The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

SURROGATE MODELING OF UNSTEADY AERODYNAMIC LOADS ACTING ON A PLUNGING FLAT PLATE

RAHUL SUNDAR[1*], VIRENDRA KUMAR[1], DIPANJAN MAJUMDAR[1], CHHOTE LAL SHAH[1] AND SUNETRA SARKAR[1]

[1] Department of Aerospace Engineering
Indian Institute of Technology Madras, Chennai - 600036, India
* e-mail:rahulsundar@smail.iitm.ac.in

Key words: Surrogate Modeling, Unsteady Aerodynamics, Proper Orthogonal Decomposition, Neural Network, Non-intrusive Reduced Order Model

Abstract. Cost-effective parameteric surrogate models of unsteady aerodynamic loads acting on a flapping wing are highly desirable. They would enable real time aerodynamic load prediction, multiobjective optimisation and optimal control of intelligent flapping wing flight devices. In the present work, a parametric surrogate modeling framework for unsteady aerodynamic loads based on a non-intrusive reduced order modeling approach is presented. The unsteady flow past a plunging 2D flat plate is considered where the aerodynamic load time histories are obtained for different plunging frequencies and amplitudes using a potential flow solver. The parametric non-intrusive reduced order model (p-NIROM) for the obtained loads is constructed using a combination of snapshot proper orthogonal decomposition (POD) for dimensionality reduction and a fully connected feed forward neural network (FCNN) for modeling the input parametric dependency. Both, linear and non-linear FCNN based p-NIROM are explored and compared on the basis of load time history reconstruction accuracy. The non-linear FCNN regression for the p-NIROM is observed to generalise well for unseen parametric instances as compared to the linear approach when a systematic data sampling strategy is adopted.

1 INTRODUCTION

Unsteady aerodynamics of flapping wings have been intensely researched over recent years [1]. This is due to their potential applications in efficient design of biomimetic flight devices. For such applications, accurately computing the unsteady aerodynamic loads is of high importance. Moreover, since such devices operate in a dynamic environment, real time load prediction across flow or structural parameters would enable lower turn around times for solving deisgn optimisation and optimal control problems. This is where, cost-effective surrogate models that generalise well in the parametric space would be beneficial.

Surrogate modeling of unsteady aerodynamic loads is an inverse problem where a cost-effective alternative model to the underlying full order model is developed with the given load data. As per Eldred *et al.* [2], surrogate models are of three types: data-fit, reduced order and heirarchical. The present study involves the second type of surrogate models where a parametric non-intrusive reduced order model (p-NIROM) [3] of the unsteady aerodynamic loads is

proposed. Typically, building a p-NIROM [3] involves the following steps: (i) data generation and input parameter space sampling (ii) dimensionality reduction, (iii) modeling parameteric dependency and regression and (iv) dynamics reconstruction for unseen parametric instances. The above steps can be further simplified into two phases in p-NIROM implementation: offline (training - steps (i)-(iii)) and online (testing - step (iv)) phases. In the context of unsteady flows, earlier works have mostly focused on traditional projection based approaches [4] towards non-intrusive reduced order modeling. However, recent works have explored a plethora of deep neural network (DNN) architectures for model order reduction of unsteady flows [3, 5–7] owing to their high expressivity and agility [8]. Although, pure deep learning based NIROMs that use either fully connected [7] or convolutional neural networks [3] are efficient in the testing phase, their training is costly owing to the large number of neural network parameters that need to be estimated. Recently, Fresca *et al.* [6] demonstrated how a prior reduction of the data using snapshot proper orthogonal decomposition (POD) [9] and then learning a reduced map from input to output data using a neural network lowered the training costs. Following the works of Fresca *et al.* [6], a p-NIROM of the lift coefficient data has been developed, given only a handful of different input kinematic parameters. To compensate for the deficiency of data, a sampling strategy based on the underlying structure in the data is also devised.

The structure of the paper is as follows: In section 2, the p-NIROM framework is discussed in the context of unsteady aerodynamic loads generated by a plunging flat plate. In section 3, efficacy of the proposed p-NIROM is demonstrated in predicting the unsteady aerodynamic lift coefficient time series for unseen kinematic parameters. Finally, conclusions are drawn along with the perspectives on further scope in section 4.

2 COMPUTATIONAL METHODOLOGY

The objective of this study, is to develop a cost efffective parametric non-intrusive reduced order model for the unsteady aerodynamic loads acting on a plunging flat plate. Specifically, the unsteady aerodynamic lift coefficient (C_L) data is considered for investigation. A p-NIROM is purely data driven and hence once trained, doesn't require any governing equations to be solved to predict the load time histories for a given input parameter [6]. As shown in the schematic figure 1, the process of constructing a p-NIROM involves the following steps: (i) data generation, (ii) dimensionality reduction, and (iii) regression and dynamics reconstruction for different input kinematic parameters.

Here, a potential flow theory based unsteady vortex lattice method (UVLM) solver is used for C_L time histories data generation. In the dimensionality reduction step, a snapshot data matrix using the C_L time series data is constructed first at different input kinematic parameters for a fixed time window. Snapshot POD is then used to obtain a truncated temporal POD basis matrix $\tilde{\boldsymbol{\Phi}}$ and kinematic parameters dependent expansion coefficient matrix $\tilde{\boldsymbol{A}}$. The FCNN is then trained to approximate a continuous map from input kinematic parameters vector $\boldsymbol{\mu}$ to the corresponding column vectors of $\tilde{\boldsymbol{A}}$, using a training dataset. Finally, the efficacy of the proposed p-NIROM is evaluated based on the reconstruction accuracy of C_L data for a testing dataset. Each of these steps are further discussed in sections 2.1-2.3, respectively.

