Techniques for Improving Neural Network-based Aerodynamics Reduced-order Models

Qingzhao Wang,[*] Renato R. Medeiros,[†] Carlos E. S. Cesnik,[‡] Krzysztof J. Fidkowski,[§]

University of Michigan, Ann Arbor, MI, 48109, USA

Joël Brezillon[¶] and Hans M. Bleecke[‖]

Airbus Operations S.A.S., Toulouse, Cedex 09, France

Neural networks are applied to create reduced-order models (ROMs) for high-fidelity, nonlinear steady and unsteady CFD aerodynamic simulations by non-intrusively relating outputs (dependent variables) to inputs (independent variables), regardless of the complex physics involved. The present study is conducted with increasing complexity in the aerodynamic system and the corresponding neural network-based ROMs. The primary goal of this paper is to introduce the development and demonstration of new techniques for improving the predictive performance of the reduced-order models. In particular, the benefits of a physics-driven approach for selecting the training inputs and delay states are investigated. The adaptive ROM system is established by performing error estimation through two statistical techniques: cross-validation and bootstrapping.

I. Introduction

In the past two decades, significant effort has been devoted to the development of reduced-order models for efficient aerodynamic computations. In many cases, it is possible to perform high-fidelity computations using CFD codes, but the cost of extensive parametric explorations or long dynamic responses is not affordable. In this context, reduced-order models allow computations at a fraction of the cost of the more complex simulations. However, the quality of these approximate models is a fundamental question that is not always investigated in detail. The present work proposes techniques for improving accuracy and efficiency of the ROM that is generated as an artificial neural network function, and addresses the issue of uncertainty calculation and model adaptation.

Reduced-order models may be generated using multiple techniques. There exist both projection-based and output-based approaches. While the projection-based methods work directly with the system of equations involved, output-based methods are non-intrusive, using only inputs and outputs to produce approximate models. This paper develops approximations using the latter approach. Even in the context of output-based ROMs, there are many different strategies that can be followed. One can choose from linear state-space identification techniques,[1–3] Volterra-based approaches,[4–6] nonlinear modeling with artificial neural networks[7–12] or Gaussian processes,[13,14] and rational-function approximations,[15,16] for instance. This particular work is focused on reduced-order models generated by artificial neural networks, which are promising for real-time applications.

The popularity of artificial neural networks applications primarily lies in the capabilities of real-time learning and representing accurately the reference function.[17] However, the samples of expensive, high-fidelity models are often inadequate. It is critical to have estimates of the errors arising from inadequate sampling

[*]Postdoctoral Research Fellow, Department of Aerospace Engineering, qzwang@umich.edu, AIAA Member.
[†]Ph.D. Candidate, Department of Aerospace Engineering, renatorm@umich.edu, AIAA Student Member.
[‡]Professor, Department of Aerospace Engineering, cesnik@umich.edu, AIAA Fellow.
[§]Associate Professor, Department of Aerospace Engineering, kfid@umich.edu, AIAA Senior Member.
[¶]Flight Physics Capability, Toulouse, joel.brezillon@airbus.com
[‖]Department of Aeroelasticity, Bremen, hans.bleecke@airbus.com

to evaluate the predicted results and to improve adaptively robustness and accuracy of the neural network fitting model.[18] Convergence analysis can provide estimates of error due to inadequate sampling, which is represented by the difference between the ideal network function (extrapolated to an infinite number of samples) and an estimated network function. The implementation of convergence analysis is straightforward but costly. The more sophisticated and relatively inexpensive statistical techniques of cross-validation[19–21] and bootstrapping[18, 22–25] play an active role in quantifying the prediction errors from various sources, such as inadequate training, inappropriate selection of inputs, and/or insufficient history of inputs for dynamic systems. Comparison of different estimates have shown that the apparent error computed at the sample points is biased. The commonly used k-fold cross-validation is nearly unbiased, but has large variance,[26] which is due to the disparity between training datasets when k is too small. Other estimates based on bootstrapping techniques compromise between the bias and variance and present the best performance in many cases, especially for small sample size. For time-series models, cross-validation[27–29] and bootstrapping techniques[30–32] are still applicable. The associated stability issue has also been discussed extensively,[33–35] and questions of stability should be taken into account for unsteady simulations using the recurrent neural networks (RNNs).

The rest of this paper is organized as follows. Section II introduces the fundamentals of neural networks and error estimation of the predicted values using the neural network models. Section III describes the test problems adopted for steady and unsteady reduced-order models, respectively. Applications of neural networks to the described test cases follow in Section IV. The effects of inputs selection and delays determination on the accuracy of the resulting ROMs are investigated using the measures provided by error estimation techniques. Section V concludes the implementation of neural networks with error estimation for high-fidelity CFD models of complex aerodynamics systems and discusses the future research directions.

II. Theoretical Formulation

A. Architecture of Neural Networks

As discussed above, the prediction ROMs used in this study are artificial neural networks, which are suitable to large-scale, complex aerodynamic problems due to the ability of training the model with large datasets.

For the steady computations, feedforward neural networks are used, while for unsteady computations recurrent neural networks are chosen. In both cases, the neurons in the hidden layers present a hyperbolic tangent sigmoid activation function. An illustration of the feedforward architecture is presented in Fig. 1, where $\mathbf{u}$ is the input vector, $\mathbf{y}$ represents the output, and W and B represent the weights and bias relative to the neurons following the arrows. The hidden layer has neurons with nonlinear activation functions, while the output layer represents a linear function.

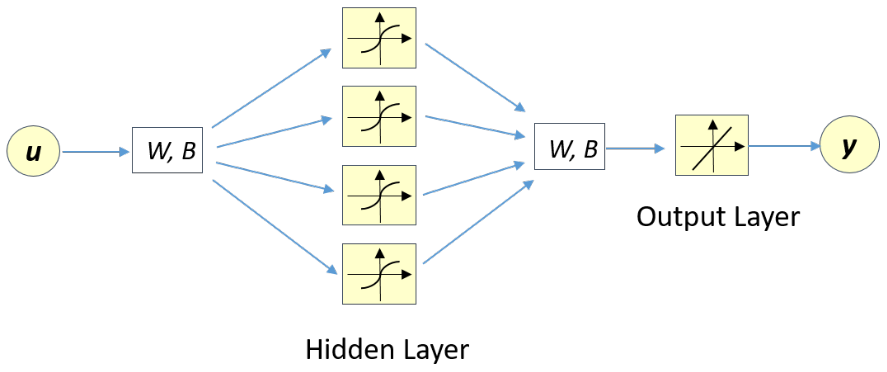


Figure 1: Architecture of the feedfoward neural network used for steady computations.

For unsteady computations, it is necessary to use a history of inputs and outputs. In this case, a similar architecture is used, but there is a feedback of the outputs of previous instants in time. This structure is a recurrent neural network, illustrated in Fig. 2. The inputs are a collection of current and previous data along the time-series. The outputs $\mathbf{y}$ may also be fed back into the RNN.

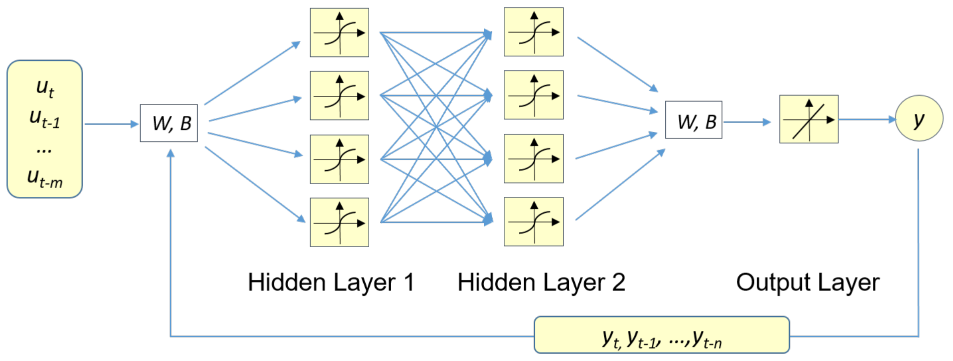


Figure 2: Architecture of the recurrent neural network used for unsteady computations.

In terms of ROMs, the recurrent neural network represents a nonlinear autoregressive network with exogenous inputs (NARX), which can be expressed as

$$\mathbf{y}\,(t) = f\left(\mathbf{u}(t), \mathbf{u}(t-d_{u1}), ..., \mathbf{u}(t-d_{um}), \mathbf{y}(t-d_{y1}), ..., \mathbf{y}(t-d_{yn})\right), \tag{1}$$

where there are m input delay states denoted by d_{u1} through d_{um}, and n feedback delay states denoted by d_{y1} through d_{yn}. It is not mandatory that the delay states are all consecutive. The feedback delay states do not necessarily align with the input delay states. As it will be explored in the paper, it is necessary to include inputs meaningful for the problem into consideration, and samples at different times in the past may be more effective than the most recent inputs or outputs.

Along the process of training the neural networks, backwards propagation is used to calculate the sensitivities of the outputs with respect to the weights. The Neural Network Toolbox of Matlab® is used to perform the training, using either the Levenberg-Marquardt algorithm or Bayesian regularization.

B. Field Representation with Proper Orthogonal Decomposition

In the training phase of neural network modeling, the variables of interest could be a set of scalars or a field. In this paper, the dependent variables with a distribution are expressed in terms of the basis vectors acquired from the Proper Orthogonal Decomposition (POD) method. The POD method extracts the basis vectors, known as POD modes, to provide a compact representation of the high-dimensional space.

Given a set of snapshot data, $\boldsymbol{S} = [\boldsymbol{S}^1, \boldsymbol{S}^2, ...\boldsymbol{S}^{n_s}]$, with each $\boldsymbol{S}^i$ a column vector of m elements, the POD basis modes are the eigenvectors $\boldsymbol{U}^j$ of the following problem:

$$\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{U}^j = \lambda_j \boldsymbol{U}^j \quad \text{for} \quad j = 1, ..., n_s. \tag{2}$$

The matrix $\boldsymbol{S}\boldsymbol{S}^T \in \mathbb{R}^{m\times m}$ is relatively large and solving for the eigenvalues is expensive. However, since usually $n_s < m$, an equivalent set of eigenvalues can be obtained by solving the symmetric eigenvalue problem of the matrix $\boldsymbol{S}^T\boldsymbol{S} \in \mathbb{R}^{n_s\times n_s}$:

$$(\boldsymbol{S}^T\boldsymbol{S})\boldsymbol{V}^j = \lambda_j \boldsymbol{V}^j \quad \text{for} \quad j = 1, ..., n_s. \tag{3}$$

The POD modes $\boldsymbol{U}^j \in \mathbb{R}^m$ can be then recovered by the following transformation from $\boldsymbol{V}^j \in \mathbb{R}^{n_s}$:

$$\boldsymbol{U}^j = \frac{1}{\lambda_j}\boldsymbol{S}\boldsymbol{V}^j \quad \text{for} \quad j = 1, ..., n_s. \tag{4}$$

By selecting the POD modes that contain as much energy as possible, corresponding to the highest eigenvalues, the POD representation of the snapshot data will provide desirable accuracy.

C. Selection of Inputs for ROM training

One of the aspects of ROM training addressed in this study is the selection of inputs. Since there are different phenomena involved in the physical process, considering meaningful inputs is essential to achieve ROMs with adequate accuracy and stability, especially in the case of unsteady computations.

It is important to consider both the kinds of inputs and the points in the previous history that are meaningful to the prediction. The fundamental physics and mathematical models of the problem can help define them. Consider, for example, the case of a thin-airfoil in incompressible flow performing pitch and plunge motion. The lift may be calculated using the Wagner's indicial response function ϕ as

$$L = \pi \rho b^2 (\ddot{h} + U\dot{\alpha} - ba\ddot{\alpha}) - 2\pi\rho U b \left[w_{3/4c}(0)\phi(s) + \int_0^s \frac{dw_{3/4c}(\sigma)}{d\sigma}\phi(s-\sigma)d\sigma \right], \tag{5}$$

where ρ is the air density, b is the airfoil semi-chord, U is the freestream velocity, a is the position of the axis for the pitch motion, relative to the mid-chord position, measured as a fraction of the semi-chord, h is the vertical position of the pitch axis, in the negative direction, and α is the angle of attack. The convolution integral is calculated with the non-dimensional time σ, defined in terms of the dimensional time t as

$$s = \frac{tU}{b}. \tag{6}$$

An important term for the convolution integral in Eq. 5 is the flow normal velocity at 3/4 of the chord, defined as

$$w_{3/4c} = -\left[\dot{h} + U\alpha + b\left(\frac{1}{2} - a\right)\dot{\alpha} \right]. \tag{7}$$

Using this formulation for the lift, there are two important terms: one corresponding to the added mass of the airfoil and the other corresponding to the circulatory lift, including the wake effects via a convolution integral. While the added mass is instantaneous, depending only on motion and its derivatives at the time of the calculation, the circulatory term depends on the motion history. Since the Wagner's function decays in time, it can be considered as a finite memory of the system. This assertion may be better appreciated if the lift of Eq. 5 is integrated by parts:

$$L = \pi \rho b^2 (\ddot{h} + U\dot{\alpha} - ba\ddot{\alpha}) - 2\pi\rho U b \left[w_{3/4c}(s)\phi(0) + \int_0^s \frac{d\phi(\sigma)}{d\sigma} w_{3/4c}(s-\sigma)d\sigma \right]. \tag{8}$$

From Eq. 8, the influence of a previous value of $w_{3/4c}$ on the lift at the current time decays according to the derivative of the Wagner's function, $\frac{d\phi}{d\sigma}$. The cut-off time for influence can be estimated based on approximations of the Wagner's function, such as the formula given by Venkatesan and Friedmann:[36]

$$\phi(\sigma) = 1.0 - 0.203\mathrm{e}^{-0.072\sigma} - 0.236\mathrm{e}^{-0.261\sigma} - 0.06\mathrm{e}^{-0.8\sigma}. \tag{9}$$

Using this equation, the non-dimensional time for a cut-off where the influence of the downwash at 3/4 of the chord decays to 5% of its initial value is $s = 15.0$, as illustrated in Fig. 3. That means the circulatory lift may be well-approximated if the selected inputs from the past history fall within this range, allowing to reconstruct the convolution integral.

