Output-Based Mesh Optimization Using Metric-Conforming Node Movement

Krzysztof J. Fidkowski*
University of Michigan, Ann Arbor, MI, 48188

This paper presents a two-step method for performing output-based mesh adaptation using node movement, for high-order discretizations. In the first step, an optimal target metric is determined over the current mesh using an optimization procedure that equidistributes a marginal output error to degree-of-freedom cost. In the second step, the nodes are moved so as to minimize the error between the mesh-implied metric and the target metric, i.e. to make the mesh better conform to the target metric. This movement is the solution of a global optimization problem, made possible by virtue of the meshes being coarse for high-order methods. No additional flow solutions or error calculations take place during this second optimization step, and hence the optimization is not computationally expensive. Results for three diverse cases demonstrate the ability of the mode movement strategy to reduce the output error relative to a given initial mesh, at a level that in one case surpasses that of a baseline re-meshing strategy.

I. Introduction

Error estimation and mesh adaptation improve the accuracy and robustness of numerical simulations [1]. Numerous types of error estimates exist, ranging from heuristic feature-based measures [2–5] to more rigorous output-based ones [1, 6–8], with the latter showing the most promise for convection-dominated aerodynamic problems. Similarly, many adaptive methods have been studied, including local mesh modification [9–15], order adaptation [16–18], and global re-meshing [19–22]. While re-meshing has been the method of choice for recent works in metric-based mesh optimization [23, 24], it is not always robust or even tractable, especially for three-dimensional meshes with curved, anisotropic elements. In this work we therefore investigate another adaptive approach, based on node movement.

Node movement, also called r-refinement [25–27], is not a novel concept. In many previous works it has been used alone or in combination with other local mesh operators. While less flexible than global re-meshing in terms of size control and degree-of-freedom distribution, it has the advantage of improved robustness, since small changes can be undone or addressed locally to prevent inversion errors.

This paper presents a metric-based node movement approach for high-order discretizations. The novelty of the work is in the use of a metric field error, instead of a direct solution or output error, to drive the node movement. The approach is also formulated as a global optimization problem for the node coordinates. Whereas such an optimization is generally not tractable for large-scale problems discretized using second-order methods, where the number of nodes can range in the millions or even billions, high-order methods can yield excellent accuracy on much coarser meshes [28]. In addition, we employ an iterative optimization strategy and are not concerned with perfect mesh optimality.

We foresee the application of the node movement approach to cases where global re-meshing is difficult or not possible, for example for structured meshes [29], for which the connectivity must remain the same. Another application is in concurrent shape and mesh optimization [30], where smooth variations of the mesh are desired. Finally, three-dimensional problems where high-order curved re-meshing is not yet automated would benefit from adaptation through node movement.

The node movement strategy is presented for a discontinuous Galerkin finite-element discretization, which is reviewed briefly in Section II. Section III presents the output error estimation procedure, and Section IV presents the metric optimization algorithm. Section V describes the node-movement strategy, which is the new contribution of this work. Results in Section VI demonstrate the error reduction attained with node movement and compare it to global re-meshing. Section VII concludes with a summary of the work and possible future directions.

*Professor, Department of Aerospace Engineering, AIAA Associate Fellow.

II. Discretization

A. Governing Equations

Consider a system of unsteady partial differential equations in conservative form,

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

where $\mathbf{u} \in \mathbb{R}^s$ is the s-component state vector, $\vec{\mathbf{F}} \in \mathbb{R}^{\dim \times s}$ is the flux vector, dim is the spatial dimension, and $\mathbf{S}$ is a source term.

For scalar advection-diffusion, $s = 1$, and the flux is $\vec{F} = \vec{V}u - \mu\nabla u$, where $\vec{V}$ is the velocity and μ is the viscosity. The source term is nonzero in this work when prescribing a manufactured solution. For the Navier-Stokes equations, we use a conservative state vector $\mathbf{u} \in \mathbb{R}^{\dim+2} = [\rho u, \rho\vec{V}, \rho E]$, where ρ is the density, $\vec{V}$ is the velocity, and E is the total energy per unit mass. The source term is nonzero when modeling turbulence using Reynolds averaging, for which an additional equation is included [31, 32].

B. The Discontinuous Galerkin Method

The discontinuous Galerkin (DG) method [33–35] is a finite-element discretization in which the approximation space is discontinuous across element boundaries. On each element, Ω_e, the state is spatially approximated by $\mathbf{u}_h$, a linear combination of polynomial basis functions of order p. Multiplying Eqn. 1 by test functions in the same space as the solution and integrating by parts to couple elements via fluxes, here the Roe [36] convective flux and the second form of Bassi and Rebay (BR2) [37] for the viscous flux, yields the weak form. Choosing a polynomial basis then yields a system of ordinary differential equations, $\mathbf{R}(\mathbf{U}) = \mathbf{0}$, where $\mathbf{U} \in \mathbb{R}^N$ is the discrete state vector, and N is the total number of unknowns. We solve this system using a Newton-Raphson method with pseudo-time continuation [38]and the generalized minimum residual (GMRES) [39] linear solver, preconditioned by an element-line Jacobi smoother with a coarse-level ($p = 1$) correction [35, 40].

C. The Output Adjoint

The adjoint solution for a scalar output is the sensitivity of the output to residual source perturbations [1, 6]. Only steady adjoint solutions are required in the present work, and these are much cheaper to compute than unsteady adjoints. Linearizing the residual and scalar output, $J(\mathbf{U})$, yields a linear system for the adjoint coefficients, $\mathbf{\Psi} \in \mathbb{R}^N$,

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)^T \mathbf{\Psi} + \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T = \mathbf{0}, \tag{2}$$

This equation is solved using a transposed version of the preconditioned-GMRES method used in the Newton-Raphson primal solver. The adjoint solution plays an important role in output-based error estimation and mesh adaptation because discretization errors originate as residual sources. The adjoint, as a residual sensitivity, weights the adjoint to produce the output error.

III. Output Error Estimation

We use an adjoint solution to estimate the numerical error in the corresponding output of interest, J, through the adjoint-weighted residual [1, 6]. Let H denote a coarse/current discretization space, and h a fine one, here obtained by increasing the approximation order by one, $p \to p + 1$. Denote by $\mathbf{U}_h^H$ the state prolongated from the coarse to the fine space. Computing the fine-space residual using the prolongated state and weighting it by the fine-space adjoint error gives an estimate of the output error between the coarse and fine spaces,

$$J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx -\delta\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H). \tag{3}$$

We obtain the adjoint error, $\delta\mathbf{\Psi}_h$, by subtracting from the fine-space adjoint, solved exactly, its projection onto the coarse space. The error estimate in Eqn. 3 is localized to elemental contributions, resulting in the elemental error indicators ε_e

$$J \approx \sum_{e=1}^{N_e} -\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H) \quad \Rightarrow \quad \varepsilon_e \equiv \left|\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)\right|, \tag{4}$$

where N_e is the number of elements, and the subscript e denotes components of the adjoint and residual associated with element e.

IV. Metric Optimization

The elemental error indicator together with additional error indicators evaluated on sub-elements, via Eqn. 4 with projected adjoints, drive a metric optimization calculation based on MOESS: mesh optimization through error sampling and synthesis [24, 41]. The optimized metric can then be used in mesh adaptation, e.g. through re-meshing or node movement.

