldsymbol{z}(t))) = \int q_\phi(\boldsymbol{z}(t)|\boldsymbol{X}(t)) \log \frac{q_\phi(\boldsymbol{z}(t)|\boldsymbol{X}(t))}{p(\boldsymbol{z}(t))} d\boldsymbol{z}, \tag{6}$$

$$p(\boldsymbol{z}(t)) = \mathcal{N}(\boldsymbol{z}(t); \mathbf{0}, \mathbf{I}), \tag{7}$$

where D_{KL} represents the Kullback–Leibler (KL) divergence, $q_\phi(\boldsymbol{z}(t)|\boldsymbol{X}(t))$ is the posterior distribution of the latent vector $\boldsymbol{z}(t)$ given the input flow field $\boldsymbol{X}(t)$, and $p(\boldsymbol{z}(t))$ is the standard normal distribution, representing the prior probability distribution of the latent vector $\boldsymbol{z}(t)$.

During the inference phase, the input flow field data is reconstructed using the trained network parameters $\boldsymbol{\phi}$ and $\{\boldsymbol{\theta}_i\}_{i=1}^{n}$as follows:

$$\tilde{\boldsymbol{X}}(t) = \sum_{i=1}^{n} \boldsymbol{\mathcal{F}}_{dec}(\mu_i(t); \boldsymbol{\theta}_i), \tag{8}$$

$$\boldsymbol{\mu}(t), \boldsymbol{\sigma}(t)^2 = \boldsymbol{\mathcal{F}}_{enc}(\boldsymbol{X}(t); \boldsymbol{\phi}). \tag{9}$$

Note: In the inference phase, the latent variable used as input to the decoder is the mean of the normal distribution representing the probability of the latent variables. This means that the decoder operates deterministically, rather than using probabilistic sampling.

Distributed parallel training

All the experiments in this study were conducted on the Supercomputer Fugaku (hereafter, Fugaku). Fugaku is equipped with a single Fujitsu A64FX CPU per node. The A64FX processor has 48 computational cores, each achieving 128 GFLOPS in single-precision floating-point arithmetic. Consequently, each node delivers 6.1 TFLOPS of single-precision performance.

For comparison, a high-end GPU, the NVIDIA H100 NVL [22], attains 60 TFLOPS for single precision (excluding tensor-core operations), which is approximately 10 times higher than the performance of a single A64FX processor. To close this gap, it is necessary to utilize a large number of nodes on Fugaku, requiring an efficient distributed (inter-node) parallel training mechanism.

To address this issue, in our previous work, a distributed parallel training framework was implemented using hybrid parallelism, which combines data parallelism and network architecture-specific model parallelism [10]. This framework has demonstrated scalability, allowing training execution to scale up to tens of thousands of nodes (millions of cores).

Table 2 presents the execution settings for distributed training. Notably, this study aims to reconstruct a flow field represented by 20 million variables using only 32 latent variables. Additionally, two experiments are conducted for comparison:

- Experiment 1: Training on flow fields around five vehicle body shapes.
- Experiment 2: Training on flow fields around two vehicle body shapes.

A detailed explanation of the hyperparameters used for neural network training is provided in Table A.3 in Appendix A.

Table 2. Execution settings for distributed training.

Target physical variables for training	Three-dimensional flow velocity (u, v, w)
Target region	0.6L × 0.4L × 0.3L
Grid resolution	384 × 288 × 192 = 21,233,664
Number of latent variables (i.e., number of decomposition modes)	32
Grid spacing	7.0 mm (uniform resolution)

Snapshot time interval	0.5 ms
Shapes used for inference	Experiment 1: PC1 = -1.5, -0.9, -0.3, 0.3, 0.9, 1.5 Experiment 2: PC1 = -1.5, 1.5
Shapes for inference	PC1 = -1.2, -0.6, 0.0, 0.6, 1.2
Training samples per shape	492
Total training samples	Experiment 1: 492 × 6 = 2,952 Experiment 2: 492 × 2 = 984
Total training iterations	12,000 (2,000 epochs)
Number of snapshots generated during inference	500
Integration time for inference	0.25 s

Figure 4 illustrates which of the 11 vehicle body shapes were used for training or testing in each experiment.

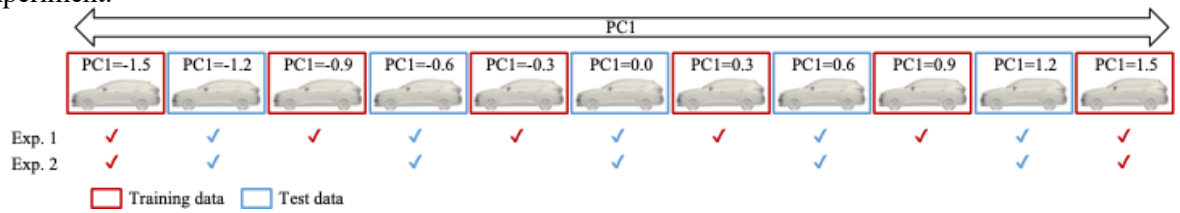

Figure 4. Vehicle body shapes used for training or testing

Furthermore, the computational resources required during training are summarized in Table 3. These experiments demanded substantial computational power: Experiment 1 consumed 1 million node hours, while Experiment 2 required 340,000 node hours on Fugaku.

Table 3. Computational resource requirements for training.

Number of compute nodes used	Experiment 1: 8,118 nodes (389,664 cores) Experiment 2: 2,706 nodes (129,888 cores)
Theoretical peak single-precision performance	Experiment 1: 6.1 TFLOPS × 8,118 nodes = 49.8 PFLOPS Experiment 2: 6.1 TFLOPS × 2,706 nodes = 16.6 PFLOPS
Training time per epoch	Experiment 1: 48 s Experiment 2: 46 s
Total training iterations	60,000 iterations (10,000 epochs)
Total training time	Experiment 1: (48 / 3600) hour × 10,000 epochs = 133 hours Experiment 2: (46 / 3600) hour × 10,000 epochs = 127 hours
Estimated total computational cost	Experiment 1: 8,118 nodes × 133 hours = 1.0×10^6 node-hours (corresponds to 6.6 ExaFLOP) Experiment 2: 2,706 nodes × 127 hours = 3.4×10^5 node-hours (corresponds to 2.2 ExaFLOP)

3. Results and discussion

This chapter presents the evaluation results of the proposed model reduction method for the two experiments. The first experiment, referred to as “Experiment 1,” evaluates model reduction using a network trained on flow fields around six vehicle body shapes. The second experiment, referred to as

"Experiment 2," evaluates model reduction using a network trained on flow fields around two vehicle body shapes.

3.1. Training Progress and Convergence Status

Figure 5 illustrates the decay of the training error as the number of training iterations progresses in Experiment 1 and Experiment 2. The training error decreases monotonically with each iteration; however, at the current stage of training (12,000 iterations), the decrease in training error remains incomplete and has not yet saturated in either experiment. Therefore, the accuracy of the model reduction presented in this chapter can be further improved by continuing the training.

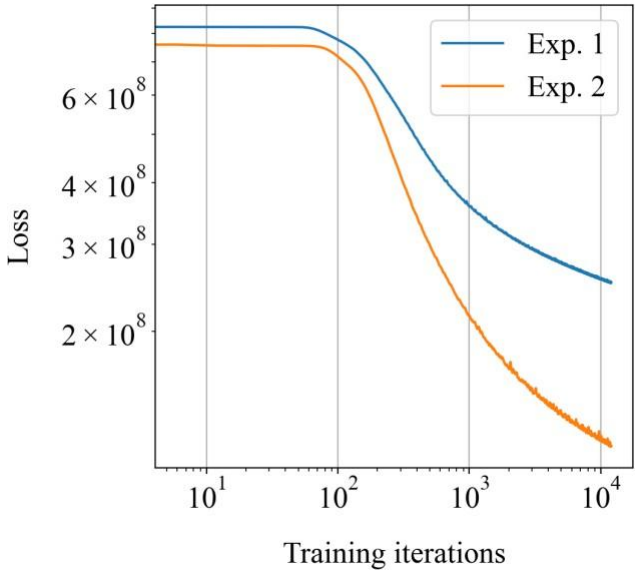

Figure 5. Training convergence.

3.2. Extracted modes

Figure 6 illustrates the time variation of selected latent vector elements corresponding to the flow field around the PC1_0.0 shape as an example. This value corresponds to $z_i(t)$ in Equation (4).

Comparing the results of Experiment 1 and Experiment 2, some elements, such as z_1 exhibit correlations between the values from both experiments. In contrast, certain elements, such as z_4, appear to have no correlation. This is likely because the spatial scale of flow fluctuations associated with a specific mode number depends on runtime conditions.

For the graphs of all latent variables, see Figure B.1 in Appendix B.

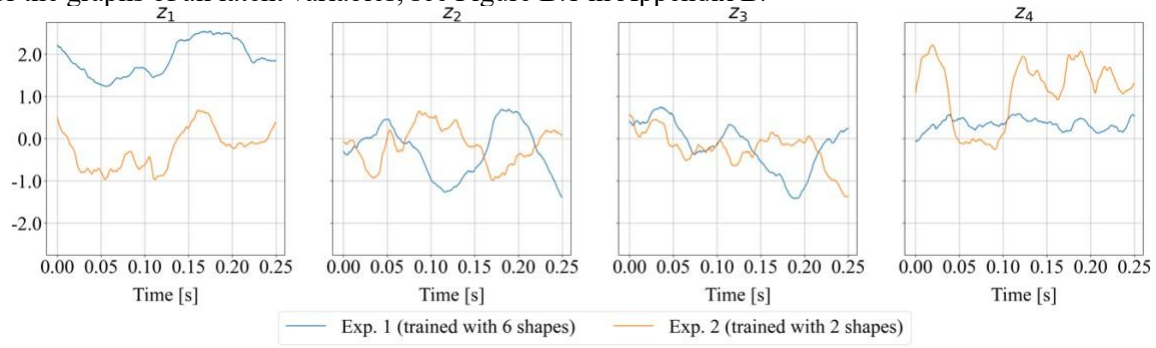

Figure 6. Representative time variation of latent vector elements for PC1_0.0.