Even for more complex physics such as phenomena described by the Navier-Stokes equations, convection is responsible for this finite memory feature, and this cut-off approach is still valuable.

D. Error Estimation Techniques

In order to evaluate the prediction performance of neural network models, the present work employs cross-validation and bootstrapping techniques for both steady and unsteady test cases. Various error estimates based on these techniques are calculated and compared for a comprehensive understanding of uncertainties in the prediction processes.

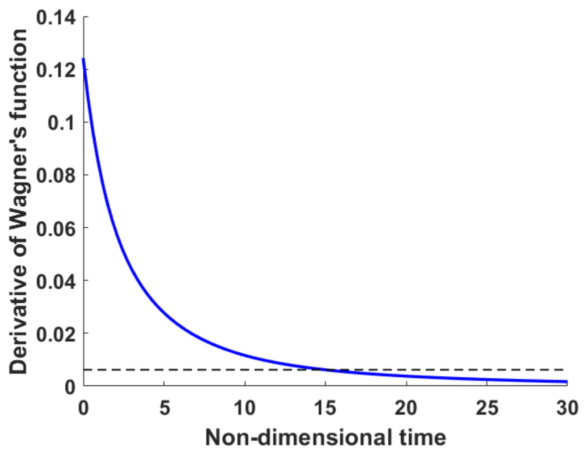


Figure 3: Derivative of Wagner's function and cut-off time of influence.

1. *k-fold cross-validation*

Among a variety of cross-validation methods, the k-fold cross-validation is adopted as a compromise of performance and efficiency. The sample dataset is first shuffled to maintain randomness and partitioned into k equal-sized subsets. Each subset serves as a test dataset only once with all the other subsets participating in the training process. Repeating this process yields k trained models and k sets of predicted values corresponding to k test datasets. The average of errors over all the test datasets approximates the prediction error. The statistical metrics of the prediction error give an insight into the ability of the predictive model to generalize to an independent (unknown) dataset on which the high-fidelity CFD or analytical solutions are unavailable.

2. *Bootstrapping*

In the cases where the underlying distribution of the sample is approximated, the bootstrapping-technique-based error estimates are often the best option to quantify and classify the errors from different sources. As a resampling method, the bootstrap constructs B samples, called bootstrap samples, by randomly selecting n sample data (snapshots) with replacement from the original sample of the same size n. For the original sample, let the n snapshots be $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ and denote the ensemble of the weights W and bias b by θ. Using Tibshirani's notation,[24] the j^{th} bootstrap sample will have the form $\{(x_1^{*j}, y_1^{*j}), \ldots, (x_n^{*j}, y_n^{*j})\}$ and the associated model parameters θ^{*j}. Considering all B bootstrap samples, the sample standard deviation at the i^{th} sample location is given by

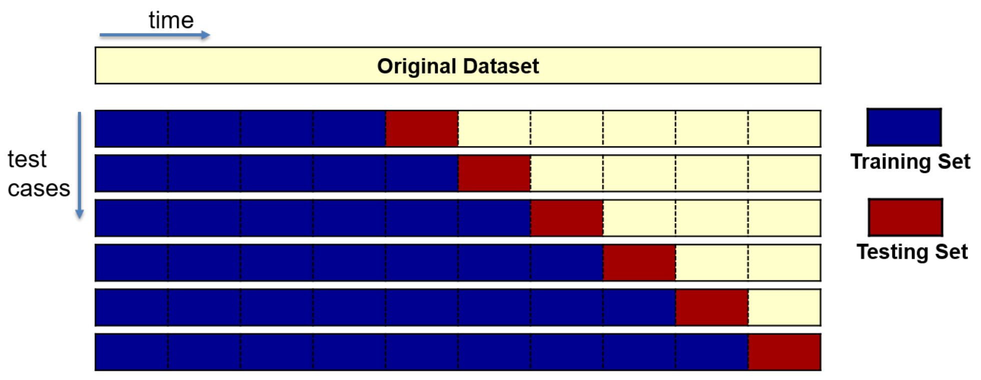
$$\hat{\sigma}(y(\boldsymbol{x_i})) = \sqrt{\frac{1}{B-1}\sum_{j=1}^{B}\left[y(\boldsymbol{x_i}, \theta^{*j}) - \frac{1}{B}\sum_{k=1}^{B} y(\boldsymbol{x_i}, \theta^{*k})\right]^2}, \tag{10}$$

which is primarily a result from randomness in the training process.

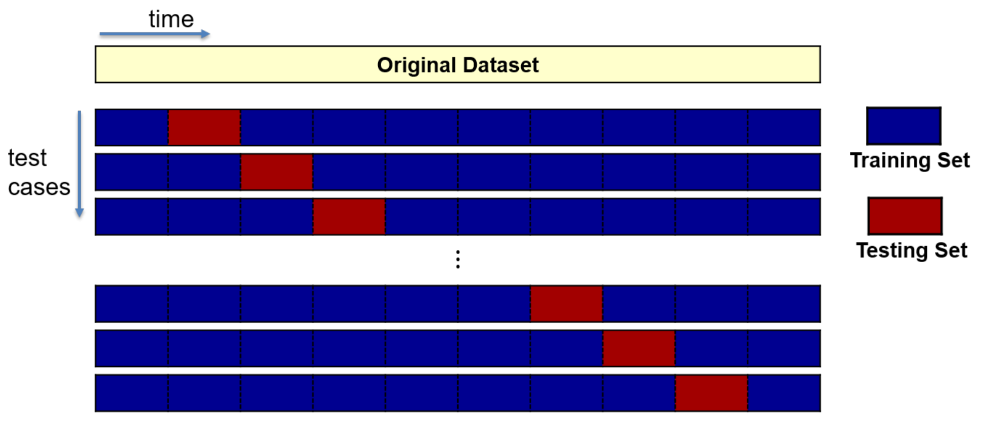
In the context of neural networks, not only does the bootstrap estimate of prediction error prevail over other error estimates, aggregation of the bootstrap samples can also improve the predictive model and avoid overfitting. This feature of the bootstrap technique is demonstrated by the steady test case in Section IV.

3. *Time-series cross-validation*

For time series applications, cross-validation preserves inherent correlation between time points by keeping the sequences in divided time intervals, in contrast to the k-fold cross-validation that shuffles data to create representative subsets. The following steps to conduct time-series cross-validation are analogous to the k-fold cross-validation. The original time sequence is partitioned into N equally sized subsets. According to the regular time-series cross-validation method,[27] training on the subsets evolves in a time-marching fashion as illustrated in Fig. 4a. Starting from the second subset, each subset time sequence will be used as a testing set to evaluate the prediction performance with all the prior subsets in time used for training. In the present work, a modification was made by predicting the responses in a testing set using all of the remaining $N-1$ subsets, similar to the *hv*-block cross-validation.[37] This block cross-validation method shown in Fig. 4b enables more subsets to be used for training. The prediction errors obtained from each testing set are averaged to yield a proper error estimate.



(a) Forward-chaining time-series cross-validation



(b) Block time-series cross-validation

Figure 4: Illustration of time-series cross-validation method.

4. Block bootstrap

Block bootstrap applies the resampling technique of bootstrapping to time series. To reserve the time dependence in the data, the original time sequence is divided into N blocks as in the time-series cross-validation method. These blocks are randomly chosen with replacement to construct a series of new time sequences, for example, 50 new time sequences. This paper presents results obtained using equally sized blocks and the same length of the original time sequence for all the reconstructed time sequences. Training on the 50 time sequences respectively leads to 50 RNN-based reduced-order models. The average of the prediction errors in all the 50 ROMs is considered as a good estimate of the true prediction error.

III. Test Cases

This paper presents application of the proposed techniques to neural network-based ROM generation for both steady and unsteady high-fidelity CFD simulations of aerodynamic systems. The k-fold cross-validation and bootstrapping techniques are demonstrated by studying a steady viscous flow over an NLR 7301 airfoil.

For the unsteady cases, investigation of the analytical solution of unsteady aerodynamics of a thin airfoil forced by pitching and plunging motions provides insight into the physics-driven selection of effective inputs for training the recurrent neural networks. The developed training technique is then applied to the ROM generation for aerodynamic responses of turbulent flow around the NLR 7301 airfoil to pitching and plunging excitations.

A. Steady Viscous Flow

In-house CFD software, xflow,[38, 39] has been used to provide high-fidelity solutions of the flow fields around the NLR 7301 airfoil as shown in Fig. 5. The incoming flow is at a Reynolds number of 2.12×10^6 and angle of attack of $2°$. Snapshots used for training of neural networks are generated by simulations at different Mach numbers ranging from 0.1 to 0.8 and covering both subsonic and transonic flow regimes. It is desirable to assign more sampling points in the region involving complex physics. Therefore, the underlying distribution of the snapshots reflects a probability density function with the highest value in the transition region, the lowest value at the Mach number of 0.1 and linearly varying in between.

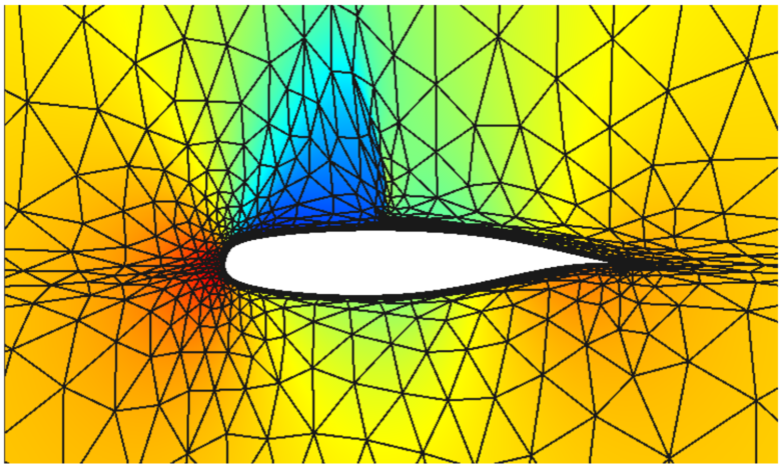
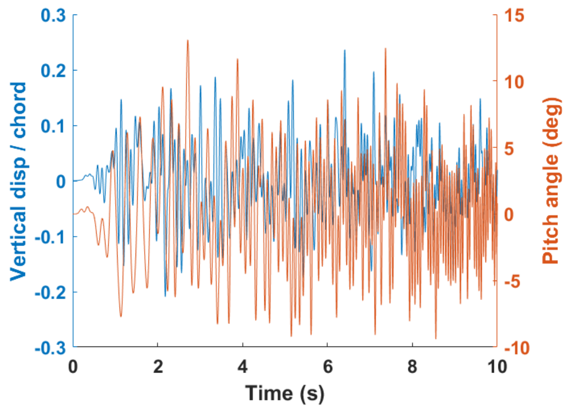


Figure 5: Geometry, mesh, and baseline solution for the NLR 7301 airfoil.