A Riemannian metric field, $\mathcal{M}(\vec{x})$, encodes information about the desired size and stretching of the elements in a computational mesh. At each point $\vec{x}$ in physical space, the symmetric positive definite metric tensor $\mathcal{M}(\vec{x}) \in \mathbb{R}^{\dim\times\dim}$ provides a yardstick for measuring a non-dimensional distance from $\vec{x}$ to a nearby, infinitesimally-close, point $\vec{x} + \delta\vec{x}$,

$$\delta\ell = \sqrt{\delta\vec{x}^T \mathcal{M} \delta\vec{x}}. \tag{5}$$

The set of points at unit metric distance from $\vec{x}$ is an ellipse, the principal axes and lengths of which are determined by the eigenvectors and eigenvalues of $\mathcal{M}$.

Affine-invariant changes [42] to the metric field are made via a symmetric *step matrix*, $\mathcal{S} \in \mathbb{R}^{d\times d}$, according to

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S}) \mathcal{M}_0^{\frac{1}{2}}. \tag{6}$$

Note that $\mathcal{S} = 0$ leaves the metric unchanged, while diagonal values in $\mathcal{S}$ of change the metric stretching sizes.

The mesh optimization algorithm requires models for how the elemental error indicator and the cost change as the metric changes. The error model is

$$\varepsilon_e = \varepsilon_{e0} \exp\left[\mathrm{tr}(\mathcal{R}_e \mathcal{S}_e)\right], \tag{7}$$

where ε_{e0} is the initial error on element e, and $\mathcal{R}_e$ is an element-specific rate tensor determined through a sampling procedure. The cost model is

$$C_e = C_{e0} \exp\left[\frac{1}{2}\mathrm{tr}(\mathcal{S}_e)\right], \tag{8}$$

where C_{e0} is the initial cost, measured in terms of degrees of freedom, `dof`. This model is purely volume based: the trace of the step matrix governs the change in an element's volume, and hence the number of degrees of freedom occupying the original volume. Details of both of these models can be found in previous work [24].

Given a current mesh with its mesh-implied metric ($\mathcal{M}_0(\vec{x})$), elemental error indicators ε_{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. This algorithm iteratively equidistributes the marginal error to marginal cost ratio over the mesh elements. In practice, the mesh optimization and flow/adjoint solution are performed several times at a given target cost, C_{target}, until the error stops changing. Then the target cost is increased to reduce the error further if desired.

V. Adaptation Using Node Movement

The result of MOESS is a step matrix field that can be used to calculate a desired metric field, which can then be used to generate a new mesh, using the current mesh as a background scaffold on which the metric is defined and interpolated. Alternatively, the current mesh can be altered in a local fashion to better conform to the prescribed metric. This latter approach is the one taken in the present work.

A. Problem Formulation

Given a target step matrix field, $\mathcal{S}^{\text{tgt}}(\vec{x})$, we pose an optimization problem to reposition the node coordinates of the current mesh so that the step matrix between the current and new mesh, $\mathcal{S}(\vec{x})$, closely matches $\mathcal{S}^{\text{tgt}}(\vec{x})$. We currently do not apply any other operations on the mesh, such as edge swapping/collapsing or node insertion, but these could be considered, at least for unstructured meshes, in future work. The connectivity of the mesh therefore remains constant.

Let $\mathcal{M}_{e0}$ be the current mesh-implied metric [41] on element e. This set of tensors over all elements is computed from the current mesh node coordinates, $\mathbf{x}_0 \in \mathbb{R}^{N_x \times \text{dim}}$, where N_x is the number of nodes in the mesh. The new mesh is obtained by displacing the node coordinates by $\Delta\mathbf{x} \in \mathbb{R}^{N_x \times \text{dim}}$, and we denote the new mesh coordinates by $\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x}$. On element e, the displacement creates a new mesh-implied metric, $\mathcal{M}_e(\Delta\mathbf{x})$. The step matrix between the current and new mesh-implied metric is

$$\mathcal{S}_e(\Delta\mathbf{x}) \equiv \mathcal{S}(\mathcal{M}_{e0}, \mathcal{M}_e(\Delta\mathbf{x})) = \log\left(\mathcal{M}_{e0}^{-1/2}\mathcal{M}_e(\Delta\mathbf{x})\mathcal{M}_{e0}^{-1/2}\right). \tag{9}$$

Our goal in optimization is to determine the $\Delta\mathbf{x}$ that minimizes the difference between this step matrix and the target step matrix over each element, $\mathcal{S}_e^{\text{tgt}}$. In MOESS, the elemental target step matrix is obtained from the nodal target step matrix by arithmetically averaging to elements [24, 41].

The optimization problem for the mesh node coordinate displacement reads

$$\Delta\mathbf{x} = \arg\min_{\Delta\mathbf{x}} J^{\text{metric}}(\Delta\mathbf{x}) \equiv \sum_{e=1}^{N_e} \frac{1}{2}\left\| \mathcal{W}_e \circ \left(\mathcal{S}_e(\Delta\mathbf{x}) - \mathcal{S}_e^{\text{tgt}}\right)\right\|_F^2 + \sum_{f=1}^{N_f} \frac{\gamma}{2}\left\|\mathcal{S}_{f^+}^{I}(\Delta\mathbf{x}) - \mathcal{S}_{f^-}^{I}(\Delta\mathbf{x})\right\|_F^2, \tag{10}$$

where N_e is the number of elements, $\|\cdot\|_F$ is the Frobenius norm, N_f is the number of faces, and f^+/f^- denote the two elements adjacent to face f. For an element e, $\mathcal{S}_e^I$ is the step matrix between the identity metric, i.e. reference element, and the new metric, which reduces to the logarithm of the new metric, $\mathcal{S}_e^I(\Delta\mathbf{x}) = \mathcal{S}(\mathcal{I}, \mathcal{M}_e(\Delta\mathbf{x})) = \log(\mathcal{M}_e(\Delta\mathbf{x}))$.

The first term in the objective, Eqn. 10, measures the difference between the realized and target step matrices, and reducing this difference is the primary goal of the optimization. The second term, which consists of a sum over faces of step matrix differences between adjacent elements, regularizes the optimization problem by preventing large changes in element size and shape. The non-dimensional regularization parameter γ controls the amount of regularization, and in this work we have obtained good results on all problems with $\gamma = .03$.

The tensor $\mathcal{W}_e$ in Eqn. 10 is an optional, non-dimensional weight on step matrix errors. The Hadamard product, $\circ$, applies the weight component-wise. In this work we use a normalized linearization of the adaptation output error with respect to the step matrix, $\mathcal{W}_e = \partial\mathcal{E}_e/\partial\mathcal{S}_e$, which is available from MOESS. The normalization consists of taking the absolute value of the tensor entries, to prevent cancellation of errors, and scaling them to the range $[\delta, 1]$, where $\delta > 0$ prevents zero weight on entries of the step matrix to which the adaptation error is least sensitive.

A constraint on the optimization is that the geometry mapping Jacobian determinant remains positive, i.e. no negative volumes. Presently we do not formally impose this constraint in the optimization problem and instead only check for it during the line search when choosing the step length. Small volumes correspond to large-magnitude step matrices, and hence, in general, the optimization naturally steers away from negative volumes.

