A Neural-Network Based Adaptive Discontinuous Galerkin Method for Turbulent Flow Simulations

Miles McGruder*, Aniruddhe Pradhan†, and Krzysztof J. Fidkowski‡
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

This paper presents an adaptive meshing technique for turbulent simulations using a neural-network based error indicator. The network is designed to be compact and has generalized well on untrained flow conditions. The discontinuous Galerkin method allows localized polynomial order refinement, eliminating costly and error-prone re-meshing. The neural-network drives adaptation by predicting unresolved scales, providing a natural error indicator when the solution is under-resolved. The proposed methods are tested on turbulent channel flow simulations and a simplified turbomachinery cooling slot geometry. Preliminary results show good performance of the network model in reconstructing fine-scale solutions and its effectiveness in driving mesh adaptation.

I. Introduction

High Reynolds number turbulent flows are of particular interest in aerospace engineering since the operating conditions of modern aircraft often fall into this flow regime. However, due to the wide range of spatial and temporal scales present in the flow fields, fully-resolved solutions of high Reynolds number turbulent flows are computationally intractable even for simple geometries. Therefore, a more practical approach is to only resolve the large (coarse) scales while modeling the effects of the small (fine) scales, a process also know as large eddy simulation (LES). In LES, either an explicit sub-grid scale (SGS) model or the numerical dissipation embedded in the discretization (*e.g.*, numerical fluxes) is used to model the unresolved turbulent motions. In this paper, the latter approach, often termed implicit large eddy simulation (ILES), is adopted. Since no explicit sub-grid scale models are employed, the solution accuracy of ILES is strongly tied to the mesh resolution. Solution accuracy in under-resolved flow regions can be improved by selectively refining the simulation in regions of high estimated error.

Solution reconstruction, also called super-resolution, has drawn much attention over the past decade, mainly due to the successful application of machine learning techniques. Motivated by the early success of data-driven approaches in Reynolds-Averaged Navier-Stokes (RANS) simulations [1–3], many attempts have been made to improve LES modeling through machine learning. Gamahara and Hattori [4] used an artificial neural-network to establish a map between the grid-scale flow field and the SGS stress without any assumption on its functional form, though no performance gains were found compared to traditional SGS models. Wang *et al.* [5] incorporated more comprehensive physical insights in the network input design, achieving an improved extrapolation capability and a better performance over conventional models. Fan *et al.* [6] investigated the performance of different machine learning models and the relative importance of different flow variables as the input features. Many different machine learning architectures and various network feature designs have also been studied recently [7–9]. These methods super-resolve the effects of sub-grid scales on the resolved scales, *i.e.*, the sub-grid scale stresses. Therefore, when used in real-time simulations, they are not very different from the explicit LES except that the traditional SGS models are replaced by machine learning models. On the other hand, network models have also been used to directly super-resolve the fine-scale solution field using the coarse space solutions [10–12]. These approaches, by themselves, can be very powerful post-processing tools for coarse-grained simulation or experimental data.

Although solution accuracy can be improved by a super-resolution model directly augmenting an ILES, the overall accuracy gains stall on a fixed mesh since the optimal accuracy (in L^2 or other well-defined norms) on a given mesh and order is limited by the approximation power. Another more traditional, yet very effective approach to improve the solution accuracy is mesh adaptation. In this work, the super-resolution approach is employed to create an error indicator to improve accuracy and computational efficiency of ILES flow solutions.

*Graduate Research Assistant
†Graduate Research Assistant
‡Professor, AIAA Associate Fellow

The remainder of this paper proceeds as follows. We describe the adopted discontinuous Galerkin (DG) discretization in Section II. The development of the neural-network based super-resolution model is presented in Section III. Section IV outlines the details of the error indicator and the adaptation strategy in the network enabled modeling framework. The preliminary results are shown in Section V, and Section VI concludes the present work and discusses potential future work.

II. Spatial Discretization

The governing equations used in this work are compressible Navier-Stokes, which can be written in a compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}) - \nabla \cdot \vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the conservative flow state vector of rank s, and $\vec{\mathbf{F}}$ and $\vec{\mathbf{G}}$ denote the inviscid and viscous fluxes, respectively. The viscous flux is assumed to be linear in the state gradients, $\vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{K}(\mathbf{u})\nabla \mathbf{u}$, where $\mathbf{K}$ denotes the diffusivity tensor.

We discretize Eq. (1) with the discontinuous Galerkin (DG) finite-element method, which is suitable for high-order accuracy and hp-refinement [13–16]. Consider a partition $\mathcal{T}_h$ of the computational domain Ω consisting of N_e non-overlapping elements Ω_e, $\mathcal{T}_h = \{\Omega_e : \cup \Omega_e = \Omega, \cap \Omega_e = \emptyset\}$. The state is approximated by piece-wise polynomials lying on the approximation space $\mathcal{V}_h$, with no continuity constraints on the approximation between adjacent elements. Formally, the approximation space is defined as $\mathcal{V}_h = [\mathcal{V}_h]^s$, where $\mathcal{V}_h = \{v \in L^2(\Omega) : v|\Omega_e \in \mathcal{P}^p, \forall \Omega_e \in \mathcal{T}_h\}$, and $\mathcal{P}^p$ denotes polynomials of order p on the reference space of element Ω_e. The weak form of Eq. (1) follows from multiplying the equation by test functions (taken from the same approximation space), integrating by parts, and coupling elements via unique inter-element fluxes,

$$\begin{aligned} &\int_{\Omega_e} \mathbf{w}_h^T \frac{\partial \mathbf{u}}{\partial t} d\Omega - \int_{\Omega_e} \nabla \mathbf{w}_h^T \cdot \left[\vec{\mathbf{F}}(\mathbf{u}_h) - \vec{\mathbf{G}}(\mathbf{u}_h, \nabla \mathbf{u}_h)\right] d\Omega + \\ &\int_{\partial \Omega_e} \mathbf{w}_h^T \left[\widehat{\mathbf{F}}(\mathbf{u}_h^+, \mathbf{u}_h^-) - \widehat{\mathbf{G}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla \mathbf{u}_h^+, \nabla \mathbf{u}_h^-)\right] \cdot \vec{n} dS - \int_{\partial \Omega_e} (\mathbf{u}_h^+ - \{\mathbf{u}_h\})^T \vec{\mathbf{G}}(\mathbf{u}_h^+, \nabla \mathbf{w}_h^+) \cdot \vec{n} dS = \mathbf{0}, \quad \forall \mathbf{w}_h \in \mathcal{V}_h. \end{aligned} \tag{2}$$

