Physics of Fluids

Fast flow field prediction over airfoils using deep learning approach

Cite as: Phys. Fluids **31**, 057103 (2019); https://doi.org/10.1063/1.5094943
Submitted: 07 March 2019 . Accepted: 22 April 2019 . Published Online: 13 May 2019

Vinothkumar Sekar, Qinghua Jiang (姜清华), Chang Shu (舒昌), and Boo Cheong Khoo

Physics of Fluids ARTICLE scitation.org/journal/phf

Fast flow field prediction over airfoils using deep learning approach

Cite as: Phys. Fluids 31, 057103 (2019); doi: 10.1063/1.5094943
Submitted: 7 March 2019 • Accepted: 22 April 2019 •
Published Online: 13 May 2019

Vinothkumar Sekar, Qinghua Jiang (姜清华), Chang Shu (舒昌),[a] and Boo Cheong Khoo

AFFILIATIONS

Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117575

[a]Author to whom correspondence should be addressed: mpeshuc@nus.edu.sg

ABSTRACT

In this paper, a data driven approach is presented for the prediction of incompressible laminar steady flow field over airfoils based on the combination of deep Convolutional Neural Network (CNN) and deep Multilayer Perceptron (MLP). The flow field over an airfoil depends on the airfoil geometry, Reynolds number, and angle of attack. In conventional approaches, Navier-Stokes (NS) equations are solved on a computational mesh with corresponding boundary conditions to obtain the flow solutions, which is a time consuming task. In the present approach, the flow field over an airfoil is approximated as a function of airfoil geometry, Reynolds number, and angle of attack using deep neural networks without solving the NS equations. The present approach consists of two steps. First, CNN is employed to extract the geometrical parameters from airfoil shapes. Then, the extracted geometrical parameters along with Reynolds number and angle of attack are fed as input to the MLP network to obtain an approximate model to predict the flow field. The required database for the network training is generated using the OpenFOAM solver by solving NS equations. Once the training is done, the flow field around an airfoil can be obtained in seconds. From the prediction results, it is evident that the approach is efficient and accurate.

I. INTRODUCTION

In the field of aerodynamics, obtaining flow field around an airfoil is an important aspect in order to obtain aerodynamic coefficients due to pressure and skin friction, to study flow separation, transition, wake formation, and vortex interaction. These coefficients and flow features are crucial for designing aerodynamic components such as an aircraft wing, wind turbine blade, and helicopter rotor blade. Conventionally, the flow field over an airfoil is obtained by solving Navier-Stokes (NS) equations on a computational mesh with appropriate boundary conditions, known as computational fluid dynamics (CFD) techniques. With the rise of high-performance computing and development of efficient numerical methods, the computational time required for a CFD simulation has been reduced to a large extent. Therefore, at present, CFD simulations are often used as computational experiments to perform aerodynamic design and analysis. However, it is still relatively time consuming in, for example, airfoil optimization and fluid-structure interaction, where there is a requirement of large iterations of flow solutions. In such situations, employing a NS solver is computationally expensive, and it delays the overall design cycle. Hence, there is a need for developing an efficient and accurate method, which can obtain the flow solutions much faster than the conventional CFD approach.

So far, with the development of computing techniques, researchers have extensively utilized CFD techniques for the aerodynamic design and analysis. Therefore, there is a dramatic increase in the data collection of fluid dynamics simulations. Hence, there are needs for techniques and capabilities to process and utilize the CFD data. On the other hand, in the recent years, there has been rapid development in the field of data science, which resulted in Machine Learning (ML) techniques for Big Data. The rise of efficient deep learning tools resulted in a new modeling paradigm for physical systems, which is known as data-driven modeling. By deploying machine learning techniques, these existing scientific databases can be utilized for modeling, analysis, and design in aerodynamics.

More recently, in fluid mechanics, researchers in the field of turbulence modeling extensively applied machine learning techniques to quantify uncertainties and improve Reynolds Averaged Navier-Stokes (RANS) models (Ling and Templeton, 2015; Singh and Duraisamy, 2016; Wu *et al.*, 2018; and Duraisamy *et al.*, 2019). Besides, Zhu *et al.* (2019) assessed the ability of a ML algorithm to

replace RANS models for high Reynolds number flows. Ma *et al.* (2015) obtained a closure model for two-fluid equations by mining Direct Numerical Simulation (DNS) data of bubbly flow. Wang *et al.* (2018) obtained a data-driven closure for subgrid scale stress in Large Eddy Simulations (LESs). Maulik *et al.* (2018) utilized neural networks to perform convolution and deconvolution in LESs to account for subgrid scale stresses. These aforementioned studies reflect the increased attention of applying ML techniques in the fluid mechanics community.

In addition, several attempts are made to predict the flow field using deep learning techniques. Guo *et al.* (2016) utilized Convolutional Neural Network (CNN) to predict the velocity field over several geometrical shapes and achieved an accuracy of about 98%. In their approach, image-to-image regression is performed using convolution-deconvolution operations. The input image is represented by a signed distance function with zero level set representing the boundary of the geometry. Then, the input image is mapped to the output image using CNN, which represents the velocity field. Although the overall result seems accurate, the prediction close to the boundary, such as pressure distribution over the surface, may suffer from inaccuracies. This is due to the fact that the geometry is approximated using image pixels, which is Cartesian in nature. This type of image data may not be able to represent geometries such as airfoil accurately, or it requires very fine pixels to approximate the airfoil curvature more accurately. Jin *et al.* (2018) proposed the $C_p - u$ model based on deep convolutional neural networks to predict the unsteady velocity field around a circular cylinder from the pressure distribution measurements. In this approach, the pressure distribution around the cylinder over time is given as an input image to CNN. The velocity field over fixed grid points is obtained as the output. The prediction results show a good accuracy in comparison with the true data. However, the method is demonstrated only for fixed geometry with fixed grid points.

In our present study, we explore the ability of the deep learning approach for the flow field prediction of a large number of airfoils (varying geometry) for the first time. The present approach consists of two steps. In the first step, a new parameterization method based on CNN is utilized to parameterize the airfoil geometry for the first time. The conventional parameterization techniques fail to generalize all airfoils and to some extent lack accuracy and require manual interference in parameterization (Samareh, 1999; Song and Keane, 2004). The present CNN-based parameterization approach is very general, flexible, easy to implement, and directly applicable to 3D general geometries, and the generalization performance is improved with an increase in the number of airfoil samples (property of deep learning). The parameterization step is a preprocessing step and is independent of the task of flow field prediction. The obtained airfoil parameters, which possess all the geometric features of the airfoil, are used as input features in the flow prediction network. In addition, the airfoil parameters can also be used in other applications such as airfoil optimization (Jameson, 1988; Skinner and Zare-Behtash, 2018).

Furthermore, airfoil parameters along with x-y grid coordinates and flow conditions (Reynolds number and angle of attack) are fed as input to a Multilayer Perceptron (MLP) network to train a model for approximating the flow field. The MLP network is trained to predict the flow field point by point in order to obtain the prediction near the boundary more accurately, as required in airfoils. Once the network is trained, in order to assess the accuracy, the trained model is tested for new airfoil cases which are not used in training. The paper is organized as follows. Section II provides the architectural and implementation details on the parameterization network (CNN) and flow prediction network (MLP). The data preparation processes required for both networks are explained in Sec. III. The prediction results of both networks are presented and discussed in detail in Sec. IV. The paper ends with concluding remarks given in Sec. V along with future scope.

II. DETAILS ON MACHINE LEARNING

In general, machine learning is broadly classified into supervised and unsupervised techniques. The supervised learning techniques are trained using labeled inputs in order to create a prediction model, whereas unsupervised techniques use nonlabeled inputs. The present study focuses on the supervised regression techniques. Deep learning is a term used to describe a class of algorithms, whose performance scales with data, more specific to neural networks with deep layers (>4). In this study, both deep MLP and deep CNN are utilized in order to predict the flow field over different airfoils. The brief details about the utilized MLP and CNN architectures are given below.

A. Details on MLP–Flow prediction network

Multilayer Perceptron (MLP) (Rosenblatt, 1957) is generally known as a neural network consisting of several neurons (also known as nodes) connected together to form a complex network. The neuron receives several input signals and produces an output. Hence, one can consider the neuron as a unit to perform a nonlinear input-output mapping.

