Error Minimization via Metric-Based Curved-Mesh Adaptation

Devina P. Sanjaya* and Krzysztof J. Fidkowski†
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

Laslo T. Diosady‡ and Scott M. Murman§
NASA Ames Research Center, Moffett Field, CA, 94035, USA

This work presents an algorithm for generating a high-order, metric-conforming mesh that builds on previous works of Yano[1] and Sanjaya–Fidkowski.[2] The new contribution is a metric-based optimization approach for determining the location of high-order geometry nodes in a curved mesh. The proposed approach does not require additional infrastructure beyond that already available to support curved elements, and its computational cost is similar to metric-based optimization for linear elements. Furthermore, the process requires minimum tuning and can be automated. This paper focuses on understanding the concept of metric conforming meshes with curved elements. Results for one-dimensional solution approximation show that for a given target cost, the resulting high-order mesh has the potential to obtain more accurate solution approximation than a linear, metric-conforming mesh.

I. Introduction

The goal of mesh generation and adaptation is to place mesh resolution in necessary regions. We refer to mesh resolution as the size and shape of a mesh element, as both of these affect the approximation power of the mesh. This information can be encoded in a metric field[3,4] over the computational domain. Our goals are to use a metric to guide the placement of high-order geometry nodes and to generate a high-order, curved, metric-conforming mesh.

Metric fields for mesh generation and adaptation can be derived through heuristic,[5–8] semi-heuristic,[9–12] and more recently, rigorous[1,13] ways. These methods have been used to generate meshes that minimize error or computational cost. Another adaptive approach is to improve the approximation power of a given mesh by increasing the approximation power of the elements. *hp*-Adaptation falls into this category by refining both the mesh and the polynomial order.[14–17] Optimizing the test space can also give us extra accuracy in a certain error norm or an output of interest.[18–21] Yet, another possibility is to tailor finite-element basis functions to the problem at hand through several methods, such as the partition of unity method,[22] the extended finite-element method,[23] isogeometric analysis,[24] and the discontinuous enrichment method.[25] Unfortunately, tailoring basis functions *a priori* is not always easy, especially for complex flows in which the locations of features such as shocks and shear layers are not known ahead of time. Alternatively, tailoring basis functions *a posteriori* is more robust, and this is the approach that was chosen in the previous work by Sanjaya and Fidkowski.[2]

In this paper we consider the rigorous output-based approach developed by Yano[1] to obtain the metric fields that give the best linear, metric-conforming mesh possible. Based on this metric field, we then sequentially optimize the location of the high-order geometry nodes such that the high-order mesh conforms to the metric of the linear mesh. This step essentially tailors the basis functions *a posteriori* on each element

*Ph.D. Candidate, AIAA Member
†Associate Professor, AIAA Senior Member
‡Science and Technology Corporation
§Aerospace Engineer, AIAA Member

and generates curved elements.[2] The final result is a high-order, metric-conforming, curved mesh, which can offer more accurate solution approximation without adding much computational cost or software complexity. Our eventual goal is to implement this algorithm as part of a mesh generation procedure to be used with a very high-order finite-element method (e.g. up to 16th order).

The outline for the remainder of this paper is as follows. We give a brief review on our chosen discretization, a high-order discontinuous-Galerkin finite-element method, in Section II. Section III describes the iterative mesh optimization approach used to determine the optimal metric on a linear mesh. Section IV presents the mechanisms for generating curved elements and its benefit. Section V details our approach for determining the optimal location of the high-order geometry nodes. Section VI shows one-dimensional results obtained from this method, and Section VII presents conclusions and plans for furture work.

II. Discretization

The proposed mesh generation algorithm can be applied to any method that supports high-order curved elements. We focus on the discontinuous-Galerkin (DG) method because we have experience with it and because it is a relatively mature high-order method suitable for convection-dominated flows that are prevalent in aerospace engineering. DG,[26–28] as a finite-element method, approximates the state $\mathbf{u}$ in functional form using linear combinations of basis functions on each element. No continuity constraints are imposed between adjacent elements. Denoting by T_h the set of N_e elements in a non-overlapping tessellation of the domain Ω, the state on element e, Ω_e, is approximated as

$$\mathbf{u}_h(\vec{x}(\vec{\xi}))\Big|_{\Omega_e} = \sum_{n=1}^{N_p} \mathbf{U}_{en}\phi_{en}^{\text{glob}}(\vec{x}(\vec{\xi})). \tag{1}$$

In this equation, N_p is the number of basis functions per element, $\mathbf{U}_{en}$ is the vector of s coefficients for the n^{th} basis function on element e: $\phi_{en}^{\text{glob}}(\vec{x}(\vec{\xi}))$, and s is the state rank. $\vec{x}$ denotes the global coordinates, and $\vec{\xi}$ denotes the reference-space coordinates in a master element. Formally, we write $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where, if the elements are not curved, $\mathcal{V}_h = \{u \in L_2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p \ \forall \Omega_e \in T_h\}$, and $\mathcal{P}^p$ denotes polynomials of order p on the element. The reference-to-global mapping, $\vec{x}(\vec{\xi})$, is polynomials of order q ($\mathcal{P}^q$), where q is the geometry order of the element. A caveat here is that for elements that are curved, the polynomial approximation is usually performed on a master reference element, so that following the reference-to-global mapping, the state approximation on curved elements is not strictly of order p in global space. We take advantage of this observation when we determine the optimal location of the high-order geometry nodes to yield better approximation properties compared to polynomials.

III. Iterative Metric-Based Optimization

In this section, we provide a brief review of the approach introduced by Yano,[1] which iteratively determines the optimal change in the mesh metric field given prescribed cost and error models. Additional details can also be found in our previous work.[13] In addition, we explain some simplifying assumptions that are used in generating our one-dimensional results.

III.A. Metric-Based Meshing

A Riemannian metric field, $\mathcal{M}(\vec{x})$, is a field of symmetric positive definite (SPD) tensors that serves as a directional yardstick for measuring distances. In multiple dimensions, the metric distance of points separated by $\delta\vec{x}$ is measured as

$$\delta\ell = \sqrt{\delta\vec{x}^T\mathcal{M}\delta\vec{x}}. \tag{2}$$

A mesh that *conforms* to a metric field is one in which each edge has approximately the same metric length, i.e. Eqn. 2 integrated along the edge.

In one dimension, the metric becomes a scalar, and for the purpose of mesh generation we write it as $1/\Delta\vec{x}^2$, where $\Delta\vec{x}$ is the local element size. In two dimensions, the initial metric field can be obtained on each element by solving a system of equations that require unit metric measure of the element edges.

The optimization procedure determines *changes* to this mesh-implied metric, $\mathcal{M}_0(\vec{x})$. Transfers of metrics between elements and nodes involve averaging, which is done using an affine-invariant algorithm.[4] Affine-invariant changes to the metric field are made via a symmetric *step matrix*, $\mathcal{S} \in \mathbb{R}^{d\times d}$, according to

$$\mathcal{M} = \mathcal{M}_0^{\frac{1}{2}} \exp(\mathcal{S}) \mathcal{M}_0^{\frac{1}{2}}. \tag{3}$$

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.