Figure 7 presents a snapshot of the fluctuation component of the decomposed flow field, consisting of the three directional components of the flow velocity vector, corresponding to the first mode, i.e., $\boldsymbol{\mathcal{F}}_{dec}(\mu_1(t);\boldsymbol{\theta}_1)$ in Equation (8).

In Experiment 1, the flow field structure appears physically natural, exhibiting a spatially continuous distribution. In contrast, the result of Experiment 2 shows a spatially discontinuous distribution, which is physically unnatural.

For the decomposed flow field corresponding to all modes, see Figure B.2 and B.3 in Appendix B.

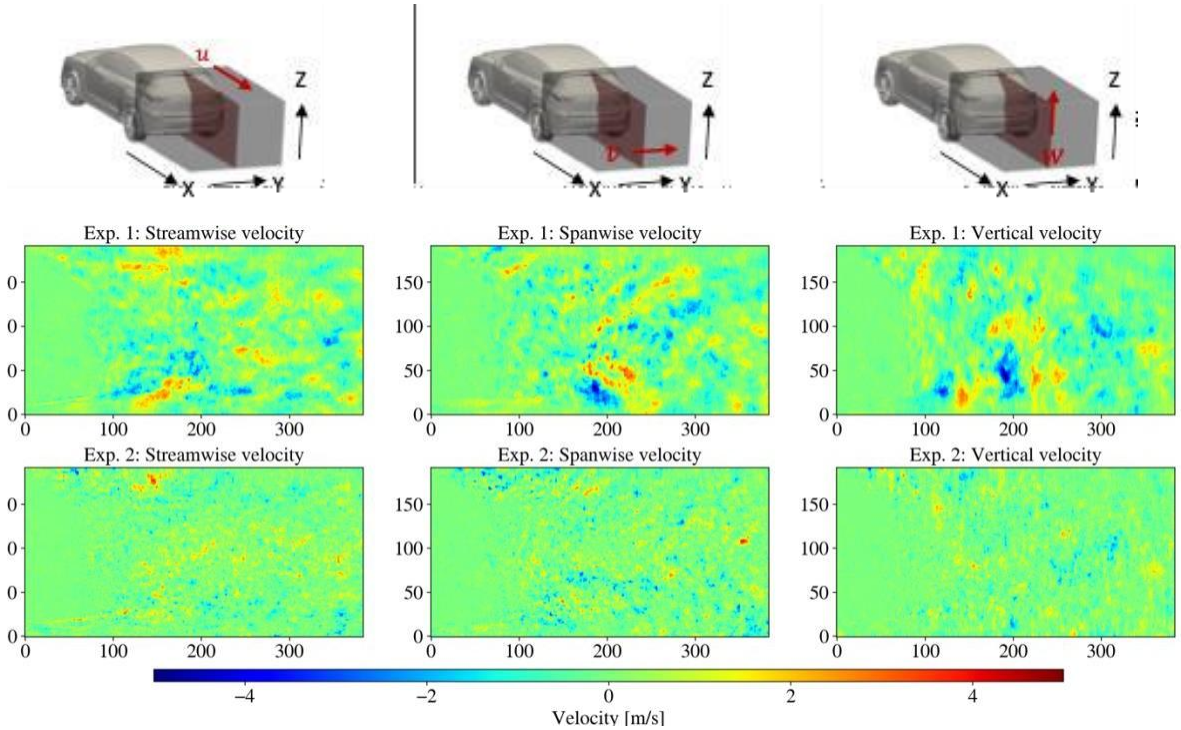

Figure 7. Snapshot of the fluctuation component of the decomposed flow field for PC1_0.0.

3.3. Evaluation of Model Reduction Accuracy

This section evaluates the accuracy of the model reduction, i.e., how well the original flow field—which was not used for training—can be reconstructed from 32 latent variables using the proposed method.

First, the reconstructed flow fields, including the three directional components of the flow velocity vector, $\tilde{\boldsymbol{X}}(t)$ in Equation (4), are compared between Experiment 1 and Experiment 2. Additionally, the vorticity in the y-direction (ω_y), computed from this velocity field, is also analyzed. As an example, the flow field around shape PC1_0.0 is shown in Figure 8.

For the flow velocity field, both Experiment 1 and Experiment 2 successfully reproduce the overall flow structure observed in the reference high-precision simulation results. However, in Experiment 2, as observed in the decomposed flow field described earlier, the spatial continuity of the flow velocity is lost, resulting in a physically less realistic flow field. This suggests that Experiment 1 achieves higher reduction accuracy for the velocity field.

On the other hand, in the vorticity field, both Experiment 1 and Experiment 2 reproduce the clockwise vortex near the roof of the car body and the counterclockwise vortex near the bottom. However, it

appears that the absolute value of the vorticity is more accurately reproduced in Experiment 2 than in Experiment 1.

For the complete flow fields around other shapes, see Figures C.1–C.4 in Appendix C.

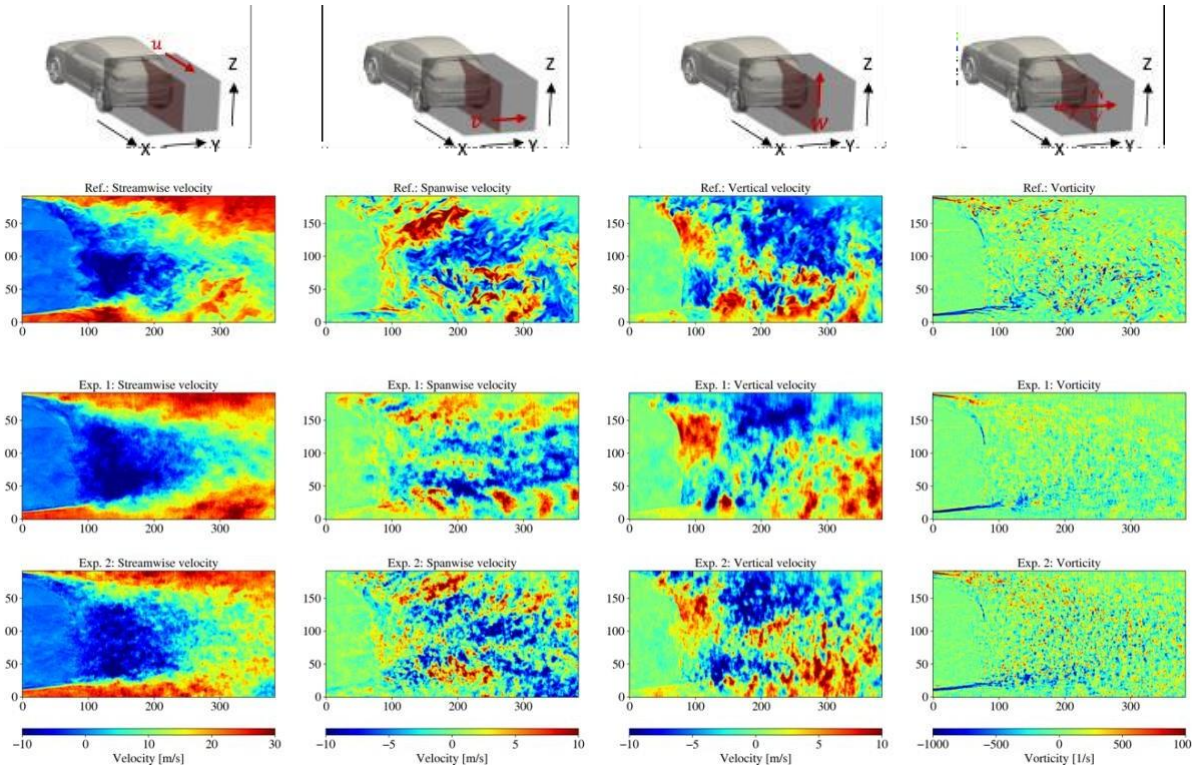

Figure 8. Snapshot of the reconstructed flow fields from 32 latent variables for PC1_0.0.

Furthermore, to confirm the three-dimensional structure of the vortex, the instantaneous flow field of the Q-criterion (the second invariant of the velocity gradient tensor) isosurface for PC1_0.0 is shown in Figure 9. The Q-criterion is formulated as follows:

$$Q = \frac{1}{2}(\|\Omega\|^2 - \|S\|^2), \tag{10}$$

where Ω and S represent the rate of rotation tensor and the rate of strain tensor, respectively.

The results of Experiment 1 indicate that the three-dimensional vortex structures observed in the high-precision simulation results are qualitatively reproduced. In contrast, the results of Experiment 2 deviate from the high-precision simulation results, as the vortices appear finer in scale and are more uniformly distributed throughout the computational domain.

For the Q-criterion isosurfaces around other shapes, see Figures D.1–D.4 in Appendix D.