The utilized MLP architecture is shown in Fig. 1, which consists of several layers with a number of neurons in each layer. The first layer of the MLP receives the user-defined input features. The first layer operates by constructing linear combination of the input features and outputs the transformed features via a nonlinear activation function. The transformed output of the first layer is received by the second layer, and the operation continues layer by layer until the last layer, which predicts the output. In this way, the given input features are mapped to the output features in steps through a series of mappings. The output of a neuron in the first layer is shown in Eq. (1), and the output of a neuron in the lth layer is given in Eq. (2)

$$a_j^1 = \sigma_1\left(\sum_{k=1}^{20} w_{jk}^1 x_k + b_j^1\right), \tag{1}$$

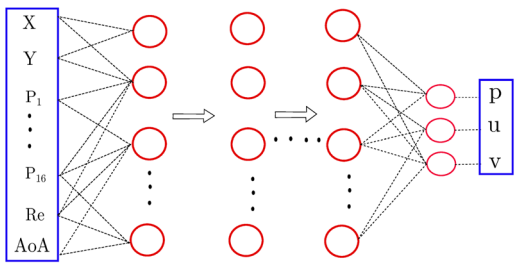

FIG. 1. MLP architecture used in the flow prediction network.

$$a_j^l = \sigma_l\left(\sum_{k=1}^{N^{(l-1)}} w_{jk}^l a_k^{l-1} + b_j^l\right). \tag{2}$$

In the above equations, σ denotes the nonlinear activation function; w_{jk} denotes the weight connection from the jth neuron of the current layer to the kth neuron in the previous layer; b denotes the bias terms and a denotes the output of each layer; and x indicates the input features. In this way, obtaining the output of each neuron from the given input is called feed-forward operation. MLP involves hyperparameters, namely, number of hidden layers, number of neurons in each layer, and nonlinear activation function (σ). It is important to choose optimum hyperparameters to train the model properly. The rectified linear unit function [ReLU = max of (0, x)] is the most general form of activation utilized in deep learning because of its ability to train the network much faster. The weights and biases of the MLP are trained using error backpropagation algorithms (Rumelhart *et al.*, 1988). Further details of MLP and its training algorithms are given in Nielsen (2015).

B. Details on CNN—Parameterization network

CNN (LeCun *et al.*, 1989) is developed in order to overcome the limitations of MLP in handling image (grid)-like data. MLP is inefficient in handling image data since it involves too many inputs and thus too many parameters to be trained. But CNN can handle it efficiently as it requires fewer parameters. Therefore, CNN is widely used in fields such as image recognition and computer vision (LeCun *et al.*, 2015). The above explored flow prediction network involves 16 airfoil parameters (P_1–P_{16}), which possess the details about the airfoil geometry. The airfoil parameters are obtained by utilizing CNN as a parameterization network. From the given airfoil image input, CNN reduces the geometrical details in the image into a few useful parameters. The utilized CNN architecture is shown in Fig. 2. The network is very similar to an autoencoder. In autoencoders, the given input image is recovered as output, whereas here the airfoil shape (y-coordinates) is recovered as output. The shown CNN architecture consists of convolutional layers, pooling layers, and fully connected layers. The brief details about these different layers are explained below.

1. Convolutional layer

The convolutional operation is an essential procedure in the whole CNN operation. The convolutional layer and the nonlinear activation function compute the output of the input via a convolutional operation as given in the following equation:

$$\mathbf{y}_i = \sigma(\mathbf{k}_i * \mathbf{x} + b_i),\ i = 1, 2, \ldots, K, \tag{3}$$

in which $\mathbf{x}$ is the input of the convolutional layer with width W_1, height H_1, and depth D_1. The operator $*$ is the convolution operation; $\mathbf{k}_i$ is the ith trainable convolutional filter with dimensions of $F \times F \times D_1$ (width, height, and depth, respectively); b_i is the ith bias for the convolutional filter $\mathbf{k}_i$; σ is the nonlinear activation function; and $\mathbf{y}_i$ indicates the ith output matrix corresponding to the ith convolutional filter. There are a number of convolutional filters (K) at each layer. The convolutional operation is demonstrated at length in Fig. 3, which holds the stride of the convolutional process to be $S = 1$. For the output of the ith convolutional process $\mathbf{y}_i$, the width is W_2, the height is H_2, and the depth dimension of the whole output is K. Every element in the ith output matrix is computed by dot-product of the corresponding window of the input matrix and the ith convolutional filter $\mathbf{k}_i$.

To adjust the width and height of the output feature map, zeroes can be padded around the borders of the input matrix. In general, setting the number of padding zeroes on each side to be $P = (F - 1)/2$ when the stride is $S = 1$ ensures that the input volume and output volume will have the same size spatially. In Fig. 3, one layer of zeroes (in gray) is inserted around each side of the original input matrix (in purple). Therefore, the width W_2 and the height H_2 of the output can be computed by

$$\begin{cases} W_2 = \dfrac{W_1 - F + 2P}{S} + 1, \\ H_2 = \dfrac{H_1 - F + 2P}{S} + 1. \end{cases} \tag{4}$$

2. Pooling and fully connected layer

The pooling operation is a downsampling process, which extracts a certain property (such as the maximum value, averaged value, and L_2-norm) as the lower-dimensional output from the sampling window of the data in the corresponding pooling window. Taking max pooling as an example, the downsampling operations of the pooling layer are illustrated in Fig. 4. The procedure of taking the maximum value in the pooling window is conducted independently on every slice in the depth dimension of the input and thus downsamples the input matrix spatially, while the depth dimension of the output matrix remains the same as the input one. Besides, the width and height of the output matrix can also be computed by the same way as in the convolutional layer, and the stride usually keeps identical to the width or height of the pooling window.

The pooling layer can progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network and hence to control overfitting as well. The fully connected (FC) layer, as the name indicates, has dense connections of neurons between two layers. The details are the same as explained in Subsection II A.

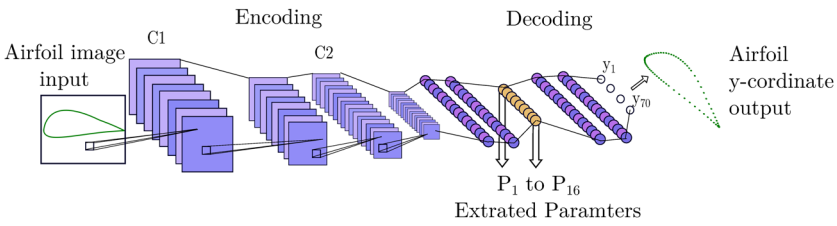

FIG. 2. CNN shape parameterization network.

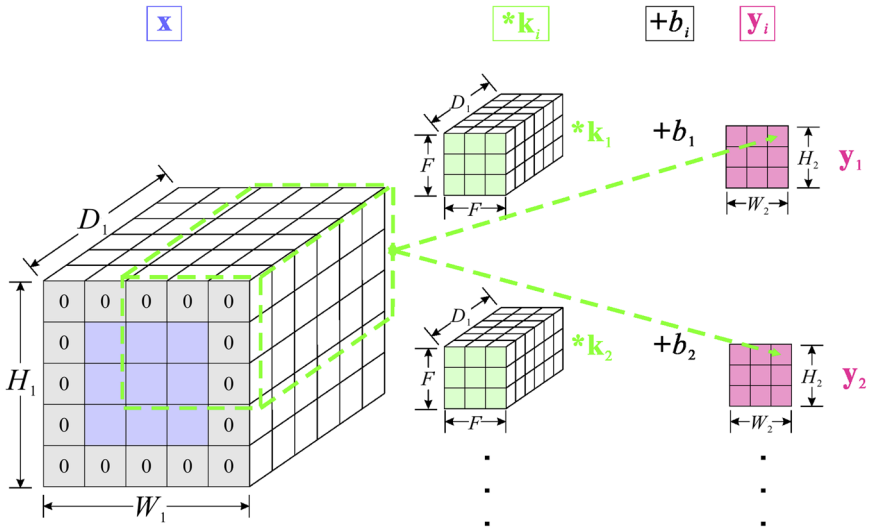

FIG. 3. Schematic diagram of the convolutional layer and convolutional process.

In this network, at some point, the number of nodes in a layer is chosen as equal to the number of required airfoil shape parameters (P_1–P_{16} in Fig. 1), and we call it parameter layer. Through a series of convolutional and pooling operations, the network learns the spatial correlation in the input airfoil image. Furthermore, the convolved features are given to FC layers to reduce it to a fixed number of airfoil parameters. Finally, the network is expanded to recover the airfoil shape (y-coordinates) from the airfoil parameters. The network before the parameter layer is considered as an encoder, and the network after it is considered as a decoder. It is clear that the airfoil parameters (16 in this case) contain the details of the airfoil geometry, and through some operation (decoding), the shape can be recovered. Hence, the airfoil parameters can be utilized to take place of the original airfoil geometry (with massive x-y coordinates) in applications such as flow prediction and shape optimization of a certain airfoil. In addition, this type of parameterization method is general and can be easily applied to any 3D geometry, once sufficiently trained.

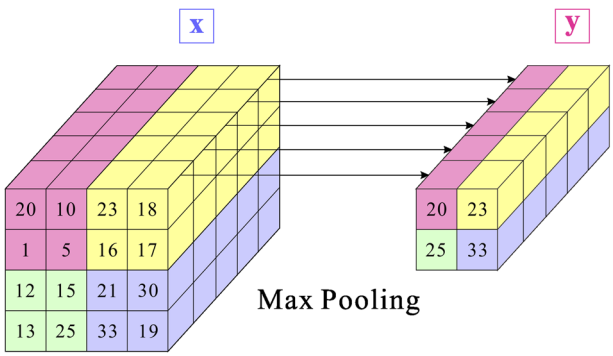

FIG. 4. Downsampling process of the max pooling layer (x and y are the input matrix and output matrix, respectively).

III. DATA PREPARATION

Database preparation is an important aspect for the deep learning to have a better training and prediction performance. The present approach consists of two stages: the parameterization of the airfoil is performed in the first stage and flow field prediction is performed in the second stage. The data acquisition in each stage is described in Secs. III A and III B.

