SIAM J. SCI. COMPUT.
Vol. 39, No. 6, pp. A2438–A2471




# PROVABLY ROBUST DIRECTIONAL VERTEX RELAXATION FOR GEOMETRIC MESH OPTIMIZATION*

RAMSHARAN RANGARAJAN† AND ADRIAN J. LEW‡

**Abstract.** We introduce an iterative algorithm called *directional vertex relaxation* that seeks to optimally perturb vertices in a mesh along prescribed directions without altering element connectivities. Each vertex update in the algorithm requires the solution of a max-min optimization problem that is nonlinear, nonconvex, and nonsmooth. With relatively benign restrictions on element quality metrics and on the input mesh, we show that these optimization problems are well posed and that their resolution reduces to computing roots of scalar equations regardless of the type of the mesh or the spatial dimension. We adopt a novel notion of mesh quality and prove that the qualities of mesh iterates computed by the algorithm are nondecreasing. The algorithm is straightforward to incorporate within existing mesh smoothing codes. We include numerical experiments which are representative of applications in which directional vertex relaxation will be useful and which reveal the improvement in triangle and tetrahedral mesh qualities possible with it.

**Key words.** mesh improvement, moving boundary, mesh motion, max-min optimization, nonsmooth optimization

**AMS subject classifications.** 65M50, 65M60, 65D18, 65K10

**DOI.** 10.1137/16M1089101

## 1. Introduction.

We introduce, analyze, and examine the performance of *directional vertex relaxation*, an algorithm designed to improve mesh qualities by perturbing a specified set of vertices along prescribed directions. Henceforth referred to as dvr, directional vertex relaxation is an iterative algorithm in which vertices of a mesh are perturbed sequentially. The defining feature of dvr is a vertexwise max-min problem furnishing vertex updates in the algorithm. Each vertex update, while restricted to a given direction, serves to maximize the poorest quality among the elements incident at it.

Figure 1 illustrates the main idea behind dvr for relaxing a vertex in a triangle mesh. For simplicity, we consider relaxing just the vertex $v$ and choose the relaxation direction to be vertical and horizontal during odd and even numbered iterations, respectively. Assuming an element quality indicator that assigns higher qualities to more regular-shaped triangles (see section 3.1 for a detailed definition), we find that the poorest quality among the triangles $K_1, \ldots, K_4$ incident at $v$ is that of $K_4$. During the first iteration, dvr computes the coordinate $\lambda^{\text{opt}}$ such that relocating $v$ to $v' = v + \lambda^{\text{opt}}\,\mathbf{e}_y$ maximizes the minimum among the qualities of the resulting four triangles $K_1', \ldots, K_4'$. The shape of triangle $K_4$ is visibly improved as a result. During the second iteration, dvr computes $\lambda^{\text{opt}}$ such that relocating $v'$ to $v'' = v' + \lambda^{\text{opt}}\,\mathbf{e}_x$ maximizes the poorest quality among the resulting triangles $K_1'', \ldots, K_4''$. This procedure can be repeated indefinitely; in the example shown, however, no improvement

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 11, 2016; accepted for publication (in revised form) August 22, 2017; published electronically November 7, 2017.

http://www.siam.org/journals/sisc/39-6/M108910.html

**Funding:** This work was supported by NSF-CMMI-1301396 and Army Research grant W911NF-07-2-0027.

†Corresponding author. Department of Mechanical Engineering, Indian Institute of Science Bangalore, Bangalore, Karnataka 560012, India (rram@iisc.ac.in).

‡Department of Mechanical Engineering, Institute of Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305 (lewa@stanford.edu).

