LATVIAN JOURNAL OF PHYSICS
AND TECHNICAL SCIENCES
2025, N 6

DOI: 10.2478/lpts-2025-0044

REDUCED ORDER MODELLING FOR THE DRONE FLOW FIELD: AUTOENCODER-DRIVEN NONLINEAR COMPRESSION VERSUS POD

R. Ralla

Institute of Mechanical and Biomedical Engineering
Riga Technical University,
1 Paula Valdena Str., Riga, LV-1048, LATVIA
E-mail: rajeev.ralla@edu.rtu.lv

Reduced-order modelling (ROM) is a computational technique that approximates the high-dimensional fidelity data by projecting onto a lower-dimensional subspace, while preserving dominant physical features of the multi-physics simulation data. This study presents a systematic comparison of two established dimensionality-reduction techniques – proper orthogonal decomposition (POD) and autoencoders (AEs) – for reducing dimensionality, while preserving the dominant physical features of the drone flow field. While POD employs linear subspace projection through singular value decomposition (SVD) to capture the energy dominant modes, the effectiveness of capturing the nonlinear relationship is limited. Since drone flow field data represents a complex multi-physics relationship, POD exhibits lesser capability of capturing maximum nonlinearity in the dataset while projecting onto subspace. In contrast, AE demonstrates superior reconstruction from the latent space, which the author of the study quantifies using performance metrics such as Mean Square Error (MSE) and the coefficient of determination (R^2). POD achieves lower MSE of 0.0352 vs. AE achieved a reconstruction error of 0.0491 and demonstrated superior variance retention with $R^2 = 0.951$ (95.1 %) compared to POD's $R^2 = 0.781$ (78.1 %), indicating stronger preservation of nonlinear features. With regard to data compression, the second aspect for the ROM selection, AE outperformed in the compression ratio of 23.37x compared to 3.61x of POD. This study explores computational requirements, revealing that AE demands greater resources than POD. While POD relies on a single SVD, AE requires training and testing with iterative gradient descent across multiple layers and back-propagation to capture the effective representations of a nonlinear dataset.

Keywords: *AE, dimensional reduction technique, MSE, Multi-physics simulation, POD, R^2, ROM, SVD.*

1. INTRODUCTION

Large-scale simulations of multiphysics system such as drone generates a large set of solution matrix [1]. The drone flow field, including velocity, pressure, and sound generation, often displays strong nonlinearities due to complex interactions that arise from the coupling governing equations across different fields. The study extracts and analyses 13 parameters from the drone simulation. These include spatial coordinates (x, y, z) in metres (m), velocity components in the inertial frame (u, v, w) in metres per second (m/s), velocity components in a rotating reference frame (u', v', w') in metres per second (m/s), total pressure (Pa), dynamic pressure (Pa), and the sound power level (in decibels). In total, there are 590,815 spatial nodal points, each described by 13 features, occupying 84.24 MB. It is challenging and expensive to capture the nonlinearity of the dataset to manipulation for the machine learning or other processes [2]. Reduced-order modelling (ROM) is used as a data compression technique, which captures the most dominant features and truncates less important features reducing data complexity, storage and computational requirements. Proper Orthogonal Decomposition (POD) is a widely used method for ROM building; it is effective in capturing linear trends in the given data. However, its ability to capturing the nonlinearity is limited, especially in case of multi-physics problems capturing the effective linear trends may lead to a significant loss of important trends in the given data [3]. Autoencoders (AEs) are a special class of neural networks capable of dealing with unsupervised learning, by mapping higher dimensional input data to a compact latent space while capturing and training of nonlinear relationships within data [4]. This study presents a comparative analysis of POD with a given dataset and evaluates effectiveness in capturing the variance within a given dataset and the compression rates. By examining both methods, the study aims to determine which technique more efficiently retains essential features of the drone flow field while reducing its dimensionality. Few metrics are used to evaluate the quality of ROM reconstructions, and the accuracy of approximations made by these methods such as Mean Squared Error (MSE), Mean Absolute Error (MAE) and R^2. The nonlinearity of the drone simulation data can be assessed and the methods compared for their ability to capture it, enabling a conclusion about which method is significantly better at compressing and reconstruction the true data [5].