As the mesh changes during the solution of the optimization problem, elements move to new locations, and hence their target step matrices change. This effect is not captured in the above formulation, where we assume that each element has a constant target step matrix, $\mathcal{S}_e^{\text{tgt}}$. Including it would require interpolating the target step matrix to elements of the new mesh and linearizing the objective for the gradient calculation. However, we have found that in an iterative solution procedure with multiple solutions and adaptations, the mesh converges even with the constant target per optimization.

B. Gradient Calculation

For a linear triangular element, the mesh-implied metric is calculated by requiring unit measure of the three edges. Let $\vec{x}_i$ be the coordinate of triangle vertex i. Define the edge vector i as $\vec{e}_i \equiv \vec{x}_{\text{mod}(i+1,3)} - \vec{x}_i = [\Delta x_i, \Delta y_i]$. Unit measure of edge i can then be expressed as

$$1 = \vec{e}_i^T \mathcal{M}\vec{e}_i = (\Delta x_i)^2 a + 2\Delta x_i \Delta y_i b + (\Delta y_i)^2 c, \tag{11}$$

where $\mathcal{M} = [a, b; b, c]$ is the metric representation in the chosen coordinate system. Writing this equation for each of the three edges then yields a 3×3 linear system of equations for the metric entries $\mathbf{a} \equiv [a, b, c]^T$:

$$\underbrace{\begin{bmatrix} (\Delta x_1)^2 & 2\Delta x_1 \Delta y_1 & (\Delta y_1)^2 \\ (\Delta x_2)^2 & 2\Delta x_2 \Delta y_2 & (\Delta y_2)^2 \\ (\Delta x_3)^2 & 2\Delta x_3 \Delta y_3 & (\Delta y_3)^2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{\mathbf{a}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{12}$$

A gradient-based solution of the optimization problem in Eqn. 10 requires the derivatives of the metric entries with respect to the vertex coordinates, $\partial\mathbf{a}/\partial\vec{x}_k$. These can be obtained by differentiating Eqn. 12 with respect to $\vec{x}_k$,

$$\frac{\partial\mathbf{A}}{\partial\vec{x}_k}\mathbf{a}+\mathbf{A}\frac{\partial\mathbf{a}}{\partial\vec{x}_k}=0 \quad\Rightarrow\quad \frac{\partial A_{il}}{\partial\vec{x}_k}a_l+A_{ij}\frac{\partial a_j}{\partial\vec{x}_k}=0 \quad\Rightarrow\quad \frac{\partial a_j}{\partial\vec{x}_k}=-(\mathbf{A}^{-1})_{ji}\frac{\partial A_{il}}{\partial\vec{x}_k}a_l. \tag{13}$$

The entries of $\frac{\partial\mathbf{A}}{\partial\vec{x}_k}$ are calculated in a straightforward manner from the expression for $\mathbf{A}$ in Eqn. 12. The resulting derivatives of a, b, c with respect to $\vec{x}_k$ can be assembled into the tensor $\partial\mathcal{M}/\partial\vec{x}_k$.

The derivative of the mesh-implied metric is required in the calculation of the step matrix in Eqn. 9, the Frobenius norm of which contributes to the objective. Denoting the argument of $\log(\cdot)$ in Eqn. 9 by $\mathcal{T}_e$, we have

$$\mathcal{T}_e\equiv\mathcal{M}_{e0}^{-1/2}\mathcal{M}_e(\mathbf{x})\mathcal{M}_{e0}^{-1/2} \quad\Rightarrow\quad \frac{\partial\mathcal{T}_e}{\partial\vec{x}_k}=\mathcal{M}_{e0}^{-1/2}\frac{\partial\mathcal{M}_e}{\partial\vec{x}_k}\mathcal{M}_{e0}^{-1/2}. \tag{14}$$

Unfortunately, the matrix logarithm does not lend itself to a straightforward linearization when $\mathcal{T}_e$ and its derivative do not commute, which will generally be the case. As such, we use finite differences to compute this derivative, in an efficient manner only for $\partial\mathcal{S}_e/\partial\mathcal{T}_e$, which requires three finite differences (in two dimensions). The chain rule then gives

$$\frac{\partial\mathcal{S}_e}{\partial\vec{x}}=\frac{\partial\mathcal{S}_e}{\partial\mathcal{T}_e}\frac{\partial\mathcal{T}_e}{\partial\vec{x}_k}. \tag{15}$$

From Eqn. 10, the gradient of the optimization objective function, with respect to $\mathbf{x}$, all of the vertex coordinates of the mesh, is assembled via a summation over elements and faces,

$$\frac{\partial J^{\text{metric}}}{\partial\mathbf{x}}=\sum_{e=1}^{N_e}\left[\mathcal{W}_e\circ\left(\mathcal{S}_e-\mathcal{S}_e^{\text{tgt}}\right)\right]:\left(\mathcal{W}_e\circ\frac{\partial\mathcal{S}_e}{\partial\mathbf{x}}\right)+\sum_{f=1}^{N_f}\gamma\left(\mathcal{S}_{f^+}-\mathcal{S}_{f^-}\right):\left(\frac{\partial\mathcal{S}_{f^+}}{\partial\mathbf{x}}-\frac{\partial\mathcal{S}_{f^-}}{\partial\mathbf{x}}\right). \tag{16}$$

In the present work, the implementation of the gradient was verified by finite-difference comparison using a random perturbation to the node coordinates.

C. Boundary Nodes

Whereas interior mesh nodes can move in all dim directions, boundary mesh nodes will be constrained to move in $0-(\dim-1)$ directions. To keep the optimization problem itself unconstrained, we reduce the degrees of freedom of boundary nodes according to the type of node. In two dimensions, most boundary nodes will have one degree of freedom, so that they can slide along their boundary edge. Corners, defined as points across which the boundary normal changes by a threshold angle or as points on two different boundary groups, are fixed and hence have zero degrees of freedom.

For each node, i, boundary or interior, we define the number of degrees of freedom of its motion, $n_i^{\mathtt{dof}}$, and a basis for its motion, $\{\vec{\phi}_j\}_i$, where $1\le j\le n_i^{\mathtt{dof}}$. For general boundary nodes, the normal direction is removed from the basis of its motion. Corner nodes cannot move, $n^{\mathtt{dof}}=0$, and edge nodes in three dimensions can only move in one direction. Denoting by $\mathbf{y}$ the vector of coefficients on the basis vectors, over all nodes, we map the gradient in Eqn. 16 to $\partial J^{\text{metric}}/\partial\mathbf{y}$ and use this gradient in the optimization:

$$\frac{\partial J^{\text{metric}}}{\partial\mathbf{y}}=\frac{\partial J^{\text{metric}}}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\mathbf{y}}. \tag{17}$$

The definition of the motion basis is straightforward for linear boundary edges, but it requires more attention for curved edges, in order to maintain geometric fidelity. On a curved edge, the motion of a boundary point is along the curve defined by the high-order nodes that make up the adjacent edges, and the motion basis (i.e. the direction) changes with the node's position. Figure 1 illustrates the setup of this motion.

The variable y is defined as the arc-length along the curved edge, and separate arc-lengths are measured for positive and negative y, as the two directions correspond to two different edges. When the node is moved, its new position is computed by "snapping" it to the curved edge, which is done by evaluating the interpolant of the high-order nodes for the given y: $y_{\max}$ corresponds to movement across the entire edge on the $y>0$ side, whereas $y_{\min}$ corresponds to movement along the edge for the $y<0$ side. We note that this motion only pertains to the element vertices (the "linear" nodes), as high-order nodes are handled separately and are not part of the optimization.

