



# A Machine-Learning Anisotropy Detection Algorithm for Output-Adapted Meshes

Krzysztof J. Fidkowski\*, and Guodong Chen†

*Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA*

This paper presents a machine-learning approach for determining the optimal anisotropy in a computational mesh, in the context of an output-based adaptive solution procedure. An artificial neural network is used to predict the desired element aspect ratio from readily accessible features of the primal and adjoint solutions. Whereas the sizing of the element is still based on an adjoint-weighted residual error estimate, the network augments this information with element stretching magnitude and direction, at lower computational and implementation costs compared to a more rigorous approach: mesh optimization through error sampling and synthesis (MOESS). The network is trained to provide a correction of the anisotropy information relative to a primal Hessian-based calculation, computed from the Mach number field, by incorporating adjoint anisotropy features. MOESS optimized meshes for a variety of steady aerodynamic flows governed by the Reynolds-averaged Navier-Stokes equations in two dimensions provide data for training the multi-layer perceptron network. The network is then deployed and tested by driving complete mesh adaptation sequences, and the results show improvements in mesh efficiency compared to pure primal Hessian-based anisotropy detection.

## I. Introduction

Anisotropic meshes are important for efficiently resolving certain flow features, such as boundary layers and shocks, in computational fluid dynamics. However, determining the optimal mesh anisotropy, i.e. one that produces the most cost-effective meshes for a chosen error measure, is not a trivial task. Heuristics based on flow features, such as the Hessian of a scalar quantity,<sup>1-10</sup> can perform well for many cases but are not generally optimal for a wide range of flow fields and approximation orders. They are also difficult to extend to systems of equations, in which multiple quantities are approximated.

A more rigorous approach for determining mesh anisotropy, and the mesh sizing, involves sampling the effects of element subdivision and regressing a model for the error behavior.<sup>11-13</sup> Such an approach directly addresses the error/cost optimization problem in the presence of unknown local convergence rates but is computationally more complex and requires non-trivial mesh operations for its implementation.

In this work, we investigate the idea of using a machine-learning approach to predict the optimal mesh anisotropy. The goal is to design an algorithm that produces results similar to MOESS, at lower implementation and computational costs, on par with feature-based detection methods. The machine-learning algorithm must be trained on a large amount of data, and for this we use

\* Associate Professor, AIAA Senior Member

† Graduate Student Research Assistant, AIAA Student Member

element-specific results of many MOESS iterations. The solution features used as inputs to the machine-learning algorithm are based on both the primal and the adjoint solution, motivated by the fact that this is the same information that is used to compute the error indicator.

For the machine-learning method, we use an artificial neural network (ANN). ANNs are a popular scientific machine learning approach for modeling nonlinear mappings between vector inputs and outputs.<sup>14–17</sup> They emulate biological systems through connected layers of neurons that are activated under sufficiently-strong input signals. They have been used for many scientific and engineering purposes, primarily for interpolation and data-driven modeling, both of which involve mapping inputs to desired outputs.<sup>18,19</sup> Such a map does not contain physics of the problem and is therefore often viewed as “black box.” Nevertheless, the map can effectively “learn” relationships between inputs and outputs that may be difficult to discern mathematically from first principles.

The outline for the remainder of this paper is as follows. Section II presents the numerical approach used in this work, a discontinuous finite-element discretization. Section III summarizes output-based error estimation, which is used to drive the adaptation algorithms presented in Section IV. Section V introduces the new machine-learning approach for identifying the correct mesh anisotropy during adaptation. Finally, Sections VI and VII present results of the training and deployment of the neural network, and Section VIII concludes with a summary and a discussion of future directions.

## II. Discretization

We present the adaptive approach in the context of a discontinuous Galerkin (DG) finite element discretization.<sup>20–22</sup> We consider a system of partial differential equations in conservative form,

$$\partial_t \mathbf{u} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}, \nabla \mathbf{u}) + \mathbf{S}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \quad (1)$$

where  $\mathbf{u} \in \mathbb{R}^s$  is the  $s$ -component state vector,  $\vec{\mathbf{F}} \in \mathbb{R}^{d \times s}$  is the flux vector,  $d$  is the spatial dimension, and  $\mathbf{S}$  is a source term associated with turbulence modeling closure equations, in this work RANS-SA.<sup>23,24</sup>

We assume a subdivision,  $T_h$ , of the computational domain,  $\Omega$ , into  $N_e$  non-overlapping elements,  $\Omega_e$ , and on each element we approximate the state by an order  $p$  polynomial. Specifically, the state approximation is  $\mathbf{u}_h^p \in \mathcal{V}_h^p = [\mathcal{V}_h^p]^s$ , where

$$\mathcal{V}_h^p \equiv \{u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \forall \Omega_e \in T_h\}. \quad (2)$$

The subscript  $h$  indicates a specific domain subdivision, i.e. the collective size/shape distribution of all of the elements, and  $\mathcal{P}^p$  is the set of order- $p$  polynomials<sup>a</sup>.

Multiplying Eqn. 1 by test functions  $\mathbf{v}_h^p \in \mathcal{V}_h^p$ , integrating by parts on each element, and using the Roe<sup>25</sup> convective flux and the second form of Bassi and Rebay (BR2)<sup>26</sup> for the viscous treatment, we obtain the following semilinear weak form:

$$\mathcal{R}_h^p(\mathbf{u}_h^p, \mathbf{v}_h^p) = 0. \quad (3)$$

Choosing bases for the trial and test spaces results in a system of nonlinear equations that is solved using a Newton-Raphson procedure. The sparse Jacobian matrix is fully stored, and the linear solves are preformed using GMRES, preconditioned by an element-line iterative solver.<sup>22</sup>

<sup>a</sup>The polynomials are defined in reference space and for curved elements they may not remain order  $p$  polynomials after the mapping to physical space.

### III. Output Error Estimation

We use an adjoint-based output error estimate to drive the mesh optimization. Theoretical details can be found in previous works.<sup>27–29</sup> For a scalar output,  $\mathcal{J}(\mathbf{u}_h^p)$ , the discrete adjoint field  $\boldsymbol{\psi}_h^p \in \mathcal{V}_h^p$  is the linearized sensitivity of  $\mathcal{J}$  to residual source perturbations,  $\delta\mathcal{R}(\cdot) : \mathcal{V}_h^p \rightarrow \mathbb{R}$  added to the left-hand side of Eqn. 3,

$$\delta\mathcal{J} = \delta\mathcal{R}(\boldsymbol{\psi}_h^p), \quad (4)$$

where the equality assumes infinitesimal perturbations. We obtain the adjoint field by solving the following linear equation,

$$\mathcal{R}'_h[\mathbf{u}_h^p](\mathbf{v}_h^p, \boldsymbol{\psi}_h^p) + \mathcal{J}'[\mathbf{u}_h^p](\mathbf{v}_h^p) = 0, \quad \forall \mathbf{v}_h^p \in \mathcal{V}_h^p, \quad (5)$$

where the prime denotes Fréchet linearization with respect to the argument in square brackets.

Output error estimation relies on a finer discretization space, which in this work is  $\mathcal{V}_h^{p+1}$ : order  $p+1$  approximation on elements of the same domain subdivision,  $T_h$ . The original (coarse) primal state  $\mathbf{u}_h^p$  does not generally satisfy the fine-space weak form; instead it satisfies a perturbed weak form, with a residual source of  $\delta\mathcal{R}(\cdot) = -\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \cdot)$ . If we have a fine-space adjoint,  $\boldsymbol{\psi}_h^{p+1}$ , we can then estimate the error in  $\mathcal{J}$  due to using the coarse primal instead of the (never calculated) fine-space primal,

$$\text{output error} = \delta\mathcal{J} \equiv \mathcal{J}(\mathbf{u}_h^p) - \mathcal{J}(\mathbf{u}_h^{p+1}) \approx -\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \boldsymbol{\psi}_h^{p+1}). \quad (6)$$

The approximation sign indicates that the adjoint-weighted residual (last term) is an *estimate* of the error between the spaces when the equations or output are not linear, since we are using linear sensitivities with non-infinitesimal perturbations. The calculation in Eqn. 6 is also only an estimate of the true numerical error, i.e. compared to the exact solution, since it uses an approximate, finite-dimensional, fine space. The richer the fine space, the better the error estimate. We rely on the fact that both the primal and the adjoint solutions are used to compute the error estimate when selecting features for the machine-learning approach presented in Section V.

### IV. Adaptation

We note that Eqn. 6 can be localized to elemental contributions,

$$\delta\mathcal{J} \approx \sum_{e=1}^{N_e} -\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \boldsymbol{\psi}_h^{p+1}|_{\Omega_e}), \quad \mathcal{E}_e \equiv |\mathcal{R}_h^{p+1}(\mathbf{u}_h^p, \boldsymbol{\psi}_h^{p+1}|_{\Omega_e})|. \quad (7)$$

$\mathcal{E}_e \geq 0$  in the above equation is the error indicator for element  $e$ . During mesh adaptation, this indicator information provides information to adapt the computational mesh. However, as just one scalar quantity per element, the error indicator is not sufficient to provide information about mesh anisotropy. This information can come from heuristics, or from a sampling approach, as described below.

#### IV.A. Hessian-Based Anisotropy Detection

One dominant approach for detecting the anisotropy is to estimate the directional interpolation error of the solution,<sup>1,30</sup> and we describe here an extension of such an approach that incorporates output error adaptive indicators.<sup>5,9</sup> For linear approximation, i.e.,  $p=1$ , the interpolation error of

a scalar solution  $u$  over an edge  $E$  in the mesh, with unit tangent vector  $\vec{s}$  and length  $h$ , is given by

$$\delta_{u,E} \propto |\vec{s}^T \mathbf{H} \vec{s}| h^2, \quad (8)$$

where  $\mathbf{H}$  is the solution Hessian matrix,

$$H_{i,j} = \frac{\partial^2 u^2}{\partial x_i \partial x_j}, \quad i, j \in [1, \dots, d], \quad (9)$$

and  $d$  is the spatial dimension. The second derivatives can be estimated by a quadratic reconstruction of the linear solution. The scalar  $u$  used in this work is the Mach number as it has been found to be effective for many types of flows, although more sophisticated quantities can also be used.

A geometric interpretation of Eqn. 8 is that the interpolation error over an edge depends on the squared edge length under the measure of  $|\mathbf{H}|$ , which is a symmetric positive definite (SPD) matrix defined by taking the absolute values of its eigenvalues  $\Lambda$  while keeping the eigenvectors  $\mathbf{Q}$  the same,  $|\mathbf{H}| = \mathbf{Q}^T |\Lambda| \mathbf{Q}$ . For higher-order approximations, the interpolation error is characterized by the  $p + 1^{\text{st}}$  derivatives, and the first  $d$  largest directional derivatives can be used to determine the principal directions  $\mathbf{Q}$  and the corresponding stretching  $\Lambda$ .<sup>9</sup> In this work, however, we only consider using the Hessian matrix of second derivatives, regardless of the solution approximation order.