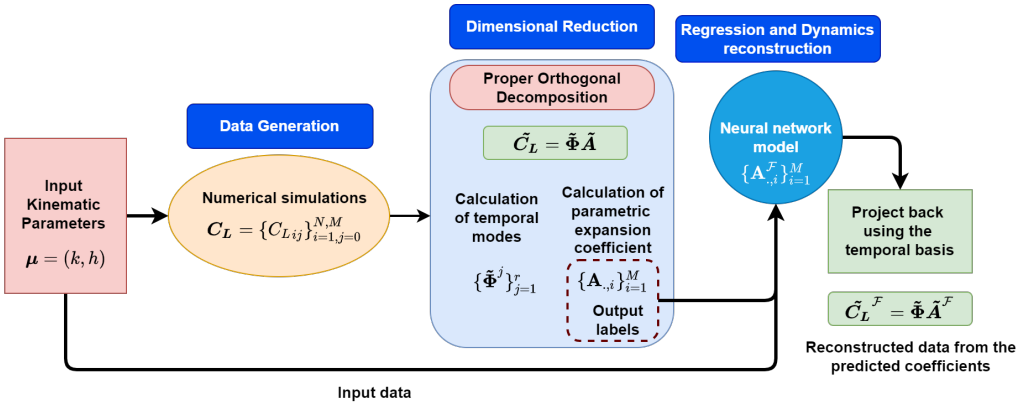

Figure 1: Schematic of the non-intrusive reduced order modeling strategy for the C_L time series data.

2.1 Data generation using UVLM

A sinusoidally plunging two-dimensional (2D) flat plate immersed in a uniform freestream has been considered here (see figure 2a). The plunging kinematics is mathematically modeled as follows

$$y(\hat{t}) = h_0 \sin(\omega \hat{t}), \tag{1}$$

$$\dot{y}(\hat{t}) = \omega h_0 \cos(\omega \hat{t}), \tag{2}$$

where, h_0 is dimensional plunging amplitude, $\omega = 2\pi f$ with f being plunging frequency, and $\hat{t}$ is dimensional time. Aligning with the earlier literature [10], the reduced frequency $k = \omega c/U_\infty$, non-dimensional plunging amplitude $h = h_0/c$ and non-dimesnional time $t = U_\infty \hat{t}/c$ are considered in further discussions, where, c and U_∞ are chord length of the flat plate and the free stream velocity, respectively.

UVLM is a method to solve for unsteady flow past flapping bodies based on the potential flow theory. In UVLM, the plunging thin flat plate is modeled as a sheet of discrete vortex panels as shown in figure 2b. Here, the free vortex elements released in the trailing edge, model the trailing wake patterns. Whereas, the bound vortex elements help to enforce the surface boundary conditions and are used to compute the pressure distribution on the flat plate. The pressure distribution is then integrated to compute the aerodynamic load coefficients [11]. The results from the present UVLM solver have been compared with that of Young [10] in figure 2c. It was observed that for $k \leq 10$, the peak C_L values obtained using the present UVLM solver match quantitatively well with that of Young [10]. It is important to note that, for $k > 10$, the results from the UVLM solver deviates from the Naviers-Stokes (NS) solver in Young [10]. Therefore, in the present work, the periodic vortex shedding regime ($kh \leq 1.0$) has been considered to generate C_L time histories for different input $k \leq 10$ and h values. As a result, the computational time can be further reduced as compared to a typical Navier-Stokes (NS) solver.