A. Data preparation—Shape parameterization

CNN is utilized to parameterize the airfoil geometry into a few useful parameters. The inverted gray images of airfoils (with a size of 216 × 216) are given as input to the CNN in order to train the network, extract airfoil shape parameters, and perform shape prediction, once trained. The inverted image consists of pixel values between 0 and 1 as described in Guo *et al.* (2016). The pixels at which the airfoil profile fully passes through will have a value 1, and the pixels at which the airfoil profile does not pass through will have a value 0. Other pixels will have a value in between 0 and 1. The required airfoil coordinate details are obtained from the UIUC database (Selig, 1996). A total number of 1550 airfoil shapes are used to form the database required for the shape parameterization network (1200 airfoils for training and validating the network and the other 350 for testing). The output values of the parameterization network for each airfoil are the y-coordinates at corresponding fixed x positions along the chord length. A total number of 70 x-locations are used: 35 for upper surface and 35 for the lower surface. The mentioned input-output sets of the database are used to train the CNN for parameterization. A typical airfoil input image to CNN is shown in Fig. 5.

B. Data preparation—Flow field prediction

MLP is utilized to predict the flow field over airfoils. The required flow field database of airfoils is generated using the OpenFOAM solver (Weller *et al.*, 1998; Jasak *et al.*, 2007). A total of 110 NACA airfoils with several angles of attack ranging from 0° to 14°

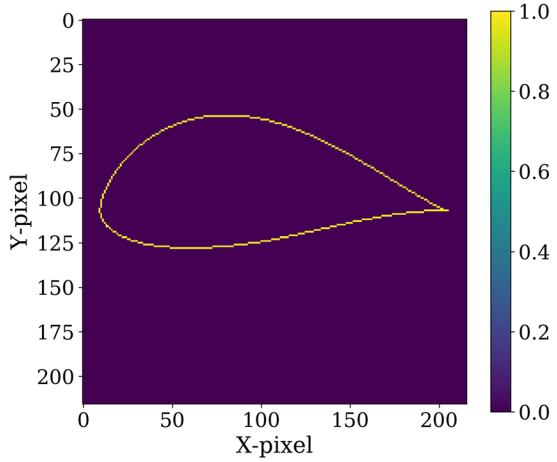

FIG. 5. Airfoil input image for the parameterization CNN.

(AoA: 0°, 2°, 4°, 6°, 8°, 10°, 12°, and 14°) and several Reynolds numbers ranging from 100 to 2000 (Re: 100, 200, 300, 400, 600, 800, 1000, 1200, 1500, 1800, and 2000) are considered for the training database. For each airfoil, flow fields are generated under 8 Reynolds numbers and 6 angles of attack (both chosen randomly from the given list of Re and AoA for each airfoil), which creates 48 flow cases for each airfoil. Therefore, a total of 5280 training cases are created for 110 airfoils. For each case, around 12 000 data points are chosen from the region close to the airfoil (x: $-0.5 \sim 2.0$, y: $-0.5 \sim 0.5$), which consists of 60×10^6 data points for all the training cases. A total of 13 airfoils (not used in network training) along with 6 Reynolds numbers (Re: 500, 900, 1300, 1700, 1900, and 2500) and 6 angles of attack (AoA: 1°, 3°, 5°, 7°, 11°, and 13°) which are not used in training, constitute the 468 testing cases for the trained model. The value of each physical variable around an airfoil is a function of its coordinate locations (x, y), geometrical shape, Reynolds number, and angle of attack. Therefore, the x-y coordinates, 16 airfoil parameters (obtained from parameterization), Reynolds number, and angle of attack are used as inputs to MLP. The outputs are the flow field, i.e., pressure (p) and velocities (u, v) corresponding to the given x-y coordinates, Reynolds number, and angle of attack. Therefore, the training database consists of $60 \times 20 \times 10^6$ inputs and $60 \times 3 \times 10^6$ outputs.

IV. RESULTS AND DISCUSSIONS

The network implementation, training, and testing are performed in Tensorflow (Abadi et al., 2015) with Keras API (Chollet, 2015) for both CNN and MLP. These two networks are constructed independently, but the airfoil parameters obtained from the parameterization network (CNN) are used in the flow field prediction network (MLP). The training details and prediction results of both networks are presented below.

A. Parameterization network—CNN

CNN is used as a parameterization network to reduce the airfoil image into useful parameters. CNN training is an optimization process, in which the weights and biases of the network are trained using the backpropagation algorithm (Rumelhart *et al.*, 1988; LeCun *et al.*, 1989). The optimization is performed such that the defined Mean Square Error (MSE) is minimized. The MSE for the parameterization case is defined for a batch of N samples as

$$E_{CNN} = \frac{1}{N}\sum_{m=1}^{N}\sum_{i=1}^{70}\left(y_{i,m}^{t} - y_{i,m}^{p}\right)^2, \tag{5}$$

where y^t indicates the true y-coordinates of the airfoils and y^p indicates CNN predictions of y-coordinates. The training of CNN involves training of weights and biases from both convolutional (k_c, b_c) and fully connected (w_f, b_f) layers. Let us denote the trainable parameters as $W = [k_c b_c w_f b_f]$. In order to optimize the trainable parameters, one needs to obtain the error gradients (E_w) with respect to trainable parameters using backpropagation algorithm. Then, a stochastic gradient descent optimizer is used to update the weights and biases as defined in the following equation:

$$W_j^{n+1} = W_j^n - \alpha\frac{\partial E_{CNN}}{\partial W_j}, \tag{6}$$

where α indicates the learning rate. A total of 1200 airfoil shapes are given as input to CNN to train the model; among them, 80% of samples are used for training and the other 20% are used for validation. Normalization is applied to output features such that all the output values are within the range of −1 to 1. No normalization is required for the input images as the pixel values of the images are within the range of 0–1. The training is performed using a stochastic gradient descent optimizer called ADAM (Kingma and Ba, 2014). So far, ADAM may be the best choice of algorithm for deep learning (Ruder, 2017). The default ADAM parameters are chosen along with an initial learning rate (LR) of 2.5×10^{-5}. A LR scheduler is utilized to reduce the LR in case the MSE of validation does not decrease over several epochs. The training is performed until the MSE of validation reaches a steady state. The training is terminated in case the MSE of validation does not decrease further even after reducing the LR. The MSEs are reduced to a value of below 1.0×10^{-5} for training and 1.0×10^{-4} for validation at the end of the training process for all the utilized architectures. The MSE convergence history of all the architectures is shown in Fig. 6. The utilized hyperparameters of the parameterization networks are listed in Table I. The rectified Linear Unit (ReLU) activation function is utilized in all convolutional layers, and hyperbolic tangent (tanh) activation is used in all the fully connected layers, except the output layer in which a linear activation function is utilized.

Several CNN architectures are investigated by varying the number of convolutional layers and number of fully connected layers as shown in Table I. Based on trial and error, a CNN architecture with 3 convolution layers and 5 fully connected layers is considered as a base model called CNN-C3F5. Further convolutional and fully connected layers are added to the base model as shown in Table I. Training is performed for all the models, and the convergence histories of MSE are shown in Fig. 6. With the number of convolutional layers increased from 3 to 5, although the convergence rate is slowed down, the error convergence is slightly improved. Furthermore, the number of convolution layers is fixed as 5, and the number of fully connected layers is varied. It is observed that increasing fully connected layers from 5 to 7 has improved the error convergence

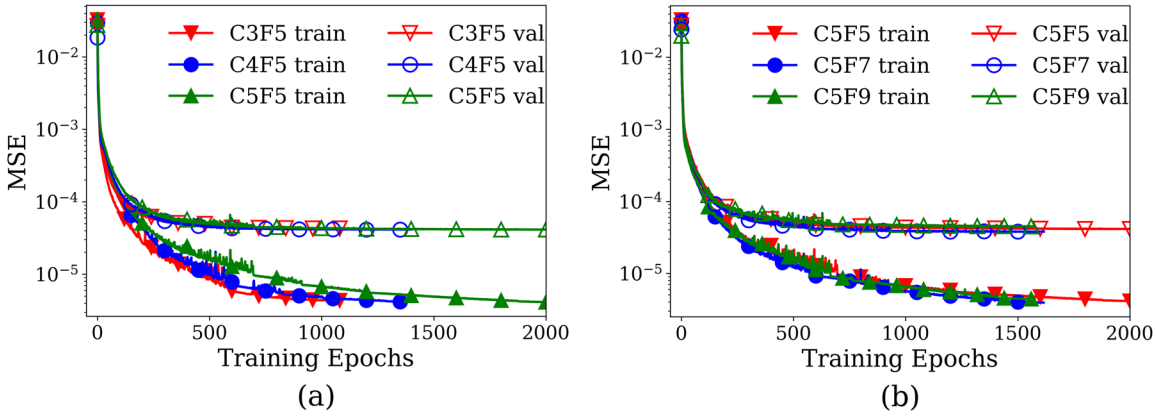

FIG. 6. CNN—MSE convergence. (a) Variation of the number of convolutional layers; (b) variation of the number of fully connected layers (train indicates training error, and val indicates validation error).

slightly, while further increasing the fully connected layers from 7 to 9 leads to a slightly larger MSE value. Besides, changing the filter size and number of filters shows no significant improvement in the error convergence. From Fig. 6, it is clear that all the CNN architectures achieve similar error convergence, and hence we can consider that the hyperparameters are converged. Among the utilized CNN architectures, although there is no significant difference in the error convergence, the CNN-C5F7 architecture is utilized for further study because of its lowest validation error.