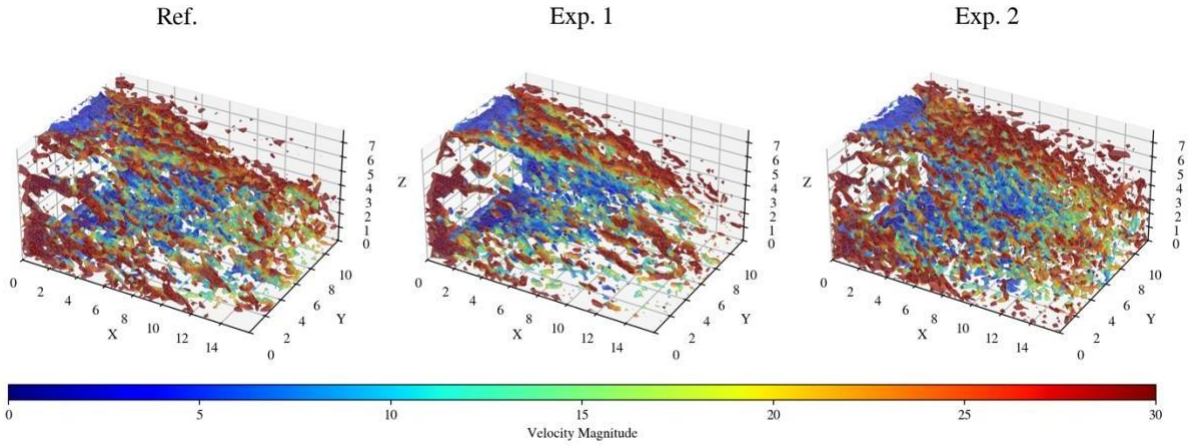

Figure 9. Instantaneous Q-criterion isosurface colored by velocity magnitude for PC1_0.0.

Figure 10 illustrates the energy spectrum at the probe point located at the rear end of the vehicle body. In Experiment 1, the amplitude of the high-frequency components more closely matches the high-precision simulation results than in Experiment 2. This indicates that Experiment 1 achieves higher reproduction accuracy for eddies with small time scales compared to Experiment 2.

For the spectra corresponding to other shapes, see Figure E.1 in Appendix E.

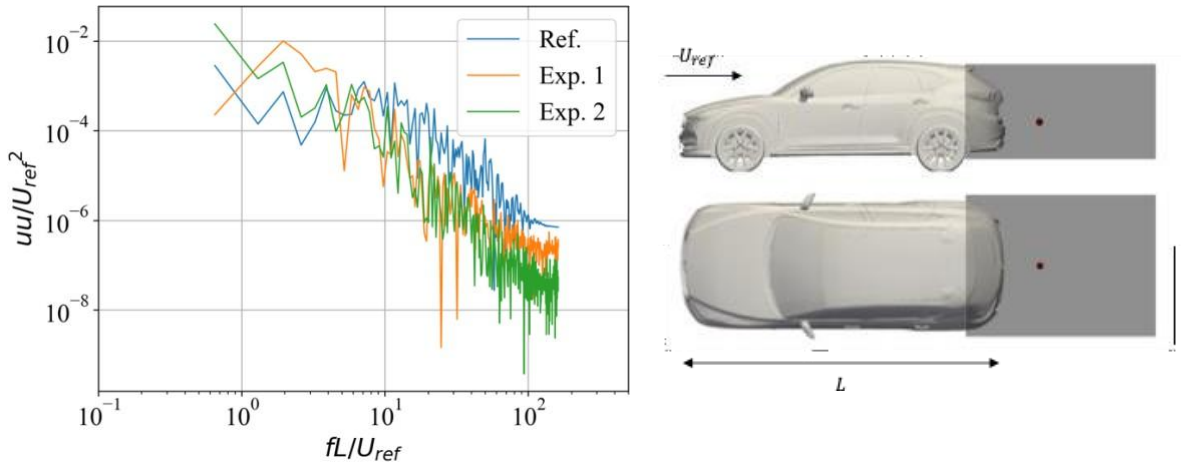

Figure 10. Energy spectrum (left) and probe point location (red dot in the right figure).

Figure 11 illustrates the probability density function (PDF) of vorticity for PC1_0.0.

For ω_y, in both Experiment 1 and Experiment 2, the reproduction accuracy is low in regions with large absolute values, corresponding to small, fast-rotating vortices.

In contrast, for ω_z, the probability density in the region with large absolute values of ω_z in Experiment 2 shows good agreement with the reference high-precision simulation results. This is likely because there is no shape variation between the left and right sides of the vehicle body, resulting in little difference in the z-axis component of vorticity ω_z between different shapes. Consequently, adding training shapes between PC1_-1.5 and PC1_1.5 leads to overfitting in reproducing ω_z.

For the PDFs corresponding to other shapes, see Figure F.1 in Appendix F.

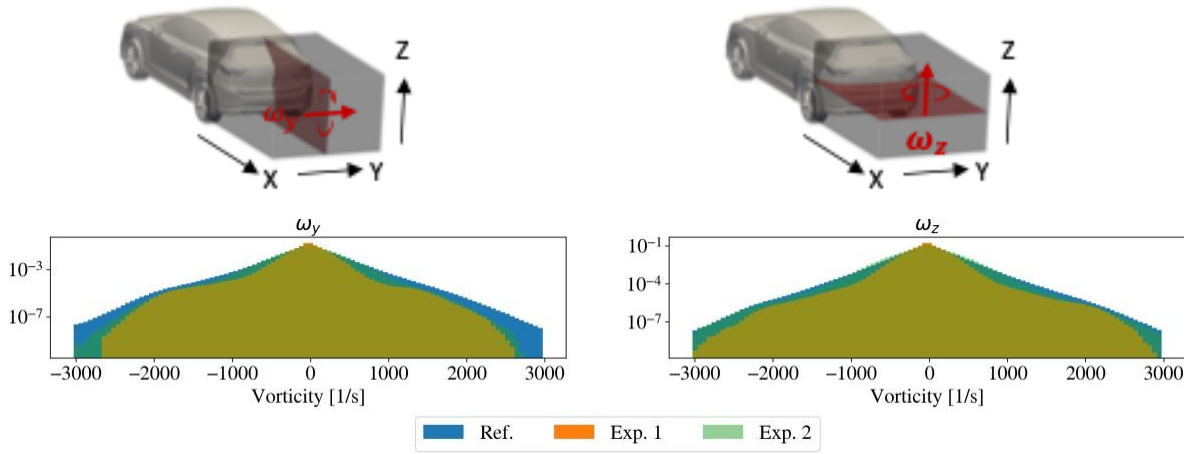

Figure 11. Probability density function of vorticity for PC1_0.0.

Figure 12 illustrates the trajectory of a physical system in phase space, spanned by three variables: turbulent kinetic energy (TKE), TKE production, and TKE dissipation for PC1_0.0. These quantities are formulated as follows:

Turbulent Kinetic Energy (TKE):

$$\int_V k dV = \int_V (u'^2 + v'^2 + w'^2) dV, \tag{11}$$

Production of TKE:

$$\int_V P dV = \int_V -u'v' \frac{dU}{dy} dV \int_V k dV, \tag{12}$$

Dissipation of TKE:

$$\int_V D dV = \int_V \nu(\|\nabla u'\|^2 + \|\nabla v'\|^2 + \|\nabla w'\|^2)\, dV, \tag{13}$$

where k represents the TKE, P is the TKE production rate, D is the TKE dissipation rate, V is the target training region, and U is the mean streamwise velocity.

In both Experiment 1 and Experiment 2, the trajectories in phase space did not match the results of the high-precision simulation, indicating that the generation and dissipation processes of turbulence-induced vortices were not well reproduced.

For the time variation of TKE corresponding to other shapes, see Figure G.1 in Appendix G.

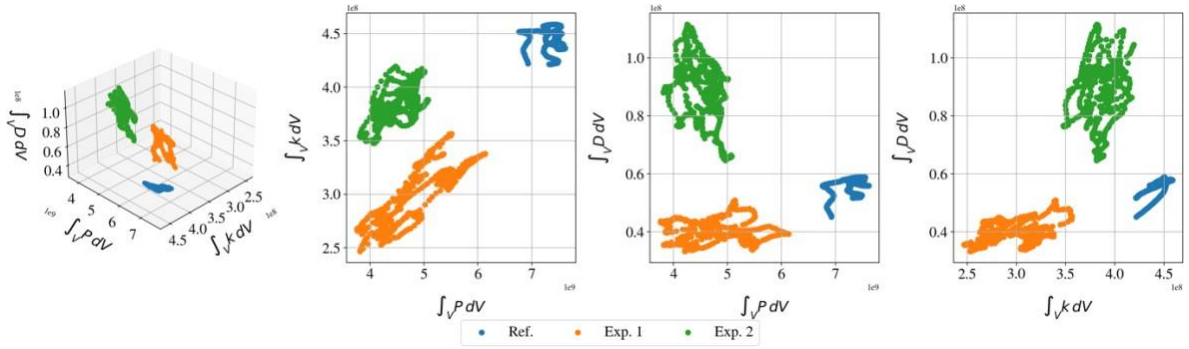

Figure 12. Phase space trajectory of the physical system, defined by turbulent kinetic energy (TKE), TKE production, and TKE dissipation, for PC1_0.0.

Figure 13 illustrates the temporal correlation coefficient for PC1_0.0. The temporal correlation coefficient is formulated as follows:

$$R(\Delta t) = \frac{\langle u'(t)u'(t\ +\ \Delta t)\rangle}{\langle u'(t)^2\rangle}, \tag{14}$$

where R represents the temporal correlation coefficient, $\langle\cdots\rangle$ denotes the spatial average, u' is the fluctuation component of the streamwise velocity, t denotes the initial time, and Δt is the time interval from the initial state.

Regarding the time variation of the time correlation coefficient, the results of Experiment 1 are closer to the high-precision simulation results than those of Experiment 2. This suggests that the time scale in which the initial state of the system is disturbed by vortex generation due to turbulence is more accurately reproduced in Experiment 1 than in Experiment 2.

For the correlation coefficients corresponding to other shapes, see Figure H.1 in Appendix H.