On the element boundary $\partial \Omega_e$, $(\cdot)^+$, $(\cdot)^-$ denote, respectively, the quantities taken from the element or its neighbor, $\{\cdot\}$ is the face/edge average or the boundary value, and $\widehat{(\cdot)} \cdot \vec{n}$ represents the uniquely defined normal numerical flux on element interfaces. The last term on the left-hand side (LHS) of Eq. (2) symmetrizes the weak form and ensures adjoint consistency.

Choosing a basis for the approximation space, the DG weak form in Eq. (2) yields a system of ordinary differential equations,

$$\frac{d\mathbf{U}_h}{dt} + \mathbf{f}_h(\mathbf{U}_h) = \mathbf{0}, \quad \mathbf{f}_h = \mathbf{M}_h^{-1}\mathbf{R}_h(\mathbf{U}_h), \tag{3}$$

where $\mathbf{U}_h \in \mathbb{R}^{N_h}$ is the discrete unknown vector of the basis function coefficients, $\mathbf{M}_h$ is the global block-diagonal mass matrix, and $\mathbf{R}_h$ is the discrete spatial residual vector.

III. Super-Resolution Neural-Network

A. Super-Resolution Model

The idea of super-resolution is to reconstruct unresolved fine-scale solutions using the readily available coarse space solution. Suppose we have the exact solution $\mathbf{u}$. We decompose it as

$$\mathbf{u} = \mathbf{u}_H + \mathbf{u}', \quad \mathbf{u} \in \mathcal{V}, \quad \mathbf{u}_H \in \mathcal{V}_H, \quad \mathbf{u}' \in \mathcal{V}, \tag{4}$$

where $\mathcal{V}$ is the infinite-dimensional space containing the exact solution, $\mathcal{V}_H$ denotes the finite-dimensional space defined by the numerical discretization, and $\mathbf{u}'$ represents the fine-scale component missing from the discrete solution $\mathbf{u}_H$. The idea of super-resolution is to reconstruct the fine-scale component $\mathbf{u}'$, and hence the true solution, using the coarse-space solution $\mathbf{u}_H$. However, given the high dimensionality of the fine scale component $\mathbf{u}'$, the super-resolution

model may be extremely complex. Therefore, we search for a surrogate of the true solution, $\mathbf{u}_h^*$, in a more accessible finer space $\mathcal{V}_h$, $\mathcal{V}_H \subset \mathcal{V}_h \subset \mathcal{V}$, such that,

$$\mathbf{u}_h^* = \mathbf{u}_H + \mathbf{u}_h', \quad \mathbf{u}_h^* \in \mathcal{V}_h, \quad \mathbf{u}_H \in \mathcal{V}_H, \quad \mathbf{u}_h' \in \mathcal{V}_h. \tag{5}$$

In this work, we define $\mathbf{u}_h^*$ as the L_2 optimal representation of the true solution $\mathbf{u}$ in $\mathcal{V}_h$,

$$\mathbf{u}_h^* = \arg\min_{\mathbf{w}_h} \int_\Omega \|\mathbf{w}_h - \mathbf{u}\|^2 d\Omega, \quad \mathbf{w}_h \in \mathcal{V}_h. \tag{6}$$

Now, the super-resolution model can be stated as finding a map f_{sr} from the coarse-scale solution to the missing fine-scale component,

$$\mathbf{u}_h' = f_{sr}(\mathbf{u}_H). \tag{7}$$

In this work, the super-resolution map is approximated by a neural-network model.

B. Fully Connected Neural-Networks

In this work, we use fully connected neural-networks (FCNNs), which are designed to emulate the response of neurons to input signals. They are also known as artificial neural-networks or multi-layer perceptrons in the early days of machine learning [17]. A simple three-layer FCNN is shown in Figure 1a. It consists of an input layer $\mathbf{x}$, an output layer $\mathbf{y}$, and a hidden layer that involves an affine transformation and a nonlinear activation. The map between $\mathbf{x}$ and $\mathbf{y}$ can be written as

$$\begin{aligned} &\mathbf{y} = f(\mathbf{\Theta}^{\text{out}}\mathbf{h}), \quad \mathbf{h} = \sigma(\mathbf{z}), \quad \mathbf{z} = \mathbf{\Theta}^{\text{in}}\mathbf{x}, \\ &\mathbf{\Theta}^{\text{in}}\mathbf{x} \equiv \mathbf{W}^{\text{in}}\mathbf{x} + \mathbf{b}^{\text{in}}, \quad \mathbf{\Theta}^{\text{out}}\mathbf{h} \equiv \mathbf{W}^{\text{out}}\mathbf{h} + \mathbf{b}^{\text{out}}. \end{aligned} \tag{8}$$

$\mathbf{z}$ is an affine transformation of the input $\mathbf{x}$ with parameters $\mathbf{\Theta}^{\text{in}}$, which contains both the linear map weights $\mathbf{W}^{\text{in}} \in \mathbb{R}^{\dim(\mathbf{z})\times\dim(\mathbf{x})}$, and a translation or bias term $\mathbf{b}^{\text{in}} \in \mathbb{R}^{\dim(\mathbf{z})}$. A nonlinear activation function σ then maps $\mathbf{z}$ element-wise to the hidden units $\mathbf{h}$, often referred to as the hidden *neurons*. The nonlinear activation σ provides the power of modeling complex phenomena and is often defined *a priori*; examples include the sigmoid or rectified linear unit (ReLU) [18] functions. From the hidden layer to the output, we only show an affine map $\mathbf{\Theta}^{\text{out}}$ in Figure 1a, although one more nonlinear or linear activation f can also be applied.

