Generation of metric-conforming structured grids with application to grid-adaptation for LES

N. Oberoi, M. K. Hasan, and J. Larsson

Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, USA

April 22, 2024

1 Introduction

Most solution techniques for partial differential equations (PDEs) require a computational grid (or mesh), the attributes of which directly control the accuracy and computational cost of the numerical solution. The most common process is to let the user build the grid by using some software package, where all the decisions about grid-spacing are made by the user based on their expertise and/or available problem-specific guidelines. A more attractive approach is that of grid-adaptation, defined here as an algorithmic or automatic approach to the iterative creation of improved grids based on the analysis of a solution on a prior grid, which promises a more systematic and more automated method [1].

Grid-adaptation is a two-step process. The first step is to estimate the residual, i.e., the source of error in the numerical solution. This can be done using truncation error analysis [2, 3], interpolation onto a finer mesh followed by direct residual computation [4], or other problem-specific ways [5]. The second step is to generate a new grid that minimizes the expected residual. A relatively simple way to generate an adapted grid is to evaluate the residual in each element and determine whether or not to refine each element based on whether the residual exceeds a certain threshold. The other option is to regenerate the grid from scratch, in a way which approximately minimizes the expected residual on the new grid. This approach requires a way to describe the optimal grid resolution in a continuous sense, which is commonly achieved by a resolution-controlling Riemannian metric $\mathbf{M}$, with elements M_{ij} [6, 7]. An ideal grid is then one for which every edge vector $\mathbf{l}$ of every element has unit length under this metric, i.e., for which

$$l_i M_{ij} l_j = 1 \tag{1}$$

holds for every edge of every element. This assumes that the edges are linear and that the metric tensor does not vary over the edge. The metric $\mathbf{M}$ contains in-

formation about the size, orientation, and anisotropy of the optimal mesh. The resolution-controlling Riemannian metric thus functions as the "interface" between the process of finding the ideal grid (which is problem- and code-specific) and the process of actually building a new grid (which is more general).

The present work is entirely focused on the creation of a new grid from a given Riemannian metric field. The method for this is described and tested in sections 2 and 3. To form a complete grid-adaptation method, the present work must therefore be combined with a residual estimation technique. Two specific examples of this are presented in section 4.

Most metric-based grid-generation methods in the literature have focused on fully unstructured grids, generally with simplex elements [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. A relatively recent review was provided by [19]. For structured grids, adaptation has been driven by block-based refinement methods (often referred to as "Cartesian AMR"), which creates discontinuous grids [20, 21, 22, 23]. In contrast, the objective of the present work is to develop a way to generate metric-conforming *structured* grids with continuously varying resolution. The obvious drawback of structured grids is the much lower geometrical flexibility, which makes automatic grid-generation much more difficult and which produces grids with unnecessarily fine resolution in parts of the domain. The advantage of structured grids is that they allow for the use of larger computational stencils (e.g., WENO schemes) and that they more naturally allow for highly anisotropic grids (e.g., in boundary layers).

The book by Liseikin [24] summarizes the state-of-the-art in structured grid-generation. Most existing methods control the resulting grid through the edges of the domain, i.e., by having the user choose a stretching function and the number of grid points along each edge. The grid in the entire volume is then built either algebraically (e.g., using transfinite interpolation) or by solving some type of (generally elliptic) PDE that promotes smoothness and/or orthogonality.

The approach proposed in this work is slightly different. We assume that there is a known Riemannian metric field $\mathbf{M}$ (created in the first part of a grid-adaptation process, based on residual estimation and some logic about what the ideal grid resolution would be) and seek to develop a method that creates the grid for a given geometry without any additional user input. As discussed below, we choose to solve this problem using a variational approach that minimizes the misfit in the required resolution defined by Eqn. (1). This produces a vector-valued PDE for the coordinate field coupled to a few algebraic equations for the number of elements in each direction, the solution to which defines the structured grid. While the variational approach has been used before to derive grid-generation PDEs (see chapter 8 in the book by Liseikin [24]), the novel aspect of this work is that it is used here to derive PDEs for the generation of grids that conform to a given Riemannian metric field.

2 Methodology

A structured grid is defined by the coordinate mapping $\mathbf{x}(\mathbf{s})$, where $\mathbf{x} \in \Omega \subseteq \mathbb{R}^3$ is the physical coordinate with x_i representing the i-th direction and $\mathbf{s} \in \hat{\Omega} = [0,1]^3$ is the coordinate in computational space with s_j representing the j-th computational direction, $i, j \in \{1,2,3\}$. The grid-spacing in computational space is $\sigma_\alpha = 1/n_\alpha$, with n_α being the number of elements in the α direction. With this, the physical size of an element in this direction is then

$$l_{i,\alpha} = \frac{\partial x_i}{\partial s_\alpha}\,\sigma_\alpha, \quad \alpha \in \{1,2,3\}, \quad \text{(no sum on } \alpha\text{)}. \tag{2}$$

The misfit in the grid-resolution requirement (1) is then

$$m_\alpha = \sigma_\alpha^2\, M_{ij}\, \frac{\partial x_i}{\partial s_\alpha}\, \frac{\partial x_j}{\partial s_\alpha} - 1, \quad \alpha \in \{1,2,3\}, \quad \text{(no sum on } \alpha\text{)}, \tag{3}$$

with a perfect grid satisfying $m_\alpha = 0$ everywhere. Positive or negative misfits imply that the actual grid-spacing is larger or smaller than desired, respectively.

Throughout this paper, we use the Einstein-indical summation convention (repeated indices implies summation over the indices) on Roman subscripts $(i, j, k, \ldots)$ which deal with physical coordinates but not on Greek subscripts (α, β) which deal with computational space coordinates. This choice has been made as based on how computational space terms appear and not conform to the Einstein-indical notation rules.

2.1 Minimizing the misfit: full equations

We want to minimize the misfit between the actual grid resolution and the desired one, which can be stated as minimizing the integral over the entire computational domain of

$$L_{\text{misfit}} = \sum_{\alpha=1}^{3} m_\alpha^2. \tag{4}$$

This is a calculus-of-variations problem which in general can be stated as the minimization of the functional

$$\mathcal{J}[\mathbf{x}(\mathbf{s})] = \int_{\hat{\Omega}} L\left(\mathbf{x}(\mathbf{s})\,, \frac{\partial \mathbf{x}}{\partial \mathbf{s}}(\mathbf{s})\,, \mathbf{s}\right) dV \tag{5}$$

with $dV = ds_1 ds_2 ds_3$, the general solution of which is the Euler-Lagrange equation

$$\frac{\partial L}{\partial x_k} - \sum_{\alpha=1}^{3} \frac{\partial}{\partial s_\alpha}\left(\frac{\partial L}{\partial x_{k,\alpha}}\right) = 0, \quad k \in \{1,2,3\}, \tag{6}$$

where we used the short-hand notation $x_{i,\alpha} = \partial x_i / \partial s_\alpha$. For our kernel L_{misfit}, the Euler-Lagrange equation (note that σ_α are fixed scalar parameters) is

$$-\sum_{\alpha=1}^{3} \underbrace{8\sigma_\alpha^4 M_{kl} \frac{\partial x_l}{\partial s_\alpha} M_{ij} \frac{\partial x_i}{\partial s_\alpha}}_{\mathcal{A}_{kj,\alpha}} \frac{\partial^2 x_j}{\partial s_\alpha^2} - \sum_{\alpha=1}^{3} 4\sigma_\alpha^4 M_{kl} \frac{\partial x_l}{\partial s_\alpha} \frac{\partial M_{ij}}{\partial x_p} \frac{\partial x_p}{\partial s_\alpha} \frac{\partial x_i}{\partial s_\alpha} \frac{\partial x_j}{\partial s_\alpha}$$
$$- \sum_{\alpha=1}^{3} m_\alpha \sigma_\alpha^2 \left(4M_{kj} \frac{\partial^2 x_j}{\partial s_\alpha^2} + 4 \frac{\partial M_{kj}}{\partial x_p} \frac{\partial x_p}{\partial s_\alpha} \frac{\partial x_j}{\partial s_\alpha} - 2 \frac{\partial M_{ij}}{\partial x_k} \frac{\partial x_i}{\partial s_\alpha} \frac{\partial x_j}{\partial s_\alpha} \right)$$
$$= 0, \quad k \in \{1,2,3\}. \tag{7}$$

Note that m_α (Eqn. 3) is defined in the computational space $\mathbf{s}$ and is a function of $\mathbf{x}$ and $\mathbf{M}$. The metric field is most naturally defined in the physical domain, but depends on the computational space coordinate through this mapping as $\mathbf{M}(\mathbf{x}(\mathbf{s}))$.