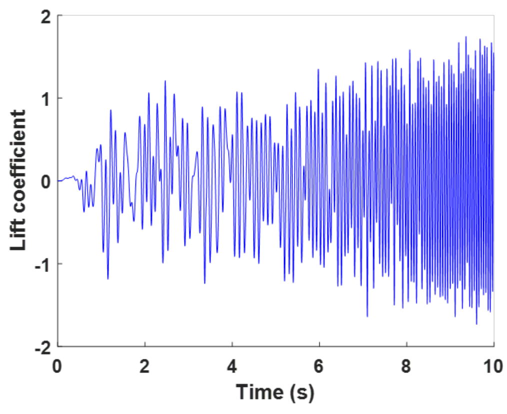
The neural network for this steady problem uses one hidden layer with four neurons, which has proven to be adequate for fitting reference functions with great accuracy. The outputs of this neural network include the lift and drag coefficients and the POD modes for the pressure coefficient distribution along the surface of the airfoil. During the training process, randomness exists in the initial weights generation and data assignment into training, validation and testing subsets. When higher POD modes are incorporated to reconstruct the pressure distribution, an adaptive training algorithm is designed to add flexibility of the number of neurons adopted.

B. Unsteady Potential Flow

This benchmark unsteady test case investigates the lift coefficient responses of a thin airfoil to the pitching and plunging excitations using the Wagners indicial function expressed by Eq. 5. A representative input signal is essential for effective training of the neural network. The qualified signal should carry as much information as possible to cover the range of the testing signals of interest. It is also ideal to have a randomized sequence of different amplitudes, frequencies and phases, to reduce the correlation between the input data points. For this example, the pitching and plunging coupled signal is used to identify the parameters for training. Randomlike time sequences for both motions are created using white Gaussian noise filtered by a low-pass filter with a pass-band frequency of 10 Hz and stop-band frequency of 15 Hz as illustrated in Fig. 6. Using a constant freestream velocity of $U = 33$ m/s and a chord of 1 m, the lift coefficient responses are calculated up to 10 s.



(a) Input



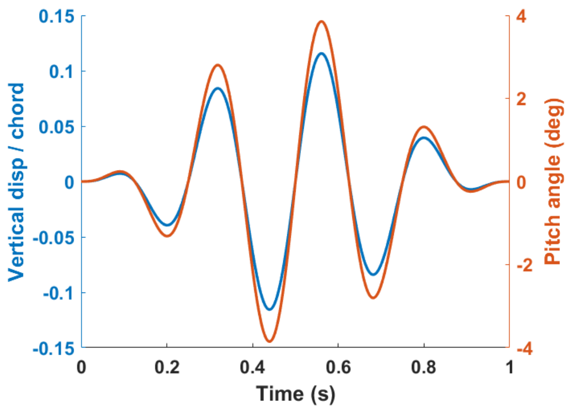
(b) Output

Figure 6: Training signal for the potential flow model.

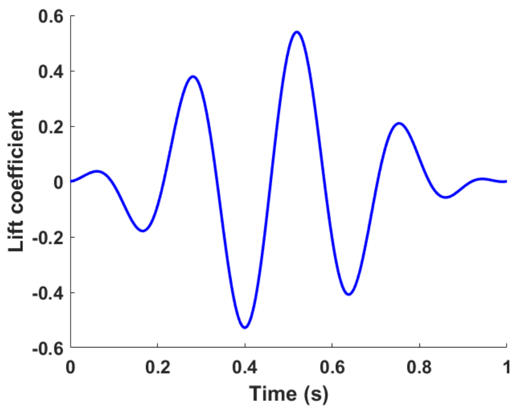
To evaluate the prediction performance of the trained model, a testing excitation of sine waves is generated at a single frequency and modulated by a cosine function as

$$F_{\text{test}}(t) = A \sin(2\pi f t)\,(1 - \cos(2\pi t))\,, \tag{11}$$

where A is a factor used to adjust the amplitudes for pitching and plunging motions accordingly, and f is the designated frequency. The input signal and the lift coefficient response are presented in Fig. 7.



(a) Input



(b) Output

Figure 7: Testing signal for the potential flow model.

This benchmark test case is critical in the investigation of subsequent RNN-based ROM generation for aerodynamic applications. The analytical solution allows us to separate the key elements that affect aerodynamic responses and to identify the influential modeling factors based on physical meanings, instead of the commonly used trial-and-error procedure. The fast solution saves considerable computational cost and expedites the investigation process.

C. Unsteady Turbulent Flow

1. Problem description

Using the same NLR 7301 airfoil geometry and mesh model as in the steady case A, the techniques developed for RNN-based ROMs are tested for increased complexity. High-fidelity unsteady CFD simulations of the pitching and plunging airfoil are performed with incoming flow at a Reynolds number of 1.7×10^6 and Mach number of 0.5. The airfoil undergoes pitching motion with a variation of 0.5° around a mean angle of 0.5°, and plunging motion with a variation of 0.1 chord lengths around a mean vertical displacement of 0. A maximum reduced frequency of 0.3 is enforced.

Based on the lessons learned from the potential flow case, the recurrent neural network is trained with the previously tuned input delays and feedback delays. Three hidden layers with five neurons each are proven to be effective provided that the inputs and delays are properly defined.

2. *Training signal*

Since the analytical mesh motion is used in the high-fidelity CFD simulations, a mildly noisy signal is preferable to ensure stability of the solution. Nonetheless, the training signal should contain variations in both amplitude and frequency. Superimposition of multiple harmonics with low to moderately high frequencies can provide a training signal that meets these requirements. The superimposed harmonic signals shown in Figs. 8 and 9 have the following form:

$$F_{\mathrm{SIH}}(t) = A_{\mathrm{m}} + \sum_{i=1}^{N} A_i \sin(2\pi f_i t + \phi_i)\sin(2\pi f_{\mathrm{m}} t), \tag{12}$$

where A_{m} and f_{m} are the central amplitude and frequency, and N is the number of harmonics in the summation. A_i, f_i, and ϕ_i are the amplitude, frequency and phase for each harmonic, and their values are assigned based on the Latin Hypercube sampling technique. Performing a Fourier transform analysis for the signal reveals the dominant frequencies and corresponding amplitudes. There are $2N$ dominant frequency components distributed symmetrically about the central frequency f_{m}. The reason is that the product of two sine functions can be decomposed into the sum of two cosine functions with new phases being the sum and difference of original phases respectively:

$$\sin(2\pi f_i t + \phi_i)\sin(2\pi f_{\mathrm{m}} t) = \frac{\cos(2\pi(f_i - f_{\mathrm{m}})t + \phi_i) - \cos(2\pi(f_i + f_{\mathrm{m}})t + \phi_i)}{2}. \tag{13}$$

In the following spectral content presentations, the mean value at zero frequency for a specific time series is removed to emphasize the dominant frequencies. The responses of c_l and c_d are presented in Figs. 10 and 11.

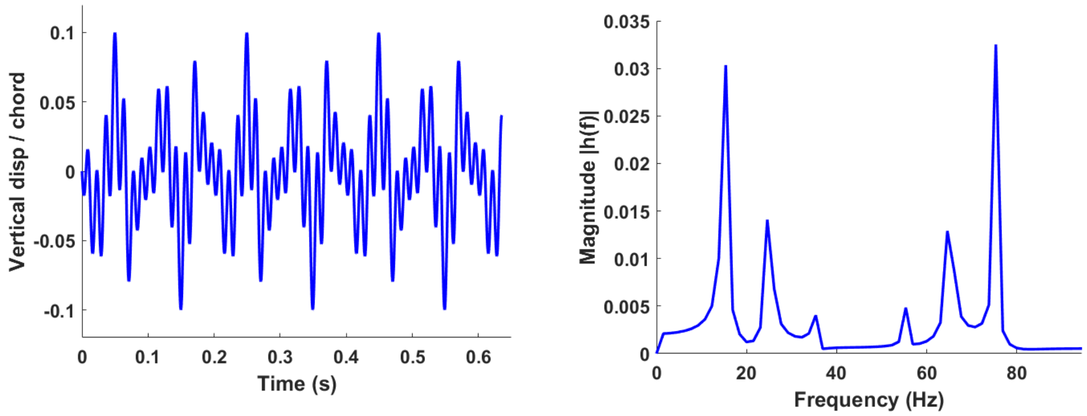


Figure 8: Excitation of plunging motion for training the turbulent flow model.

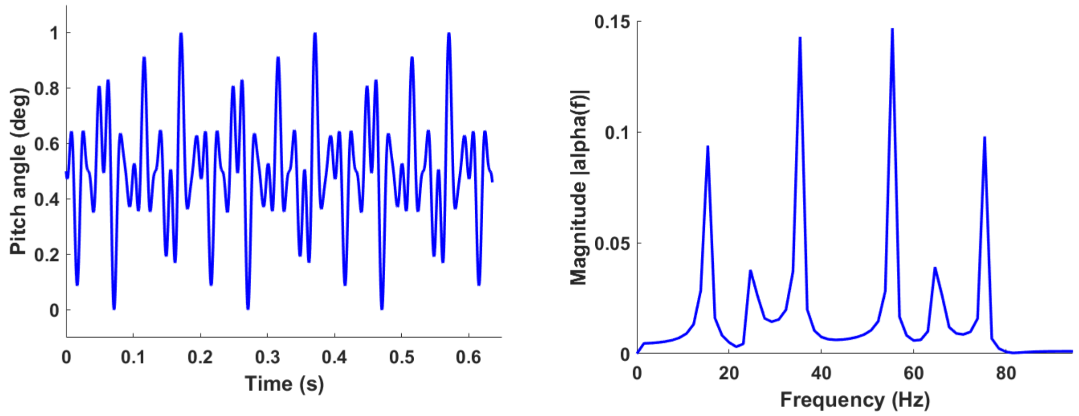


Figure 9: Excitation of pitching motion for training the turbulent flow model.

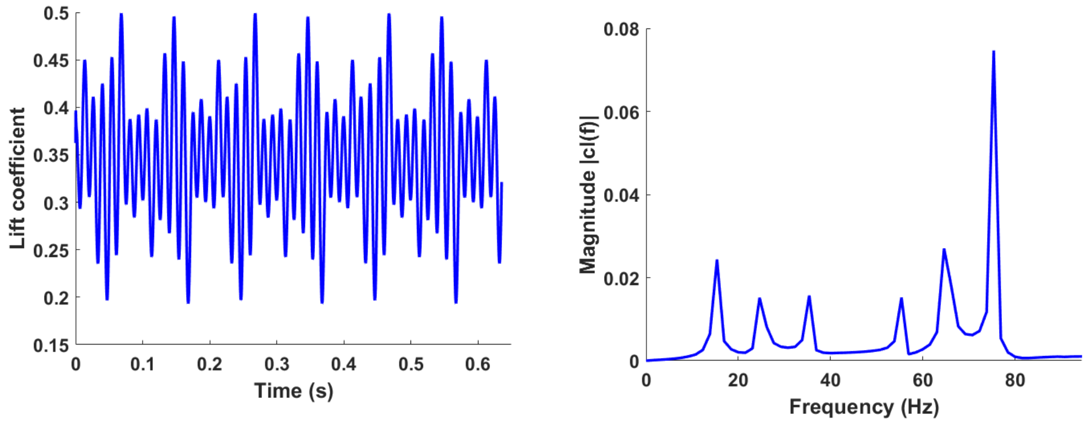


Figure 10: Lift coefficient outputs for training the turbulent flow model.

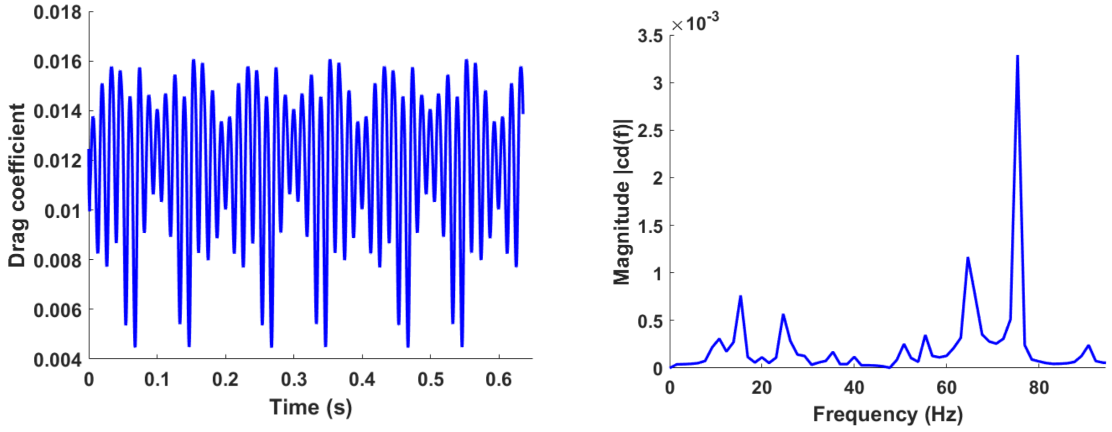


Figure 11: Drag coefficient outputs for training the turbulent flow model.