The complexity and approximation power of a network increase as the number of neurons increases. One can also stack the hidden layers to increase the approximation capacity, resulting in a multi-layer network. Deep FCNNs are usually obtained by increasing both the number of hidden layers and the number of neurons in each layer, as shown in Figure 1b. For a neural-network of L hidden layers, the corresponding model can be written as,

$$\begin{aligned} &\mathbf{h}^1 = \sigma(\mathbf{\Theta}^1\mathbf{x}) \in \mathbb{R}^{N_1}; \\ &\mathbf{h}^l = \sigma(\mathbf{\Theta}^l\mathbf{h}^{l-1}) \in \mathbb{R}^{N_l}, \quad l = 2, 3, ..., L; \\ &\mathbf{y} = f(\mathbf{\Theta}^{L+1}\mathbf{h}^L). \end{aligned} \tag{9}$$

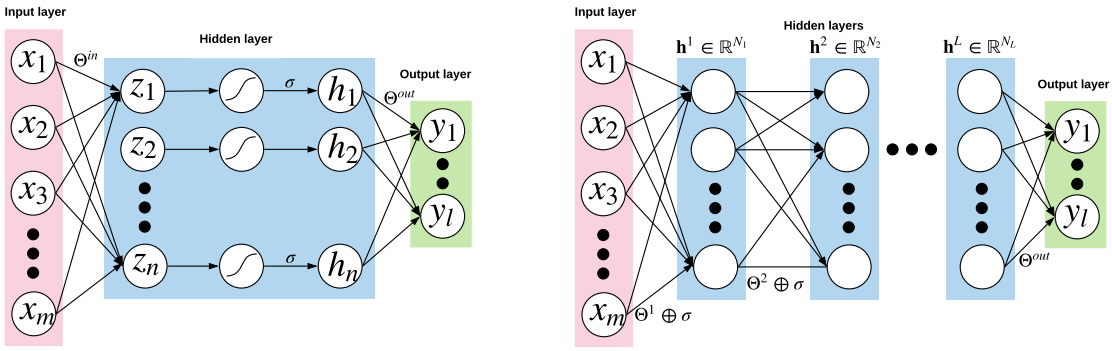
The number of hidden layers, L, and the dimension of each hidden layer, N_l, are hyper-parameters of the network, which can be fine-tuned to achieve higher efficiency and better performance. The network trainable parameters (weights and bias) $\mathbf{\Theta}^l$, $l = 1, ..., L+1$, are obtained by minimizing an objective function, often called the *loss function*, measuring the deviation between the model outputs and the target values from the observed data.

C. Network Design

The goal of the current work is to construct a neural-network model f_{sr}^{nn} to approximate the super-resolution map given in Eq. (7),

$$\mathbf{u}_h' = f_{sr}(\mathbf{u}_H) \approx f_{sr}^{nn}(\mathbf{u}_H). \tag{10}$$

However, due to the convective nature of the flow, the coarse-scale solutions in the elements along the characteristics should all affect the fine-scale components, resulting in a high-dimensional map that is extremely complex and difficult to train. In order to increase the compactness of the network model, we assume that only the local coarse-scale solutions



(a) single-layer FCNN **(b) deep FCNN**

Fig. 1 Structures of fully connected neural-networks (FCNNs).

on the element and its neighbors affect the super-resolved component. Consider a super-resolution model in discrete form,

$$\mathbf{U}'_{h,e} = F_{sr}(\mathbf{U}_{H,e}, \{\mathbf{U}_{H,n}\}), \tag{11}$$

where $\mathbf{U}_{H,e}$ and $\mathbf{U}'_{h,e}$ are the basis coefficients of the coarse-scale solution and the fine-scale components in element e, and $\{\mathbf{U}_{H,n}\}$ denotes a set of solution vectors from the neighboring elements. Two more parameters that dominate the interactions between coarse and fine scales are also considered, which are the fluid kinematic viscosity ν and the local length scale (mesh size) Δ.

For the model to generalize well on unseen data, a nondimensionalization process, or in the machine learning perspective, a data pre-processing procedure, is applied before the network training, resulting in a model in the form of

$$\frac{\mathbf{U}'_{h,e}}{\mathbf{U}_{\text{rms}}} = F_{sr}^{nn}\left(\frac{\mathbf{U}_{H,e} - \mathbf{U}_m}{\mathbf{U}_{\text{rms}}}, \frac{\{\mathbf{U}_{H,n} - \mathbf{U}_m\}}{\mathbf{U}_{\text{rms}}}, \log\left(\frac{V_{\text{rms}}\Delta}{\nu}\right)\right), \tag{12}$$

where $\mathbf{U}_m$ and $\mathbf{U}_{\text{rms}}$ are the mean and the root mean square of the state solution, and V_{rms} represents the flow speed, all measured in element e. The last input feature in Eq. (12) is the elemental Reynolds number, which is an indicator of the under-resolution present in the local element. Due to the wide range of the elemental Reynolds number, a logarithm is taken to help the model training.

In this work, we consider DG solutions on hexahedral meshes using tensor-product nodal basis functions. The network is then trained on three-dimensional rectangular meshes. A simplified two dimensional version of the proposed network structure for super-resolving one state component from $p = 1$ to $p = 3$ is sketched in Figure 2. A final input data permutation step generates additional samples by rotating a given training sample through all positive volume rotational symmetries of a cube to remove directional bias in the trained model.

IV. Adaptation Strategy

In the context of ILES, as no explicit sub-grid scale model is employed, the discretization error and the modeling error are tightly coupled. Refining the mesh, which reduces the discretization error and hence the modeling error, would yield asymptotically a DNS solution. However, for effective LES modeling, only scales that are large enough to affect our quantities of interest, *e.g.*, mean drag values, need to be well-resolved, while the smaller scales should be modeled. Therefore, effective adaptive LES should be targeting areas that are most important for accurate output predictions.

