AIAA 2018-1501

# Robust metric-aligned quad-dominant meshing using $L_p$ centroidal Voronoi tessellation

Dirk Ekelschot*, Marco Ceze†, Anirban Garai‡, and Scott Murman§
*NASA Ames Research Center, Moffett Field, CA, USA*

**We introduce a meshing algorithm that can be used to both generate and adapt meshes for bounded domains in an anisotropic manner. This is particularly beneficial when anisotropic flow features like shock waves or contact discontinuities are present in the computational domain. The algorithm presented in this paper is based upon meshing under the imposed Riemannian metric tensor, which controls the orientation and size of the mesh elements. In this way there is no need for user intervention to recognize these features. We demonstrate that the method indeed aligns the elements with the underlying metric and produces right-angled simplices that can be recombined into quadrilateral elements. The aim is to eventually incorporate this meshing strategy in the monolithic high-order spectral element solver that is currently being developed at NASA Ames. This paper has two main contributions: First, we demonstrate that we can generate quad-dominant metric-aligned meshes for bounded domains using a generalized form of $L_p$-Centroidal Voronoi Tessellation ($L_p$-CVT). Unlike previous works, we do not rely on a background mesh and discretize the bounded domain in a hierarchical way by first discretizing the boundaries and then the volume using the underlying metric. Second, we present an alternative for clipping the Voronoi cells on the boundary, which is common practice in CVT-based meshing algorithms, by reconstructing the Voronoi cells using the defined metric field. In this way we avoid the geometrical complexity of the clipping procedure and we show that we evaluate the energy and its gradient correctly. We show that the reconstruction of the computational domain is consistent with the Lloyds' algorithm that is used to compute the $L_p$-CVT.**

*Post-doctoral fellow with Universities Space Research Association.
†Former post-doctoral fellow with Universities Space Research Association.
‡Science and Technology Corporation
§AIAA member.

# I. Introduction

Despite decades of investment from government, academia, and industry, mesh generation is still limiting the performance of algorithms and workflows for Computational Fluid Dynamics (CFD) in the majority of aerospace applications. NASA's CFD Vision 2030 Study specifically highlights mesh generation and adaptivity as significant bottlenecks that need to be addressed [1]. Our research group is developing a Discontinuous-Galerkin spectral-element solver for scale-resolving simulations [2–4]. These types of high-order *h*-*p* adaptive schemes add further complexity to the mesh generation problem in that the boundary-conforming elements must be curved, while the majority of existing mesh generation algorithms and software assume planar approximations. Next to that, we are interested in solving internal and external compressible flow problems that exhibit anisotropic flow features like shock waves and contact discontinuities. With this in mind, we started this effort to develop a mesh generation algorithm that allows us to generate anisotropic meshes depending on an arbitrary metric that is delivered by the high-order spectral element solver. The current work represents the first step towards developing this reliable adaptive mesh capability.

A robust algorithm for generating a hexahedral mesh is to subdivide each element of a tetrahedral mesh, however the resulting mesh is unsuitable for most applications due to poor quality. Clearly robust methods alone are insufficient, and must be driven by accuracy or error minimization to be effective. The most common method of addressing this issue for hex-dominant meshing is to utilize a Cartesian method that restricts the cell types to ideal cubes. Examples include immersed-boundary [5], cut-cells [6], lattice Boltzmann methods [7], etc. While these methods are based upon robust geometric algorithms, they suffer from two major drawbacks - the computational cells are both coordinate-aligned and isotropic. Algorithmic approaches which allow boundary-conforming and anisotropic hexahedral elements include submapping [8] advancing-front decompositions [9], octree-based [10], and sweeping methods [11], among others. These approaches typically require some form of heuristics and/or do not guarantee practical accuracy. The standpoint taken here is that suitable mesh generation strategies should be capable of providing reliable adaptive capabilities and must be built upon a foundation of provable mathematical algorithms - *ad-hoc*, manual, and empirical methods are the leading culprit in the current limitations. Metric-based mesh adaptation and generation provides this mathematical foundation.

Alauzet [12] provides a nice review of the current state-of-the-art of metric-based mesh adaptation and generation [13–16]. The idea to generate anisotropic meshes by computing a unit mesh in a given Riemannian metric space rather than the classical Euclidean space was first documented by Hecht and Mohammadi [17]. The metric field is used to locally describe how length is measured along a manifold, and in this way, controls the orientation and the size of each element. This metric can be derived from surface curvature, feature-based adaption, truncation error estimates, output-based error predictions, etc. The source of the metric tensor however is not important in the scope of the current work. Loseille and Alauzet [18, 19] proposed a continuous mesh model that is based on Riemannian metric spaces. They introduced the concept of constructing a continuous metric field and generating a discrete mesh that conforms

to this metric field. More recently, Loseille [14] proposes an algorithm that generates metric-orthogonal anisotropic meshes where locally an anisotropic structured mesh is created. This is done iteratively by inserting vertices using a frontal approach taking into account the eigenvectors and eigenvalues that are prescribed by the underlying Riemannian metric space. However, most of these approaches rely on an estimate of the interpolation error based on the Hessian of one of the solution variables.

Levy and Lui [20] introduced $L_p$-Centroidal Voronoi Tessellation ($L_p$-CVT). This method minimizes an energy functional equal to the sum of the $L_p$ moment of inertia of the Voronoi cells. They essentially generalized the classical (i.e. in Euclidean space) CVT approach by rewriting the energy functional such that a metric tensor is incorporated and that $L_p$ distances are measured with respect to this metric. They applied this method mainly to periodic domains. The fundamental goal here is to align edges of the computational elements with the characteristic vectors of the metric tensor, while the magnitude of the characteristic values determine the element sizing. In regions where the metric is small the elements will be large, and *vice versa* when the metric is large. The meshing is thus both controllable, by modifying the metric tensor, and automatic, requiring no user intervention to recognize features. By incorporating a higher $p$-norm in the energy function, the Lloyd's algorithm returns right-angled triangles which can then be easily recombined to quadrilateral shaped elements.

More recently, Baudouin et al. [21] also presented an $L_p$-CVT algorithm for quadrilateral surface mesh generation. However they compute the energy and its gradient using Gaussian integration techniques as opposed to analytical formulas that were used by Levy and Lui [20]. Furthermore, they are interested in bounded domains and use a similar Voronoi clipping strategy as presented by Levy and Lui [20]. Caplan et al. [22] also pursue a more mathematically rigorous mesh generation approach. They discuss a Delaunay-based anisotropic mesh generation algorithm that relies on an embedding algorithm that transforms the anisotropic problem to a uniform problem using a Riemannian metric field. They demonstrate the isometrically embedding of arbitrary mesh-metric pairs in higher dimensional Euclidean spaces. Once the mesh-metric pair is embedded, isotropic mesh generation strategies are applied in the embedded space. However the emphasis in this work lies on the generation of meshes that consists of simplices.

In contrast to the aforementioned papers, we are demonstrating a prototype of a metric-aligned quad-dominant meshing algorithm for bounded domains that relies on $L_p$-CVT. This algorithm generates the mesh in a hierarchical way by first meshing the boundary edges by minimizing the energy such that the boundary edge is subdivided into small edges that are all unit length under the given metric. Once the locations of the nodes on the boundary are determined, we use the same energy minimization algorithm for the internal nodes similarly to what has been shown by [20, 21]. The previous works rely on clipping the Voronoi cells on the boundary. However, here we show that by reconstructing the Voronoi cells on the boundary using the underlying metric, we achieve an improved energy gradient computation. By reconstructing the Voronoi cells we avoid the geometrical complexity of the clipping problem. We demonstrate that the reconstruction can be done locally.

The outline of the paper is as follows: First, a description of the proposed anisotropic quad-dominant mesh generation algorithm given in section II. In section III, some preliminary results are discussed that demonstrate the proof-of-concept of the proposed algorithm. Finally, conclusions are drawn and the necessary steps that need to follow are discussed in section IV.