Since the kernel L_{misfit} also depends on the 3 parameters σ_α, we additionally need

$$\frac{\partial L_{\text{misfit}}}{\partial \sigma_\alpha} = 0, \quad \alpha \in \{1,2,3\}, \tag{8}$$

which yields (with fixed coordinate fields $\mathbf{x}(\mathbf{s})$)

$$\sigma_\alpha = \sqrt{\frac{\int_{\widehat{\Omega}} p_\alpha \, dV}{\int_{\widehat{\Omega}} p_\alpha^2 \, dV}}, \quad \alpha \in \{1,2,3\}, \tag{9}$$

where $p_\alpha = M_{ij} \, x_{i,\alpha} \, x_{j,\alpha}$ and $\widehat{\Omega}$ is the computational domain.

The coupled solution of the PDE (7) and the algebraic relations (9) then produces the coordinate mapping and element-spacing in computational space that minimizes the global misfit. The element-spacings σ_α are real-valued, and one would therefore have to round these to the nearest (inverse of an) integer in practice.

2.2 Minimizing the misfit: approximate PDEs

Close inspection of the PDE (7) shows that it may become singular, either at the optimal solution or during the iterative solution process. Specifically, since m_α can take any real value, the second-order derivative may locally vanish thus making the PDE locally singular. This can make the PDE difficult to solve. Techniques for dealing with this numerically are discussed in section 2.4. Alternatively, the PDE can be regularized to prevent singularities. Two such approaches are discussed here.

The first approach is to neglect all terms in (7) that include the factor m_α. This makes the PDE non-singular, and can be somewhat justified on the basis that $m_\alpha = 0$ for a perfect solution. The constraint of having a structured grid will prevent m_α from being zero in most situations, but one could assume that

m_α will at least be small at convergence which then provides some justification for neglecting terms that include it. The approximate PDE is then

$$-\sum_{\alpha=1}^{3} \underbrace{8\sigma_\alpha^4 M_{kl} \frac{\partial x_l}{\partial s_\alpha} M_{ij} \frac{\partial x_i}{\partial s_\alpha}}_{\mathcal{A}_{kj,\alpha}} \frac{\partial^2 x_j}{\partial s_\alpha^2} - \sum_{\alpha=1}^{3} 4\sigma_\alpha^4 M_{kl} \frac{\partial x_l}{\partial s_\alpha} \frac{\partial M_{ij}}{\partial x_p} \frac{\partial x_p}{\partial s_\alpha} \frac{\partial x_i}{\partial s_\alpha} \frac{\partial x_j}{\partial s_\alpha}$$

$$= 0, \quad k \in \{1, 2, 3\}. \quad (10)$$

A second approach to regularize the PDE is to consider the alternative cost functional kernel

$$L_{\text{misfit}-\text{alt}} = \sum_{\alpha=1}^{3} \sigma_\alpha^2 \, p_\alpha, \quad (11)$$

where $p_\alpha = M_{ij} \, x_{i,\alpha} \, x_{j,\alpha}$ as defined around Eqn. (9). This kernel does not minimize the misfit, but rather aims to make the scaled grid resolution uniformly distributed in physical space. The resulting Euler-Lagrange equation is

$$-\sum_{\alpha=1}^{3} \underbrace{2\sigma_\alpha^2 M_{kj}}_{\mathcal{A}'_{kj,\alpha}} \frac{\partial^2 x_j}{\partial s_\alpha^2} - \sum_{\alpha=1}^{3} \sigma_\alpha^2 \left(2 \frac{\partial M_{kj}}{\partial x_p} \frac{\partial x_p}{\partial s_\alpha} \frac{\partial x_j}{\partial s_\alpha} - \frac{\partial M_{ij}}{\partial x_k} \frac{\partial x_i}{\partial s_\alpha} \frac{\partial x_j}{\partial s_\alpha} \right)$$

$$= 0, \quad k \in \{1, 2, 3\}. \quad (12)$$

When solving either of these approximate versions of the Euler-Lagrange equation for the coordinate field $\mathbf{x}(\mathbf{s})$, one would still use Eqn. (9) to compute the computational space element size σ_α.

2.3 Promoting orthogonality

In general, grids with skewed elements produce numerical errors, and thus it is useful to promote orthogonality when generating a grid. This can be accomplished most naturally in a variational framework by augmenting the cost functional to include terms that penalize non-orthogonality, with some weight. Liseikin [24] describes several ways to construct such penalization terms; in the present work we take the kernel

$$L_{\text{orthogonality}} = g_{12}^2 + g_{13}^2 + g_{23}^2 \quad (13)$$

where

$$g_{\alpha\beta} = \frac{\partial x_l}{\partial s_\alpha} \frac{\partial x_l}{\partial s_\beta}, \ \alpha, \beta \in \{1, 2, 3\}, \quad (14)$$

is the covariant metric tensor. We then take the full kernel in the cost functional as

$$L_{\text{misfit}} + \frac{\lambda_{\text{ortho}}}{\mathcal{L}_{\text{domain}}^4} L_{\text{orthogonality}} \quad (15)$$

where λ_{ortho} is a weight and $\mathcal{L}_{\text{domain}}$ is a characteristic size (length) of the physical domain; the latter is needed to make the two terms dimensionally

consistent and of comparable magnitudes. Linearity then implies that the full Euler-Lagrange equation is simply the sum of the Euler-Lagrange equations from the misfit and the orthogonality. The equation from $L_{\text{orthogonality}}$ alone is

$$-2\frac{\partial}{\partial s_1}\left(g_{12}\frac{\partial x_k}{\partial s_2}+g_{13}\frac{\partial x_k}{\partial s_3}\right)-2\frac{\partial}{\partial s_2}\left(g_{12}\frac{\partial x_k}{\partial s_1}+g_{23}\frac{\partial x_k}{\partial s_3}\right)$$
$$-2\frac{\partial}{\partial s_3}\left(g_{13}\frac{\partial x_k}{\partial s_1}+g_{23}\frac{\partial x_k}{\partial s_2}\right)=0,\ k\in\{1,2,3\}. \quad (16)$$

2.4 Numerical implementation

The objective is to generate a structured grid in the physical domain $\Omega \subseteq \mathbb{R}^3$ by using a mapping from the computational domain that is defined to be the unit cube $\hat{\Omega} = [0,1]^3$. For the numerical implementation, second-order accurate central finite difference schemes are used with a uniform spacing in the computational domain. The grid-generation PDEs are solved on a set of fine and coarse meshes to ensure that the solution obtained is grid converged. For the cases in this paper, 60 to 300 grid points in each direction were found to be sufficient to obtain converged solutions for $\mathbf{x}(\mathbf{s})$. This number will of course depend on the curvature of the boundary and rate of variation of $\mathbf{M}$ in physical space. This produces a highly nonlinear system of equations that has the structure

$$\sum_{\alpha=1}^{3}\mathcal{A}_{kj,\alpha}(\mathbf{x},\mathbf{M})\,\mathcal{D}_{\alpha\alpha}x_k+\mathcal{B}_{\text{ortho}}(\mathbf{x})\,x_k+\mathcal{R}(\mathbf{x},\mathbf{M})=0,\ k\in\{1,2,3\}, \quad (17)$$

where $\mathcal{A}_{kj,\alpha}$ denotes the first factor in each version of the PDE, $\mathcal{D}_{\alpha\alpha}$ is the operator formed by the finite differences of the second derivative with respect to s_α, and $\mathcal{R}$ is the collection of remaining terms in each equation. $\mathcal{B}_{\text{ortho}}$ is the operator formed after applying finite differences to the orthogonality-promoting PDE. These highly nonlinear equations are solved iteratively after linearizing Eqn. (17) about the value at iteration level n as

$$\sum_{\alpha=1}^{3}\mathcal{A}_{kj,\alpha}(\mathbf{x}^n,\mathbf{M})\,\mathcal{D}_{\alpha\alpha}x_k^*+\mathcal{B}_{\text{ortho}}(\mathbf{x}^n)\,x_k^*+\mathcal{R}(\mathbf{x}^n,\mathbf{M})=0,\ k\in\{1,2,3\}. \quad (18)$$

This equation is used to obtain $\mathbf{x}^*$, which is used to update the value to the next level $n+1$ using under-relaxation as

$$\mathbf{x}^{n+1}=\mathbf{x}^n+\epsilon(\mathbf{x}^*-\mathbf{x}^n), \quad (19)$$

where ϵ is the under-relaxation parameter.