III.B. Error Convergence Model

The mesh optimization algorithm requires a model for how the error changes as the metric changes. We consider one element, Ω_e, with a current error $\mathcal{E}_{e0}$ and a proposed metric step matrix of $\mathcal{S}_e$. Then, the general elemental error model can be written as:

$$\mathcal{E}_e = \mathcal{E}_{e0} \exp\left[\mathrm{tr}(\mathcal{R}_e \mathcal{S}_e)\right] \quad \Rightarrow \quad \frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e} = \mathcal{E}_e \mathcal{R}_e. \tag{4}$$

The total error over the mesh is the sum of the elemental errors, $\mathcal{E} = \sum_{e=1}^{N_e} \mathcal{E}_e$. During optimization, we will be able to change the step matrices at the mesh vertices, $\mathcal{S}_v$, which will map to the step matrices at the elements, $\mathcal{S}_e$. The rate tensor, $\mathcal{R}_e$, is determined separately for each element, normally through sampling.

A typical error model involves refinement and sampling procedures, but for our initial tests in one dimension, we use the least-squares error between the approximated and exact solutions. That is, no sampling is performed. Thus, $\mathcal{E}_{e0}$ is set to the local squared least-squares error, and $\mathcal{R}_e$ is assumed to be $-2(p+1)$, where p is the solution approximation order.

III.C. Cost Model

We base our cost model on degrees of freedom, `dof`, which on each element just depends on the approximation order p, assumed constant over the elements. In one dimension, this cost is $(p+1)$ `dof` per element. By Eqn. 3 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 $\exp\left[\frac{1}{2}\mathrm{tr}(S_e)\right]$. Equivalently, the number of new elements, and hence degrees of freedom, occupying the original area Ω_e increases by this factor. So, the elemental cost model is

$$C_e = C_{e0} \exp\left[\frac{1}{2}\mathrm{tr}(\mathcal{S}_e)\right] \quad \Rightarrow \quad \frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e} = \mathcal{C}_e \frac{1}{2}\mathcal{I}, \tag{5}$$

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

III.D. 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,

$$\mathcal{S}_e = \frac{1}{|V_e|} \sum_{v\in V_e} \mathcal{S}_v, \tag{6}$$

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 $\mathcal{S}_v$, which are computed using the chain rule and Eqn. 6.

The cost only depends on the *trace* of the step matrix; i.e. the trace-free part of $\mathcal{S}_e$ 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 ($\widetilde{\mathcal{S}}_v$) parts,

$$\mathcal{S}_v = s_v\mathcal{I} + \widetilde{\mathcal{S}}_v. \tag{7}$$

Derivatives of the error with respect to s_v and $\widetilde{\mathcal{S}}_v$ are

$$\frac{\partial \mathcal{E}}{\partial s_v} = \mathrm{tr}\left(\frac{\partial \mathcal{E}}{\partial \mathcal{S}_v}\right), \qquad \frac{\partial \mathcal{E}}{\partial \widetilde{\mathcal{S}}_v} = \frac{\partial \mathcal{E}}{\partial \mathcal{S}_v} - \frac{\partial \mathcal{E}}{\partial s_v}\frac{\mathcal{I}}{d} \tag{8}$$

The optimization algorithm is then the same as presented by Yano:[1]

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: $i = 1 \ldots n_{\text{step}}$
   (a) Calculate $\mathcal{S}_e$ from Eqn. 6, $\frac{\partial \mathcal{E}_e}{\partial \mathcal{S}_e}$ from Eqn. 4, and $\frac{\partial \mathcal{C}_e}{\partial \mathcal{S}_e}$ from Eqn. 5.
   (b) Calculate derivatives of $\mathcal{E}$ and $\mathcal{C}$ with respect to s_v and $\widetilde{\mathcal{S}}_v$ using Eqn. 8.
   (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 $|\lambda_v|$: $\mathcal{S}_v = \mathcal{S}_v - \delta s \mathcal{I}$
   (d) Update the trace-free part of S_v to enforce stationarity with respect to shape changes at fixed area: $S_v = S_v + \delta s(\partial \mathcal{E}/\partial \widetilde{\mathcal{S}}_v)/(\partial \mathcal{E}/\partial s_v)$.
   (e) Rescale $S_v \to S_v + \beta \mathcal{I}$, where β is a global constant calculated from Eqn. 5 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, λ_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 λ_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 $\delta s_{\max} = 2\log 2$.

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. For reporting the final error and cost at each adaptive iteration, the values are averaged over the last few solution iterations at each target cost. The optimization approach explained in this section is known as Mesh Optimization via Error Sampling and Synthesis (MOESS).

IV. High-Order Curved Elements

Curved elements are primarily generated out of necessity in creating a valid mesh, driven ultimately by geometry representation requirements on the domain boundary. Typically, we do not pay attention to the precise location of the high-order geometry nodes. Instead, heuristics, such as maximizing the validity of an element (i.e., no clustering of nodes) are often used to determine the location of these high-order geometry nodes. Recently, Sanjaya and Fidkowski[2] investigated the importance of the location these high-order nodes for better solution approximation and output prediction. To obtain such benefits, some clustering of the high-order nodes within an element is allowed as long as the validity of the element is maintained. Here, the improved accuracy is obtained by using the mesh metric information to determine the optimal location of the high-order nodes, resulting in curved elements.

We choose to use a polynomial mapping from the reference element to the global element, as this allows us to take advantage of the existing infrastructure that is already available in most high-order codes. The formula for the mapping function is given in Figure 1, where q is the order of this polynomial, $N_q = (q+1)(q+2)/2$ is the total number of degrees of freedom in the mapping, $\vec{\xi} = [\xi, \eta]^T$ is the coordinate in reference space, and $\vec{x} = [x, y]^T$ is the coordinate in global space. Using Lagrange basis functions, $\phi_i^{\text{Lag}}(\vec{\xi})$, in the mapping allows for an intuitive specification of the high-order element: the coordinates of the N_q nodes $\vec{x}_i$ fully define the mapping function, and $\vec{x}(\vec{\xi}_i) = \vec{x}_i$. We note that other basis functions could also be used to define the element geometry, as long as the resulting mapped elements constitute a complete, non-overlapping tessellation of the domain.

The coordinates $\vec{x}_i$ should be chosen consistently with the corresponding reference-space nodes, $\vec{\xi}_i$, which are equally spaced on the reference element. For example, in Figure 1, $\vec{\xi}_6$ is the centroid of the reference triangle, so $\vec{x}_6$ should be located somewhere in the middle of the curved element. On edges/faces that are on domain boundaries, these nodes are typically on the geometry. However, these requirements do not pin

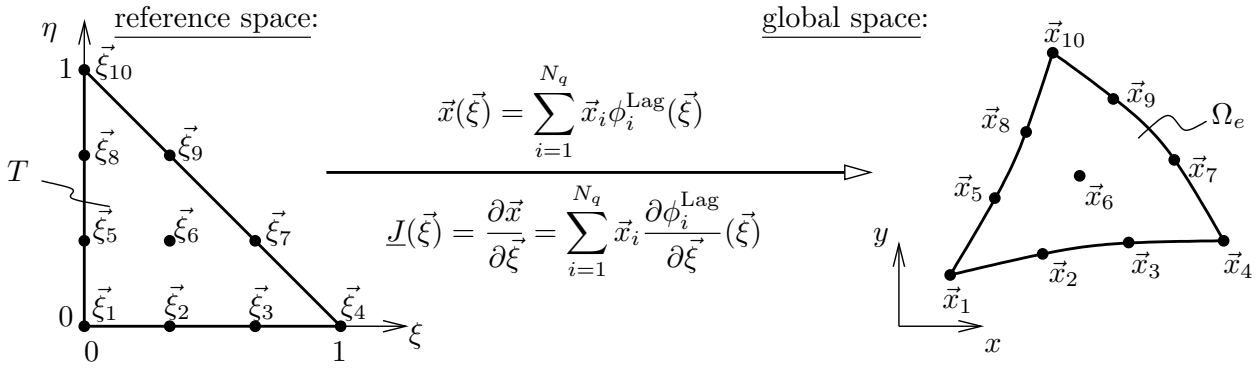


Figure 1. Example of a mapping from a unit reference triangle to a curved-element in global space. The mapping functions for the global coordinates x, y (rolled into $\vec{x}$) are polynomials of order $q = 3$ in this case, for a total of $N_q = 10$ geometry nodes.

down their locations, and heuristics or quality metrics such as maximizing the Jacobian determinant are often used in high-order node placement.