## II. Metric-aligned quad-dominant meshing algorithm

The current work uses a modified implementation of Levy and Liu's $L_p$ Centroidal Voronoi tessellation (CVT) algorithm [20]. A CVT is a Voronoi tessellation whose generating points are the centers of mass, also referred to as centroids, of the corresponding Voronoi regions. The CVT is obtained by minimizing the CVT energy functional [23] over the Voronoi domain. This energy functional and its gradient that govern this minimization procedure are explained in more detail in sections A and B. Levy and Lui [20] include both a metric and a $L_p$-norm in the energy functional in order to align the elements with the metric and to enforce right-angled corners. The right-angled corners of the elements are a result of measuring distances in $L_p$-norm. Distances in $L_p$-norm are determined by

$$\|\mathbf{y} - \mathbf{x}\|_p^p = |y_1 - x_1|^p + |y_2 - x_2|^p \tag{1}$$

In Fig. 1, the effect of measuring distance using an $L_p$-norm in Euclidean space is demonstrated by plotting the unit circle for different $p$-norms. Hence, the goal here is to enforce right-angled simplices that eventually are easily recombined

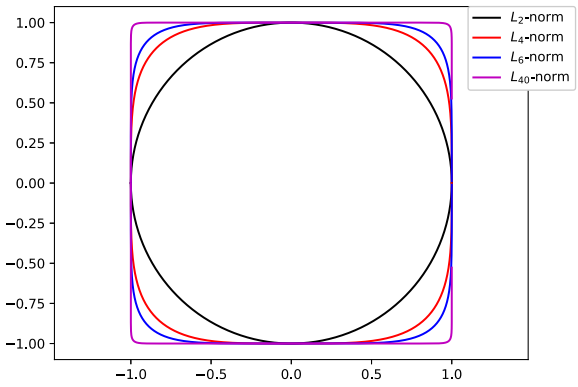



**Fig. 1 Unit circles for a range of $L_p$-norms.**

to generate quadrilateral elements. Next to that, the metric field controls the orientation (alignment and size) of the elements.

### A. Energy functional

In Fig. 2, a sketch is given of a Delaunay triangulation around a point $\mathbf{x}_i$ and its corresponding Voronoi cell is defined by the surrounding Voronoi vertices (red dots) and the edges (red lines) that connect the Voronoi vertices. Each Delaunay site has a corresponding Voronoi cell and Lloyds' algorithm essentially minimizes the energy functional which is equal to the sum of $L_p$ moment of inertia of these Voronoi cells. Hence the classical minimization problem in $L_2$ states that the following energy functional should be minimized:

$$E(\mathbf{x}_1, ..., \mathbf{x}_n) = \sum_{i=1}^{n} \int_{\Omega_i \cap \Omega} \|\mathbf{y} - \mathbf{x}_i\|^2 d\mathbf{y} \qquad i = 1, ..., n \tag{2}$$

where $\mathbf{x}_i$ represent the coordinates of the internal mesh nodes for which we want to find the optimal solution provided that the energy is minimum and $\mathbf{y}$ represents the coordinates over which we integrate. Furthermore, $\Omega$ denotes the computational domain that is spanned by the Voronoi diagram and $\Omega_i$ is a Voronoi region that belongs to vertex $\mathbf{x_i}$. The number of mesh vertices is indicated by $n$. As mentioned before, Levy and Liu generalize the CVT energy to include a metric tensor, $M$, and arbitrary $p$-norm:

$$E_{L_p}(\mathbf{x}_1, ..., \mathbf{x}_n) = \sum_{i=1}^{n} \int_{\Omega_i \cap \Omega} \|M(\mathbf{y})\,(\mathbf{y} - \mathbf{x_i})\|_p^p d\mathbf{y} \qquad i = 1, ..., n \tag{3}$$

The anisotropy of the mesh is essentially determined by the symmetric positive definite metric tensor $\mathbf{g}$. By taking the singular-value decomposition, we obtain

$$\mathbf{g} = \mathbf{R^t \Lambda R} = \mathbf{M^t M} \tag{4}$$

where the columns of $M$ are the axes of anisotropy ellipsoid. The geodesic distance is then given by

$$l_M(\vec{\mathbf{dx}}) = \sqrt{\vec{\mathbf{dx}}^{\,t}\, \mathbf{g} \vec{\mathbf{dx}}} = \sqrt{\vec{\mathbf{dx}}^{\,t} M^t M \vec{\mathbf{dx}}} = \sqrt{\left(M\vec{\mathbf{dx}}\right)^t M \vec{\mathbf{dx}}} \tag{5}$$

Eqn. (3) thus utilizes a generalization of the geodesic distance to a $p$-norm. Typically, the introduced minimization problem is solved using Lloyds' algorithm [24] or using a quasi-Newton method. Here, we use BFGS which is a quasi-Newton algorithm. This method evaluates the energy functional provided in Eqn. (3) and its gradient. The required gradients of Eqn. (3) with respect to the positions of the internal nodes, $\mathbf{x}_i$, are available in closed form and we therefore follow a similar approach as the one described by Baudouin et al. [21].

### B. Gradient of the energy functional

Baudouin et al. [21] evaluate both the energy functional and its gradient by splitting each Voronoi cell into simplices. They use Gaussian integration techniques to evaluate the integral term in Eqn. (3). The same idea is adopted here.

A sketch of a Voronoi Delaunay vertex, its corresponding Voronoi cell and its direct neighbors is shown in Fig. 2. The energy integral over $\Omega_i$ is evaluated by summing over the simplices. Considering simplex $S$ that is part of $\Omega_i$,

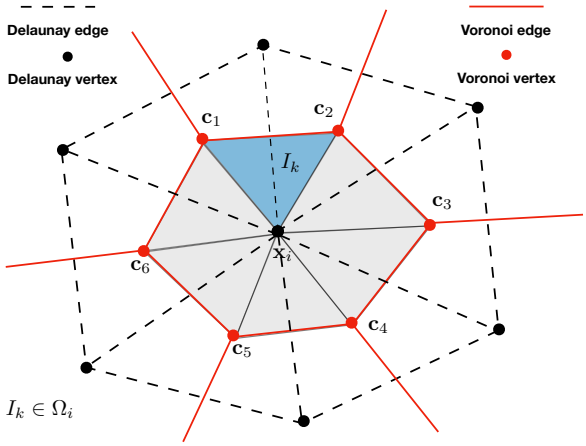



**Fig. 2 Sketch of a Voronoi cell $\Omega_i$ that corresponds to Delaunay vertex $\mathbf{x}_i$**

(highlighted in blue in Fig. 2) the energy integral $I_S$, can then be written as

$$I_S(\mathbf{x}_i) = \int_S \| M(\mathbf{y})\,(\mathbf{y} - \mathbf{x_i})\,\|_p^p d\mathbf{y} = \int_{S'} \| M(\mathbf{T}(\boldsymbol{\xi}))\,(\mathbf{T}(\boldsymbol{\xi}) - \mathbf{x_i})\,\|_p^p J d\boldsymbol{\xi} \tag{6}$$

where $\mathbf{y} = \mathbf{T}(\boldsymbol{\xi})$ represents the linear transformation of local element coordinates to standard element coordinates ($\boldsymbol{\xi}$) and $J$ represents the determinant of the Jacobian ($J = |\frac{\partial \mathbf{y}}{\partial \boldsymbol{\xi}}|$) of the linear transformation [21]. We use Gaussian integration techniques for triangles that were presented by Dunavant [25]. We use the following formula to compute the gradient of the energy with respect TO the node $\mathbf{x}_i$[21]:

$$\frac{d}{d\mathbf{x}_k} E(\mathbf{x}_1, ..., \mathbf{x}_n) = \frac{\partial I^{\Omega_k}(\mathbf{x}_k)}{\partial \mathbf{x}_k} + \sum_{i=1}^{n} \sum_{j=1}^{m_i} \frac{dI^{\Omega_i}(\mathbf{x}_i)}{d\mathbf{c}_{i,j}} \frac{d\mathbf{c}_{i,j}}{d\mathbf{x}_k} \qquad j = 1, ..., n \tag{7}$$

The symbol $m_i$ represents the number of Voronoi vertices that surround the $i^{th}$ Delaunay vertex. For the Voronoi cell that is shown in Fig. 2 we have that $m_i = 6$ but this may vary for each Delaunay vertex. $I^{\Omega_i}$ represents the energy contribution determined by Eqn. (6) for which holds that $I^{\Omega_i} = \sum_m I_{S_m}$. The first term on the right hand side of Eqn. (7) can be written as

$$\frac{\partial I_S(\mathbf{x}_i)}{\partial \mathbf{x}_i} = \int_{S'} \frac{\partial \|F\|_p^p}{\partial F} \left[ \frac{\partial M}{\partial \mathbf{T}} \frac{\partial \mathbf{T}}{\partial \mathbf{x}_i} (\mathbf{T}(\boldsymbol{\xi}) - \mathbf{x}_i) + M \left( \frac{\partial \mathbf{T}}{\partial \mathbf{x_i}} - 1 \right) \right] J + \|F\|_p^p \frac{\partial J}{\partial \mathbf{x}_i} d\boldsymbol{\xi} \tag{8}$$

where $F = M(\mathbf{y})\,(\mathbf{y} - \mathbf{x_i})$. Taking again simplex $S$ in Fig. 2 as an example, we see that it has two Voronoi vertices, $\mathbf{c}_{i,1}$

and $\mathbf{c}_{i,2}$. To calculate their contributions to the gradient of the energy with respect to the Delaunay vertex $\mathbf{x}_i$, we use

$$\frac{\partial I_S(\mathbf{x}_i)}{\partial \mathbf{c}_{i,j}} = \int_{S'} \frac{\partial \|F\|_p^p}{\partial F} \left[ \frac{\partial M}{\partial \mathbf{T}} \frac{\partial \mathbf{T}}{\partial \mathbf{c}_{i,j}} (\mathbf{T}(\boldsymbol{\xi}) - \mathbf{x}_i) + M \left( \frac{\partial \mathbf{T}}{\partial \mathbf{c}_{i,j}} \right) \right] J + \|F\|_p^p \frac{\partial J}{\partial \mathbf{c}_{i,j}} d\boldsymbol{\xi} \tag{9}$$

### C. Reconstruction of the boundary Voronoi cells

Baudouin et al. [21] rely on a clipped Voronoi diagram to solve the previously introduced minimization problem. Clipping the Voronoi diagram essentially means that the Voronoi cells that correspond to the boundary vertices are cut off. Consequently, they derive specialized forms of the gradient formula depending on whether the vertex of interest is connected to a boundary vertex or not. In this section we show that this results in an incorrect evaluation of the energy and its gradient and instead, we propose to reconstruct the Voronoi cells on the boundary.