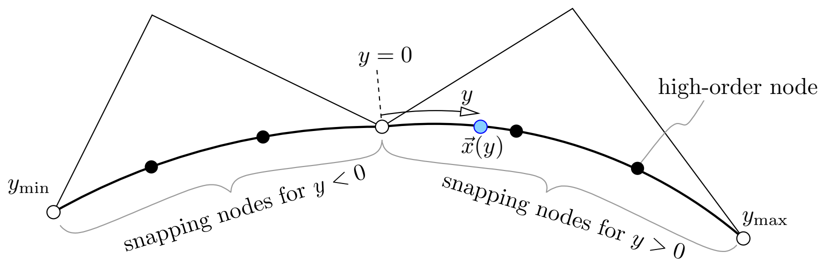


Fig. 1 Motion of a boundary node on a curved boundary in two dimensions.

D. Iterative Optimization

The optimization problem in Eqn. 10 can be solved using an iterative technique, such as steepest descent or the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [43]. In the current work, we use the latter, with a history of 20 updates for approximating the inverse Hessian, and 100 iterations are taken. Each optimization iteration requires evaluation of the objective and its gradient, and a line search that consumes several more objective evaluations. However, as no system solution is needed for the objective evaluation, the optimization iterations are rapid, so that the dominant cost remains the flow and adjoint solutions in the outer mesh adaptation loop. Furthermore, as the solution approximation order increases, relatively coarser meshes are needed, making the node optimization even cheaper.

The result of the optimization is a vector of node displacements, $\Delta\mathbf{x} = \{\Delta\vec{x}_i\}$, or equivalently $\Delta\mathbf{y}$ in the case of boundary constraints, which can be mapped to $\Delta\mathbf{x}$, as described in Section V.C. To prevent negative volumes and to determine the initial step size for the first line search, this update is limited by measuring the node displacement using the mesh-implied metric of each adjacent element,

$$\Delta l_e \equiv \|\Delta\vec{x}_i^T \mathcal{M}_e \Delta\vec{x}_i\|. \tag{18}$$

Recalling that a metric measure is dimensionless, and that all element edges have unit measure under the mesh-implied metric, we require that $\Delta l_e < f$, where f is a constant, here chosen as $f = 0.5$. Negative volumes can still occur and they are handled through backtracking during the line search.

E. High-order nodes

Discretizations that approximate the solution at high order require curved elements for boundary representation [44], which means additional nodes per element beyond the linear, or $q = 1$, vertices. The optimization problem can be applied to these nodes as well, by extending the mesh-implied metric and gradient calculations presented in Section V.B to include the high-order nodes. For the present work, however, we only optimize the location of the $q = 1$ nodes. To avoid re-curving, the displacements of the high-order nodes are calculated by evaluating the $q = 1$ displacement field at the high-order node locations. Including the high-order nodes in the optimization problem will be investigated in future work.

VI. Results

This section presents results of applying the metric-based node optimization. Following a flow solution, MOESS is applied to obtain a target metric field. This field is then used to move the nodes so as to make the mesh-implied metric conform to the target. The optimization procedure is inexpensive as it does not require additional flow solutions or adjoints: the target metric field remains fixed throughout the optimization. The process then repeats starting with the optimized mesh, which will in general yield a different solution and target metric. The results of the node-movement adaptation are compared to MOESS with metric-based re-meshing [45].

A. L_2 Error Minimization

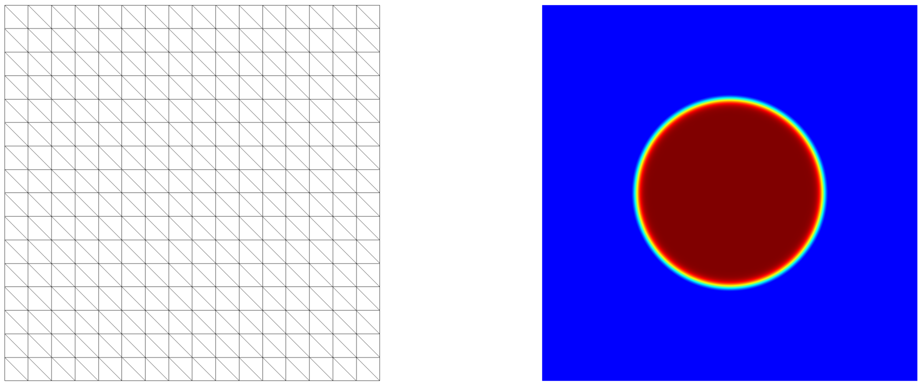
In this problem we consider a manufactured solution of the two-dimensional scalar advection-diffusion problem,

$$u(x, y) = 0.5\,(1 - \tanh(40(r - 0.5)))\,, \tag{19}$$

over a square domain $\vec{x} \in [-1, 1]^2$, where $r^2 = x^2 + y^2$. The Péclet number for the advection-diffusion problem is 10 based on a unit distance. The output of interest is the L_2 error between the manufactured and approximated solutions,

$$E_{L_2} \equiv \|u(x, y) - u_h(x, y)\|_2. \tag{20}$$

Figure 2 shows the initial mesh and manufactured solution. The initial mesh has 512 triangles, generated from a structured 16×16 mesh in which each square is split into two triangles. The solution approximation order is $p = 2$, and all elements are linear, $q = 1$. The manufactured solution exhibits sharp variation from near 1 to near 0 at $r = 0.5$, and this is the area that we expect adaptation to target.



(a) Initial mesh (512 elements) (b) Manufactured solution

Fig. 2 Initial mesh and manufactured solution for the scalar case.

Figure 3 shows the meshes obtained by the presented node-movement strategy, and by the baseline metric-based re-meshing using BAMG. These are shown at *outer* adaptive iterations, each of which consists of solving the scalar advection diffusion problem with the manufactured solution source term, computing the optimal step matrix field using MOESS, and adapting the mesh, either via node-movement or re-meshing. The node-movement adaptation itself consists of multiple L-BFGS iterations, as described in Section V.D. The re-meshing is performed at a constant (approximately) degrees of freedom equal to that of the initial mesh.

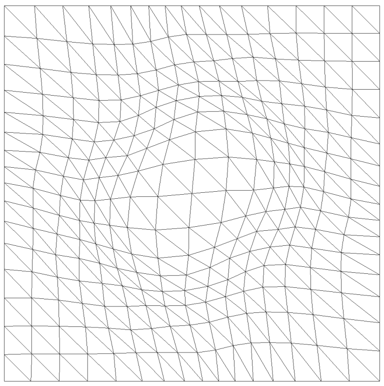
In the node-movement meshes, we see a gravitation of the nodes to the $r = 0.5$ circle, where the elements also become stretched to efficiently approximate the sharp variation of the scalar in only the radial direction. Comparing to 5th iteration meshes, limitations of the fixed-topology in the node-movement strategy become evident: elements outside of the circle become stretched to enable adequate resolution at the circle, but this stretching is not necessary for approximating the nearly-constant scalar field there. More efficient meshes could be obtained by including mesh topology changes, such as edge swapping and collapse.

