AIAA JOURNAL
Vol. 62, No. 8, August 2024

Neural Network with Local Converging Input for Unstructured-Grid Computational Fluid Dynamics

Weiming Ding,* Haoxiang Huang,† Tzu-Jung Lee,* Yingjie Liu,‡ and Vigor Yang§
Georgia Institute of Technology, Atlanta, GA 30332

https://doi.org/10.2514/1.J063885

In recent years, surrogate models based on deep neural networks have been widely used to solve partial differential equations for fluid flow physics. This kind of model focuses on global interpolation of the training data and thus requires a large network structure. The process is both time consuming and computationally costly. In the present study, we develop a neural network with local converging input (NNLCI) for high-fidelity prediction using unstructured data. The framework uses the local domain of dependence with converging coarse solutions as input, thereby greatly reducing computational resource and training time. As a validation case, the NNLCI method is applied to study two-dimensional inviscid supersonic flows in channels with bumps. Different bump geometries and locations are examined to benchmark the effectiveness and versatility of this new approach. The NNLCI method can accurately and efficiently capture the structure and dynamics of the entire flowfield, including regions with shock discontinuities. For a new bump configuration, the method can perform prediction with only one neural network, eliminating the need for repeated training of multiple networks for different geometries. A saving of computing wall time is achieved by several orders of magnitude against the high-fidelity simulation with the same level of accuracy. The demand on training data is modest, and the training data can be allocated sparsely. These features are especially advantageous compared with conventional global-to-global deep learning methods and physics-informed methods.

I. Introduction

NUMERICAL solutions of partial differential equations are an essential aspect of learning physical phenomena in many natural and engineering science disciplines. The spatio-temporal discretization of the underlying governing equations is often computationally expensive and time-consuming. In recent years, with the rapid development of machine learning techniques, researchers have proposed alternative ways to handle problems more efficiently. Surrogate models have been developed to improve, or even replace in certain circumstances, traditional numerical simulations by mapping between the problem setting and its solution.

Surrogate models can be broadly classed into three categories: data-fit models, reduced-order models (ROMs), and hierarchical (multifidelity) models [1,2]. Data-fit models attempt to approximate an unknown function from the calculated and measured data. They are typically constructed through interpolation or regression, using data points distributed over the entire domain of interest. Examples include response surface models, kriging, radial basis functions, neural network, splines, etc. [3–10]. Because of their nature of global interpolation, the previously mentioned methods often require large training datasets and may encounter challenges for high-dimensional problems. Efforts thus have been made to incorporate physical principles and theoretical formulation in the prediction process to circumvent this obstacle [11–14]. The functional forms of governing equations and constraints such as boundary conditions are imposed by embedding penalty terms in the loss function or modifying the neural network architecture. Examples include the deep Galerkin method, deep Ritz method, and physics-informed neural network (PINN) [15–19]. The use of neural operators further improves the model ability to address a wider range of problems [20–22]. The incorporation of governing equations reduces the demand for data compared to purely data-driven approaches. However, the kind of approaches does not provide complexity reduction over conventional numerical schemes. The computation of the functional forms adds a burden to the training process of the neural network, which may hinder its application to complicated nonlinear problems involving discontinuities and especially interactions among such discontinuities.

Reduced-order models (ROMs) are another kind of surrogate models that can retain the physics and features of the solution to a given problem, while overcoming the computational challenges associated with the curse of dimensionality. ROMs construct a low-dimensional latent space by extracting a representative basis from the high-dimensional full-order model. They can be further divided into two types: intrusive and nonintrusive [23–25]. Intrusive ROMs require knowledge of the underlying full-order model such as governing equations or numerical schemes. Examples include reduced-basis methods, Galerkin projection, and residual minimization principles [26–31]. Nonintrusive ROMs are purely data-driven. For example, proper-orthogonal decomposition (POD) is often used in combination with machine learning tools to construct surrogate models from training datasets. Swischuk et al. studied the performance of POD-based surrogate models using different regression approaches for several engineering problems [32]. Yang and colleagues [33–37] developed POD-based surrogate models for emulating detailed spatio-temporal evolution of turbulent flows calculated from large-eddy simulations. As a demonstration study, the flow dynamics in a swirl injector were examined systematically. To overcome the limitation of linear subspace in POD, dynamic mode decomposition was developed with the Koopman analysis [38,39]. With the advancement of deep learning, autoencoder-based ROMs were attempted to obtain nonlinear manifolds from high-dimensional datasets through an encoder–decoder structure [40,41].

Hierarchical (multifidelity) models use low-fidelity models to predict the results of their high-fidelity counterparts, where the low-fidelity models can be obtained by using coarse grids, lower-order discretization, or simplified physical assumptions [42]. Various types of multifidelity approaches have been developed. The scaling function-based approaches use additive, multiplicative corrections or the combination of both to properly scale a low-fidelity model to its high-fidelity counterpart [43–45]. The space mapping approach attempts to construct conversion functions for the mapping between these two models by means of the Gaussian process or radial basis function [46,47]. Cokriging models extend one-variable kriging to a

Received 19 December 2023; revision received 22 March 2024; accepted for publication 25 April 2024; published online 28 June 2024.

*Doctoral Student, School of Aerospace Engineering.
†Doctoral Student, School of Mechanical Engineering.
‡Professor, School of Mathematics; yingjie@math.gatech.edu (Corresponding Author).
§Regents Professor, School of Aerospace Engineering; vigor.yang@aerospace.gatech.edu (Corresponding Author).

multifidelity context by constructing kriging and a covariance matrix connecting the low- and high-fidelity data [48–50]. Neural networks are also widely applied to obtain high-fidelity results from limited sensor information [51] or low-fidelity data [52].

Although the previously mentioned methods have shown promising results, there exist considerable limitations prohibiting their application to high-fidelity prediction of complex nonlinear problems. Most of existing data-driven surrogate models are based on global computation of the entire field. An effective mapping from a low-fidelity database or a low-dimensional latent space to its high-fidelity counterpart needs to be constructed, which requires a large covariance matrix or neural network structure. Physics-driven models, such as intrusive ROMs and PINNs, alleviate the issues at the cost of additional modifications to the solvers or surrogate models. This leads to extra complexity and burden in the implementation process and downgrades the performance and robustness of the method. To address these issues, surrogate models focusing on local features were explored. Trask et al. developed a convolutional neural network with a generalized moving-least-square method [53]. Scattered data inputs are used to construct local regression functions. The local features and underlying information of the data, however, are not fully used. Kochkov et al. solved the governing equations by incorporating neural network learned interpolation coefficients in a conventional numerical scheme [54]. Although the method preserves fine-scale features of the solution, it could develop a growing deviation toward the coarse-grid simulation as time evolves. Recently, Huang et al. [55,56] proposed a novel method, known as neural networks with local converging inputs (NNLCIs), to efficiently solve conservation laws. The method predicts high-resolution solution at a space–time location from two converging, low-fidelity input solution patches. With the use of local domain of dependence, the method extracts key local features for accurate prediction, while at the same time substantially reducing the demand of computational resource and training data. The NNLCI method has shown great prediction accuracy and efficiency for solving the Euler equations in both one dimension [55] and two dimensions [56] as well as Maxwell's equations [57].

The application of the NNLCI method with structured data is relatively easy to implement, due to the numerical-grid consistency and uniformity in the local domain of dependence. Many scientific and engineering problems, however, involve complex geometries and unstructured data with irregular grids. Extension of the NNLCI method to such situations is important. In the present work, we extend the NNLCI method to accommodate unstructured data. As a validation case, inviscid flow through a converging-diverging channel with two smooth Gaussian (or triangular) bumps is studied systematically. The new method is capable of capturing the flow behaviors in the entire field, including regions with smooth variations and steep gradients such as shock discontinuities.

This paper is structured as follows. Section II describes the theoretical formulation and numerical scheme for inviscid channel flows. The data generation and preprocessing steps are also discussed. Section III introduces the neural network framework with local converging inputs (NNLCI). The method for determining the local domain of dependence is developed for unstructured grids. Section IV presents the results of the proposed method. The effectiveness of the new approach is demonstrated by a variety of channel geometries. In Sec. V, conclusions are reported.

II. Theoretical and Numerical Framework

A. Problem Setup

In this study, we consider a 2D inviscid flow through a channel with two smooth Gaussian bumps, as shown schematically in Fig. 1. The computational domain is bounded by $x \in [-1.5, 1.5]$ in the axial direction and $y \in [0, 0.8]$ in the vertical direction. Two Gaussian bumps are placed on the top and bottom walls of the channel. The lower bump geometry is fixed and defined as

$$y = 0.0625e^{-25x^2} \tag{1}$$

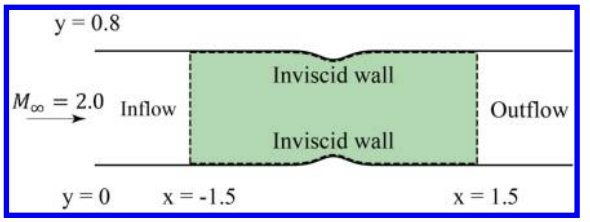

Fig. 1 Inviscid flow through a channel with two smooth Gaussian bumps.

The upper bump is perturbed from the original location and is defined as

$$y = 0.08 - 0.0625e^{-25(x-\Delta x)^2} \tag{2}$$

where Δx stands for the perturbation from the baseline geometry.

The problem is governed by the 2D Euler equations for compressible flows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \tag{3}$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2 + p)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} = 0 \tag{4}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho vu)}{\partial x} + \frac{\partial(\rho v^2 + p)}{\partial y} = 0 \tag{5}$$

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial(\rho uH)}{\partial x} + \frac{\partial(\rho vH)}{\partial y} = 0 \tag{6}$$

where ρ, u, v, p and E are the density, x velocity, y velocity, pressure, and total energy, respectively. $H = E + p/\rho$ is the total enthalpy. Calorically perfect gas is assumed. The resulting equation of state takes the form

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho \|\boldsymbol{v}\|_2^2\right) \tag{7}$$

The ratio of specific heats γ is taken as 1.4. Inviscid wall boundary conditions are applied at the top and bottom walls. The total temperature $T_{t,\infty}$ and total pressure $p_{t,\infty}$ of the inflow are

$$\frac{T_{t,\infty}}{T_\infty} = 1 + \frac{\gamma - 1}{2}M_\infty^2 \tag{8}$$

$$\frac{p_{t,\infty}}{p_\infty} = \left(\frac{T_{t,\infty}}{T_\infty}\right)^{\gamma/(\gamma-1)} \tag{9}$$

where the subscript ∞ stands for the inflow condition. In the present study, the inlet Mach number is set at $M_\infty = 2.0$.