A. Error Indicator

Since the super-resolution model estimates the difference between a coarse state and a fine state, the magnitude can be used as an error indicator. Specifically, we have chosen to take the discrete L^1 norm of the super-resolution output on each element k.

$$e_k = \|F_{sr}^{nn}(\mathbf{U}_{H,k})\mathbf{U}_{\text{rms},k}\|_1 \tag{13}$$

The element-wise indicator allows localized element adaptation. The per-element indicator may be averaged over statistically constant regions, such as wall normal layers in a turbulent channel, to reduce noise in the indicated error.

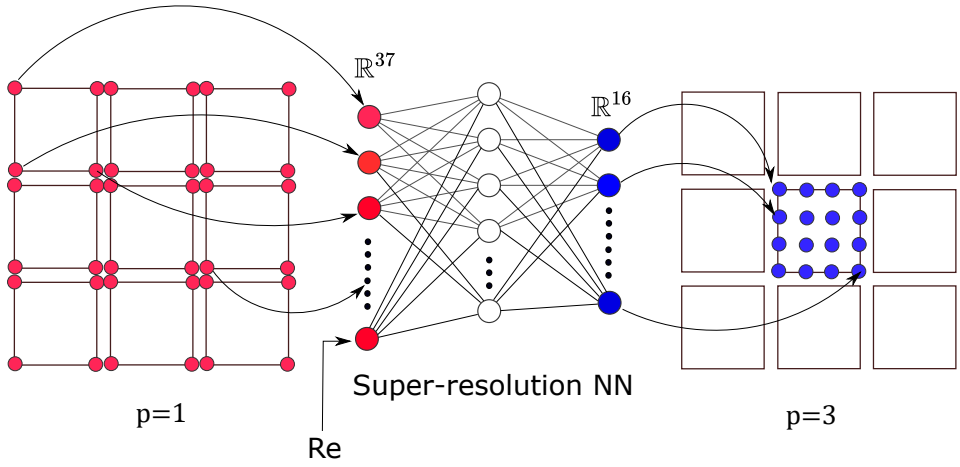


Fig. 2 Simplified 2D super-resolution neural-network model from $p = 1$ to $p = 3$.

We also average the error indicator over many snapshots to further reduce noise for statistically steady flows. This is a fairly straightforward error indicator that will simply show error where under-resolution is present but not necessarily indicate the cause. An adjoint-weighted residual indicator should address this issue and is the subject of future work.

V. Results

A. Data Generation and Network Training

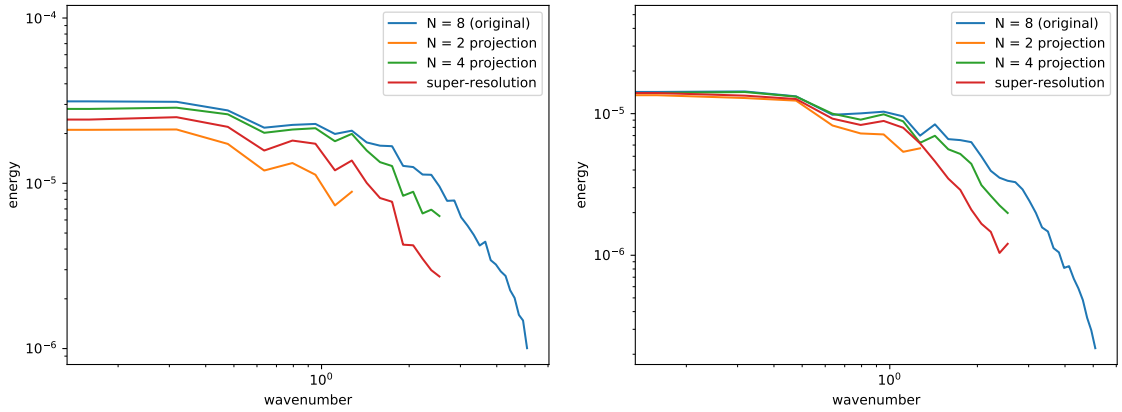
Training data for the super-resolution neural-network are generated by capabilities we have added to NASA's *eddy* code[19, 20]. All training and test cases use hexagonal elements with tensor-product basis functions. In line with *eddy*, all orders are listed in terms of N, the number of basis functions in each dimension of an element *i.e.* $p + 1$ where p is polynomial order. For super-resolution, the network input neighborhood consists of the central element of interest and all elements directly across a face for a total of seven elements. The length scale used for element Reynolds number generation is the length of the axis-aligned bounding box in each dimension. The training data are generated by projecting cases simulated at high-order down to the relatively low-orders required for super-resolution. We have generated training data by sampling an $Re_\tau = 950$ turbulent channel case. Once we have adapted an initial constant order solution there will be multiple solution orders in the same domain. This situation is handled by training a separate neural-network for each potential order. When training data are generated, neighboring elements are always projected down to a coarse order based on the solution order of the central element. We use the same core code for training and adaptation, the only difference being when training, the input data are generated by projecting a high-order solution to low-order and when online, the input data to the network are generated by the present simulation state.

B. Offline Flow Reconstruction Testing

Testing the network performance is tricky because in general we do not know how the high-resolution state should behave. A scenario where this is not true involves projected data from a high-order solution. In order to test the network we perform a process similar to training, where a high-resolution state is projected to two lower-orders. We are interested in how well the super-resolution network is able to reproduce the higher-order projected state from the lower-order projected state. We do not expect the network to perform perfectly but the network should perform better than the low resolution baseline. Given the importance of turbulent energy spectra in the analysis of turbulent flow, we use the energy spectrum to quantify the accuracy of our model.

One such test is shown in Figure 3. This figure shows the super-resolved spectrum between the high and low resolution projected spectra as expected. While the network is trained on $Re_\tau = 950$ data, the non-dimensionalization allows the network to perform correctly on $Re_\tau = 395$ input data as designed. For each sample in Figure 3 the energy at the lowest wavenumber is slightly different. Since we are only sampling a plane of data parallel to the wall for

the spectrum, the turbulent spectra are subject to out-of-plane projection effects. This is acceptable however, since we should expect super-resolution to handle out-of-plane regions of the same element correctly and move the $N = 2$ spectrum toward the $N = 4$ spectrum regardless. Wavenumbers consisting primarily of noise are cut off in Figure 3. It is assumed that noise takes hold in the spectrum when the wave number exceeds half the number of degrees of freedom in the sampling direction divided by the sampling distance.