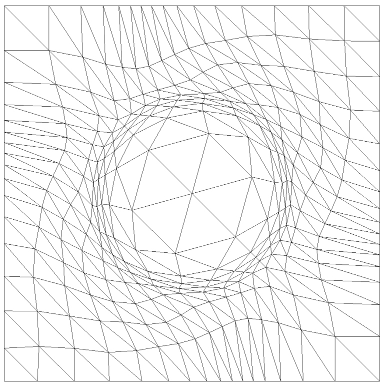
Figure 4 shows the convergence of the output, which is the L_2 error norm between the solution on a given adapted mesh and the analytical manufactured solution. We see similar errors on the first two adaptive iterations, after which the node-movement strategy stagnates, while the re-meshing strategy continues to a lower error value before also stagnating. The stagnation is expected at a fixed degrees of freedom, and the higher errors with node movement are likely due to the sub-optimality of the fixed mesh topology.

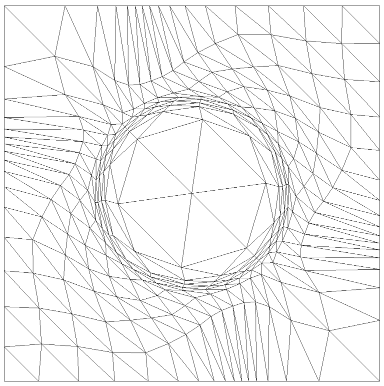
B. Transonic Airfoil

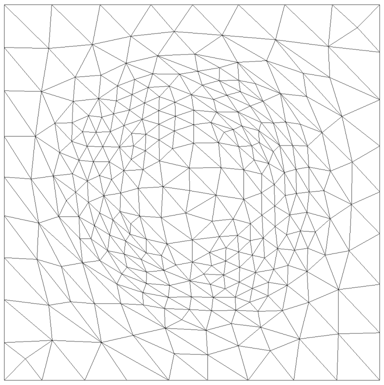
This case consists of the NACA 0012 airfoil at free-stream Mach number of $M_\infty = 0.8$ and angle of attack of $\alpha = 1.25°$. The initial mesh, shown in Figure 5, has 632 elements, curved to $q = 3$ geometry order at the airfoil boundary. This figure also shows the Mach number contours, which exhibit a strong shock on the upper surface and a weaker shock on the lower surface. The solution approximation order is set to $p = 2$, and element-wise constant artificial-viscosity is used to capture the shocks [46].

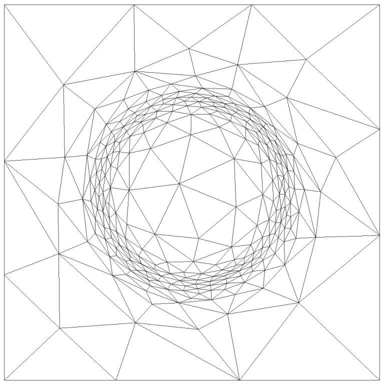
The adaptation was performed with the drag on the airfoil as the target output. Figure 6 shows the adapted meshes using node movement and re-meshing. We see that the adaptation focuses on the foot of the shocks and the leading and trailing edges. The strong upper surface shock itself is not targeted for heavy adaptation, although the element edges


(a) Node movement, iteration 1


(b) Node movement, iteration 3


(c) Node movement, iteration 5


(d) Re-meshing, iteration 1


(e) Re-meshing, iteration 3

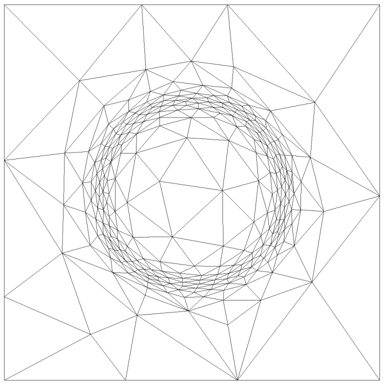

(f) Re-meshing, iteration 5

Fig. 3 Meshes at various outer adaptation iterations for the presented node movement strategy and the baseline metric-based re-meshing, for the scalar case.

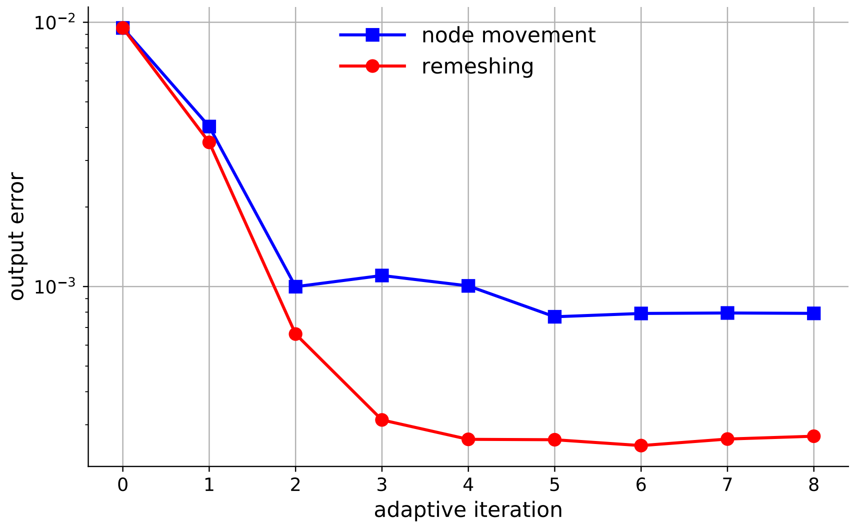
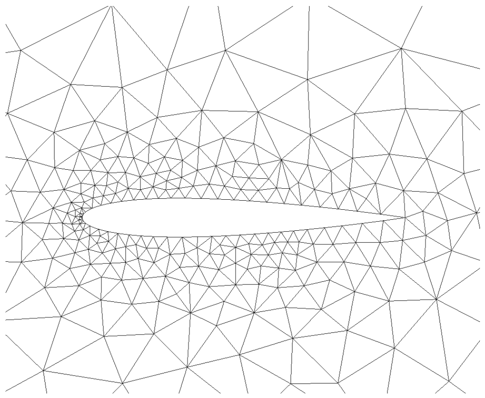


Fig. 4 The analytic L_2 error norm output convergence for the scalar case.


(a) Initial mesh (632 elements)