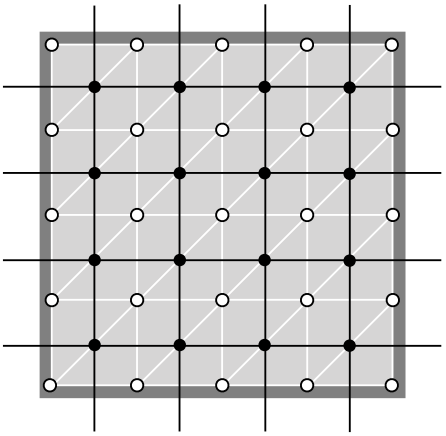
To demonstrate the difference between clipping the Voronoi cells and reconstructing them, we consider a simple square domain with a underlying metric tensor that is equal to the identity matrix. The expected minimum energy state in this case ($p \to \infty$) is a uniform lattice (see Fig. 3a). The white dots and lines represent the Delaunay vertices and edges respectively. The black dots and lines represent the Voronoi vertices and edges that build up the Voronoi diagram. The thick grey line represents the boundary of the computational domain. The uniform lattice with the clipped Voronoi diagram are shown in Fig. 3b. Extra Voronoi vertices are introduced at the halfway points of each boundary edge and the Voronoi regions corresponding to the boundary vertices are cut off. Levy's algorithm [20], which we have adopted here, is seeking a minimum energy solution such that all vertices are the centroids of their own Voronoi cell. This is contradicted by clipping the Voronoi diagram in this way. Fig. 3b shows that the Delaunay vertices on the boundary (red and blue) are not in the centroidal location of their Voronoi cell. The colors in Fig. 3b indicate the individual energy contribution of each Voronoi cell. Hence, the algorithm gets violated by clipping the Voronoi cells on the boundary since the Delaunay vertices on the boundary are fixed.

Mathematically, this argument can be reinforced by looking at the gradient of the energy functional [26, 27]. Generally, the gradient of the energy should be zero once the energy is minimized. Hence, the Fréchet derivative of the energy can be written as:

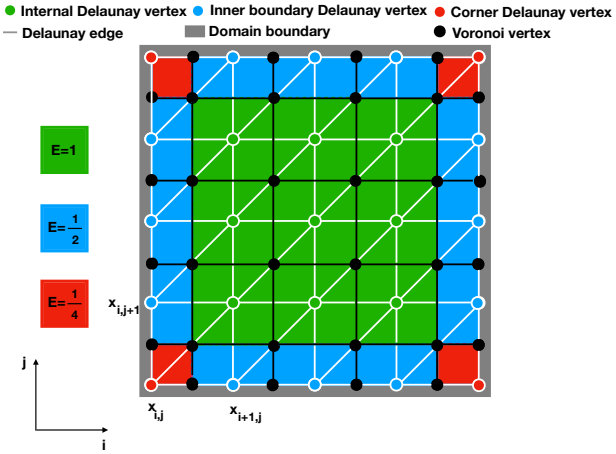
$$\lim_{\epsilon \to 0} \frac{E(\mathbf{x}_i + \epsilon \mathbf{u}) - E(\mathbf{x}_i)}{\epsilon} = 0 \tag{10}$$

For simplicity, we assume that $p = 2$, we can write Eqn. (10) as

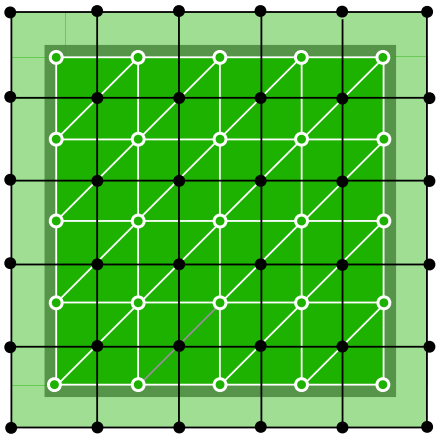
$$\begin{aligned}
\lim_{\epsilon \to 0} \frac{E(\mathbf{x}_i + \epsilon \mathbf{u}) - E(\mathbf{x}_i)}{\epsilon} &= \lim_{\epsilon \to 0} \left[ \frac{1}{\epsilon} \int_{\Omega_i} (\mathbf{y} - \mathbf{x}_i - \epsilon \mathbf{u})^t M(\mathbf{y}) (\mathbf{y} - \mathbf{x}_i - \epsilon \mathbf{u}) d\mathbf{y} - \frac{1}{\epsilon} \int_{\Omega_i} (\mathbf{y} - \mathbf{x}_i)^t M(\mathbf{y}) (\mathbf{y} - \mathbf{x}_i) d\mathbf{y} \right] \\
&= \lim_{\epsilon \to 0} \left[ \int_{\Omega_i} -2\mathbf{y}^t M(\mathbf{y}) \mathbf{u} + 2\mathbf{x}_i^t M(\mathbf{y}) \mathbf{u} + \epsilon \mathbf{u}^t M(\mathbf{y}) \mathbf{u} d\mathbf{y} \right] \\
&= 2 \lim_{\epsilon \to 0} \left[ \int_{\Omega_i} \mathbf{u} M(\mathbf{y}) (\mathbf{x_i} - \mathbf{y}) d\mathbf{y} \right] = 0
\end{aligned} \tag{11}$$



(a) Uniform lattice plotted with its Voronoi diagram.




(b) Sketch of a clipped Voronoi diagram.



(c) Sketch of a reconstructed Voronoi diagram

**Fig. 3 A sketch of the energy distribution for a square domain with unit metric (a) when the domain is clipped and (b) when the boundary Voronoi cells are reconstructed.**

This is only zero when $\mathbf{y} = \mathbf{x}_i$. This states that in order to find the minimum of the energy functional, we have to ensure in general that the vertices that span the Voronoi diagram should be in the centroid of their Voronoi cells. In the clipped configuration, which is shown in Fig. 3b, this is not the case. This problem is solved by reconstructing the Voronoi cells on the boundary using the underlying metric and the assumption that for a sufficiently high $p$-norm, the corners of the Voronoi cells are going to be right-angled at a minimum energy state. The reconstruction procedure is graphically presented in Fig. 4. The location of point $A$ is found by reconstructing a line that is in the direction

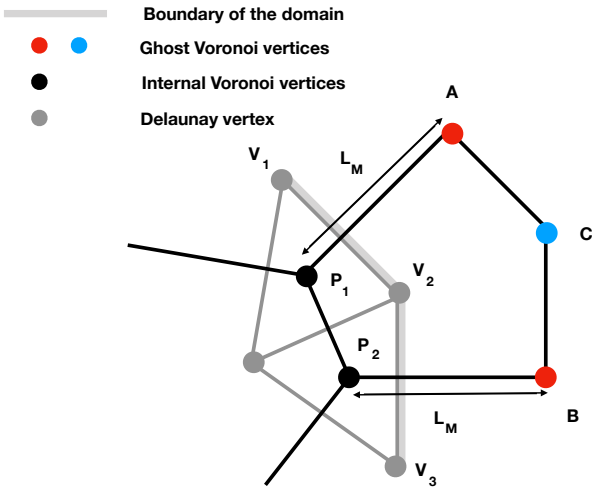



**Fig. 4 Sketch of the Voronoi cell reconstruction at the boundary.**

perpendicular to boundary edge $V_1$-$V_2$ and has unit length under the metric ($L_M = 1$ in Fig. 4). The same is done to find the location of point $B$ but now we use the boundary edge $V_2$-$V_3$ to determine the direction with respect to $P_2$. Point $C$ is found by taking the tangential directions from point $A$ and point $B$ and we find the intersection between those edges. Note that this procedure is entirely local to each boundary Voronoi cell and is straightforward to extend to arbitrary dimensions.

We compared the energy and their gradients for the clipped and reconstructed implementation. This was done by computing the energy and their gradients at an expected minimal energy state. Hence we defined a computational domain $\Omega = [0, 4]^2$ and defined a metric tensor that is equal to the identity matrix everywhere in the domain. Furthermore we used $p = 6$. The expected result would be a uniform lattice like the one drawn in Fig. 3. For this case, the gradient is expected to be zero for each internal vertex. We introduced the internal vertices at these expected locations and evaluate both the energy and its gradient. The energy gradient values are recorded for both implementations in Table 1.