2. DATA CLEANING AND MASKING

The raw drone solution matrices consist of field variables in columns and spatial points in rows. During pre-processing, extraction of the solid region was performed using blank space or zeros, which often correspond to regions inside the drone geometry or areas not physically meaningful for flow or acoustics. Such non-physical information is not practically feasible for additional exercise on these data points. Thus, a special concept called masking can be implemented to forward the data points or field variables without interrupting the meaningful solution points [6]. To prevent

these regions from corrupting the learning process, two options are considered. The first one is treating zeros as a class domain, including geometry. The second option is to mask the geometry or zero values to exclude from the dataset and focus only on the active flow regions. The solution matrices of drone simulation data contain zero – especially in the pressure, velocity and acoustic power level values – originating from the solid region and lacking physical meaning. Masking these regions helps prevent machine learning models from being skewed by unrealistic values. Field features, such as velocity, RRF, and acoustic power level, contain zeros that indicate solid regions. However, one field may be zero while another has a value; this can be meaningful but yet not part of the active flow field. The second approach was selected to preserve the integrity of physically meaningful data. Features, such as rotating frame velocities and acoustic power levels, typically reveal where solid regions are present (i.e., zeros across fields). By excluding these, the learning model is prevented from interpreting artefacts as real flow features [7]. Preparation of the final dataset is a crucial step before conducting the POD or AE; both methods require standard pre-processing before the data is fed into the models. The drone simulation is steady state simulation with a rotating blade of 3500 rpm, standard atmospheric pressure and temperature, and air as the working fluid. Before cleaning, the initial file contained 715,697 rows and 13 columns. After masking out the solid region, 590,815 rows and 13 columns remained for use in POD and AE to perform data compression and capture the dominant modes.

3. VISUALISATION OF THE DATASET

The spatial arrangement of data points, each carrying 13 features, constitutes a high-dimensional dataset that captures complex multi-physics phenomena.

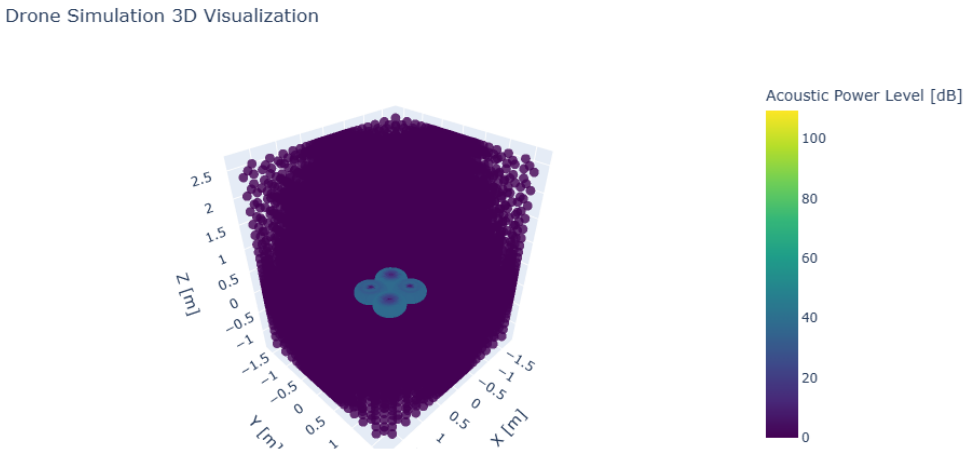

Fig. 1. Acoustic power level [dB].

Figure 1 shows that colour variations in the data points represent the active points in the spatial locations. These data points collectively form a multidimensional field, where fixed spatial points (in metres) and feature-wise correlations. These features exhibit nonlinear interdependences, slight change among the features cause the unpredictable effects to the other features. This complex behaviour makes it challenging to capture nonlinearity within the data points.

4. NORMALIZATION OF FLOW FIELD DATA

It is an essential step before any further process, since the drone field flow variables, such as pressure, velocities, and acoustic power level, span different units and magnitudes. Normalization is an essential pre-process before further analysis. Without normalization, some variables may dominate the learning process leading to numerical instability. POD decomposes data into modes based on variance (energy). Without normalizing, higher-magnitude variables (such as pressure) can bias the decomposition towards pressure-dominated nodes rather than coherent flow structures, potentially obscuring lower energy features. Autoencoder relies on gradient-based optimisation; without normalization of the dataset, weight updates become skewed towards the higher magnitude variables, the convergence rate slows and a coherent latent representation of flow may not be learnt. Physical interpretation without normalization of the dataset becomes very disruptive in ROM building, compressibility will be high, but capturing the multiphysics features is essential. Normalizing ensures that no single variable dominates due to its inherent scale, and standard scaling preserves relative fluctuations while removing bias from the absolute magnitude. StandardScaler is a normalization technique that transform the dataset to have mean (μ) of 0 and a standard deviation (σ) of 1 to ensure numerical stability and balanced feature contribution [8].

$$X_{scaled} = \frac{X-\mu}{\sigma}, \quad (1)$$

where
X – a flow field variable;
μ = a mean of the dataset;
σ = standard deviation of the dataset.