Error equidistribution suggests equally distributing the squared edge length under  $|\mathbf{H}|$ . Consider two principal directions  $\vec{e}_i$  and  $\vec{e}_j$  from  $\mathbf{Q}$ . Error equidistribution yields the mesh stretching as

$$\frac{h_i}{h_j} = (|u_{\vec{e}_j}|/|u_{\vec{e}_i}|)^{1/(p+1)} = (|\lambda_j|/|\lambda_i|)^{1/(p+1)}, \quad (10)$$

where  $u_{\vec{e}_i}$  is the  $p + 1^{\text{st}}$  derivate along the direction  $\vec{e}_i$ , or equivalently the magnitude of the  $i^{\text{th}}$  eigenvalue  $\lambda_i$ . The key idea of Hessian-based mesh adaptation with output error estimation is to use Eqn. 10 to control the mesh stretching (relative mesh size) while using the output error indicator in Eqn. 7 to determine the absolute mesh sizes.

#### IV.B. Element Sizing using A Priori Rate Estimates

In order to perform the mesh adaptation, we need to predict the desired element sizes, or the number of the elements  $N^f$  in the adapted (fine) mesh. Let  $n_k$ , not necessary an integer, be the number of the adapted mesh elements contained in element  $k$  at the original mesh. Denoting the current element size by  $h_i^c$  and the requested element size  $h_i^f$ , where  $i$  again indexes the principal directions,  $n_k$  can be approximated as

$$n_k = \prod_i^d (h_i^c/h_i^f). \quad (11)$$

The current sizes  $h_i^c$  are calculated as the singular values of the mapping from a unit equilateral triangle to element  $k$ .<sup>5</sup> Given an output error tolerance  $e_0$ , to satisfy the error equidistribution, each fine-mesh element is allowed an error  $e_0/N_f$ , which means that each element  $k$  is allowed an error of  $n_k e_0/N_f$ . We relate the growth in the number of elements to an error reduction factor through an *a priori* estimate

$$\underbrace{n_k \frac{e_0}{N_f}}_{\text{allowable error}} = \underbrace{\mathcal{E}_k^c \left( \frac{h_i^f}{h_i^c} \right)^{\bar{p}_k + 1}}_{\text{a priori estimate}}, \quad (12)$$

where  $\mathcal{E}_k^c$  indicates the current error indicator in element  $k$ ,  $\bar{p}_k = \min(p_k, \gamma_k)$ , and  $\gamma_k$  is the lowest order of any singularity within element  $k$ . Substituting Eqn. 11 into Eqn. 12 yields a relation between  $n_k$  and  $N_f$

$$n_k \frac{e_0}{N_f} = \mathcal{E}_k^c \left[ \prod_i \left( \frac{h_i^f}{h_i^c} \right) \right]^{(\bar{p}_k+1)/d} = \mathcal{E}_k^c n_k^{-(\bar{p}_k+1)/d} \Rightarrow n_k^{1+(\bar{p}_k+1)/d} = \frac{\mathcal{E}_k^c}{e_0/N_f}. \quad (13)$$

Substituting Eqn. 13 into  $N^f = \sum_k n_k$ , we can solve for  $N^f$  if  $\bar{p}_k$  is constant on the entire mesh, otherwise it is solved iteratively. Adaptation stops when we meet the error tolerance,  $\epsilon \equiv \sum_k \epsilon_k \leq e_0$ . In practice, the maximum allowable error  $e_0$  can be modified or fixed-growth refinement strategy can also be adopted to control the mesh adaptation.<sup>9</sup>

### IV.C. Sampling-Based Mesh Optimization

The goal in this method, MOESS, is to optimize the computational mesh,  $T_h$ , in order to minimize the output error at a prescribed computational cost. We present the approach introduced by Yano,<sup>12</sup> which iteratively determines the optimal change in the mesh metric field given a prescribed metric-cost relationship and a sampling-inferred metric-error relationship.

#### IV.C.1. Metric-Based Meshing

A Riemannian metric field,  $\mathcal{M}(\vec{x})$ , is a field of symmetric positive definite (SPD) tensors that can be used to encode information about the desired size and stretching of a computational mesh. At each point in physical space,  $\vec{x}$ , the metric tensor  $\mathcal{M}(\vec{x})$  provides a ‘‘yardstick’’ for measuring the distance from  $\vec{x}$  to another point infinitesimally far away,  $\vec{x} + \delta\vec{x}$ . After choosing a Cartesian coordinate system and basis for physical space,  $\mathcal{M}$  can be represented as a  $d \times d$  SPD matrix. The set of points at unit metric distance from  $\vec{x}$  is an ellipse: eigenvectors of  $\mathcal{M}$  give directions along the principal axes, while the length of each axis (stretching) is the inverse square root of the corresponding eigenvalue. The aspect ratio is the ratio of the largest stretching magnitude to the smallest.

A mesh that *conforms* to a metric field is one in which each edge has the same length, to some tolerance, when measured with the metric. An example of a two-dimensional metric-conforming mesher is the Bi-dimensional Anisotropic Mesh Generator (BAMG),<sup>31</sup> and this is used to obtain the results in the present work.

BAMG generates a mesh given a metric field, which is specified at nodes of a background mesh – the current mesh in an adaptive setting. The optimization determines *changes* to the current, mesh-implied, metric,  $\mathcal{M}_0(\vec{x})$ . Affine-invariant<sup>32</sup> changes to the metric field are made via a symmetric *step matrix*,  $\mathcal{S} \in \mathbb{R}^{d \times d}$ , according to

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S}) \mathcal{M}_0^{\frac{1}{2}}. \quad (14)$$

Note that  $\mathcal{S} = 0$  leaves the metric unchanged, while diagonal values in  $\mathcal{S}$  of  $\pm 2 \log 2$  halve/double the metric stretching sizes.

#### IV.C.2. Error Convergence Model

The mesh optimization algorithm requires a model for how the error changes as the metric changes. We consider one element,  $\Omega_e$ , with a current error  $\mathcal{E}_{e0}$ , the absolute value of the element’s contribution to Eqn. 7, and a proposed metric step matrix of  $\mathcal{S}_e$ . The error on  $\Omega_e$  following refinement with this step matrix is given by

$$\mathcal{E}_e = \mathcal{E}_{e0} \exp[\text{tr}(\mathcal{R}_e \mathcal{S}_e)], \quad (15)$$

where  $\mathcal{R}_e$  is a symmetric *rate tensor*. The total error over the mesh is the sum of the elemental errors,  $\mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e$ . The rate tensor,  $\mathcal{R}_e$ , is determined separately for each element through a sampling procedure.<sup>13</sup>

#### IV.C.3. Cost Model

To measure the cost of refinement, we use degrees of freedom,  $\text{dof}$ , which on each element just depends on the approximation order  $p$ , assumed constant over the elements. By Eqn. 14 and properties of the metric tensor, when the step matrix  $S_e$  is applied to the metric of element  $e$ , the area of the element decreases by  $\exp\left[\frac{1}{2}\text{tr}(S_e)\right]$ . Equivalently, the number of new elements, and hence degrees of freedom, occupying the original area  $\Omega_e$  *increases* by this factor. So the elemental cost model is

$$C_e = C_{e0} \exp\left[\frac{1}{2}\text{tr}(S_e)\right], \quad (16)$$

where  $C_{e0} = \text{dof}_{e0}$  is the current number of degrees of freedom on element  $e$ . The total cost over the mesh is the sum of the elemental costs,  $\mathcal{C} = \sum_{e=1}^{N_e} C_e$ .

#### IV.C.4. Metric Optimization Algorithm

Given a current mesh with its mesh-implied metric ( $\mathcal{M}_0(\vec{x})$ ), elemental error indicators  $\mathcal{E}_{e0}$ , and elemental rate tensor estimates,  $\mathcal{R}_e$ , the goal of the metric optimization algorithm is to determine the step matrix field,  $\mathcal{S}(\vec{x})$ , that minimizes the error at a fixed cost.

The step matrix field is approximated by values at the mesh vertices,  $\mathcal{S}_v$ , which are arithmetically-averaged to adjacent elements<sup>b</sup>:

$$\mathcal{S}_e = \frac{1}{|V_e|} \sum_{v \in V_e} \mathcal{S}_v, \quad (17)$$

where  $V_e$  is the set of vertices ( $|V_e|$  is the number of them) adjacent to element  $e$ . The optimization problem is to determine  $\mathcal{S}_v$  such that the total error  $\mathcal{E}$  is minimized at a prescribed total cost  $\mathcal{C}$ .

First-order optimality conditions require derivatives of the error and cost with respect to  $\mathcal{S}_v$ . We note that the cost only depends on the *trace* of the step matrix; i.e. the trace-free part of  $S_e$  stretches an element but does not alter its area. We therefore separate the vertex step matrices into trace ( $s_v \mathcal{I}$ ) and trace-free ( $\tilde{\mathcal{S}}_v$ ) parts, with  $\mathcal{I}$  the identity tensor,

$$\mathcal{S}_v = s_v \mathcal{I} + \tilde{\mathcal{S}}_v. \quad (18)$$

The optimization algorithm is then the same as presented by Yano:<sup>12</sup>

1. Given a mesh, solution, and adjoint, calculate  $\mathcal{E}_e, \mathcal{C}_e, \mathcal{R}_e$  for each element  $e$ .
2. Set  $\delta s = \delta s_{\max}/n_{\text{step}}$ ,  $\mathcal{S}_v = 0$ .
3. Begin loop:  $i = 1 \dots n_{\text{step}}$ 
  - (a) Calculate  $\mathcal{S}_e$  from Eqn. 17,  $\frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e}$  from Eqn. 15, and  $\frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e}$  from Eqn. 16.
  - (b) Calculate derivatives of  $\mathcal{E}$  and  $\mathcal{C}$  with respect to  $s_v$  and  $\tilde{\mathcal{S}}_v$ .
  - (c) At each vertex form the ratio  $\lambda_v = \frac{\partial \mathcal{E}/\partial s_v}{\partial \mathcal{C}/\partial s_v}$  and
    - Refine the metric for 30% of the vertices with the largest  $|\lambda_v|$ :  $\mathcal{S}_v = \mathcal{S}_v + \delta s \mathcal{I}$
    - Coarsen the metric for 30% of the vertices with the smallest  $|\lambda_v|$ :  $\mathcal{S}_v = \mathcal{S}_v - \delta s \mathcal{I}$

<sup>b</sup>There is no need for an affine-invariant average because entries of  $\mathcal{S}$  are coordinate system independent.

- (d) Update the trace-free part of  $S_v$  to enforce stationarity with respect to shape changes at fixed area:  $S_v = S_v + \delta s(\partial\mathcal{E}/\partial\tilde{S}_v)/(\partial\mathcal{E}/\partial s_v)$ .
- (e) Rescale  $S_v \rightarrow S_v + \beta\mathcal{I}$ , where  $\beta$  is a global constant calculated from Eqn. 16 to constrain the total cost to the desired dof value:  $\beta = \frac{2}{d} \log \frac{\mathcal{C}_{\text{target}}}{\mathcal{C}}$ , where  $\mathcal{C}_{\text{target}}$  is the target cost.