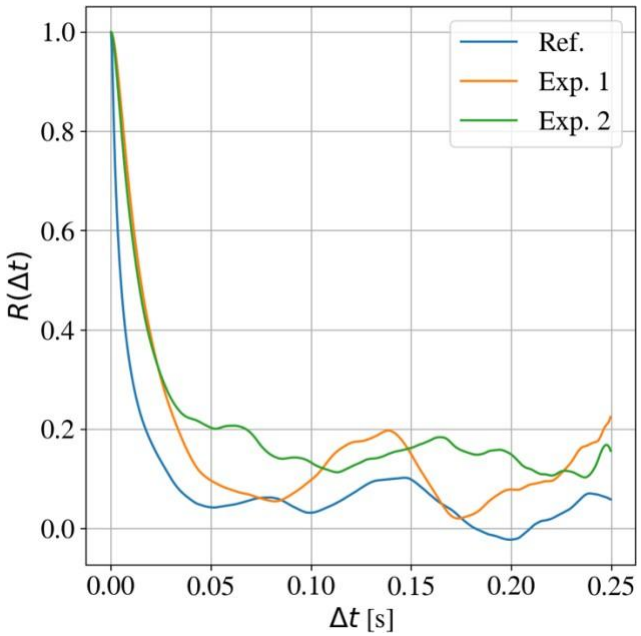

Figure 13. Temporal correlation coefficient for PC1_0.0.

4. Summary and conclusions

In this study, we applied a new variational autoencoder (VAE) mechanism to our distributed parallel training-based model reduction framework for high-precision flow field data, developed in our previous work. We focused on the region near the rear end of the vehicle body to evaluate its robustness in handling high-Reynolds-number flows around unknown vehicle shapes.

Specifically, we designed two experiments, where the flow fields around either five or two vehicle shapes were used for training. We then evaluated whether vortex generation at various spatial and temporal scales could be reconstructed from 32 latent variables for flow fields around shapes not included in training. In the largest experiment, training was conducted for 26 hours using 8,118 nodes on Fugaku.

As a result, both experiments demonstrated high accuracy in reproducing large-scale vortices in space and time; however, challenges remained in capturing small-scale vortices. Additionally, for y-direction vortices, which are strongly affected by shape variations at the top and bottom of the vehicle body, training with five shapes yielded higher reproduction accuracy than training with two shapes. In contrast, for z-direction vortices, which are less influenced by shape changes, reducing the number of training shapes helped mitigate overfitting, leading to better reproduction accuracy.

For future work, we aim to enhance the network architecture to further improve the accuracy of small-scale vortex reconstruction and optimize hyperparameters, including determining the appropriate number of decomposition modes based on vortex scale.

Acknowledgments

This research is an extension of the method originally proposed by Mr. Takaaki Murata, Dr. Kai Fukami, and Professor Koji Fukagata of Keio University. We sincerely appreciate their valuable contributions, which laid the foundation for this study, as well as their insightful advice throughout the research.

We also extend our gratitude to Professor Takuji Nakashima from Hiroshima University for providing high-quality vehicle body models and for engaging in fruitful discussions.

The computational resources of “Fugaku” were provided through the HPCI System Research Project (Project ID: hp240205 and ra240008).

Appendices

Appendix A

The detailed network architectures for the encoder and decoder are presented in Tables A.1 and A.2

Table A.1. Layer-wise details of the encoder network architecture.

Layer type		Output shape (X, Y, Z, channel)
Input layer		(384, 288, 192, 3)
1st	3D convolutional layer	(384, 288, 192, 4)
	Layer normalization	(384, 288, 192, 4)
	3D max pooling layer	(192, 144, 96, 4)
2nd	3D convolutional layer	(192, 144, 96, 2)
	Layer normalization	

	3D max pooling layer	(96, 72, 48, 2)
3rd	3D convolutional layer	(96, 72, 48, 2)
	Layer normalization	(96, 72, 48, 2)
	3D max pooling layer	(48, 36, 24, 2)
4th	3D convolutional layer	(48, 36, 24, 1)
	Layer normalization	(48, 36, 24, 1)
	3D max pooling layer	(24, 18, 24, 1)
5th	3D convolutional layer	(24, 18, 24, 3)
	Layer normalization	(24, 18, 24, 3)
	3D max pooling layer	(12, 9, 24, 3)
6th	3D convolutional layer	(12, 9, 24, 3)
	Layer normalization	(12, 9, 24, 3)
	3D max pooling layer	(6, 4, 24, 3)
7th	Fully connected layer	(32, 1, 1, 1)
	Layer normalization (latent vector output)	(32, 1, 1, 1)

Table A.2. Layer-wise details of the decoder network architecture.

Layer type		Output shape (X, Y, Z, channel)
Initial input (single element of the latent vector)		(1, 1, 1, 1)
1st	Fully connected layer	(6, 4, 24, 3)
	Layer normalization	(6, 4, 24, 3)
2nd	3D interpolation layer	(12, 9, 24, 3)
	3D convolutional layer	(12, 9, 24, 3)
	Layer normalization	(12, 9, 24, 3)
3rd	3D interpolation layer	(24, 18, 24, 3)
	3D convolutional layer	(24, 18, 24, 1)
	Layer normalization	(24, 18, 24, 1)
4th	3D interpolation layer	(48, 36, 24, 1)
	3D convolutional layer	(48, 36, 24, 2)
	Layer normalization	(48, 36, 24, 2)
5th	3D interpolation layer	(96, 72, 48, 2)
	3D convolutional layer	(96, 72, 48, 2)
	Layer normalization	(96, 72, 48, 2)
6th	3D interpolation layer	(192, 144, 96, 2)
	3D convolutional layer	(192, 144, 96, 2)
	Layer normalization	(192, 144, 96, 2)
7th	3D interpolation layer	(384, 288, 192, 2)
	3D convolutional layer	(384, 288, 192, 3)
	Layer normalization (decomposed flow field)	(384, 288, 192, 3)

The hyperparameters used in this study are presented in Table A.3.

Table A.3. Hyperparameter settings.

Filter size (convolutional layers)	$3 \times 3 \times 3$
Pooling size (max pooling layers)	$2 \times 2 \times 2$
Weight initialization method	Xavier uniform
Optimization algorithm	Adam
Initial learning rate	0.001
Batch size per iteration	492
Activation function	tanh

Appendix B

The time-variation of elements of latent vector are shown in Figure B.1.

Figure B.1. Time variation of all latent vector elements for PC1_0.0.

Here, snapshots of the fluctuating components of the decomposed streamwise flow field around PC1_0.0 for each mode are presented. Figures B.2 and B.3 correspond to Experiment 1 and Experiment 2, respectively.

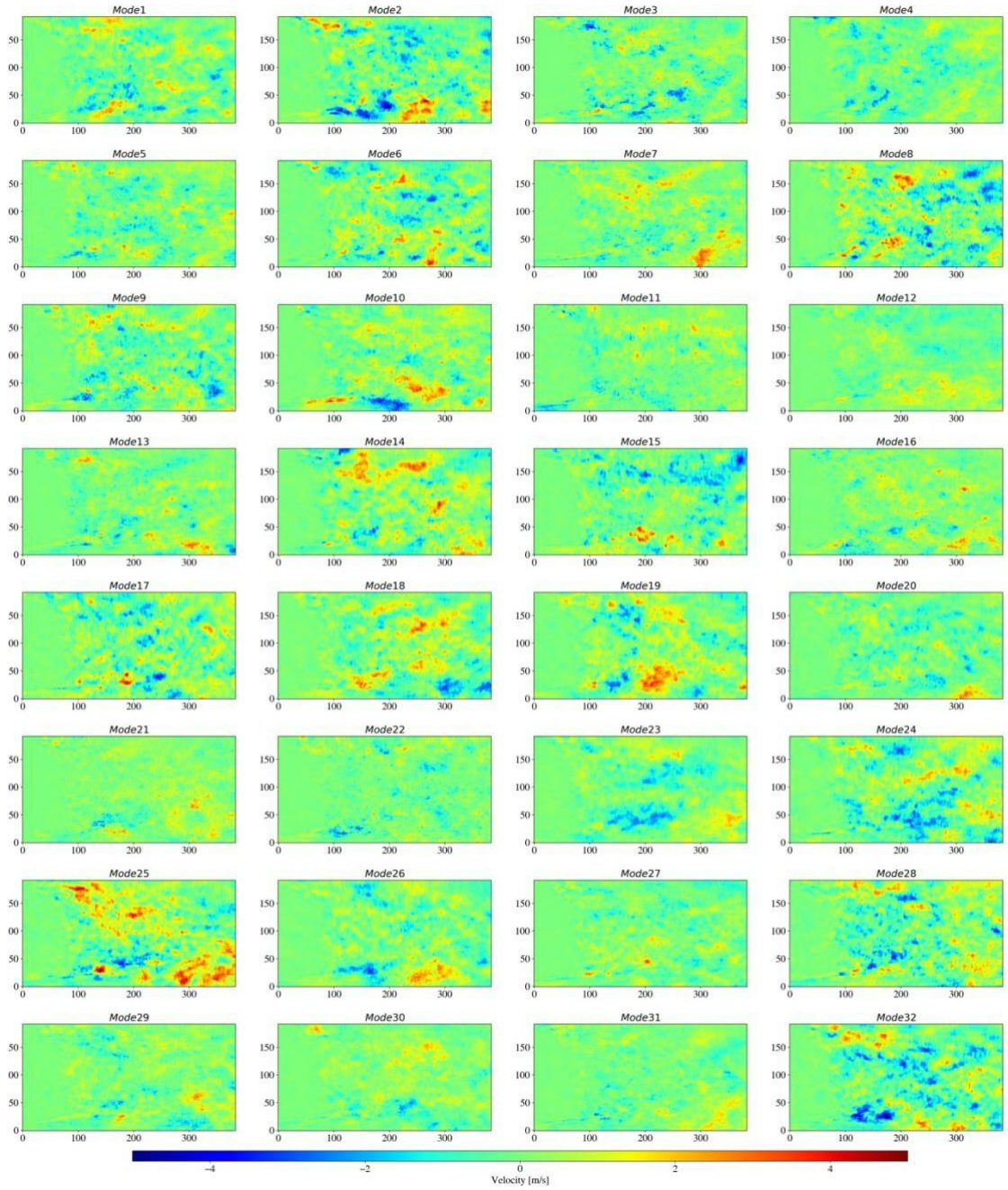

Figure B.2. Snapshot of the fluctuating components of the streamwise flow velocity u for each mode (Exp. 1).