5. AUTOENCODER

Autoencoders (AEs) are a separate class of neural networks specialised in unsupervised learning. Learning from high-dimensions data efficiently and compressing input data into a lower-dimensional latent space is called dimensionality reduction. AEs are good enough to learn and compress nonlinear data such as flow field data of drone simulation. The architecture of AE comprises an encoder, a latent space and a decoder. The encoder maps higher-dimensional input data onto the latent space, and the decoder – mirroring the encoder – reconstructs the data to its original dimensionality. Nonlinear flow data is very difficult to train, and extracting the features heavily depends on the architecture of training neural units to minimise the cost function or loss function; performance of the model will tell how well the reconstructed model is to the original

model. Training a model on a simple dataset – by computing the weight sum of input features and a bias – is simple task in case of linear regression prediction model. Training means setting the parameters so the model best fits the training dataset. The performance measure of the model can be evaluated by mean square error (MSE), which explains the how well the model parameters minimise the cost function [9]. Training an AE means learning through multiple layers of weights and biases through nonlinear transformation, which makes it much more complex to find minimisation of the cost function. The general functional equation of the autoencoders is the following:

$$\check{X} = f_D\big(W_D\big(f_E(W_E x + b_E)\big) + b_D\big), \qquad (2)$$

where
f_D – an activation function for decoder;
f_E – an activation function for encoder;
W_E, b_E – encoder weights and biases;
W_D, b_D – decoder weights and biases;
X – original input data;
$\check{X}$ – reconstruction input values.

The goal of autoencoders is to approximate the $\check{X}$ reconstructed input values to the X original values; the encoder is a halfway process to latent space. Drone simulation data contains 13 features, few features are strongly correlated; velocity components (x, y, z) and pressure may show strong interdependence. These feature correlations carry overlapping or redundant information, and dimensional reduction becomes easy as the strongly correlated two or more features are compressed onto 1-dimensional space, removing the weak correlations. AE detects the correlations of the features in the input data during the training and captures the patterns, represents them in fewer features in the latent space. Compression takes place into fewer dimensional space without losing much information. Modelling the learning algorithm to capture the nonlinearity in the dataset required a careful design of the neural network architecture, including measures to prevent under- and overfitting data (regularization), activation function, optimizer, and layer selections. Regularization is an import step before feeding the input data into the training process; some feature, such as pressure and velocity components, show strong correlation; pressure and sound generation are also correlated in the drone simulation. During the learning process, the algorithm overgeneralized to a few features and ignored the correlation of other features – a trap known as overfitting – resulting in the same features captured with high cost or loss function during reconstruction. Underfit has an opposite effect – it occurs when the model is too simple to learn the underlying structure or the correlation results in inaccurate performance. It is necessary to maintain a right balance between fitting the training data and keeping the model simple enough to generalize well. The model must be constrained during training with the regularization parameter such as a dropout technique. During training, neurons can be randomly selected to dropout in a layer or their output can be ignored. The dropout rate = r is attached to the individual layers to prevent the over- or underfitting to improve the training processes [10].

The mathematical construction of an individual layer with weight and bias is linear in relationship, and the input data points are injected into a slack of linearly transformed:

$$Z = f(WX + b), \qquad (3)$$

where Z *i*s an output of the neuron layer, W is a weight matrix and b is biased vector. The other layers are also linearly transformed and connected to hidden layers; without additional function a neural network behaves like a linear model and fails

to fit the nonlinear dataset. The activation function will introduce the nonlinearity to the model, thus the network with a nonlinearity activation model can learn the complex and nonlinear patterns by adjusting the activation function to approximate the dataset nonlinearity. Rectified Linear Unit (ReLU) is a widely used activation function while dealing with complex datasets because of its simplicity and effectiveness.

$$ReLU(x) = max(0, x).$$

$$\text{If } x > 0, then\ ReLU(x) = x;$$
$$\text{If } x \leq 0, then\ ReLU(x) = 0.$$

ReLU activation function allows passing only the positive signal or values and blocking negative values; this causes the dead neurons or inactive neurons.

The individual neuron with activation functions is passed through the encoder's part of an autoencoder; it gets linearly transformed by weight and biases, and then nonlinearity is activated at each layer.

$$Layer\ 1: a_1 = ReLU(W_1 * x + b_1).$$
$$Layer\ 2: a_2 = ReLU(W_2 * a_1 + b_2).$$
$$Layer\ 3: a_3 = ReLU(W_3 * a_2 + b_3).$$
$$Layer\ 4: a_4 = ReLU(W_4 * a_3 + b_4).$$

Adding a dropout rate to the layers will cover all elements required for constructing the neural network. In AE, the encoder is typically composed of successive layers of decreasing width, culminating in a compact bottleneck (latent space) containing only a small number of neurons.

$$Layer\ 1: a_1 = ReLU(W_1 * x + b_1).$$
$$Layer\ 1: a_{drop1} = Dropout(a_1, r = 20\,\%).$$
$$Layer\ 2: a_2 = ReLU(W_2 * a_{drop1} + b_2).$$
$$Layer\ 2: a_{drop2} = Dropout(a_2, r = 15\,\%).$$
$$Layer\ 3: a_3 = ReLU(W_3 * a_{drop2} + b_3).$$
$$Layer\ 4: a_4 = ReLU(W_4 * a_3 + b_4).$$