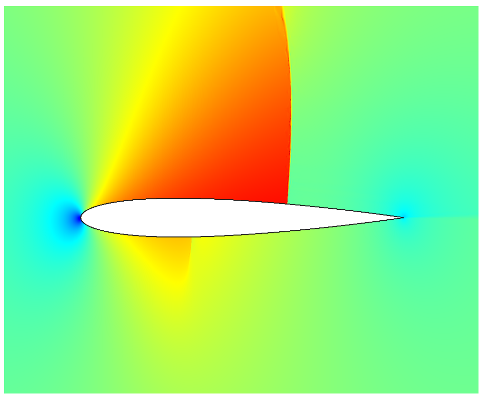
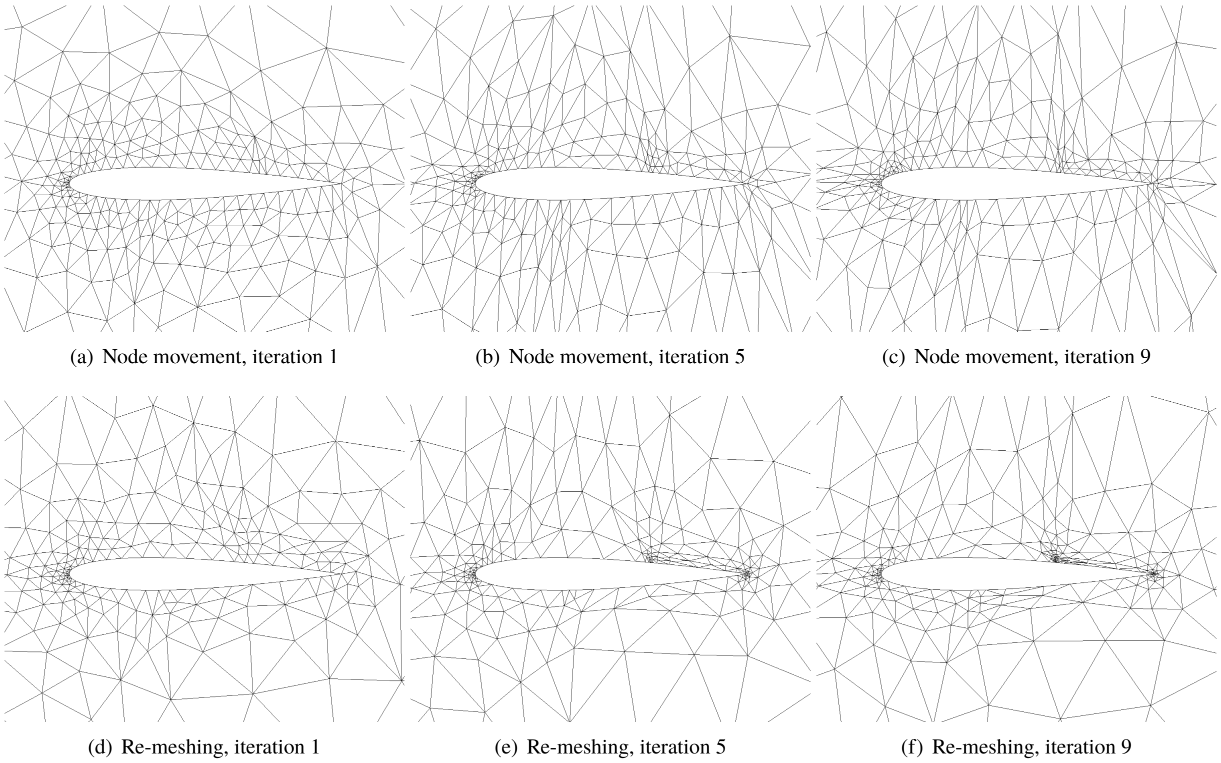

(b) Mach number (0-1.6)

Fig. 5 Initial mesh and Mach contours for the transonic airfoil case.

appear aligned to the shock by the 9th adaptive iteration, notably in the node movement strategy. The region behind the shock, near the airfoil surface, is also targeted for refinement, more so in the case of MOESS. This is likely due to a spurious entropy trail caused by the artificial-viscosity shock capturing, which is strongest at the foot of the shock.


(a) Node movement, iteration 1 (b) Node movement, iteration 5 (c) Node movement, iteration 9

(d) Re-meshing, iteration 1 (e) Re-meshing, iteration 5 (f) Re-meshing, iteration 9

Fig. 6 Drag-adapted meshes for the transonic airfoil case.

Figure 7 shows the convergence of the drag output for both adaptive strategies. The truth output was obtained from a re-meshing run at the same order, but with 10 times more degrees of freedom. Due to the presence of the shocks, the relatively coarse meshes used, and the simple artificial-viscosity shock capturing method, the adaptations are somewhat noisy. There is nevertheless a downward trend in the error with both strategies, and by the later stages, the drag error on the meshes obtained with node movement is lower than the error on the meshes obtained with re-meshing.

A reason for this could be the that slight element position changed during re-meshing, caused by the non-uniqueness of a metric-conforming mesh, makes fitting the shock to edges unreliable. On the other hand, optimization of the node position keeps the same mesh topology and is more likely to keep shocks fit along mesh edges, a configuration in which the error is generally lowest.

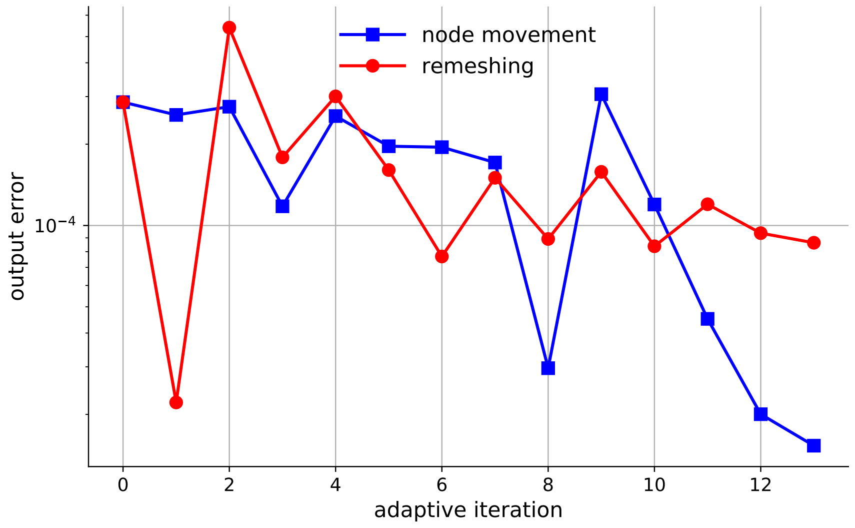
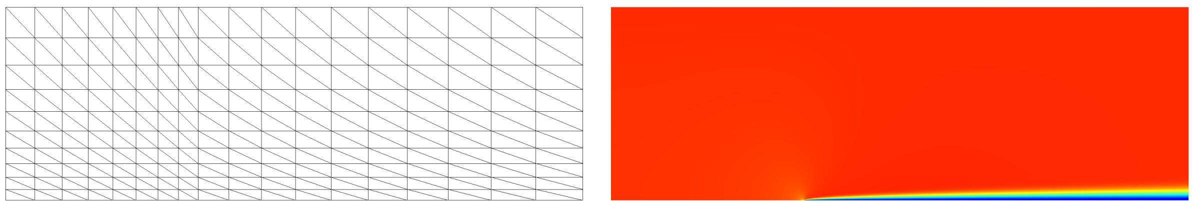


Fig. 7 Drag output convergence for the transonic airfoil case.

C. Laminar Flat Plate

The last case is that of laminar flow over a flat plate. The Reynolds number based on unit distance along the flat plate is $Re = 10^4$. Figure 8 shows the initial mesh of 360 linear triangles, obtained from subdividing elements of a structured quadrilateral mesh. This figure also shows the Mach number contours, from which the boundary layer is evident. The flat plate is of unit length and starts one third of the way downstream from the inflow. The boundary conditions are no-slip adiabatic wall on the plate, total temperature and pressure inflow, static-pressure outflow and top-boundary, and symmetry ahead of the plate. The solution approximation order is $p = 2$.



(a) Initial mesh (360 elements) (b) Mach number (0-0.5)

Fig. 8 Initial mesh and Mach contours for the laminar flat plate case.