To prevent problems caused by the exact Euler-Lagrange equation (7) becoming locally singular during the convergence process, the term

$$q_\alpha\sigma_\alpha^2 4M_{kj}\frac{\partial^2 x_j}{\partial s_\alpha^2} \quad (20)$$

is added to both sides of the equation and evaluated at the new and old iterations, respectively, with q_α taken as the maximum of $|m_\alpha|$ over the full domain. This ensures that the second derivative terms taken at the new iteration are non-singular.

In practice, $\mathbf{M}$ will be known only on some background mesh. It must then be interpolated to the grid points in the computational domain using the solution at iteration level n. This interpolated data should be used to find the spatial derivatives of $\mathbf{M}$. In practice, filtering of the $\mathbf{M}$ field after interpolation may be needed.

The equations have been implemented in Python and solved on a desktop computer. In the current work, the sparse direct linear solver Pardiso [25] is used to solve the linearized system in Eqn. 18.

The boundary conditions on $\mathbf{x}(\mathbf{s})$ are enforced by locally approximating the boundary as a planar surface, which is defined by a point $\mathbf{x}_{\text{bnd}}$, a vector $\mathbf{n}$ that is normal to the plane, and vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ which span the plane. The boundary condition is then

$$\mathbf{x}^{n+1} \cdot \mathbf{n} = \mathbf{x}_{\text{bnd}} \cdot \mathbf{n}, \tag{21a}$$

$$\frac{\partial}{\partial s_\gamma}\left(\mathbf{x}^{n+1} \cdot \mathbf{t}_1\right) = 0, \tag{21b}$$

$$\frac{\partial}{\partial s_\gamma}\left(\mathbf{x}^{n+1} \cdot \mathbf{t}_2\right) = 0, \tag{21c}$$

where γ is the computational space direction normal to the boundary.

The local coordinate system of $\mathbf{t}_1, \mathbf{t}_2$, and $\mathbf{n}$ can be found from an analytically given boundary surface or it can be approximated from discrete data at the previous iteration.

3 Results for given Riemannian metrics

The three different versions of the proposed grid-generation method are tested on a sequence of problems of increasing complexity, all in two dimensions. The methods differ in which PDE they solve for the coordinate field x_i: the "Full" method solves Eqn. (7), the "Approximate" method solves Eqn. (10), and the "Alternate" method solves Eqn. (12). All three solve Eqn. (9) for the element size σ_α in computational space. The element size is allowed to be any real number during the solution process, but after convergence, the number of elements n_α is taken as $1/\sigma_\alpha$ rounded to the nearest integer, which is then used to define the final grid.

For all cases, a simple and interpretable Riemannian metric is given, and we assess: (a) whether the created grid is consistent with the given metric; (b) how robust or fast the convergence is; and (c) how the three different methods differ in these aspects. Results are shown both without ($\lambda_{\text{ortho}} = 0$) and with ($\lambda_{\text{ortho}} > 0$) the orthogonality-promoting terms in the PDE, and with different under-relaxation factors ϵ when discussing the convergence behavior.

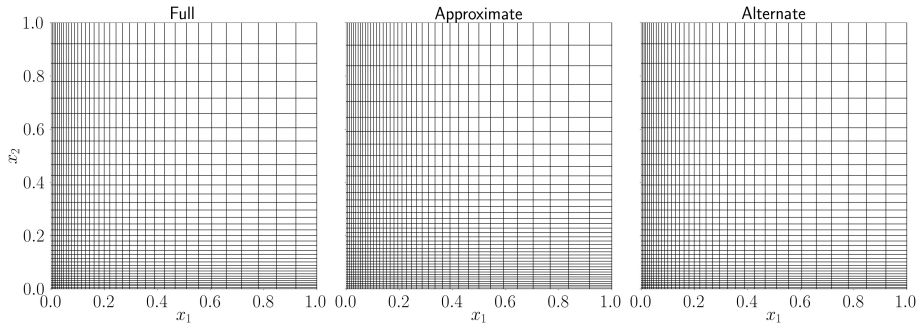

Figure 1: Final grids using $\mathbf{M}$ from Eqn. (22) and $\lambda_{\text{ortho}} = 0$ (section 3.1).

3.1 Rectangular domain, Cartesian metric

First, all three versions of the PDE were examined in terms of grid quality and rate of convergence for a simple rectangular domain $[0,1]^2$ in physical space. The Riemannian metric field was taken as

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} 40000\,(1+15x_1)^{-2} & 0 \\ 0 & 40000\,(1+15x_2)^{-2} \end{bmatrix}. \tag{22}$$

This choice of $\mathbf{M}$ was such that the optimal grid distribution in each Cartesian direction can be computed beforehand and used to assess the correctness of the proposed methods. Specifically, Eqn. (1) implies that $\Delta x_{1,\text{optimal}} = 1/\sqrt{M_{11}}$ and $\Delta x_{2,\text{optimal}} = 1/\sqrt{M_{22}}$. Grid dependency tests were conducted for all three versions with the conclusion that 40 grid points in each direction is sufficient when solving the PDE for this particular test case (note that the number of points used when solving the PDE is independent of the resulting values of σ_α).

The grids generated from all three versions of the PDE are shown in Fig. 1. The orthogonality promotion has no effect for this special case. Figure 1 shows that the grid lines are clustered near the bottom left boundary, qualitatively agreeing with the specified metric field. A more careful analysis shows that the misfit m_α is in fact zero for all of these grids. The optimal number of grid elements ($n_1 = 1/\sigma_1$ and $n_2 = 1/\sigma_2$) for all three algorithms is approximately 37.

Figure 2 shows the convergence behavior for different under-relaxation parameters ϵ. For this particular test case, we note that the iterative procedure for obtaining the solution to the Full version was initialized with the converged solutions to the Approximate version, and no change in the solution was observed. For this reason, the convergence behavior of the Full version equations is not included in Fig. 2. It is clear from Fig. 2a that the number of iterations required for each under-relaxation parameter for the Approximate version is the same as for the Alternate version. The iterative method was stopped when the residual of the solution dropped below 10^{-10}. Not surprisingly, the required number of iterations decreases with the increase of the under-relaxation parameter. The

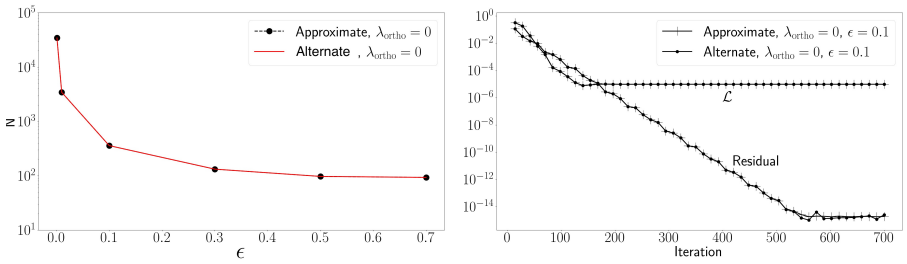

Figure 2: Convergence using $\mathbf{M}$ from Eqn. (22) in section 3.1, showing the number of iterations to convergence vs the under-relaxation parameter ϵ (left) and the decrease of the cost functional $\mathcal{J}$ and the residual (right).

solutions to all three versions diverged for higher value of the under-relaxation parameter ($\epsilon > 0.7$). Figure 2b shows how the cost functional and the numerical residual decrease for $\epsilon = 0.1$, giving a sense of the speed of the convergence process.