Dropout was employed exclusively in the first two layers—20% in layer 1 and 15% in layer 2. By randomly suppressing a fraction of activations, the network avoids over-reliance on particular neurons and is encouraged to learn redundant, and thus more robust, representations. If neuron A is very active and learns a pattern, the model will never train neuron B to learn the same pattern. This prevents any single neuron from being solely responsible for recognising a pattern; if multiple neurons learn the same pattern, the load is shared rather than concentrated on neuron A. Dropout was not applied to the remaining layers: early layers were encouraged to share the load of pattern extraction, and extending dropout deeper risked underfitting and unstable performance [11]. Once the model is trained to carry forward the input dataset values in the layers, an objective of minimising the loss function is specified. Consequently, the model's parameters are iteratively adjusted toward this objective by means of an optimisation algorithm. Optimisation algorithm follows an iterative approach that gradually tweaks the model parameters in the direction that reduces a loss the most, typically using gradient decent of the loss or cost function with respect to the weight and bias, updating step by step and setting the parameters for the lowest possible loss. Optimizers such as Adam enhance the process by adapting the learning rate for each parameter, accelerating convergence. Learning step size or rate is proportional to the slope of the cost function, if a rate is small then the algorithm will have to go through many iterations to converge. Batch gradient descent calculates the gradient of the loss function treating the whole training dataset to update one single weight update of model parameters; this makes a strong foundation of the learning process of single neurons capturing most of the nonlinear-

ity not as bits and pieces. This method is accurate and computationally expensive yet suitable for the nonlinear activation and nonlinear dataset [12]. The partial derivative of the cost function with respect to the parameters or gradient of MSE loss is computed, and the parameters are updated using gradient descent with learning rate η.

$$MSE(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\check{X}^{(i)} - X^{(i)})^2. \quad (4)$$

$$\frac{\partial}{\partial\theta_i}MSE(\theta) = \frac{\partial}{\partial\theta_i}\left[\frac{1}{m}\sum_{i=1}^{m}(\check{X}^{(i)} - X^{(i)})^2\right]. \quad (5)$$

$$= \frac{1}{m}\sum_{i=1}^{m} 2\left(\check{X}^{(i)} - X^{(i)}\right)\cdot\frac{\partial\check{X}^{(i)}}{\partial\theta_j}.$$

$$\frac{\partial\check{X}^{(i)}}{\partial\theta_i} = X_j^{(i)}.$$

$$\frac{\partial}{\partial\theta_j}MSE(\theta) = \frac{2}{m}\sum_{i=1}^{m}\left(\check{X}^{(i)} - X^{(i)}\right)X_j^{(i)}.$$

$$\theta = \theta - \eta\cdot\frac{\partial}{\partial\theta_j}MSE(\theta). \quad (6)$$

Equation (4) expresses the MSE, quantifying the average square difference between the AE reconstruction (or latent space) and the original dataset. Equation (5) expresses the gradient of MSE with respect to parameters or taking derivative of MSE with respect to θ. In Eq. (6), the gradient descent update iteratively adjusts the parameters by subtracting the product of learning rate η and the gradient from the prior parameter values, ensuring progressive minimization of the reconstruction error [13].

The model architecture (with 13 features and 590,815 nodes) required multi-layered networks along with hyperparameters. The main aim is to achieve compressibility and minimization of cost function, thereby reflecting the accurate reconstruction of the input dataset. The encoder portion of the autoencoder is constructed up to the latent space, and its ability to capture most of the dataset's nonlinearity is assessed.

Epoch is an iterative process. One epoch is one full pass of the entire training dataset through the neural networks. In our case, an epoch = 1,000; all neural networks were trained for 1,000 epochs over the training set; the parameters of each neuron were updated at every iteration. With 590,815 samples trained for 1,000 epochs, a total sample examined 50 million sample exposures. A batch size of 50 is employed: the gradient is averaged over each set of 50 samples, and the weights are updated in mini-batches. Step size is an important factor in terms of updating each weight or size step for weight updates during gradient descent η = 0.0002, i.e., weights are adjusted by 0.02 % of the gradient magnitude in each update.