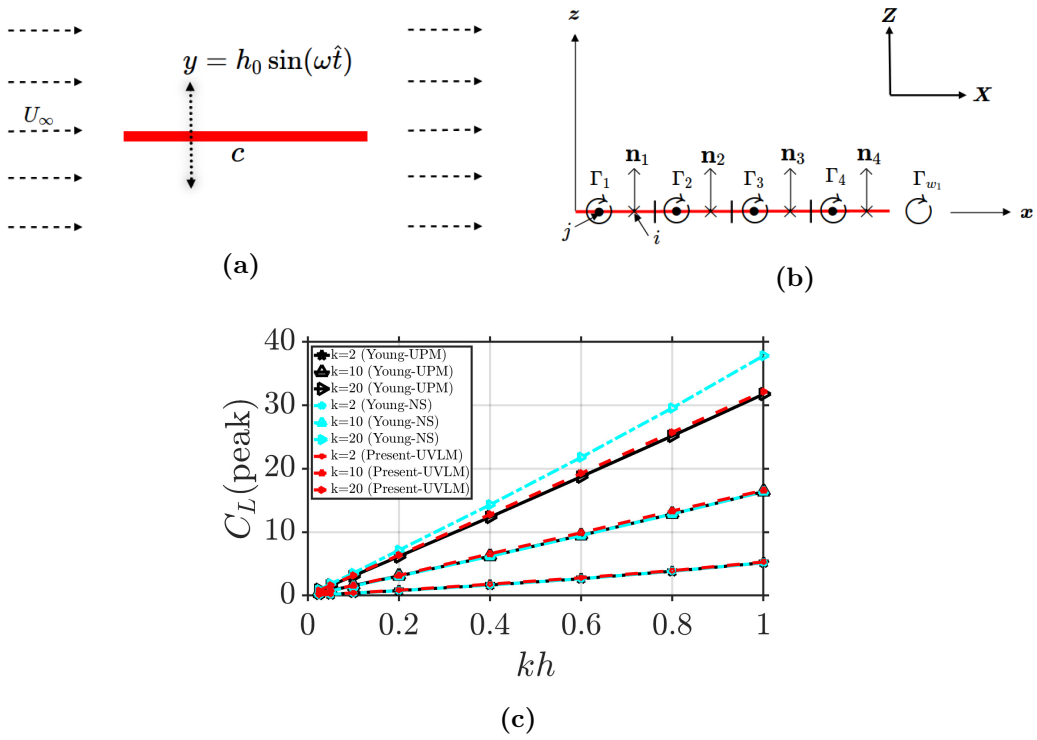

Figure 2: (a) Schematic of the plunging flat plate, (b) schematic of the discretisation for the unsteady vortex lattice method and (c) validation of peak C_L obtained from the current UVLM solver with those reported in the literature [10].

2.2 Snapshot proper orthogonal decompositon based dimensionality reduction

Dimensionality reduction of the obtained C_L time series data is carried out using snapshot POD [9]. Firstly, the snapshot data matrix $\boldsymbol{C_L} \in \mathbb{R}^{N\times M}$ is constructed such that

$$\boldsymbol{C_L} = [C_L(t,\boldsymbol{\mu_1})\;\; C_L(t,\boldsymbol{\mu_2})\cdots C_L(t,\boldsymbol{\mu_j})\;\cdots C_L(t,\boldsymbol{\mu_M})]_{N\times M}, \tag{3}$$

where, $C_L(t,\boldsymbol{\mu_j}) \in \mathbb{R}^N$ for $j = 1,2\cdots M$ is the lift coefficient time series data for a given parametric instance $\boldsymbol{\mu}_j = (k_j, h_j)$. Here, M and N are the number of parametric instances and time stamps, respectively, where $M << N$. A data correlation matrix $\boldsymbol{C} \in \mathbb{R}^{N\times N}$ is then constructed such that

$$\boldsymbol{C} = \frac{1}{N-1}\boldsymbol{C_L}\boldsymbol{C_L}^T. \tag{4}$$

The eigen decomposition of $\boldsymbol{C}$ results in the eigen vector matrix or the temporal POD modes $\boldsymbol{\Phi} \in \mathbb{R}^{N\times N}$, and a set of eigen values, $\boldsymbol{\lambda} \in \mathbb{R}^N$, where $\lambda_1 > \lambda_2 > \cdots > \lambda_j \cdots > \lambda_N$. Now, $\boldsymbol{\Phi}$ which is orthonormal in nature, can be used to project $\boldsymbol{C_L}$ onto a lower dimensional space thereby obtaining the parametric expansion coefficient matrix $\boldsymbol{A} \in \mathbb{R}^{N\times M}$ such that

$$\boldsymbol{A} = \boldsymbol{\Phi}^T\boldsymbol{C_L}. \tag{5}$$

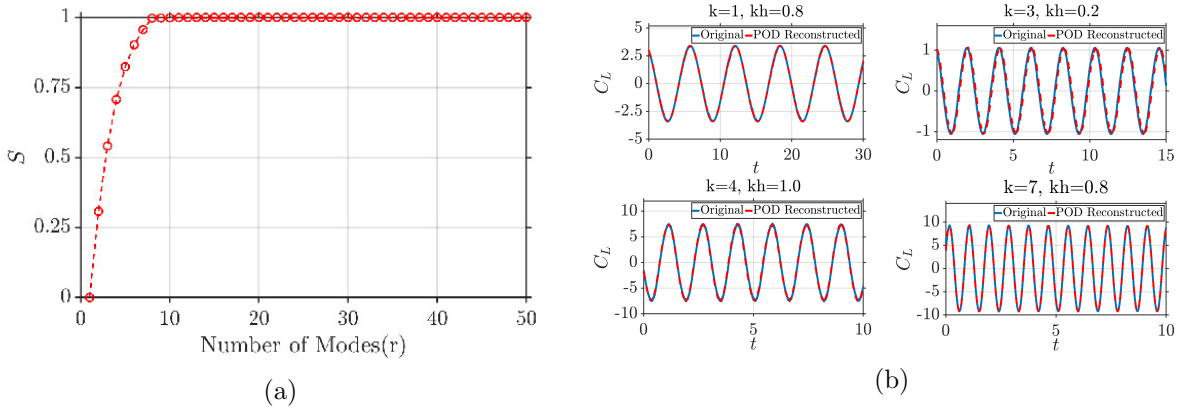

Figure 3: (a) Cumulative contribution of temporal POD modes and (b) comparison of ground truth and POD reconstructed C_L time series data for different input parameters.