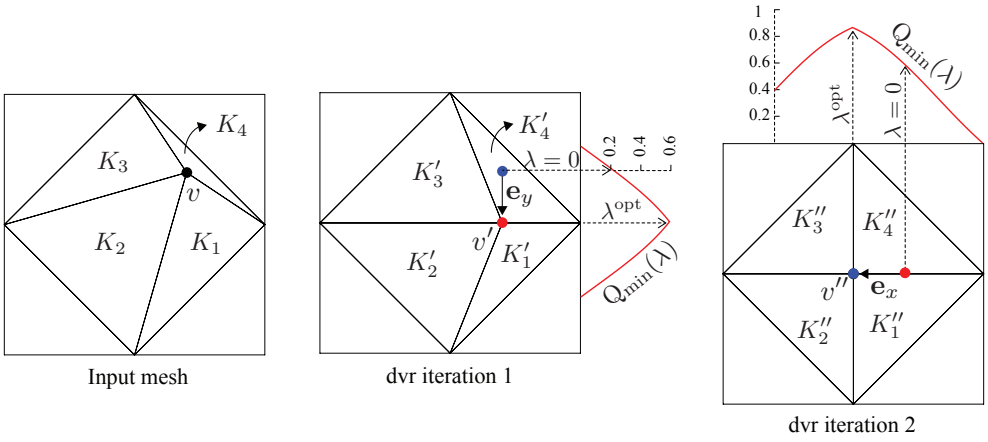



FIG. 1. *An illustration of the* dvr *algorithm for improving triangle qualities around a vertex* $v$. *During the first iteration,* $v$ *is relocated to* $v'$ *along the vertical direction by maximizing the function* $\lambda \mapsto \mathrm{Q}_{\min}(\lambda)$, *which computes the poorest quality among the elements* $K_1, \ldots, K_4$ *as the common vertex* $v$ *is perturbed to* $v + \lambda \mathbf{e}_y$. *Similarly,* $v'$ *is perturbed to* $v''$ *during the second iteration, this time along the horizontal direction. In the figure, triangle qualities are defined using the mean ratio metric.*

is possible beyond two iterations.

The dvr algorithm is now straightforward to describe—given a mesh, an ordered subset of its vertices that can be relaxed, and the directions along which these vertices can be relaxed, each dvr iteration consists in repeating the procedure illustrated in Figure 1, one vertex at a time. A detailed description of the algorithm and the max-min problem defining vertex updates is given in section 2.

Among our main goals here is to rigorously answer a few compelling questions concerning the dvr algorithm. First, how do we know that vertex updates in dvr are well defined? That is, under what conditions are the max-min problems defining optimal perturbations well posed? Second, how can we compute these optimal perturbations? A foolproof algorithm is essential because each vertex update requires the resolution of a max-min problem. Third, in what sense, if at all, do vertex updates computed by dvr improve the mesh quality? In particular, what can we say about the quality of the poorest element in the mesh? The last point is particularly significant in finite element computations, since the poorest element quality has a direct bearing on the conditioning of finite element matrices and for the choice of increments in time-stepping schemes. To this end, we examine sufficient conditions for the max-min problems furnishing vertex updates to be well posed, identify simple yet robust algorithms for resolving them, and prove monotonic improvement in mesh qualities with each vertex update. The results on mesh quality improvement with each vertex update in dvr (Theorems 4.2 and 4.10) have a broader significance for algorithms relying on relaxing vertices one by one—it rigorously explains when and why local improvements in element qualities resulting from sequential vertex updates improve the mesh quality overall. Finally, through an extensive set of numerical experiments, we demonstrate the enhancement in triangle and tetrahedral mesh qualities possible with dvr. While the choice of relaxation directions has a definitive influence on the quantum of mesh improvement possible with dvr, our experiments reveal that even arbitrary ones (e.g., randomly generated or along the cardinal axes) yield meshes with

significantly better qualities.

The guarantees afforded by dvr together with the lack of any heuristics render it a useful tool for handling mesh deformations required for simulating moving boundary problems. Such problems, which are ubiquitous in engineering, pose the challenge of adapting meshes to conform to evolving fronts such as moving boundaries, interfaces, cracks, and shocks [12]. Mesh updates in the vicinity of the moving front are required after each time/load step when simulating a crack propagating through a solid, for instance [46], or after every domain update during the iterative resolution of a shape optimization problem [2, 56]. In such a context where interactive mesh improvement is simply impossible, robustness of the algorithm adopted for preserving good mesh quality is nonnegotiable for the feasibility and success of numerical simulations. This is precisely what dvr accomplishes. Numerical experiments such as the ones presented in section 5 reveal that dvr is a compelling choice when reasonably small and localized vertex perturbations suffice for mesh improvement. This is often the case when adapting meshes in the vicinity of a moving front, where perturbations comparable to the local mesh size serve to preserve good element qualities while retaining a conforming mesh.