Note,  $\lambda_v$  is a Lagrange multiplier in the optimization. It is the ratio of the marginal error to marginal cost of a step matrix trace increase (i.e. mesh refinement). The above algorithm iteratively equidistributes  $\lambda_v$  globally so that, at optimum, all elements have the same marginal error to cost ratio. Constant values that work generally well in the above algorithm are  $n_{\text{step}} = 20$  and  $\delta s_{\text{max}} = 2 \log 2$ .

In practice, the mesh optimization and flow/adjoint solution are performed several times at a given target cost,  $\mathcal{C}_{\text{target}}$ , until the error stops changing. Then the target cost is increased to reduce the error further if desired.

## V. Machine-Learning Anisotropy Detection

As an alternative to mesh subdivision sampling and regression to determine the anisotropy information, we present an approach that uses a neural network to determine anisotropy from relevant features of the primal and adjoint solution. A key choice in the design of the neural network is the set of features from which the anisotropy information can be detected. For generality across problems, the features should be insensitive to scaling of the problem and the choice of physical units. The Mach number Hessian matrix provides such information from the primal solution, specifically when it is used to determine anisotropy and stretching directions. Note that the element size is obtained from the output error estimate in all cases in this work.

The output error estimation formula involves both the primal (via the residual) and the adjoint (as a weight) solutions, and we therefore seek to incorporate features of the adjoint solution into the anisotropy measure. Our choice for the adjoint features is motivated by the success of the primal Hessian, even in the context of high-order solutions. Without an obvious equivalent all-purpose scalar such as the Mach number in the adjoint solution, we compute and use Hessian information from all of the adjoint variables as the features.

Denote by  $\mathbf{H}_k^\psi$  the Hessian of the  $k^{\text{th}}$  adjoint variable,

$$\left[\mathbf{H}_k^\psi\right]_{i,j} = \frac{\partial^2 \psi_k}{\partial x_i \partial x_j}, \quad i, j \in [1, \dots, d], \quad (19)$$

where  $k \in [1, \dots, s]$  ranges over the state rank. In two spatial dimensions, to normalize the Hessian, we compute the aspect ratio and direction of the first eigenvector,

$$AR = \sqrt{\lambda_2^H / \lambda_1^H}, \quad \theta = \arg(\vec{v}_1), \quad (20)$$

where  $\lambda_1^H \leq \lambda_2^H$  are the Hessian eigenvalues and  $\vec{v}_1$  is the eigenvector corresponding to  $\lambda_1^H$ . The normalized Hessian metric is then

$$\mathbf{M}_k^\psi = \begin{bmatrix} \lambda_1 \cos^2 \theta + \lambda_2 \sin^2 \theta & (\lambda_1 - \lambda_2) \sin \theta \cos \theta \\ (\lambda_1 - \lambda_2) \sin \theta \cos \theta & \lambda_1 \sin^2 \theta + \lambda_2 \cos^2 \theta \end{bmatrix}, \quad \text{where } \lambda_1 = \frac{1}{AR}, \lambda_2 = AR. \quad (21)$$

Note that this matrix does not encode sizing, which is not a meaningful quantity from the Hessian matrix, and hence has only two independent quantities (e.g.  $AR$  and  $\theta$ ). This gives a total of  $2s$  adjoint features for use in the machine-learning algorithm.

An artificial neural network is used to map the adjoint features to a measure of the element's anisotropy and orientation. Inputs into neural networks should generally be normalized to not bias

the training towards extreme cases, e.g. in this case to elements of large aspect ratio. In addition, care must be taken when using an angle as an input, due to the equivalence of 0 and  $2\pi$ . To address both problems, we do not use the adjoint metrics as defined in Eqn. 21 directly as inputs. Nor do we use  $AR$  and  $\theta$ . Instead, we rely on non-dimensional *step matrices* computed relative to a baseline metric,  $\mathbf{M}_0$ , which we take as the Mach number Hessian. These step matrices are defined as

$$\mathbf{S}_k^\psi = \log \left( \mathbf{M}_0^{-\frac{1}{2}} \mathbf{M}_k^\psi \mathbf{M}_0^{-\frac{1}{2}} \right). \quad (22)$$

Since we use normalized metrics (unit size), the step matrices are trace-free and have only two independent components: one on-diagonal and one off-diagonal in two dimensions. This is consistent with the expected  $2s$  inputs from the  $s$  adjoint features.

Figure 1 shows the structure of the network, which is a multi-layer perceptron. It consists of one hidden layer,  $\mathbf{x}_1$ , between the input layer, i.e. the features,  $\mathbf{x}_0$ , and the output layer,  $\mathbf{x}_2$ . The output layer contains the desired anisotropy information, as encoded by a step matrix from the baseline (Hessian) metric,  $\mathbf{M}_0$ . This step matrix is also trace free so that the number of outputs is only 2,  $\mathbf{x}_2 \in \mathbb{R}^2$ . The size of the hidden layer is chosen to be twice the size of the input layer,  $\mathbf{x}_1 \in \mathbb{R}^{n_1}$ , where  $n_1 = 2n_0$ ,  $\mathbf{x}_0 \in \mathbb{R}^{n_0}$ , and  $n_0 = 2s$ .

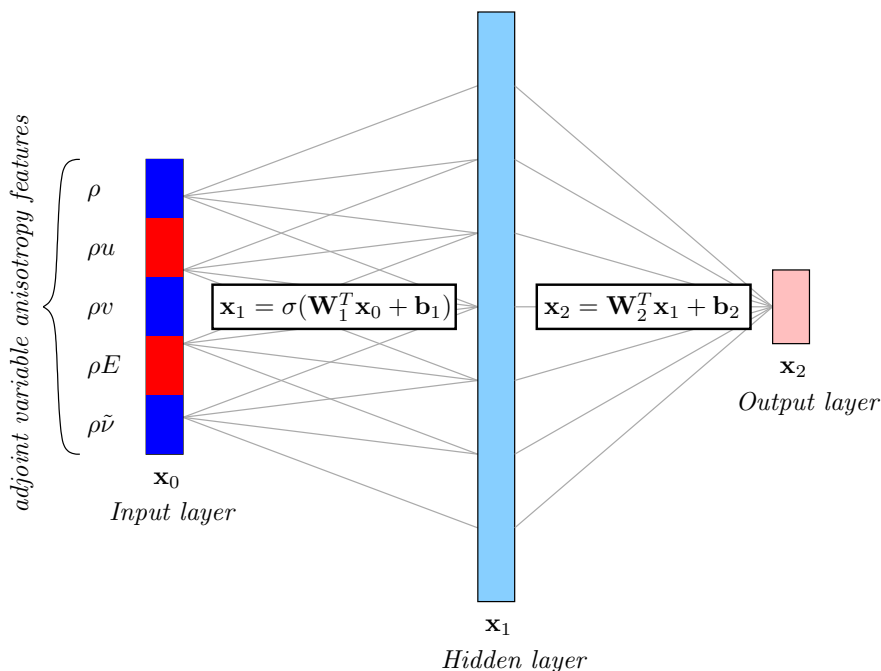


Figure 1. Structure of the artificial neural network used for predicting element anisotropy and orientation.

The network is fully-connected, and the formulas for computing the states within each layer are shown in Figure 1. The map from the input to the hidden layer involves an entry-wise sigmoid activation function,  $\sigma(x) = 1/(1 + e^{-x})$ , whereas no activation function is used for the output layer calculation. The parameters associated with the network consist of the weights and biases,

$$\mathbf{W}_1 \in \mathbb{R}^{n_0 \times n_1}, \quad \mathbf{b}_1 \in \mathbb{R}^{n_1}, \quad \mathbf{W}_2 \in \mathbb{R}^{n_1 \times n_2}, \quad \mathbf{b}_2 \in \mathbb{R}^{n_2}.$$

The values of these parameters are determined using an optimization procedure, the Adam algorithm in TensorFlow,<sup>33</sup> that minimizes the mean squared error loss function between predicted and actual output layer values. The actual values come from training data, which are obtained from



multiple MOESS iterations on prototypical cases. Each MOESS iteration produces one “training point” for each element of the mesh, so for meshes with many elements, a large amount of data can be obtained with a relatively small number of MOESS iterations.

Once the network is trained, it is implemented in the adaptive code as a replacement for the Hessian-only metric anisotropy and stretching calculation in the a priori output-based mesh refinement method. Figure 2 illustrates the entire calculation process from primal and adjoint features to the desired element anisotropy, encoded by the metric  $\mathbf{M}$  presented on the lower-left in the figure. This metric then replaces the Mach Hessian in an otherwise equivalent adaptive procedure.