A finite-volume solver is implemented for solving the theoretical formulation by means of the MUSCL scheme [58] along with the Rusanov flux [59]. An improved second-order Runge–Kutta scheme with the total-variation-diminishing feature [60] is implemented for temporal evolution. In the end, a steady state solution can be obtained.

B. Data Generation and Preprocessing

To generate the training and testing data sets, a variety of bump locations are considered. The upper bump location is translated in the x direction with $\Delta x = 0.00, \pm 0.15, \pm 0.3, \pm 0.45$ and ± 0.60 in Eq. (2). Similarly, for the testing data set, the upper bump location is randomly perturbed within the range of $[-0.60, 0.60]$ In addition, the inflow Mach number is perturbed by $\pm 5\%$ for the training cases. This ensures that the training database contains all the features of highly nonlinear flow behaviors and increases the robustness of the

Table 1 Training and testing cases: inflow Mach number M_∞ perturbed by ±5% > for each bump location

Dataset	Description	M_∞ perturbation
Training	$\Delta x = 0.00, \pm 0.15, \pm 0.30, \pm 0.45, \pm 0.6$	$\Delta M_\infty = 0, \pm 5\%$
Testing	$\Delta x = -0.350, -0.225, 0.120, 0.525$	None

network. Table 1 shows the details of the training and testing data sets.

For each geometry setting, the flowfield is calculated on unstructured triangular grids with different resolutions: coarse, finer, and high resolution. The finer and high-resolution meshes are constructed by implementing uniform mesh adaptation from the coarse grid. Figures 2 and 3 show the calculated density and Mach-number fields, respectively, for the upper bump translation at $\Delta x = -0.60, -0.30, 0.00, 0.30$, and 0.60 The coarse, finer, high, and extra-high-fidelity simulation results are presented from left to right. The corresponding numbers of numerical cells are 800, 3200, 51,200 and 204,800, respectively. The flow structure changes with the translation of the upper bump. The high-fidelity simulation results exhibit much richer details in the shock intersection and flow expansion regions. In the present work, the results with the coarse- and finer-grid resolutions will be used as the input to the neural network, and the high-fidelity result as the validation benchmark, because it captures all the important flow physics revealed by extra-high-fidelity simulations.

To fully use the low-fidelity information and extract common flow features from different bump translation cases, preprocessing is performed on the coarse- and finer-grid data. Figure 4 shows an example of the data selection process. In each cell of the coarse grids, several points (red dots) are selected as the training locations. The data at the corresponding locations in the finer- and high-resolution grids are extracted to form the input-image pairs for the training dataset. Approximately 10,000 points are used in each training case. This avoids computationally expensive interpolation of a large dataset and accelerates the training process.

For each data point, all four state variables $\boldsymbol{u} = [\rho, \rho u, \rho v, \rho E]$ are used. Normalization and standardization are implemented to facilitate the training of the neural network. In the present study, the data are rescaled by the difference between the maximum and minimum values:

$$\tilde{\boldsymbol{u}} = \frac{\boldsymbol{u} - \min(\boldsymbol{u})}{\max(\boldsymbol{u}) - \min(\boldsymbol{u})} \tag{10}$$

The tanh function is used as the activation function in the neural network:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{11}$$

III. Neural Network with Local Converging Inputs

In this section, the neural network with local converging inputs (NNLCI) for unstructured data is constructed. Figure 5 provides an overview of the standard NNLCI solution procedure. First, we obtain the solutions from the simulations on the coarse and finer grids. The local domain of dependence is then determined for each data location (i, j) denoted by the red dot. A 3×3 patch $[i-1, i+1] \times [j-1, j+1]$ is formed around the desired location (red point). The intention is to filter the data at a proper scale to include all the local features for accurate prediction, while discarding far-end information for low training costs. Then, a neural network is used to map the solutions of the two low-fidelity patches to their high-fidelity counterpart at the center location (i, j). The process is repeated across the whole domain to achieve high-fidelity prediction for the entire field.

The previously mentioned procedure is straightforward on structured grid; however, it becomes nontrivial when applied to unstructured grid. To address the issue, we construct similar patches around the desired location based on the local cell size of the unstructured grid. Figure 6 gives two examples of such a process. We form a

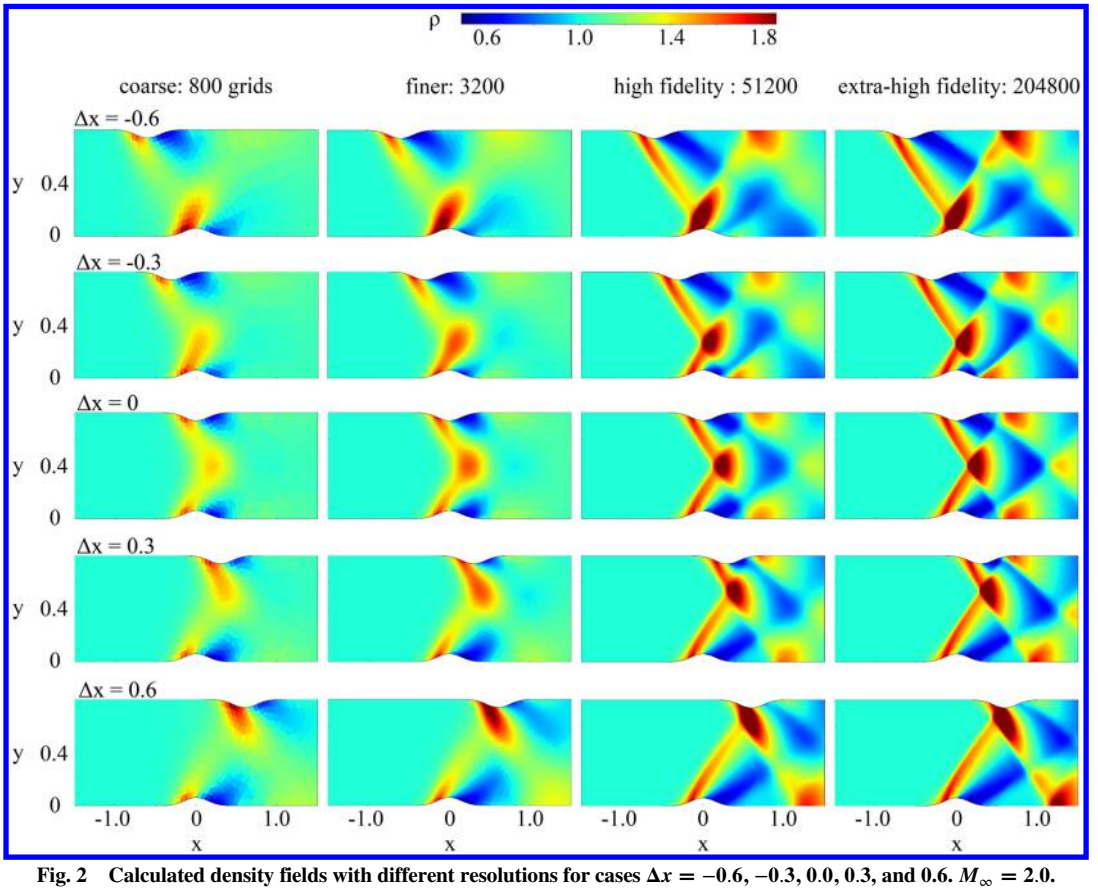

Fig. 2 Calculated density fields with different resolutions for cases $\Delta x = -0.6, -0.3, 0.0, 0.3$, and 0.6. $M_\infty = 2.0$.

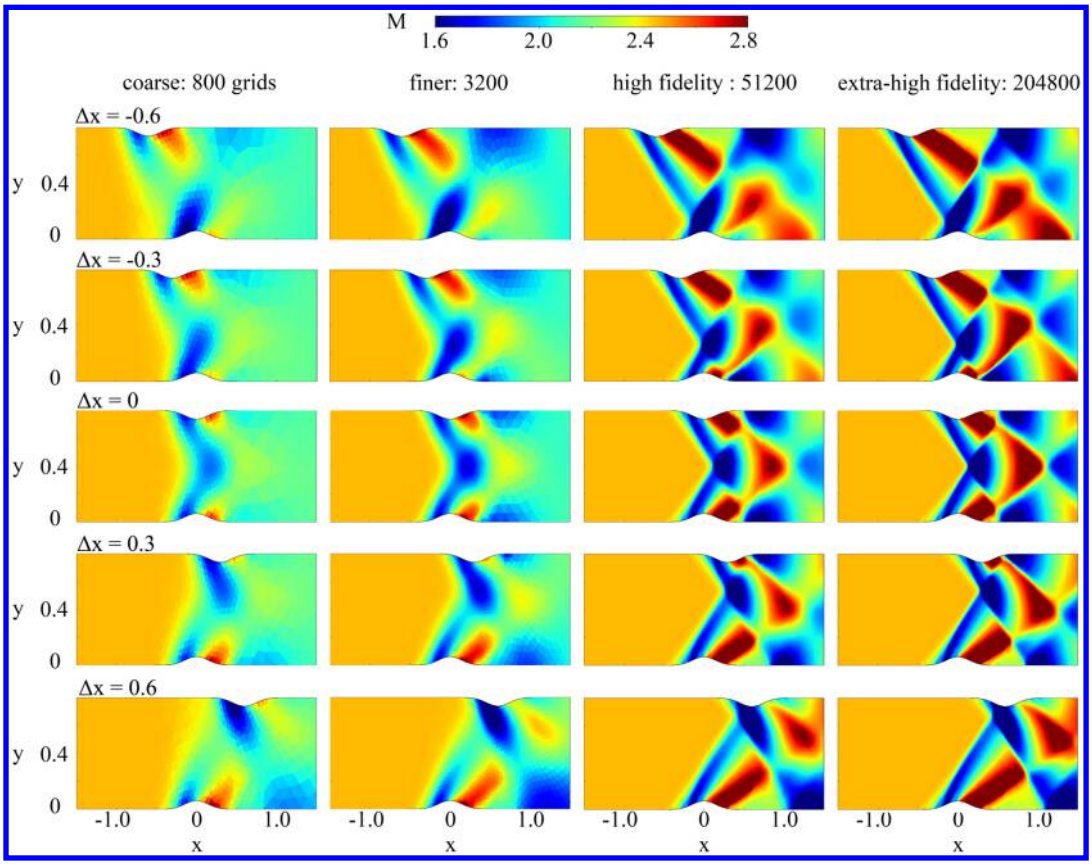

Fig. 3 Calculated Mach-number fields with different resolutions for cases $\Delta x = -0.6, -0.3, 0.0, 0.3$, and 0.6. $M_\infty = 2.0$.