We emphasize unequivocally that dvr is neither intended nor can it serve as a comprehensive solution for mesh improvement. A cursory survey of the literature will immediately reveal that a combination of mesh relaxation, topological operations (face swapping, edge removal, etc.), and vertex insertion/removal operations can improve a mesh by far more than if any one of these operations were performed in isolation [30, 22, 23, 32, 4]. By design, therefore, dvr is predestined to neither be the best nor the most sophisticated mesh improvement algorithm available. It is, however, erroneous to presume that mesh improvement algorithms have attained maturity. To quote [27], "mesh optimization procedures have a lot to do with *black magic*—even if the ingredients required to construct a mesh optimization procedure are well known (essentially swapping and smoothing), there is no known *best recipe*, i.e., no known optimal way of combining those smoothing and swapping operators.'. In this context at least, and especially for the purpose of simulating moving boundary problems, the robustness guaranteed by dvr stands out.

Within the restricted context of geometric mesh quality optimization algorithms which preserve element connectivities and improve a mesh by perturbing its vertices, numerical experiments suggest that dvr is a competitive alternative to comparable methods in the literature based on relaxing vertices one at a time. Though not representative of the state of the art, Laplacian smoothing [40], with its well-recorded shortcomings, falls in this category. A variant of Laplacian smoothing using weighted averages rather than an arithmetic mean to relocate vertices can be found in [54]. Perhaps the closest counterpart to dvr is the algorithm proposed in [19] and subsequently expounded in [18, 20, 23], which differs from dvr essentially in not restricting the directions along which vertices are relaxed. Hence if the vertex updates proposed in [19] were to be computed accurately by resolving the multidimensional analogue of the max-min problem manifested in dvr, the resulting meshes would ostensibly have better qualities than those computed by dvr. Herein lies a key difficulty—resolving the resulting nonlinear, nonsmooth, and nonconvex multivariate max-min problem is nontrivial and inevitably requires adopting various ad hoc measures (e.g., linearizations, coupling with intermittent Laplacian smoothing). Varied approaches have been proposed to circumvent the complications arising from nonsmoothness, including derivative-free optimization algorithms that resort to sampling the objective at a predefined stencil of points [41]. In contrast, even though the vertex-update problem in dvr remains

nonlinear, nonsmooth, and nonconvex, it is rendered one-dimensional by intentionally restricting the search directions. Then we exploit plausible features of the quality metric (mainly convexity of certain level sets [3, 1, 45]) to accurately compute the optimizer without resorting to any heuristics.

A second class of geometric mesh improvement algorithms we mention are ones that permit relaxing multiple vertices simultaneously. Evidently, these algorithms can explore mesh configurations that ones such as dvr relying on relaxing one vertex at a time cannot and therefore may be able to compute meshes with better qualities. However, this expectation is confounded by the choices for functionals that are optimized in practice. The pervasive choice, namely $\ell^p$-norms of the set of element qualities [61, 21, 33], are preferred because of the smoothness of the resulting optimization problem. But such choices do not guarantee improvement in the minimum element quality and require additional measures to prevent element inversions. The challenge of resolving large multivariate optimization problems realized in these methods also remains an active area of research in the numerical optimization and meshing literature [31, 60, 58, 38].

The distinction between algorithms that relax vertices sequentially and simultaneously may not always be evident. An example that helps illustrate this point can be found in [50, 51], where the nonsmooth problem of maximizing the poorest element quality in a mesh is approximated by a sequence of smooth problems with a modified objective function involving the entire set of vertices that may be perturbed. Of course, the literature on numerical optimization is replete with approaches based on resolving a sequence of regularized problems to approximate the solution of a nonsmooth one; cf. [44]. A comparison of the performances of a vertex-by-vertex mesh improvement algorithm with one that permits relaxing multiple vertices simultaneously for a representative set of meshes can be found in [15]. The open-source library [8] provides a convenient way for conducting such tests, which are often helpful for inferring useful rules of thumb for special classes or application-specific meshes. It is, however, unclear whether any general conclusions can be deduced from them.

An alternative paradigm to optimization-based mesh improvement algorithms is a large assortment of physics-based mesh relaxation methods. The ideas proposed in this context range from ones invoking analogies with spring-mass systems [6, 17], truss networks [43], linear/nonlinear elasticity [4, 13, 49], and harmonic/biharmonic mappings [28], to name a few. Though found to work well "most of the time," these methods invariably fail to offer any guarantees for mesh improvement. For simulating moving boundary problems in particular, arbitrary Lagrangian-Eulerian [16, 35] and moving mesh methods [29, 9, 59, 7] achieve adaptivity by introducing intimate couplings between the underlying physical model and the mesh motion. Although auxiliary sets of partial differential equations to compute the mesh motion extend the system to be solved, adaptation spanning length scales much larger than the typical mesh size can be achieved. It is conceivable that a pairing between dvr and such mesh motion schemes can help realize nonlocal mesh adaptation for the purpose of tailoring meshes based on error estimators [5, 11] without sacrificing mesh quality. In such a scenario, the nodal velocities can serve as a natural choice for the relaxation directions in dvr.