Note as well that the vertex in the middle ($\mathbf{x}_{\mathbf{2,2}}$) has a gradient value in both directions that is $O(10^{-16})$. However, the gradients for vertex $\mathbf{x}_{\mathbf{3,1}}$ do not match this order of accuracy and are in the order $O(10^{-7})$. This can be ascribed to the fact that the energy gradient associated to a vertex that is connected to a boundary vertex is not evaluated properly.

**Table 1 Values of energy gradient for the case that the Voronoi diagram is clipped ($\frac{dE}{d\mathbf{x}_i}_C$) and for the case that the Voronoi cells on the boundary are reconstructed ($\frac{dE}{d\mathbf{x}_i}_R$).**

| $x_i$ | 1.0 | 1.0 | 1.0 | 2.0 | 2.0 | 2.0 | 3.0 | 3.0 | 3.0 |
|---|---|---|---|---|---|---|---|---|---|
| $y_i$ | 1.0 | 2.0 | 3.0 | 1.0 | 2.0 | 3.0 | 1.0 | 2.0 | 3.0 |
| $\frac{dE}{dx_i}_C$ | 3.13$e$-07 | 2.65$e$-07 | 2.16$e$-07 | 4.89$e$-08 | -1.46$e$-16 | -4.89$e$-08 | -2.16$e$-07 | -2.65$e$-07 | -3.13$e$-07 |
| $\frac{dE}{dy_i}_C$ | 2.16$e$-07 | -4.89$e$-08 | -3.13$e$-07 | 2.65$e$-07 | -7.63$e-17$ | -2.65$e$-07 | 3.13$e$-07 | 4.89$e$-08 | -2.16$e$-07 |
| $\frac{dE}{dx_i}_R$ | 1.39$e$-17 | 2.78$e$-17 | 0.0 | -1.39$e$-17 | -2.78$e$-17 | -2.78$e$-17 | -2.22$e$-16 | -3.05$e$-16 | -2.08$e$-16 |
| $\frac{dE}{dy_i}_R$ | 1.39$e$-17 | 2.78$e$-17 | -2.22$e$-17 | 1.25$e$-16 | -1.39$e$-17 | -2.08$e$-16 | 1.39$e$-17 | -5.55$e$-17 | -1.52$e$-16 |

Some of the summation terms in the second term on the right hand side of Eqn. (7) are not taken into account due to the clipping of the Voronoi cells on the boundary. However, with the reconstructed Voronoi cells on the boundary, we can use Eqn. (7) to evaluate the gradient for each internal vertex. We do not need to apply special formulas as the ones presented in [21] for vertices that are on bisectors. The gradients at the nodes that are obtained in this case are also presented in Table 1 and show that they are machine precision across the computational domain. This small test, next to the standard comparison of the analytic energy gradient implementation against finite differences, served as a test case to verify the energy gradient implementation.

### D. Meshing process

We adopt a hierarchical meshing strategy. In this strategy, the edges (boundaries) of each two-dimensional surface are first discretized to the minimization of the energy functional that was described in the previous section. For this operation we use an adaptive quadrature that evaluates the energy functional to machine precision along with a quasi-Newton method, in order to guarantee that boundaries with identical dimension and metric field will generate identical discretization. The boundaries are then held fixed and the interior (surface) is generated to minimize the $L_p$-CVT energy. This same procedure can then be applied to generate a volume mesh from the bounding surfaces, and so on. This will give us a mesh that consists of simplices but the triangles are generally right-angled due to the higher $p$-norm that is applied. In order to end up with a quad-dominant mesh, we recombine the triangles using the Blossom-Quad algorithm [28]. To summarise:

1) Specify the symmetric Riemannian metric tensor.
2) Discretize the edges of the domain to satisfy Eqn. (3).
3) Introduce internal nodes and reconstruct the boundary Voronoi cells.
4) Minimize the $L_p$-CVT energy using a quasi-Newton method.
5) Repeat steps 2 and 3 until an equi-distribution of points in metric space is obtained.
6) Apply Blossom-Quad [28] to combine the simplices.

## III. Results

The results presented in this section serve as a proof-of-concept for the proposed method. For now, analytic metric tensor fields are defined in a simple domain like a square. However, this method can be adopted in its current form in the framework of a multi-block meshing algorithm.

### A. Recovering uniform lattice

As a first test case, we consider a simple rectangular domain $[0, 4]^2$ in two dimensions. The aim here is to retrieve a uniform lattice from a random initial point distribution using a metric tensor that is equal to the identity matrix. Note that for example $p = 2$ does not provide a unique minimum energy state. For the numerical results that are presented in this section, we use $p = 6$ as a norm to measure the length under the metric. We start by meshing the boundary edges using the underlying metric. We measure the length of each boundary edge, determine the number of points that are required to split this edge up into unit length sub-edges and using a Newton-method approach, we find the locations of the additional boundary vertices. We expect to obtain a uniform lattice when we would introduce nine internal nodes. We use a uniform random distribution function with bounds that lie within the computational domain $[0, 4]^2$ to get an initial guess of the locations of these nine internal mesh nodes.

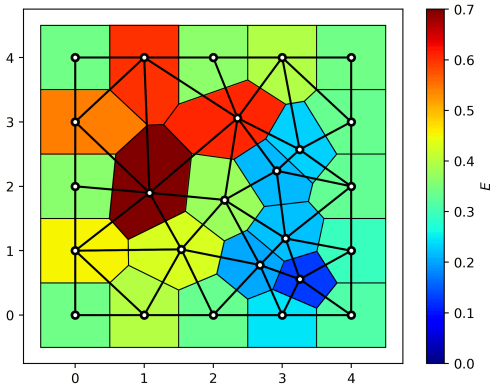
The initial configuration is shown in Fig. 5a. The nine internal points are represented by the black solid dots and their $x$- and $y$-coordinates are the variables for which we are seeking an optimal solution such that the overall energy functional given in Eq. (3) is minimized. The goal is to recover a uniform lattice for which all edges within the computational domain have unit length and the energy is equally distributed over the computational domain. Fig. 5d shows the optimal solution for the coordinates of the internal nodes such that the energy is minimal. Fig. 5b and Fig. 5c show two intermediate states that are outputted during the optimization procedure.

### B. Quadratic metric

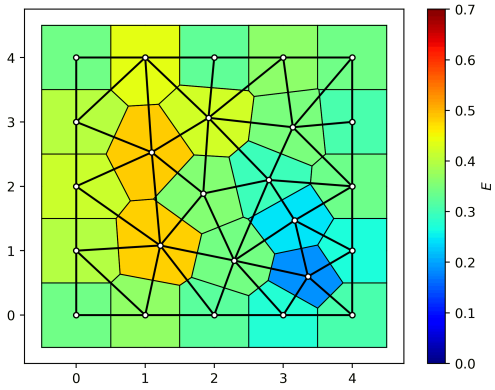
As a second test case, we apply a metric tensor that results in a quadratic progression in $y$-direction. We define a metric tensor that is based on the Hessian derived from $f(x, y) = y^2$. Hence, the metric tensor becomes:

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 + 4y^2 \end{pmatrix} \tag{12}$$

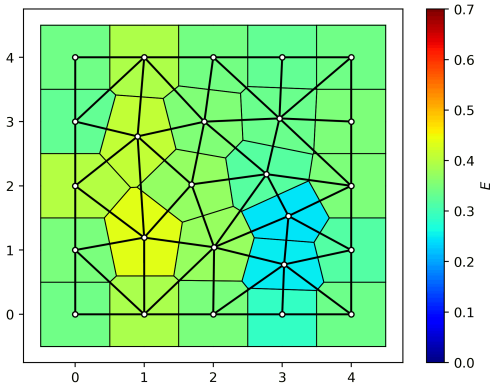
where the computational domain is rectangular with dimensions $\Omega = [0, 5] \times [0, 1.3]$. We follow the same procedure as described for the uniform lattice case to obtain a $L_p$-CVT. We use $p = 6$ and we introduce 16 internal nodes by using a random distribution function for each vertex coordinate. The metric essentially also prescribes the density of points locally. A random initialization using a uniform distribution function will result in a reduced quality of the initial guess once the metric starts to vary throughout the domain.