Table 1. Encoder Architecture Containing the Neurons per Layer and Activation Functions

Encoder layers	Neurons per layer	Activation function	Dropout rate
Layer 1	128	ReLU	0.20
Layer 2	64	ReLU	0.15
Layer 3	32	ReLU	0
Layer 4	16	ReLU	0
Layer 5	8	ReLU	0

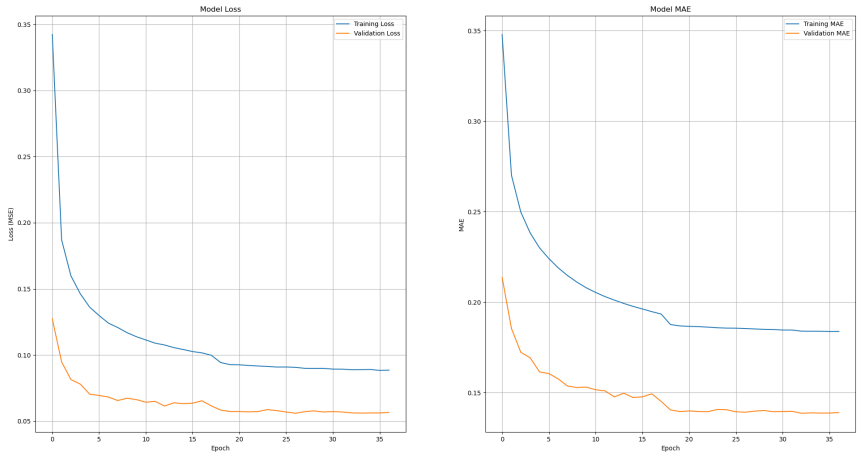

Fig. 2. Training loss vs. validation loss.

The above graphs (Fig. 2) report the training process that was monitored using both MSE and MAE. These metrics offer a model learning behaviour and generalization performance. In the MSE-based loss graph, both the training and validation exhibits a similar trend. The training loss begins around 0.34 and steadily declines and stabilizes around 0.0885 units, validation loss starts at 0.13 and decreases steadily and converges quickly, reaching 0.0566 units. This shows that the model is not overfitting and generalizes well to unseen data. The validation loss is another key metric to check the model on the unseen dataset or test the set. Then, the training loss is compared to the validation loss to assess the how well the model learns nonlinearity in the training dataset and how it performs on the test set. MAE measures tell how much predictions are off from the true values without considering their direction. The training loss initially starts from 0.35 and reduces drastically and stabilizes at 0.18 for the remaining epochs. The validation losses start from approximately 0.22 and follow decreasing trends and flatten at 0.1392 units; model prediction differs from a true value by 0.1392 units.

Overall, the learning curves exhibit the model convergence well by minimal epoch despite a large size epoch of 1,000. The validation curves for both metrics also indicate strong performance on the test dataset, with consistently low values. The hyperparameter constrains the model learning process via additional controllers that can be adjusted to regularization for a better training outcome.

Table 2. Performance Metrics and Values of Autoencoders for Model Evaluation

Metrics	**Values**
Mean Squared Error (MSE)	0.0491
Mean Absolute Error (MAE)	0.1392
R-squared (R^2)	0.9510
Final Training Loss	0.0885
Final Validation Loss	0.0566
Original Data Size (MB)	84.27
Latent Space Size (MB)	3.61
Compression Ratio	23.37x

R-squared (R^2) is also called a coefficient of determination, which is a statistical measure used to evaluate the performance of regression or a reconstruction model [14]. This provides the insight of the performance capturing the nonlinearity of the input dataset to the prediction values. R^2 of 0.9510 means that 95.1 % of the variance in the original data is captured by reconstructed data, which is a high and strong result.

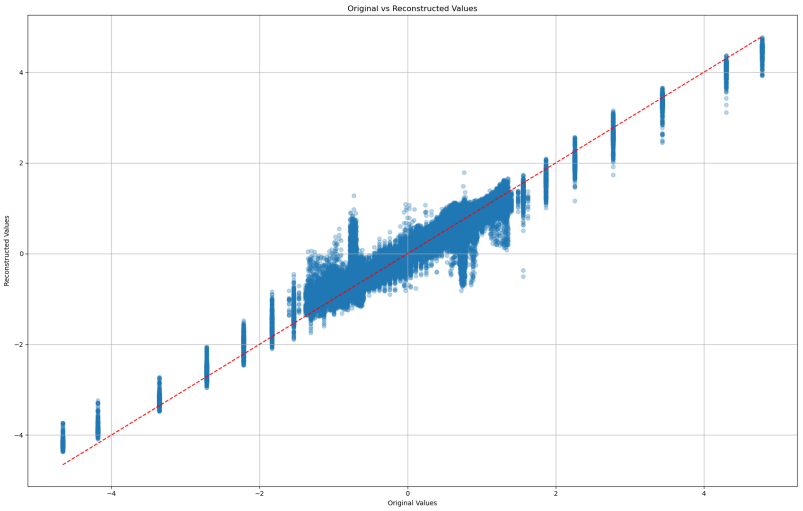

Fig. 3. Original vs. reconstructed values.