In the remainder of the article, we have necessarily overlooked certain aspects of mesh relaxation that are important in practice. This includes incorporating constraints to relax vertices restrained to lie on boundaries (curves and surfaces in general); cf. [24, 25]. For the sake of definiteness, we consider meshes to be composed of triangles in two spatial dimensions and tetrahedra in three dimensions in our descrip-

tion of the dvr algorithm, its analysis, and in all the numerical experiments shown. We emphasize, however, that there is *no* inherent limitation of the algorithm or of its analysis to these specific mesh types—both apply verbatim to general mesh types (quads, hexes, etc). However, detailed numerical investigations are necessary to examine the quantum of mesh improvement possible with specific mesh types. Similar remarks also apply to the element quality metric adopted. The choice of quality metric is intimately related to the purpose of mesh relaxation, and can therefore be defined in a data-, size- or orientation-dependent manner, and to adapt meshes based on error estimates in numerical simulations [5, 10]. In sections 4.1 and 4.3, we identify conditions on quality metrics that suffice to ensure that the dvr algorithm is well defined and to guarantee monotonic improvement in mesh qualities. These assumptions are not particularly restrictive but have to be verified on a case-by-case basis nevertheless. The mean ratio metric [36, 37], which we adopt in all our examples, is just one example of a quality metric satisfying all these requirements. By choosing the mean ratio metric, the goal of mesh optimization in these examples is simply to render triangles as close to equilateral and tetrahedra as close to regular as possible. Minor adaptations of the metric, as described in [35, 39], for instance, enable targeting alternative element shapes as well. We mention that the crucial assumption concerning convexity of certain level sets is satisfied by many commonly used metrics [3, 1].

**2. Directional vertex relaxation.** To describe the dvr algorithm and the sequence of mesh iterates computed by it, we briefly introduce some notation related to multisets, triangulations, and orderings.

**2.1. Multisets.** It is convenient to consider lists of vertex coordinates and element qualities to be multisets, so that distinct vertices and elements may have identical coordinates and qualities, respectively. To this end, we recall the definition of multisets as unordered lists in which members are permitted to appear more than once. To explicitly distinguish sets from multisets, we denote the latter using braces in boldface $\{\!\!\{\}\!\!\}$ and their unions with $\uplus$. For example, $\{\!\!\{1,1,2,3,3,3,3,4\}\!\!\}$ is a multiset and $\{\!\!\{1,1,2,3,3,3,3,4\}\!\!\} \uplus \{\!\!\{4\}\!\!\} = \{\!\!\{1,1,2,3,3,3,3,4,4\}\!\!\}$, whereas the set $\{1,1,2,3,3,3,3,4\}$ equals $\{1,2,3,4\}$.

The function asc maps multisets in $\mathbb{R}^n$ to vectors in $\mathbb{R}^n$ with components arranged in ascending order, i.e.,

$$\text{asc}(\{\!\!\{\mathrm{u}_1 \le \mathrm{u}_2 \le \cdots \le \mathrm{u}_\mathrm{n}\}\!\!\}) \triangleq (\mathrm{u}_1, \mathrm{u}_2, \ldots, \mathrm{u}_\mathrm{n}),\ \mathrm{n} \in \mathbb{N}. \tag{1}$$

The dimension $n$ of the argument of asc will always be evident. For this reason, we omit $n$ from the notation used for asc. Note that $\text{asc}(\mathbf{u})$ is sometimes referred to as the *lexicographic* reordering of the components of a vector $\mathbf{u} \in \mathbb{R}^n$; cf. [32].