3.2 Rectangular domain, non-Cartesian metric

We next investigate the behavior for a diagonal metric field where both M_{11} and M_{22} vary in two directions simultaneously. The metric field is taken as

$$\mathbf{M} = \begin{bmatrix} 1000 + C & 0 \\ 0 & 1000 - C \end{bmatrix} \text{ with } C = 600 \sin(2\pi x_1) \sin(2\pi x_2). \tag{23}$$

For the present case, we again considered a simple rectangular domain of $[0, 1]^2$ in physical space. The motivation for choosing the particular metric field is to examine if the solvers can handle the skewness embedded in the metric field. As before, all the PDEs were solved with 40 points in each direction.

Figure 3 displays the grids obtained from the solutions to the Approximate and Alternate version for $\lambda_{\text{ortho}} = 0$. Note that the Full version actually did not converge for $\lambda_{\text{ortho}} = 0$, at least with our initial condition of a uniform grid. The chosen metric field creates a twisted grid, with the degree of grid twisting being much stronger for the Approximate version than for the Alternate one.

Figure 4 displays the meshes obtained by solving all three versions with the orthogonality-promoting terms with weight parameter $\lambda_{\text{ortho}} = 1$. With orthogonality-promotion, the Full method converges, and all three versions produce very similar grids. It is interesting to note that the Alternate version of the method creates similar grids both with and without the orthogonality-promotion.

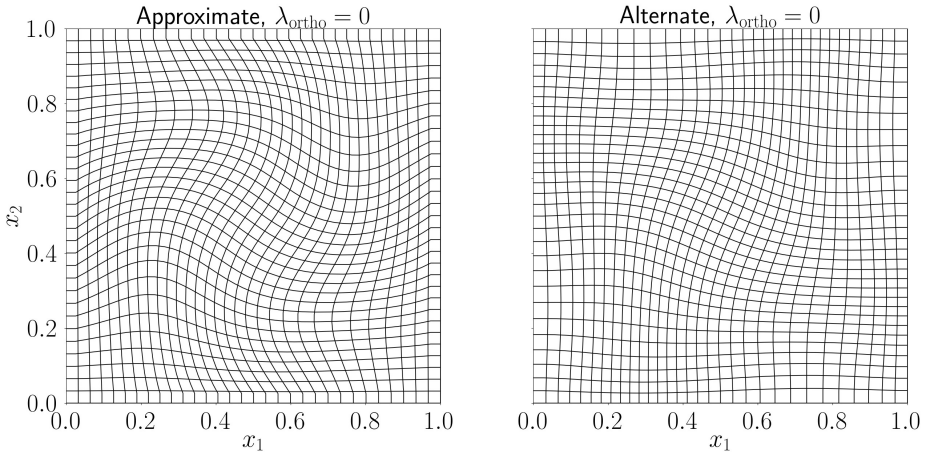

Figure 3: Final grids using $\mathbf{M}$ from Eqn. (23) and $\lambda_{\text{ortho}} = 0$ (section 3.2).

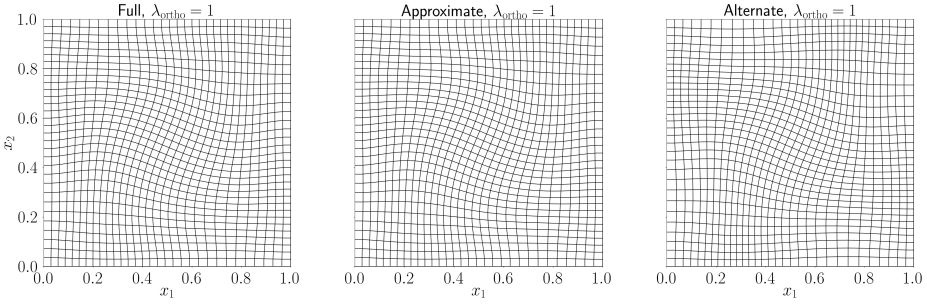

Figure 4: Final grids using $\mathbf{M}$ from Eqn. (23) and $\lambda_{\text{ortho}} = 1$ (section 3.2).

3.3 Moderately curved domain

We next consider a non-Cartesian physical domain with the boundaries defined by

$$\begin{aligned} x_1 - A\sin(2\pi x_2) &= 0 \ \text{ at } \ s_1 = 0 \ \ \text{(left)}, \\ x_1 - A\sin(2\pi x_2) &= 1 \ \text{ at } \ s_1 = 1 \ \ \text{(right)}, \\ x_2 + A\sin(2\pi x_1) &= 0 \ \text{ at } \ s_2 = 0 \ \ \text{(lower)}, \\ x_2 + A\sin(2\pi x_1) &= 1 \ \text{ at } \ s_2 = 1 \ \ \text{(upper)}, \end{aligned} \tag{24}$$

respectively, where A is the amplitude of the boundary curvature. In the present section, $A = 0.1$. Since the objective is to evaluate the effect of boundary curvature, the simple isotropic metric field

$$\mathbf{M} = \begin{bmatrix} 2000 & 0 \\ 0 & 2000 \end{bmatrix}, \tag{25}$$

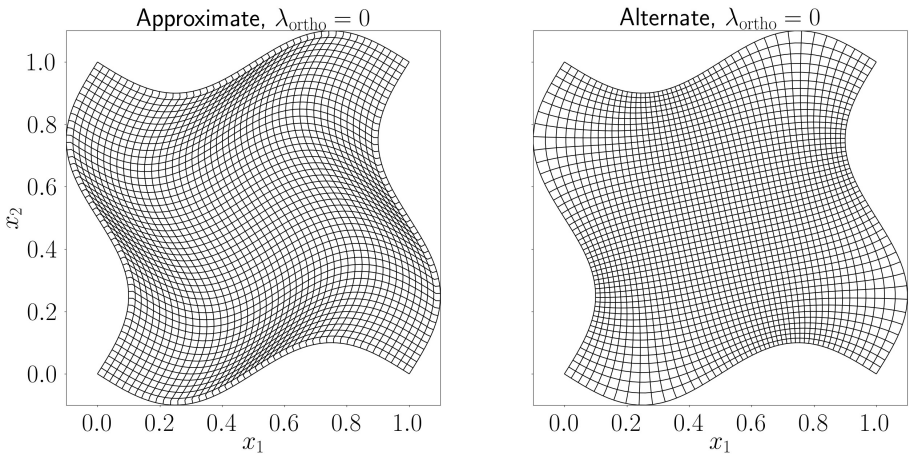

Figure 5: Final grids for the moderately curved domain case in section 3.3 using $\lambda_{\text{ortho}} = 0$.

was used. After a grid convergence study, we decided to use 80 points in each direction to solve the PDEs.

Figure 5 shows the results without orthogonality-promotion for the Approximate and Alternate methods; the Full method again did not converge. The resulting grids are quite different from each other, with the Approximate version creating a grid that responds to the metric field to a larger degree but also is highly skewed near the boundaries. The Alternate version seems to promote orthogonality over conforming to the metric.

The results with $\lambda_{\text{ortho}} = 5$ are shown in Fig. 6 for all three methods. The grids for the Full and Approximate versions are similar, but the grid from the Alternate version is different both in the center of the domain and especially near the outward-bulging parts of the boundary. To provide more nuanced information, the elements in the figure are colored by either the misfit in the first direction m_1 or the deviation from orthogonality Θ. Recall from Eqn. (3) that a positive misfit implies that the grid is locally too large in that direction of computational space (i.e., along the grid lines in that direction). We thus see that the Alternate version produces a grid that is slightly too fine in the center and too coarse near the bulges of the boundary.

The deviation from orthogonality was computed as $\Theta = |90^\circ - \Gamma|$ where Γ is the angle between two intersecting grid lines. Even with the orthogonality-promoting terms, the Full and Approximate versions produce slightly more skewed grids than the Alternate version does.