Once the training is done, the trained model architecture CNN-C5F7 is tested using 350 airfoil samples, which are unseen by the model during training. Figure 7 shows the comparison of the true airfoil shapes and the shapes predicted by CNN-C5F7 (i.e., shapes recovered from the 16 parameters) for a sample training airfoil and a sample testing airfoil. The L_2 relative error distribution of the parameterization process is shown in Fig. 8. It is observed from the results that the model is able to reproduce all the shapes from the input image with an accuracy of over 99.5% (L_2 relative error) for training airfoils and over 99.0% (L_2 relative error) for testing airfoils. Therefore, a parameterization network is able to parameterize the given airfoil shape effectively and is able to recover the shape from the parameters. For the current network, it is observed that a total of 16 parameters are required to parameterize all the airfoils accurately. However, depending on the shapes to be parameterized and by constructing an efficient network in the future, the parameters may be reduced further. With the aid of the trained network, the parameters can be decoded to obtain the airfoil shape (i.e., airfoil coordinates). This indicates that the parameters contain all the geometrical details of the airfoil in an encoded format, which therefore can be made use of representing the airfoil geometry. For each airfoil, the parameters are extracted from the network and utilized further to predict the flow field near the airfoil.

This type of CNN-based parameterization is very simple, accurate, flexible, and easy to deploy. Another merit of this parameterization technique is that it can be used for any general airfoil shapes without any restriction. Obviously, this method is superior to conventional methods, which are sometimes erroneous for some types of airfoils and cannot parameterize them accurately. In addition, this type of parameterization requires less manual interference compared to the conventional methods. Furthermore, this method can

TABLE I. CNN architectures with their hyperparameters. (In the 1st convolutional layer of CNN-C3F5, 4 × 4 indicates the convolutional filter size, 32 indicates the number of convolutional filters, and 3 × 3 indicates the size of the pooling filter. In the fully connected layer of CNN-C3F5, 2 × 100 indicates 2 layers and 100 neurons in each layer.)

Layer type	CNN-C3F5	CNN-C4F5	CNN-C5F5	CNN-C5F7	CNN-C5F9
Input	216 × 216 × 1	216 × 216 × 1	216 × 216 × 1	216 × 216 × 1	216 × 216 × 1
1st convolution	4 × 4, 32, 3 × 3	4 × 4, 32, 3 × 3	4 × 4, 32, 3 × 3	4 × 4, 32, 3 × 3	4 × 4, 32, 3 × 3
2nd convolution	4 × 4, 64, 3 × 3	4 × 4, 32, 3 × 3	4 × 4, 32, 3 × 3	4 × 4, 32, 3 × 3	4 × 4, 32, 3 × 3
3rd convolution	4 × 4, 128, 3 × 3	4 × 4, 64, 3 × 3	4 × 4, 64, 3 × 3	4 × 4, 64, 3 × 3	4 × 4, 64, 3 × 3
4th convolution	...	4 × 4, 128, 2 × 2	4 × 4, 64, 2 × 2	4 × 4, 64, 2 × 2	4 × 4, 64, 2 × 2
5th convolution	...	...	4 × 4, 128, 2 × 2	4 × 4, 128, 2 × 2	4 × 4, 128, 2 × 2
Fully connected	2 × 100	2 × 100	2 × 100	3 × 100	4 × 100
Fully connected	1 × 16	1 × 16	1 × 16	1 × 16	1 × 16
Fully connected	2 × 100	2 × 100	2 × 100	3 × 100	4 × 100
Output layer	70	70	70	70	70

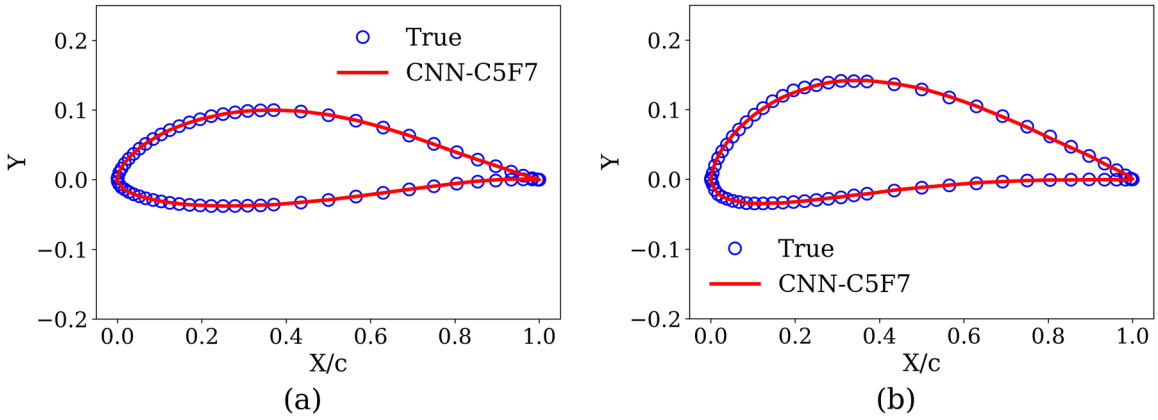

FIG. 7. Comparison of the original airfoil shape with the shape recovered from its parameters. (a) A sample training airfoil. (b) A sample testing airfoil.

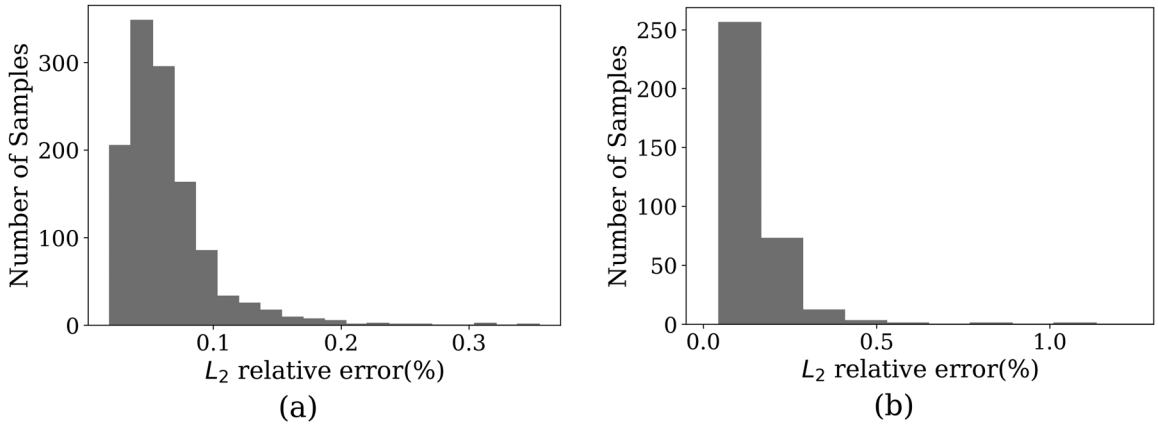

FIG. 8. L_2 relative error distribution of airfoil parameterization. (a) Training set. (b) Testing set.

be directly applicable to general 3D geometrical shapes and is not limited to airfoils. Also, the network size can also be scaled easily depending on the number of input sample shapes, which may vary from hundreds to millions.

B. Flow prediction network—MLP

The MLP model is utilized here as a flow prediction network. The model is trained such that the defined MSE of training is reduced. The MSE for the prediction network can be defined as shown in the following equation:

$$E_{MLP} = \frac{1}{3 \times N} \sum_{m=1}^{N} \left[\left(p_m^t - p_m^p\right)^2 + \left(u_m^t - u_m^p\right)^2 + \left(v_m^t - v_m^p\right)^2 \right], \quad (7)$$

where p, u, and v are pressure, u-velocity component, and v-velocity component, respectively, and superscript t indicates the true value and p indicates the predicted value. Training of MLP involves training the weights and biases using the backpropagation algorithm, which is similar to the training of fully connected layers in CNN.

The airfoil parameters along with flow parameters are fed as inputs to the MLP network in order to obtain a model to predict the flow field. Since the airfoil parameters possess the airfoil geometrical features, these features will be decoded by the MLP during training such that the flow field can be obtained. Hence, the model can be represented as

$$(p, u, v) = f_{MLP}(x, y, P_1, \ldots, P_{16}, \mathrm{Re}, \mathrm{AoA}), \quad (8)$$

where f_{MLP} indicates the trained MLP model. A total of 60×10^6 data points constructed by 110 NACA airfoils with various flow

TABLE II. MLP networks with their hyperparameters. (In the hidden layer of MLP-1, 8×800 indicates 8 layers and 800 neurons in each layer.)