3. Testing signal

The testing signals used for this case are a sine wave (Figs. 12 and 13) and a Gaussian pulse with nonlinear chirp (Figs. 16 and 17). The mathematical form of the adopted Gaussian pulse is expressed in Eq. 14 as

$$F_{\mathrm{GP}}(t) = A\mathrm{e}^{-t_{\mathrm{s}}^2} \cos\left(50t_{\mathrm{s}} - 8\pi\mathrm{e}^{-2t_{\mathrm{s}}^2}\right), \tag{14}$$

where t_{s} is the scaled time defined by

$$t_{\mathrm{s}} = \frac{t - s_{1/2} - t_0}{s_{1/2}} l_{\mathrm{eff}}. \tag{15}$$

The half length of the total simulation time $s_{1/2}$ and the coefficient l_{eff} are used to center the effective portion of the signal that contains major information. The generated ROM is expected to identify all the different components in the new signal shown in Figs. 14, 15, 18 and 19.

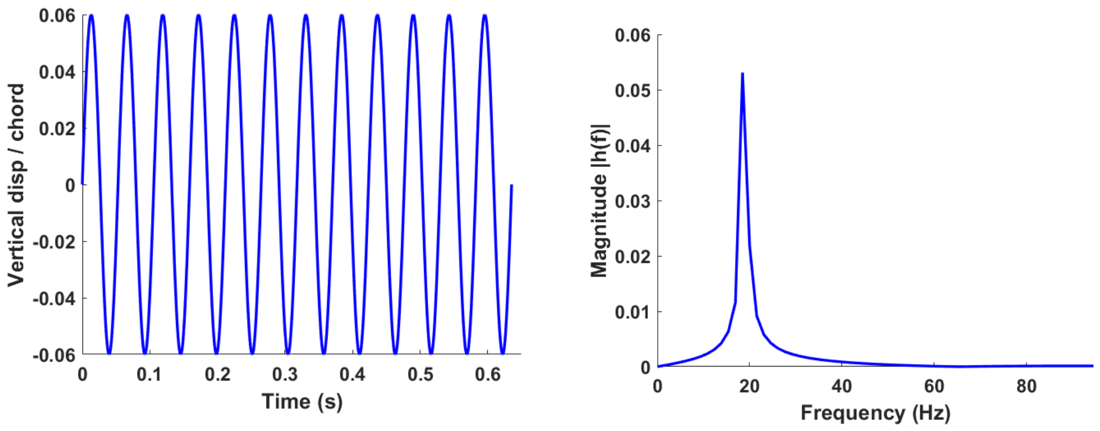


Figure 12: Sine wave excitation of plunging motion for testing the turbulent flow model.

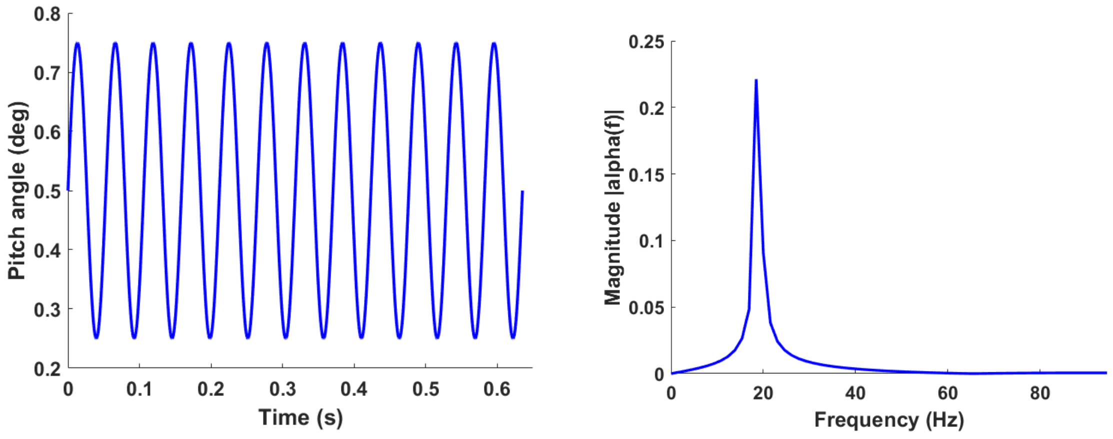


Figure 13: Sine wave excitation of pitching motion for testing the turbulent flow model.

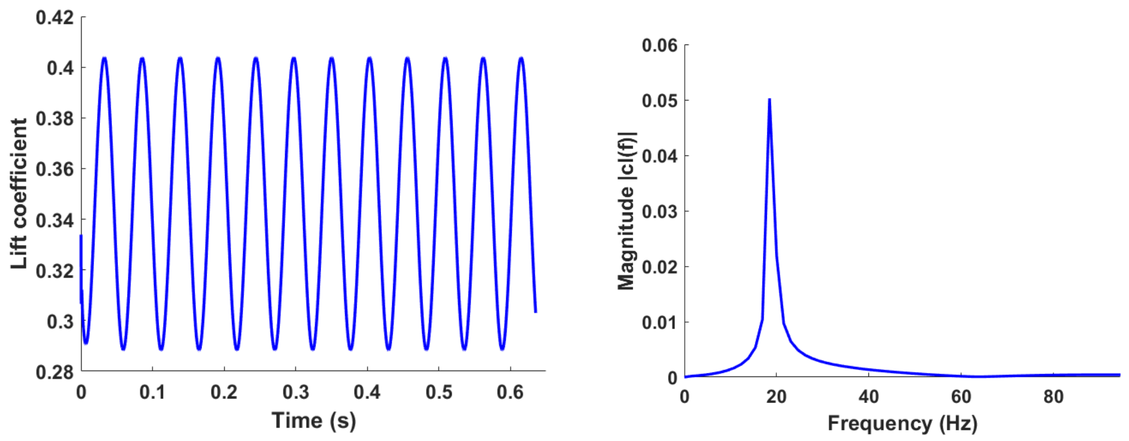


Figure 14: Lift coefficient responses to sine wave excitation for testing the turbulent flow model.

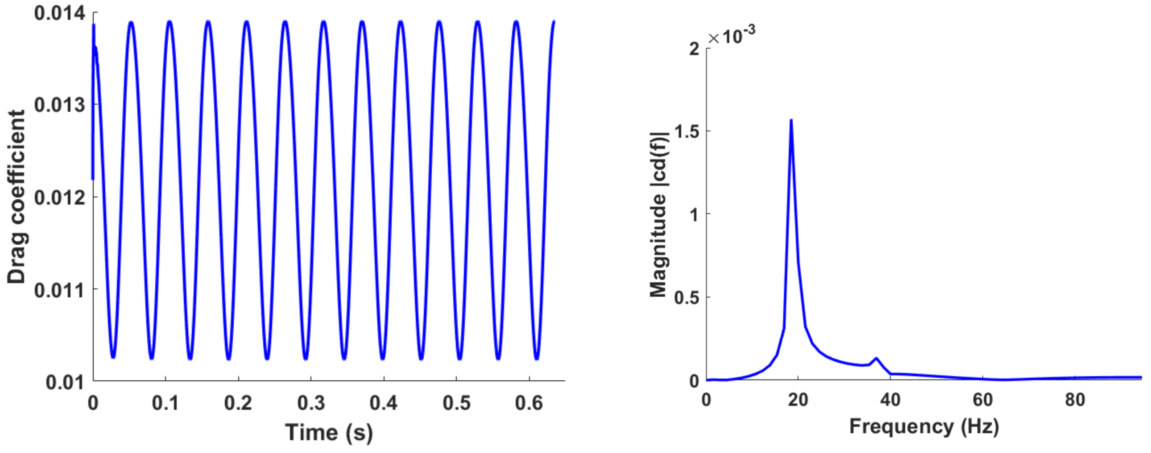


Figure 15: Drag coefficient responses to sine wave excitation for testing the turbulent flow model.

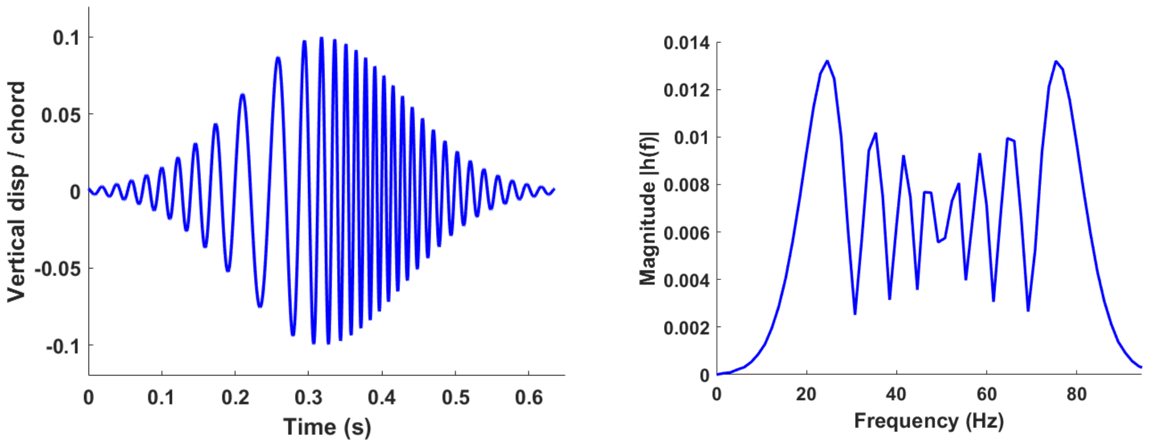


Figure 16: Gaussian excitation of plunging motion for testing the turbulent flow model.

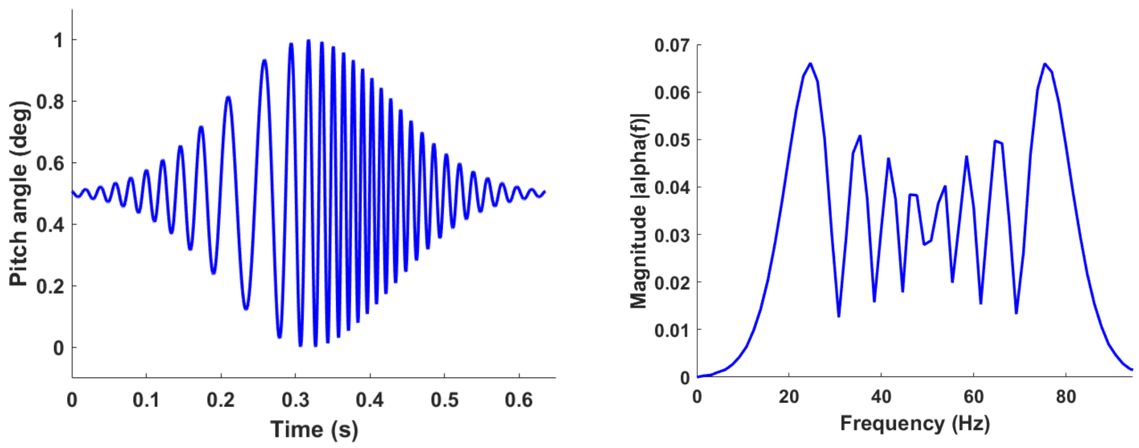


Figure 17: Gaussian excitation of pitching motion for testing the turbulent flow model.

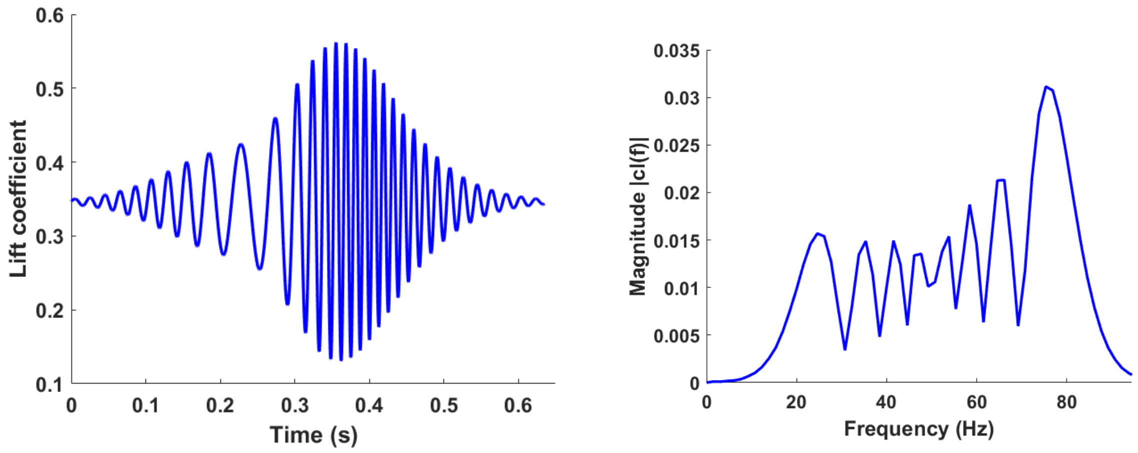


Figure 18: Lift coefficient responses to Gaussian excitation for testing the turbulent flow model.