The optimal number of grid points ($n_\alpha = 1/\sigma_\alpha$) obtained from all the versions for different λ_{ortho} are assembled in Table 1. Due to symmetry, the number of elements is the same in both directions. The optimal number of elements is almost the same for the Full and Approximate versions, but somewhat higher for the Alternate version. We also notice that the variation in the number of

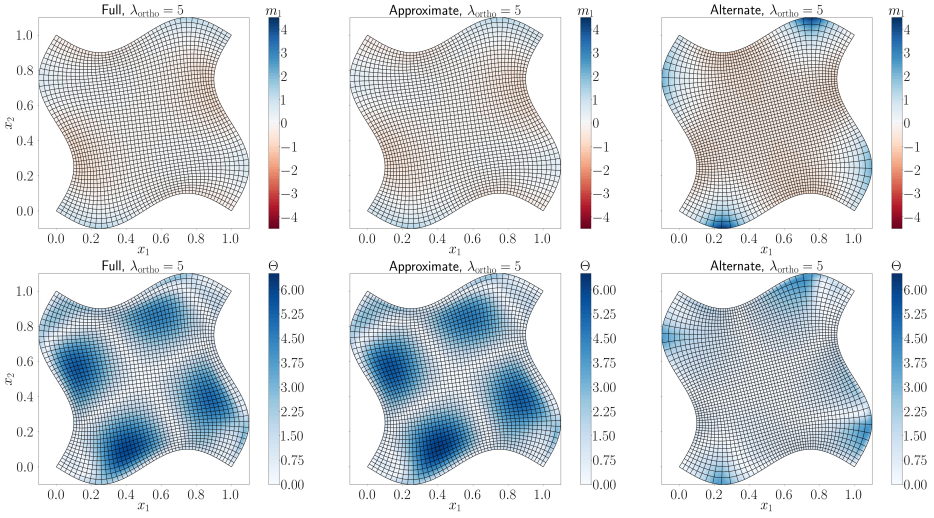

Figure 6: Final grids for the moderately curved domain case in section 3.3 using $\lambda_{\text{ortho}} = 5$, colored by the misfit in one direction and the deviation from orthogonality. The misfit m_1 is defined in Eqn. (3), the deviation from orthogonality Θ is measured in degrees.

λ_{ortho}	Full	Approximate	Alternate
0	-	46.7	48.8
0.5	-	46.3	48.9
1	46.2	46.3	49.0
1.5	46.4	46.4	49.2
2	46.4	46.4	49.3
5	46.7	46.7	50.6
8	46.8	46.8	-

Table 1: Optimal values of $1/\sigma_\alpha$ (=the number of elements) for the moderately curved domain case in section 3.3.

elements with respect to λ_{ortho} is negligible for all three versions.

All three versions also succeeded with $\lambda_{\text{ortho}} = 1$; however, the maximum value of Θ increased by approximately 15° for the Full version and 13° for the Approximate version (not shown). No significant change in the maximum value of Θ was noticed for the Alternate version.

We next quantify the impact of the orthogonality weight λ_{ortho} and the under-relaxation parameter ϵ on the convergence behavior in Fig. 7. Overall, having larger orthogonality weights requires more under-relaxation in order to have convergence. For the Approximate and Alternate versions, the number of iterations required to reach convergence increases with increasing λ_{ortho}. For the Full method, on the other hand, there is a "sweet spot" of $\lambda_{\text{ortho}} \approx 1.5 - 2$

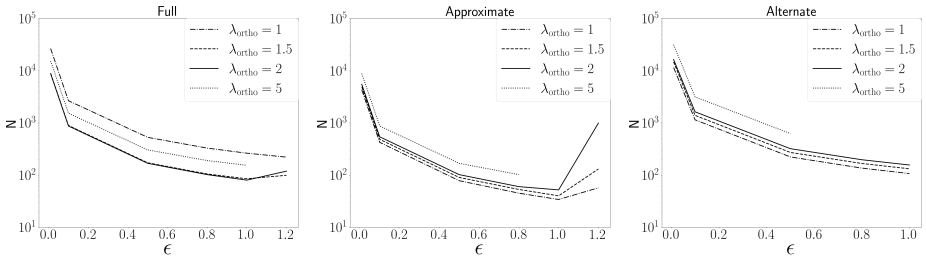

Figure 7: Number of iterations required for convergence for the moderately curved domain case in section 3.3. The curves cover only those values of the under-relaxation parameter ϵ that produced converged results.

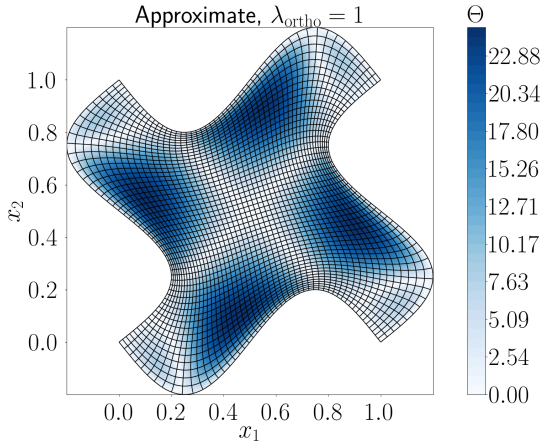

Figure 8: Final grid for the strongly curved domain case in section 3.4 using $\lambda_{\text{ortho}} = 1$, colored by the deviation from orthogonality Θ.

for which the convergence is the fastest.

These simple test cases collectively lead us to choose the Approximate method as the most promising going forward. This method produces essentially identical grids to the Full method, but with greater numerical robustness. In addition, it produces grids with less misfit than the Alternate method.

3.4 Strongly curved domain

We next consider the same case as in section 3.3 but with $A = 0.2$ which produces more strongly curved boundaries. The Approximate method was used, with 320 grid points in each direction and $\lambda_{\text{ortho}} = 1$. The resulting grid colored by the deviation from orthogonality is shown in Fig. 8. In a qualitative sense, the grid is quite reasonable given the strongly curved nature of the boundaries.

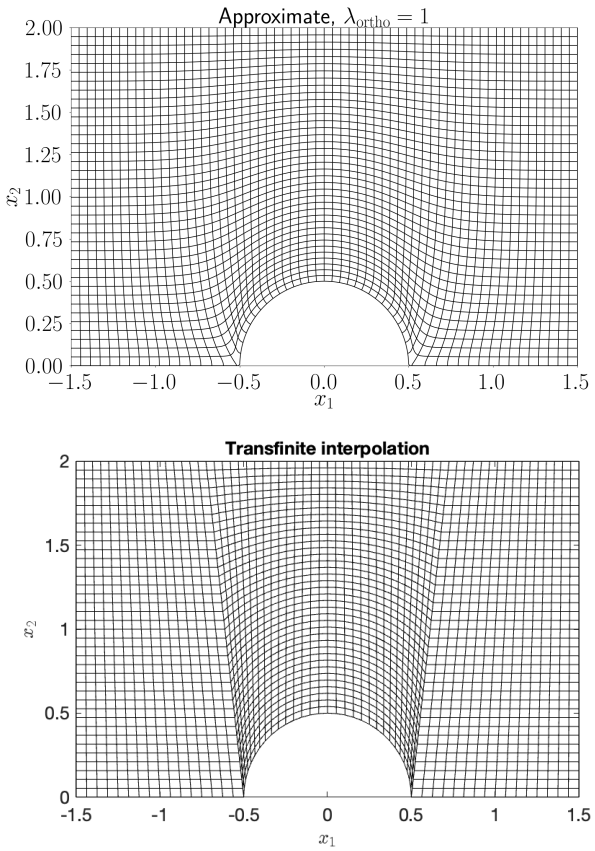

Figure 9: Non-smooth boundary case from section 3.5, comparing to the transfinite interpolation method.

3.5 Non-smooth boundary

To test how the proposed method handles non-smooth boundaries, this section considers a bottom boundary composed of a half-circle on top of a planar boundary. The half-circle is defined by the equation $x_1^2 + x_2^2 = 0.5^2$. The metric field for this case is taken as

$$M = \begin{bmatrix} 400 & 0 \\ 0 & 400 \end{bmatrix}. \tag{26}$$

The PDE was solved on a computational domain with 200 points in each direction, with the orthogonality weight $\lambda_{\text{ortho}} = 1$.