Here, $\boldsymbol{A}$ depends solely on the input kinematic parameters $\boldsymbol{\mu}_j = (k_j, h_j)$ for $j = 1, 2, \cdots M$, therefore, independent of temporal effects such that

$$\boldsymbol{A}_{N\times M}(\boldsymbol{\mu}) = [\boldsymbol{a}_1(\boldsymbol{\mu_1}) \;\; \boldsymbol{a}_2(\boldsymbol{\mu_2}) \;\; \cdots \boldsymbol{a}_j(\boldsymbol{\mu_j}); \cdots \boldsymbol{a}_M(\boldsymbol{\mu_M})]_{N\times M}, \tag{6}$$

with $\boldsymbol{a}_j(\boldsymbol{\mu_j}) \in \mathbb{R}^N$ denoting the column vectors, simply written as $\boldsymbol{a}_j$ or $\boldsymbol{a}$ in further discussions. Low rank approximation $\tilde{\boldsymbol{C_L}}$ can be obtained in terms of truncated temporal POD modes $\tilde{\boldsymbol{\Phi}} \in \mathbb{R}^{N\times r}$ and expansion coefficient matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{r\times M}$ with its column vectors $\tilde{\boldsymbol{a}}_j \in \mathbb{R}^r$ for $j = 1, 2, \cdots M$ such that

$$\boldsymbol{C_L} \approx \tilde{\boldsymbol{C_L}} = \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{A}} \tag{7}$$

where, $\text{rank}(\tilde{\boldsymbol{C_L}}) = r \leq \text{rank}(\boldsymbol{C_L}) = M$. While $\tilde{\boldsymbol{\Phi}}$ is obtained by retaining the first r column vectors of $\boldsymbol{\Phi}$, $\tilde{\boldsymbol{A}}$ is obtained by retaining the first r rows of $\boldsymbol{A}$. For a good data reconstruction accuracy, the choice of r is crucial. This in turn depends on the cumulative contribution from the temporal POD modes denoted by S, which can be evaluated as a function of the eigen values λ_j for $j = 1, 2, \cdots N$, such that $S = \frac{\Sigma_{j=1}^{r}\lambda_j}{\Sigma_{j=1}^{N}\lambda_j}$. In our present study, $r = 15$ has been chosen for performing data reconstruction such that $S = 0.9999$ (see figure 3a). The comparison of the POD reconstructed C_L time series obtained using $r = 15$ modes with that of ground truth shows a very good match (see figure 3b).

Now, a FCNN can be used to model the input kinematic parametric dependency of $\tilde{\boldsymbol{A}}$ as a continuous function. The mathematical setup for the FCNN based regression is presented in the following section.

2.3 Neural network based regression

A FCNN is a simple neural network architecture where each layer's neurons are completely connected to its preceeding/succeeding layer of neurons [12]. As a regression model, a FCNN approximates the parametric dependency of $\tilde{\boldsymbol{a}}$ as a continuous function of any given $\boldsymbol{\mu}$. This is

achieved by using the discrete instances of kinematic parameters $\boldsymbol{\mu}_j$ for a $1 \leq j \leq M$ as input, and the corresponding POD computed expansion coefficient vectors $\tilde{\boldsymbol{a}}_j$ as the target output. Here, the FCNN is represented by $\mathcal{F}$ such that

$$\tilde{\boldsymbol{a}}_j^{\mathcal{F}} = \mathcal{F}(\boldsymbol{\mu_j}, \boldsymbol{\theta}), \tag{8}$$

where, $\boldsymbol{\theta}$ denotes the neural network parameters that are to be estimated by solving an error/loss minimisation problem (see section 2.4). The mathematical representation of $\mathcal{F}$ with N_h hidden layers and N_n hidden neurons, relating the input $\boldsymbol{\mu_j} \in \mathbb{R}^2$, network parameters ($\boldsymbol{\theta}$) and target output $\tilde{\boldsymbol{a}}_j^{\mathcal{F}} \in \mathbb{R}^r$ for $j = 1, 2, \cdots M$ can be written as follows

$$\boldsymbol{X} = \boldsymbol{\mu}_j \tag{9}$$

$$\boldsymbol{h}_1 = g(W_1 \boldsymbol{X} + b_1) \tag{10}$$

$$\boldsymbol{h}_l = g(W_l \boldsymbol{h}_{l-1} + b_l), \quad 2 \leq l \leq L-1 \tag{11}$$

$$\tilde{\boldsymbol{a}}_j^{\mathcal{F}} = W_L \boldsymbol{h}_{L-1} + b_L. \tag{12}$$

Here, $\boldsymbol{X}$ is the input layer, $\boldsymbol{h}_l$ for $l = 1, 2 \cdots N_h$ are the activation units at each l^{th} hidden layer with $g(.)$ denoting a non-linear activation function. Here, the neural network parameters are such that $\boldsymbol{\theta} = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{N_h}$, where, $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ denote the weights and biases of any l^{th} hidden layer, respectively. Now, given any input kinematic parameter instance $\boldsymbol{\mu}_j$, and the resultant FCNN approximation $\tilde{\boldsymbol{a}}_j^{\mathcal{F}}$, the corresponding C_L time histories can be reconstructed using the truncated temporal basis $\tilde{\boldsymbol{\Phi}}$ obtained earlier from snapshot POD (see section 2.2) such that

$$\tilde{C_L}^{\mathcal{F}}(t, \boldsymbol{\mu}) = \tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{a}}^{\mathcal{F}}. \tag{13}$$

In the present work, a single hidden layer FCNN with linear activation function is considered as a baseline for regression where equations (9)-(12) simplify to

$$\tilde{\boldsymbol{a}}_j^{\mathcal{F}} = \boldsymbol{W}_1^T \boldsymbol{\mu}_j + \boldsymbol{b_1}, \quad \text{for } j = 1, 2, \cdots M. \tag{14}$$

In the present study, results from both linear and non-linear FCNN based regression will be compared in section 3. Now, the methodology to train and evaluate the constructed FCNN is discussed next.