**2.2. An ordering relation $\geq$ in $\mathbb{R}^n$.** We will soon introduce a notion of vector-valued mesh qualities. To facilitate comparing meshes using their qualities, we introduce an ordering $\geq$ over $\mathbb{R}^n$. Let $u_i$ denote the $i$th component of $\mathbf{u} \in \mathbb{R}^n$, where $1 \le i \le n$. For a pair of vectors $\mathbf{u}, \mathbf{w} \in \mathbb{R}^n$, define the binary relation $>$ as

$$\mathbf{u} > \mathbf{w} \iff u_i > w_i, \quad \text{where } i = \arg \min_{1 \le j \le n} \{u_j \ne w_j\}. \tag{2}$$

For example, $(1,2,3) > (1,1.5,4)$ in $\mathbb{R}^3$ with index $i = 2$ in definition (2). With equality of vectors defined in a componentwise sense, it is easily checked that the binary relation $\geq$ defines a total ordering in $\mathbb{R}^n$. For $n = 1$, the relation $\geq$ is consistent with the usual ordering over scalars.

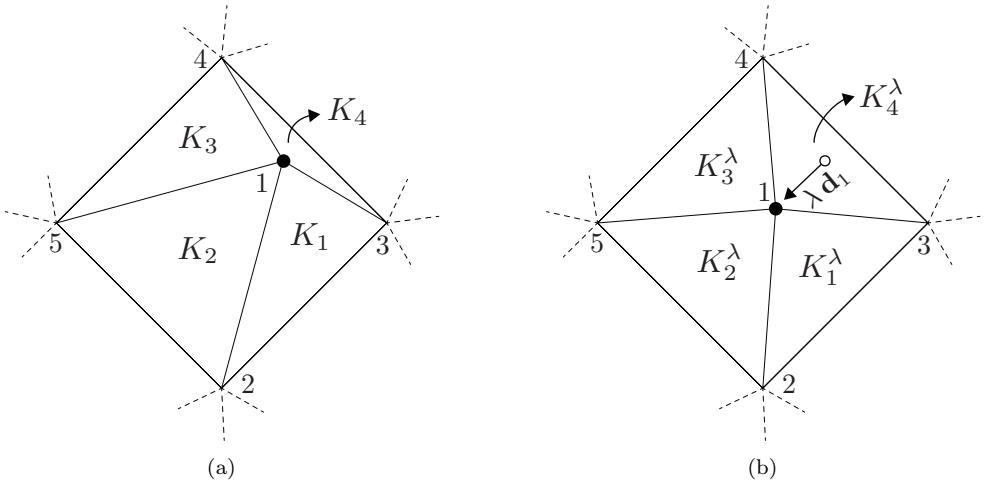



FIG. 2. *An example explaining the notation introduced in section* 2.3 *for a triangle mesh* $\mathcal{T}$. *Figure* 2a *shows vertices* $1,\dots,5$ *and triangles* $K_1,\dots,K_4$ *in the* 1*-ring of vertex* 1. *Hence* $\overset{\star}{\mathrm{I}}(1;C) = \{2,3,4,5\}$ *and* $\overset{\star}{\mathrm{e}}(1;\mathcal{T}) = \{K_1,K_2,K_3,K_4\}$. *Relocating vertex* 1 *to* $V_1 + \lambda\,\mathbf{d}_1$ *alters the triangles in its* 1*-ring as shown in Figure* 2b. *The resulting mesh is denoted by* $\mathcal{T}^{1,\lambda}$ *and the minimum among the qualities of the resulting triangles* $\{K_1^\lambda,\dots,K_4^\lambda\}$ *around vertex* 1 *by* $q_{1,1}(\lambda,\mathcal{T})$. *Notice that the perturbation shown in* (b) *may also alter the minimum quality among elements in the* 1*-rings of vertices* $2,\dots,5$. *These qualities are denoted by* $q_{1,2}(\lambda,\mathcal{T}),\dots,q_{1,5}(\lambda,\mathcal{T})$, *respectively.*