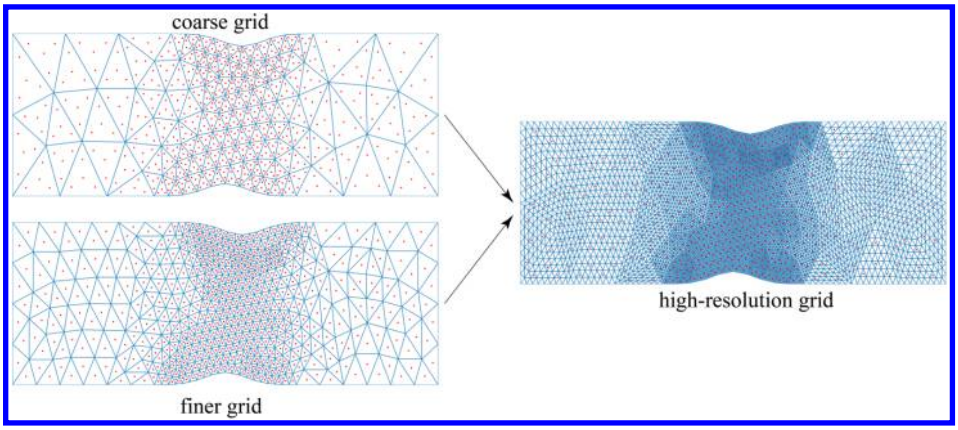

Fig. 4 Example of unstructured NNLCI training dataset. The coarse- and finer-resolution data are used as the inputs to the neural network, and the high-resolution data is used for training.

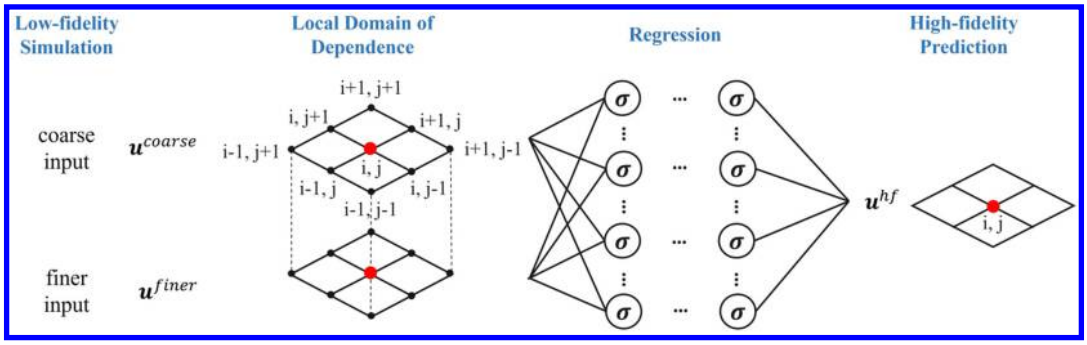

Fig. 5 Overview of the standard NNLCI structure. Four major steps are involved: multifidelity simulation, local domain of dependence, neural network regression, and high-fidelity prediction.

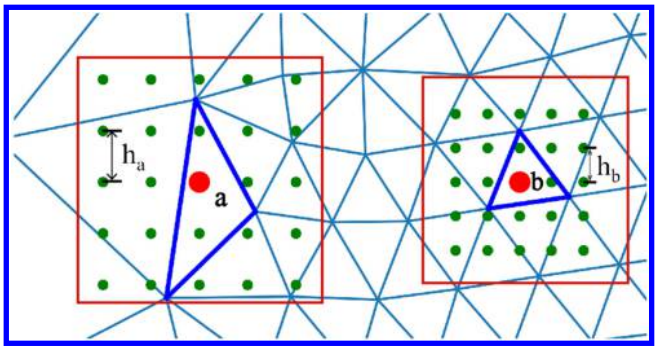

Fig. 6 Examples of 5×5 points for NNLCI local domain of dependence. Red dots: selected locations; green dots: local domain of dependence; blue lines: corresponding cells.

5×5 rectangular stencil in $[x - 2h_{\text{coarse}}, x + 2h_{\text{coarse}}] \times [y - 2h_{\text{coarse}}, y + 2h_{\text{coarse}}]$ around the selected location (red dots) based on the coarse grid [where h_{coarse} is the local cell size of the coarse grid around (x, y)] as the local domain of dependence (local stencil), shown by green dots. The size of the local domain of dependence varies with the local cell size. Some of the data points near the boundaries are discarded in this process, because the selected stencil points are out of bound.

To determine the local cell size, the information of the current and adjacent cells needs to be extracted. For each selected location on a given grid, the corresponding cells E (blue lines) can be found efficiently by adopting a hierarchical data structure described in Ref. [61]. The computational domain is divided into a binary tree of blocks of cells based on the cell centroid locations. By comparing the data location (x, y) with the cell centroid, one can descend from the root to subblocks and find the cell E containing it. This greatly reduces the computational cost and search time. The adjacent cells E_{adj} that share edges or vertices with E are identified based on the connectivity information, and the local cell size can be calculated. Here, the local cell size h_E of a triangular cell E is defined as the average cell size of itself and its adjacent cells:

$$h_E = \frac{1}{N} \sum_{k \in E_{\text{adj}}} \sqrt{A_k} \tag{12}$$

where A_k is the area of cell k, E_{adj} represents the cells adjacent to cell E, and N is the total number of A_k in the summation.

Once a stencil is determined, the values of state variables at the 5×5 locations determined through interpolation from the coarse and finer meshes are used as the input to the neural network. Also included in the input are h_{coarse} and h_{finer}, the local cell size at (x, y) for the coarse and finer meshes. These two values are critical for the neural network to approximate nonuniform properties, given the states interpolated from the coarse and finer meshes. The output is the predicted state variables at (x, y). The procedure for determining the state variables follows. Let $\boldsymbol{x_s} = (x_s, y_s)$ be one of the 5×5 points in a particular stencil. We first locate the corresponding cell E_0 that contains this point using the hierarchical approach described previously. The center of cell E_0 is denoted as $\boldsymbol{x}_0 = (x_0, y_0)$, and the corresponding state variable is $\boldsymbol{u}_0$. Then, the three adjacent triangles E_1, E_2, and E_3 are located, with the cell centers (x_i, y_i) and state variables, $\boldsymbol{u}_i$, where $i = 1, 2$, and 3. The state at the stencil point location (x_s, y_s) can be interpolated with a linear polynomial:

$$\boldsymbol{u}(\boldsymbol{x_s} - \boldsymbol{x}_0) = \boldsymbol{a}_0 + \boldsymbol{a}_1(x_s - x_0) + \boldsymbol{a}_2(y_s - y_0) \tag{13}$$

It is easily seen that $\boldsymbol{a}_0 = \boldsymbol{u}_0$. To solve for the coefficients $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, three combinations of cells can be used: (E_0, E_1, E_2), (E_0, E_2, E_3), and (E_0, E_1, E_3). Each set of cells, say set k, will determine a set of candidate values for $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, say $\boldsymbol{a}_{1_k}$ and $\boldsymbol{a}_{2_k}$. The minmod function is then applied to determine the best coefficient values:

$$\boldsymbol{a}_1 = \min \text{mod}(\boldsymbol{a}_{1_k}, k = 1, 2, 3) \tag{14}$$

$$\boldsymbol{a}_2 = \min \text{mod}(\boldsymbol{a}_{2_k}, k = 1, 2, 3) \tag{15}$$

where the minmod function is acting componentwisely and is defined as

$$\min \text{mod}(a_1, a_2, \ldots, a_n) = \begin{cases} \min(a_1, a_2, \ldots, a_n), & \text{if all } a_i > 0 \\ \max(a_1, a_2, \ldots, a_n), & \text{if all } a_i < 0 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

The procedure is repeated on the coarse, finer, and high-fidelity solutions to generate the input and reference values for the neural network. The local cell sizes are used as inputs to the neural network to include the local mesh size information, which gives us an input size of 202, including the four state variables at the 5×5 locations and the two local cell sizes. The output size is 4. The input–output pair is shown in Fig. 7.

Table 2 lists the hyperparameters of the selected neural network, determined from manual search, in the present study. A network of 10 hidden layers of 600 nodes is designed to learn the mapping from low-fidelity solutions in a local domain to the high-fidelity solution at a point. The network is trained using the Adam optimizer with a learning rate of 1×10^{-4} with 1×10^{-8} regularization. The tanh function is used as the activation function. The relative mean squared error (RMSE) is selected as the loss function to measure the difference between the NNLCI prediction and the benchmark high-fidelity data:

$$\mathcal{L} = \frac{\sum_k \|\tilde{\boldsymbol{u}}_{\boldsymbol{k}} - \boldsymbol{u}_{\boldsymbol{k}}\|_2^2}{\sum_k \|\boldsymbol{u}_{\boldsymbol{k}}\|_2^2} \tag{17}$$

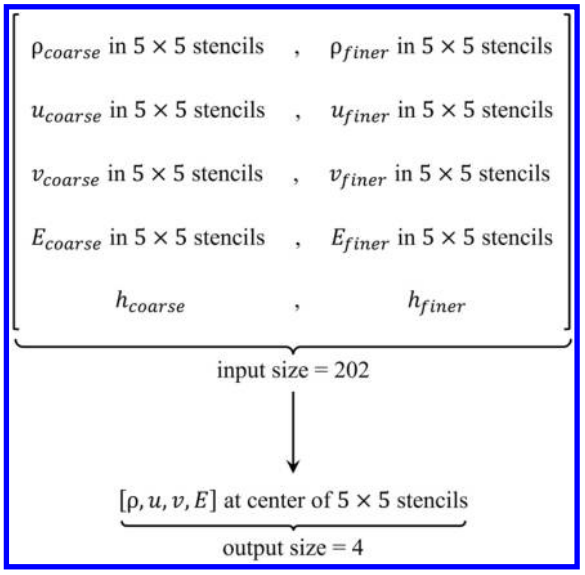

Fig. 7 Input–output pair of unstructured NNLCI. The input contains the state variables on 5×5 stencil points and the local mesh sizes of coarse and finer grids. The output is the state variable value at the cell center.