2.4 Model training and evaluation

For the FCNN alone, the input-output pairs are $\{\boldsymbol{\mu}_j, \tilde{\boldsymbol{a}}_j\}_{j=1}^{M}$. Whereas, for the entire p-NIROM, the input-output pairs are $\{\boldsymbol{\mu}_j, C_L(t, \boldsymbol{\mu_j})\}_{j=1}^{M}$. In order to avoid overfitting [12] of the FCNN, the data set consisting of input-output pairs $\{\boldsymbol{\mu}_j, \tilde{\boldsymbol{a}}_j\}_{j=1}^{M}$ is split into mutually exclusive training, validation and testing datasets, such that $M = M_{train} + M_{val} + M_{test}$. Here, these datasets are represented as $\{\boldsymbol{\mu}_j^{train}, \tilde{\boldsymbol{a}}_j^{train}\}_{j=1}^{M_{train}}$, $\{\boldsymbol{\mu}_j^{val}, \tilde{\boldsymbol{a}}_j^{val}\}_{j=1}^{M_{val}}$, and $\{\boldsymbol{\mu}_j^{test}, \tilde{\boldsymbol{a}}_j^{test}\}_{j=1}^{M_{test}}$, respectively. Similarly, for evaluating the p-NIROM, we have the datasets, $\{\boldsymbol{\mu}_j^{train}, C_L(t, \boldsymbol{\mu_j})^{train}\}_{j=1}^{M_{train}}$, $\{\boldsymbol{\mu}_j^{val}, C_L(t, \boldsymbol{\mu_j})^{val}\}_{j=1}^{M_{val}}$, and $\{\boldsymbol{\mu}_j^{test}, C_L(t, \boldsymbol{\mu_j})^{test}\}_{j=1}^{M_{test}}$. The training and validation FCNN datasets are used to estimate the optimal network parameters and hyperparameters (see table 1), respectively. The testing datasets for the FCNN and p-NIROM are used to evaluate the efficacy of

Table 1: Hyperparameter settings used for the non-linear and linear FCNN regression models to fit the expansion coefficient vectors $\tilde{\boldsymbol{a}}_j$ for $j = 1, 2 \cdots M$.

Hyperparameter	**Value (Range)**	
Optimiser	ADAM	
Epochs	1000	
Learning rate	1e-03	
Weights and biases initialisation	Kaiming [13]	
x	**linear**	**non-linear**
Activation function	Linear	ReLU
Hidden neurons per layer	100	$60, 70, 80, 90, 100$
Hidden layers	1	$2, 3, 4$

the p-NIROM. The numerical details of the dataset split are presented in section 3, where a sampling strategy is discussed.

Here, the loss function ($\mathcal{L}$) considered to train the FCNN is the mean squared error ($\boldsymbol{MSE}_{train}$) between predicted $\tilde{\boldsymbol{a}}_j^{\mathcal{F}}$, and training data $\tilde{\boldsymbol{a}}_j^{train}$, which can be written as follows

$$\mathcal{L} = \boldsymbol{MSE}_{train} = \frac{1}{M_{train}} \sum_{i=1,j=1}^{r,M_{train}} \|\tilde{\boldsymbol{a}}_{ij}^{train} - \tilde{\boldsymbol{a}}_{ij}^{\mathcal{F}}\|_{L_2}^2. \tag{15}$$

A stochastic gradient descent based optimisation algorithm, ADAM [14] is used to minimize $\mathcal{L}$ and estimate the optimal network parameters $\boldsymbol{\theta}$. Here, the gradients of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ are computed using back propagation and automatic differentiation [15] through a Tensorflow [16] implementation. To determine the effect of hyperparameter variation and whether the model is overfitting, the loss is computed on the training and validation datasets at every iteration and have been tracked. Finally, the efficacy of the p-NIROM is evaluated based on the mean squared error ($\boldsymbol{MSE}_{test}$) in $\tilde{\boldsymbol{a}}_j^{\mathcal{F}}$ and averaged root mean squared error ($\boldsymbol{aRMSE}_{test}^{C_L}$) between $\tilde{C_L}^{\mathcal{F}}(t, \boldsymbol{\mu}_j)$ and $C_L(t, \boldsymbol{\mu}_j)^{test}$ over all the M_{test} samples as shown in the following expressions

$$\boldsymbol{MSE}_{test} = \frac{1}{M_{test}} \sum_{i=1,j=1}^{r,M_{test}} \|\tilde{\boldsymbol{a}}_{ij}^{test} - \tilde{\boldsymbol{a}}_{ij}^{\mathcal{F}}\|_{L_2}^2, \quad \text{and} \tag{16}$$

$$\boldsymbol{aRMSE}_{test}^{C_L} = \frac{1}{M_{test}} \sum_{j=1}^{M_{test}} \sqrt{\frac{\Sigma_{i=1}^{N} \|C_L(t_i, \boldsymbol{\mu}_j)^{test} - \tilde{C_L}^{\mathcal{F}}(t_i, \boldsymbol{\mu}_j)\|_{L_2}^2}{N}}. \tag{17}$$

3 RESULTS AND DISCUSSION

The key results for both the linear and nonlinear FCNN regression based p-NIROMs are discussed in this section. Here, C_L time histories are first generated using the UVLM solver.