In general, for arbitrary curved elements, a function that is an order p polynomial in reference space does not remain an order p polynomial, or even a polynomial at all, in the global space coordinates. Specifically, a polynomial basis function in reference space, $\phi^{\text{ref}}(\vec{\xi})$, maps to a global basis function according to

$$\phi^{\text{glob}}(\vec{x}(\vec{\xi})) = \phi^{\text{ref}}(\vec{\xi}), \tag{9}$$

where $\vec{x}(\vec{\xi})$ is the geometry mapping given in Figure 1. Moving an element's high order geometry nodes changes this mapping and gives us control over the appearance of the high-order basis functions. We take advantage of this observation by using the mesh metric as a guide to place the high-order nodes at their optimal location.

V. High-Order Mesh Optimization

In the previous sections, we introduced the ease of using metric fields to store the mesh resolution information and the idea of curving mesh elements to improve solution approximation power. In this section, we describe our approach for combining both concepts to generate a high-order, metric-conforming mesh. Currently, we determine the optimal locations of the high-order geometry nodes through a post-processing optimization procedure following the metric-based optimization procedure described in Section III.D. The post-processing step is performed on each mesh element locally.

Before we discuss the details of the post-processing optimization procedure, we note some changes made to the procedure described in Section III.D. First, we use a large number of steps, n_{step}, to ensure a smooth metric. We also find that as the target cost is increased, larger n_{step} may be needed. Second, we let the number of mesh elements grow within the MOESS procedure and only use the target cost to ensure fair comparison between metric-conforming meshes with different q orders. Third, MOESS is performed on a very fine mesh to obtain an accurate metric to guide the placement of the high-order nodes.

V.A. Design Variables

Our design variables are the locations of the high-order geometry nodes ($\boldsymbol{\delta}$) within an element. For our one-dimensional tests, we only have one or two design variables per element; that is, we only consider high-order nodes within the element in $q = 2$ and $q = 3$ meshes. For higher q, the number of design variables grows, resulting in a large number of design variables. This may cause difficulty in finding the optimum solution. We have not considered such high-order elements yet, though we expect to observe large benefits for even small/moderate orders q.

V.B. Objective Function

Our goal is to place the high-order geometry nodes at their optimal location such that the high-order mesh is metric conforming. To do so, we calculate the least-squares error in the metric approximation on each

element ($\varepsilon_{\mathcal{M},e}$). The desired metric is obtained through globally fitting a high-order polynomial to the metric information obtained from the $q = 1$ mesh. This means that the desired metric for $q = 2$ mesh is obtained using the iterative metric-based optimization, discussed in Section III.D. For $q = 3$, the desired metric is obtained from the metric of a uniformly-refined $q = 2$ mesh. The splitting of the elements is performed in the reference space and the total number of mesh elements in the refined mesh is roughly the same as in MOESS. The global polynomial fit ensures that the desired metric is smooth in the entire domain. The approximated metric is calculated based on the reference-to-global mapping Jacobian matrix (defined in Figure 1). Thus, our objective function is

$$(\varepsilon_{\mathcal{M},e})^2 = \int_{\Omega_e} \left| \underline{J}(\boldsymbol{\delta})^{-T} \underline{J}(\boldsymbol{\delta})^{-1} - \mathcal{M}_{\text{desired}}(\boldsymbol{\delta}) \right| dA_e. \tag{10}$$

V.C. Constraints

The proposed optimization problem involves a volume constraint:

$$c_e \equiv \frac{\min(|\underline{J}(\vec{\xi}_i)|)}{V_0} - \eta_V > 0 \tag{11}$$

The volume constraint is set by enforcing a limit in how small the Jacobian determinant can be as a fraction (η_V) of initial element volume (V_0). The Jacobian determinant is calculated at interrogation points in the reference space ($\vec{\xi}_i$), which consist of a large number of equally-spaced points within the element. Note that a non-negative Jacobian determinant is needed to ensure that all mesh elements are valid so that we obtain physical solution. The more interrogation points that are used, the better the chances are that the element is valid everywhere, although this is not guaranteed for for a finite number of points.

V.D. Optimization Problem Formulation

We formulate our constrained optimization problem as follows:

$$\begin{aligned} &\text{minimize } \int_{\Omega_e} \left| \underline{J}(\boldsymbol{\delta})^{-T} \underline{J}(\boldsymbol{\delta})^{-1} - \mathcal{M}_{v,\text{desired}}(\boldsymbol{\delta}) \right| dA_e, \quad \text{w.r.t. } \boldsymbol{\delta}, \\ &\text{subject to } c_e \equiv \frac{\min(|\underline{J}(\vec{\xi}_i)|)}{V_0} - \eta_V > 0 \end{aligned} \tag{12}$$

For our one-dimensional problem, we solve the constrained optimization problem using Matlab's `fmincon` function with the interior-point algorithm. An automatic tuning procedure is implemented to determine η_V for each element. Currently, gradient calculations are performed using finite differences.

Another important step in solving an optimization problem is choosing a good initial guess. A common option is to set the initial guess to be the linear, metric-conforming mesh. Here, we perform some intermediate steps (shown in the blue boxes in Figure 2) to generate a better initial guess:

1. Generate a linear, metric-conforming mesh with q times the desired number of elements based on the target cost.

2. Treat a subset of nodes from the linear mesh generated in step 1 as high-order nodes. For example, for $q = 2$, we treat every other node as a $q = 2$ node.

Figure 2 describes the optimization workflow for generating a high-order, metric-conforming mesh based on a given q and target cost. The purple boxes (boxes 1, 3, and 5) are the main steps in the optimization workflow. The green boxes (boxes 2 and 4) are the inputs to the purple boxes (boxes 3 and 5, respectively). The outputs of the workflow are shown in red boxes (boxes 6).