The output of interest for adaptation is the drag force on the flat plate. Figure 9 shows the adapted meshes obtained using node movement and re-meshing. As the adaptation starts, the node movement strategy compresses the elements above the plate vertically, so that by the fifth adaptive iteration, the boundary layer is already well-resolved. The remaining adaptive iterations bring more nodes from the inflow region closer to the start of the flat plate, where additional resolution is required. The re-meshing approach has the advantage of not being tied to the initial topology, and more elements are placed above the start of the flat plate compared to the node-movement strategy, which must balance resolution there with the resolution at the plate start itself given its fixed budget of elements across the height of the channel. In addition, the fixed topology during node movement leads to inefficient vertically anisotropic elements above and away from the middle of the flat plate, a region where isotropic and overall larger elements could be used, as indicated in the re-meshing results.

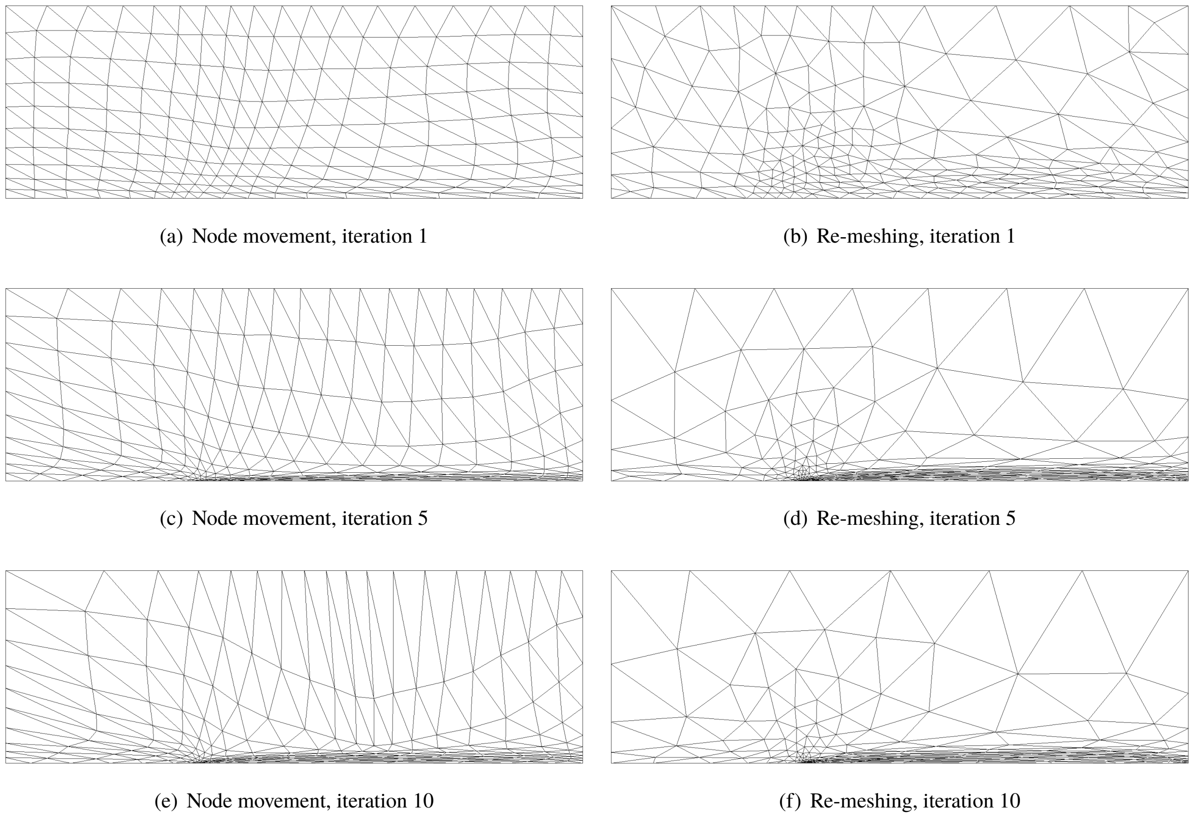

(a) Node movement, iteration 1
(b) Re-meshing, iteration 1
(c) Node movement, iteration 5
(d) Re-meshing, iteration 5
(e) Node movement, iteration 10
(f) Re-meshing, iteration 10

Fig. 9 Drag-adapted meshes for the laminar flat plate case.

Figure 10 shows the convergence of the drag output error for both strategies. The truth output is obtained from a re-meshing run using 5 times more degrees of freedom. We see that both methods show a fairly steady drop in the output error, and that the re-meshing strategy performs better than the node-repositioning strategy. The inefficiencies in node-repositioning outlined in the previous paragraph, tied to the mesh fixed topology, are likely responsible for this difference. More sophisticated local mesh operations could address these inefficiencies. Nevertheless, discovery of the boundary layer starting from a mesh with no boundary-layer resolution and the associated error drop of over three orders of magnitude is impressive for optimization of the fixed-topology mesh.

VII. Conclusions

This paper presents a method for adapting meshes using node movement. Many previous approaches have looked at node movement for adaptation in the past, driven by various error measures. The novelty of this work is the use of a metric-based approach, in which the error estimate and prediction of the optimal mesh are disconnected from the adaptation. Mesh optimization through error sampling and synthesis provides a target desired step matrix field, generated from a general solution error measure, here the adjoint-weighted residual method. The mesh nodes are then moved to optimize the error between the target step matrix and the realized step matrix from node repositioning, over all of the elements. No additional flow/adjoint solutions or output error calculations are required for the solution of this optimization problem. Instead, all calculations needed for the node optimization involve the mesh node coordinates and the target step matrix. Results for a diverse set of three simulations show favorable performance of the node-repositioning strategy when compared to metric-based re-meshing. Whereas the latter in general performs better, cases involving discontinuous features such as shocks may be more amenable to node movement, which appears better-suited for preserving alignment of the discontinuities with element edges. Additional efficiency improvements are expected by

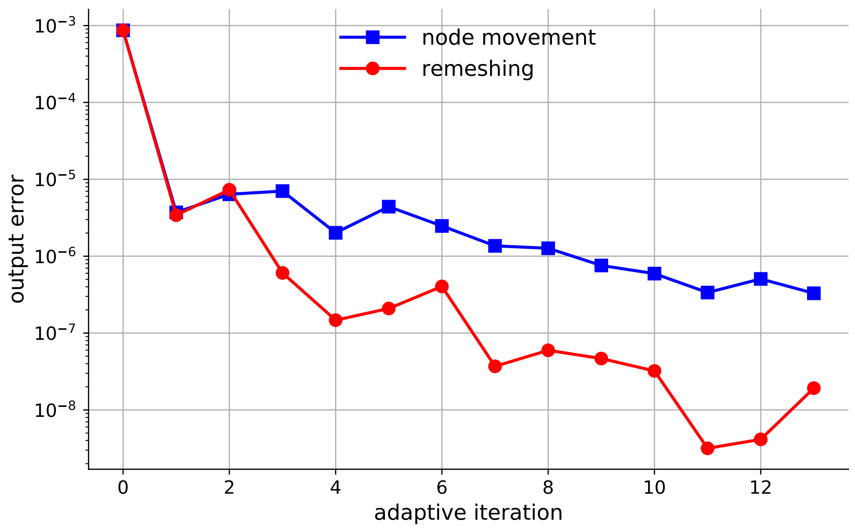


Fig. 10 Drag output convergence for the laminar flat plate case.

including topology-changing capabilities in the node repositioning strategy. Finally, we note that the results of node repositioning depend on the initial mesh, and perhaps the most useful application of the strategy will be when used in tandem with, or following, re-meshing.