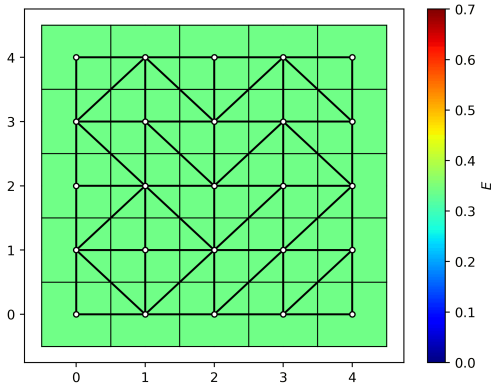

**(a) Initial random point distribution.**




**(b) Point distribution after 2 iterations.**



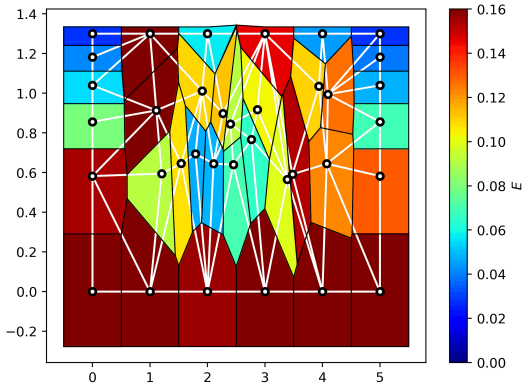

**(c) Point distribution after 4 iterations.**



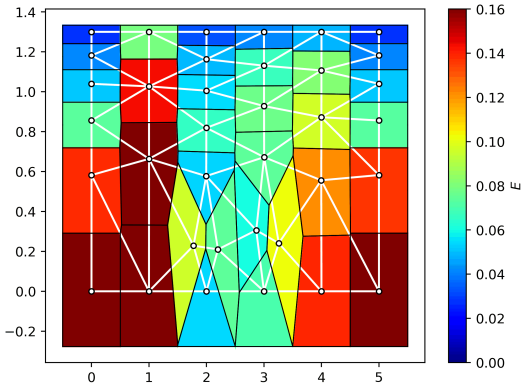

**(d) Final point distribution after 12 iterations.**

**Fig. 5 Recovering a uniform lattice from a random point initialization. The colors indicate the energy of each individual Voronoi cell.**

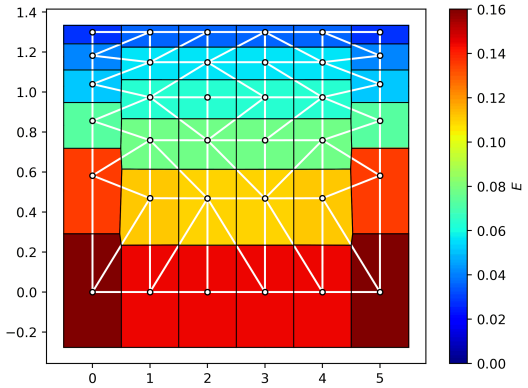
The initial distribution is shown in Fig. 6a. For this particular initial guess, the BFGS stalls after 51 quasi-Newton



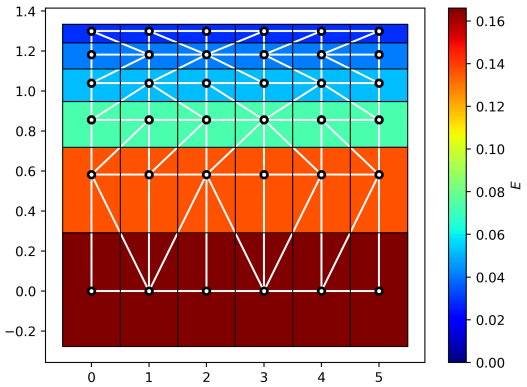

**(a) Initial distribution.** ($n = 16, E = 3.458$)**.**




**(b) The point distribution after 51 iterations.** ($n = 16, E = 2.982$)**.**



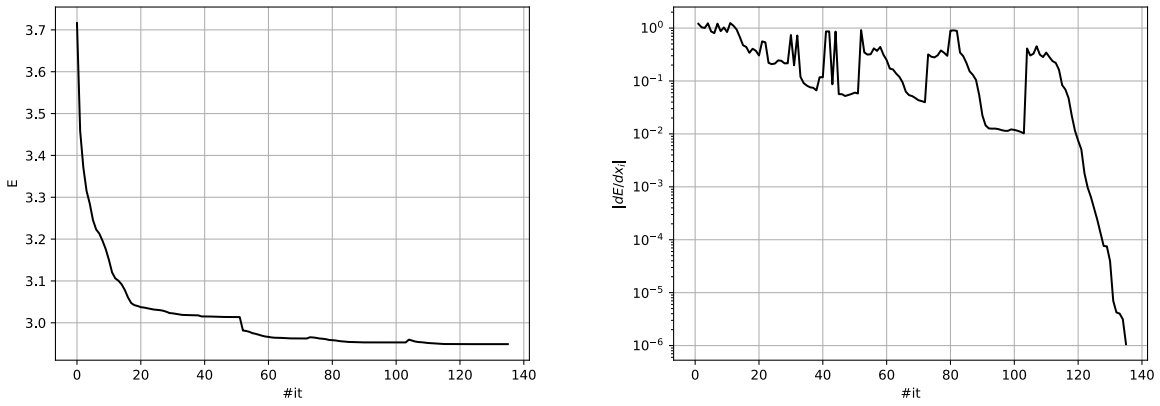

**(c) The optimal point distribution after point swaps.** ($n = 16, E = 2.944$)**.**




**(d) Points aligned with the boundary nodes resulting in** $E = 2.976$ **which is not the minimal solution.**

**Fig. 6** $L_p$**-CVT for a quadratic metric tensor defined in Eqn. (12). The colors indicate the energy in each Voronoi cell.**

iterations and the point distribution shown in Fig. 6b is returned. Some of the points are locked and the algorithm gets stuck in a local minimum. To avoid this problem, we used a point swap algorithm which is based on the length of the edges under the metric. Finding an appropriate node removal/insertion strategy is currently a topic of interest during the further development of this algorithm. In this case, the shortest and longest edge under the metric are determined. We aim to remove the nodes that form the shortest edge. This is done by computing the coefficient of variation for those two vertices. This coefficient of variation is based on the length under the metric of all the connected edges to that vertex. The vertex with the highest coefficient of variation is then removed from the mesh. For the edge that has the maximum length under the metric, we introduce a new point at the halfway point of that edge in Euclidean space. This process is




**(a) Convergence of the energy value.** **(b) Convergence of the norm of the gradient.**

**Fig. 7 Convergence history for $L_p$-CVT for a quadratic metric tensor defined in Eqn. (12)**

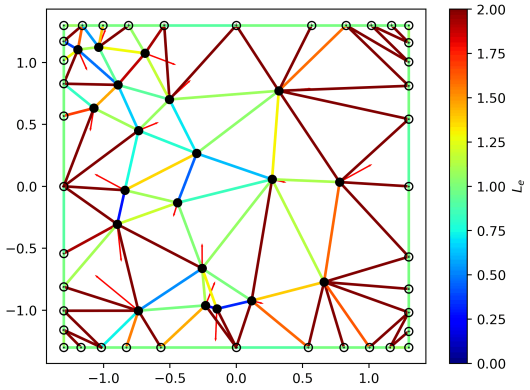
repeated until the optimal solution is found which in this case is the point distribution shown in Fig. 6c. This result shows that the internal points are perfectly aligned with each other but not with the nodes on the boundary. The energy distribution and the total energy for the aligned case is computed ($E = 2.976$). Based on a comparison between Fig. 6c and Fig. 6d we concluded that the aligned point distribution is not the minimal energy state for this configuration.

To introduce anisotropy in both directions, we also consider the function $f(x, y) = x^2 - y^2$ from which we derive the metric tensor.

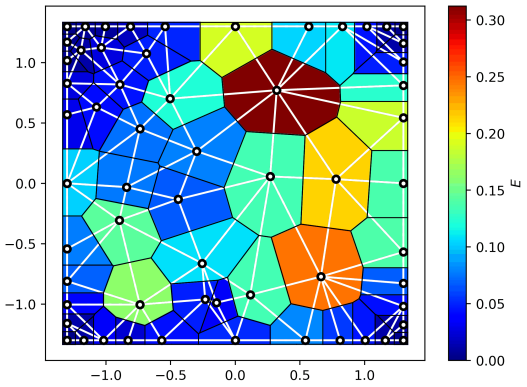
$$M = \begin{pmatrix} 1 + 4x^2 & -4xy \\ -4xy & 1 + 4y^2 \end{pmatrix} \tag{13}$$

We consider a square domain $[1.3, 1.3]^2$. As for the previous test case we again consider a norm $p = 6$. We start with a small number of randomly distributed points, in this case $n_{in} = 20$. Based on the number of initial points, we find an optimal distribution. Once the optimizer is finished due to stalling or due to the fact that the norm of the gradient is sufficiently decreased, we measure the lengths of the edges and introduces new points near the edge for which holds $L_e > \sqrt{2}$ where $L_e$ is the edge length.