Layer type	MLP-1	MLP-2	MLP-3
Input	1×20	1×20	1×20
Hidden	8×800	10×1000	12×1200
Output	1×3	1×3	1×3

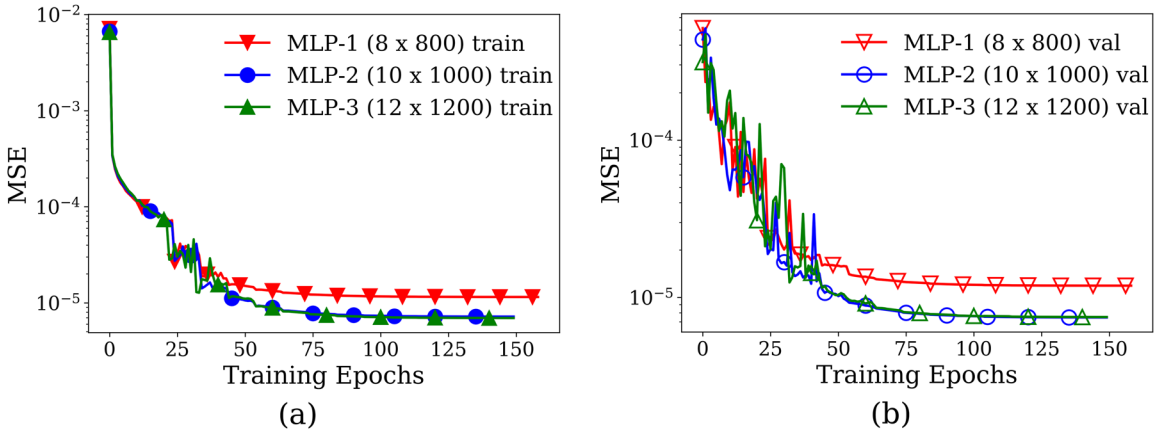

FIG. 9. MSE convergence of the utilized MLP flow prediction networks. (a) MSE of training (train indicates training error); (b) MSE of validation (val indicates validation error).

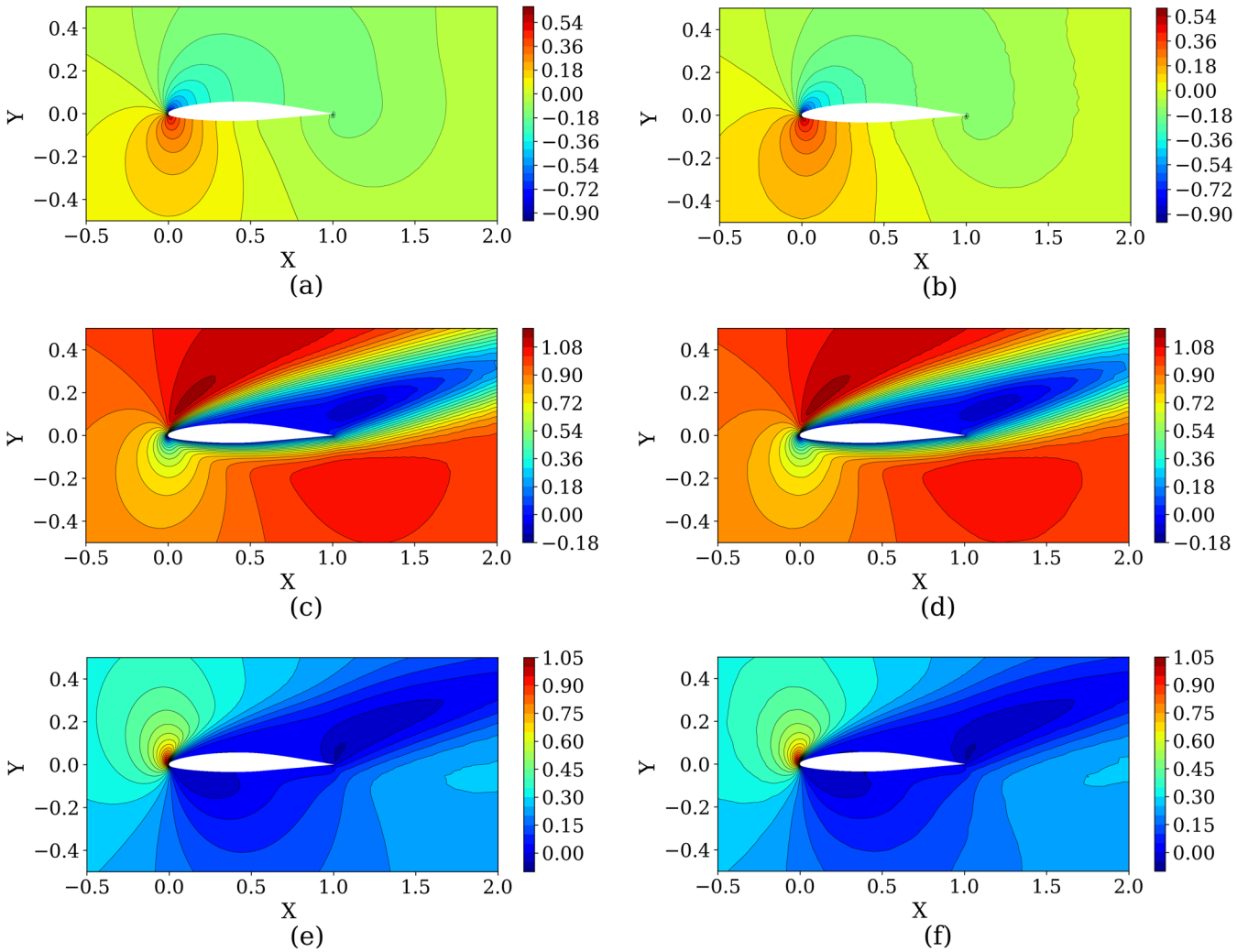

FIG. 10. Training airfoil NACA 65209 (Re = 400, AoA = 14°)—comparison of the flow field. (a) *p*-CFD, (b) p-prediction, (c) u-CFD, (d) u-prediction, (e) v-CFD, and (f) v-prediction.

conditions are used for training. Normalization is applied to all the input features such that all the input values are within the range of 0–1, and no normalization is applied to output features as most data values are within the range of −1 to 1. Trainings are performed with a training-validation split of 90%–10% of the fed input data. An ADAM optimizer is used to train the weights and biases of the MLP. A LR of 5.0×10^{-4} is used as the initial learning rate with a scheduler. Training is performed until MSE of validation reaches a steady state. Three MLP networks are used with varying hyperparameters as shown in Table II. The ReLU activation function is utilized in all layers except the output layer, which utilizes a linear activation function. Based on trial and error, an 8 × 800 (8 layers and 800 neurons in each layer) network is chosen as the base model (MLP-1). From MLP-1, increasing the parameters to 10 × 1000 (MLP-2) improves the error convergence significantly. Furthermore, increasing the parameters to 12 × 1200 (MLP-3) shows no significant improvement over 10 × 1000 (MLP-2). The utilized MLP networks are given in Table II, and the convergence of MSE of training and validation for the networks is shown in Fig. 9. MLP-2 is chosen for further study because of its lowest validation error.

A total of 13 NACA airfoils which are unseen by the model during the training, as mentioned in Sec. III B, are used to test the trained model. The trained MLP-2 model is utilized to perform the prediction for both training and testing cases. The prediction results of the flow field (pressure and velocity components) for a training airfoil case (NACA 65209, Re = 400, AoA = 14°) and a testing airfoil case (NACA 63415, Re = 1900, AoA = 7°) are shown in Figs. 10 and 11, respectively. The predicted contour patterns of the flow fields show a good match with the CFD results. In addition, the absolute error contours of the predicted flow field are shown in Figs. 12 and 13 for the training and the testing airfoil, respectively.