Figure 9 shows the resulting grid, and also the grid resulting from using the transfinite interpolation method with similar boundary distributions. The mesh produced by the proposed method is smooth and relatively orthogonal, in sharp contrast to the mesh produced by transfinite interpolation. We note how the

mesh responds to the non-smooth boundary throughout the full domain, while trying to maintain the specified uniform and isotropic grid resolution. We also note that the grid lines approach the bottom boundary in an almost orthogonal manner, as specified by the boundary conditions. The optimal number of grid points are approximately 62 and 38 in the x_1 and x_2 direction, respectively.

This test case was also used by Piperni [26] (Fig. 3 in that paper) who used an elliptic grid-generation method. The present grid is at least as good as their result in terms of skewness and adherence to the desired grid-spacings.

4 Assessment of numerical accuracy

While this manuscript is focused on the generation of a grid from a given Riemannian metric field, this section provides two examples for which the numerical error or accuracy in the computation is assessed.

4.1 Numerical error of gradient operation

A two-dimensional version of Runge's function is $u = \left(1 + 25\overline{x}^2 + 25\overline{y}^2\right)^{-1}$ with $\overline{x} = 2(x - 0.5)$ and $\overline{y} = 2(y - 0.5)$. The right-biased first-order accurate finite difference gradient is defined at grid point (i, j) as

$$\left.\frac{\partial u}{\partial s_\alpha}\right|_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta s_\alpha|_{i,j}} \ , \quad \alpha = 1, 2, \tag{27}$$

and the gradient ∇_{num} can then be computed after a coordinate transformation. The numerical error is then defined as

$$\mathcal{R} = \|\nabla_{\text{num}} u - \nabla u\|. \tag{28}$$

The numerical error can be modeled approximately by the model described in Eqn. (30) with $\eta = 1$ and c_α computed analytically from the second derivatives of Runge's function. Section 4.2 then describes how minimization of the integrated error yields the optimal grid-spacing $\Delta_{\alpha,\text{opt}}$ in Eqn. (38). This optimal grid-spacing is then encoded in a Riemannian metric field $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \ \mathbf{V} = \left[\mathbf{v}^{(1)}, \mathbf{v}^{(2)}\right], \ \mathbf{\Lambda} = \text{diag}\left(\Delta_{1,\text{opt}}^{-2}, \Delta_{2,\text{opt}}^{-2}\right), \tag{29}$$

with $\mathbf{v}^{(\alpha)}$ taken as the Cartesian basis vectors.

The grid-generation method is then used to create a sequence of grids for this metric field, with different numbers of elements and different skewness penalizations. The grids are shown in Fig. 10 and compared to a completely uniform grid with the same number of elements.

The figure also shows the convergence of the error under grid refinement. The metric-conforming grids produce lower error at all resolutions, by shifting the resolution towards the middle of the domain where the solution has the most curvature. The error is larger with larger orthogonality promotion parameter

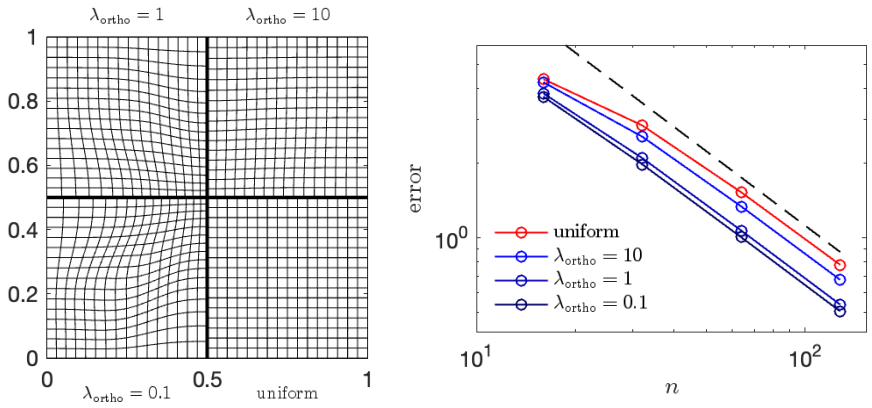

Figure 10: Convergence of numerical error (infinity norm) for the 2D Runge's function, with the dashed line showing the expected first-order accuracy. One quarter of each grid is shown in the figure.

λ_{ortho}, since this prevents the grid from adjusting towards the middle. The right balance between grid skewness and grid adjustment is solver- and problem-specific.

4.2 Grid-adaptation in large eddy simulation

The method for metric-conforming grid-generation is finally applied to the problem of finding adapted grids in wall-modeled large eddy simulation. The specific problem chosen here is the subsonic flow over a smooth backward-facing ramp, illustrated in Fig. 11. This problem is the subject of the High Fidelity CFD Verification workshop and working group[1], a summary of which can be found in [27] which also contains the full problem specification. The exact version of the problem considered here corresponds to the 2024 workshop, in which participants made blind predictions and then compared with DNS at the workshop (note that the 2024 case is different from 2022 case [28], with a lower Reynolds number and a narrower domain; thus comparison with the 2022 case are not appropriate). Prior to the workshop, the only DNS data released to participants are related to the incoming boundary layer; this is the data used for comparisons here.

The initial grid was created by manually specifying a metric field and then using the Approximate method with orthogonality promotion to generate the grid. This manual metric field was taken to have target grid-spacings in the streamwise, wall-normal and spanwise directions $(\Delta_{1,\text{target}}, \Delta_{2,\text{target}}, \Delta_{3,\text{target}})$ of $(0.12, 0.02, 0.08)\delta_{\text{ref}}$ at the bottom wall and $(0.36, 0.32, 0.32)\delta_{\text{ref}}$ at the top wall, with linear variation in between. These grid-spacings were defined in a coordinate system aligned with the walls and linearly varying in between.

[1] See wmles.umd.edu/working-group-on-wmles

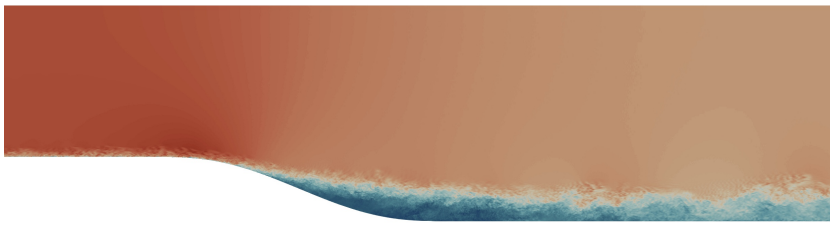

Figure 11: Instantaneous streamwise velocity in a wall-modeled large eddy simulation of the flow over a smooth ramp. The flow is from left to right.

Here, $\delta_{\text{ref}} = 0.028L$ is the nominal thickness of the incoming boundary layer and L is the length of smooth ramp [27]. The computational domain extends from $x/L = -1.6$ (inflow) to $x/L = 8$ (outflow), with the ramp located at $0 \leq x/L \leq 1$. The spanwise width of the domain is $0.288L$. The resulting initial grid, labeled "grid-0" here, has 40M cells.

The compressible Navier-Stokes equations are solved using an in-house finite-volume solved with four-point stencils used for interpolation and gradients (in the face-normal direction) at faces. The convective terms are computed by a kinetic-energy preserving method, blended with third-order WENO using the HLLC flux in regions marked by a sensor; for this problem, about 10-20% WENO is used in the potential flow regions of the flow, and essentially none in the turbulent regions. A Vreman subgrid model [29] is used, and an equilibrium wall-model [30].

After discarding the initial transient, about 10 instantaneous snapshots of the solution were stored to disk, and the directional LES residual $\mathcal{R}_\alpha$ (the approximate source of error due to LES under-resolution in each direction $\alpha = 1, 2, 3$ of the grid) was estimated using the method of Toosi and Larsson [31]. This estimate of the LES residual is based on heuristics about the unresolved subgrid motions, but is similar (for incompressible flows) to the interpolation-based estimator by Alauzet and Loseille [32]. The residual is modeled to vary with arbitrary grid-spacing Δ_α according to the model in Eqn. (30) with $\eta = 2$. The residual "densities" c_α are then computed from the estimated residuals $\mathcal{R}_\alpha$ and the actual grid-spacings Δ_α on the old grid. With this, the target grid-spacings $\Delta_{\alpha,\text{target}}$ for the next grid are computed according to Eqn. (38). These target grid-spacings are defined in the local coordinate system defined by the old grid; while likely not the optimal orientation, we nevertheless use this local coordinate system to define the target metric field $\mathbf{M}$ in the same manner as in the previous section. The resulting metric field is then used with the Approximate method to generate the first adapted grid, labeled "grid-1" here. In this process we aimed for a doubling of the number of cells; grid-1 ended up having 87M cells. The same process is then repeated with residual estimation on grid-1 leading to the creation of grid-2, with 193M cells, and grid-3 with 413M cells.