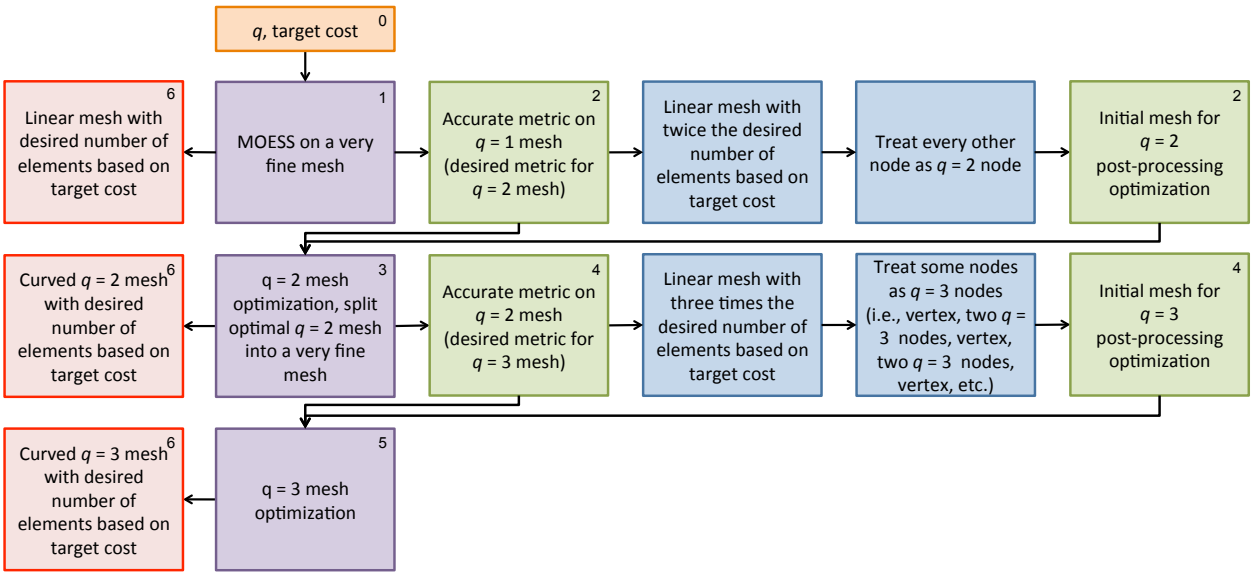


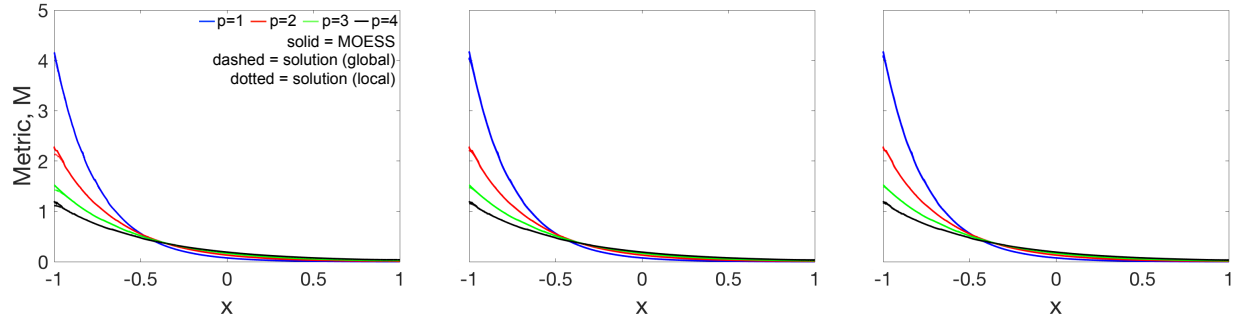
Figure 2. Optimization workflow to generate metric-conforming meshes for $q \leq 3$.

VI. Results

VI.A. Solution-Based versus Metric-Based Optimization

The motivation for our algorithm is that optimizing the approximated metric towards the desired metric is equivalent to optimizing the solution, and that the metric depends on the solution order p, but not (strongly) on the geometry order q. This concept has been investigated for linear meshes,[29] but not for curved meshes. Here, we investigate the effect of q by comparing the metric from MOESS with the approximated metric from global and local solution-based optimizations. When optimizing the solution globally, all interior vertices and all high-order geometry nodes are treated as design variables. Thus, global solution-based optimization is significantly more expensive than local solution-based optimization, in which the only design variables are the high-order nodes within each element. In other words, the local optimization is performed on each element independently, and all vertices in the mesh are non-movable.

For our metric comparison, we consider $p = 1$ to $p = 4$ and $q = 1$ to $q = 3$. We also set the number of mesh elements for solution-based optimization to be the same as the desired number of mesh elements from MOESS, mainly for the convenience for comparing both metrics without scaling. We will discuss metric scaling more in the next section. First, we consider a smooth function, $u^{\text{exact}}_{\text{smooth}} = \exp(-5x)$. Figure 3 shows that all metrics visually coincide. The similarity in the metrics suggests that the optimum metric depends strongly on p, not q, and that the metric optimization is equivalent to minimizing the solution error. Next,



(a) $q = 1$ (b) $q = 2$ (c) $q = 3$

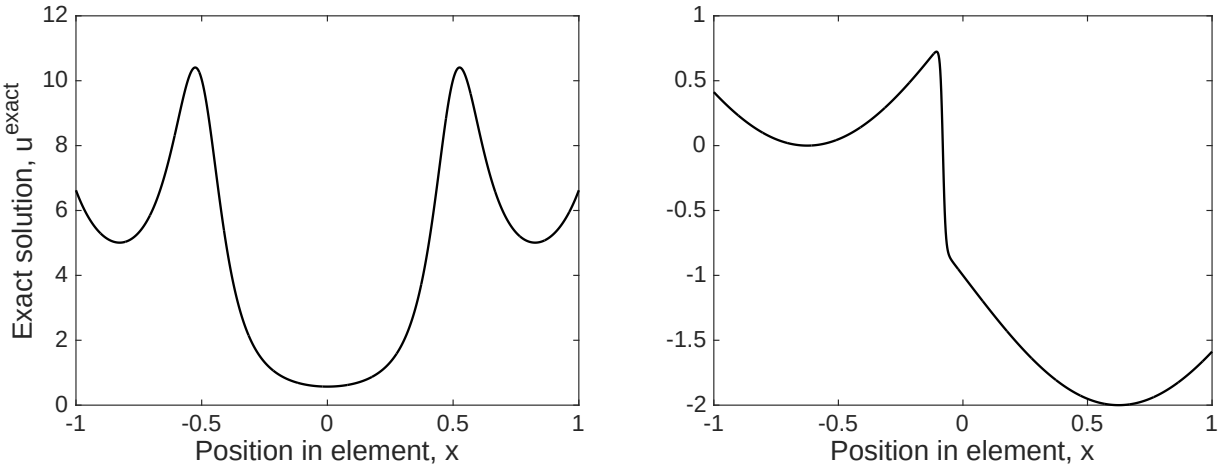
Figure 3. Comparison of metrics at vertices from MOESS and solution-based optimization for a smooth function, $u^{\text{exact}}_{\text{smooth}} = \exp(-5x)$ on 200 mesh elements.

we consider functions with rapid changes (see Figure 4), specifically

$$u_{\text{cont}}^{\text{exact}} = \frac{\cosh(\pi x)}{(1 + 0.75\cos(2\pi x))}, \tag{13}$$

$$u_{\text{shock}}^{\text{exact}} = (1 + \sin(0.8\pi x))\tanh(15 - 30\pi(x + 0.24)). \tag{14}$$

In the continuous case, we intentionally choose a symmetrical function to see if the optimizer will maintain



(a) continuous ($u_{\text{cont}}^{\text{exact}}$) (b) shock ($u_{\text{shock}}^{\text{exact}}$)