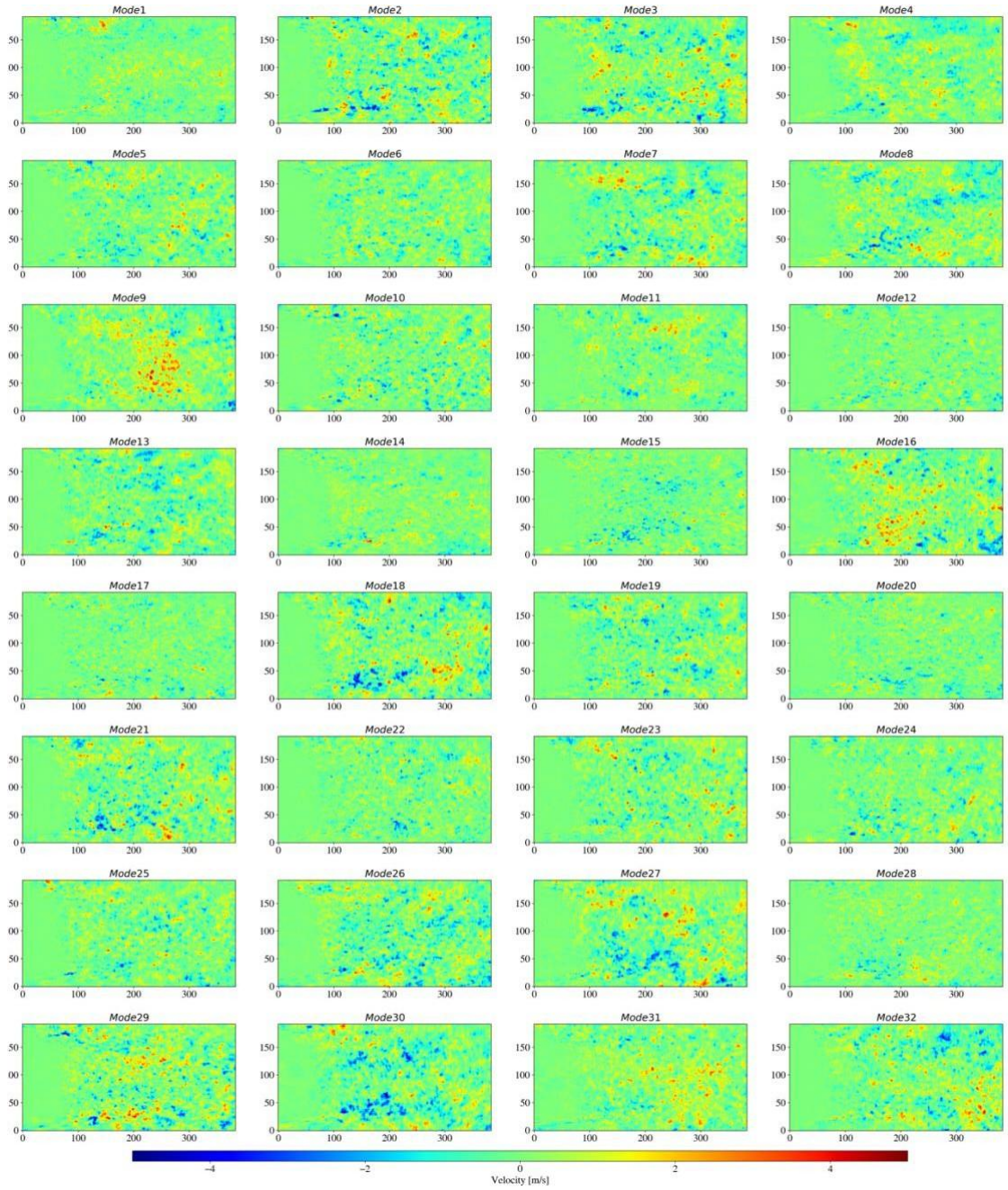

Figure B.3. Snapshot of the fluctuating components of the streamwise flow velocity u for each mode (Exp. 2).

Appendix C

Figure C.1–C.4 presents a snapshot of the reconstructed flow fields from 32 latent variables for all vehicle geometries.

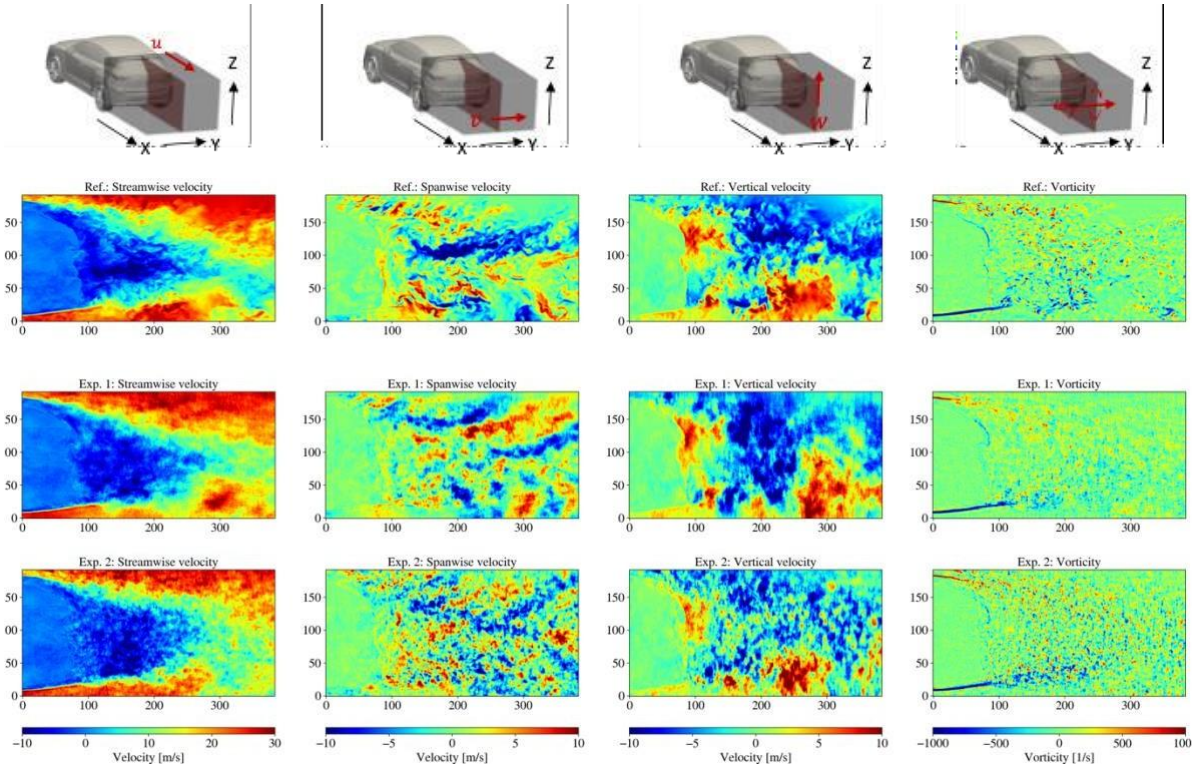

Figure C.1 Snapshot of the reconstructed flow fields from 32 latent variables for PC1_-1.2.

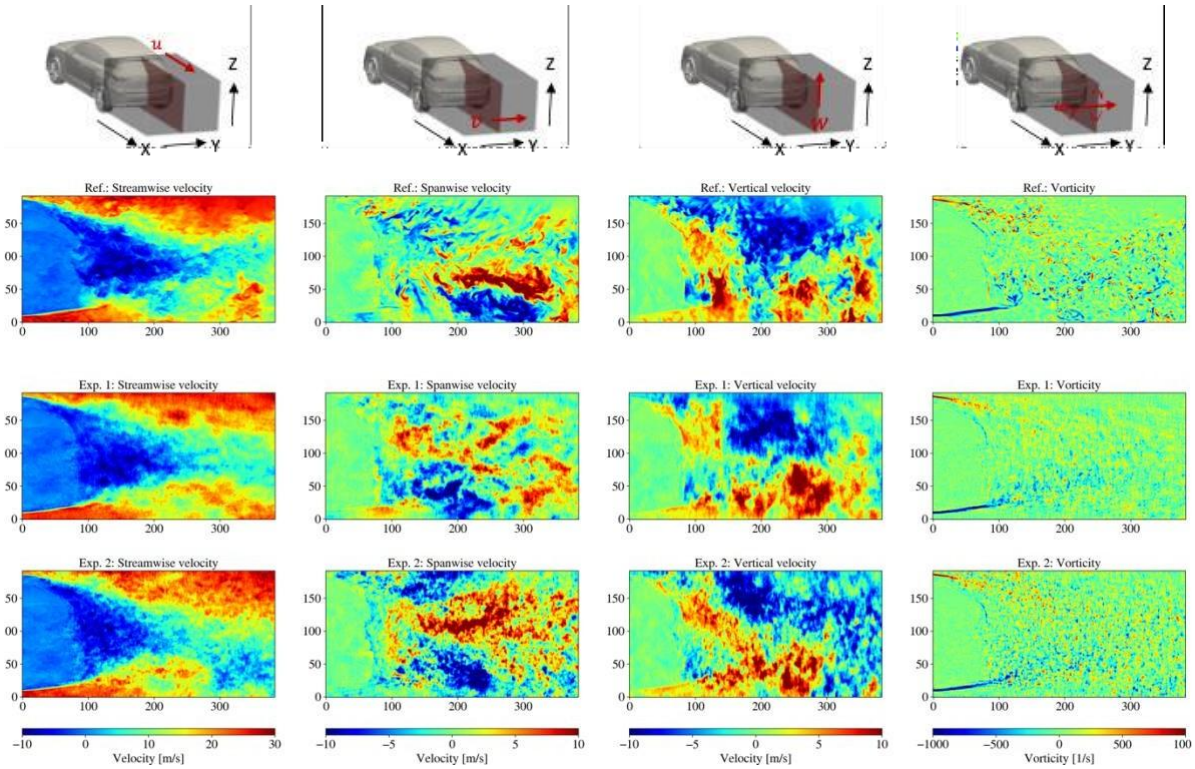

Figure C.2 Snapshot of the reconstructed flow fields from 32 latent variables for PC1_-0.6.