References

[1] 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. https://doi.org/10.2514/1.J050073.

[2] De Zeeuw, D., and Powell, K. G., "An Adaptively Refined Cartesian Mesh Solver for the Euler Equations," *Journal of Computational Physics*, Vol. 104, 1993, pp. 56–68.

[3] Coirier, W. J., and Powell, K. G., "Solution-adaptive cut-cell approach for viscous and inviscid flows," *AIAA Journal*, Vol. 34, No. 5, 1996, pp. 938–945.

[4] Pirzadeh, S. Z., "An Adaptive Unstructured Grid Method by Grid Subdivision, Local Remeshing, and Grid Movement," , No. 99-3255, 1999.

[5] Aftosmis, M., and Berger, M., "Multilevel Error Estimation and Adaptive h-Refinement for Cartesian Meshes with Embedded Boundaries," AIAA Paper 2002-14322, 2002.

[6] 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.

[7] 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.

[8] Nemec, M., and Aftosmis, M. J., "Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes," AIAA Paper 2007-4187, 2007.

[9] 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.

[10] Freitag, L. A., and Ollivier-Gooch, C., "Tetrahedral Mesh Improvement Using Swapping and Smoothing," *International Journal for Numerical Methods in Engineering*, Vol. 40, 1997, pp. 3979–4002.

[11] Wood, W. A., and Kleb, W. L., "On Multi-dimensional Unstructured Mesh Adaptation," AIAA Paper 99-3254, 1999.

[12] 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.

[13] Park, M. A., "Adjoint-Based, Three-Dimensional Error Prediction and Grid Adaptation," AIAA Paper 2002-3286, 2002.

[14] Park, M. A., "Three–Dimensional Turbulent RANS Adjoint–Based Error Correction," AIAA Paper 2003-3849, 2003.

[15] Park, M. A., and Darmofal, D. L., "Parallel Anisotropic Tetrahedral Adaptation," AIAA Paper 2008-917, 2008.

[16] Burgess, N. K., and Mavriplis, D. J., "An *hp*-Adaptive Discontinuous Galerkin, Solver for Aerodynamic Flows on Mixed-Element Meshes," AIAA Paper 2011-490, 2011.

[17] Ceze, M. A., and Fidkowski, K. J., "An anisotropic hp-adaptation framework for functional prediction," *AIAA Journal*, Vol. 51, 2013, pp. 492–509. https://doi.org/10.2514/1.J051845.

[18] Ahrabi, B. R., Anderson, W. K., and Newman, J. C., "An adjoint-based hp-adaptive stabilized finite-element method with shock capturing for turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, Vol. 318, 2017, pp. 1030–1065. https://doi.org/https://doi.org/10.1016/j.cma.2017.02.001, URL https://www.sciencedirect.com/science/article/pii/S0045782516304947.

[19] 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.

[20] Peraire, J., Peiró, J., and Morgan, K., "Adaptive Remeshing for Three-Dimensional Compressible Flow Computations," *Journal of Computational Physics*, Vol. 103, 1992, pp. 269–285.

[21] 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.

[22] Alauzet, F., Loseille, A., and Olivier, G., "Time accurate anisotropic goal-oriented mesh adaptation for unsteady flows," *Journal of Computational Physics*, Vol. 373, No. 15, 2018, pp. 28–63. https://doi.org/10.1016/j.jcp.2018.06.043.

[23] Yano, M., and Darmofal, D., "An optimization-based framework for anisotropic simplex mesh adaptation," *Journal of Computational Physics*, Vol. 231, No. 22, 2012, p. 7626–7649. https://doi.org/10.1016/j.jcp.2012.06.040.

[24] Fidkowski, K. J., "A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization," AIAA Paper 2016–0835, 2016. https://doi.org/10.2514/6.2016-0835.

[25] Capon, P. J., and Jimack, P. K., "On the Adaptive Finite Element Solution of Partial Differential Equations Using h-r Refinement," Tech. Rep. 96.03, University of Leeds, School of Computing, 1996.

[26] McRae, D. S., "r-Refinement Grid Adaptation Algorithms and Issues," *Computer Methods in Applied Mechanics and Engineering*, Vol. 2000, No. 4, 189, pp. 1161–1182.

[27] Ding, K., Fidkowski, K. J., and Roe, P. L., "Continuous adjoint based error estimation and r-refinement for the active-flux method," AIAA Paper 2016–0832, 2016. https://doi.org/10.2514/6.2016-0832.

[28] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845. https://doi.org/10.1002/fld.3767.

[29] Ait-Ali-Yahia, D., Baruzzi, G., Habashi, W. G., Fortin, M., Dompierre, J., and Vallet, M.-G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part II: Structured grids," *International Journal for Numerical Methods in Fluids*, Vol. 39, 2002, pp. 657–673.

[30] Chen, G., and Fidkowski, K. J., "Discretization error control for constrained aerodynamic shape optimization," *Journal of Computational Physics*, Vol. 387, 2019, pp. 163–185. https://doi.org/10.1016/j.jcp.2019.02.038.

[31] 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.

[32] Ceze, M. A., and Fidkowski, K. J., "High-Order Output-Based Adaptive Simulations of Turbulent Flow in Two Dimensions," AIAA Paper 2015–1532, 2015. https://doi.org/10.2514/6.2015-1532.

[33] Reed, W., and Hill, T., “Triangular Mesh Methods for the Neutron Transport Equation,” Los Alamos Scientific Laboratory Technical Report LA-UR-73-479, 1973.

[34] 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/https://doi.org/10.1023/A:1012873910884.

[35] 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. https://doi.org/10.1016/j.jcp.2005.01.005.

[36] Roe, P., “Approximate Riemann solvers, parameter vectors, and difference schemes,” *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372. https://doi.org/https://doi.org/10.1016/0021-9991(81)90128-5.

[37] 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. https://doi.org/https://doi.org/10.1002/fld.338.

[38] Ceze, M. A., and Fidkowski, K. J., “Constrained pseudo-transient continuation,” *International Journal for Numerical Methods in Engineering*, Vol. 102, 2015, pp. 1683–1703. https://doi.org/10.1002/nme.4858.

[39] Saad, Y., and Schultz, M. H., “GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems,” *SIAM Journal on Scientific Computing*, Vol. 7, No. 3, 1986, pp. 856–869. https://doi.org/https://doi.org/10.1137/0907058.

[40] Persson, P.-O., and Peraire, J., “Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier-Stokes Equations,” *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 2709–2733. https://doi.org/https://doi.org/10.1137/070692108.

[41] 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.

[42] Pennec, 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.

[43] Liu, D. C., and Nocedal, J., “On the Limited Memory BFGS Method for Large Scale Optimization,” *Mathematical Programming*, Vol. 45, 1989, pp. 503 – 528. https://doi.org/10.1007/BF01589116.

[44] Bassi, F., and Rebay, S., “High–order accurate discontinuous finite element solution of the 2-D Euler equations,” *Journal of Computational Physics*, Vol. 138, 1997, pp. 251–285.

[45] 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.

[46] Persson, P.-O., and Peraire., J., “Sub-cell shock capturing for discontinuous Galerkin methods,” AIAA Paper 2006-112, 2006. https://doi.org/https://doi.org/10.2514/6.2006-112.