A time step size of $\Delta t = 0.005$, and discretization of the flat plate into 250 panels with bound vortices are decided after carrying out appropriate convergence tests. In the present study, $kh = [0.1, 0.2, 0.4, 0.6, 0.8, 1.0]$ and $k = [1, 2, 3, 4, 5, 6, 7]$ are chosen such that $M = 42$ different input parameter sets $\boldsymbol{\mu}_j = (k_j, h_j)$ for $j = 1, 2, ...M$ and corresponding C_L time histories have been obtained. For each input kinematic parameter setting, the solver is marched in time for $N = 7000$ time stamps.

For a p-NIROM, the input parameter space sampling plays an important role in developing a generalisable model for unseen parametric instances [17]. The importance of sampling is compounded especially in the limited data regime like in the present work where there are only $M = 42$ data samples. The line plots and contour plots of POD obtained expansion coefficient

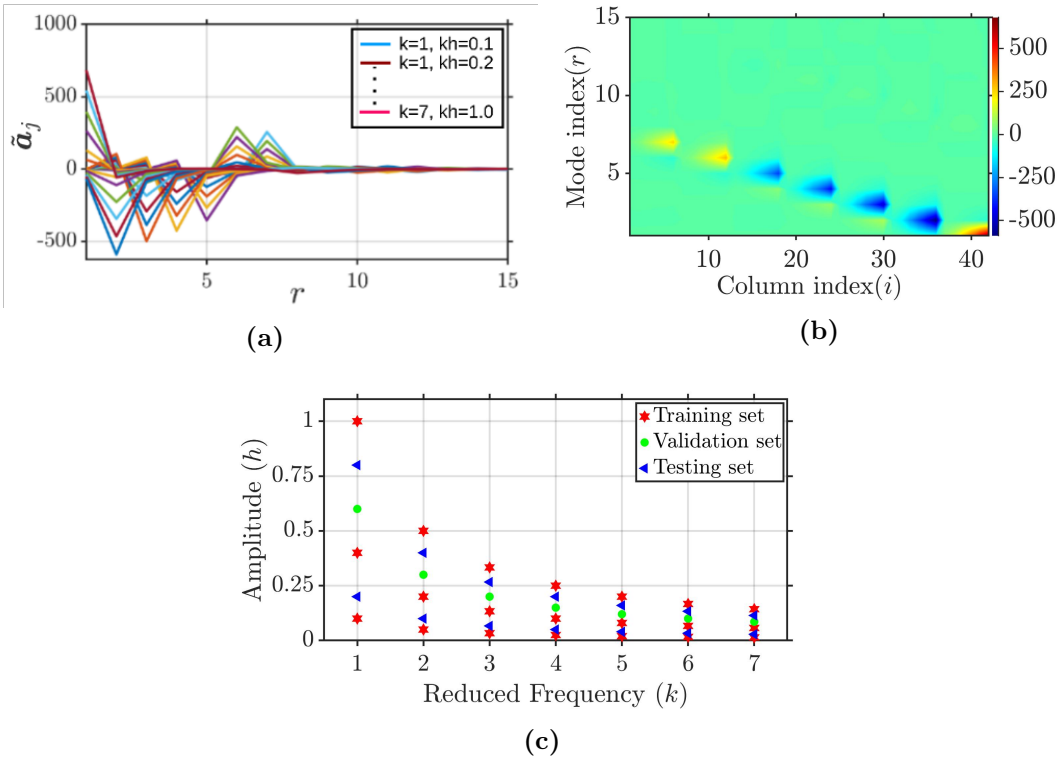

Figure 4: (a) Line plots of the column vector entries of the expansion coefficient matrix $\tilde{\boldsymbol{A}}$ for each parametric instance and (b) contour plot of $\tilde{\boldsymbol{A}}$ showing an overall sparse and sharp localised jumps at specific row and column indices and (c) input parameter space sampling using a systematic approach.

vectors $\tilde{\boldsymbol{a}}_j$ and matrix $\tilde{\boldsymbol{A}}$, shown in figures 4a and 4b, indicate localised high valued jumps. Moreover, for a given k, these jumps are located at the same row index of $\tilde{\boldsymbol{A}}$ and grow in their absolute value only as a function of h. This is because, for a given k, the C_L amplitudes are a function of the plunging amplitude h as seen in figure 2c. Hence, instead of choosing