Table 2 Hyperparameters of the neural network with local converging input

	Hyperparameters
Number of epochs	50000
Number of hidden layers	10
Network structure	$[202, 10 \times 600, 4]$
Learning rate	1×10^{-4}
L_2 regularization	1×10^{-8}
Activation function	tanh
Optimizer	Adam

where $\tilde{\boldsymbol{u}}_k$ represents the predicted results from the NNLCI method and $\boldsymbol{u}_k$ the data from high-fidelity simulation. For unstructured data, the RMSE needs to be weighted by the cell area. The cell-weighted RMSE is given by

$$\mathcal{L} = \frac{\sum_k A_k \|\tilde{\boldsymbol{u}}_k - \boldsymbol{u}_k\|_2^2}{\sum_k A_k \|\boldsymbol{u}_k\|_2^2} \tag{18}$$

IV. Results and Discussion

The NNLCI method is applied to predict the flowfields with several different bump geometries on unstructured grids. The coarse and finer solutions of each case are used as the inputs to the neural network. Unlike in the training cases, we select the cell centroids of the high-resolution grids as the prediction locations to enhance the resolution of final prediction results. Figure 8 shows an example of the input-image pairs. The red dots denote the selected locations for prediction. For each case, nearly 50,000 points are selected for prediction. The interpolation technique described in Sec. III is implemented to obtain the data for these locations on both the coarse and finer grids.

First, we present the NNLCI prediction for bump translation cases, as listed in Table 1. Figure 9 shows the NNLCI predicted Mach-number fields for the upper bump translations of $\Delta x = -0.35$, $-0.225, 0.12$, and 0.525 High-fidelity simulation results are also presented for comparison. The flowfields exhibit distinct behaviors and features, depending on the bump translation. In the case of $\Delta x = -0.35$ the upper shock develops upstream and intersects with the lower shock near the lower bump. Consequently, the flow expansion area is shifted toward the lower wall. The secondary shock is well-developed downstream of the upper bump, whereas it can hardly be observed near the lower bump. The case of $\Delta x = 0.525$ has opposite behavior. In the cases of $\Delta x = 0.12$ and -0.225, a secondary shock is observed originating from both the upper and lower bumps, with shock intersection near the center of the channel. Despite such complex flow features, the NNLCI method achieves promising results. It accurately captures the primary shock location and structure for all cases. The expansion region and secondary shock are also well-reconstructed.

Table 3 summarizes the prediction errors of the four cases. Here, the area-weighted relative L_1 norm, $\mathcal{L}_1$,

$$\mathcal{L}_1 = \frac{\sum_k A_k \|\tilde{\boldsymbol{u}}_k - \boldsymbol{u}_k\|_1}{\sum_k A_k \|\boldsymbol{u}_k\|_1} \tag{19}$$

and the area-weighted relative root-mean-square error (RRMSE), $\mathcal{L}$

$$\mathcal{L} = \sqrt{\frac{\sum_k A_k \|\tilde{\boldsymbol{u}}_k - \boldsymbol{u}_k\|_2^2}{\sum_k A_k \|\boldsymbol{u}_k\|_2^2}} \tag{20}$$

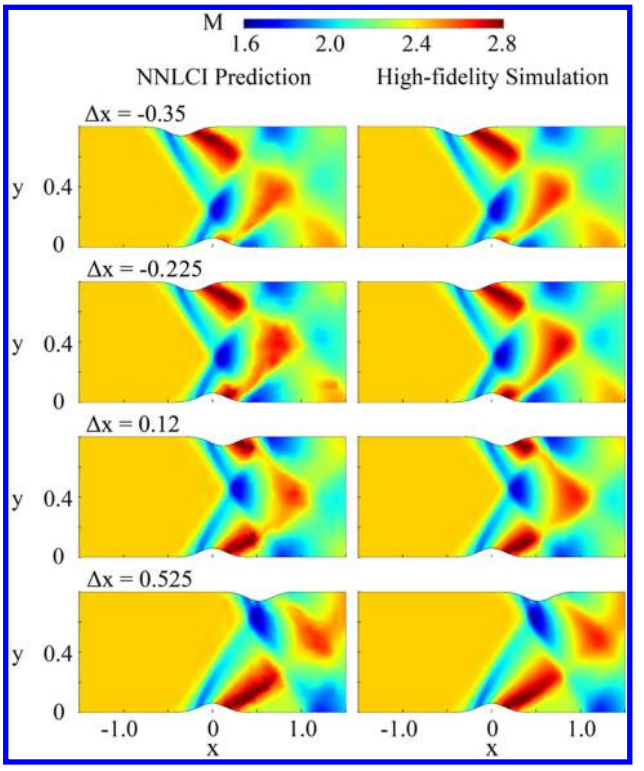

Fig. 9 Mach-number fields of NNLCI prediction (left) and high-fidelity simulation (right). Upper bump translations are $\Delta x = -0.35, -0.225, 0.12$, and 0.525, respectively.

are used to measure the performance of the NNLCI prediction. The relative L_1 error of the low-fidelity simulation is also included for comparison.

The NNLCI method achieves a relative L_1 error of less than 1%, in comparison with the low-fidelity simulation error of 7.2%. The prediction accuracy is improved by about 10 times. In particular, the resolution in regions with shock discontinuities is substantially improved (see Figs. 2 and 3). Figure 10 shows the Mach-number distribution along the centerline of the channel, $y = 0.4$. The NNLCI prediction (red line) agrees well with the high-fidelity simulation result (51,200 cells, green line). The lower-fidelity simulation result (3200 cells, blue line) used as the input in the NNLCI method is also presented for reference. The flow development and shock structure are accurately captured by the NNLCI prediction for all the test cases. These results testify to the effectiveness of the NNLCI method for predicting complex flows, including regions with smooth evolution and shock discontinuities.

The NNLCI method allows for simultaneous predictions of all the state variables. Figures 11 and 12 show the results of the density and

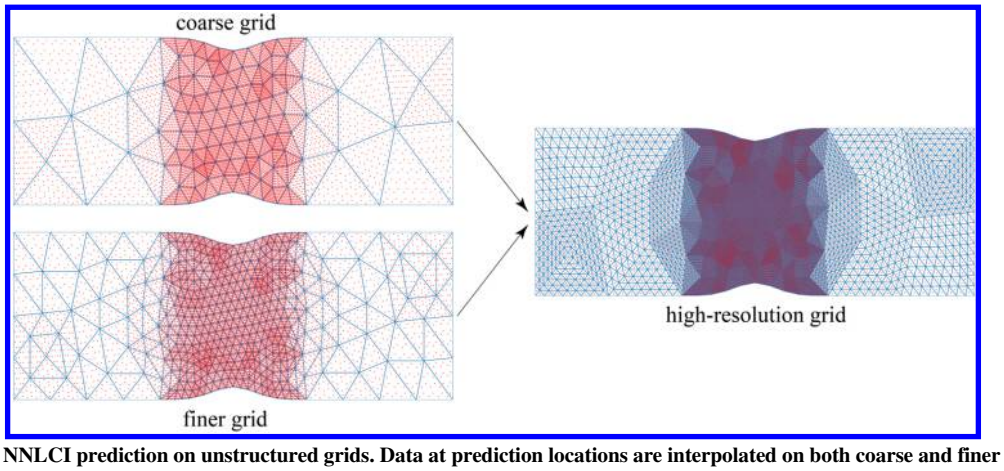

Fig. 8 NNLCI prediction on unstructured grids. Data at prediction locations are interpolated on both coarse and finer grids.

Table 3 Relative L_1 norm and relative root-mean-square error of NNLCI predictions

Bump translation Δx	NNLCI relative L_1 norm	NNLCI RRMSE	Low-fidelity relative L_1 norm
−0.35	0.734%	1.124%	7.538%
−0.225	0.979%	1.597%	7.742%
0.12	0.818%	1.492%	7.137%
0.525	0.654%	1.149%	6.347%
Total	0.797%	1.358%	7.211%

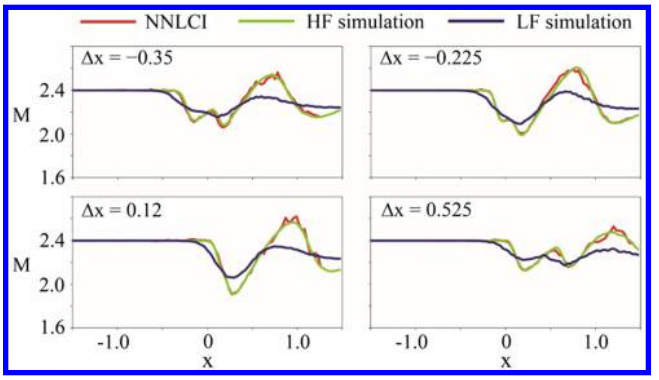

Fig. 10 Mach-number distribution along the centerline of the channel $y = 0.4$.