**2.3. Triangulations.** A triangulation $\mathcal{T}$ in $\mathbb{R}^d$ is a collection of $d$-simplices identified by (i) an index set I for its $N$ vertices, (ii) a corresponding multiset V for their coordinates identified with a set in $\left[\mathbb{R}^d\right]^N$, and (iii) a set of connectivities C consisting of $(d+1)$-tuples of indices in I. Hence $\mathcal{T}$ is defined as the 3-tuple $\mathcal{T} = (\mathrm{V},\mathrm{I},\mathrm{C})$. The case $d = 2$ corresponds to a mesh of planar triangles and the case $d = 3$ to a mesh of tetrahedra. The position of a vertex with index $i \in \mathrm{I}$ in $\mathbb{R}^d$ is denoted by $\mathrm{V}_i \in \mathrm{V}$ and a simplex $K$ in $\mathcal{T}$ by $K \in \mathcal{T}$. For $i \in \mathrm{I}$, $\overset{\star}{\mathrm{e}}(i;\mathcal{T})$ denotes the set of simplices in $\mathcal{T}$ sharing the vertex $i$ and $\overset{\star}{\mathrm{I}}(i;\mathrm{C}) \subset \mathrm{I}\setminus\{i\}$ denotes the set of indices of vertices of the simplices in $\overset{\star}{\mathrm{e}}(i;\mathcal{T})$ while omitting $i$. The arguments of $\overset{\star}{\mathrm{I}}$ emphasize its exclusive dependence on the connectivity of the mesh. Figure 2 shows an example explaining the identification of $\overset{\star}{\mathrm{I}}(i;\mathrm{C})$ and $\overset{\star}{\mathrm{e}}(i;\mathcal{T})$.

**2.4. The algorithm.** dvr is an iterative algorithm in which (i) the connectivity C of a given triangulation $\mathcal{T}$ is maintained, (ii) the locations of a given collection of vertices $\mathrm{I_R} \subseteq \mathrm{I}$ are altered while the remaining vertices in $\mathrm{I}\setminus\mathrm{I_R}$ are held fixed, and (iii) the perturbation direction for each vertex $i \in \mathrm{I_R}$ is prescribed. We shall reserve $k$ to denote the iteration count and $i,j$ for indices in $\mathrm{I_R}$. We assume $\mathrm{I_R}$ to be an ordered list of $m$ vertices and, for convenience, that $\mathrm{I_R} = (1,2,\dots,\mathrm{m})$. The relaxation direction prescribed for $i \in \mathrm{I_R}$ during the $k$th iteration is denoted by $\mathbf{d}_{k,i}$.

We shall denote the mesh resulting from perturbing vertex $i \in \mathrm{I_R}$ of the mesh $\mathcal{T}$ by a signed distance (i.e., coordinate) $\lambda \in \mathbb{R}$ by $\mathcal{T}^{i,\lambda} = (\mathrm{V}^{i,\lambda},\mathrm{I},\mathrm{C})$. For notational simplicity, we omit explicitly mentioning the perturbation direction, since it is assumed to be known from the context (iteration count). We label the element quality metric by Q. The requisite properties for Q and specific examples for it will be discussed in subsequent sections. For now, it suffices to note that Q is a scalar-valued function

defined over simplices in $\mathbb{R}^d$ and that Q assigns larger values to simplices of better quality. We introduce the shorthand

$$q_{i,j}(\lambda, \mathcal{T}) \triangleq \min\{\mathrm{Q}(K) : K \in \overset{\star}{\mathrm{e}}\,(j, \mathcal{T}^{i,\lambda})\} \text{ for } \lambda \in \mathbb{R} \text{ and } i, j \in \mathrm{I_R} \tag{3}$$

to refer to the minimum among the qualities of simplices in the 1-ring of vertex $j$ in the mesh resulting from perturbing vertex $i$ of $\mathcal{T}$ by a coordinate $\lambda$ along the prescribed direction. In particular, perturbing vertex $i$ can only alter the values of $q_{i,j}(\lambda, \mathcal{T})$ for $j \in \overset{\star}{\mathrm{I}}\,(i; \mathrm{C}) \cup \{i\}$; see Figure 2.

**Algorithm 1:** Directional vertex relaxation (dvr)

**Input:**
$\mathcal{T} = (\mathrm{V}, \mathrm{I}, \mathrm{C})$ : Input triangulation
$\mathrm{I_R} = (1, \ldots, \mathrm{m})$ : Ordered list of vertices to relax
$\{\mathbf{d}_{k,i}\}_{i \in \mathrm{I_R}, \mathrm{k} \in \mathbb{N}}$ : Relaxation directions
$N_\mathrm{R}$ : Number of relaxation iterations