Figure 4. Functions with rapid changes

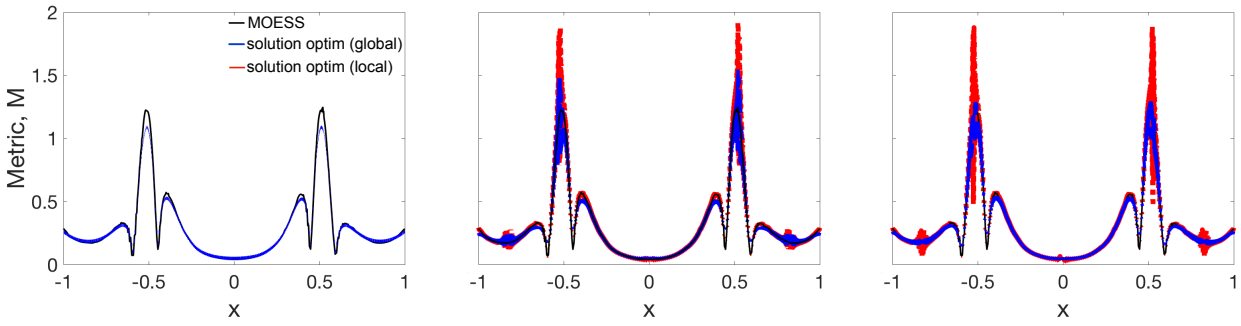
this symmetry. In the shock case, we design the exact solution such that the state to the left and right side of the shock is not constant, so as to avoid optimizing around near zero-error noise. Figure 5 shows that all metrics from our continuous case visually coincide, except at the peaks. The discrepancies between metric from MOESS, global solution optimization, and local solution optimization are less visible as q increases. Also, note that sufficiently high p is needed to obtain a good representation of the metric. This is why the $p = 1$ metrics from local and global solution-based optimization do not match the MOESS metric well. Figure 6 shows the zoomed-in view of the metrics from our shock case; the metric on the two ends of the domain is constant when the mesh is very fine. Here, we see similar behavior as before: the discrepancies between metrics from MOESS, global solution optimization, and local solution optimization are less visible as q increases. Therefore in the asymptotic limit, metric-based optimization is equivalent to solution-based optimization, and that the metric depends on p, but not q.

VI.B. Metric Scaling

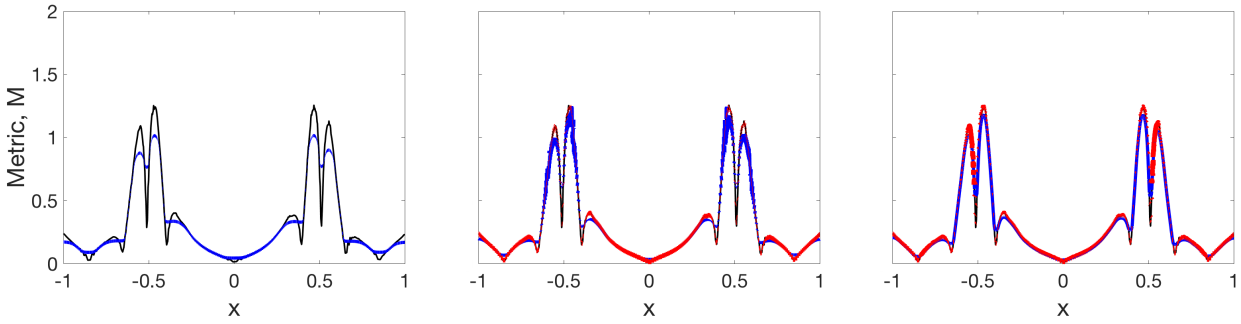
Another important question is how the metric changes with the number of mesh elements. In one dimension, the metric is a scalar, equal to $1/\Delta x^2$ for an element of size Δx. Thus, the metric of a finer mesh is larger than that of a coarser mesh (see Figure 7(a)). Within our optimization workflow, we extract the desired metric from a very fine mesh to make sure that the metric is accurate. On the other hand, we are interested in generating a coarser mesh in comparison to the mesh used to obtain the metric. Therefore, we need to scale the desired metric on the very fine mesh to be desirable for our mesh of interest. This scaling takes the following form,

$$\mathcal{M}_{\text{desired,coarse}} = \left(\frac{N_{\text{desired,coarse}}}{N_{\text{desired,fine}}}\right)^2 \mathcal{M}_{\text{desired,fine}}, \tag{15}$$

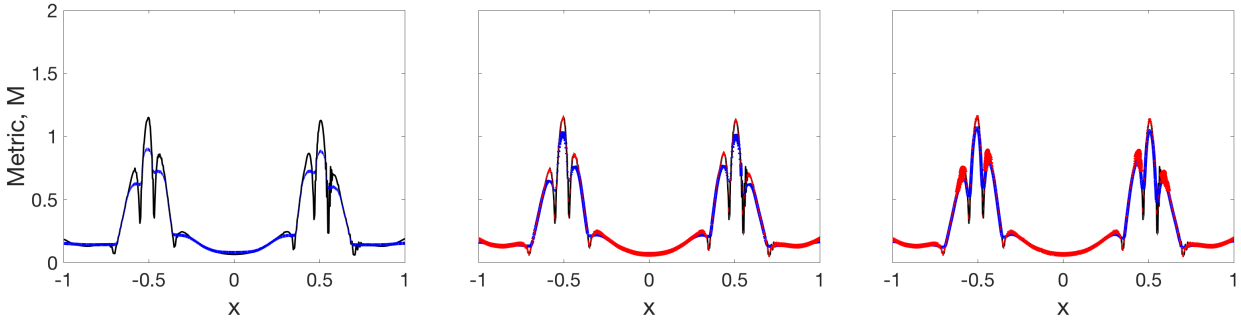
where $N_{\text{desired,fine}}$ and $N_{\text{desired,coarse}}$ refer to the number of desired elements on the fine and coarse meshes, respectively. With this scaling, metrics for a given p and various target costs collapse into one line (see Figure 7(b)).



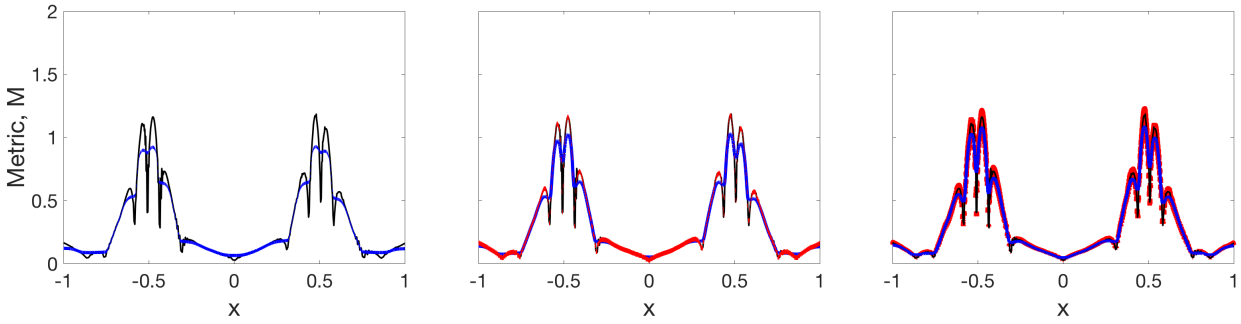
(a) $p = 1$, $q = 1$ (b) $p = 1$, $q = 2$ (c) $p = 1$, $q = 3$