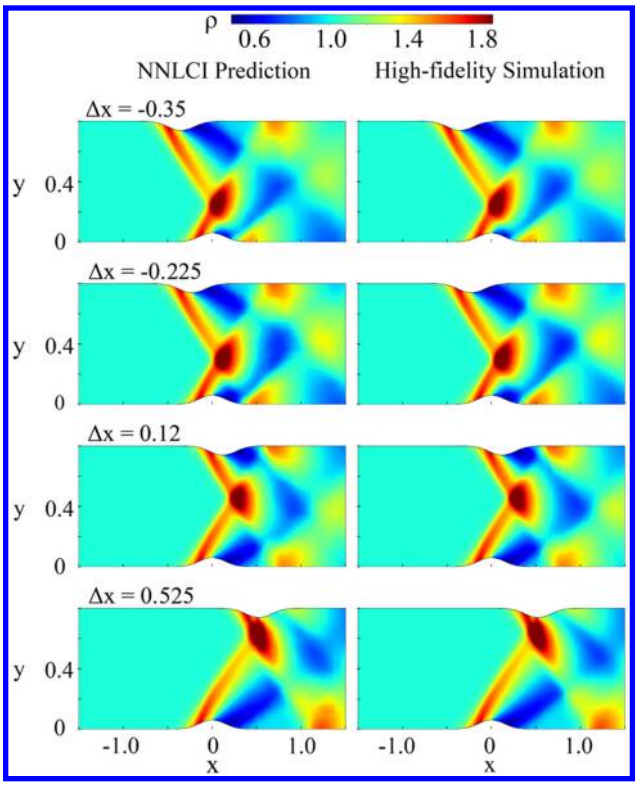

Fig. 11 Density fields: NNLCI prediction (left) and high-fidelity simulation (right). Upper bump translations are $\Delta x = -0.35$, -0.225, 0.12, and 0.525, respectively.

pressure fields, respectively. The NNLCI method accurately predicts the structure and magnitude of the entire field, including both shock discontinuities and smooth regions. It is worth noting that the pressure and density predictions achieve better accuracy than their Mach-number counterpart. The latter is a derived variable, and

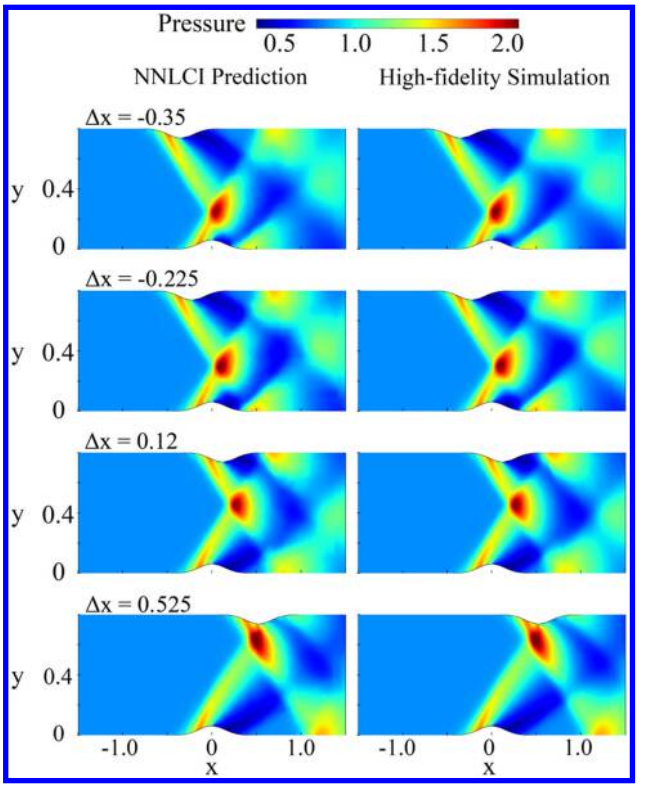

Fig. 12 Pressure fields: NNLCI prediction (left) and high-fidelity simulation (right). Upper bump translations are $\Delta x = -0.35$, -0.225, 0.12, and 0.525, respectively.

its calculation involves manipulations of primary variables, which easily accumulates errors. For a new bump configuration, the NNLCI method can perform prediction with only one neural network and thus eliminates repeated training of multiple networks for different geometric variables. In the present study, the wall time for low- and high-fidelity simulation of each design setting takes 1 min and 2 h respectively, on a single CPU (Intel Core i7-10750H). In comparison, the NNLCI predicts a new case in less than 1 s on the same hardware. A time saving of more than two orders of magnitude is achieved between the NNLCI prediction and high-fidelity simulation.

In general, the NNLCI method works like a local scanner that scans two coarse-grid numerical solutions (with one more accurate than the other) at a given location and then predicts the high-fidelity solution at the scanned location. Compared to other neural network methods that map global information onto high-fidelity solution, the localness of NNLCI is fundamental to its efficiency and leads to many distinct features. For example, the high-fidelity solution to be predicted could be quite different from the high-fidelity solutions used in training. This issue, however, can be properly handled by the NNLCI method, in which the similarity of local patches of the solution in the input is much greater than that among the global solution. It is thus possible to use considerably fewer high-fidelity solutions for training and allocate these high-fidelity solutions more sparsely (thereby allowing for a larger domain of interest). In fact, NNLCI's demand for training data is modest because each high-fidelity solution used in training provides a vast number of local samples for training, and a small number of high-fidelity solutions are usually adequate for training. The local input patches taken from a converging sequence of numerical solutions provide information sufficient for the neural network to accurately capture the local flow features, whether or not in regions of smooth evolution or steep gradients.

It remains a challenge for those neural network methods that are based on minimization of the residue of the governing equations (e.g., [15–19]) to predict solutions containing discontinuities or steep variations. The situation is further exacerbated for problems involving interactions of discontinuities. Moreover, these methods do not provide complexity reduction over conventional numerical schemes.

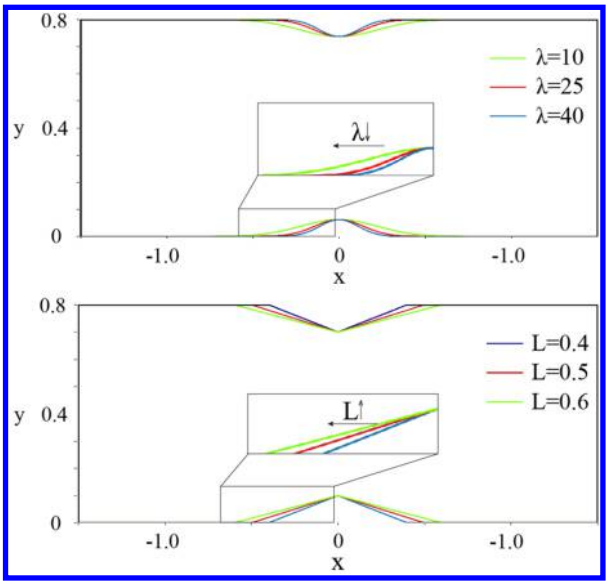

Fig. 13 Gaussian and triangular bumps with different geometric parameters λ and L.

Because the NNLCI method only requires the neural network to map local patches of coarse solutions onto the high-fidelity solution at a local point, the computational burden on the neural network is modest. A relatively small and standard neural network is often adequate. Also, due to its localness, the NNLCI method is convenient to implement in complicated computational domains for real applications. The training can be achieved in a set of varying computational domains, and the prediction can be performed in different domains. This capability is most suitable for design optimization, which requires the survey of a large parameter space.

To further evaluate the performance of the NNLCI method, two different bump geometries, a triangular bump and a Gaussian bump with changing variance, as shown in Fig. 13, are investigated. The shape of the Gaussian bump is varied by tuning the variance λ in the shape function, whereas the height remains fixed:

$$y = 0.0625e^{-\lambda x^2} \tag{21}$$

For the triangular wedge, the wedge height h is fixed as 0.1, whereas the length L varies in the range of [0.4, 0.6].

Figures 14 and 15 show the calculated density and Mach-number fields for the two different types of bump shapes with different resolutions. The geometric parameters λ or L dictate the incident flow angle at the bump leading edge, resulting in different flow structures and shock dynamics. As the wedge angle increases, the primary oblique shock angle becomes larger, and the postshock subsonic region is compressed, resulting in a weaker secondary shock due to limited expansion. Such change of flow behavior greatly increases the difficulty of prediction. In the present study, instead of constructing a separate neural network, the already trained network is further improved with supplementary data. The resultant NNLCI is expected to predict the flowfield for all bump geometries with high fidelity. Table 4 summarizes the training and validation cases. For each bump shape, the free-stream Mach number M_∞ is perturbed by $\pm 5\%$. For the validation cases of both bump shapes, the validation

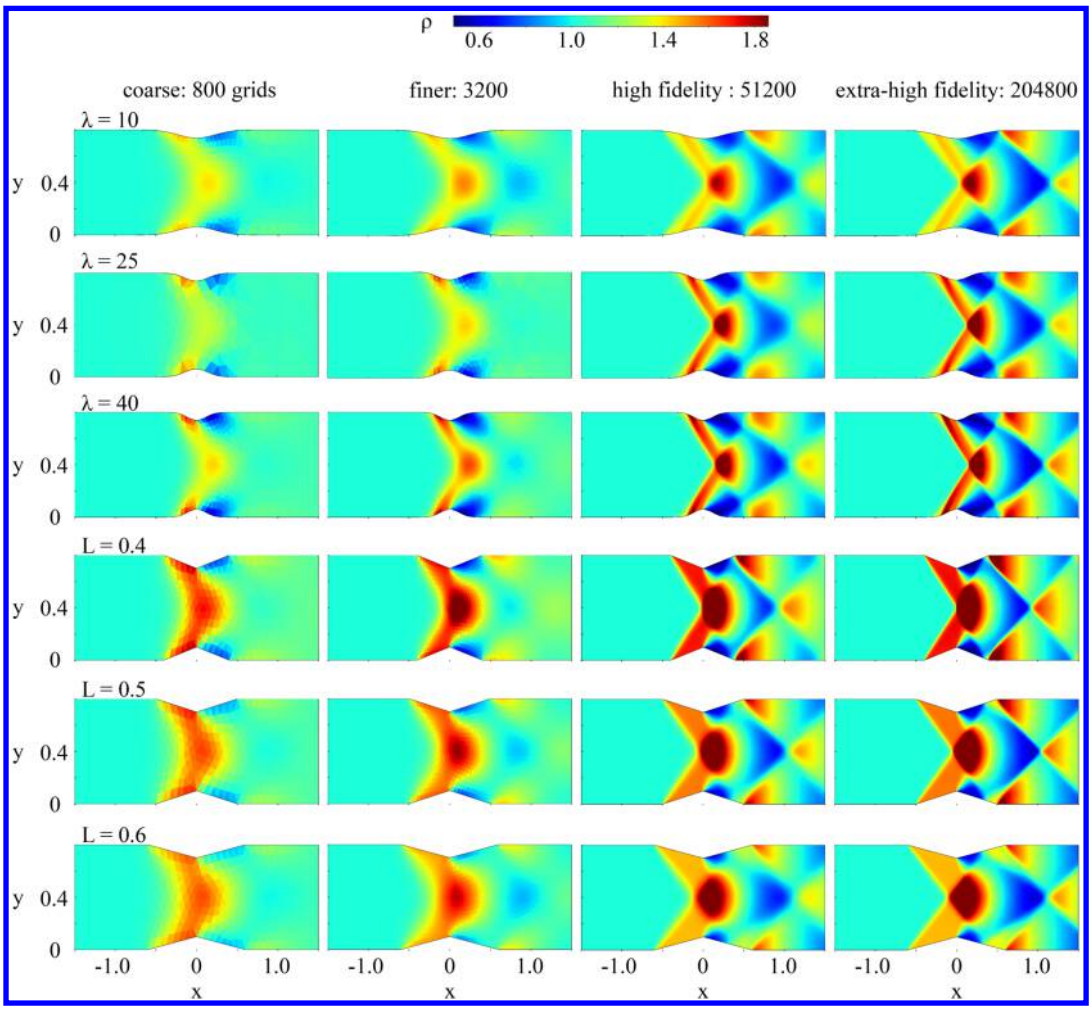

Fig. 14 Calculated density fields with different resolutions for Gaussian bumps with $\lambda = 10$, 25, and 40, respectively, and triangular bumps with $L = 0.4$, 0.5, and 0.6, respectively. $M_\infty = 2.0$.