1 **for** $k = 1$ **to** $N_R$ **do**
2  **for** $i = 1$ **to** $m$ **do**
3  Compute $\mathrm{L} \triangleq \arg\max_{\lambda \in \mathbb{R}} q_{i,i}(\lambda, \mathcal{T})$
4  **if** $\#L = 1$  ▷ `Always satisfied for the mean ratio metric`
5  **then**
6  Set $\lambda^{\mathrm{opt}} \leftarrow$ element in L
7  **else**
8  Find $\lambda^{\mathrm{opt}} \in \arg\max_{\lambda \in \mathrm{L}} \mathbf{asc}(\{q_{i,j}(\mathcal{T}, \lambda) \,:\, j \in \overset{\star}{\mathrm{I}}\,(i, \mathrm{C}) \cap \mathrm{I_R}\})$
9  **end**
10  Update: $\mathrm{V}_i \leftarrow \mathrm{V}_i + \lambda^{\mathrm{opt}}\, \mathbf{d}_{k,i}$
11  **end**
12 **end**
13 **return** $(\mathrm{V}, \mathrm{I}, \mathrm{C})$

dvr is defined by Algorithm 1. Therein, we have denoted the cardinality of the set L by #L. Observe that $\arg\max_{\lambda \in \mathrm{L}} \mathrm{asc}(\cdot)$ in step 9 implicitly uses the ordering introduced in section 2.2. In the algorithm, we have also assumed the number of relaxation iterations to be limited to $N_\mathrm{R}$; alternate termination criteria are discussed in section 4.5.

**2.5. Discussion of the algorithm.** We use the example shown in Figure 2 to discuss the steps in Algorithm 1. Specifically, we will examine the finer points involved in computing the update for vertex 1 in the mesh $\mathcal{T}$ shown in Figure 2a during a representative iteration. For simplicity, we denote the relaxation direction for vertex 1 by $\mathbf{d}_1$ without explicit reference to the iteration counter.

**2.5.1. Nonsmoothness.** The crux in computing the optimal perturbation $\lambda^{\mathrm{opt}}$ for vertex 1 along $\mathbf{d}_1$ lies in identifying the set of candidate coordinates

$$\mathrm{L} = \arg\max_{\lambda \in \mathbb{R}} q_{1,1}(\lambda, \mathcal{T})$$

in step 4 of the algorithm. By definition (3), we have

$$q_{1,1}(\lambda, \mathcal{T}) \triangleq \min\{\mathrm{Q}(K_1^\lambda), \mathrm{Q}(K_2^\lambda), \mathrm{Q}(K_3^\lambda), \mathrm{Q}(K_4^\lambda)\}.$$

Presuming that Q is a smooth function of vertex coordinates of a triangle, $\lambda \mapsto \mathrm{Q}(K_\ell^\lambda)$ is a smooth map for each $\ell = 1, \ldots, 4$. As a minimum over these smooth maps, however, $q_{1,1}(\lambda, \mathcal{T})$ is expected to be continuous but not differentiable with respect to $\lambda$. Consequently, L cannot be computed by straightforward derivative-based methods.

As discussed at length in section 1, nonsmoothness of the function being optimized is a recurring challenge in mesh optimization methods that is invariably resolved by regularizing the problem. Notice, for instance, that replacing $q_{1,1}(\lambda, \mathcal{T})$ by

$$q_{1,1}^{\mathrm{reg}}(\lambda, \mathcal{T}) \triangleq \left(\mathrm{Q}(K_1^\lambda)^2 + \mathrm{Q}(K_2^\lambda)^2 + \mathrm{Q}(K_3^\lambda)^2 + \mathrm{Q}(K_4^\lambda)^2\right)^{1/2}$$

yields a smooth function of $\lambda$. Hence maximizers of $\lambda \mapsto q_{1,1}^{\mathrm{reg}}(\lambda, \mathcal{T})$ can be easily found using Newton-type methods. Such convenience, however, comes at an important price. While the maximizer $\lambda^{\mathrm{opt}} \in \arg\max_\lambda q_{1,1}(\lambda, \mathcal{T})$ in the dvr algorithm guarantees improvement of the minimum quality among triangles in the 1-ring of vertex 1, i.e., $q_{1,1}(\lambda^{\mathrm{opt}}, \mathcal{T}) \geq q_{1,1}(0, \mathcal{T})$, no such guarantees exist for the maximizer of $q_{1,1}^{\mathrm{reg}}(\lambda, \mathcal{T})$.

**2.5.2. Computing maximizers.** Despite the nonsmoothness of $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$, reasonable assumptions on Q facilitate a straightforward calculation of its maximizers. Specifically, if each smooth curve $\lambda \mapsto \mathrm{Q}(K_\ell^\lambda)$ for $1 \leq \ell \leq 4$ has finitely many maxima, and if each pair of distinct curves intersect at finitely many points, then computing the maximizers of $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$ reduces to finding roots of (nonlinear) scalar equations; see section 3.2 for details.