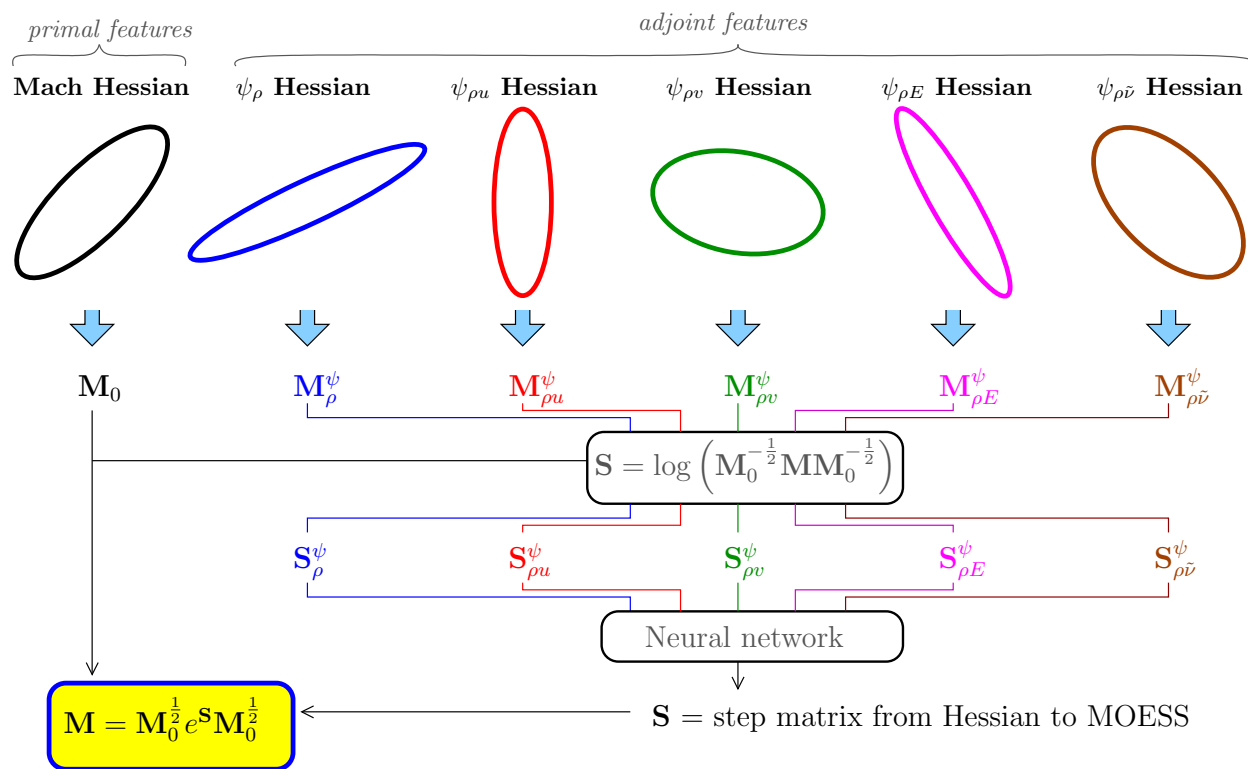


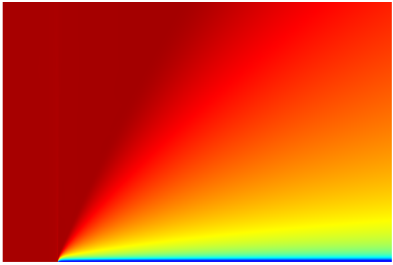
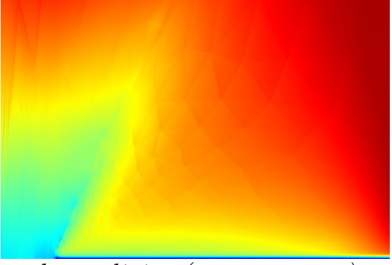
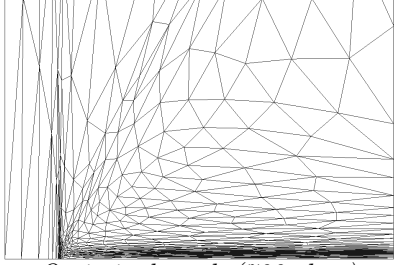
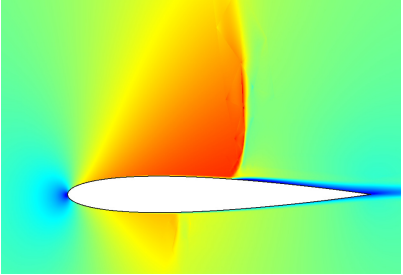
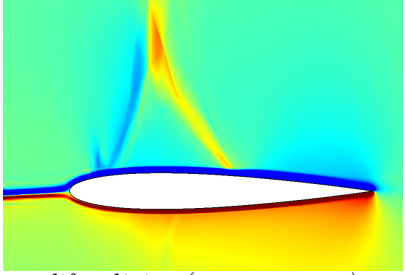
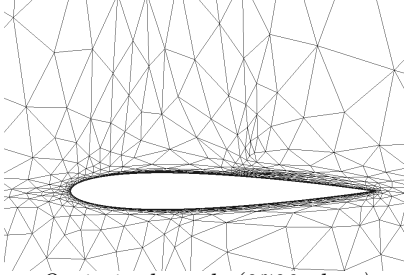
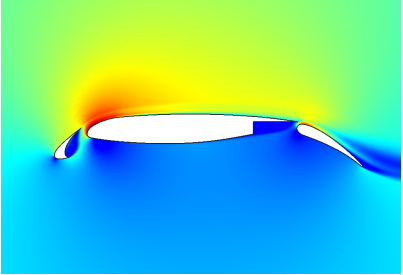
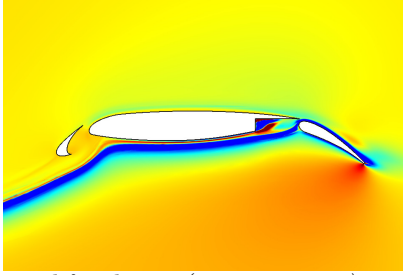
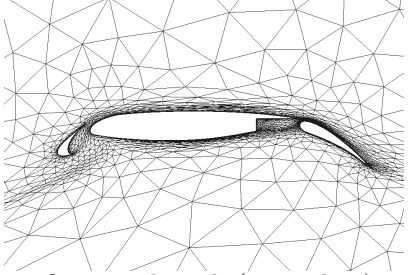
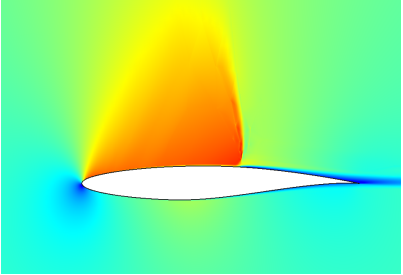
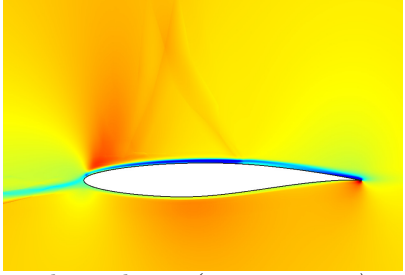
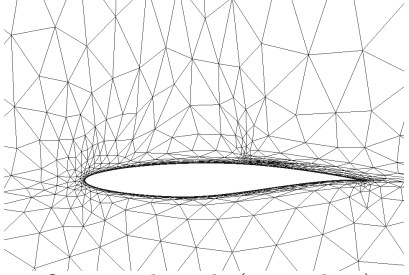
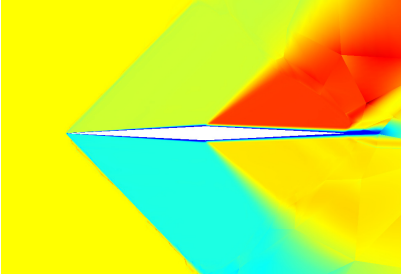
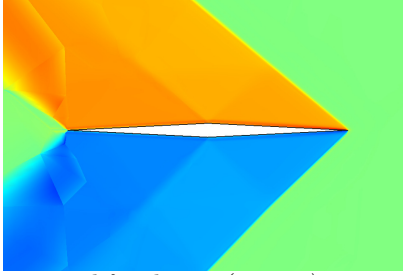
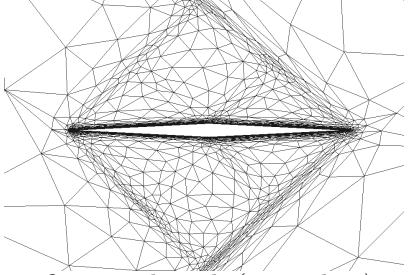
Figure 2. Flowchart of neural-network implementation.

## VI. Neural Network Training

Several prototypical aerodynamic flow cases are run to provide training data for the anisotropy prediction neural network. Table 1 describes these cases and includes figures of the primal and adjoint solutions, and of the optimized meshes. All cases are two-dimensional and use the Reynolds-averaged Navier-Stokes equations. A variety of flow Mach numbers are included, ranging from subcritical to supersonic. The Reynolds numbers are representative of aircraft flight conditions,  $\mathcal{O}(10^6)$ . Force outputs, drag and lift, are considered for error estimation and adaptation. In some cases, for a given flow condition, both outputs yield two different adaptation sequences.

Adapted meshes are generated at a chosen target degree-of-freedom cost using MOESS, at a solution approximation order of  $p = 2$ . The meshes for the different cases are not all made to be of the same size, but the number of adaptive iterations collected is chosen such that the total amount of training data (product of mesh size and number of iterations) is approximately the same among the cases. Data collection begins once the mesh size and outputs stabilize following initialization of MOESS iterations with a user-generated, sub-optimal mesh. Specifically, data from ten MOESS

Table 1. Neural network training cases

<b>Flat plate:</b> $M_\infty = 0.2, 0.5$ , $Re = 10^5, 10^6$ , output = drag (images scaled 100x vertically)		
 <p><i>Mach contours</i></p>	 <p><i>drag adjoint (x-momentum)</i></p>	 <p><i>Optimized mesh (700 elem)</i></p>
<b>NACA 0012:</b> $M_\infty = 0.8$ , $\alpha = 1.25^\circ$ , $Re = 5 \times 10^6$ , outputs = drag, lift		
 <p><i>Mach contours</i></p>	 <p><i>lift adjoint (x-momentum)</i></p>	 <p><i>Optimized mesh (2700 elem)</i></p>
<b>MDA 30P/30N:</b> $M_\infty = 0.2$ , $\alpha = 5^\circ$ , $Re = 10^6$ , outputs = drag, lift		
 <p><i>Mach contours</i></p>	 <p><i>lift adjoint (x-momentum)</i></p>	 <p><i>Optimized mesh (5300 elem)</i></p>
<b>RAE 2822:</b> $M_\infty = 0.734$ , $\alpha = 2.79^\circ$ , $Re = 6.5 \times 10^6$ , outputs = drag, lift		
 <p><i>Mach contours</i></p>	 <p><i>drag adjoint (x-momentum)</i></p>	 <p><i>Optimized mesh (2600 elem)</i></p>
<b>Diamond airfoil:</b> $M_\infty = 1.5$ , $\alpha = 2^\circ$ , $Re = 10^6$ , output = lift		
 <p><i>Mach contours</i></p>	 <p><i>lift adjoint (energy)</i></p>	 <p><i>Optimized mesh (3900 elem)</i></p>

iterations are discarded before collection.

The input data, which consist of the normalized Mach number and adjoint variable Hessian matrices, are computed from the current-space primal and adjoint solutions for each element. For  $p = 2$ , these matrices are constant across elements, whereas for  $p > 2$ , the matrices would be averaged across the element interiors. The output data are computed using the normalized grid-implied metric, since we are using MOESS to generate the meshes, converted to a step matrix relative to the normalized Mach number Hessian.

The total number of training data points (elements) over all cases and adaptive iterations is 121,840. These data are randomized and split 70%/30% into training and test categories. The training data are used to drive the optimization, whereas the test data are used to monitor the loss on untrained data. The training data are broken into mini-batches of size 500 for the optimizer, and the learning rate is set to .001. Several tens of thousands of optimization iterations typically lead to a stabilization of the mean-squared error, as shown in Figure 3. Typically, an order of magnitude drop in the loss is observed, without a significant difference between training and test data loss. This indicates that we are not over-fitting the data, which would be arguably difficult to do with the small neural network size presently considered. Furthermore, the results of the training were not found to be overly sensitive to the choices of the mini-batch size or the learning rate.