(d) $p = 2$, $q = 1$ (e) $p = 2$, $q = 2$ (f) $p = 2$, $q = 3$

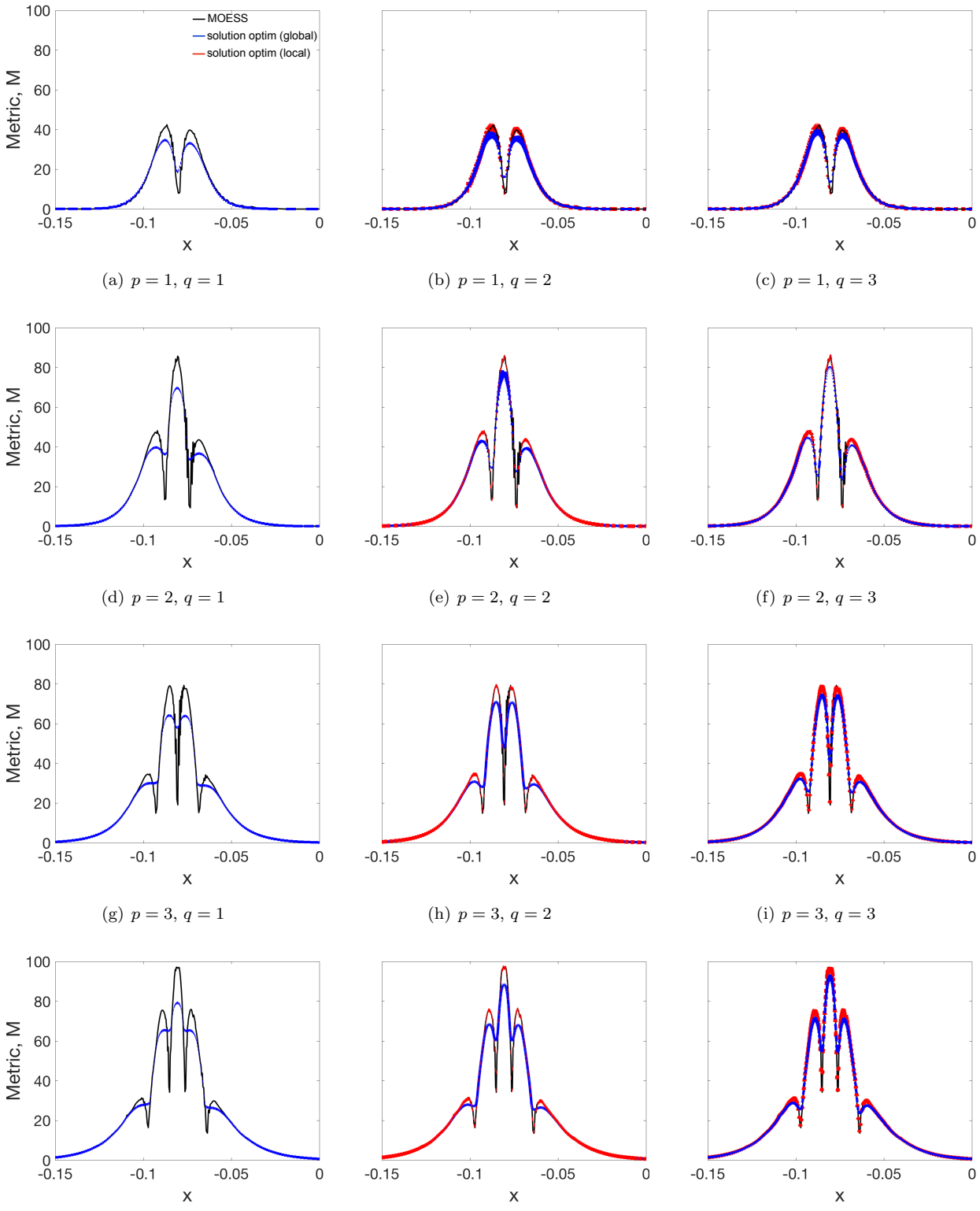


(g) $p = 3$, $q = 1$ (h) $p = 3$, $q = 2$ (i) $p = 3$, $q = 3$



(j) $p = 4$, $q = 1$ (k) $p = 4$, $q = 2$ (l) $p = 4$, $q = 3$

Figure 5. Comparison of metrics at vertices from MOESS and solution-based optimization for a continuous function with rapid changes, $u_{\mathbf{cont}}^{\mathbf{exact}} = \cosh(\pi x)/(1 + 0.75\cos(2\pi x))$ on 200 mesh elements.



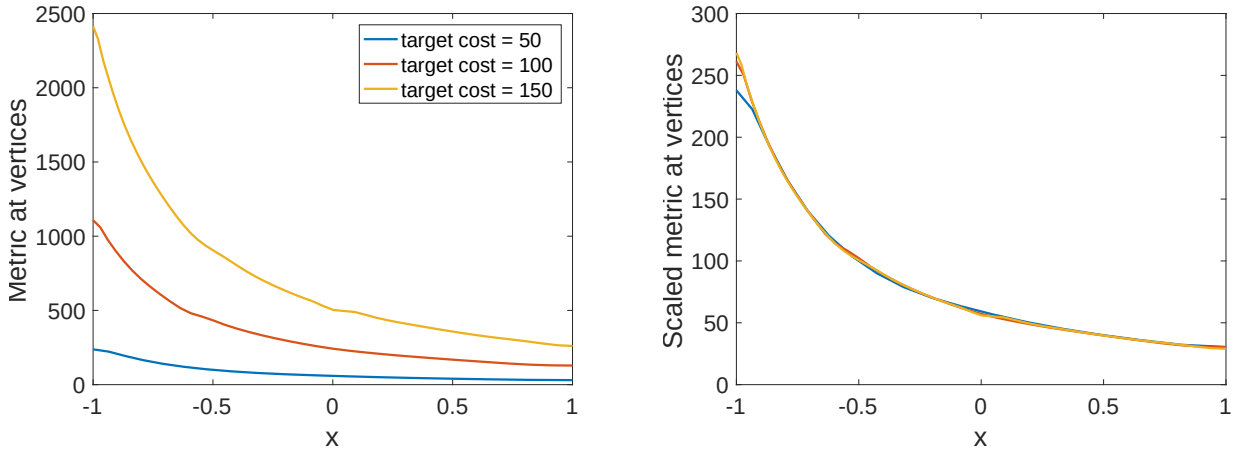
(a) $p = 1$, $q = 1$ (b) $p = 1$, $q = 2$ (c) $p = 1$, $q = 3$

(d) $p = 2$, $q = 1$ (e) $p = 2$, $q = 2$ (f) $p = 2$, $q = 3$

(g) $p = 3$, $q = 1$ (h) $p = 3$, $q = 2$ (i) $p = 3$, $q = 3$

(j) $p = 4$, $q = 1$ (k) $p = 4$, $q = 2$ (l) $p = 4$, $q = 3$

Figure 6. Comparison of metrics at vertices (zoomed-in view) from MOESS and solution-based optimization for a shock function, $u_{\mathbf{shock}}^{\mathbf{exact}} = (1 + \sin(0.8\pi x)) \tanh(15 - 30\pi(x + 0.24))$ on 200 mesh elements.



(a) Metric at vertices

(b) Scaled metric at vertices

Figure 7. $p = 2$ **metric from MOESS at various target costs. The exact solution is** $\exp(-5x)$**.**

VI.C. Global versus Local Optimization

Given a deeper understanding of metric-based and solution-based optimizations, we must decide on whether to perform the post-processing optimization locally or globally. Global optimization is expected to be more powerful than local optimization, but this comes with a higher computational cost. We compare the mesh node distributions from local and global optimizations, and we find that the node distributions are about the same as long as the vertex locations are placed such that the linear mesh is metric conforming. To further assess this observation, we compare the metrics from global and local optimization. Figures 8 and 9 show that the metric from local solution-based optimization is similar to the one form global solution-based optimization, and the agreement between these metrics improves as q increases. Based on this observation, we choose to perform local element-based optimizations. That is, we spend the bulk of the computational effort generating a good initial guess for the post-processing optimization step and perform the optimization locally. This is a more feasible and practical approach since it can be easily parallelized.