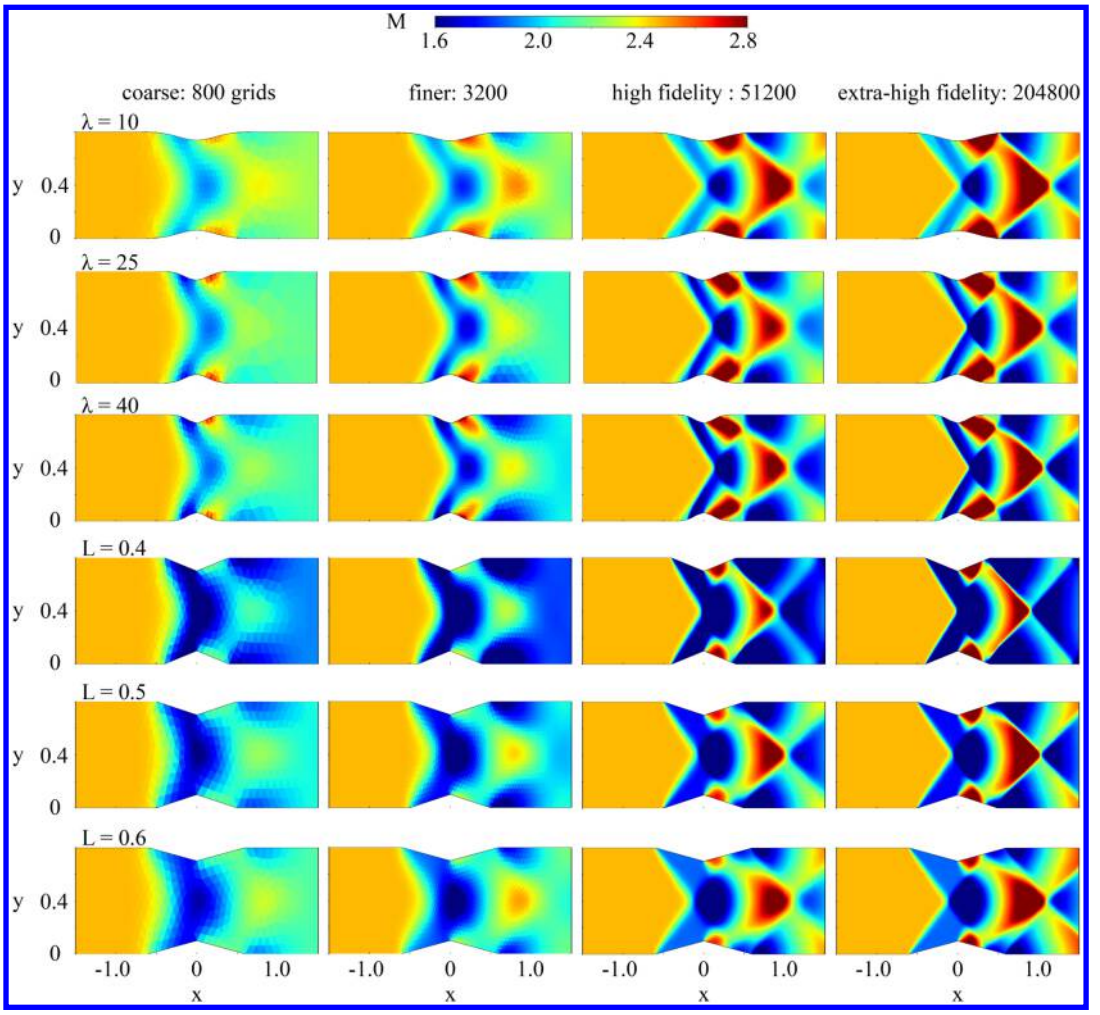

Fig. 15 Calculated Mach-number fields with different resolution for Gaussian bumps with $\lambda = 10$, 25, and 40, respectively, and triangular bumps with $L = 0.4$, 0.5, and 0.6, respectively. $M_\infty = 2.0$.

Table 4 Training and validation cases for Gaussian and triangular bumps

Dataset	Bump type	Parameters	M_∞ perturbation
Training	Gaussian	$\lambda = 10, 25, 40$	$\Delta M_\infty = 0, \pm 5\%$
	triangular	$L = 0.4, 0.5, 0.6$	
Validation	Gaussian	$\lambda = 17.5, 45$	None
	triangular	$L = 0.55, 0.65$	

points within ($\lambda = 17.5$, $L = 0.55$) and outside of ($\lambda = 45$, $L = 0.65$) the training range are tested to demonstrate the generalization capability of the NNLCI method.

Figure 16 shows the Mach-number fields obtained from the NNLCI predictions and high-resolution simulations. The NNLCI method accurately predicts the flow behaviors for all cases, including those outside of the training range.

Table 5 summarizes the NNLCI prediction errors. An accuracy of more than 99% is achieved. As noted previously, only one neural network is built and trained for all the cases. This eliminates the need for construction and training of new neural networks for different geometries. When predicting a new case, the existing neural network can be trained with additional data and obtain accurate results in a short time. The NNCLI approach significantly reduces the turnaround time for a new case. With extra training cases for new bump shapes, the NNLCI method can further improve the prediction accuracy. Overall, an order-of-magnitude improvement of accuracy is obtained against low-fidelity simulation results.

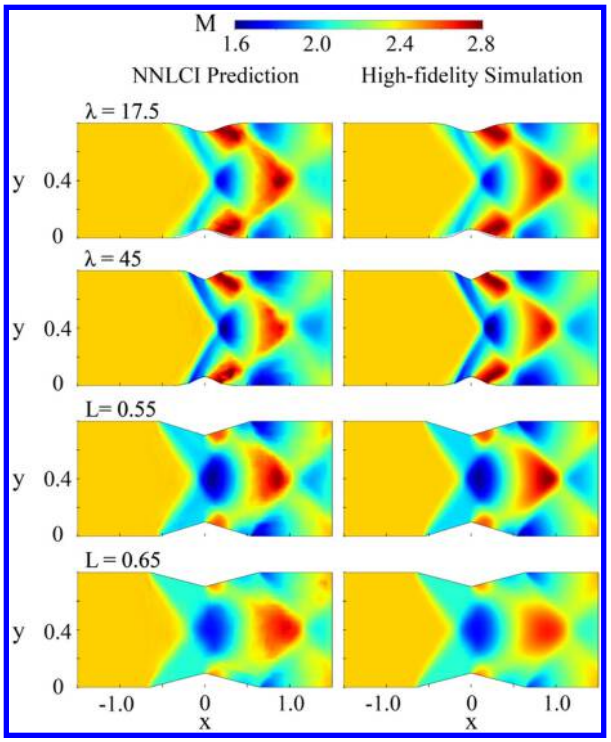

Fig. 16 Mach-number fields calculated by the NNLCI method (left) and high-fidelity simulation (right) for Gaussian bumps with $\lambda = 17.5$ and 45 and triangular bumps with $L = 0.55$ and 0.65.

Table 5 Relative L_1 norm and relative root-mean-square errors for various bump shapes

Bump shape	Relative L_1 norm	RRMSE	Low-fidelity relative L_1 norm
$\lambda = 17.5$	0.923%	1.812%	7.179%
$\lambda = 45$	0.873%	1.455%	8.612%
$L = 0.55$	0.887%	1.413%	7.865%
$L = 0.65$	1.149%	1.768%	6.135%
Bump translation	0.755%	1.261%	7.211%
Total	0.862%	1.466%	7.356%

Figure 17 shows the distributions of Mach number along the center of the channel for the Gaussian and triangular bumps. The NNLCI method accurately predicts the shock location and amplitude, despite the disparity of flow behaviors between the two cases. Figures 18 and 19 show the calculated density and pressure fields, respectively. Good agreement is achieved between the NNLCI predictions and high-fidelity simulations.

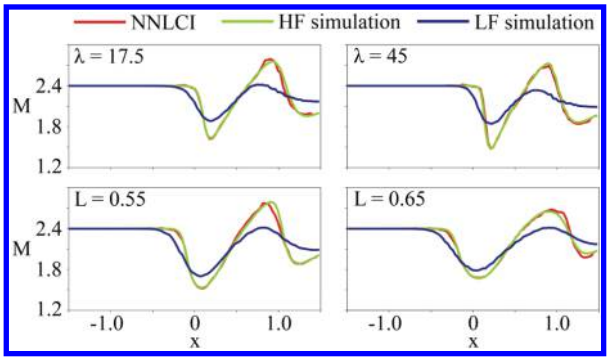

Fig. 17 Mach-number distribution along the center of the channel $y = 0.4$ for Gaussian bump with $\lambda = 17.5$ and 45 (top) and triangular bump with $L = 0.55$ and 0.65 (bottom).

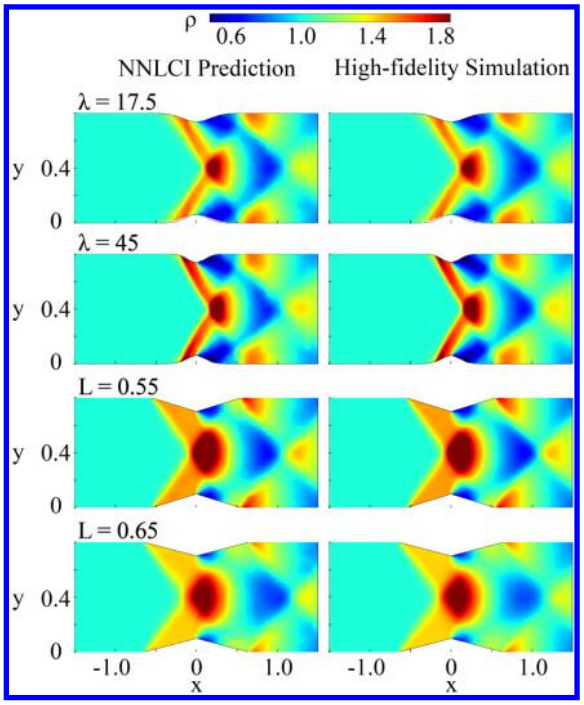

Fig. 18 Density fields: NNLCI prediction (left) and high-fidelity simulation (right) for Gaussian bump with $\lambda = 17.5$ and 45 and triangular bump with $L = 0.55$ and 0.65.