Figure 3 shows how well the model captures the nonlinearity through the distribution of the points. The X-axis is the original data points and Y-axis is the reconstructed or predicted data from the model. The red dashed line in the graph has one–one relationship at an angle of 45, the reconstructed is original since we have a higher accuracy of 95.1%. If slope $m = 1$ and the intercept $c = 0$, then line equation $y = mx+c$ becomes $y = (1)x+0$. The blue circles in the plot represent the individual points. Many of these points are very close to the red dashed line; this shows that the model has learnt to reconstruct a majority of inputs with higher accuracy. However, some points are scattered above the line where the reconstructed value is higher than the original value, and others lie below it, indicating underestimation. A majority of the points are scattered at the centre range of zero and closely clustered around the line, indicating that nonlinear relations in the data are well captured. The graph ranges from -4 to 4.5. Some data points lie at the extremes and others near the centre, and the encoder manages to capture both ranges well. If values fall beyond the extreme range (i.e., are outliers), the reconstruction is likely to be less precise.

In terms of compressibility, the original data occupied 84.27 (MB) of storage space. Through the encoder architecture, the data was sequentially compressed by narrowing down the dimensionality from 13 to 8 in the latent space. The dimensionality reduction not only captures the patterns but also reduces the data storage capacity. The resulting latent space occupies 3.61 (MB), with a compression ratio of 23.37x. This demonstrates the encoder's effectiveness in capturing critical information and high efficiency in terms of storage.

6. PROPER ORTHOGONAL DECOMPOSITION (POD)

The POD is a widely used technique in dimensional reduction, on Single Value Decomposition (SVD) algorithm the POD identifies the set of coherent structures (modes) that are optimal for representing the drone simulation data. Dimensional reduction serves here by truncating the modes based on the capturing the maximum variance and energy contribution. Each mode is important in preserving the dominant features; however, energy contribution and captured variance decrease as the number of modes increases. The most energetically significant patterns are captured in the initial modes; subsequent modes capture less dominant, orthogonal structures. POD is well known for leveraging singular values to find orthogonal linear modes, onto which data is projected using a fixed orthogonal basis. Practicing the POD to extract modes from nonlinear datasets is a fundamentally different approach to AE.

To proceed with the construction of POD for the drone simulation dataset using the decomposing techniques, the simulation dataset will be organised into a matrix of snapshots, containing the spatial distribution of the flow field [2].

$$Snapshotmatrix\_X = U\Sigma V^T, \quad (7)$$

where $U \in \mathbb{R}^{mxm}$ contains the left singular vector matrix U of shape m × m, which only includes the time-dependent structures. $\Sigma \in \mathbb{R}^{nxn}$ is a diagonal matrix n × n with singular values σ_i, which are the energy modes. POD utilises this energy of the modes for data compression or dimensionality reduction. $V^T \in \mathbb{R}^{nxn}$ contains the right singular vector matrix V^T of n × n or spatial modes.

A diagonal matrix contains real and non-negative values on the diagonal, and zeros are present outside the diagonal. Singular values are arranged in descending order, from highest to lowest. Truncated SVD is based on ranks or modes, where the highest singular values are preserved, and the lower ones are discarded. If the singular values capture the higher singular values in the first 3 or 4 singular values, capturing 99 % of the physical system variance, truncating the matrix to Rank 3 or Mode 3 can effectively reduce dimensionality of the matrix, preserving the most significant modes and eliminating the less important ones.

$$Trancated_{snapshotmatrix_{X_r}} = U_r\Sigma_r{V^T}_r. \quad (8)$$

$U_r \in \mathbb{R}^{mxr}$ contains the left singular vector (m, r) reduction based on rank columns.

$\Sigma_r \in \mathbb{R}^{rxr}$ is a diagonal matrix r × r with singular values σ_r; it becomes a square matrix size based on rank. Since most energy modes are captured by the initial singular values and the remaining smaller ones have a negligible impact, truncating the smaller singular values leads to a reduction in the size of the diagonal matrix, effectively lowering the rank and reducing the dimensionality of the system while retaining the most significant features. ${V^T}_r \in \mathbb{R}^{rxn}$ contains the right singular vector matrix of ${V^T}_r$ of shape n × n; it plays major role to project the reduced truncated matrix back to the original space and is aligned with the columns of the original matrix.

Truncated_snapshotmatrix_X_r provides an approximation of the original matrix on the low-rank space. Each component in the matrix decomposition has significant dimensionality reduction based on the rank *r*. The truncated left singular vectors and diagonal matrix $U_r\Sigma_r$ represent the transformation that projects data onto the low-rank subspace, with dimensions corresponding to

the row of the original matrix. Truncated right singular vector $V^T{}_r$ corresponds to the dimension of the column of the original matrix used to reproject the low-rank space back onto the original space, reconstructing an approximation of the original matrix [15].