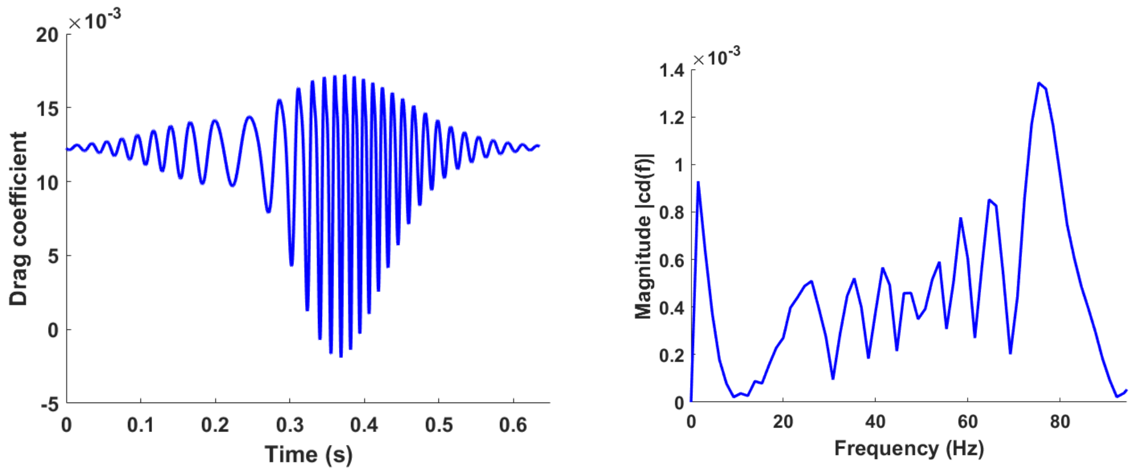


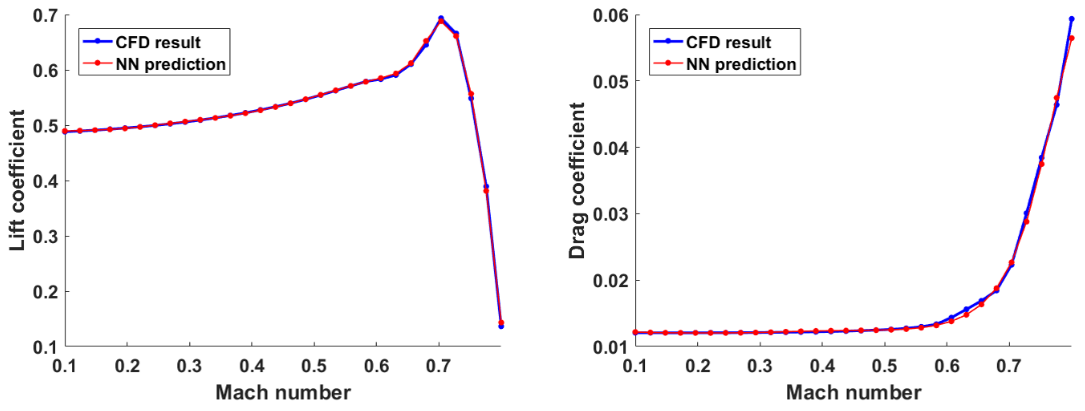
Figure 19: Drag coefficient responses to Gaussian excitation for testing the turbulent flow model.

IV. Results

A. Assessment of Feedforward Neural Network Performance

The most straightforward way of assessing the performance of a neural network is to to look at the fitting accuracy. However, high fitting accuracy does not imply high predictive performance. Therefore, the neural network-based ROM obtained by training on the distributed 100 snapshots according to the defined probability density function is used to predict another set of 30 uniformly distributed snapshots. CFD simulations at the 30 snapshots are conducted to provide the true outputs.

Figure 20 shows the comparison of full-order model and ROM results for the lift and drag coefficients with distributed snapshots used for training. It is evident that the predictions have a good agreement with the true solutions. It should be noted that this single neural network-based ROM has been able to capture the features in different flow regimes and yields a smooth transition in between. The drag coefficient is slightly off for Mach numbers greater than 0.6, when the flow is transonic. With a focus on the transonic flow regime, a separate ROM could be created. Since the purpose of this steady test case is to demonstrate the error estimation techniques for improving the quality of reduced models, such models are not presented in this paper.



(a) Lift (b) Drag

Figure 20: Comparison between ROM and full-order model results for lift and drag coefficients with angle of attack at 2° and Reynolds number of 2.12×10^6.

The magnitudes of the fitted POD modes for pressure coefficient from the same training are displayed in Fig. 21. The outputs of POD modes align with the targets for the lower modes, but exhibit small discrepancies for the higher modes.

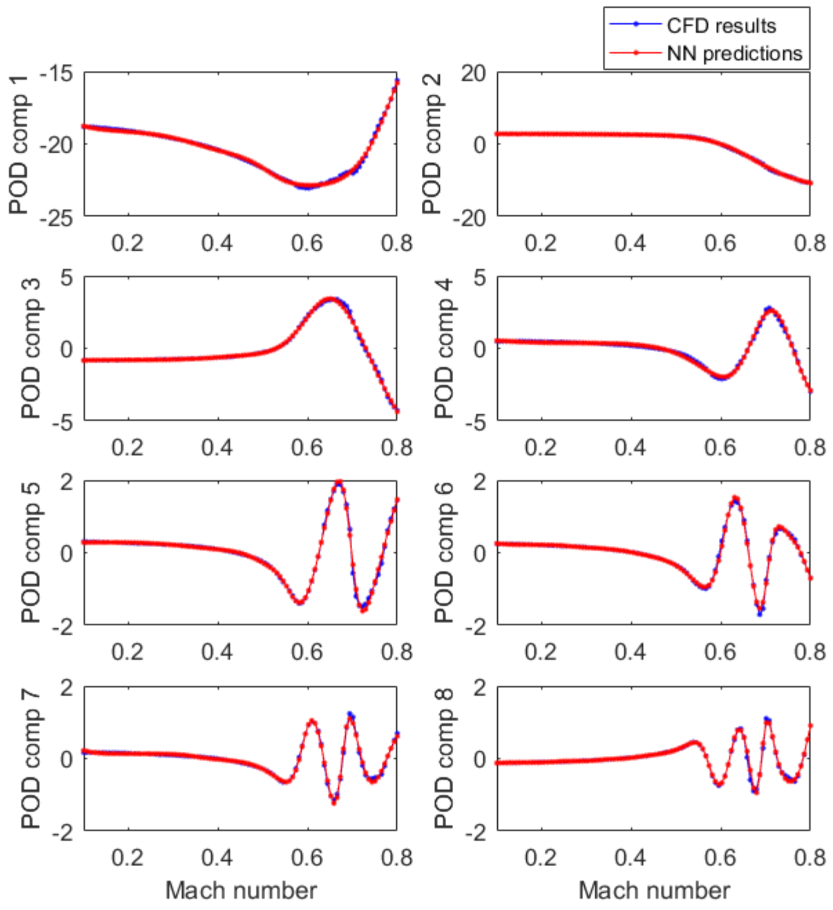
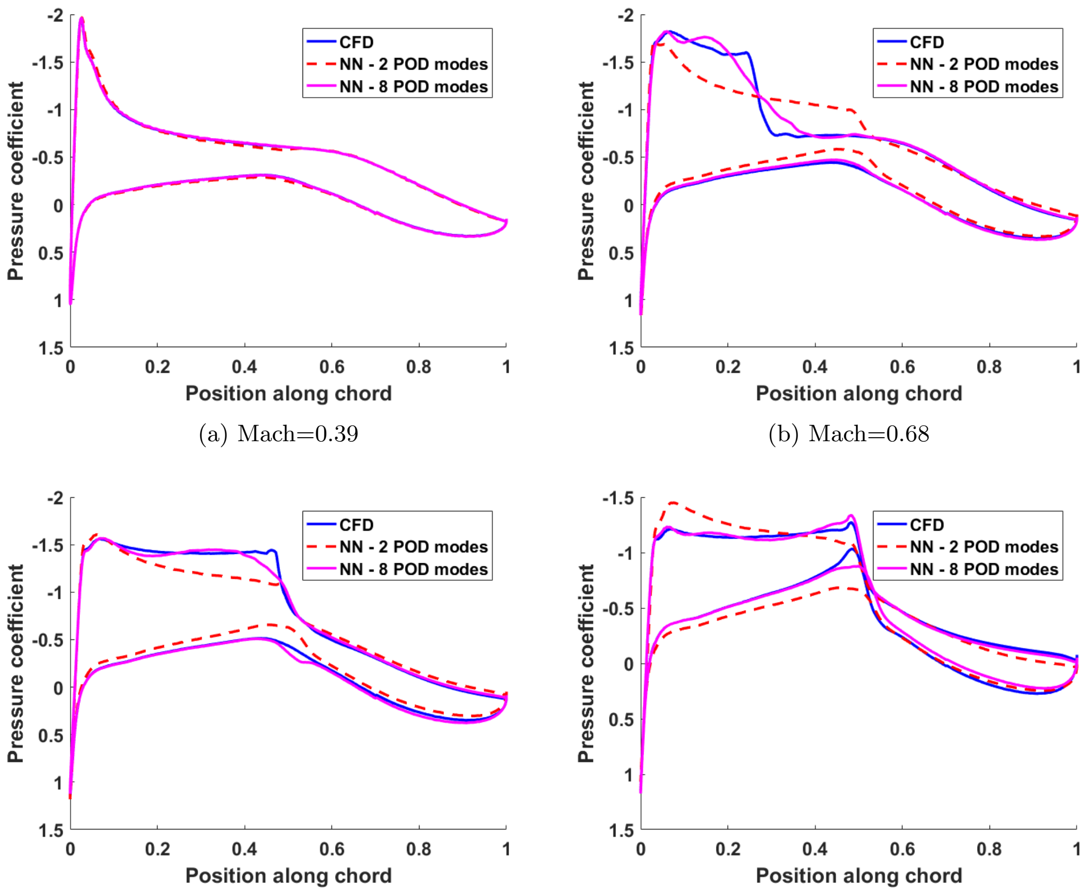


Figure 21: Fitting for the amplitudes of POD modes.

Reconstructing the pressure distribution by employing the POD modes yields Fig. 22, which provides a qualitative evaluation of the prediction performance. Within the subsonic regime, using only two POD modes could provide adequate accuracy for C_p distributions. As the flow evolves to the transonic regime, the predicted C_p distribution with eight POD modes still matches the full-order model results in terms of the shock location. The ROM also captures the trend of the shock wave moving towards the trailing edge as the Mach number increases to 0.8.



(a) Mach=0.39 (b) Mach=0.68

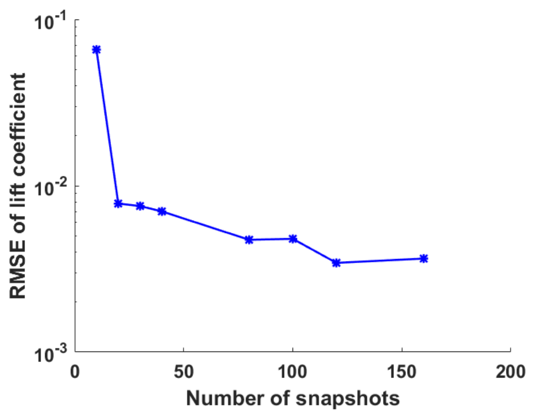
(c) Mach=0.72 (d) Mach=0.77

Figure 22: Prediction of pressure coefficient distribution for subsonic and transonic flow regimes.

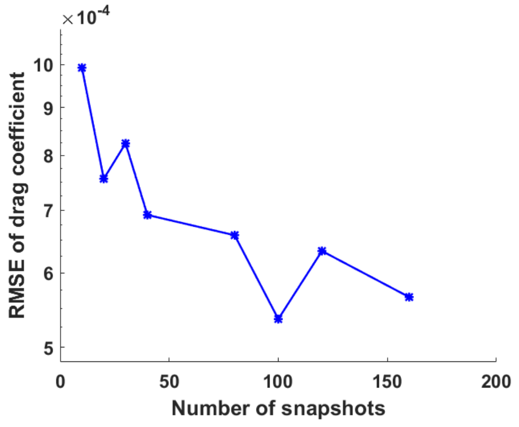
Given the fact that only 40 snapshots between 0.6 and 0.8 are used as the training set, and that there exist sharp gradients over this interval, the accuracy of the outputs can be enhanced by including more snapshots to train the neural network. This leads to convergence studies on the prediction accuracy with variations in the number of snapshots and POD modes used.

1. *Convergence analysis*

A series of samples containing an increasing number of snapshots is used to create different neural network models for the same problem presented above. Note from Fig. 23 that although the root mean square errors (RMSEs) for lift and drag coefficients decrease as the number of snapshots increases in general, there are models with more snapshots yielding larger RMSEs. This non-monotonic behavior usually exists for results obtained from an individual training process that has inherent randomness. Averaging over more training trials can eliminate this particular behavior.