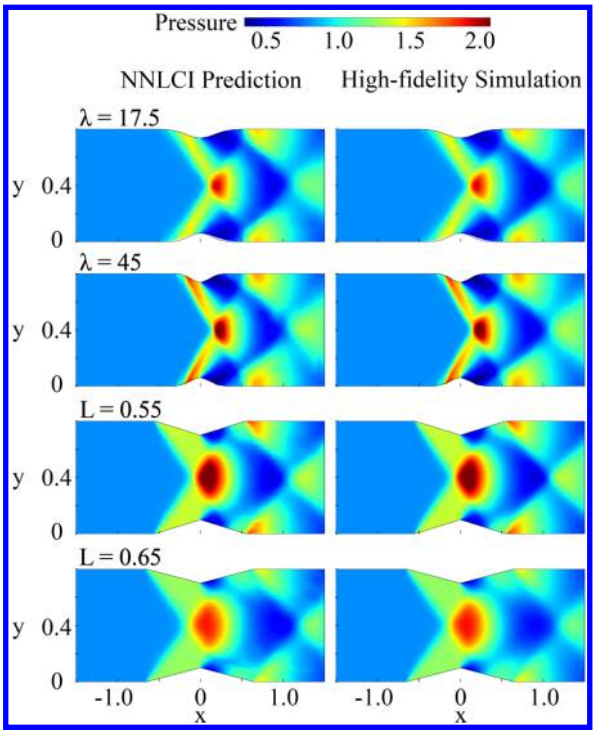

Fig. 19 Pressure fields: NNLCI prediction (left) and high-fidelity simulation (right) for Gaussian bump with $\lambda = 17.5$ and 45 and triangular bump with $L = 0.55$ and 0.65.

V. Conclusion

This paper presents a NNLCI for predicting fluid flow dynamics with unstructured data. The work provides an effective means for treating problems with complex physics and geometries. The approach employs an effective sampling-and-interpolation technique to construct local converging input from low-resolution simulation results. The neural network builds up the regression from local converging inputs to high-fidelity prediction results. As part of the validation effort, the NNLCI method for unstructured data is applied to predict two-dimensional supersonic inviscid flows in a channel with two different types of bumps on the walls. The bump geometry is varied to create distinct flow structures and shock dynamics. Excellent agreement between the NNLCI predictions and high-fidelity simulations is achieved. The NNLCI method allows for simultaneous predictions of all the flow variables and their gradients over the entire flowfield, including regions with shock discontinuities. Furthermore, the NNLCI method can effectively treat different bump geometries with only one neural network. It can learn flow features directly from the training data and produce accurate predictions for new geometries. The demand on training data is modest, and the training data can be allocated sparsely. In the present demonstration case, computational time efficiency is improved by two orders of magnitude compared to high-fidelity simulations at the same level of accuracy.

Acknowledgements

This work was partly sponsored by the Ralph N. Read Endowment of the Georgia Institute of Technology.

References

[1] Frangos, M., Marzouk, Y., Willcox, K., and van Bloemen Waanders, B., "Surrogate and Reduced-Order Modeling: A Comparison of Approaches for Large-Scale Statistical Inverse Problems," *Large-Scale Inverse Problems and Quantification of Uncertainty*, Wiley, New York, 2010, pp. 123–149. https://doi.org/10.1002/9780470685853.ch7

[2] Eldred, M., and Dunlavy, D., "Formulations for Surrogate-Based Optimization with Data Fit, Multifidelity, and Reduced-Order Models," *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*,

AIAA Paper 2012-7117, June 2012.
https://doi.org/10.2514/6.2006-7117
[3] Jones, D. R., "A Taxonomy of Global Optimization Methods Based on Response Surfaces," *Journal of Global Optimization*, Vol. 21, Dec. 2001, pp. 345–383.
https://doi.org/10.1023/A:1012771025575
[4] Qian, Z., Seepersad, C. C., Joseph, V. R., Allen, J. K., and Jeff Wu, C., "Building Surrogate Models Based on Detailed and Approximate Simulations," *Journal of Mechanical Design*, Vol. 128, No. 4, 2006, pp. 668–677.
https://doi.org/10.1115/1.2179459
[5] Joseph, V. R., Hung, Y., and Sudjianto, A., "Blind Kriging: A New Method for Developing Metamodels," *Journal of Mechanical Design*, Vol. 130, No. 3, 2008, Paper 031102.
https://doi.org/10.1115/1.2829873
[6] Gutmann, H.-M., "A Radial Basis Function Method for Global Optimization," *Journal of Global Optimization*, Vol. 19, No. 3, 2001, pp. 201–227.
https://doi.org/10.1023/A:1011255519438
[7] Buhmann, M. D., "Radial Basis Functions: Theory and Implementations," *Acta Numerica*, Vol. 9, Jan. 2003, pp. 1–38.
https://doi.org/10.1017/S0962492900000015
[8] Wold, S., "Spline Functions in Data Analysis," *Technometrics*, Vol. 16, No. 1, 1974, pp. 1–11.
https://doi.org/10.1080/00401706.1974.10489142
[9] Sun, G., and Wang, S., "A Review of the Artificial Neural Network Surrogate Modeling in Aerodynamic Design," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 233, No. 16, 2019, pp. 5863–5872.
https://doi.org/10.1177/0954410019864485
[10] Zapata Usandivaras, J. F., Urbano, A., Bauerheim, M., and Cuenot, B., "Data Driven Models for the Design of Rocket Injector Elements," *Aerospace*, Vol. 9, No. 10, 2022, p. 594.
https://doi.org/10.3390/aerospace9100594
[11] Dissanayake, M. W. M. G., and Phan-Thien, N., "Neural-Network-Based Approximations for Solving Partial Differential Equations," *Communications in Numerical Methods in Engineering*, Vol. 10, No. 3, 1994, pp. 195–201.
https://doi.org/10.1002/cnm.1640100303
[12] Chen, T., and Chen, H., "Universal Approximation to Nonlinear Operators by Neural Networks with Arbitrary Activation Functions and Its Application to Dynamical Systems," *IEEE Transaction on Neural Networks*, Vol. 6, No. 4, 1995, pp. 911–917.
https://doi.org/10.1109/72.392253
[13] Liu, D., and Wang, Y., "Multi-Fidelity Physics-Constrained Neural Network and Its Application in Materials Modeling," *Journal of Mechanical Design*, Vol. 141, No. 12, 2019, Paper 121403.
https://doi.org/10.1115/1.4044400
[14] Nguyen, H., and Tsai, R., "Numerical Wave Propagation Aided by Deep Learning," *Journal of Computational Physics*, Vol. 475, Feb. 2023, Paper 111828.
https://doi.org/10.1016/j.jcp.2022.111828
[15] Sirignano, J., and Spiliopoulos, K., "DGM: A Deep Learning Algorithm for Solving Partial Differential Equations," *Journal of Computational Physics*, Vol. 375, Dec. 2018, pp. 1339–1364.
https://doi.org/10.1016/j.jcp.2018.08.029
[16] Weinan, E., and Yu, B., "The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems," *Communications in Mathematics and Statistics*, Vol. 6, No. 1, 2018, pp. 1–12.
https://doi.org/10.1007/s40304-018-0127-z
[17] Raissi, M., Perdikaris, P., and Karniadakis, G. E., "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational physics*, Vol. 378, Feb. 2019, pp. 686–707.
https://doi.org/10.1016/j.jcp.2018.10.045
[18] Zhang, Z., Shin, Y., and Em Karniadakis, G., "GFINNs: GENERIC Formalism Informed Neural Networks for Deterministic and Stochastic Dynamical Systems," *Philosophical Transactions of the Royal Society A*, Vol. 380, No. 2229, 2022, Paper 20210207.
https://doi.org/10.1098/rsta.2021.0207
[19] Jin, X., Cai, S., Li, H., and Karniadakis, G. E., "NSFNets(Navier-Stokes Flow Nets): Physics-Informed Neural Networks for the Incompressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 426, Feb. 2021, Paper 109951.
https://doi.org/10.1016/j.jcp.2020.109951
[20] Chen, T., and Chen, H., "Universal Approximation to Nonlinear Operators by Neural Networks with Arbitrary Activation Functions and Its Application to Dynamical Systems," *IEEE Transactions on Neural Networks*, Vol. 6, No. 4, 1995, pp. 911–917.
https://doi.org/10.1109/72.392253
[21] Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E., "Learning Nonlinear Operators via Deeponet Based on the Universal Approximation Theorem of Operators," *Nature Machine Intelligence*, Vol. 3, No. 3, 2021, pp. 218–229.
https://doi.org/10.1038/s42256-021-00302-5
[22] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A., "Fourier Neural Operator for Parametric Partial Differential Equations," arXiv preprint arXiv:2010.08895, 2020.
https://doi.org/10.48550/arXiv.2010.08895
[23] Yu, J., Yan, C., and Guo, M., "Non-Intrusive Reduced-Order Modeling for Fluid Problems: A Brief Review," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 233, No. 16, 2019, pp. 5896–5912.
https://doi.org/10.1177/0954410019890721
[24] Benner, P., Gugercin, S., and Willcox, K., "A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems," *SIAM Review*, Vol. 57, No. 4, 2015, pp. 483–531.
https://doi.org/10.1137/130932715
[25] Czech, C., Lesjak, M., Bach, C., and Duddeck, F., "Data-Driven Models for Crashworthiness Optimisation: Intrusive and Non-Intrusive Model Order Reduction Techniques," *Structural and Multidisciplinary Optimization*, Vol. 65, No. 7, 2022, p. 190.
https://doi.org/10.1007/s00158-022-03282-1
[26] Yano, M., "A Space-Time Petrov-Galerkin Certified Reduced Basis Method: Application to the Boussinesq Equations," *SIAM Journal on Scientific Computing*, Vol. 36, No. 1, 2014, pp. A232–A266.
https://doi.org/10.1137/120903300
[27] Urban, K., and Patera, A., "An Improved Error Bound for Reduced Basis Approximation of Linear Parabolic Problems," *Mathematics of Computation*, Vol. 83, No. 288, 2014, pp. 1599–1615.
https://doi.org/10.1090/S0025-5718-2013-02782-2
[28] Rowley, C. W., Colonius, T., and Murray, R. M., "Model Reduction for Compressible Flows Using POD and Galerkin Projection," *Physica D: Nonlinear Phenomena*, Vol. 189, Nos. 1–2, 2004, pp. 115–129.
https://doi.org/10.1016/j.physd.2003.03.001
[29] Baumann, M., Benner, P., and Heiland, J., "Space-Time Galerkin POD with Application in Optimal Control of Semilinear Partial Differential Equations," *SIAM Journal on Scientific Computing*, Vol. 40, No. 3, 2018, pp. A1611–A1641.
https://doi.org/10.1137/17M1135281
[30] Carlberg, K., Bou-Mosleh, C., and Farhat, C., "Efficient Non-Linear Model Reduction via a Least-Squares Petrov–Galerkin Projection and Compressive Tensor Approximations," *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, 2011, pp. 155–181.
https://doi.org/10.1002/nme.3050
[31] Constantine, P. G., and Wang, Q., "Residual Minimizing Model Interpolation for Parameterized Nonlinear Dynamical Systems," *SIAM Journal on Scientific Computing*, Vol. 34, No. 4, 2012, pp. A2118–A2144.
https://doi.org/10.1137/100816717
[32] Swischuk, R., Mainini, L., Peherstorfer, B., and Willcox, K., "Projection-Based Model Reduction: Formulations for Physics-Based Machine Learning," *Computers & Fluids*, Vol. 179, Jan. 2019, pp. 704–717.
https://doi.org/10.1016/j.compfluid.2018.07.021
[33] Mak, S., Sung, C. L., Wang, X., Yeh, S. T., Chang, Y. H., Joseph, V. R., Yang, V., and Wu, C., "An Efficient Surrogate Model of Large Eddy Simulations for Design Evaluation and Physics Extraction," *Journal of the American Statistical Association*, Vol. 113, No. 524, 2018, pp. 1443–1456.
https://doi.org/10.1080/01621459.2017.1409123
[34] Yeh, S. T., Wang, X., Sung, C. L., Mak, S., Chang, Y. H., Zhang, L., Wu, C. J., and Yang, V., "Common Proper Orthogonal Decomposition-Based Spatiotemporal Emulator for Design Exploration," *AIAA Journal*, Vol. 56, No. 6, 2018, pp. 2429–2442.
https://doi.org/10.2514/1.J056640
[35] Xingjian, W., Shiang-Ting, Y., Yu-Hung, C., and Vigor, Y., "A High-Fidelity Design Methodology Using Les-Based Simulation and Pod-Based Emulation: A Case Study of Swirl Injectors," *Chinese Journal of Aeronautics*, Vol. 31, No. 9, 2018, pp. 1855–1869.
https://doi.org/10.1016/j.cja.2018.07.004
[36] Chang, Y. H., Zhang, L., Wang, X., Yeh, S. T., Mak, S., Sung, C. L., Jeff Wu, C. F., and Yang, V., "Kernel-Smoothed Proper Orthogonal Decomposition–Based Emulation for Spatiotemporally Evolving Flow Dynamics Prediction," *AIAA Journal*, Vol. 57, No. 12, 2019, pp. 5269–5280.
https://doi.org/10.2514/1.J057803