VI.D. Convergence of Metric-Conforming, Curved Meshes

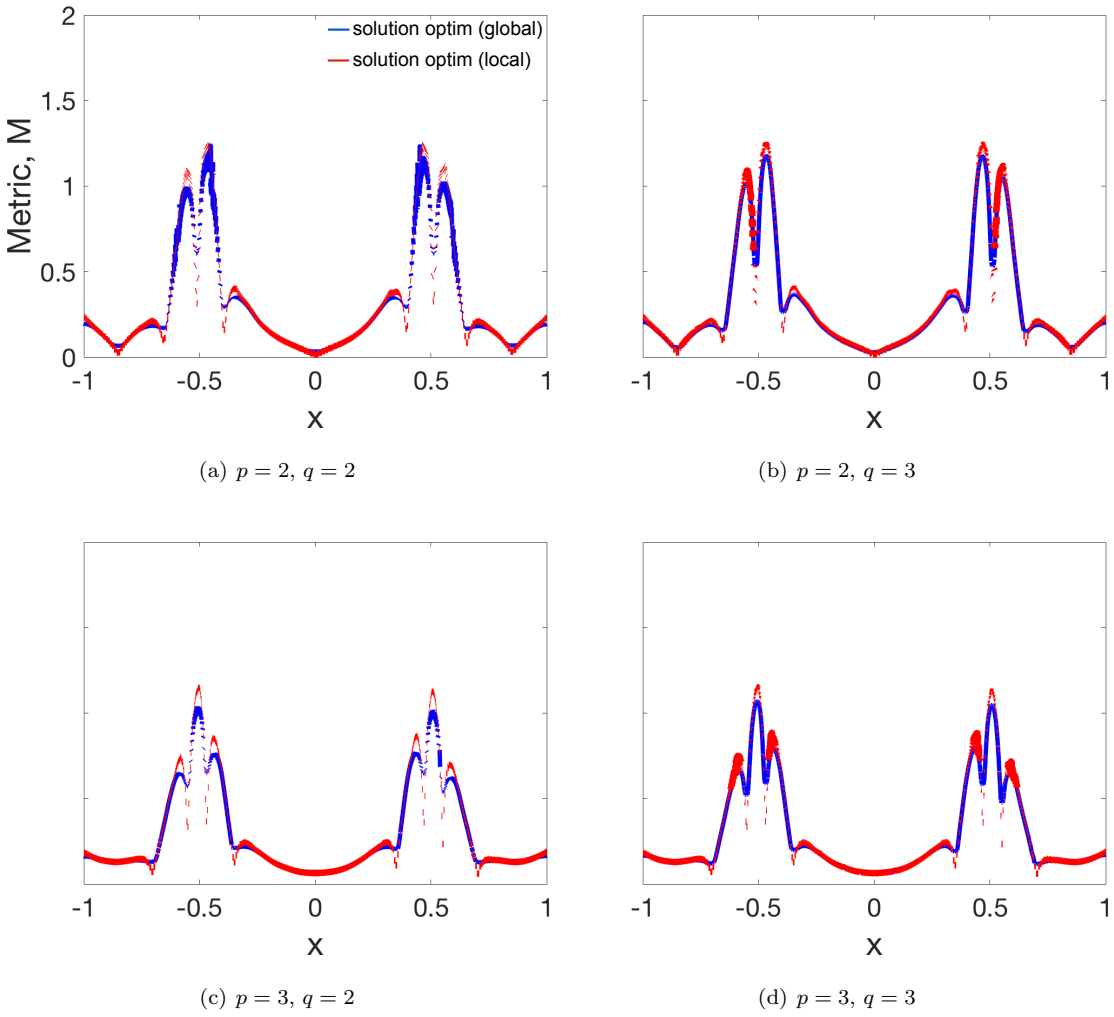
For our convergence study, we choose a boundary-layer-like function as the exact solution, $u^{\text{exact}}_{\text{smooth}} = \exp(-5x)$. To see the benefit of the high-order, metric-conforming mesh adaptation, we compare the least-squares error in the solution approximation between uniformly-refined meshes, $q = 1$ metric-conforming meshes, $q = 2$ metric-conforming meshes, and $q = 3$ metric-conforming meshes. The convergence plot is shown in Figure 10. The convergence rate is $p + 1$ for all meshes, but the $q = 3$ mesh gives us the lowest solution-based error. Compared to $q = 1$ metric-conforming mesh, the high-order mesh gives an additional 40% error reduction for $p = 1$ and an additional 80% error reduction for $p = 2$.

VI.E. Error Minimization of Functions with Rapid Changes

For $p = 2$ and target cost $= 100$, the $q = 2$ mesh gives an additional 10% error reduction for the continuous case and an additional 2% for the shock case, compared to $q = 1$ mesh after MOESS. The solution least-squares error (LSE) values for these cases are shown in Table 1.

Table 1. LSE in solution for continuous and shock cases with $p = 2$ **and target cost = 100.**

	$q = 1$ (MOESS)	$q = 2$
continuous	4.5905×10^{-2}	4.1344×10^{-2}
shock	5.7657×10^{-3}	5.6336×10^{-3}



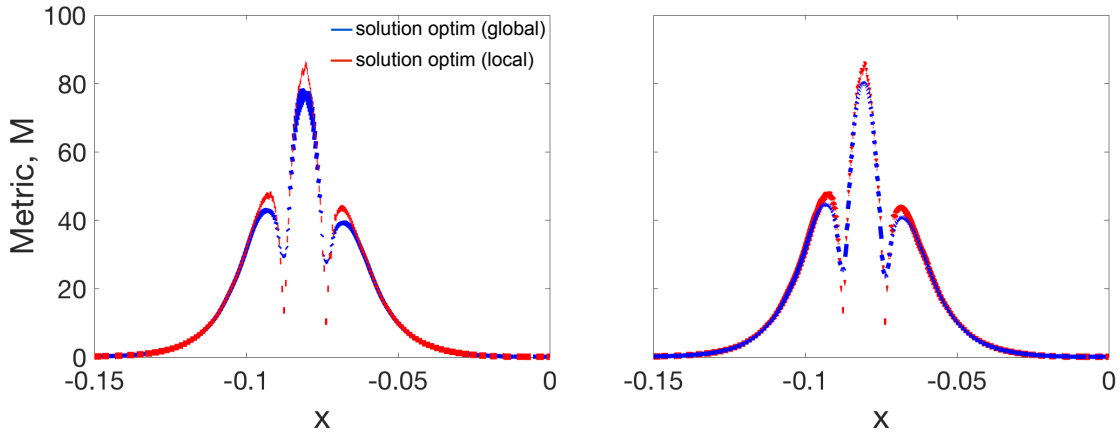
(a) $p = 2$, $q = 2$

(b) $p = 2$, $q = 3$

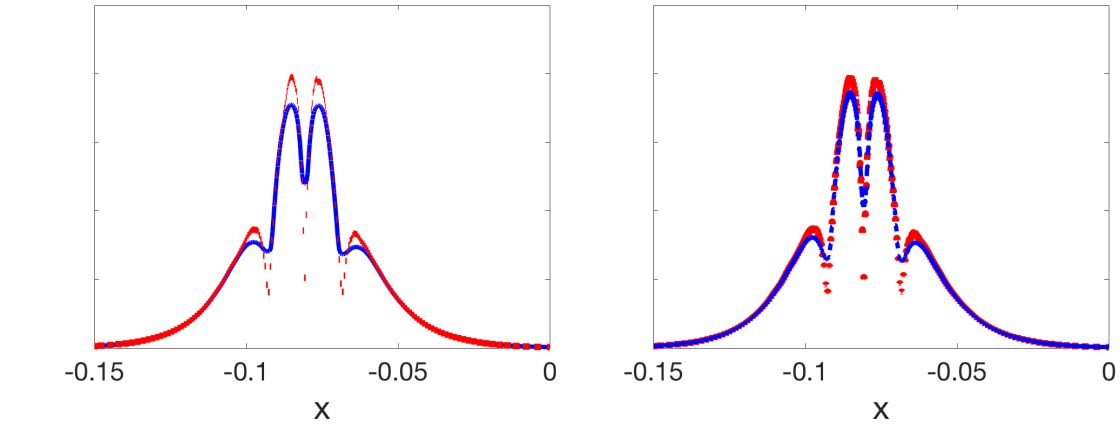
(c) $p = 3$, $q = 2$

(d) $p = 3$, $q = 3$

Figure 8. Comparison of metrics from global and local solution-based optimizations for $u_{\mathbf{cont}}^{\mathbf{exact}} = \cosh(\pi x)/(1 + 0.75\cos(2\pi x))$.