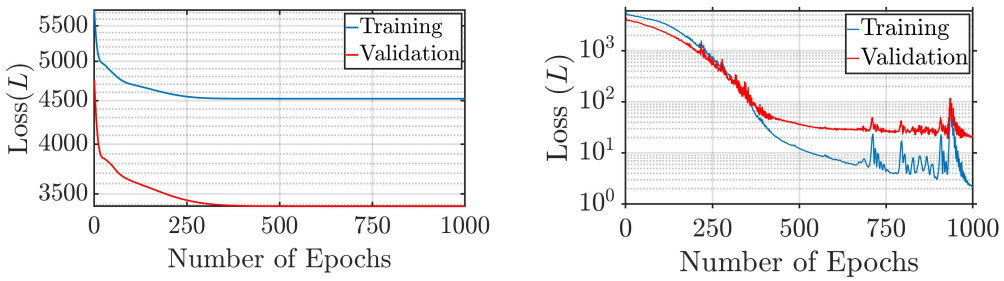

Figure 5: Comparison of the loss convergence behavior for (a) linear FCNN and (b) non-linear FCNN based regression.

the training, validation and testing datasets randomly, particular values of k and h for each dataset are chosen in a systematic manner such that $(kh)^{train} = [0.1, 0.4, 1.0]$, $(kh)^{val} = [0.6]$ and $(kh)^{test} = [0.2, 0.8]$, and $M = M_{train} + M_{val} + M_{test}$ with $[M_{train}, M_{val}, M_{test}] = [21, 7, 14]$; see figure 4c.

As the model is expected to perform well on the training data, only the results for the testing dataset prediction are presented here for the sake of brevity. While a linear activated FCNN is considered as a baseline, ReLU activation function is considered for the non-linear FCNN regression (see equation (9)), since the matrix $\tilde{\boldsymbol{A}}$ is sparse and consists of discrete jumps (see figure 4a and 4b) which otherwise can't be captured well by smooth activation functions like tanh [12]. The hyperparameters chosen for the non-linear FCNN regression are mentioned in table 1. However, the results presented are for the best non-linear FCNN configuration with $N_h = 3$ and $N_n = 90$. The training and validation losses for the linear FCNN rapidly decrease for the first few iterations and then plateau immediately (see figure 5a). Whereas, training and validation losses for the non-linear FCNN (see figure 5b) decrease by atleast 2 orders of magnitude after 250 epochs of training. Moreover, it is surprising that the validation loss although very high ($O(10^3)$), is still lower than training loss for linear FCNN when compared to non-linear FCNN. A possible reason for this is that, the training dataset (see figure 4c) correspond to the minimum and maximum C_L amplitudes in the overall dataset, which indirectly gets reflected in $\tilde{\boldsymbol{a}}_j^{train}$ for $j = 1, 2, \cdots M_{train}$. Whereas, the validation and testing datasets (see figure 4c) contain the input-output pairs values within the range already observed in the training dataset. Also, the linear regression model is incapable of capturing discrete jumps in the values of $\tilde{\boldsymbol{a}}_j^{test}$ for $j = 1, 2, \cdots M_{test}$, and as a result, C_L reconstruction is highly inaccurate as reflected in the high $\boldsymbol{aRMSE}_{test}^{C_L}$ with an order of $O(1)$; see table 2. However, the non-linear FCNN based p-NIROM performs significantly better in training, validation (see figure 5b) and testing (see figure 6 and table 2) as compared to linear FCNN based p-NIROM. Notably, the mean squared errors in $\tilde{\boldsymbol{a}}_j^{\mathcal{F}}$ shown in table 2 are high because $\tilde{\boldsymbol{a}}_j$ for $j = 1, 2, \cdots M$ are not normalised due to which the order of magnitude of entries in $\tilde{\boldsymbol{a}}_j$ is $O(10^2)$ (see figures 4a and 4b).

It is interesting to note that C_L time history reconstructions from the predicted $\tilde{\boldsymbol{a}}_j^{\mathcal{F}}$ for

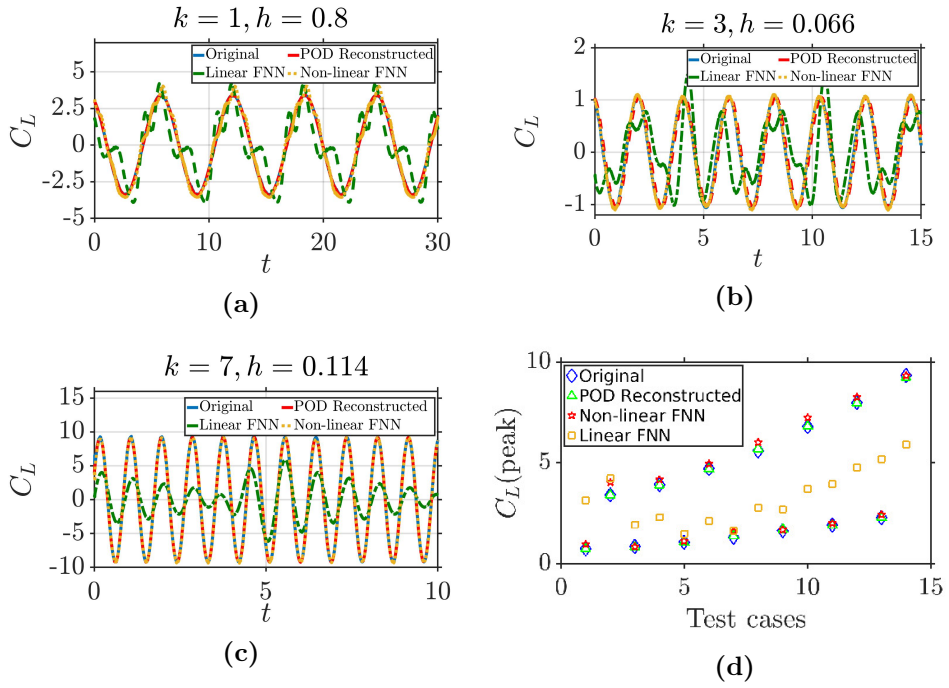

Figure 6: Comparison of (a)-(c) ground truth C_L time series data with POD, linear FCNN and Non-linear FCNN reconstructed C_L time series for three different test input parameters and (d) peak C_L for all the test cases for POD, linear FCNN, nonlinear FCNN and ground truth data.