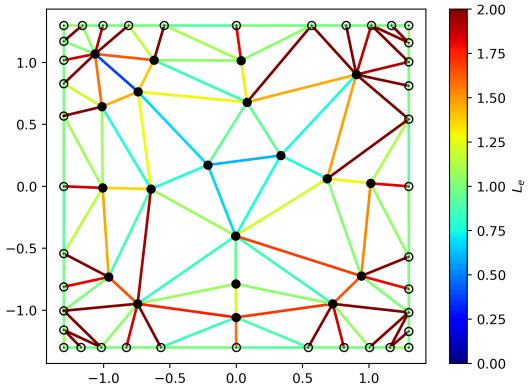
The initial guess and the corresponding initial edge lengths are plotted in Fig. 8a. The initial energy distribution throughout the domain is plotted in Fig. 8b. The $L_p$-CVT that is obtained based on this initial guess is shown in Fig. 8c and Fig. 8d. This is taken as a new initial guess and the next optimization step is carried out. The final result is shown demonstrated in Fig. 8e and Fig. 8f. The convergence of the optimization is plotted in Fig. 9. These plots indicate that the norm of the gradient drops significantly and is $O(10^{-6})$. After the introduction of the new points based on the edge lengths, a new minimum of the energy is found based on the number of internal points. Fig. 8e shows well aligned triangles that can be recombined and form quadrilateral elements.



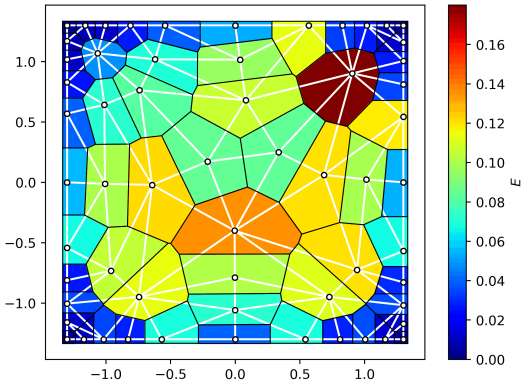

**(a) Initial random point distribution** ($n_{in} = 20, E = 3.95$)**.**



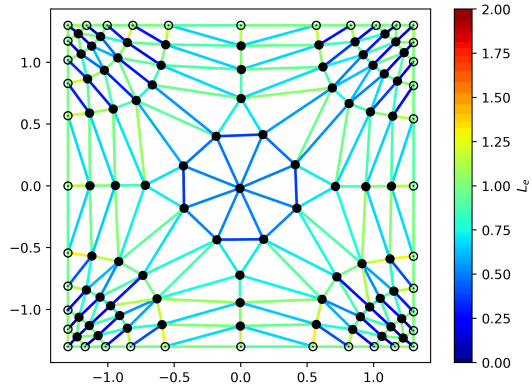

**(b) Initial energy distribution** ($n_{in} = 20, E = 3.95$)**.**



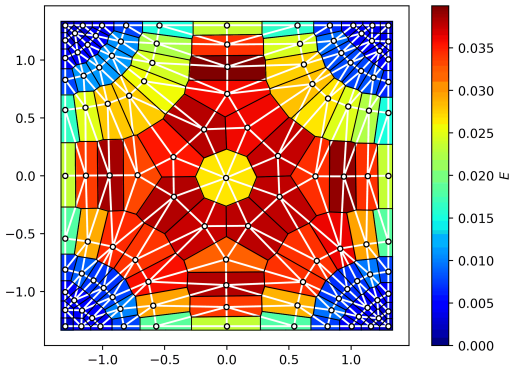

**(c) Point distribution after** 45 **iterations.** ($n = 61, E = 2.07$)




**(d) Energy distribution after** 45**.** ($n_{in} = 61, E = 2.07$)

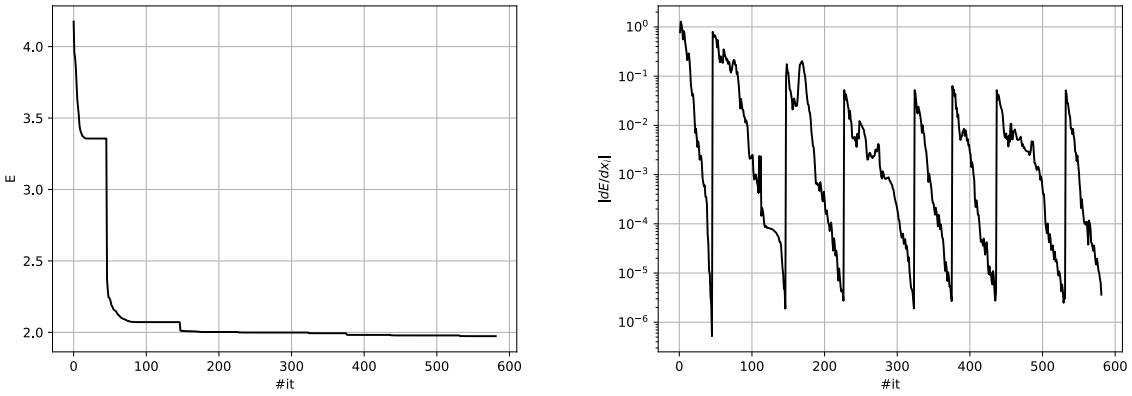



**(e) Final edge lengths.** ($n = 69, E = 2.00$)




**(f) Final energy distribution** ($n_{in} = 69, E = 2.00$)**.**

**Fig. 8** $L_p$**-CVT for a quadratic metric tensor defined in Eqn. (13). The colors on the left indicate the length of the edge under metric and the colors on the right indicate the local energy value.**




**(a) Convergence of the energy value.** **(b) Convergence of the norm of the gradient.**

**Fig. 9 Convergence history for the metric given in Eqn. (13)**

### C. *S*-shaped discontinuity

One of the intentions of the current algorithm is to improve the approximation of anisotropic features in the flow. Therefore we would like to replicate the metric tensor field that is associated to a shock wave. Ideally, the elements near the shock wave should be quadrilateral and align parallel to the shock to generally improve the numerical stability of the simulation. The test case presented here aims to test the performance of the proposed algorithm when dealing with shock waves. To mimic a discontinuity we use the following function

$$f(x, y) = \frac{3}{4}\tanh(2(\sin(5y) - 3x)) \tag{14}$$

The Hessian of Eqn. (14) is used as an underlying metric that will determine the local orientation and size of the elements in the mesh. Note that the sinusoid in the hyperbolic tangent introduces a wavy *S*-shaped feature and this metric therefore resembles an *S*-shaped discontinuity. The sinusoid introduces anisotropy in both directions. The computational domain in this case is $\Omega = [-1, 1]^2$ and the function given in Eqn. (14) is plotted in Fig. 10. We start by measuring the length of the boundary edges subject to the underlying metric and we determine the number of boundary nodes that are required to mesh each boundary edge such that each sub-edge has unit length. The location of these boundary vertices are then found using the Newton method. Once, the boundary edges are meshed in a metric conforming way, we randomly place 150 internal vertices and use that as an initial guess for our optimization procedure. The initial state with its Delaunay triangulation is plotted in Fig. 11a and the corresponding Voronoi diagram is given in Fig. 11b. The minimal energy state that is obtained after running the proposed $L_p$-CVT algorithm for the given initial point distribution and the given underlying metric is presented in Fig 11c and 11d. The convergence history is shown in Fig. 12a-12b. These figures demonstrate a good alignment of the triangles with the prescribed metric, however, the

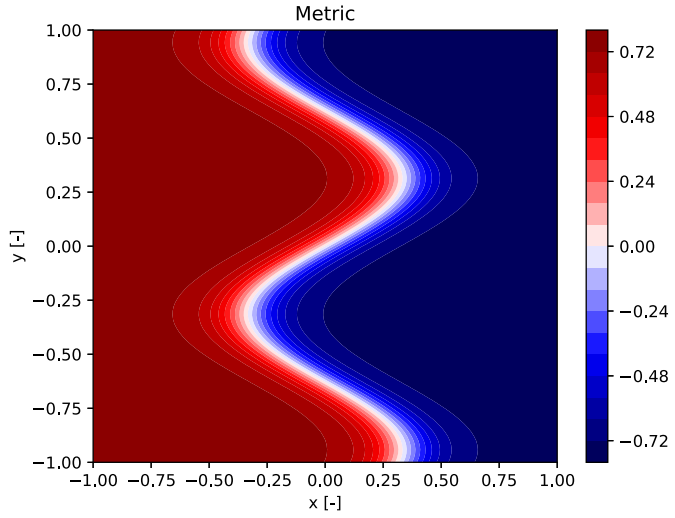
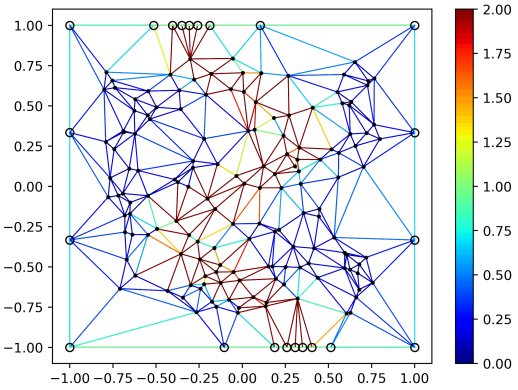