Key details of the grids are listed in Table 2, and visualized in Fig. 12. The

Case	$n_1 \times n_2 \times n_3$	$(\Delta x, \Delta y_w, \Delta z)/\delta_{\text{ref}}$
grid-0	$1380 \times 223 \times 130$	(0.15, 0.016, 0.079)
grid-1	$1907 \times 226 \times 202$	(0.10, 0.014, 0.051)
grid-2	$2640 \times 302 \times 242$	(0.073, 0.018, 0.043)
grid-3	$3418 \times 408 \times 296$	(0.056, 0.016, 0.035)

Table 2: Grid sizes and grid-spacing in the incoming boundary layer.

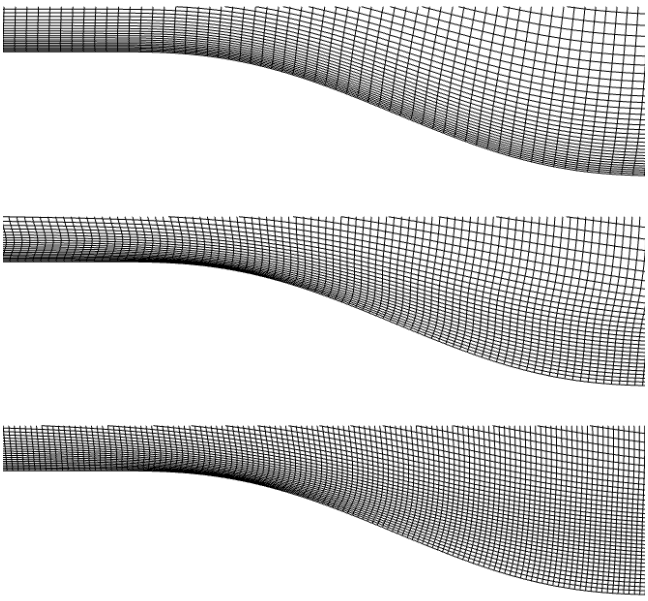

Figure 12: The initial grid (top) and the first two adapted grids for the smooth ramp problem, showing every 5th grid line.

first two grids (grid-0 and grid-1) use $\lambda_{\text{ortho}} = 1$; for the remaining grids this was increased to $\lambda_{\text{ortho}} = 2.5$ to promote higher quality near the bottom wall.

Some sample results are shown in Figs. 13, 14, and 15. For all quantities, the 95% confidence interval due to averaging was estimated using the method of Russo and Luchini [33]. As described briefly above, the only DNS data that available at the time of writing is for the incoming flow, specifically the mean velocity U and the Reynolds stress $\overline{u_i' u_j'}$ at three locations in the incoming boundary layer; the former also implies knowledge of the boundary layer thickness δ and skin friction coefficient c_f at those locations.

The skin friction coefficient converges towards the DNS results at the three locations, as does the mean velocity profile. The Reynolds stresses agree rather well with the DNS in the region outside of the wall-modeled layer, at least considering the fact that this is a wall-modeled LES on a much coarser grid than the DNS. The size of the separation bubble increases under grid-refinement, and

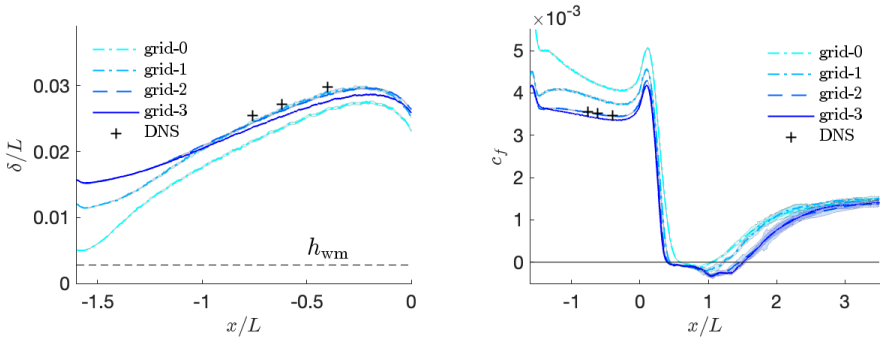

Figure 13: Boundary layer thickness in the incoming flow (left) and skin friction coefficient (right) for the smooth ramp problem.

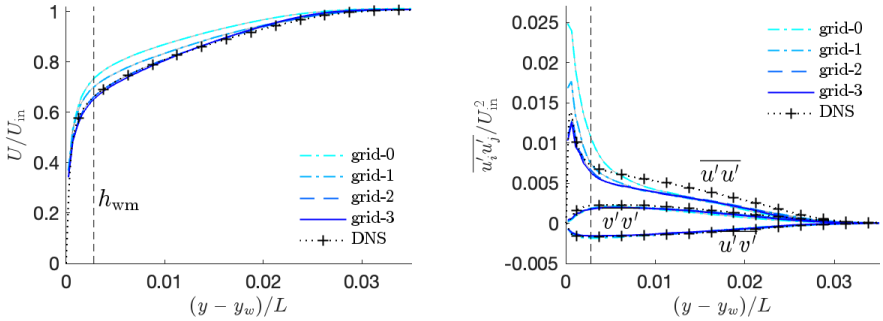

Figure 14: Mean velocity (left) and Reynolds stresses (right) at $x/L = -0.62$ for the smooth ramp problem.

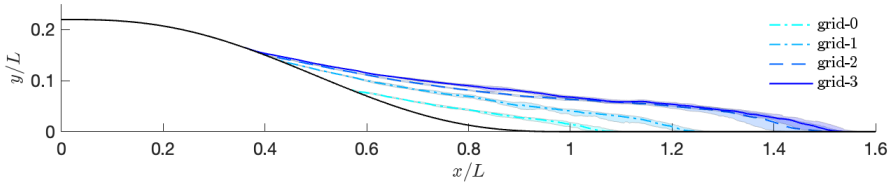

Figure 15: Contour of $U = 0$ for the smooth ramp problem.

is converged to within the 95% confidence interval of the averaging between the final two grids.

Summary

A novel algorithm for generating structured grids that conform to a given Riemannian metric field has been developed using a variational framework. By requiring a Riemannian metric field, the method is clearly meant to be used within a grid-adaptation framework, in which the optimal Riemannian metric field is produced from residual estimation or similar methods, as demonstrated in the smooth ramp example. The long-term objective is to reduce the need for human input (and experience) in the grid-generation process. However, it could also be used in a stand-alone manner, with the user then having to provide the desired metric field as opposed to the number of grid points and their distribution along the block edges.

Three different versions of the algorithm were developed and tested on a sequence of two-dimensional canonical problems. Based on the results, the two recommended versions of the algorithm are the "Approximate" method with the orthogonality-promoting terms and the "Alternate" method without the need for those. In general, the best approach is likely to first solve using the Alternate method and then to use that solution as a very good initial condition to the Approximate method. Typically $\lambda_{\text{ortho}} = 1$ was found to be sufficient to produce reasonable grids. The skewness of grids can affect the numerical accuracy of the solution, however over-compensating for it can cause the grids to move away from their metric-conforming nature. The right balance between the two is solver and problem-specific.

Acknowledgments

This work was supported by the NNSA Predictive Science Academic Alliance Program (PSAAP; grant DE-NA0003993) and by the NASA Transformational Tools and Technologies project (grant 80NSSC22M0297).