[37] Chang, Y. H., Wang, X., Zhang, L., Li, Y., Mak, S., Wu, C. F. J., and Yang, V., "Reduced-Order Modeling for Complex Flow Emulation by Common Kernel-Smoothed Proper Orthogonal Decomposition," *AIAA Journal*, Vol. 59, No. 9, 2021, pp. 3291–3303. https://doi.org/10.2514/1.J060574

[38] Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P., and Henningson, D. S., "Spectral Analysis of Nonlinear Flows," *Journal of Fluid Mechanics*, Vol. 641, Dec. 2009, pp. 115–127. https://doi.org/10.1017/S0022112009992059

[39] Schmid, P. J., "Dynamic Mode Decomposition of Numerical and Experimental Data," *Journal of Fluid Mechanics*, Vol. 656, Aug. 2010, pp. 5–28. https://doi.org/10.1017/S0022112010001217

[40] Xu, J., and Duraisamy, K., "Multi-Level Convolutional Autoencoder Networks for Parametric Prediction of Spatio-Temporal Dynamics," *Computer Methods in Applied Mechanics and Engineering*, Vol. 372, Dec. 2020, Paper 113379. https://doi.org/10.1016/j.cma.2020.113379

[41] Milan, P. J., Torelli, R., Lusch, B., and Magnotti, G. M., "Data-Driven Model Reduction of Multiphase Flow in a Single-Hole Automotive Injector," *Atomization and Sprays*, Vol. 30, No. 6, 2020, pp. 401–429. https://doi.org/10.1615/atomizspr.2020034830

[42] Peherstorfer, B., Willcox, K., and Gunzburger, M., "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization," *SIAM Review*, Vol. 60, No. 3, 2018, pp. 550–591. https://doi.org/10.1137/16M1082469

[43] Song, X., Lv, L., Sun, W., and Zhang, J., "A Radial Basis Function-Based Multi-Fidelity Surrogate Model: Exploring Correlation Between High-Fidelity and Low-Fidelity Models," *Structural and Multidisciplinary Optimization*, Vol. 60, Sept. 2019, pp. 965–981. https://doi.org/10.1007/s00158-019-02248-0

[44] Liu, Y., and Collette, M., "Improving Surrogate-Assisted Variable Fidelity Multi-Objective Optimization Using a Clustering Algorithm," *Applied Soft Computing*, Vol. 24, Nov. 2014, pp. 482–493. https://doi.org/10.1016/j.asoc.2014.07.022

[45] Gano, S. E., Renaud, J. E., and Sanders, B., "Hybrid Variable Fidelity Optimization by Using a Kriging-Based Scaling Function," *AIAA Journal*, Vol. 43, No. 11, 2012, pp. 2422–2433. https://doi.org/10.2514/1.12466

[46] Jiang, P., Xie, T., Zhou, Q., Shao, X., Hu, J., and Cao, L., "A Space Mapping Method Based on Gaussian Process Model for Variable Fidelity Metamodeling," *Simulation Modelling Practice and Theory*, Vol. 81, Feb. 2018, pp. 64–84. https://doi.org/10.1016/j.simpat.2017.11.010

[47] Zhou, Q., Jiang, P., Shao, X., Hu, J., Cao, L., and Wan, L., "A Variable Fidelity Information Fusion Method Based on Radial Basis Function," *Advanced Engineering Informatics*, Vol. 32, April 2017, pp. 26–39. https://doi.org/10.1016/j.aei.2016.12.005

[48] Forrester, A. I., Sóbester, A., and Keane, A. J., "Multi-Fidelity Optimization via Surrogate Modelling," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 463, No. 2088, 2007, pp. 3251–3269. https://doi.org/10.1098/rspa.2007.1900

[49] Kennedy, M. C., and O'Hagan, A., "Predicting the Output from a Complex Computer Code when Fast Approximations are Available," *Biometrika*, Vol. 87, No. 1, 2000, pp. 1–13. https://doi.org/10.1093/biomet/87.1.1

[50] Krishnan, K. V., and Ganguli, R., "Multi-Fidelity Analysis and Uncertainty Quantification of Beam Vibration Using Co-Kriging Interpolation Method," *Applied Mathematics and Computation*, Vol. 398, June 2021, Paper 125987. https://doi.org/10.1016/j.amc.2021.125987

[51] Xiao, M., Zhang, G., Breitkopf, P., Villon, P., and Zhang, W., "Extended Co-Kriging Interpolation Method Based on Multi-Fidelity Data," *Applied Mathematics and Computation*, Vol. 323, April 2018, pp. 120–131. https://doi.org/10.1016/j.amc.2017.10.055

[52] Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W., and Kutz, J. N., "Shallow Neural Networks for Fluid Flow Reconstruction with Limited Sensors," *Proceedings of the Royal Society A*, Vol. 476, No. 2238, 2020, Paper 20200097. https://doi.org/10.1098/rspa.2020.0097

[53] Trask, N., Patel, R. G., Gross, B. J., and Atzberger, P. J., "GMLS-Nets: A Framework for Learning from Unstructured Data," arXiv preprint arXiv:1909.05371, 2019. https://doi.org/10.48550/arXiv.1909.05371

[54] Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S., "Machine Learning–Accelerated Computational Fluid Dynamics," *Proceedings of the National Academy of Sciences*, Vol. 118, No. 21, 2021, Paper e2101784118. https://doi.org/10.1073/pnas.2101784118

[55] Huang, H., Yang, V., and Liu, Y., "Neural Networks with Local Converging Inputs (NNLCI) for Solving Conservation Laws, Part I: 1D Problems," *Communications in Computational Physics*, Vol. 34, No. 2, 2023, pp. 290–317. https://doi.org/10.4208/cicp.oa-2022-0285

[56] Huang, H., Yang, V., and Liu, Y., "Neural Networks with Local Converging Inputs (NNLCI) for Solving Conservation Laws, Part II: 2D Problems," *Communications in Computational Physics*, Vol. 34, No. 4, 2023, pp. 907–933. https://doi.org/10.4208/cicp.OA-2023-0026

[57] Cobb, H., Lee, H., and Liu, Y., "Solving Maxwell's Equation in 2D with Neural Networks with Local Converging Inputs," arXiv:2302.02860, 2023. https://doi.org/10.48550/arXiv.2302.02860

[58] Van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101–136. https://doi.org/10.1016/0021-9991(79)90145-1

[59] Rusanov, V. V., "The Calculation of the Interaction of Non-Stationary Shock Waves with Barriers," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, Vol. 1, No. 2, 1961, pp. 267–279. https://doi.org/10.1016/0041-5553(62)90062-9

[60] Gottlieb, S., and Shu, C.-W., "Total Variation Diminishing Runge-Kutta Schemes," *Mathematics of Computation*, Vol. 67, No. 221, 1998, pp. 73–85. https://doi.org/10.1090/s0025-5718-98-00913-2

[61] Garth, C., and Joy, K. I., "Fast Memory-Efficient Cell Location in Unstructured Grids for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, No. 6, 2010, pp. 1541–1550. https://doi.org/10.1109/tvcg.2010.156

T. I. Shih
Associate Editor