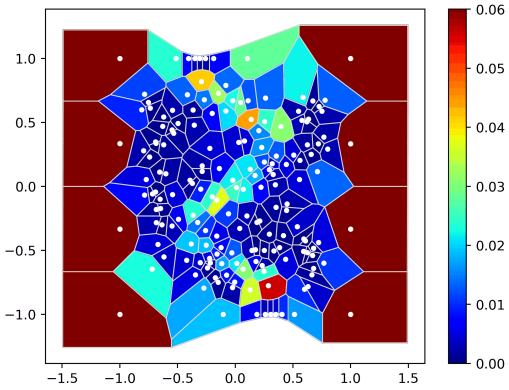

**Fig. 10 Metric resembling a weak $s$-shock.**

lengths of the edges are too long ($L_e > \sqrt{2}$) near the $S$-shape and too short far away from the $S$. This vanishes once we incorporate point insertion. This has been tested by introducing an extra vertex at the middle point (in Euclidean space) of each edge that exceeds twice the unit length under the metric. The edges are well aligned with the metric after the first optimization procedure (see. Fig.11c). After this, we introduce extra points based on the directionality of the edge and we run again the $L_p$-CVT algorithm to find the new optimal point distribution. The Delaunay triangulation and the corresponding Voronoi diagram that are obtained after the insertion of extra points are shown in Fig. 11e and Fig. 11f.

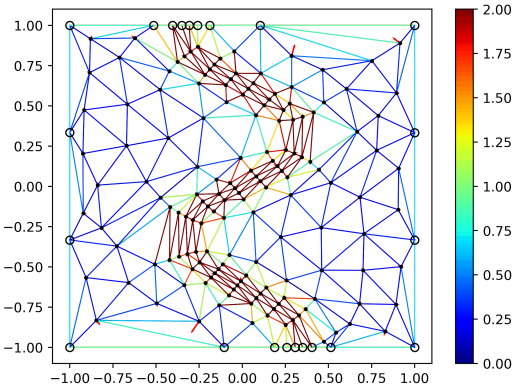
From Fig. 12 can be seen that the optimization procedure gets stuck after the first optimization iteration and also after the point insertion. This means that we need to add more points to obtain a fully converged solution like to one shown in Fig. 8e for example. To enable this, we need to work on a parallel implementation. The optimization problem becomes more costly since the number of input parameters is increasing rapidly. The current results are obtained using a serial implementation in python. To demonstrate the full algorithm, we have taken the meshes that were shown in Fig. 11c and Fig. 11e even though this result is not fully converged yet, we ran the Blossom-Quad algorithm to recombine the simplices and create a quad-dominant meshes. The results after the first optimization iteration and the second optimization iteration (after extra points are inserted) are shown in Fig. 13a and Fig. 13b respectively. It can be seen that the number of triangles within the $S$ are decreasing after extra points are inserted and that the quadrilateral elements are getting more aligned with each other. It is therefore expected that once we are increasing the number of points further, we improve the alignment and reduce the number of triangles within the $S$.



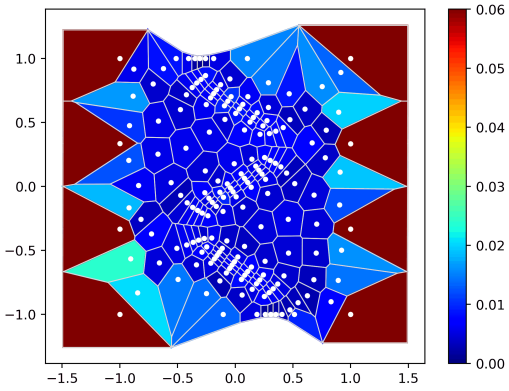
(a) Initial Delaunay triangulation of the 150 randomly distributed internal nodes.



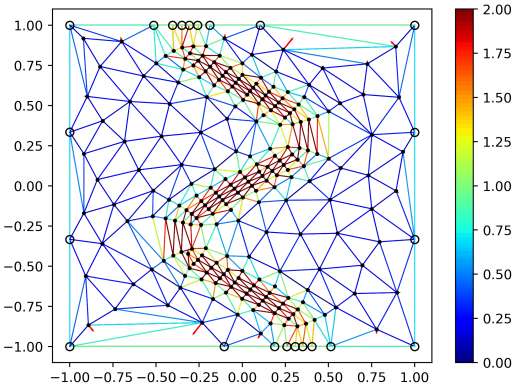
(b) Initial Voronoi diagram of the 150 randomly distributed internal nodes.



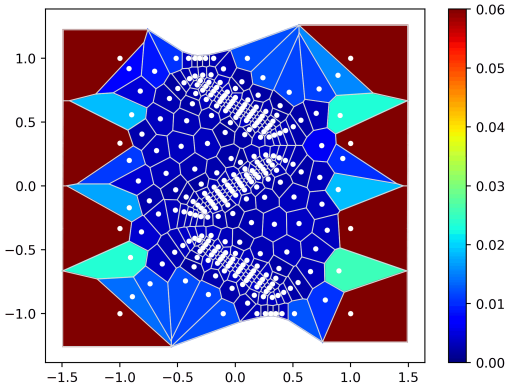
(c) Delaunay triangulation that corresponds to $L_p$-CVT after the first optimization cycle.



(d) Voronoi diagram that corresponds to $L_p$-CVT after the first optimization cycle.

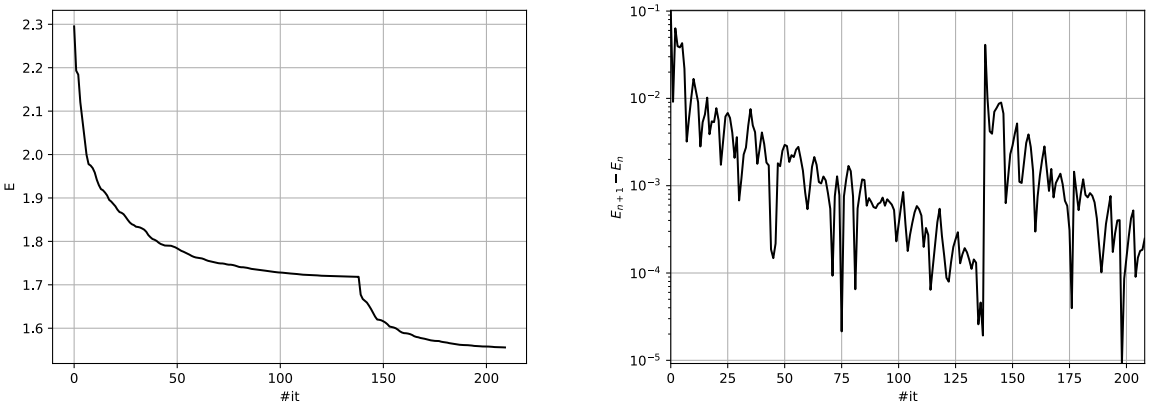


(e) Delaunay triangulation that corresponds to $L_p$-CVT after the extra points insertion.



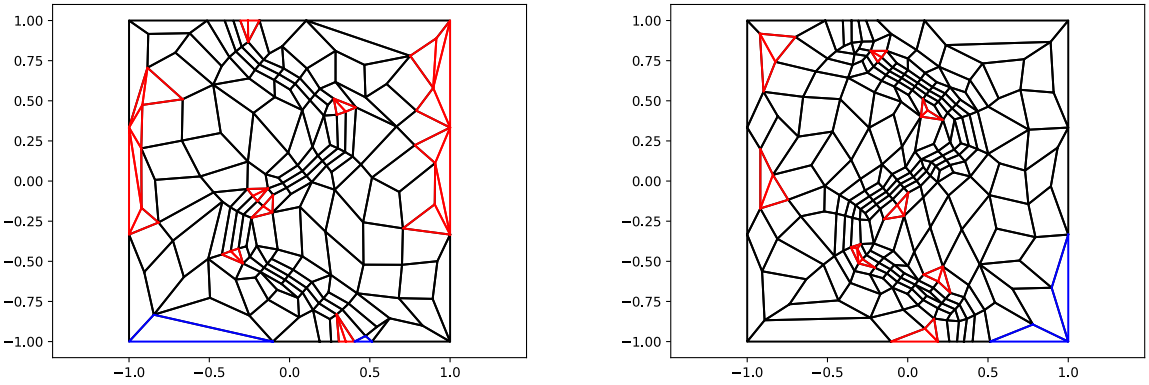
(f) Voronoi diagram that corresponds to $L_p$-CVT after the extra points insertion.