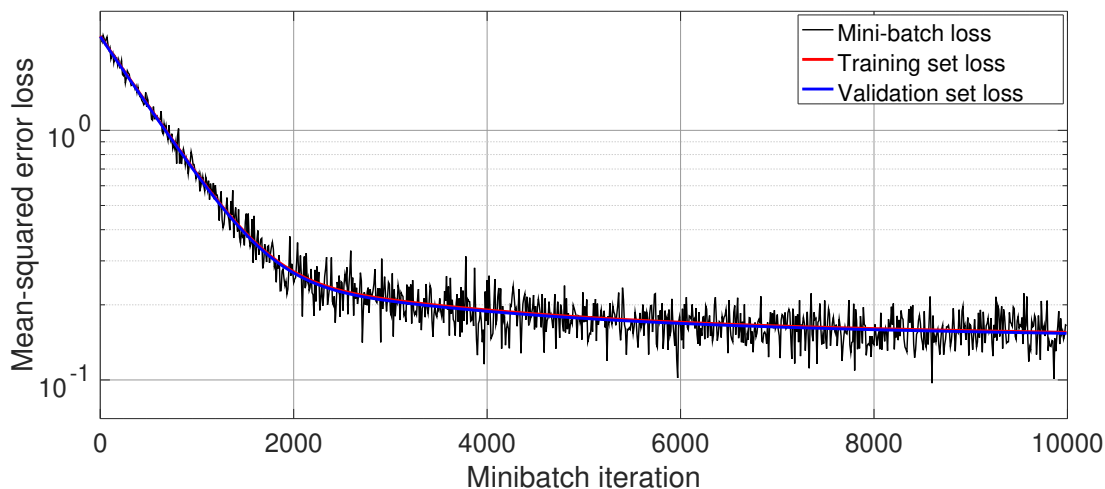


Figure 3. Neural network training loss history, using a 70% training, 30% test split.

## VII. Adaptive Simulation Results

This section presents results obtained by implementing the trained neural network in an adaptive solution process and using it to generate adapted meshes for various flow cases at different degrees of freedom. When using the network to determine anisotropy information, the same a-priori element sizing technique, described in Section IV.B, is used as in the Hessian-based approach. The neural network results are compared to both Mach number Hessian-based adaptation and MOESS.

### VII.A. NACA 0012 airfoil

This test case consists of RANS fully-turbulent flow over a NACA 0012 airfoil at  $M_\infty = 0.8$ ,  $\alpha = 1.25^\circ$ , and  $Re = 5 \times 10^6$ . The computational domain consists of a square farfield boundary 100 chords away from the airfoil. An adiabatic no-slip wall boundary conditions is imposed on the airfoil. Shock capturing is performed using element-based artificial viscosity.<sup>34</sup> The output of interest is

the drag coefficient on the airfoil. A hand-generated initial isotropic mesh of 2800 elements is constructed with sufficient boundary-layer resolution to enable a converged RANS solution.

Adaptive simulations are performed at  $p = 2$  solution approximation order for three target degree-of-freedom (dof) costs: 8000, 16000, 32000. At each target dof, 10 adaptive iterations are performed, using the final mesh from the previous target dof as the starting mesh. The three adaptive methods compared are MOESS, Mach-number Hessian, and neural-network-augmented Hessian.

Figure 4 presents the output convergence history obtained from the adaptive runs. For each method, only one data point is shown at a given dof target: this is the average output at the average dof value computed over the last 4 adaptive iterations at that target dof. An “exact” value is also indicated, obtained by running a MOESS simulation at the same approximation order but a target dof set to four times the finest dof in the study.

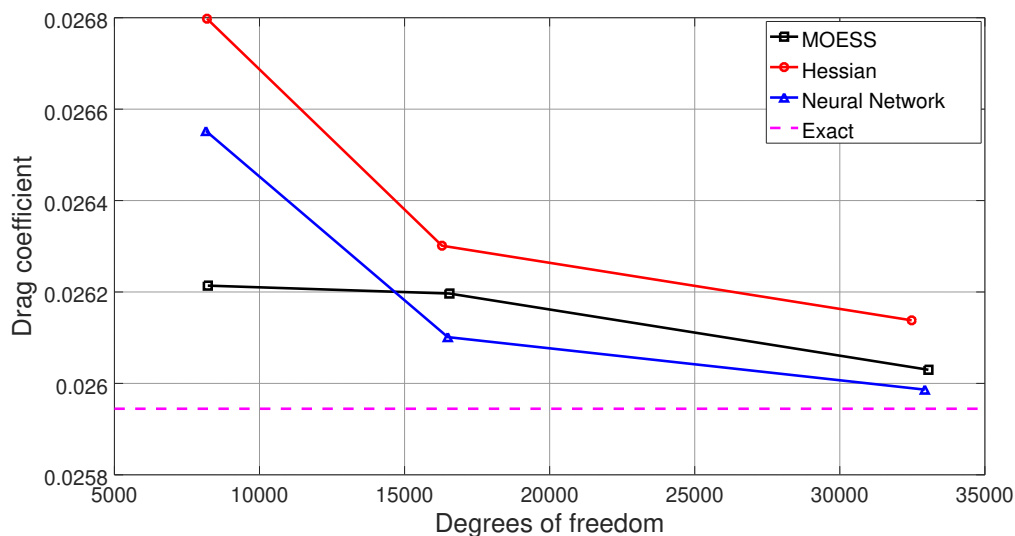


Figure 4. NACA 0012,  $M_\infty = 0.8$ ,  $\alpha = 1.25^\circ$ ,  $Re = 5 \times 10^6$ : output convergence history.

We see that at all target dof, the adaptive results using neural-network anisotropy are closer to the exact value compared to those using the Mach number Hessian anisotropy. As both methods are driven by the same output-based element sizing information, the observed error reduction is made possible by more efficient meshes resulting from improved anisotropy information. Specifically, correct anisotropy identification allows for optimal use of degrees of freedom to reduce the output error.

The neural network approach can achieve more optimal mesh anisotropy because it incorporates information not only from the primal (Mach Hessian), but also from the adjoint (all variables). The network was trained to reproduce the element stretching obtained from MOESS, and hence the slightly-improved performance of the network over MOESS is likely not generalizable – it could be due to a problem-specific sub-optimality of the output-based error indicator.

Figures 5–7 show the final adapted meshes at the highest target dof for the three adaptive approaches tested. The flow is transonic with a strong shock on the upper surface, a weaker shock on the lower surface, and rapid boundary-layer growth in the vicinity of the shocks. From the three figures, we see that the resolution of the shock, which is minimal to start with in the output-based setting, is similar among the methods, with the neural-network indicator exhibiting slightly less primal-based anisotropy compared to the Hessian indicator and MOESS. The resolution for the foot of the lower shock is strongest for the Hessian-based method.

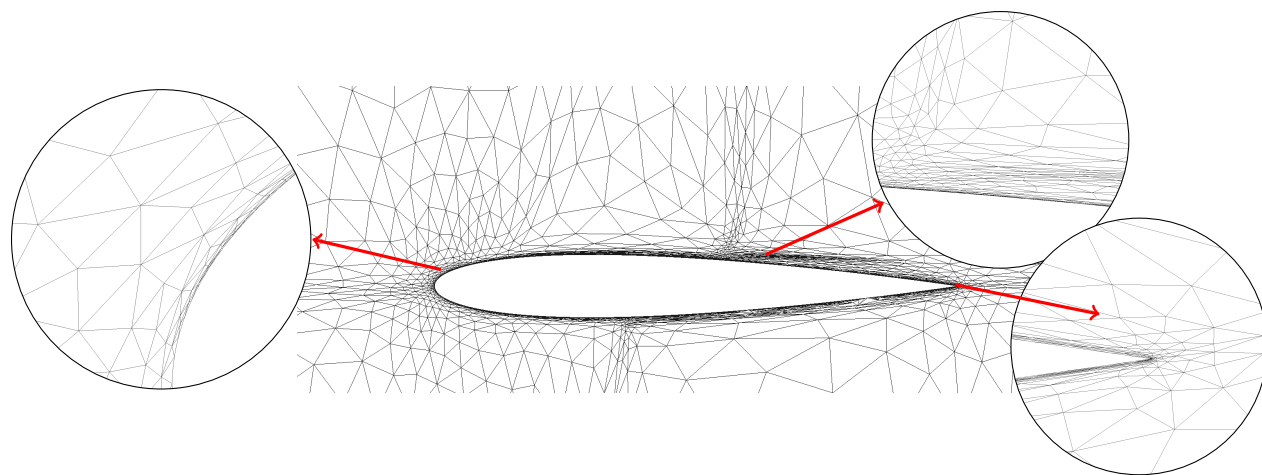


Figure 5. NACA 0012: MOESS final adapted mesh.

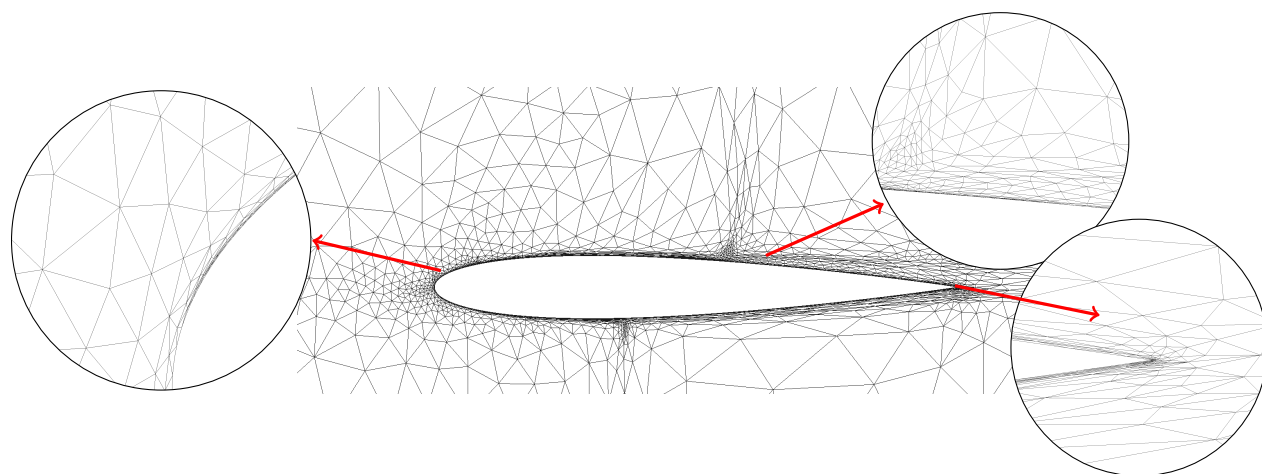


Figure 6. NACA 0012: final Hessian adapted mesh.

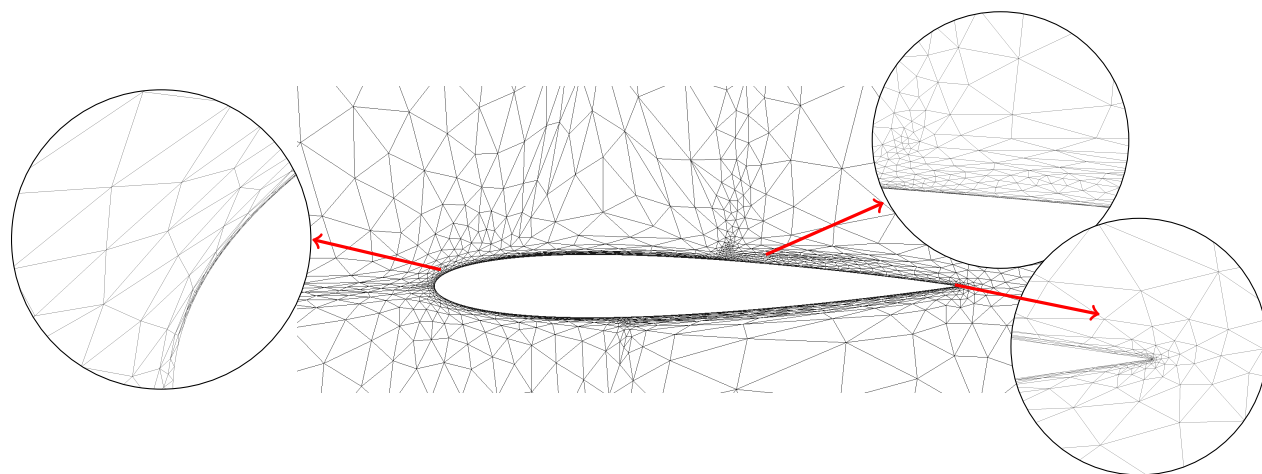


Figure 7. NACA 0012: final neural-network adapted mesh.