(a) $y/\delta = 0.3$ **(b)** $y/\delta = 0.7$

Fig. 3 Streamwise turbulent energy spectra of a $Re_\tau = 395$ turbulent channel simulated at N = 8 projected to N = 4 and N = 2 with N = 4 super-resolution at two wall-normal locations.

C. Test Case: Turbulent Channel

1. Meshes

In a turbulent channel simulation mesh resolution will typically be focused near the walls where turbulent kinetic energy is highest, with the channel size set such that two-point velocity correlations are small at half domain distance in the periodic directions. Instead of biasing resolution toward the walls in our adapted cases, we have chosen to begin adaptation on a mesh with uniformly spaced elements in all directions. Starting with uniform elements allows the adaptation algorithm to determine a resolution distribution we can compare to prior knowledge of resolution requirements. Comparing Table 1 to recommended wall-resolved LES resolutions in [21], the uniform channel meshes should be adequately resolved in the streamwise and spanwise directions but element spacing at and near the wall should be extremely lacking. For the channel size we have followed [22], $2\pi\delta$ in the streamwise direction and $\pi\delta$ in the spanwise direction, where δ is the channel half height. At $Re_\tau = 395$, these choices result in the mesh spacing shown in Table 1. These are approximate spacings found by dividing element size in wall units by the number of one dimensional degrees of freedom for that element.

Table 1 Approximate grid spacing in wall units for uniformly spaced meshes.

	Δx^+	Δz^+	Δy^+
N = 4	78	39	12
N = 8	39	20	6

2. Adapted Results

The initial condition is a uniform velocity field with sine wave variation of various frequencies in all velocities and in all directions. This field is integrated forward in time until a linear profile of total shear stress, $\overline{u'v'} - \mu\partial\bar{u}/\partial y$, is achieved as discussed in [23]. Averaging over the streamwise and spanwise directions is employed to accelerate convergence of the statistical profiles. Once a low-order solution has reached statistically steady state it is used to seed

high-order solutions which undergo the same process with an improved initial condition. The initial $N = 4$ solution is adapted for three iterations with approximately 20% of elements incremented by two polynomial approximation orders each iteration. While the error indicator is computed for each element, the adaptation takes advantage of the statistical streamwise homogeneity, spanwise homogeneity, and center-line mirror of the channel case. This means all elements at a particular wall normal distance are aggregated when determining adaptation groups.

After three adaptation iterations, the adapted solution uses 243,712 degrees of freedom. The uniform $N = 4$ solution uses 65,536, and the uniform $N = 8$ solution uses 524,288. To emphasize the degree of under-resolution at $N = 4$, notice that the bump in the $N = 4$ velocity profile in Figure 4 around $y^+ = 50$ marks the end of the element bordering the wall. The adapted solution removes this bump and remains close to the $N = 8$ solution until around $y^+ = 100$ where the higher resolution solution predicts higher velocity in the bulk of the channel in line with the DNS result from [22].

The turbulence intensity proves difficult to resolve correctly. The $N = 4$ profile barely resembles the final profile in the first element between $y^+ = 0$ and $y^+ = 50$. Despite significant improvement, the $N = 8$ solution also struggles in the first element, showing significant oscillations. The first element of the adapted solution at $N = 10$ performs significantly better than the uniform $N = 8$ solution near the wall despite continued oscillation. Closer to the center of the channel, both the adapted and $N = 8$ solutions continue to underestimate the turbulence intensity in similar fashion.

Reynolds stress proves to be the easiest statistical profile to match. Beyond the first element, even the $N = 4$ solution is reasonable. The $N = 8$ solution shows oscillations around the DNS profile while the adapted solution shows little deviation at all at less than half the computational expense of $N = 8$.

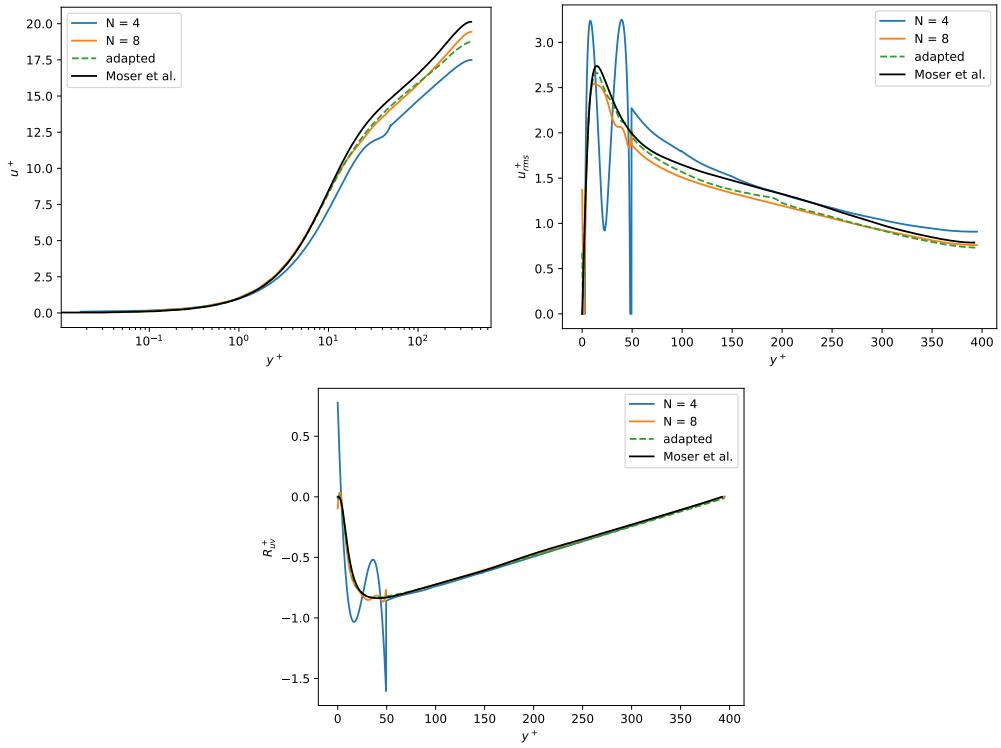


Fig. 4 Normalized velocity, turbulence intensity, and Reynolds stress profiles for the $Re_\tau = 395$ turbulent channel case.