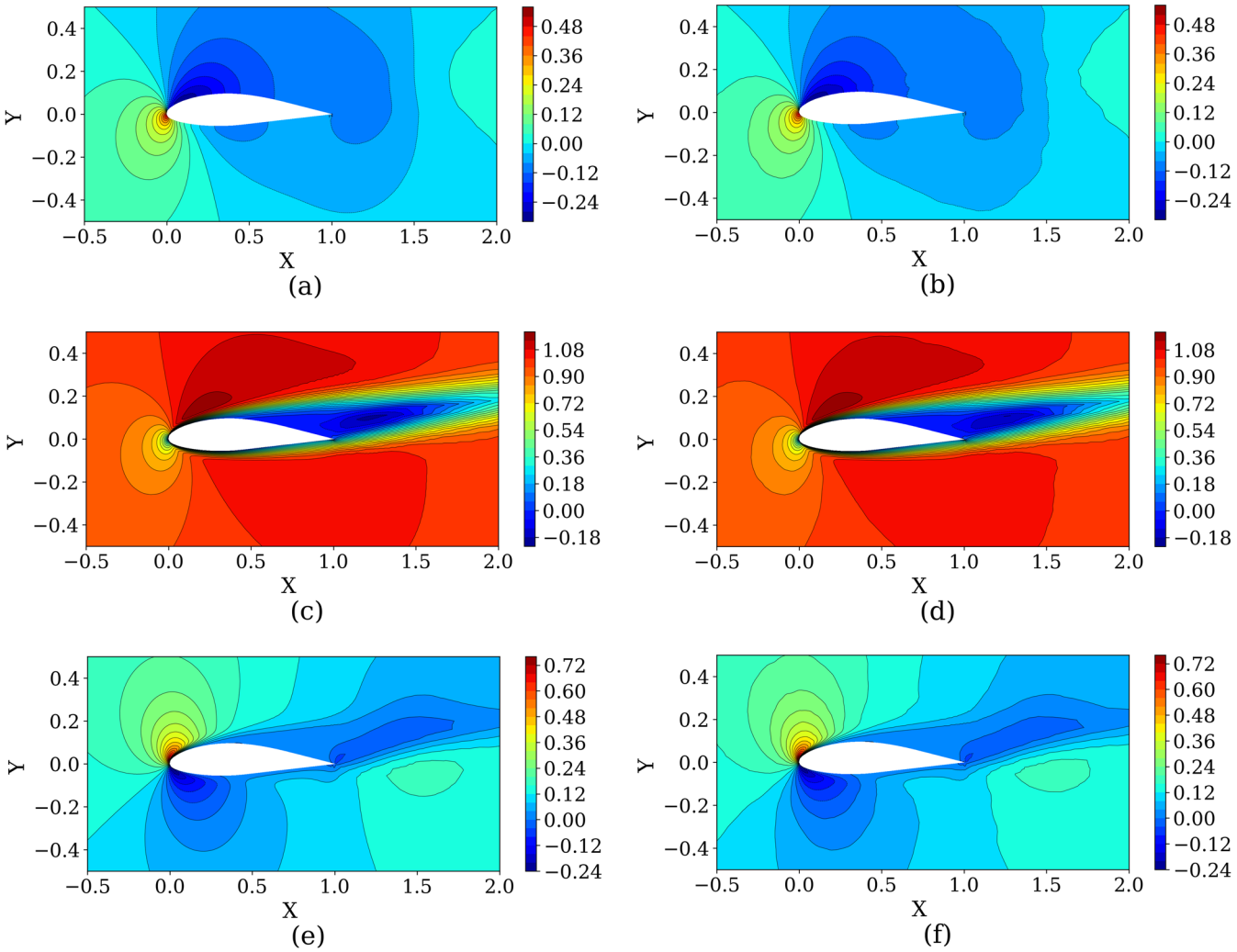

FIG. 11. Testing airfoil NACA 63415 (Re = 1900, AoA = 7°)—comparison of the flow field. (a) *p*-CFD, (b) p-prediction, (c) u-CFD, (d) u-prediction, (e) v-CFD, and (f) v-prediction.

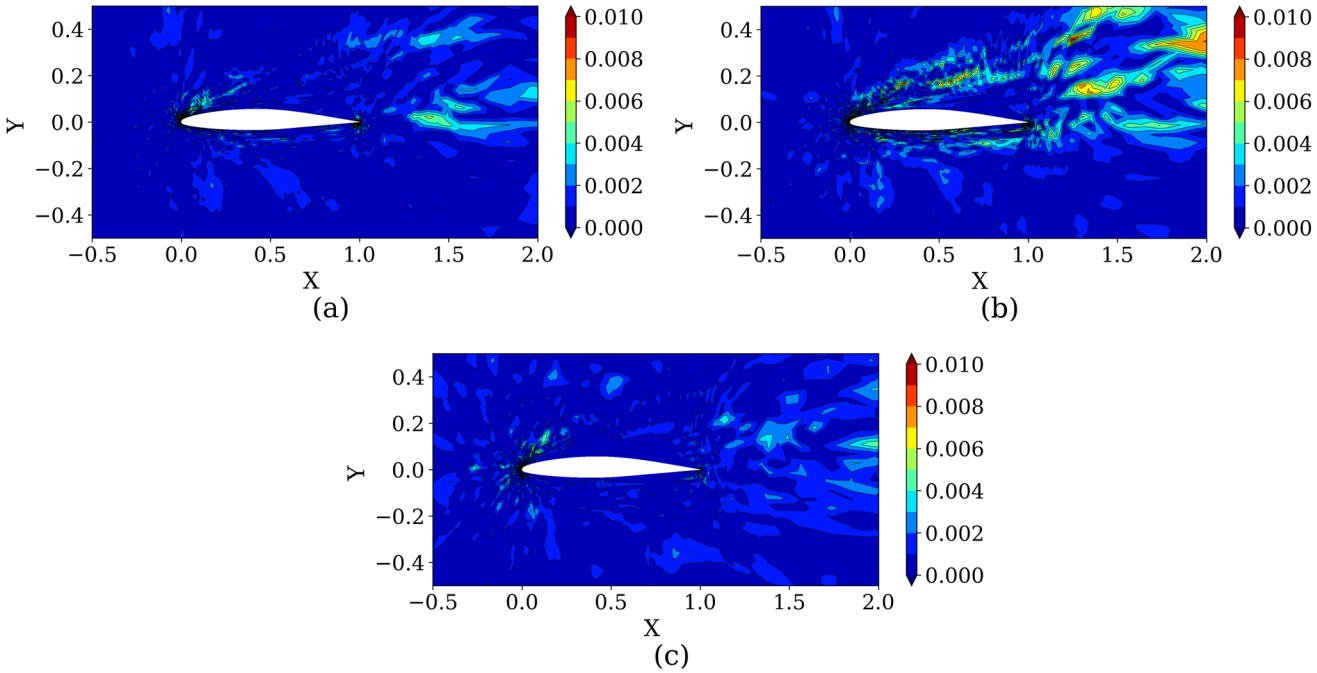

FIG. 12. Training airfoil NACA 65209 (Re = 400, AoA = 14°)—comparison of the absolute error between the CFD and predicted flow field. (a) p-error, (b) u-error, and (c) v-error.

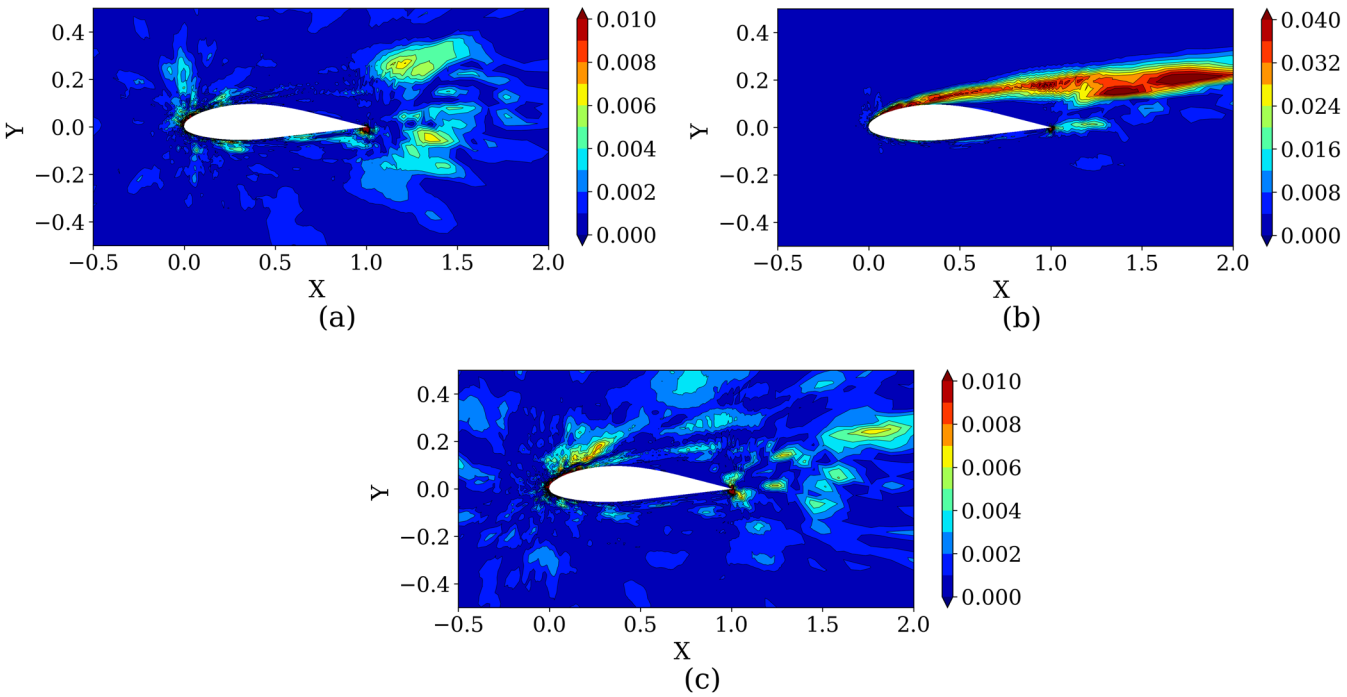

FIG. 13. Testing airfoil NACA 63415 (Re = 1900, AoA = 7°)—comparison of the absolute error between the CFD and predicted flow field. (a) p-error, (b) u-error, and (c) v-error.