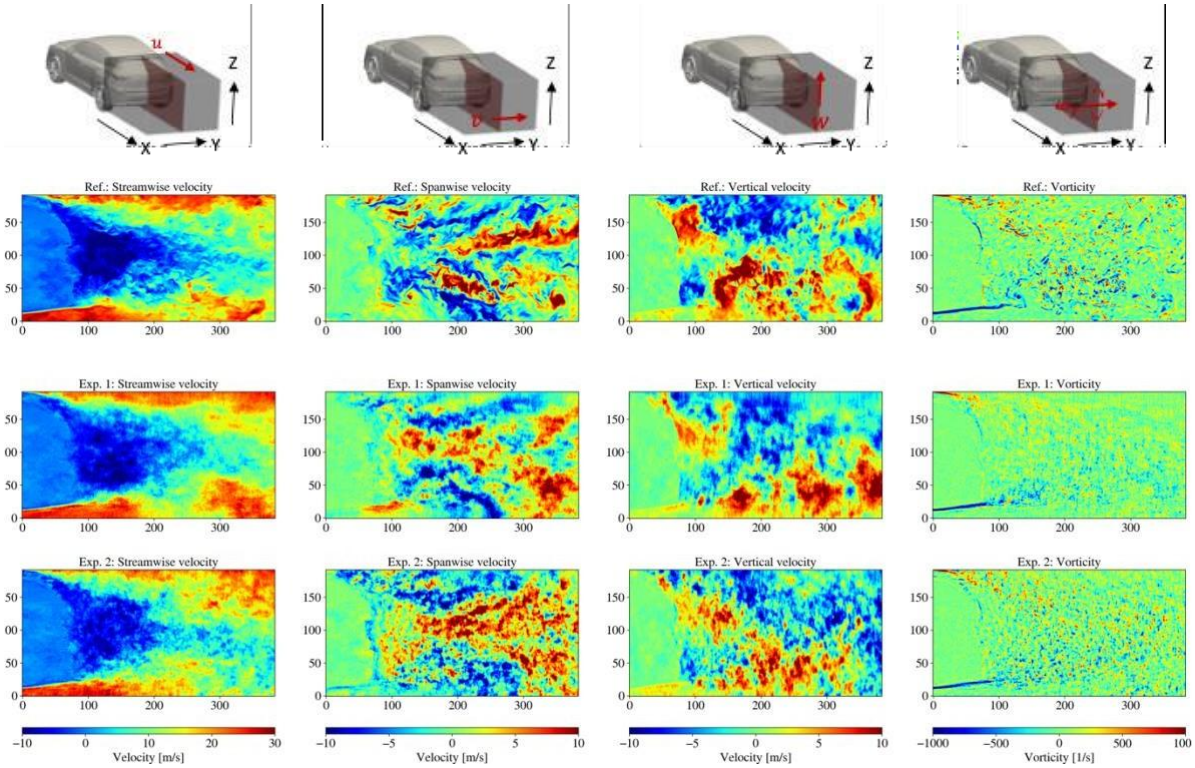

Figure C.3 Snapshot of the reconstructed flow fields from 32 latent variables for PC1_0.6.

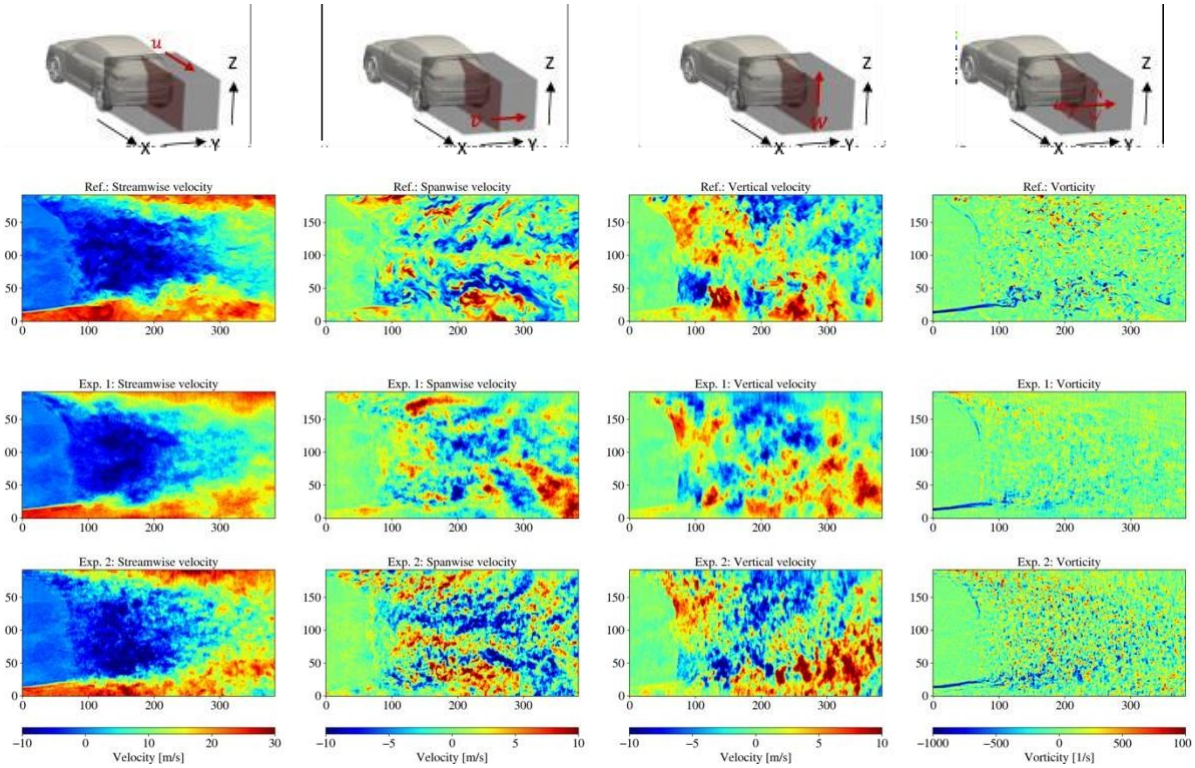

Figure C.4 Snapshot of the reconstructed flow fields from 32 latent variables for PC1_1.2.

Appendix D

Figure D.1 presents a snapshot of the isosurface of the Q-criterion (the second invariant of the velocity gradient tensor).

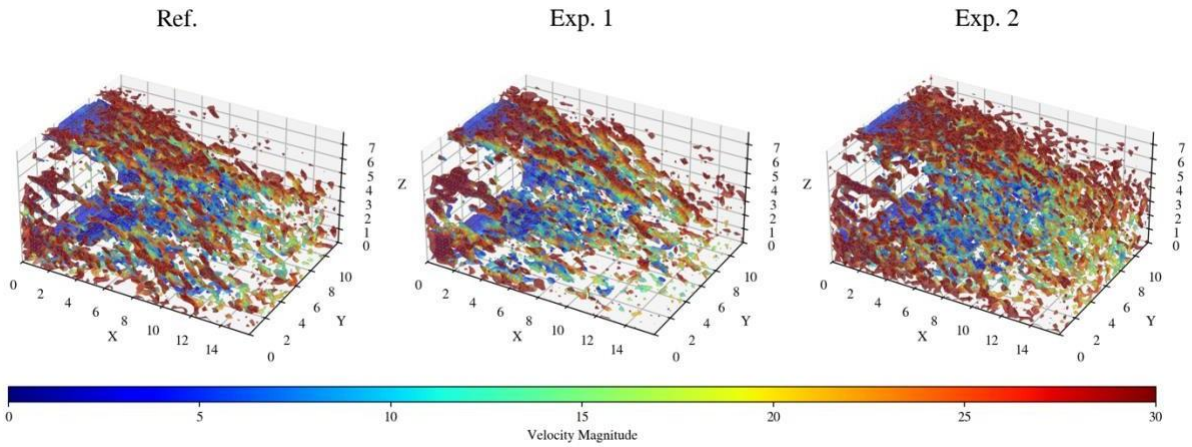

Figure D.1. Instantaneous Q-criterion isosurface colored by velocity magnitude for PC1_-1.2.

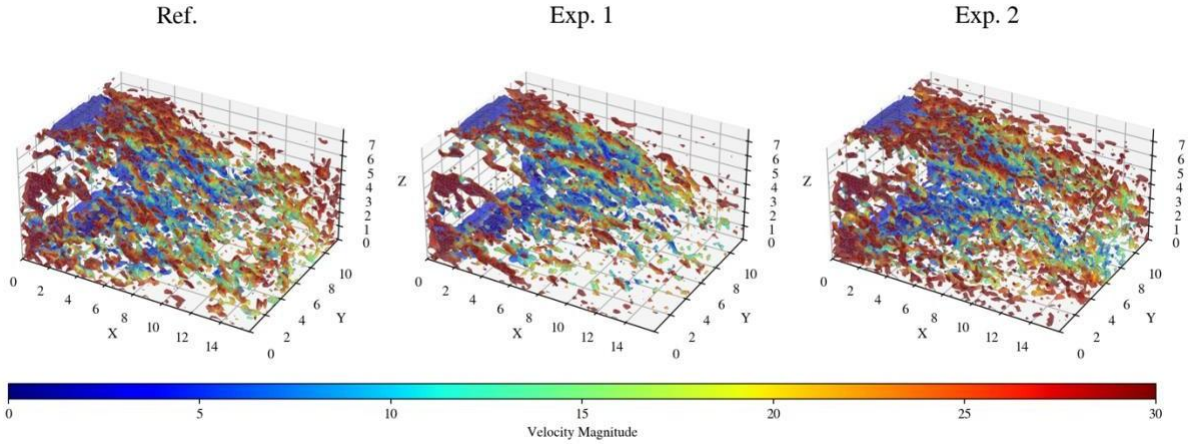

Figure D.2. Instantaneous Q-criterion isosurface colored by velocity magnitude for PC1_-0.6.