The aggregation of elements for adaptation leads to the banded adaptation pattern shown in Figure 5. As expected, the error indicator targets the edges of the channel domain, reaching $N = 10$ in the first layer, followed by three layers at $N = 6$, and all remaining layers at $N = 4$.

D. Demonstration Case: Cooling Slot

To demonstrate our method, we have chosen a turbulent mixing test case meant to mimic a cooling slot in turbomachinery, presented in [24]. This case uses auxiliary domains to produce a turbulent boundary condition at

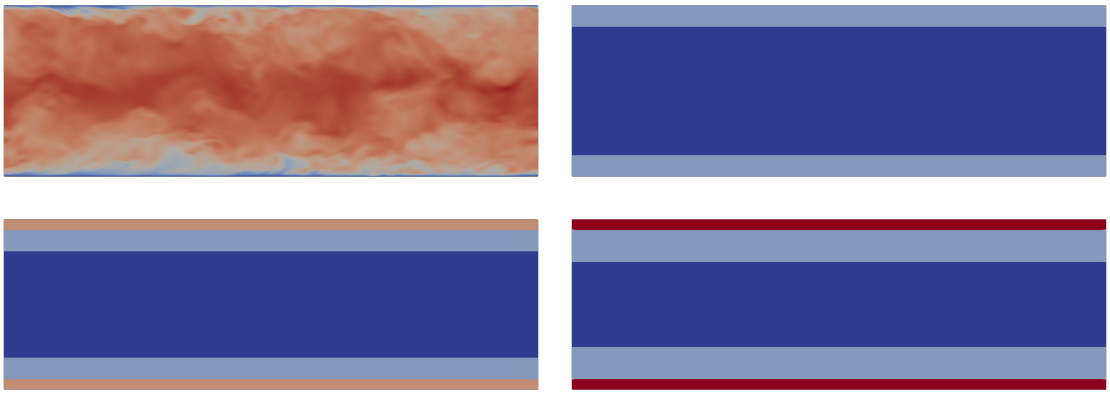


Fig. 5 $Re_\tau = 395$ **turbulent channel adapted order distributions. From top left to bottom right: high resolution velocity field for visual reference, order field after 1, 2, and 3 adaptive iterations. Adapted orders range from** $N = 4$ **to** $N = 10$ **by two.**

the inlet. The upper auxiliary domain is a one-way coupled periodic boundary layer simulation where boundary layer thickness is maintained by a specialized forcing field. Source terms are added to the Navier-Stokes equations corresponding to the mean gradient of each state according to a log-law and wake profile. The lower auxiliary domain is a one-way coupled turbulent channel driven by a simple constant body force. The lower auxiliary domain connects directly to the cooling slot of height y_c. Inflow from the auxiliary domains is initially separated by a small lip in the main computational domain before mixing, as shown in Figure 6. The end of the lip is positioned at $x/y_c = 0$.

Each adaptation iteration refines approximately 20% of elements. Since the problem is statistically homogeneous in the spanwise direction by construction, elements sharing spanwise faces are grouped and adaptation is performed in an effectively two dimensional space with these element groups. Only elements with a downstream distance less than approximately $x/y_c = 32$ are adapted to ensure there is no excessive downstream refinement.

The flow-through time scale for the main computational domain is $t = 80y_c/U_\infty$. In all cases, averaging is started after waiting at least $50.6t$ from the initial condition and averaging takes place for at least $10.1t$, all at a time step of approximately $5 \times 10^{-4}t$. This integration time exceeds that of [24], which used $2 - 3t$. This ensures that the difference between the profile at half averaging time and full averaging time is small relative to the difference between profiles for different simulations. The profile at each station is generated from the average of 20 spanwise velocity samples and all simulations from this paper use the same high resolution (N = 8) inlet regions.

Figure 7 shows that after two adaptive iterations the velocity profiles notably improve at each station except the first. The adapted velocity field performs slightly worse than $N = 6$ at all stations but uses approximately a third fewer degrees of freedom. In Figure 7, statistics from [24] are provided where available, showing the present simulations are still thoroughly under-resolved.

As shown in the order plot of Figure 6, the adaptation has primarily focused on the region immediately behind the lip. It is surprising that relatively little emphasis has been placed on the flow separation at the corners of the lip but a better error indicator could solve this issue.

Table 2 Slot case degree of freedom counts in primary computational domain. The adapted result has undergone two adaptive iterations.

N = 4	adapted	N = 6	N = 8
592,640	1,313,600	2,000,160	4,741,120

VI. Conclusion and Future Work

Large eddy simulation, as a high-fidelity modeling tool for turbulent flows, offers more accurate outputs and extra physical insights compared to widely used RANS models. Despite the excellent accuracy, the computational cost for