As shown in Table 1, the adjoint variable possesses a  $\lambda$  structure variation on the upper surface, and both the neural-network anisotropy measure and MOESS show evidence of its resolution. The Mach number Hessian measure does not align elements to this structure, which is specific to the adjoint. The most notable difference is the leading-edge stagnation streamline anisotropy, which is again an adjoint feature that is resolved by MOESS and the neural network. Although the extent to which this feature needs to be resolved for accurate output prediction is still a point of debate, it is reassuring to see the neural network reproduce the MOESS behavior, for which it was trained. Finally, near the trailing edge, the Hessian-based measure shows more flow-aligned anisotropy, due to the primal wake, which is not as apparent in MOESS and the neural-network approach.

## VII.B. Diamond airfoil

This test case consists of RANS supersonic flow over a diamond airfoil at  $M_\infty = 1.5$ ,  $\alpha = 2^\circ$ , and  $Re = 1 \times 10^6$ . The leading and trailing edge corner angles are both  $2 \tan^{-1}(.05)$ . The computational domain consists of a square farfield boundary 100 chords away from the diamond. An adiabatic no-slip wall boundary conditions is imposed on the airfoil. Shock capturing is again performed using element-based artificial viscosity.<sup>34</sup> The output of interest is the lift coefficient on the airfoil. A hand-generated initial isotropic mesh of 1500 elements is constructed with sufficient boundary-layer resolution to enable a converged RANS solution.

Adaptive simulations are performed at  $p = 2$  solution approximation order for three target degree-of-freedom (dof) costs: 10000, 20000, 40000. At each target dof, 10 adaptive iterations are performed, using the final mesh from the previous target dof as the starting mesh. As in the previous case, MOESS, Mach-number Hessian anisotropy, and neural-network-augmented Hessian anisotropy methods are compared.

Figure 8 presents the output convergence history obtained from the adaptive runs, where again averaging has been performed over the last 4 iterations at each target dof. The “exact” value is also indicated, obtained from a four-times finer-mesh run at the same approximation order.

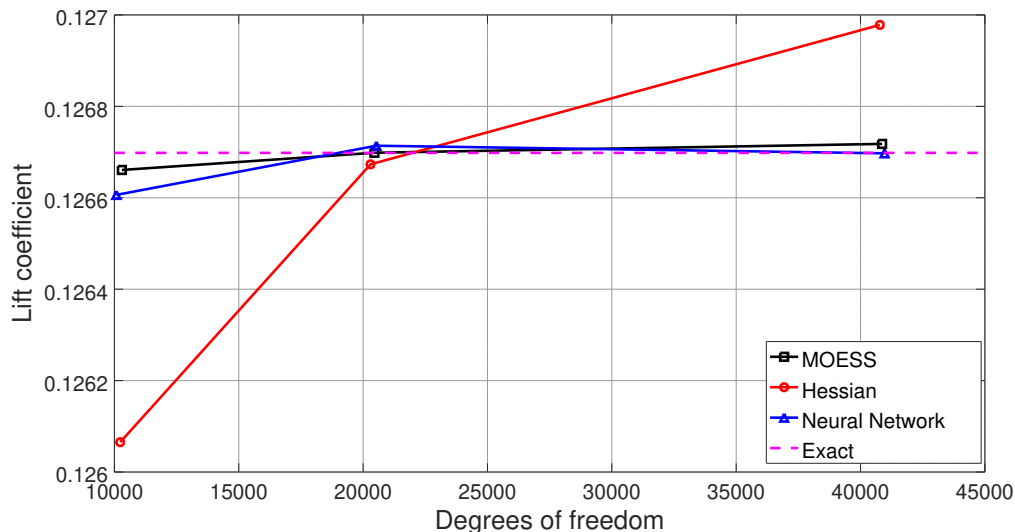


Figure 8. Diamond airfoil,  $M_\infty = 1.5$ ,  $\alpha = 2^\circ$ ,  $Re = 1 \times 10^6$ : output convergence history.

We see that at all target dof, the adaptive results using neural-network anisotropy are much closer to the exact value compared to those using the Mach number Hessian anisotropy. In this case, the difference between Hessian-based anisotropy and MOESS is significant, and the machine-learning approach follows the MOESS results quite closely. As in the previous case, the accuracy

improvement is made possible by a more optimal distribution of degrees of freedom due to correct anisotropy in regions where the output error is more sensitive to resolution in one direction than another. In this case, the performance of MOESS and the neural-network are quite close in terms of output accuracy versus degrees of freedom.

Figures 9–11 show the final adapted meshes at the highest target dof for the three adaptive approaches tested. As shown, the meshes among the three approaches are drastically different in their placement of anisotropic elements.

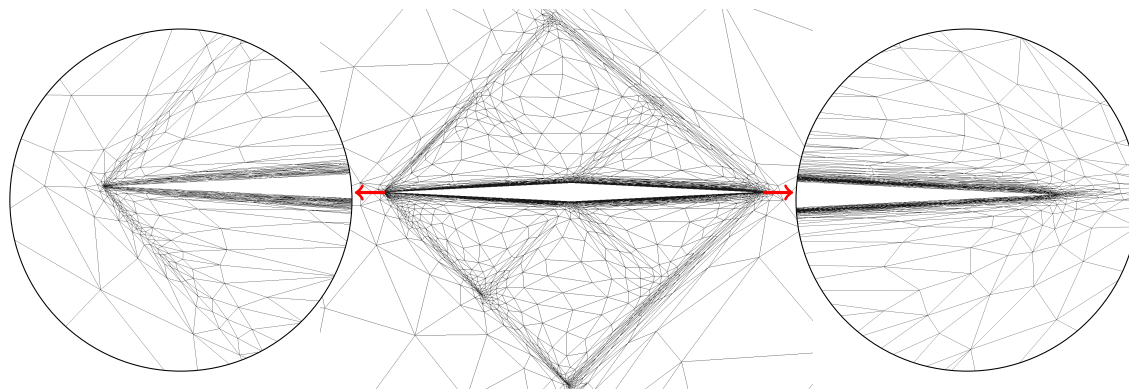


Figure 9. Diamond airfoil: final MOESS adapted mesh.

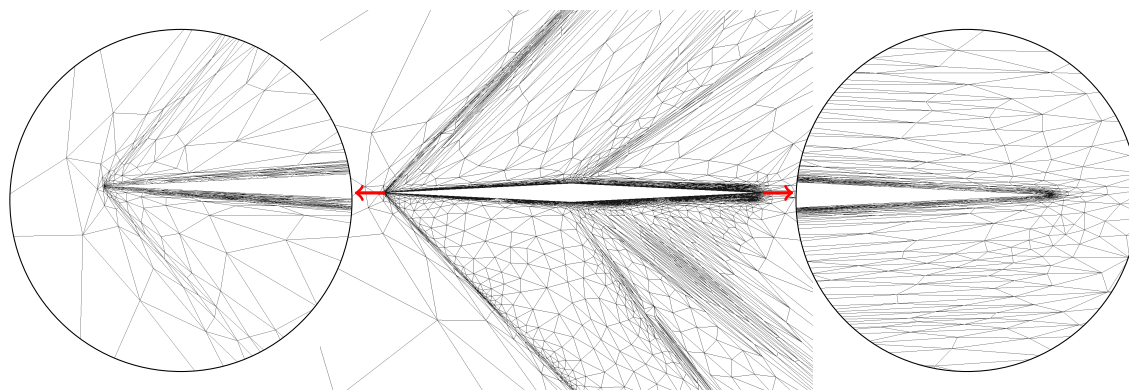


Figure 10. Diamond airfoil: final Hessian adapted mesh.

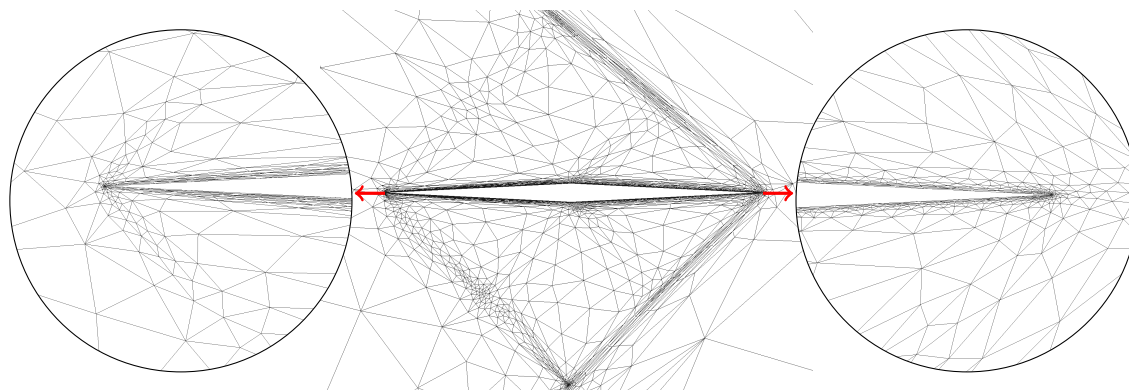


Figure 11. Diamond airfoil: final neural-network adapted mesh.

The flow over the diamond airfoil is supersonic, with oblique shocks emanating from the lead-

ing and trailing edges, and expansions originating from the mid-chord corners. The Hessian-based primal anisotropy detection approach places shock/expansion-aligned stretched elements in these areas. In contrast, MOESS symmetrically places anisotropic elements in these areas and the corresponding adjoint features, which mimic the primal shock/expansion structure but in reverse (right to left instead of left to right). The result is a balanced primal/adjoint resolving anisotropic mesh. Finally, the neural network anisotropy detection appears to prefer the adjoint features, as it places anisotropic elements mostly in these areas, with more isotropic (and less pronounced overall) resolution of the primal features.

We note the top/bottom asymmetry of the resolution is expected due to the presence of an angle of attack. In addition, all methods target the boundary layer with anisotropic elements, although the Hessian-based approach places more emphasis on anisotropy away from the wall near the trailing edge/wake compared to the other two methods. Combined with the excessive resolution of the shocks and expansions due to a primal-only anisotropy measure there, this leads to a less efficient distribution of degrees of freedom, and hence larger error for a given target `dof`.