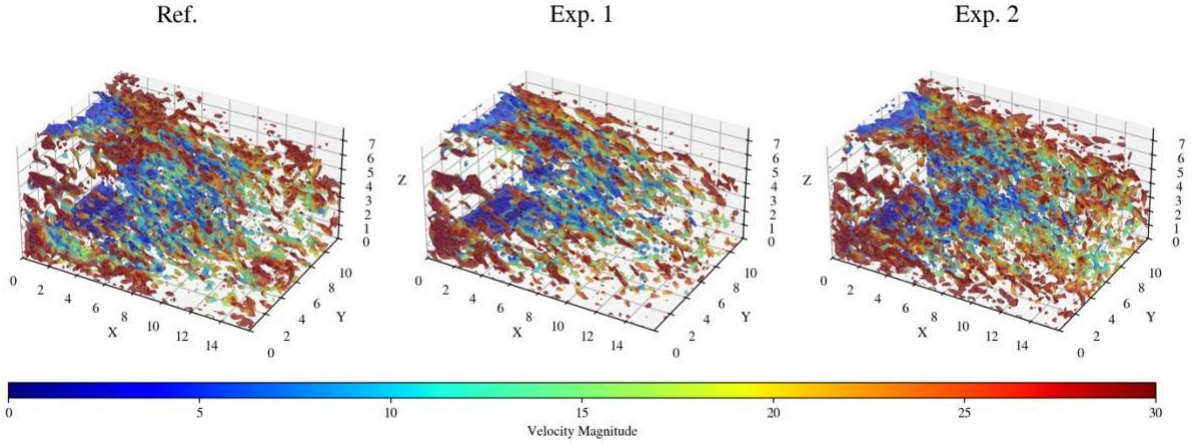

Figure D.3. Instantaneous Q-criterion isosurface colored by velocity magnitude for PC1_0.6.

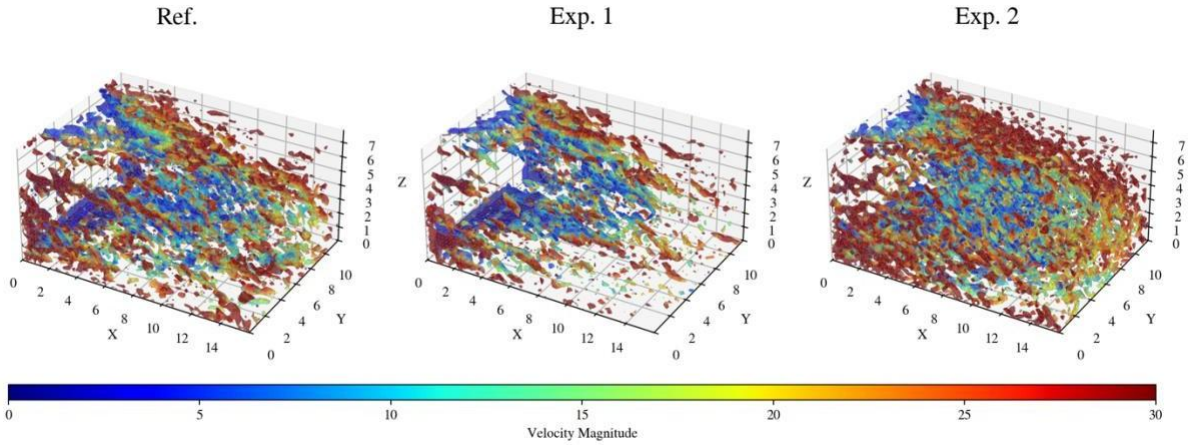

Figure D.4. Instantaneous Q-criterion isosurface colored by velocity magnitude for PC1_1.2.

Appendix E

Figure E.1 presents the energy spectrum of the flow field for all vehicle shape patterns.

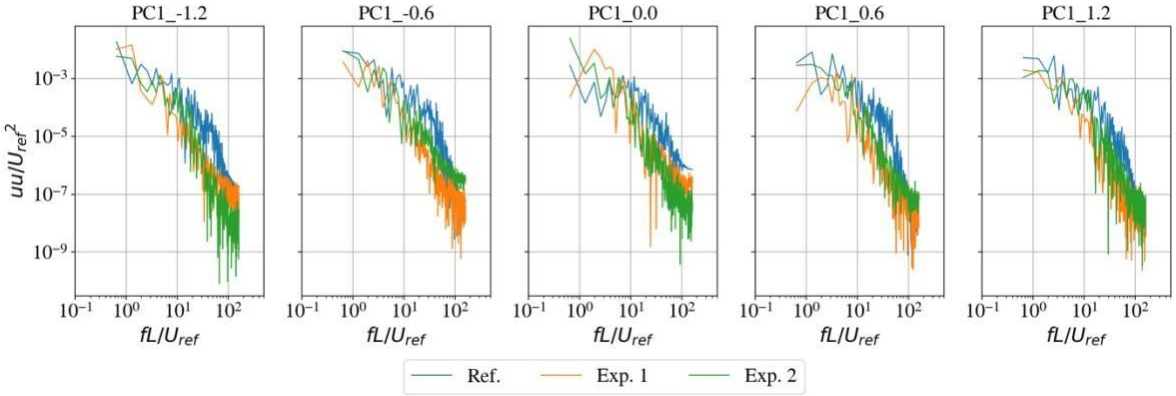

Figure E.1. Energy spectra of the flow fields for all vehicle shape variations.

Appendix F

Figure F.1 presents the probability density function of vorticity for the flow field of all vehicle shape patterns.

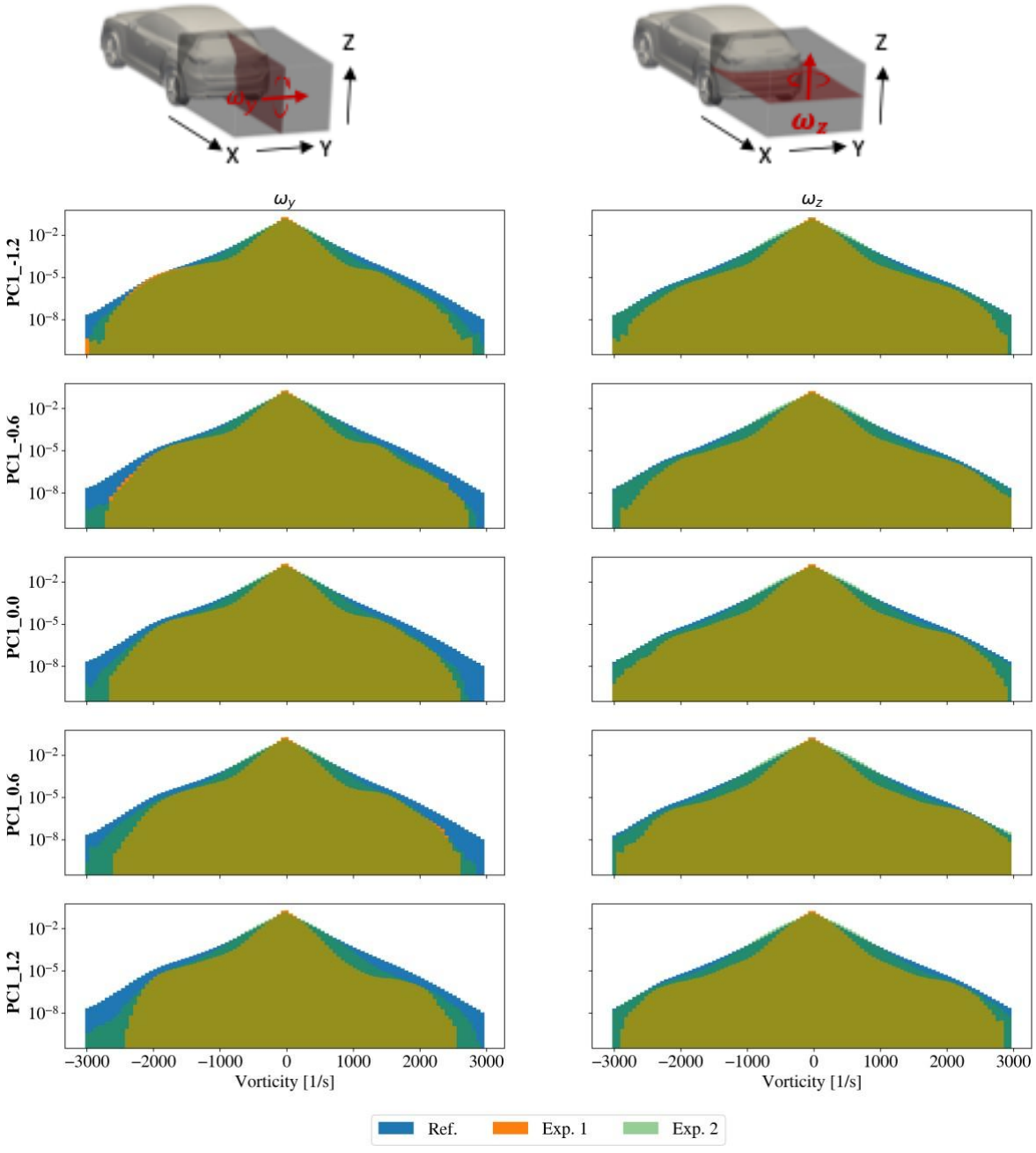

Figure F.1. Probability density function of vorticity for the flow fields around all vehicle shape patterns.