**Fig. 11 Demonstration of the proposed metric-aligned $L_p$-CVT algorithm for a $S$-shaped discontinuity. The colors on the left indicate the length of the edge under metric and the colors on the right indicate the local energy value.**




**(a) Convergence of the energy value.** **(b) Difference in energy of the between iterations**

**Fig. 12 Convergence history for the $S$-shaped metric up to the current status.**



**(a) Quad-dominant mesh obtained after the first optimization iteration.** **(b) Quad-dominant mesh obtained after the second optimization iteration.**

**Fig. 13 The quad-dominant meshes that are obtained after the Quad-Blossom algorithm is applied. The red triangles are a consequence of a bad quadrilateral element that was split again into simplices. The blue triangles where not considered to be recombined with one of their neighboring triangles.**

## IV. Conclusion

The aim of this paper was to demonstrate a prototype of a metric-aligned $L_p$-Centroidal Voronoi Tesselation meshing algorithm. This method relies on a underlying metric, that can be derived from the flow solution for example, and a definition of the geometry. In this way, this method can be used to both generate and adapt a mesh. It is shown that the algorithm produces right-angled elements that follow this underlying metric carefully based on the provided metric and a domain definition. Furthermore, we have demonstrated that clipping the Voronoi diagram in order to compute the minimization of the energy functional on a bounded domain is not necessary and that alternatively, the Voronoi cells on the boundary can be reconstructed using the metric field. It was shown that this results in a correct evaluation of both the energy functional and its corresponding gradient. In contrast to previous works, we do not need to use a general formula and we do not need special formulas to compute the gradient for vertices that are near a boundary edge.

As a first test case, we were able to recover a uniform lattice from a random point distribution (see the example presented in section III A). The examples presented in sections III B and III C show that the algorithm produces anisotropic and aligned elements that have right angled corners which can be used as a proper conditioning to apply an element recombination algorithm like Blossom-Quad. Particularly the results presented in section III B show good convergence. The preliminary results presented in III C) show good alignment with the metric and demonstrate quadrilateral elements that are parallel with the shock definition. However it also shows that we need to improve the efficiency since this method becomes very costly when the number of vertices increases. The current implementation of the algorithm is written in python and is not yet parallelized. The results presented in this paper served as a proof-of-concept of the proposed mesh generation algorithm. The next steps forward are therefore the parallelization of this method. Currently, we are exploring different methods to improve the initialization of the method and we are also developing different node insertion/removal strategies. One of the drawbacks of the Voronoi reconstruction strategy is the need to extrapolate the metric field which we are currently investigating as well. Furthermore, we are currently developing the algorithms to interpolate higher-order metrics that are provided by the monolithic high-order spectral element solver that is currently being developed at NASA Ames. Future work will therefore entail the embedding of the presented meshing algorithm in this code.

## Acknowledgments

This research is supported by an appointment to the NASA Postdoctoral Program at the NASA Ames Research Center, administered by Universities Space Research Association under contract with NASA.

## References

[1] Slotnick, J., Khodadoust, A., Alonso, A., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," Tech. Rep. CR-2014-218178, NASA, 2014.

[2] Ceze, M., Diosady, L., and Murman, S., "Development of a high-order space-time matrix-free adjoint solver," *54st AIAA Aerospace Sciences Meeting, AIAA Paper 2016-0833*, San Diego, California, US, 2016.

[3] Diosady, L., and Murman, S., "Design of a Variational Multiscale Method for Turbulent Compressible Flows," *21st AIAA Computational Fluid Dynamics Conference, AIAA Paper 2013-2870*, San Diego, California, US, 2013.

[4] Diosady, L., and Murman, S., "Higher-Order Methods for Compressible Turbulent Flows Using Entropy Variables,," *21st AIAA Computational Fluid Dynamics Conference, AIAA Paper 2015-0294*, San Diego, California, US, 2015.

[5] Peskin, C. S., "Numerical analysis of blood flow in the heart," *Journal of Computational Physics*, Vol. 25, No. 3, 1977, pp. 220 – 252. doi:http://dx.doi.org/10.1016/0021-9991(77)90100-0, URL `http://www.sciencedirect.com/science/article/pii/0021999177901000`.

[6] Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," *AIAA Journal*, Vol. 36, No. 6, 1998, pp. 952–960.

[7] Chen, H., "Volumetric formulation of the lattice Boltzmann method for fluid dynamics: Basic concept," *Phys. Rev. E*, Vol. 58, 1998, pp. 3955–3963.

[8] Ruiz-Gironés, E., Roca, X., and Sarrate, J., "The receding front method applied to hexahedral mesh generation of exterior domains," *Engineering with Computers*, Vol. 28, No. 4, 2012, pp. 391–408.

[9] Staten, M., Kerr, R., Owen, S., Blacker, T., Stupazzini, M., and Shimada, K., "Unconstrained plastering-hexahedral mesh generation via advancing-front geometry decomposition," *Numerical Methods in Engineering*, Vol. 81, No. 2, 2009, pp. 135–171.

[10] Maréchal, L., "Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features," *Proceedings of the 18th International Meshing Roundtable*, edited by B. Clark, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 65–84.

[11] Roca, X., and Sarrate, J., "An automatic and general least-squares projection procedure for sweep meshing," *Engineering with Computers*, Vol. 26, No. 4, 2010, pp. 391–406.

[12] Alauzet, F., and Loseille, A., "A Decade of Progress on Anisotropic Mesh Adaptation for Computational Fluid Dynamics," *Comput. Aided Des.*, Vol. 72, No. C, 2016, pp. 13–39.

[13] Loseille, A., Dervieux, A., and Alauzet, F., "A 3D goal-oriented anisotropic mesh adaptation applied to inviscid flows in aeronautics," *48th AIAA aerospace sciences meeting including the new horizons forum and aerospace Exposition, AIAA Paper 2010-1067*, Orlando, Florida, US, 2010.

[14] Loseille, A., "Metric-orthogonal Anisotropic Mesh Generation," *Procedia Engineering*, Vol. 82, No. Supplement C, 2014, pp. 403 – 415. 23rd International Meshing Roundtable (IMR23).

[15] Marcum, D., and Alauzet, F., "Aligned Metric-based Anisotropic Solution Adaptive Mesh Generation," *Procedia Engineering*, Vol. 82, 2014, pp. 428–444.

[16] Barral, N., Alauzet, F., and Loseille, A., "Metric-Based Anisotropic Mesh Adaptation for Three-Dimensional Time-Dependent Problems Involving Moving Geometries," *53rd AIAA Aerospace Sciences Meeting, AIAA Paper 2015-2039*, Kissimi, Florida, US, 2015.

[17] Hecht, F., and Mohammadi, B., "Mesh adaption by metric control for multi-scale phenomena and turbulence," *AIAA Paper 97-0859*, 1997.

[18] Loseille, A., and Alauzet, F., "Continuous mesh framework Part 1: Well-posed continuous interpolation error," *SIAM Journal on Numerical Analysis*, Vol. 49, No. 1, 2011, pp. 38–60.

[19] Loseille, A., and Alauzet, F., "Continuous mesh framework Part 2: Validations and applications," *SIAM Journal on Numerical Analysis*, Vol. 49, No. 1, 2011, pp. 61–86.

[20] Lévy, B., and Yang, L., "$L_p$ Centroidal Voronoi Tessellation and Its Applications," *ACM Transactions on Graphics*, Vol. 9, No. 4, 2010, pp. 1–11.

[21] Baudouin, T., Remacle, J.-F., Marchandise, E., Lambrechts, J., and Henrotte, F., "Lloyd's energy minimization in the Lp norm for quadrilateral surface mesh generation," *Engineering with Computers*, Vol. 30, No. 1, 2014, pp. 97–110.

[22] Caplan, P., Haimes, R., Darmofal, D., and Galbraith, M., "Anisotropic geometry-conforming d-simplicial meshing via isometric embeddings," *Procedia Engineering*, Vol. 203, No. Supplement C, 2017, pp. 141 – 153. 26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain.

[23] Du, Q., Faber, V., and Gunzburger, M., "Centroidal voronoi tessellation: Applications and algorithms," *SIAM Review*, Vol. 41, 1999, pp. 637–676.

[24] Lloyd, S. P., "Least squares quantization in PCM," *IEEE Trans. Information Theory*, Vol. 28, 1982, pp. 129–136.

[25] Dunavant, D., "High degree efficient symmetrical Gaussian quadrature rules for the triangle," *International Journal for Numerical Methods in Engineering*, Vol. 21, 1985, pp. 1129–1148.

[26] Mokris, D., "Mesh generation and optimization," , 2010. Mathematical Institute of Charles University.

[27] Du, Q., Emelianenko, M., and Ju, L., "Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations," *SIAM Journal on Numerical Analysis*, Vol. 44, No. 1, 2006, pp. 102–119.

[28] Remacle, J.-F., Lambrechts, J., Seny, B., Marchandise, E., Johnen, A., and Geuzainet, C., "A non-uniform quadrilateral mesh generator using a minimum-cost perfect matching algorithm," *International Journal for Numerical Methods in Engineering*, Vol. 89, 2012, pp. 1102–1119.