### VII.C. MDA 30P/30N airfoil

The third test case consists of subcritical RANS flow over the McDonnell Douglas Aerospace (MDA) 30P/30N three-element airfoil at  $M_\infty = 0.2$ ,  $\alpha = 5^\circ$ , and  $Re = 5 \times 10^6$ . The computational domain consists of a C-shaped farfield boundary that is 100-200 main-element chords away from the airfoil. An adiabatic no-slip wall boundary conditions is imposed on the airfoil main element, the leading-edge slat, and the trailing-edge flap. The flow is subcritical and hence no shock capturing is employed. The output of interest is the lift coefficient on the airfoil. An Euler-flow adapted initial isotropic mesh of 3000 elements is constructed with sufficient boundary-layer resolution to enable a converged RANS solution.

Adaptive simulations are performed at  $p = 2$  solution approximation order for three target degree-of-freedom (`dof`) costs: 8000, 16000, 32000. At each target `dof`, 10 adaptive iterations are performed, using the final mesh from the previous target `dof` as the starting mesh. Again, MOESS, Mach-number Hessian anisotropy, and neural-network-augmented Hessian anisotropy methods are compared.

Figure 12 presents the output convergence history obtained from the adaptive runs, where again averaging has been performed over the last 4 iterations at each target `dof`. The “exact” value is also indicated, obtained from a four-times finer-mesh run at the same approximation order.

We see that the Hessian-based anisotropy detection method yields meshes that have the lift output closer to the exact value, although the prediction undershoots the exact value on the way to convergence. On the other hand, the neural network approach closely follows the MOESS results for the latter two `dof` targets, with monotonic convergence from below the exact value.

Figures 13–15 show the final adapted meshes at the highest target `dof` for the three adaptive approaches tested. The meshes among the three approaches are similar in this case. The most notable difference is the placement of anisotropy on the leading-edge stagnation streamline, which occurs in the MOESS and particularly the neural-network approach. The Hessian-based approach resolves this region instead with isotropic elements, as its anisotropy detection is insensitive to the adjoint variables. A similar situation occurs, in the extension of this streamline underneath the main element on the way to the flap leading edge: the MOESS and neural-network approaches employ anisotropic elements due to the adjoint anisotropy, whereas the Mach number Hessian approach uses isotropic elements.

In this high-lift configuration case, the stagnation streamline is a large adjoint feature exhibiting anisotropy. Both MOESS and the neural-network method target it with anisotropic elements, at a non-trivial `dof` cost due to small element size in the direction of large adjoint variation. If the



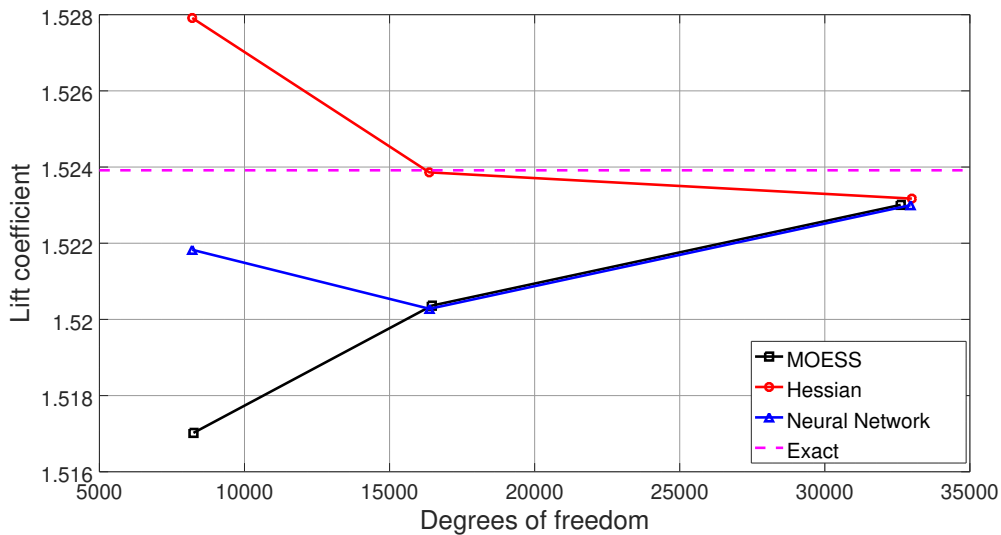


Figure 12. MDA 30P/30N,  $M_\infty = 0.2$ ,  $\alpha = 5^\circ$ ,  $Re = 5 \times 10^6$ : output convergence history.

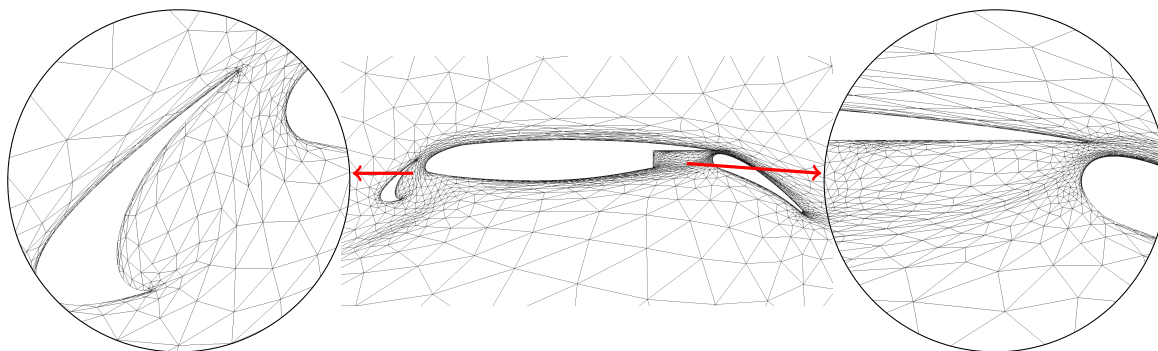


Figure 13. MDA 30P/30N airfoil: final MOESS adapted mesh.

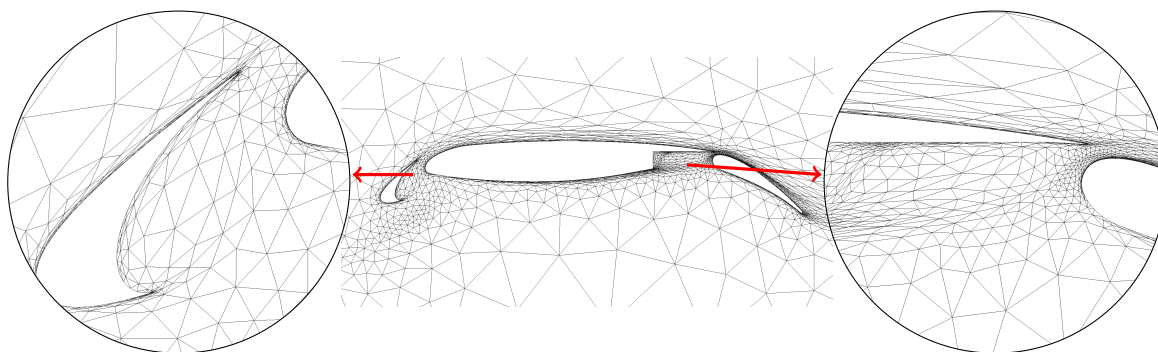


Figure 14. MDA 30P/30N airfoil: final Hessian adapted mesh.

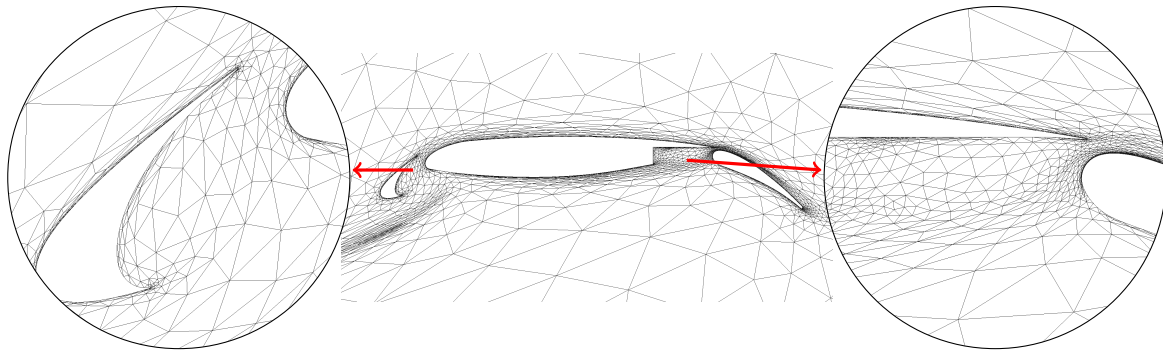


Figure 15. MDA 30P/30N airfoil: final neural-network adapted mesh.

full resolution of this feature is not in fact critical to the output, but instead an artifact, e.g. of an imperfect adjoint approximation, then additional cost savings in MOESS and the neural network may be possible. We do note, however, that the neural network approach does appear to favor anisotropy in adjoint features more than primal features, relative to the more balanced MOESS meshes.

#### VII.D. Extrapolation test: diamond airfoil at $M_\infty = 2.0$

Finally, we test the proposed neural network anisotropy detection approach on an “extrapolation” case, i.e. one for which the network was not trained. We choose the diamond airfoil geometry but at a higher Mach number,  $M = 2.0$ , and a different angle of attack,  $\alpha = 0$ . We also consider the drag output instead of the lift. Note that the training data only included one lift prediction simulation at a different flow condition. Aside from these changes, the rest of the case setup remains the same as described in Section VII.B.

Figure 16 presents the output convergence history obtained from the adaptive runs, where again averaging has been performed over the last 4 iterations at each target dof. The “exact” value is also indicated, obtained from a four-times finer-mesh run at the same approximation order.

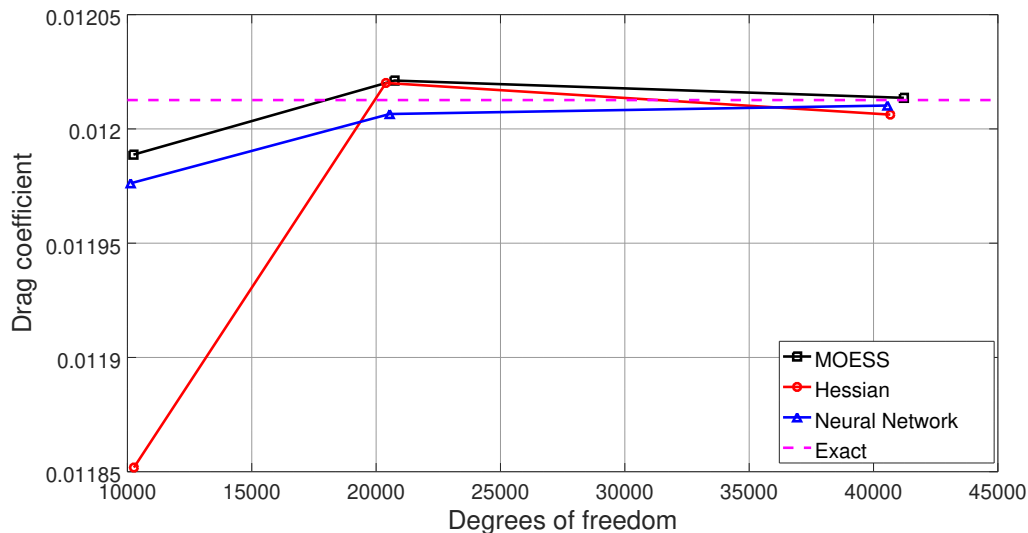


Figure 16. Diamond airfoil,  $M_\infty = 2.0$ ,  $\alpha = 0^\circ$ ,  $Re = 1 \times 10^6$ : output convergence history.