Appendix G

Figure G.1 presents the time variation of turbulent kinetic energy for the flow field of all vehicle shape patterns.

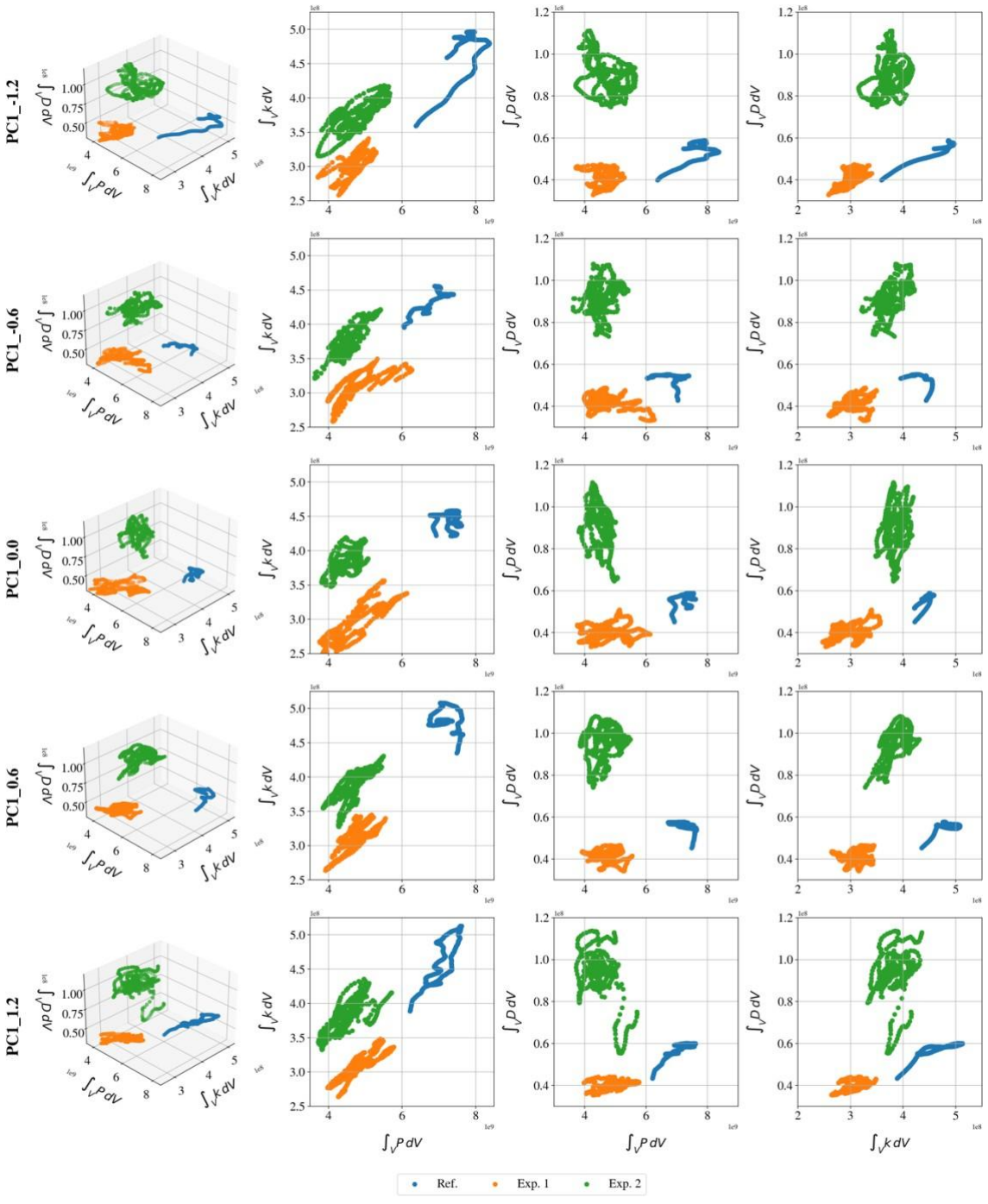

Figure G.1. Phase space trajectory of the physical system, defined by turbulent kinetic energy (TKE), TKE production, and TKE dissipation, for all vehicle shape variations.

Appendix H

Figure H.1 presents the correlation coefficient of the flow field for all vehicle shape patterns.

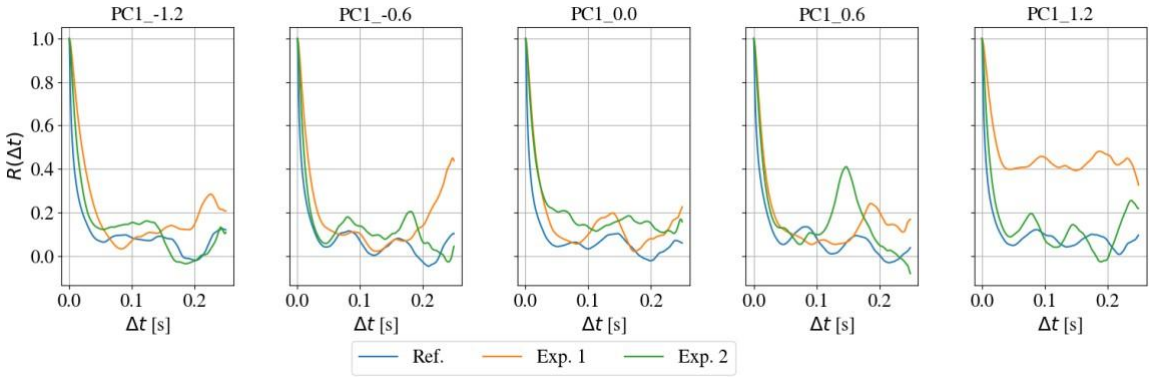

Figure H.1. Temporal correlation coefficient for all vehicle shape variations.

References

[1] C. Kato, Y. Yamade, K. Nagano, K. Kumahata, K. Minami, T. Nishikawa, 'Toward Realization of Numerical Towing-Tank Tests by Wall-Resolved Large Eddy Simulation based on 32 Billion Grid Finite-Element Computation', in: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020, pp. 23–35.

[2] T. Yoshida, 'Fujitsu High Performance CPU for the Post-K Computer', in: Hot Chips, 2018. [Online]. Available: https://www.fujitsu.com/global/documents/solutions/business-technology/tc/catalog/20180821hotchips30.pdf.

[3] R-CCS, 'About Fugaku', 2021. [Online]. Available: https://www.r-ccs.riken.jp/en/fugaku/about/ (Accessed 12 September 2023).

[4] N. Jansson, M. Karp, A. Perez, T. Mukha, Y. Ju, J. Liu, S. Páll, E. Laure, T. Weinkauf, J. Schumacher, P. Schlatter, S. Markidis, 'Exploring the Ultimate Regime of Turbulent Rayleigh–Bénard Convection Through Unprecedented Spectral-Element Simulations', in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Association for Computing Machinery, New York, NY, USA, 2023, pp. 1–9.

[5] LUMI consortium, 'LUMI's Full System Architecture Revealed', 2025. [Online]. Available: https://www.lumi-supercomputer.eu/lumis-full-system-architecture-revealed/ (Accessed 14 February 2025).

[6] K.L. Vasudev, R. Sharma, S.K. Bhattacharyya, 'A multi-objective optimization design framework integrated with CFD for the design of AUVs', Meth. Oceanogr. 10 (2014) 138–165.

[7] M. Rüttgers, J. Park, D. You, 'Large-eddy simulation of turbulent flow over the DrivAer fastback vehicle model', J. Wind Eng. Ind. Aerodyn. 186 (2019) 123–138.

[8] J.L. Lumley, 'The Structure of Inhomogeneous Turbulent Flows', in: Atmospheric Turbulence and Radio Wave Propagation, A.M. Yaglom, V.I. Tatarski (Eds.), Nauka, 1967.

[9] T. Murata, K. Fukami, K. Fukagata, 'Nonlinear mode decomposition with convolutional neural networks for fluid dynamics', J. Fluid Mech. 882 (2020) A13. https://doi.org/10.1017/jfm.2019.822

[10] K. Ando, K. Onishi, R. Bale, A. Kuroda, M. Tsubokura, 'Nonlinear reduced-order modeling for three-dimensional turbulent flow by large-scale machine learning', Comput. Fluids 266 (2023) 106047.

[11] S. Hochreiter, J. Schmidhuber, 'Long Short-Term Memory', Neural Comput. 9 (1997) 1735–1780.

[12] B. Koo, H. Kim, T. Jo, S. Kim, J.Y. Yoon, 'Proper orthogonal decomposition–Galerkin projection method for quasi-two-dimensional laminar hydraulic transient flow', J. Hydraul. Res. 59 (2021) 224–234.

[13] P. Benner, S. Gugercin, K. Willcox, 'A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems', SIAM Rev. 57 (2015) 483–531.

[14] 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 (2020) 367–383.

[15] K. Hasegawa, K. Fukami, T. Murata, K. Fukagata, 'CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers', Fluid Dyn. Res. 52 (2020) 065501.

[16] A. Higashida, K. Ando, M. Rüttgers, A. Lintermann, M. Tsubokura, 'Robustness evaluation of large-scale machine learning-based reduced order models for reproducing flow fields', Future Gener. Comput. Syst. 159 (2024) 243–254.

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

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

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

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

[21] 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.

[22] NVIDIA, 'NVIDIA H100 Tensor Core GPU Datasheet', 2024. [Online]. Available: https://resources.nvidia.com/en-us-tensor-core/nvidia-tensor-core-gpu-datasheet (Accessed 27 February 2025).