Appendix: Finding the optimal grid-spacing

A rather general model of the residual is

$$\mathcal{R} = \sum_{\alpha=1}^{d} c_\alpha \Delta_\alpha^\eta, \tag{30}$$

where d is the number of geometric dimensions of the problem. The optimal grid resolution should minimize the cost functional

$$\mathcal{J}_\Delta[\Delta_\alpha] = \int_\Omega \sum_\alpha c_\alpha \Delta_\alpha^\eta dV_x + \phi \left[\int_\Omega \Pi_\alpha \Delta_\alpha^{-1} dV_x - N \right], \tag{31}$$

where N is the desired number of elements, ϕ is a Lagrange multiplier, and the integrals are evaluated over the physical domain Ω with $dV_x = dx_1 dx_2 ....$ The

solution to this variational problem is the Euler-Lagrange equation

$$\eta c_\beta \Delta_\beta^{\eta-1} - \phi \frac{1}{\Delta_\beta} \Pi_\alpha \Delta_\alpha^{-1} = 0, \quad \beta = 1, 2, \ldots, d \tag{32}$$

and the constraint

$$N = \int_\Omega \Pi_\alpha \Delta_\alpha^{-1} dV_x. \tag{33}$$

Eqn. (32) can be re-written as

$$c_\beta \Delta_\beta^{\eta} = \frac{\phi}{\eta} \Pi_\alpha \Delta_\alpha^{-1}, \quad \beta = 1, 2, \ldots, d. \tag{34}$$

Multiplying this over all β (i.e., applying the operation Π_β) yields (changing to α as the dummy subscript)

$$\Pi_\alpha \Delta_\alpha^{-1} = \left(\frac{\phi}{\eta}\right)^{-d/(d+\eta)} \Pi_\alpha c_\alpha^{1/(d+\eta)}. \tag{35}$$

Inserting (35) into (33) and (34) yields, respectively,

$$c_\beta \Delta_\beta^{\eta} = \left(\frac{\phi}{\eta}\right)^{\eta/(d+\eta)} \Pi_\alpha c_\alpha^{1/(d+\eta)}, \quad \beta = 1, 2, \ldots, d, \tag{36}$$

$$N = \left(\frac{\phi}{\eta}\right)^{-d/(d+\eta)} \int \Pi_\alpha c_\alpha^{1/(d+\eta)} dV_x. \tag{37}$$

Eliminating the Lagrange multiplier between these equations then finally yields the optimal grid resolution in each direction as

$$\Delta_\beta = \left(N^{-1} \int \Pi_\alpha c_\alpha^{1/(d+\eta)} dV_x\right)^{1/d} \left(\frac{\Pi_\alpha c_\alpha^{1/(d+\eta)}}{c_\beta}\right)^{1/\eta}, \quad \beta = 1, 2, \ldots, d. \tag{38}$$

References

[1] L. Remaki and W. Habashi. Pacing CFD: automatic mesh adaptation as an efficient tool to improve CFD accuracy. *Int. J. CFD*, 19(8):571–580, 2005.

[2] F. E. Ham, F. S. Lien, and A. B. Strong. A Cartesian grid method with transient anisotropic adaptation. *J. Comput. Phys.*, 179:469–494, 2002.

[3] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Comp. Meth. Appl. Mech. Eng.*, 194:5068–5082, 2005.

[4] K. J. Fidkowski and D. L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA J.*, 49:673–694, 2011.

[5] S. Toosi and J. Larsson. The Germano identity error and the residual of the LES governing equation. *J. Comput. Phys.*, 443:110544, 2021.

[6] A. Loseille and F. Alauzet. Continuous mesh framework part i: well-posed continuous interpolation error. *SIAM J. Num. Ana.*, 49(1):38–60, 2011.

[7] A. Loseille and F. Alauzet. Continuous mesh framework part ii: validations and applications. *SIAM J. Num. Ana.*, 49(1):61–86, 2011.

[8] M. Yano. An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes. PhD thesis, Massachusetts Institute of Technology, 2012.

[9] K. J. Fidkowski. A local sampling approach to anisotropic metric-based mesh optimization. *54th AIAA Aerospace Sciences Meeting, AIAA 2016-0835*, 2016.

[10] H. Borouchaki, P. L. George, F. Hecht, P. Laug, and E. Saltel. Delaunay mesh generation governed by metric specifications. part i. algorithms. *Finite Elements in Analysis and Design*, 25:61–83, 1997.

[11] K. J. Fidkowski and G. Chen. Metric-based, goal-oriented mesh adaptation using machine learning. *J. Comput. Phys.*, 426:109957, 2021.

[12] D. P. Sanjaya, K. J. Fidkowski, L. T. Diosady, and S. M. Murman. Error minimization via metric-based curved-mesh adaptation. *23rd AIAA Computational Fluid Dynamics Conference, AIAA 2017–3099*, 2017.

[13] D. P. Sanjaya, K. J. Fidkowski, and S. M. Murman. Comparison of algorithms for high-order, metric-based mesh optimization. *AIAA Scitech 2020 Forum, AIAA 2020–1141*, 2020.

[14] A. Belme, F. Alauzet, and A. Dervieux. An a priori anisotropic goal-oriented error estimate for viscous compressible flow and application to mesh adaptation. *J. Comput. Phys.*, 376:1051–1088, 2019.

[15] F. Alauzet and A. Loseille. A decade of progress on anisotropic mesh adaptation for computational fluid dynamics. *Computer-Aided Design*, 72:13–39, 2016.

[16] A. Dervieux, F. Alauzet, A. Loseille, and B. Koobus. *Mesh Adaptation for Computational Fluid Dynamics, Volume 2: Unsteady and Goal-oriented Adaptation*. John Wiley & Sons, 2022.

[17] M. Giacomini and S. Perotto. Anisotropic mesh adaptation for region-based segmentation accounting for image spatial information. *Comp. Math. Appl.*, 121:1–17, 2022.

[18] J. G. Wallwork, N. Barral, D. A. Ham, and M. D. Piggott. Goal-oriented error estimation and mesh adaptation for tracer transport modelling. *Computer-Aided Design*, 145:103187, 2022.

[19] A. Balan, M. A. Park, S. L. Wood, W. K. Anderson, A. Rangarajan, D. P. Sanjaya, and G. May. A review and comparison of error estimators for anisotropic mesh adaptation for flow simulations. *Comp. Fluids*, 234:105259, 2022.

[20] Z. Ali, P. C. Dhanasekaran, P. G. Tucker, R. Watson, and S. Shahpar. Optimal multi-block mesh generation for CFD. *Int. J. CFD*, 31(4-5):195–213, 2017.

[21] T. Misaka, D. Sasaki, and S. Obayashi. Adaptive mesh refinement and load balancing based on multi-level block-structured Cartesian mesh. *Int. J. CFD*, 31(10):476–487, 2017.

[22] X. Gao and C. P. T. Groth. A parallel adaptive mesh refinement algorithm for predicting turbulent non-premixed combusting flows. *Int. J. CFD*, 20(5):349–357, 2006.

[23] T. Sumi, T. Kurotaki, and J. Hiyama. Interpolated characteristic interface conditions for zonal grid refinement of high-order multi-block computations. *Int. J. CFD*, 26(1):23–43, 2012.

[24] V. D. Liseikin. Grid generation methods. *2nd edition, Springer, Berlin*, 2010.

[25] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[26] P. Piperni. Differential operator for the modeling of curvature in structured grid generation. *AIAA J.*, 56:4453–4462, 2018.

[27] J. Larsson, I. Bermejo-Moreno, and 21 others. Summary of the smooth body separation test case at the 2022 high fidelity CFD verification workshop. AIAA Paper 2023-1241, 2023.

[28] D. P. Rizzetta and D. J. Garmann. Wall-resolved large-eddy simulation of smooth-body separated flow. *Int. J. CFD*, 36:1–20, 2022.

[29] A. W. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Phys. Fluids*, 16(10):3670–3681, 2004.

[30] S. Kawai and J. Larsson. Wall-modeling in large eddy simulation: length scales, grid resolution and accuracy. *Phys. Fluids*, 24:015105, 2012.

[31] S. Toosi and J. Larsson. Anisotropic grid-adaptation in large eddy simulations. *Comp. Fluids*, 156:146–161, 2017.

[32] F. Alauzet and A. Loseille. High-order sonic boom modeling based on adaptive methods. *J. Comput. Phys.*, 2010:561–593, 2010.

[33] S. Russo and P. Luchini. A fast algorithm for the estimation of statistical error in DNS (or experimental) time averages. *J. Comput. Phys.*, 347:328–340, 2017.