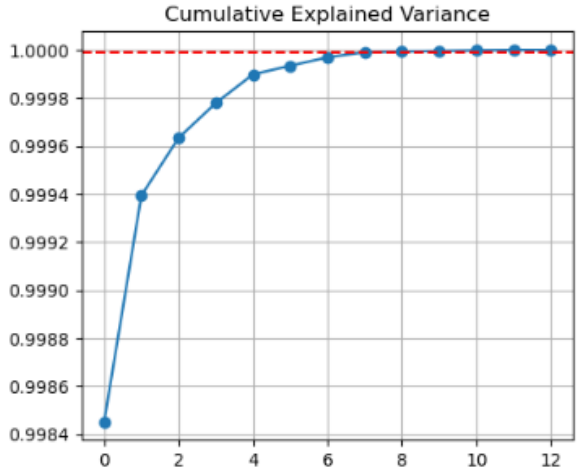

Fig. 4. Cumulative explained variance.

Cumulative explained variance is an important parameter to determine how many modes are required to represent the majority of the dataset energy or variance. In Fig. 4, one can observe the graph modes on X-axis to Y-axis energy; Mode 4 captures 99.96 % of variance and subsequent modes capture everything.

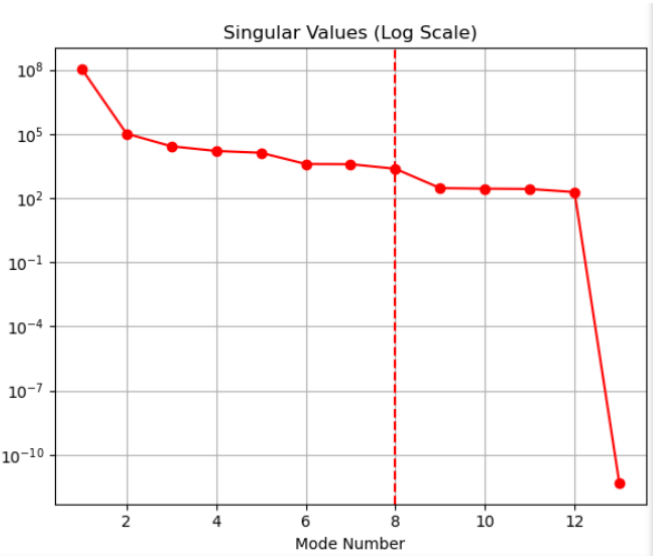

Fig. 5. Singular values.

Singular values (log scale) (Fig. 5) of the initial first mode (10^8) and second mode (10^5) have the highest values and capture a dominant portion of the system energy or variance. The values decay more slowly from Mode 3 to Mode 8, contributing mod-

erate information to fine details or secondary structures. Truncation rank = 8 captures all the variance in the dataset, and singular values of rank 8 ($\sim 10^{3.5}$) still contribute significantly, indicating sensitivity to the secondary structures of fluid flow field and noise. Modes beyond rank = 8 contribute little, so truncation leads to dimensional reduction. POD analysis metric indicates how well the original simulation dataset is reconstructed in the dimensional reduced space. MSE and R^2 metrics are used to evaluate the performance of the AE and POD in terms of reconstruction quality and data compression.

Table 3. Performance Metric of POD for Performance Evaluation

Metric	**Values**
Mean Square Error (MSE)	0.0352
R^2	0.7813
Original data (MB)	84.27
Reconstructed data (MB)	52.03
Compression ratio	1.62x

The POD achieves an MSE = 0.0352 (in data units^2), indicating a very small average squared difference between the original and reconstructed data. Coefficient of determination or R^2 is 0.7813, indicating that the variance in the data is explained, but some nonlinear structures in the dataset may be missed. The original data occupied 84.27 MB and reconstructed data occupied 52.03 MB, yielding a compression ratio of 1.62x.

7. RESULTS

In the comparative study of the AE and POD, both methods performed on the same drone simulation dataset and focused on the performance metric to evaluate the reconstruction data or latent space onto a lower space. Reduced order modelling of the drone simulation was achieved by both methods via approximating the original data and projecting onto a lower dimensional space by truncating or careful selection of training hyperparameters. POD and AE achieved substantial data compression, yielding a significant compression ratio from the original to the reconstructed data.

Table 4. Comparative Performance Metric of POD vs. AE

Metrics	**POD**	**AE**
Mean Square Error (MSE)	0.0352	0.0491
R-squared (R^2)	0.7813	0.9510
Final Training Loss	N/A	0.0885
Final Validation Loss	N/A	0.0566
Original Data Size (MB)	84.27	84.27
Latent Space Size (MB)	52.03	3.61
Compression ratio	1.62x	23.37x