We see that at all target dof, the adaptive results using neural-network anisotropy are closer

to the exact value compared to those using the Mach number Hessian anisotropy. The difference is particularly noticeable on the coarser meshes, where the neural network performance is closer to MOESS. The improvement is on par with the previous results for cases within the training data.

Figure 17 shows the final adapted meshes at the highest target dof for the three adaptive approaches tested. As in the previous supersonic result, the meshes among the three approaches are drastically different in their placement of anisotropic elements: the Mach number Hessian approach focuses on the primal anisotropy, MOESS evenly addresses the primal and adjoint anisotropy, whereas the neural network focuses more on the adjoint anisotropy. With more work on the training and feature selection, the neural network mesh can likely be made to more closely follow the MOESS result.

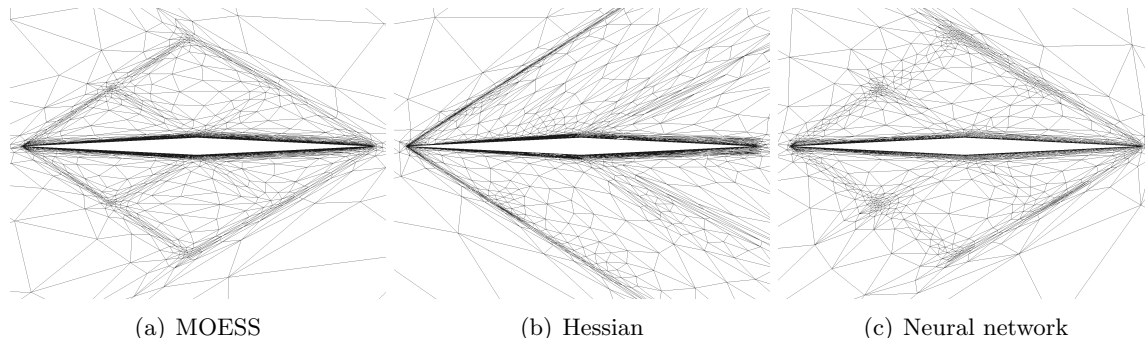


Figure 17. Diamond airfoil at  $M_\infty = 2.0$ : final adapted meshes.

## VIII. Conclusions

This paper introduces a machine-learning approach for determining the optimal anisotropy in a mesh, in the context of an output-based adaptive solution procedure. An artificial neural network is used to predict the desired element stretching ratio and direction from features of the adjoint solution. The network is trained to reproduce anisotropy calculated by MOESS, using features that are more easily calculated than the mesh-refinement sampling procedure of MOESS. These features consist of aspect ratio and direction data computed from the Hessian of each of the adjoint variables. They are invariant to scaling of the equations or outputs, and hence generalizable across different flow conditions and unit choices.

A variety of RANS cases, including transonic and supersonic flows, provides training data for the neural network, which consists of a simple structure with one small hidden layer of size twice that of the input layer. Training of the network yields a reduction in the mean-squared loss of about one order of magnitude, where the loss is measured over two independent components of the non-dimensional step matrix between the Hessian metric and the optimal MOESS metric.

The test cases compare the proposed neural-network anisotropy detection approach to the Mach number Hessian anisotropy detection and to MOESS. When used to drive adaptation, the neural-network approach yields output errors that more closely follow the MOESS results, and which are generally more accurate than the Hessian anisotropy approach. However, we note that this agreement with MOESS does not always extend to the computational meshes. The results indicate that the present neural network favors adjoint features over primal features, leading to meshes that anisotropically/heavily resolve leading-edge stagnation streamlines and adjoint shocks, at the expense of less anisotropy/resolution in primal-specific features such as shocks.

The improved anisotropy detection results hold not only for cases within the training data set, but also for an extrapolation simulation outside the training set, indicating the generalizability of

the approach. As only one neural network structure and feature choice is studied in the present work, a natural next step is to investigate the impact of the neural network architecture and input features on the mesh quality. By incorporating more primal features, it should be possible to shift the focus from adjoint-dominated anisotropy to a balanced combination, as observed in MOESS. This would then even better align the outputs of the network-driven results to MOESS. Furthermore, the network should be tested on general outputs, approximation orders, and flow conditions that are not part of the training data, to assess generalizability.

## Acknowledgments

The authors acknowledge support from the Department of Energy under grant DE-FG02-13ER26146/DE-SC0010341, and from The Boeing Company, with technical monitor Dr. Mori Mani.

## References

- <sup>1</sup>Castro-Diaz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., “Anisotropic unstructured mesh adaptation for flow simulations,” *International Journal for Numerical Methods in Fluids*, Vol. 25, 1997, pp. 475–491.
- <sup>2</sup>Buscaglia, G. C. and Dari, E. A., “Anisotropic mesh optimization and its application in adaptivity,” *International Journal for Numerical Methods in Engineering*, Vol. 40, No. 22, November 1997, pp. 4119–4136.
- <sup>3</sup>Habashi, W. G., Dompierre, J., Bourgault, Y., Ait-Ali-Yahia, D., Fortin, M., and Vallet, M.-G., “Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles,” *International Journal for Numerical Methods in Fluids*, Vol. 32, 2000, pp. 725–744.
- <sup>4</sup>Xia, G., Li, D., and Merkle, C. L., “Anisotropic Grid Adaptation on Unstructured Meshes,” AIAA Paper 2001-0443, 2001.
- <sup>5</sup>Venditti, D. A. and Darmofal, D. L., “Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows,” *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22–46.
- <sup>6</sup>Park, M. A., “Three-Dimensional Turbulent RANS Adjoint-Based Error Correction,” AIAA Paper 2003-3849, 2003.
- <sup>7</sup>Schall, E., Leservoisier, D., Dervieux, A., and Koobus, B., “Mesh adaptation as a tool for certified computational aerodynamics,” *International Journal for Numerical Methods in Fluids*, Vol. 45, No. 2, 2004, pp. 179–196.
- <sup>8</sup>Formaggia, L., Perotto, S., and Zunino, P., “An Anisotropic a posteriori error estimate for a convection-diffusion problem,” *Computing and Visualization in Science*, Vol. 4, 2001, pp. 99–104.
- <sup>9</sup>Fidkowski, K. J. and Darmofal, D. L., “A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations,” *Journal of Computational Physics*, Vol. 225, 2007, pp. 1653–1672.
- <sup>10</sup>Yano, M., Modisette, J., and Darmofal, D., “The Importance of mesh adaptation for higher-order discretizations of aerodynamics flows,” AIAA Paper 2011-3852, 2011.
- <sup>11</sup>Yano, M. and Darmofal, D., “An Optimization Framework for Anisotropic Simplex Mesh Adaptation: Application to Aerodynamic Flows,” AIAA Paper 2012-0079, 2012.
- <sup>12</sup>Yano, M., *An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.
- <sup>13</sup>Fidkowski, K. J., “A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization,” AIAA Paper 2016-0835, 2016.
- <sup>14</sup>Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science*, Ph.D. thesis, Harvard University, 1974.
- <sup>15</sup>LeCun, Y., Bengio, Y., and Hinton, G., “Learning representations by back-propagating errors,” *Nature*, Vol. 323, October 1986, pp. 533–536.
- <sup>16</sup>Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F., “Comprehensive Review of Neural Network-Based Prediction Intervals and New Advances,” *IEEE Transactions on Neural Networks*, Vol. 22, No. 9, Sep. 2011, pp. 1341–1356.
- <sup>17</sup>Shanmuganathan, S. and Samarasinghe, S., *Artificial Neural Network Modelling*, Springer International Publishing, Switzerland, 2016.

- <sup>18</sup>Faller, W. E. and Schreck, S. J., “Unsteady Fluid Mechanics Applications of Neural Networks,” *Journal of Aircraft*, Vol. 34, No. 1, 1997, pp. 48–55.
- <sup>19</sup>Huang, R., Hu, H., and Zhao, Y., “Nonlinear Reduced-Order Modeling for Multiple-Input/Multiple-Output Aerodynamic Systems,” *AIAA Journal*, Vol. 52, No. 6, 2014, pp. 1219–1231.
- <sup>20</sup>Reed, W. and Hill, T., “Triangular Mesh Methods for the Neutron Transport Equation,” Los Alamos Scientific Laboratory Technical Report LA-UR-73-479, 1973.
- <sup>21</sup>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.
- <sup>22</sup>Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., “ $p$ -Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations,” *Journal of Computational Physics*, Vol. 207, 2005, pp. 92–113.
- <sup>23</sup>Allmaras, S., Johnson, F., and Spalart, P., “Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model,” Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902, 2012.
- <sup>24</sup>Ceze, M. A. and Fidkowski, K. J., “High-Order Output-Based Adaptive Simulations of Turbulent Flow in Two Dimensions,” AIAA Paper 2015-1532, 2015.
- <sup>25</sup>Roe, P., “Approximate Riemann solvers, parameter vectors, and difference schemes,” *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.
- <sup>26</sup>Bassi, F. and Rebay, S., “Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes, equations,” *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 197–207.
- <sup>27</sup>Becker, R. and Rannacher, R., “An optimal control approach to a posteriori error estimation in finite element methods,” *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.
- <sup>28</sup>Fidkowski, K. J. and Darmofal, D. L., “Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics,” *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694.
- <sup>29</sup>Fidkowski, K., “High-Order Output-Based Adaptive Methods for Steady and Unsteady Aerodynamics,” *37<sup>th</sup> Advanced CFD Lectures series; Von Karman Institute for Fluid Dynamics (December 9–12 2013)*, edited by H. Deconinck and R. Abgrall, von Karman Institute for Fluid Dynamics, 2013.
- <sup>30</sup>Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C., “Adaptive remeshing for compressible flow computations,” *Journal of Computational Physics*, Vol. 72, 1987, pp. 449–466.
- <sup>31</sup>Borouchaki, H., George, P., Hecht, F., Laug, P., and Saltel, E., “Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes,” INRIA-Rocquencourt, France. Tech Report No. 2741, 1995.
- <sup>32</sup>Penec, X., Fillard, P., and Ayache, N., “A Riemannian framework for tensor computing,” *International Journal of Computer Vision*, Vol. 66, No. 1, 2006, pp. 41–66.
- <sup>33</sup>Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,” 2015, Software available from tensorflow.org.
- <sup>34</sup>Persson, P.-O. and Peraire, J., “Sub-cell shock capturing for discontinuous Galerkin methods,” AIAA Paper 2006-112, 2006.