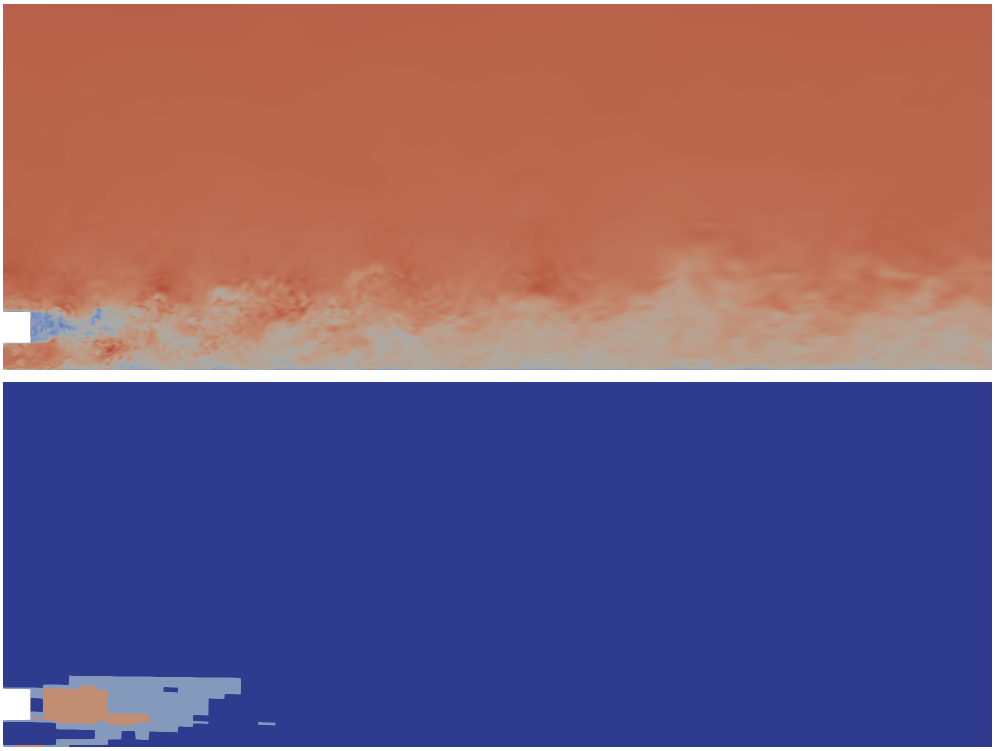


Fig. 6 Mixed $N = 4, 6, 8$ adapted order distribution and reference velocity field.

well-resolved LES remains intractable for practical problems. In order to enable LES modeling in industry-relevant problems, the computational efficiency has to be improved for traditional LES. Actively adapting the computational mesh using a machine-learning based error indicator has been considered as a potential solution in this work.

Neural-network models are designed to reconstruct high polynomial approximation order turbulent flow solutions using low-order approximations. Properly trained networks perform well on reconstructing the solution field. With careful design of the network input and output features, the network generalizes well to unseen data.

The network structure considered so far is relatively simple, and more complicated architectures can be used to improve the super-resolution model. The error indicator used in this work is also a very simple state difference. We are working on an entropy-adjoint based weighted-residual indicator to improve performance. The testing of the super-resolution model is not yet complete and the model performance on various flow conditions and geometries will also be assessed in the future.

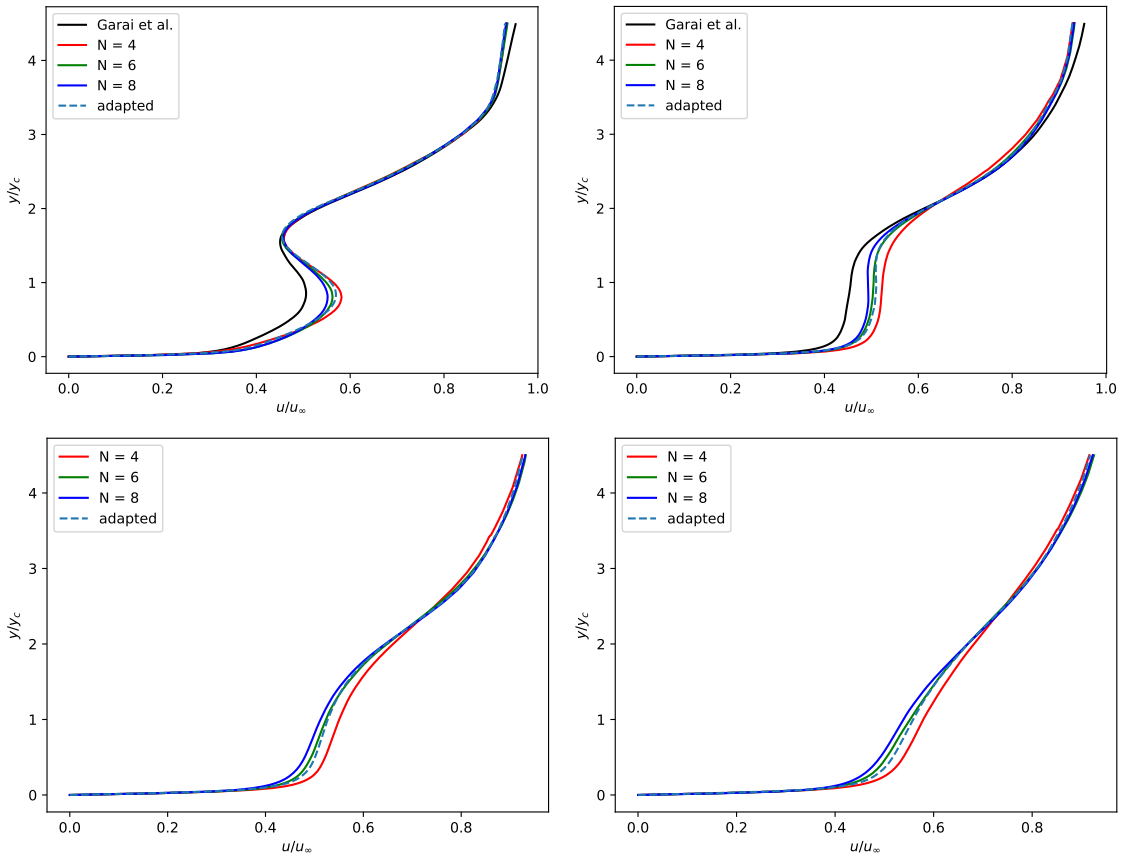


Fig. 7 Normalized velocity profiles for the slot case at various downstream stations. From top left to bottom right the stations are x/y_c = 4, x/y_c = 10, x/y_c = 20 and x/y_c = 30.

Acknowledgments

The authors acknowledge the support of NASA, grant number 80NSSC18M0149, with technical monitor Gary Coleman.

References

[1] Singh, A. P., and Duraisamy, K., "Using field inversion to quantify functional errors in turbulence closures," *Physics of Fluids*, Vol. 28, No. 4, 2016, p. 045110. https://doi.org/10.1063/1.4947045, URL https://doi.org/10.1063/1.4947045.