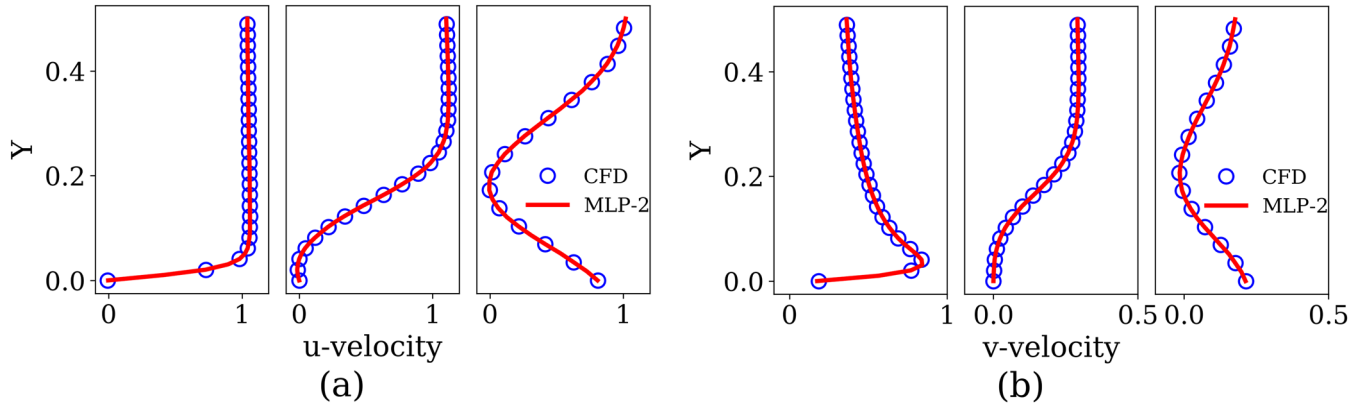

FIG. 14. Training airfoil NACA 65209 (Re = 400, AoA = 14°)—comparison of velocity profiles. (a) u-velocity profile comparison at x/c = 0, 0.5 and 1.5 and (b) v-velocity profile comparison at x/c = 0, 0.5, and 1.5.

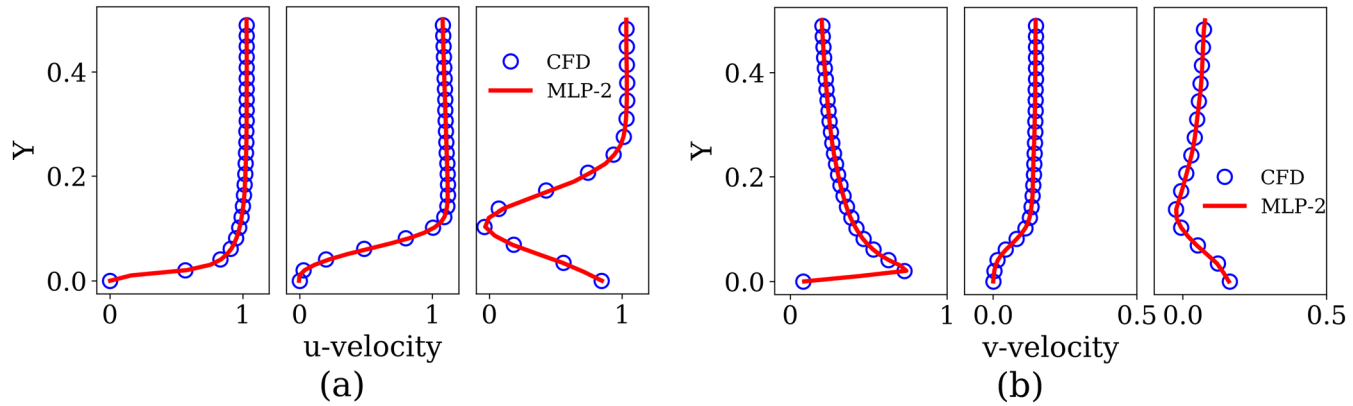

FIG. 15. Testing airfoil NACA 63415 (Re = 1900, AoA = 7°)—comparison of velocity profiles. (a) u-velocity profile comparison at x/c = 0, 0.5, and 1.5 and (b) v-velocity profile comparison at x/c = 0, 0.5, and 1.5.

A comparison of velocity profiles on the upper side of the airfoil at different locations of X/c = 0, 0.5, and 1.5 is shown in Figs. 14 and 15 for the training and the testing airfoil, respectively. It is evident from the comparison that the predicted velocity profiles have a good match with the CFD results. In addition, the predicted pressure coefficient (C_p) distributions over the airfoil surface are shown in Figs. 16 and 17, respectively, along with CFD results. It is clear from the results that the pressure field prediction near the boundary

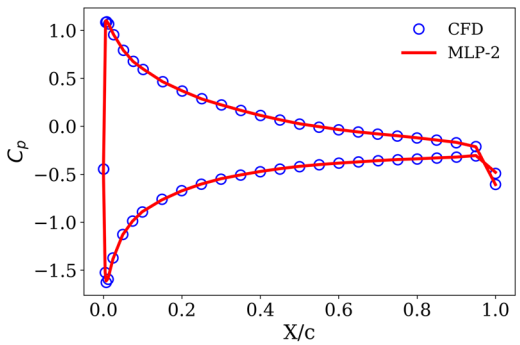

FIG. 16. Training airfoil NACA 65209 (Re = 400, AoA = 14°)—comparison of the C_p distribution.

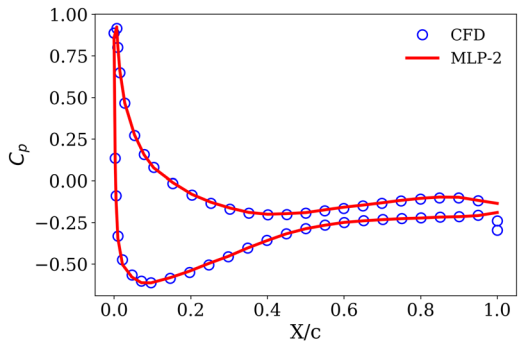

FIG. 17. Testing airfoil NACA 63415 (Re = 1900, AoA = 7°)—comparison of the C_p distribution.

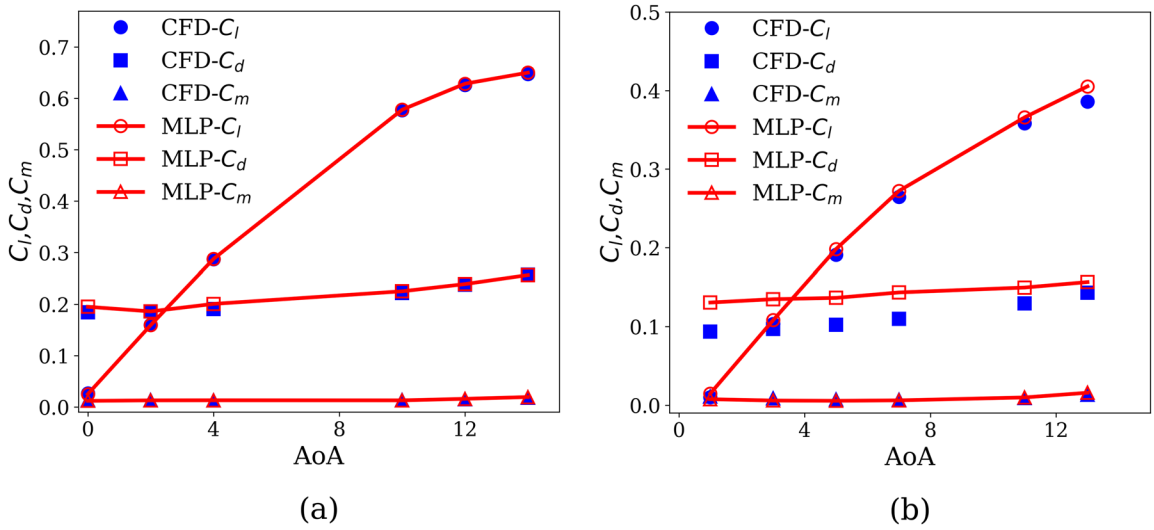

FIG. 18. Comparison of C_l, C_d, and C_m with variation of angle of attack. (a) Training airfoil NACA 65209 (Re = 400). (b) Testing airfoil NACA 63415 (Re = 1900).

also keeps good accuracy, in comparison with CFD results. Besides, the comparison with CFD results of variation of the predicted lift coefficient (C_l), drag coefficient (C_d), and moment coefficient (C_m) with respect to angles of attack for the training and testing airfoils are shown in Figs. 18(a) and 18(b), respectively. As shown in Fig. 18(a), for the training airfoil, the aerodynamic coefficients calculated from the predicted flow field show good match with the CFD results. For the testing airfoil, as shown in Fig. 18(b), the drag coefficient shows slight mismatch with the CFD results. This is due to the fact that although the velocity field has good accuracy near the boundary, the grid size near the boundary is in the order of 10^{-3} unit, which indicates that a slight error in the predicted velocity field results in a larger error in the velocity gradients. Hence, there is a slight mismatch of the predicted drag coefficients. Overall, the results are reasonably accurate.

From the presented results, it is observed that the trained model shows a good performance in prediction of the flow field. This also confirms that the parameterization is effective and the CNN network is able to recover the geometric features from the parameters such that the flow field can be predicted accurately. The computational times needed by the network training and prediction are shown in Table III, along with the time required for CFD simulations. Compared to the conventional CFD methods, the present approach is much faster and takes about only a few seconds to predict the flow field near the given airfoil, once trained. In addition, due to the implementation of boundary conditions, the CFD methods require a flow domain of very large size to be considered around the airfoil. But in the present approach, the flow domain of interest is flexible to choose. In most situations such as flow around an airfoil, the flow field around close proximity to the airfoil and in the wake region is of high importance, as considered in this approach close to the airfoil.