(a) Lift



(b) Drag

Figure 23: Convergence of c_l and c_d predictions with respect to the number of snapshots.

The effects of the number of POD modes have been investigated using the pressure coefficient results with a training set of 120 snapshots. Fig. 24 reveals that the normalized root mean squared error (NRMSE) decreases with more POD modes used to represent the pressure field. The rate of change in NRMSE decreases and asymptotically reaches a value above 0. It should also be noted that the first six POD modes carry the most energy, indicating that a good representation of the pressure field can be achieved by using only the first six POD modes, thereby saving computational time.

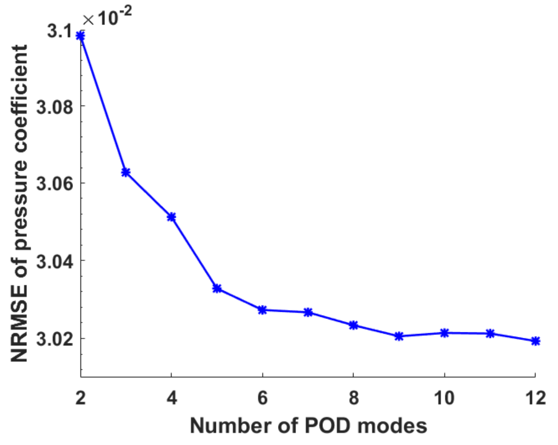
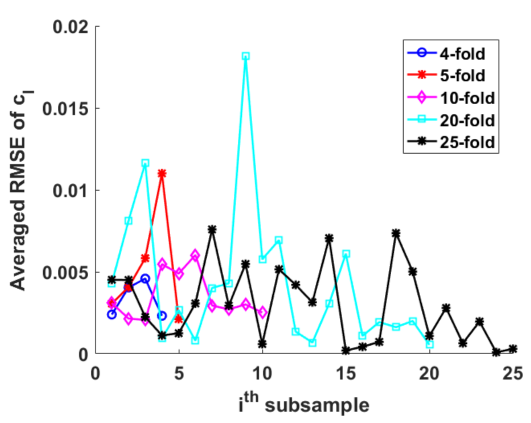


Figure 24: Convergence of predicted pressure coefficient with respect to the number of POD modes.

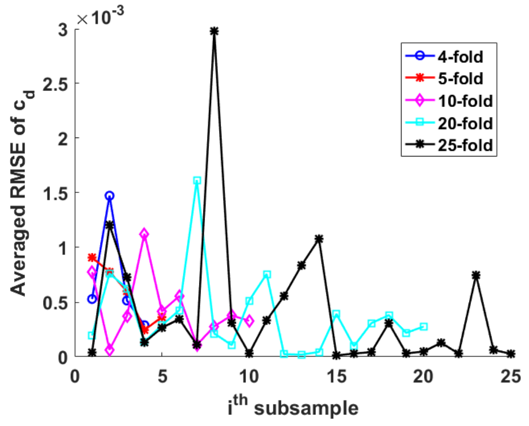
2. *k-fold cross-validation error estimate*

In most cases, the error estimated by the difference between targets and outputs at the training locations is biased, because the predictive model is created by the learning and calibration processes of the high-fidelity snapshots at these locations. Although the true prediction error is unavailable for an unknown independent dataset, the k-fold cross-validation technique will generalize the prediction error within the training set to an unknown set.

The number of folds tested, k, varies from 4 to 25 for the case with 100 snapshots and 8 POD modes. As shown in Fig. 25, the use of more folds leads to a more accurate predictive model based on the RMSE results for lift and drag coefficients. The jumps occurring at a specific k reflect the fact that cross-validation estimates can have large variance. The most important advantage of the k-fold cross-validation is that the prediction error is nearly unbiased. Therefore, an average over all k test sets is promising in estimating the prediction error with small bias and variance as depicted in Fig. 26. The use of 10 subsets in the division of the original training data set is a good compromise between cost and accuracy for estimating the prediction error of lift and drag coefficients in this test case.

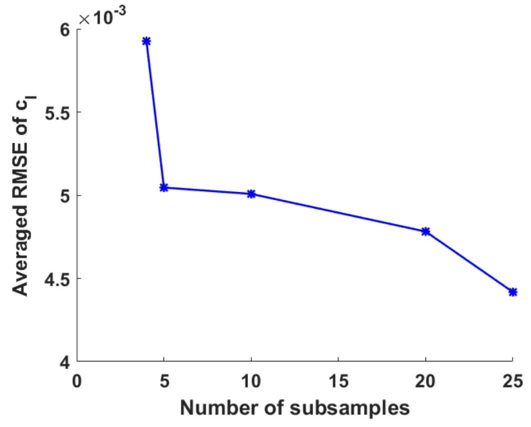


(a) Lift

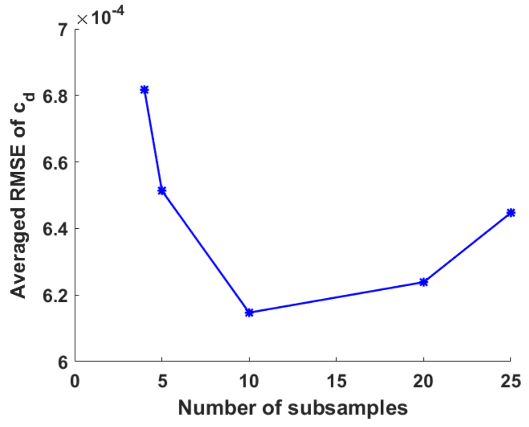


(b) Drag

Figure 25: Prediction errors using cross-validation estimates in each testing subset.



(a) Lift

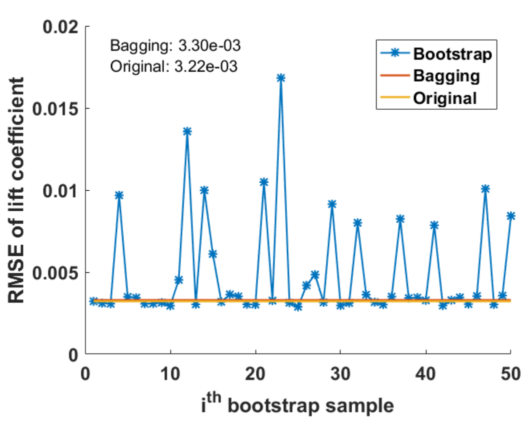


(b) Drag

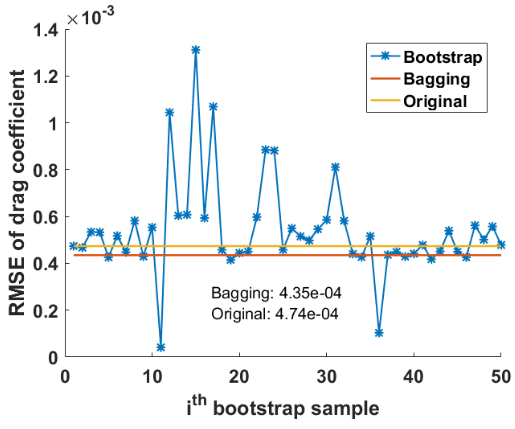
Figure 26: Variation of cross-validation estimates with respect to the number of subsets adopted.

3. Bootstrapping aggregation

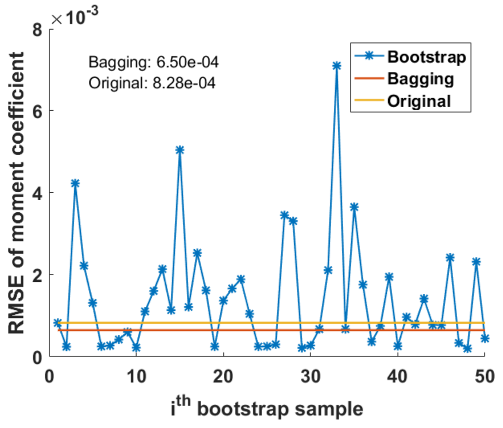
Fifty bootstrap samples are generated from the original samples with 100 snapshots. Each bootstrap sample produces a predictive neural network model. All the models are used to predict the value at a selected data point independent of the original dataset. Taking the average of all the predicted values at this specific data point yields a bagged prediction. Figure 27 compares the bagged solution with the original predictive model for lift, drag and moment coefficients. The plots of drag and moment coefficients both show improved accuracy with the use of bootstrapping aggregation. The bagged solution for lift coefficient does not show an improvement. It is possible that more bootstrap samples are necessary to exclude potential outliers and enhance lift coefficient prediction, but there is a trade-off between the benefits of a slightly better solution and the drawback of cost with more training processes involved.



(a) Lift



(b) Drag

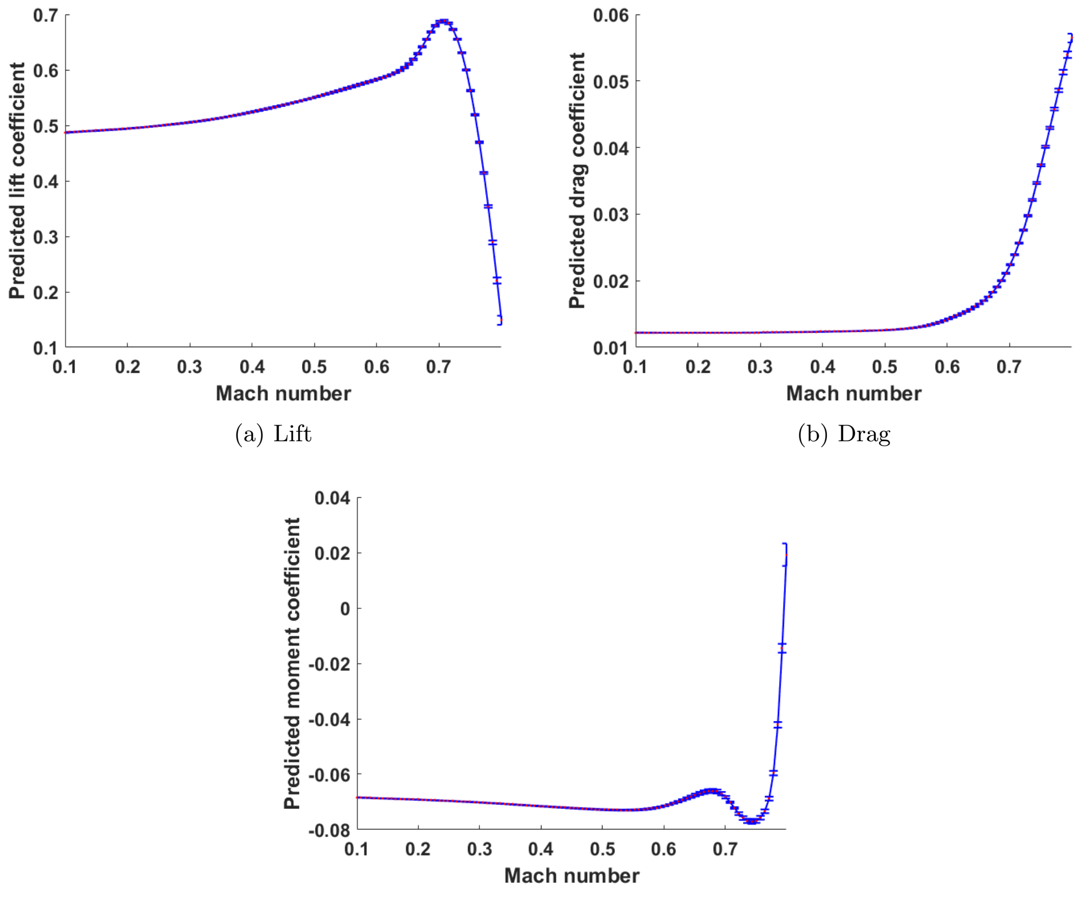


(c) Moment

Figure 27: Improvement of prediction using bootstrapping aggregation.

4. Bootstrapping error estimates

The bagged solution is statistically considered as the less biased approximation of the true regression.[40] The variance of the predictions obtained from the individual bootstrapping model can then be utilized to estimate the variance arising from the randomization of initial weights and data devision in the neural network training phase. Figure 28 provides an intuitive assessment of the constructed confidence intervals. It is noted that as Mach number increases, the width of the confidence interval increases, indicating that denser snapshot sets are needed in this region to yield more reliable predictions.



(a) Lift

(b) Drag

(c) Moment

Figure 28: Confidence intervals constructed by the bootstrapping aggregation.

B. Investigation of Meaningful Inputs for a Thin-Airfoil Problem

Although a neural network is an output-based model reduction method, there are more effective ways of creating a recurrent neural network using selective inputs based on physics of the studied system. While this technique will not be as efficient as a linear identification method for this case, it indicates how to improve the set of inputs in the context of a NARX model.