[2] Parish, E. J., and Duraisamy, K., "A paradigm for data-driven predictive modeling using field inversion and machine learning," *Journal of Computational Physics*, Vol. 305, 2016, pp. 758–774. https://doi.org/10.1016/j.jcp.2015.11.012, URL https://doi.org/10.1016/j.jcp.2015.11.012.

[3] Ling, J., Kurzawski, A., and Templeton, J., "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, Vol. 807, 2016, pp. 155–166. https://doi.org/10.1017/jfm.2016.615.

[4] Gamahara, M., and Hattori, Y., "Searching for turbulence models by artificial neural network," *Physical Review Fluids*, Vol. 2, No. 5, 2017. https://doi.org/10.1103/physrevfluids.2.054604, URL https://doi.org/10.1103/physrevfluids.2.054604.

[5] Yang, X. I. A., Zafar, S., Wang, J.-X., and Xiao, H., "Predictive large-eddy-simulation wall modeling via physics-informed neural networks," *Physical Review Fluids*, Vol. 4, No. 3, 2019. https://doi.org/10.1103/physrevfluids.4.034602, URL https://doi.org/10.1103/physrevfluids.4.034602.

[6] Wang, Z., Luo, K., Li, D., Tan, J., and Fan, J., "Investigations of data-driven closure for subgrid-scale stress in large-eddy simulation," *Physics of Fluids*, Vol. 30, No. 12, 2018, p. 125101. https://doi.org/10.1063/1.5054835, URL https://doi.org/10.1063/1.5054835.

[7] Beck, A., Flad, D., and Munz, C.-D., "Deep neural networks for data-driven LES closure models," *Journal of Computational Physics*, Vol. 398, 2019, p. 108910. https://doi.org/10.1016/j.jcp.2019.108910, URL https://doi.org/10.1016/j.jcp.2019.108910.

[8] Xie, C., Wang, J., Li, K., and Ma, C., "Artificial neural network approach to large-eddy simulation of compressible isotropic turbulence," *Physical Review E*, Vol. 99, No. 5, 2019. https://doi.org/10.1103/physreve.99.053113, URL https://doi.org/10.1103/physreve.99.053113.

[9] Xie, C., Li, K., Ma, C., and Wang, J., "Modeling subgrid-scale force and divergence of heat flux of compressible isotropic turbulence by artificial neural network," *Physical Review Fluids*, Vol. 4, No. 10, 2019. https://doi.org/10.1103/physrevfluids.4.104605, URL https://doi.org/10.1103/physrevfluids.4.104605.

[10] Fukami, K., Fukagata, K., and Taira, K., "Super-resolution reconstruction of turbulent flows with machine learning," *Journal of Fluid Mechanics*, Vol. 870, 2019, pp. 106–120. https://doi.org/10.1017/jfm.2019.238, URL https://doi.org/10.1017/jfm.2019.238.

[11] Deng, Z., He, C., Liu, Y., and Kim, K. C., "Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework," *Physics of Fluids*, Vol. 31, No. 12, 2019, p. 125111. https://doi.org/10.1063/1.5127031, URL https://doi.org/10.1063/1.5127031.

[12] Liu, B., Tang, J., Huang, H., and Lu, X.-Y., "Deep learning methods for super-resolution reconstruction of turbulent flows," *Physics of Fluids*, Vol. 32, No. 2, 2020, p. 025105. https://doi.org/10.1063/1.5140772, URL https://doi.org/10.1063/1.5140772.

[13] Reed, W., and Hill, T., "Triangular mesh methods for the neutron transport equation," Tech. rep., Los Alamos Scientific Lab, October 1973. Available: https://www.osti.gov/servlets/purl/4491151.

[14] Bassi, F., and Rebay, S., "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations," *Journal of Computational Physics*, Vol. 131, No. 2, 1997, pp. 267–279. https://doi.org/10.1006/jcph.1996.5572.

[15] Cockburn, B., and Shu, C.-W., "Runge–Kutta discontinuous Galerkin methods for convection-dominated problems," *Journal of Scientific Computing*, Vol. 16, No. 3, 2001, pp. 173–261. https://doi.org/10.1023/a:1012873910884.

[16] Hartmann, R., and Houston, P., "Adaptive Discontinuous Galerkin Finite Element Methods for the Compressible Euler Equations," *Journal of Computational Physics*, Vol. 183, No. 2, 2002, pp. 508–532. https://doi.org/10.1006/jcph.2002.7206.

[17] Rosenblatt, F., *Priciples of neurodynamics; perceptrons and theory of brain mechanics*, Spartan Books, Washington, D.C., 1962.

[18] Nair, V., and Hinton, G. E., "Rectified linear units improve restricted boltzmann machines," *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[19] Diosady, L., and Murman, S., "Design of a Variational Multiscale Method for Turbulent Compressible Flows," *21st AIAA Computational Fluid Dynamics Conference*, 2013.

[20] Diosady, L., and Murman, S., "Higher-Order Methods for Compressible Turbulent Flows Using Entropy Variables," *53rd AIAA Aerospace Sciences Meeting*, 2015.

[21] Georgiadis, N. J., Rizzetta, D. P., and Fureby, C., "Large-eddy simulation: current capabilities, recommended practices, and future research," *AIAA journal*, Vol. 48, No. 8, 2010, pp. 1772–1784.

[22] Moser, R. D., Kim, J., and Mansour, N. N., "Direct numerical simulation of turbulent channel flow up to Ret=590," *Physics of Fluids*, Vol. 11, No. 4, 1999.

[23] Kim, J., Moin, P., and Moser, R., "Turbulence statistics in fully developed channel flow at low Reynolds number," *Journal of Fluid Mechanics*, Vol. 177, 1987, pp. 133–166.

[24] Garai, A., Murman, S., and Nateri, M., "Scale-Resolving Simulations of a Fundamental Trailing-Edge Cooling Slot Using a Discontinuous-Galerkin Spectral-Element Method," *Proceedings of ASME Turbo Expo 2019: Turbomachinery Technical Conference and Exposition*, 2019.