In the present method, based on the MLP neural network, the prediction is performed point by point. By this way, one is able to obtain the solutions more accurately near the airfoil surface. In the case of image-to-image regression, due to the Cartesian nature of the image data, the accuracy of the results close to the surface is significantly affected. In addition, prediction is performed even inside the airfoil geometry, where the flow field values are not defined. This may also be one reason for the loss of accuracy near the surface. Often, the results close to the boundary is more important as the airfoil coefficients are calculated from the surface flow field distributions. Hence, using the present approach, one can approximate the airfoil geometry more accurately than Cartesian image data. With Cartesian image data, one needs to use very fine pixels to approximate the airfoil curvature more accurately, which increases complexity and requires increased computational requirements. One can directly use the airfoil image without parameterizing it to predict the results point by point. However, each data point requires processing of the airfoil image and associated storage requirements for large data as in this case. The parameterizing approach is more elegant as once parameterized, the geometrical details can be reduced and easily used in MLP. In addition, in the case of airfoil optimization, these parameters can be directly used to perform optimization.

The L_2 relative error distribution of the prediction of training and testing cases is shown in Fig. 19. Overall, the prediction has an

TABLE III. Comparison of CPU time requirement.

Operation	CPU time (Intel Xeon at 3.3 GHz)
Training: Parameterizing network (CNN)	53.5 h
Training: Prediction network (MLP)	1440 h
Prediction time per airfoil case	1.11×10^{-3} h (4 s)
OpenFOAM simulation time per case (average)	0.16 h (600 s)
Prediction speed up	150 times

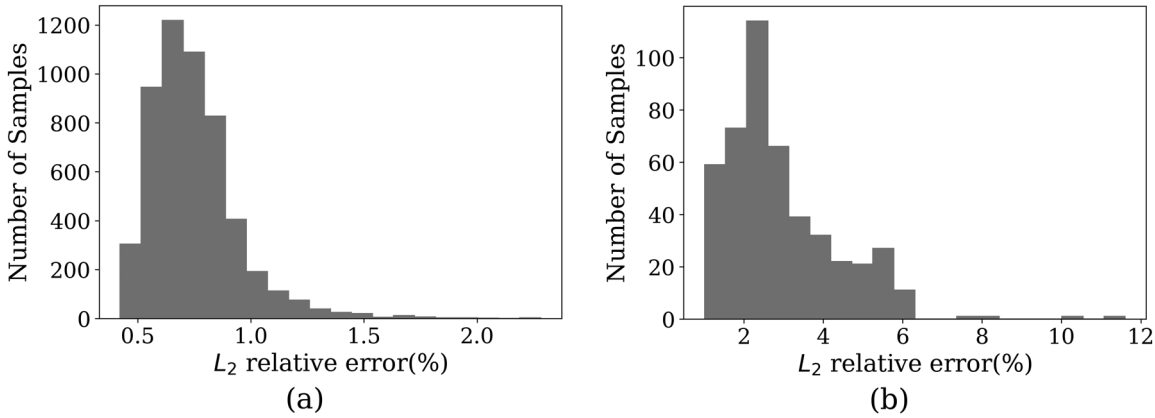

FIG. 19. L_2 relative error distribution of flow prediction. (a) Training set. (b) Testing set.

average accuracy of over 99.0% for the training cases and over 97.0% for the testing cases. Hence, we can conclude that the network is able to map the input parameters to the flow field with reasonable accuracy. In addition, it is worthy of mentioning that probably, it is the first time to use such a large database to train a deep regression network in fluid mechanics applications, which provides the proof for the ability of the deep networks to learn and approximate any nonlinear functions, such as NS equations. Overall, the proposed methodology generalizes well for the task of flow field prediction.

V. CONCLUSIONS

In this study, we presented an approach to predict the flow field over an airfoil using deep learning techniques. The approach consists of two steps. In the first step, CNN is used to parameterize the airfoils from the given input airfoil image into 16 airfoil parameters. These parameters contain all the geometrical information of the airfoil, and the airfoil shape can be recovered from these airfoil parameters. The network performs the parameterization with an accuracy of 99.5% for the training airfoils and 99.0% for the testing airfoil. In addition, compared to the conventional parameterization methods, the CNN based parameterization is very general, flexible, and directly applicable to any 2D and perhaps 3D geometries.

In the second step, a deep MLP network is used to predict the flow field over airfoils. The parameters obtained from the parameterization network along with Reynolds number, angle of attack, and x-y coordinates are used as input to train the model to predict the flow field (pressure and velocity components). The trained model has an accuracy of over 99.0% for the training cases and over 97.0% for the testing cases. The present method uses a point-by-point prediction approach, which can have a better accuracy in the near airfoil region than image-to-image prediction. This is due to the fact that the latter uses images to approximate the airfoil, which is Cartesian and cannot approximate the airfoil geometry properly. But using the present approach, the flow field near the airfoil surface can be obtained more accurately. Hence, from the prediction results, we can infer that the proposed method shows a good prediction accuracy to map the input parameters into the flow variables (p, u, and v). In addition, this is probably the first time to train a network utilizing such a large database in the application of fluid mechanics, which gives a proof that the deep network is capable of learning and approximating the nonlinear NS solutions with reasonable accuracy.

In the future, the prediction capability of the deep neural network for the turbulent flows, which is still challenging in nature, will be explored. In addition, it would be interesting to extend the work for 3D wings and any general 3D geometries.

ACKNOWLEDGMENTS

The authors are thankful to the National Supercomputing Centre (NSCC), Singapore, for providing access to the supercomputing facility. In addition, the first two authors are thankful to National University of Singapore (NUS) and Ministry of Education (MOE), Singapore, for research scholarship.

REFERENCES

Abadi, M. *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," Software available from https://www.tensorflow.org/, 2015.

Chollet, F., Keras, 2015, https://keras.io.

Duraisamy, K., Iaccarino, G., and Xiao, H., "Turbulence modeling in the age of data," Annu. Rev. Fluid Mech. **51**, 357–377 (2019).

Guo, X., Li, W., and Iorio, F., "Convolutional neural networks for steady flow approximation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)* (ACM, New York, NY, USA, 2016), pp. 481–490.

Jameson, A., "Aerodynamic design via control theory," J. Sci. Comput. **3**(3), 233–260 (1988).

Jasak, H., Jemcov, A., and Tukovic, Z., "OpenFOAM: A C++ library for complex physics simulations," in *International Workshop on Coupled Methods in Numerical Dynamics* (IUC Dubrovnik Croatia, 2007), Vol. 1000, pp. 1–20.

Jin, X., Cheng, P., Chen, W.-L., and Li, H., "Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder," Phys. Fluids **30**, 047105 (2018).

Kingma, D. P. and Ba, J., "ADAM: A method for stochastic optimization," CoRR, Vol. abs/1412.6, 2014.

LeCun, Y., Bengio, Y., and Hinton, G., "Deep learning," Nature **521**, 436 (2015).

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., "Backpropagation applied to hand written zip code recognition," Neural Comput. **1**(4), 541–551 (1989).

Ling, J. and Templeton, J., "Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty," Phys. Fluids **27**(8), 085103 (2015).

Ma, M., Lu, J., and Tryggvason, G., "Using statistical learning to close two-fluid multiphase flow equations for a simple bubbly system," Phys. Fluids **27**(9), 092101 (2015).

Maulik, R., San, O., Rasheed, A., and Vedula, P., "Data-driven deconvolution for large eddy simulations of Kraichnan turbulence," Phys. Fluids **30**(12), 125109 (2018).

Nielsen, M. A., *Neural Network and Deep Learning* (Determination Press, 2015).

Rosenblatt, F., "The Perceptron, a perceiving and recognizing automaton project para," Report Vol. 85, Nos. 460–461, Cornell Aeronautical Laboratory, 1957.

Ruder, S., "An overview of gradient descent optimization algorithms," [cs.LG], 2017.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., *Neurocomputing: Foundations of Research*, Learning Representations by Back-propagating Errors (MIT Press, Cambridge, MA, USA, 1988), pp. 696–699.

Samareh, J. A., "A survey of shape parameterization techniques," paper presented at the CEAS/AIAA/ICASE/NASA LaRC International Forum on Aeroelasticity and Structural Dynamics, Williamsburg, VA, 22–25 June 1999, pp. 333–343.

Selig, M. S., *UIUC Airfoil Data Site* (Department of Aeronautical and Astronautical Engineering, University of Illinois at Urbana-Champaign, 1996).

Singh, A. P. and Duraisamy, K., "Using field inversion to quantify functional errors in turbulence closures," Phys. Fluids **28**(4), 045110 (2016).

Skinner, S. N. and Zare-Behtash, H., "State-of-the-art in aerodynamic shape optimisation methods," Appl. Soft Comput. **62**, 933–962 (2018).

Song, W. and Keane, A., "A study of shape parameterisation methods for airfoil optimisation," AIAA Paper 2004-4482, 2004.

Wang, Z., Luo, K., Li, D., Tan, J., and Fan, J., "Investigations of data-driven closure for subgrid-scale stress in large-eddy simulation," Phys. Fluids **30**(12), 125101 (2018).

Weller, H. G., Tabor, G., Jasak, H., and Fureby, C., "A tensorial approach to computational continuum mechanics using object-oriented techniques," Comput. Phys. **12**(6), 620 (1998).

Wu, J.-L., Xiao, H., and Paterson, E., "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework," Phys. Rev. Fluids **3**, 074602 (2018).

Zhu, L., Zhang, W., Kou, J., and Liu, Y., "Machine learning methods for turbulence modeling in subsonic flows around airfoils," Phys. Fluids **31**(1), 015105 (2019).