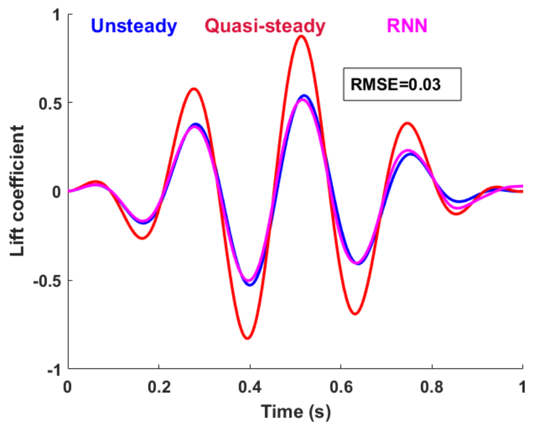
This subsection shows the progressive increment of quality for the predictions of the ROM model when the meaningful inputs are selected and when the added mass is considered separately from the circulatory lift. In addition, the possible advantages and disadvantages of using a feedback from the output are discussed.

Initially, the neural networks used for unsteady predictions do not consider feedback from the outputs. Only the history of inputs is taken into account. From the Wagner's indicial lift (Eq. 5), it is clear that a history of inputs is sufficient for accurate computation of loads.

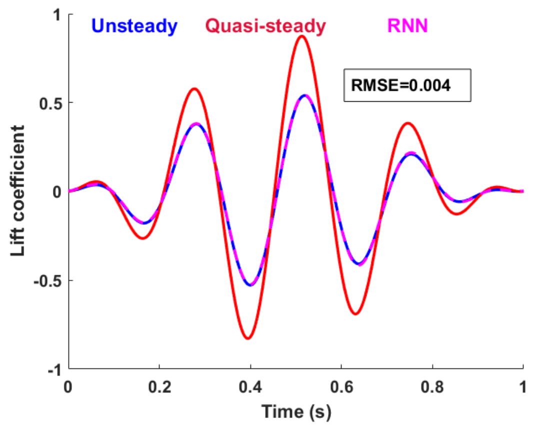
For all the following comparisons, the test signal is a mixed pitch and plunge motion of 4 Hz in phase, corresponding to a reduced frequency of 0.38. The inputs are the first and second derivatives of the vertical position, as well as the angle of attack and its derivatives up to the second order.

1. Effects of time history selection

If 6 points in the time history are taken uniformly from the first previous time step until the non-dimensional time of $s = 50$, for example, the predictions have an RMSE of 0.03. However, if the points are distributed only until the non-dimensional time of 15, trying to capture derivative of the Wagner's function, then the response is improved, and the new RMSE is 0.004, as shown in Fig. 29.



(a) Uniform input times until $s = 60$



(b) Selected input times until $s = 15$

Figure 29: Comparison between uniform inputs and selected inputs.

2. *Effect of feedback on the RNN prediction*

The NARX models can also use a feedback on the outputs to estimate a prediction for the next step. Using this feedback is not strictly needed, as seen above, but can improve the predictions. In fact, adding the feedback reduces the time history length requirement. In this example, the number of previous time inputs is reduced by one and a feedback on the last calculated lift coefficient is included as input for the RNN.

The analytical unsteady response and the predicted values from the RNN are shown in Fig. 30. Comparing to the previous case that considered only the history of past inputs, the quality of the prediction decreases. However, this approach presents another option for a more accurate solution, with the advantage of employing a shorter time history to feed the RNN.

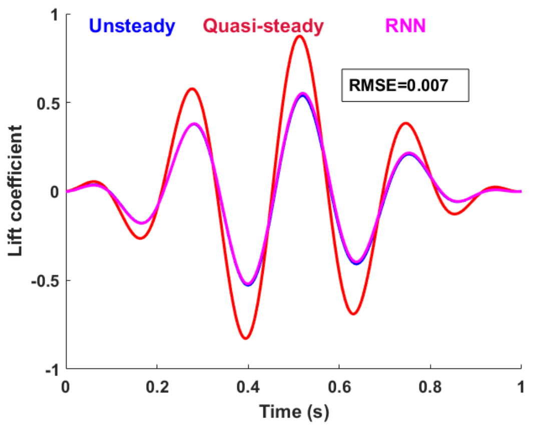


Figure 30: RNN prediction considering a feedback of the lift coefficient.

3. *Modeling the circulatory lift only*

One of the points to consider is that different phenomena are modeled with the same ROM. While the added mass contribution to the lift only depends on the instantaneous inputs, the circulatory lift depends on the history of motion. It is interesting to see the impact of adding the non-circulatory lift separately, using the analytical formulation, while the circulatory part is computed with the RNN model.

In this model, the lift coefficient feedback is preserved. The inputs are the same of the previous case. The difference is that the training of the RNN is performed with the circulatory portion of the lift only. The added mass is included with the analytical formula.

For this case, modeling only the circulatory portion of the lift with a neural network results in a good agreement between the predicted values and the analytical unsteady solution, as shown Fig. 31.

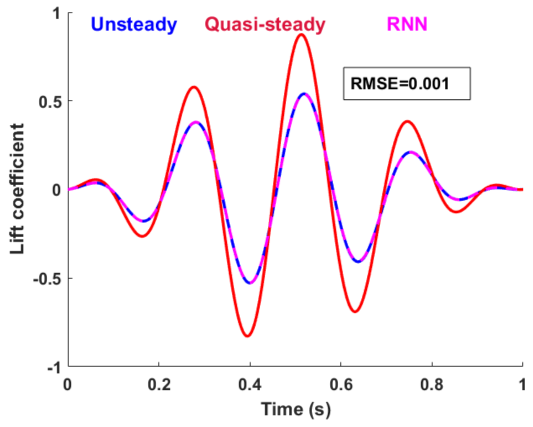


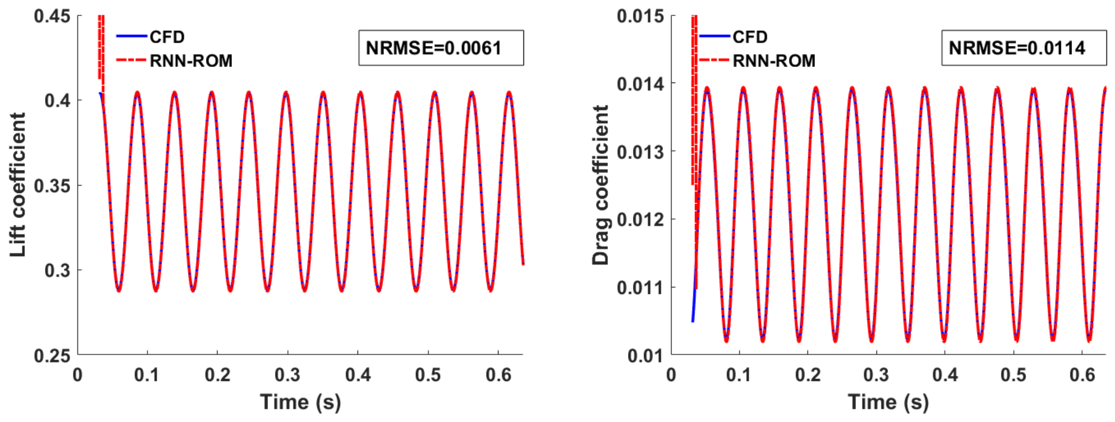
Figure 31: RNN prediction approximating only the circulatory contribution to the lift coefficient.

The observation that removing the added mass from the prediction of the neural network may result in a better prediction opens the possibility of identifying the non-circulatory lift separately from the circulatory one. This is a topic for future studies.

C. Assessment of Recurrent Neural Network-based ROM

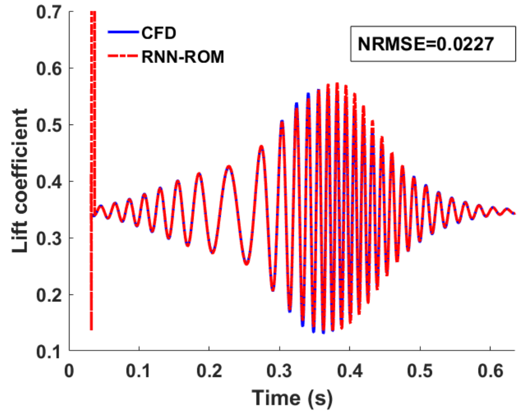
1. *Predicted results*

The predicted responses of lift and drag coefficients to the sine and Gaussian excitation signals are shown in Figs. 32 and 33. The transient effects at the beginning of the prediction are due to the lack of "memory" to regress the outputs. The normalized root-mean-square errors are calculated without the initial oscillations. The predicted outputs align well with the full-order model results obtained from xflow.

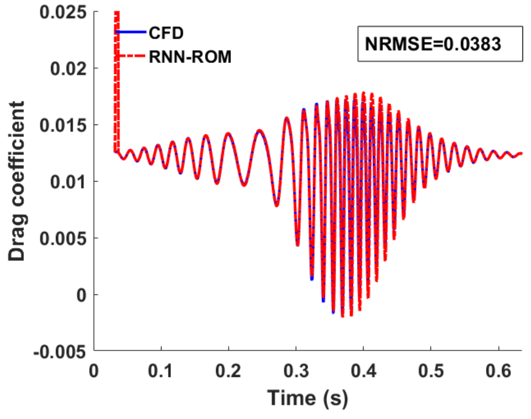


(a) Lift (b) Drag

Figure 32: Predicted responses to sine excitation for the turbulent flow model.



(a) Lift

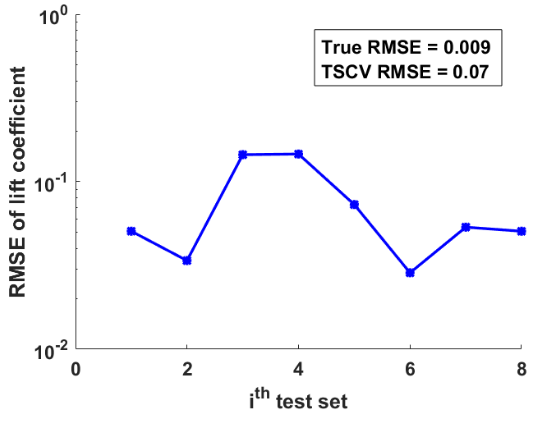


(b) Drag

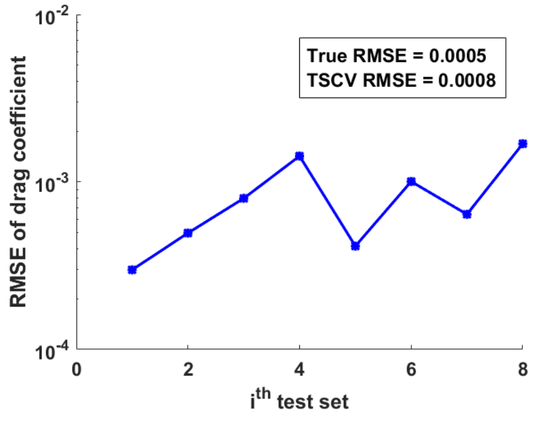
Figure 33: Predicted responses to Gaussian excitation for the turbulent flow model.

2. Time-series cross-validation error estimate

Ten equally-sized subsets in the original time sequence are created without data overlapping. The block cross-validation is performed 20 times to reduce the influence of randomness in the training process. The prediction error varies across the testing subsets as seen in Fig. 34. Some high prediction errors result from extrapolation when the cross-validation training set cannot fully represent the features of the testing set. This error can be reduced by employing a randomlike training signal. That happens because for a randomlike signal, the subsets are more likely to cover the full spectrum. The overall root-mean-square error estimated by time-series cross-validation is one order of magnitude larger than the true prediction error for lift coefficient, which is relatively conservative for the case investigated.



(a) Lift



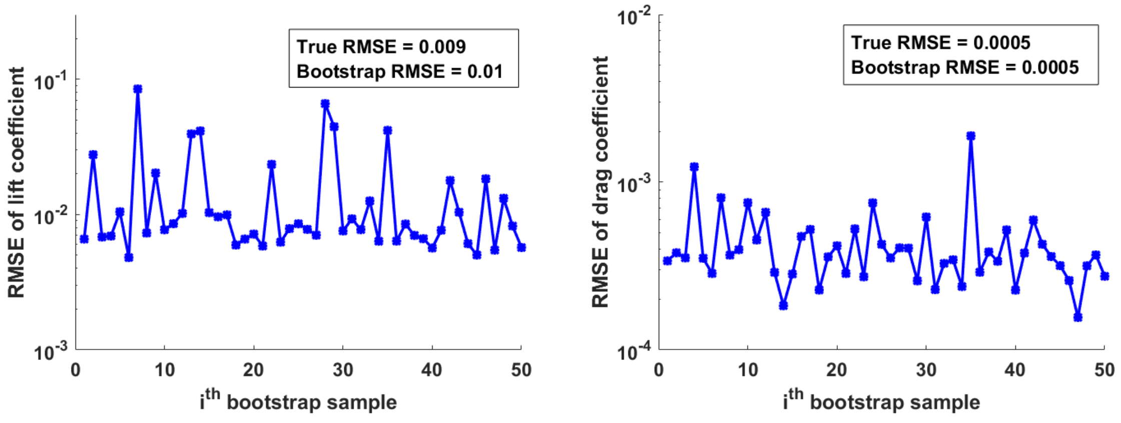
(b) Drag

Figure 34: Time-series cross-validation error estimates for lift and drag coefficients.