Table 2: Model performance metrics highlighting the best performing nonlinear FCNN model as compared to the linear FCNN.

Model	N_h	N_n	MSE_{train}	MSE_{val}	MSE_{test}	$aRMSE_{test}^{C_L}$
Nonlinear FCNN	2	80	8.158	69.426	34.005	–
Nonlinear FCNN	**3**	**90**	**2.235**	**19.870**	**18.564**	**0.1787**
Nonlinear FCNN	4	70	2.244	44.936	34.543	–
Linear FCNN	1	100	4522.81	3386.36	3846.54	2.03

$j = 1, 2, \cdots M_{test}$ is qualitatively and quantitatively matching quite closely with the ground truth irrespective of the plunging frequency k chosen at test phase (see figures 6a - 6c). Moreover, the model is also able to capture the peak values of the C_L time series data very well in all the test cases as seen in figure 6d. This shows that, in addition to capturing the discrete jumps, the

non-linear FCNN based p-NIROM generalises well to unseen input parameter instances.

As far as the computational budget is concerned, the FCNN based regression model approximates only a reduced map between the input parameter space and the parametric expansion coefficients. Therefore, in comparison to the full map approximation between the input parameters and C_L, the p-NIROM works with lesser target data points ($q \times M$ now, versus $N \times M$ where $q << N$) for training and testing. As a result, the FCNN requires lesser network parameters to train and thereby resulting in lower training times. For 1000 epochs of training, the model takes approximately $t_{train} \approx 5 - 10$ minutes on a GPU system with a T40 NVIDIA GPU card and it takes $t_{test} \approx 5$ seconds to test the trained model as compared to $2 - 3$ hours of data generation for each parametric instance using the UVLM solver. This shows the potential benefit of developing p-NIROMs that would enable real time prediction.

4 CONCLUSIONS

A hybrid POD-FCNN based p-NIROM has been proposed in the present study for surrogate modeling of unsteady aerodynamic loads acting on a plunging airfoil. The key contributions of the work are: (i) a strategic parameter space sampling methodology making use of the underlying structure of the POD expansion coefficient matrix, and (ii) a generalisable p-NIROM solely dependent on input kinematic parameters thereby making the neural network agnostic to the temporal scales present in the time series data. The nonlinear POD-FCNN based p-NIROM performs better than the linear version in reconstructing the load time histories accurately across different plunging frequencies and amplitudes from the testing dataset. While current model is capable of reconstructing the lift coefficient data spreading across different frequencies with a low training budget, the ability to reuse the available temporal basis to reconstruct lift coefficients for unseen parameters for which temporal basis has not been precomputed, and needs to be tested in future.

REFERENCES

[1] Diana D Chin and David Lentink. Flapping wing aerodynamics: from insects to vertebrates. *Journal of Experimental Biology*, 219(7):920–932, 2016.

[2] Michael Eldred, Anthony Giunta, and S Collis. Second-order corrections for surrogate-based optimization with model hierarchies. In *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 4457, 2004.

[3] Stefania Fresca, Luca Dede, and Andrea Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *Journal of Scientific Computing*, 87(2):1–36, 2021.

[4] Jian Yu, Chao Yan, and Mengwu Guo. Non-intrusive reduced-order modeling for fluid problems: A brief review. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(16):5896–5912, 2019.

[5] Sandeep R Bukka, Allan Ross Magee, and Rajeev K Jaiman. Deep convolutional recurrent autoencoders for flow field prediction. In *International Conference on Offshore Mechanics*

and Arctic Engineering, volume 84409, page V008T08A005. American Society of Mechanical Engineers, 2020.

[6] Stefania Fresca and Andrea Manzoni. Pod-dl-rom: enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388:114181, 2022.

[7] Hugo FS Lui and William R Wolf. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *Journal of Fluid Mechanics*, 872:963–994, 2019.

[8] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[9] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of applied mathematics*, 45(3):561–571, 1987.

[10] John Young. *Numerical simulation of the unsteady aerodynamics of flapping airfoils*. PhD thesis, University of New South Wales, Australian Defence Force Academy, School of . . . , 2005.

[11] Joseph Katz and Allen Plotkin. *Low-speed aerodynamics*, volume 13. Cambridge university press, 2001.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[15] Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

[16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. pages 265–283, 2016.

[17] Michalis Frangos, Youssef Marzouk, Karen Willcox, and Bart van Bloemen Waanders. Surrogate and reduced-order modeling: a comparison of approaches for large-scale statistical inverse problems. *Large-Scale Inverse Problems and Quantification of Uncertainty*, pages 123–149, 2010.