POD has a lower MSE of 0.052 than AE of 0.0491, indicating better minimisation of a reconstruction error onto a lower-dimensional space. This suggests the POD is more effective in retaining the dominant spatial structures. POD captures only 78 % of the variance ($R^2 = 0.78$). Given truncation of rank = r, the cumulative explained variance and truncation based on the energy contribution suggests the best possible choice between 6 and 8 modes. Since the AE uses an 8-dimensional latent space, it is reasonable to truncate POD at r = 8. The fundamental concept of dimensional reduction by AE is based on the correlation among features and redundant overlapping features and compressing them into a single dimension. POD has a lower reconstruction error, but it struggles to capture the nonlinear relationships present in the original dataset. In contrast, AE captured 95.1 % of the variance in the dataset because of their nature neural networks can learn complex, nonlinear patterns through training and testing on the original dataset. MSE and R^2 show the performance of two techniques, indicating satisfactory reconstruction of the original dataset in a lower-dimensional space. The second task checked the effective data compression by volume occupation or storage capacity. AE showed a superior ability in compression ratio of the original dataset to the reconstruction space or latent space. The original data (84.27 MB) was reduced to 3.61 MB with AE (23.7x), but only to 52.03 MB with POD (1.6x), indicating AE's superior compression.

8. CONCLUSION

Developing a ROM based on AE is preferable if the original dataset has a nonlinear relationship; POD is advisable if the dataset has a linear behaviour. As best practice, target low reconstruction error (MSE) and high explained variance – metrics on which POD typically excels for linear structure. It especially concerns drone simulation data, which contains complex muti-physics interactions underlying a nonlinear relationship. While POD is effective in capturing linear dominant modes and spatial structures, it may struggle to recognise and present nonlinear dependencies within the original dataset. This limitation can lead to a reduced accuracy and inability to capture the variation, thereby on a large scale resulting in partial development of model reduction. Encoder can serve as an alternative and deliver better performance in recognising the hidden nonlinear patterns and performing superior data compression by volume. Implementing the ROM through AE is very effective, especially for complex drone simulation. AE utilises neural networks a complex construction of neurons and carefully selects hyperparameters to maintain generalization and learning capacity, allowing it to effectively capture nonlinear dynamics and interactions present in high-fidelity data. Moreover, the high value of R^2 and significant data compression achieved by AE indicate its strength in both dimensional reduction and accurate reconstruction, making it a powerful tool for effective use in ROM development and predictive modelling.

REFERENCES

1. Holmes, P., Lumley, J. L., Berkooz, G., & Rowley, C. W. (2012). *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press.
2. Brunton, S. L., & Kutz, J. N. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press.
3. Kadeethum, T., Ballarin, F., O'Malley, D., Choi, Y., Bouklas, N., & Yoon, H. (2022). Reduced Order Modelling for Flow and Transport Problems with Barlow Twins Self-Supervised Learning. *Scientific Reports, 12*, 14240. Available at: https://www.nature.com/articles/s41598-022-24545-3
4. Berahmand, K., Daneshfar, F., Salehi, E., & Li, Y. (2024). Autoencoders and Their Applications in Machine Learning: A Survey. *Artificial Intelligence Review, 57* (2), 52. https://doi.org/10.1007/s10462-023-10662-6
5. Jierula, A., Wang, S., OH, T.-M., & Wang, P. (2021). Study on Accuracy Metrics for Evaluating the Predictions of Damage Locations in Deep Piles Using Artificial Neural Networks with Acoustic Emission Data. *Applied Sciences*, *11* (5), 2314. https://doi.org/10.3390/app11052314
6. Taira, K., Brunton, S. L., Dawson, S. T. M., Rowley, C. W., Colonius, T., McKeon, B. J., … & Ukeiley, L. S. (2017). Modal Analysis of Fluid Flows: An Overview. *AIAA Journal, 55* (12), 4013–4041. https://doi.org/10.2514/1.J056060
7. Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds Averaged Turbulence Modelling Using Deep Neural Networks with Embedded Invariance. *Journal of Fluid Mechanics, 807*, 155–166. https://doi.org/10.1017/jfm.2016.615
8. Duraisamy, K., Iaccarino, G., & Xiao, H. (2019). Turbulence Modelling in the Age of Data. *Annual Review of Fluid Mechanics, 51*, 357–377. https://doi.org/10.1146/annurev-fluid-010518-040547
9. Fukami, K., Nakamura, T., & Fugata, K. (2020). Convolutional Neural Network Based Hierarchical Autoencoder for Nonlinear Mode Decomposition of Fluid Field Data. *Physics of Fluids, 32* (9), 095110. https://doi.org/10.1063/5.0020721
10. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Autoencoders. In *Deep Learning* (pp. 499–523). MIT Press.
11. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15* (56), 1929–1958.
12. Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*: *Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.) O'Reilly Media.
13. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Machine Learning Basics. In *Deep Learning* (pp. 96–161). MIT Press.
14. Chicco, D., & Warrens, M. J., & Jurman, G. (2021). The Coefficient of Determination R-Squared is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation. *PeerJ Computer Science*, *7*, e623. https://doi.org/10.7717/peerj-cs.623
15. Volkwein, S. (August 27, 2013). *Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling*. University of Konstanz. Lecture Notes. Available at: https://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Book.pdf