3. Block bootstrap error estimate

For neural network applications, the bootstrapping technique often yields the most reliable error estimates. This subsection examines the accuracy of the block bootstrap method in the context of time series prediction.

Fifty time sequences are constructed by resampling the original signal with ten equally-sized blocks. These blocks are connected with each other by their end time points. Fifty RNN-based ROMs are generated by training the fifty time sequences separately. Figure 35 shows the error produced by using each of the fifty ROMs. The bootstrap estimate is indicated as the overall root-mean-square error by averaging all the fifty errors calculated previously. By comparing with the true prediction error for signals of Gaussian pulse type, it can be noted that the block bootstrap error estimate closely presents the true prediction error. Although it is time consuming to create fifty reduced-order models, once the error estimate system is established, the prediction error can be computed for different testing signals. Therefore, the computational cost should not hinder the application of bootstrapping techniques to neural network models.



(a) Lift (b) Drag

Figure 35: Block bootstrap error estimates for lift and drag coefficients.

V. Concluding Remarks

This paper addresses the development and demonstration of a neural network-based reduced-order modeling technique with error estimation to effectively capture nonlinear steady and unsteady high-fidelity aerodynamic responses from CFD solutions. In the steady case, the feedforward neural network-based ROM accurately captures the flow behavior around the NLR 7301 airfoil with variations in Mach number covering both subsonic and transonic flow regimes. To assess predictive performance, the k-fold cross-validation error estimate has been used to approximate the true prediction error. Despite the variance of the prediction errors in subsamples, the averaged root-mean-square error still provides an estimated prediction error in the same order of magnitude as the true prediction error, which is calculated using the CFD solution for the independent testing dataset. The bootstrapping technique has also been applied for error estimation. The bagging results present the potential to improve the prediction accuracy compared with the original reduced-order model. In addition, the bagging results for different outputs have been used to compute the standard deviation associated with the errors in model misspecification and to construct confidence intervals for the bootstrap-approximated regression.

For the unsteady study, a potential flow model has been used to establish a systematic way of selecting the effective training inputs to generate and tune the RNN-based ROM. To the best of the authors' knowledge, this is the first time that a physics-based technique has been put forward and applied to the selection of meaningful inputs for training of neural network in the framework of ROM generation for aerodynamic systems. A non-dimensional time corresponding to the point when the derivative of the Wagner's function decreases to 5% of the initial value indicates the effective length of the input delay states for training. The influential factors identified from the potential flow test case are shown to have a significant impact on the performance of the predictive models for the NLR 7301 CFD model with forced pitching and plunging motions. With properly defined inputs and modeling parameters, the RNN-based ROM can predict aerodynamic responses to an arbitrary input signal to a desired accuracy level. The accuracy level is quantified by the time-series cross-validation and non-overlapping block bootstrap techniques. The results show that both techniques can provide good metrics for assessing the prediction performance. Although the bootstrap techniques require a large number of subsamples to be trained, once the error estimation system is built, the prediction error will be convenient to obtain for any unknown input signals. The combination of neural network-based ROMs for aerodynamic applications and error estimation techniques developed for evaluating neural network models constitutes another contribution of the present work.

In summary, this study developed a physics-based method to tune the RNN-based reduced-order model. Implementation of this method for aerodynamic simulations of the NLR 7301 airfoil demonstrated its advantage in improving the performance of the reduced-order model. Future work will extend this technique and statistical error analyses to more complex aerodynamic systems. For those, robustness of the cross-validation and bootstrap estimates will be further investigated.

Acknowledgements

This material is based upon work supported by Airbus in the frame of the Airbus-Michigan Center for Aero-Servo-Elasticity of Very Flexible Aircraft (CASE-VFA).

References

[1]Silva, W. A., "Recent Enhancements to the Development of CFD-Based Aeroelastic Reduced-Order Models," *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, American Institute of Aeronautics and Astronautics, Honolulu, Hawaii, 2007.

[2]Silva, W. A., "AEROM: NASA's Unsteady Aerodynamic and Aeroelastic Reduced-Order Modeling Software," *International Forum on Aeroelasticity and Structural Dynamics*, Como, Italy, 25-28 June 2017.

[3]Raveh, D. E., "Identification of Computational-Fluid-Dynamics Based Unsteady Aerodynamic Models for Aeroelastic Analysis," *Journal of Aircraft*, Vol. 41, No. 3, May 2004, pp. 620–632.

[4]Silva, W. A., "Reduced-order Models Based on Linear and Nonlinear Aerodynamic Impulse Responses," *40th Structures, Structural Dynamics, and Materials Conference and Exhibit*, American Institute of Aeronautics and Astronautics, St. Louis,MO,U.S.A., 12-15 April 1999.

[5]Raveh, D. E., "Reduced-Order Models for Nonlinear Unsteady Aerodynamics," *AIAA Journal*, Vol. 39, No. 8, August 2001, pp. 1417–1429.

[6]Skujins, T. and Cesnik, C. E. S., "Reduced-Order Modeling of Unsteady Aerodynamics Across Multiple Mach Regimes," *Journal of Aircraft*, Vol. 51, No. 6, September 2014, pp. 1681–1704.

[7]Winter, M. and Breitsamter, C., "Application of Unsteady Aerodynamic Reduced-Order Modeling Techniques to a Complex Configuration," *International Forum on Aeroelasticity and Structural Dynamics*, Como, Italy, 25-28 June 2017.

[8]Winter, M., Heckmeier, M., and Breitsamter, C., "CFD-Based Aeroelastic Reduced-Order Modeling Robust to Structural Parameter Variations," *Aerospace Science and Technology*, Vol. 67, 2017, pp. 13–30.

[9]Winter, M. and Breitsamter, C., "Neurofuzzy-Model-Based Unsteady Aerodynamic Computations Across Varying Freestream Conditions," *AIAA Journal*, Vol. 54, No. 9, June 2016, pp. 2705–2720.

[10]Winter, M. and Breitsamter, C., "Efficient Unsteady Aerodynamic Loads Prediction Based on Nonlinear System Identification and Proper Orthogonal Decomposition," *Journal of Fluids and Structures*, Vol. 67, No. Supplement C, 2016, pp. 1–21.

[11]Winter, M. and Breitsamter, C., "Unsteady Aerodynamic Modeling Using Neuro-Fuzzy Approaches Combined with POD," *64th Deutscher Luft- und Raumfahrtkongress*, Rostock, Germany, September 2015.

[12]Mannarino, A. and Mantegazza, P., "Nonlinear Aeroelastic Reduced Order Modeling by Recurrent Neural Networks," *Journal of Fluids and Structures*, Vol. 48, No. Supplement C, 2014, pp. 103–121.

[13]Han, Z.-H. and Görtz, S., "Hierarchical Kriging Model for Variable-Fidelity Surrogate Modeling," *AIAA Journal*, Vol. 50, No. 9, September 2012, pp. 1885–1896.

[14]Han, Z., Zimmerman, R., and Görtz, S., "Alternative Cokriging Method for Variable-Fidelity Surrogate Modeling," *AIAA Journal*, Vol. 50, No. 5, May 2012, pp. 1205–1210.

[15]Liu, L., Padthe, A. K., Friedmann, P. P., Quon, E., and Smith, M. J., "Unsteady Aerodynamics of an Airfoil/Flap Combination on a Helicopter Rotor Using Computational Fluid Dynamics and Approximate Methods," *Journal of the American Helicopter Society*, Vol. 56, No. 3, 2011, pp. 1–13.

[16]Liu, L., Padthe, A. K., and Friedmann, P. P., "Computational Study of Microflaps with Application to Vibration Reduction in Helicopter Rotors," *AIAA Journal*, Vol. 49, No. 7, July 2011, pp. 1450–1465.

[17]Bishop, C. M., *Neural Networks for Pattern Recognition*, Oxford University Press, Inc., New York, NY, USA, 1995.

[18]Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F., "Comprehensive Review of Neural Network-Based Prediction Intervals and New Advances," *Transactions on Neural Networks*, Vol. 22, No. 9, September 2011, pp. 1341–1356.

[19]Stone, M., "Cross-Validatory Choice and Assessment of Statistical Predictions," *Journal of the Royal Statistical Society, Ser. B*, Vol. 36, 1974, pp. 111–147.

[20]Bengio, Y. and Grandvalet, Y., "No Unbiased Estimator of the Variance of K-Fold Cross-Validation," *Journal of Machine Learning Research*, Vol. 5, 2003, pp. 1089–1105.

[21]Arlot, S. and Celisse, A., "A Survey of Cross-validation Procedures for Model Selection," *Statistics Surveys*, Vol. 4, 2010, pp. 40–79.

[22]Efron, B., "Bootstrap Methods: Another Look at the Jackknife," *The Annals of Statistics*, Vol. 7, No. 1, 1979, pp. 1–26.

[23]Jain, A. K., Dubes, R. C., and Chen, C. C., "Bootstrap Techniques for Error Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 5, September 1987, pp. 628–633.

[24]Tibshirani, R., "A Comparison of Some Error Estimates for Neural Network Models," *Neural Computation*, Vol. 8, No. 1, January 1996, pp. 152–163.

[25]Zhang, J., "Developing Robust Non-linear Models Through Bootstrap Aggregated Neural Networks," *Neurocomputing*, Vol. 25, No. 1, 1999, pp. 93 – 113.

[26]Efron, B., "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation," *Journal of the American Statistical Association*, Vol. 78, No. 382, 1983, pp. 316–331.

[27]Tashman, L. J., "Out-of-Sample Tests of Forecasting Accuracy: An Analysis and Review," *International Journal of Forecasting*, Vol. 16, No. 4, 2000, pp. 437–450.

[28]Bergmeir, C. and Bentez, J. M., "On the Use of Cross-Validation for Time Series Predictor Evaluation," *Information Sciences*, Vol. 191, 2012, pp. 192 – 213.

[29]Bergmeir, C., Hyndman, R. J., and Koo, B., "A Note on the Validity of Cross-Validation for Evaluating Autoregressive Time Series Prediction," *Computational Statistics & Data Analysis*, Vol. 120, 2018, pp. 70 – 83.

[30]Härdle, W., Horowitz, J., and Kreiss, J.-P., "Bootstrap Methods for Time Series," *International Statistical Review / Revue Internationale de Statistique*, Vol. 71, No. 2, 2003, pp. 435–459.

[31]Politis, D. N., "The Impact of Bootstrap Methods on Time Series Analysis," *Statistical Science*, Vol. 18, No. 2, May 2003, pp. 219–230.

[32]Zio, E., "A Study of the Bootstrap Method for Estimating the Accuracy of Artificial Neural Networks in Predicting Nuclear Transient Processes," *IEEE Transactions on Nuclear Science*, Vol. 53, No. 3, June 2006, pp. 1460–1478.

[33]Varma, S. and Simon, R., "Bias in Error Estimation When Using Cross-Validation for Model Selection," *BMC Bioinformatics*, Vol. 7, No. 1, February 2006, pp. 91.

[34]Cuevas, A. and Romo, J., "On Robustness Properties of Bootstrap Approximations," *Journal of Statistical Planning and Inference*, Vol. 37, No. 2, 1993, pp. 181 – 191.

[35]Salibián-Barrera, M., Van Aelst, S., and Willems, G., "Fast and Robust Bootstrap," *Statistical Methods and Applications*, Vol. 17, No. 1, February 2008, pp. 41–71.

[36]Venkatesan, C. and Friedmann, P. P., "New Approach to Finite-State Modeling of Unsteady Aerodynamics," *AIAA Journal*, Vol. 24, No. 12, December 1986, pp. 1889–1897.

[37]Racine, J., "Consistent Cross-Validatory Model-Selection for Dependent Data: hv-block Cross-Validation," *Journal of Econometrics*, Vol. 99, No. 1, 2000, pp. 39 – 61.

[38]Fidkowski, K. J. and Roe, P. L., "An Entropy Adjoint Approach to Mesh Refinement," *SIAM Journal on Scientific Computing*, Vol. 32, No. 3, 2010, pp. 1261–1287.

[39]Fidkowski, K. J. and Luo, Y., "Output-based Space-Time Mesh Adaptation for the Compressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 230, No. 14, 2011, pp. 5753–5773.

[40]Heskes, T., "Practical Confidence and Prediction Intervals," *Advances in Neural Information Processing Systems 9*, edited by M. C. Mozer, M. I. Jordan, and T. Petsche, MIT Press, 1997, pp. 176–182.