(a) $p = 2$, $q = 2$ (b) $p = 2$, $q = 3$



(c) $p = 3$, $q = 2$ (d) $p = 3$, $q = 3$

Figure 9. Comparison of metrics (zoomed-in view) from global and local solution-based optimizations for $u^{\text{exact}}_{\text{shock}} = (1 + \sin(0.8\pi x)) \tanh(15 - 30\pi(x + 0.24))$.

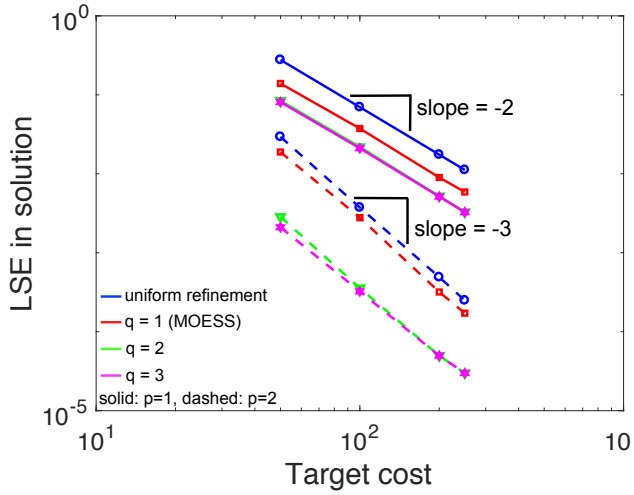


Figure 10. Convergence plots of uniformly-refined, $q = 1$ metric-conforming, $q = 2$ metric-conforming, and $q = 3$ metric-conforming meshes with various p and target costs.

VII. Conclusions and Future Work

In this paper, we present the concept of high-order, metric-conforming mesh adaptation to better approximate a solution. The mesh optimization algorithm presented here combines the work of Yano[1] and Sanjaya–Fidkowski.[2] The advantage of this approach is that the marginal error-to-cost ratio is equidistributed over the entire mesh, and the improvement in accuracy can be obtained without adding much computational cost and software complexity. We also provide a deeper understanding of the concept of metric conformity in curved meshes. We show that the shape of the metric depends on the solution order p, but not the geometry order q. We then observe error reduction by increasing p and/or q, though increasing q does not change the convergence rate. The next step is to extend these ideas to multiple-dimensional problems. Eventually we foresee applying this optimization algorithm to generate high-order meshes for use with very high-order finite-element methods.

Acknowledgments

The authors acknowledge the financial support of the 2015 Michigan Institute for Computational Discovery and Engineering (MICDE) Fellowship, the François–Xavier Bagnoud Fellowship, the NASA Entry Systems Modeling project at the NASA Ames Research Center, and the 2016 Amelia Earhart Fellowship.

References

[1]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.

[2]Sanjaya, D. P. and Fidkowski, K. J., "Improving high-order finite element approximation through geometrical warping," *AIAA Journal*, (2016), accessed September 8, 2016.

[3]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.

[4]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.

[5]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.

[6]Baker, T. J., "Mesh Adaptation Strategies for Problems in Fluid Dynamics," *Finite Elements in Analysis and Design*, Vol. 25, 1997, pp. 243–273.

[7]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.

[8]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.

[9]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.

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

[11]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.

[12]Yano, M., Modisette, J., and Darmofal, D., "The Importance of mesh adaptation for higher-order discretizations of aerodynamics flows," AIAA Paper 2011-3852, 2011.

[13]Fidkowski, K. J., "A Local Sampling Approach to Anisotropic Metric-Based Mesh Optimization," AIAA Paper 2016–0835, 2016.

[14]Houston, P., Senior, B., and Süli, E., "hp-Discontinuous Galerkin Finite Element Methods for Hyperbolic Problems: Error Analysis and Adaptivity," *International Journal for Numerical Methods in Fluids*, Vol. 40, 2002, pp. 153–169.

[15]Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *American Institute of Aeronautics and Astronautics Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

[16]Wang, L. and Mavriplis, D., "Adjoint-based $h-p$ adaptive discontinuous Galerkin methods for the 2D compressible Euler equations," *Journal of Computational Physics*, Vol. 228, 2009, pp. 7643–7661.

[17]Fidkowski, K., "High-Order Output Based Adaptive Methods for Steady and Unsteady Aerodynamics," *37^{th} 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.

[18]Demkowicz, L. and Gopalakrishnan, J., "A Class of Discontinuous Petrov-Galerkin Methods. Part I: The Transport Equation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 199, No. 23-24, 2010, pp. 1558–1572.

[19]Demkowicz, L. and Gopalakrishnan, J., "A Class of Discontinuous Petrov-Galerkin Methods. Part II: Optimal Test Functions," *Numerical Methods for Partial Differential Equations*, 2011, pp. 70–105.

[20]Demkowicz, L., Gopalakrishnan, J., and Niemi, A., "A class of discontinuous Petrov-Galerkin methods. Part III: Adaptivity," *Applied Numerical Mathematics*, Vol. 62, 2012, pp. 396–427.

[21]Kast, S. M., Dahm, J. P., and Fidkowski, K. J., "Optimal test functions for boundary accuracy in discontinuous finite element methods," *Journal of Computational Physics*, Vol. 298, 2015, pp. 360–386.

[22]Melenk, J. and Babuška, I., "The partition of unity finite element method: Basic theory and applications," *Computer Methods in Applied Mechanics and Engineering*, Vol. 139, No. 1 - 4, 1996, pp. 289 – 314.

[23]Dolbow, J. and Belytschko, T., "A finite element method for crack growth without remeshing," *International Journal for Numerical Methods in Engngineering*, Vol. 46, No. 1, 1999, pp. 131–150.

[24]Benson, D., Bazilevs, Y., De Luycker, E., Hsu, M.-C., Scott, M., Hughes, T., and Belytschko, T., "A generalized finite element formulation for arbitrary basis functions: from isogeometric analysis to XFEM," *International Journal for Numerical Methods in Engineering*, Vol. 83, No. 6, 2010, pp. 765–785.

[25]Farhat, C., Harari, I., and Franca, L. P., "The discontinuous enrichment method," *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, No. 48, 2001, pp. 6455–6479.

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

[27]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.

[28]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.

[29]Houston, P., Georgoulis, E. H., and Hall, E., "Adaptivity and A Posteriori Error Estimation for DG Methods on Anisotropic Meshes," *Proceedings of the International Conference on Boundary and Interior Layers (BAIL)*, edited by G. Lube and G. Rapin, University of Göttingen, 2006.