In practice, it is likely that L may be computed only approximately. This may be a result of the choice of the numerical algorithm employed for the calculation or an intentional compromise because approximations may be computable very efficiently. Irrespective of the reason, we have to acknowledge the possibility that not all maximizers of $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$ may be identified, and even the computed ones may be approximate. The perturbation $\lambda^{\mathrm{opt}}$ identified in practice may therefore be suboptimal. We consider such scenarios and their consequences in section 4.5.

**2.5.3. Choice of optimal coordinate.** If the map $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$ has a unique maximizer, L is a singleton set and the choice of the optimal coordinate $\lambda^{\mathrm{opt}}$ is obvious. In section 4.3, we will identify certain convexity conditions on (level sets of) Q to ensure that this is indeed the case. Algorithm 1, however, accommodates more general quality metrics when it may be possible that multiple maximizers exist, i.e., $\#\mathrm{L} > 1$. The rationale behind choosing $\lambda^{\mathrm{opt}}$ using the more intricate check in step 9 of the algorithm is related to ensuring monotonic mesh qualities and is discussed in section 4.3. We point out that the check in step 4 is in principle redundant, because step 9 identifies $\lambda^{\mathrm{opt}}$ even in the case when $\#\mathrm{L} = 1$. Nevertheless, we include step 4 to emphasize the simplification that results when $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$ has a unique maximizer. By adopting the mean ratio metric defined in section 3.1, this is the case in all our numerical experiments.

**2.5.4. Relaxation directions.** True to its name, the assumption of prescribed relaxation directions plays a crucial role in dvr. It is by virtue of this assumption that we arrive at a one-dimensional problem for computing vertex updates, irrespective of the spatial dimension. The choice of relaxation directions directly influences the improvement in mesh qualities possible with dvr. Numerical experiments presented in the following section suggest providing at least $d$ (the spatial dimension) linearly independent relaxation directions for each vertex. Good choices for these directions can result in drastic improvements in mesh quality, as well as reduce the number of iterations required in the relaxation algorithm.

**2.5.5. Relaxation iterations.** The optimization problem defining the update for vertex 1 during the $k$th relaxation iteration is in general different from that realized during the $(k+1)$st iteration. This is because one or more among the vertices $2, \ldots, 5$ in its 1-ring may have been perturbed between successive updates of vertex 1. Furthermore, the direction prescribed for relaxing vertex 1 may be different during successive iterations, which again contributes to rendering different optimization problems for its update during each iteration. In Theorem 4.2, we will prove that with certain reasonable assumptions on the quality metric and on the quality of the initial mesh, each vertex update in dvr can only improve the mesh quality. It is therefore desirable to perform multiple iterations of the algorithm until either mesh qualities have converged or are deemed to have improved sufficiently. For simplicity, we have limited the number of iterations to $N_{\mathrm{R}}$ in Algorithm 1.

**3. The mean ratio metric, vector-valued mesh qualities, and illustrative examples.** Before proceeding to scrutinize the performance of Algorithm 1 and properties of the meshes it computes, we pause to discuss additional details involved in implementing the algorithm. For this purpose, we adopt a specific element quality metric, namely the mean ratio metric. In fact, we use the mean ratio metric for measuring simplex qualities in all our numerical experiments improving triangle and tetrahedral meshes. Here we recall the definition of the mean ratio metric and highlight a few of its important properties. Then we introduce a vector-valued notion for measuring mesh qualities and discuss using it in conjunction with the ordering relation defined in section 2.2 to compare the qualities of mesh iterates computed by dvr. Deferring a detailed examination of the performance of dvr and its representative applications to section 5, we present a couple of illustrative examples at the end of this section which, besides providing a glimpse of the mesh improvement possible with dvr, also serve to set the stage for the analysis that follows in section 4.

**3.1. The mean ratio metric for element quality.** Let $\mu(K)$ denote the signed measure of simplex $K$ (signed area in case of a triangle and signed volume in case of a tetrahedron) and $\ell_{pq}(K)$ denote the length of the edge joining its $p$th and $q$th vertices. The *mean ratio* metric introduced in [36] is defined as

$$\mathrm{Q}(K) \triangleq C(d)\,\frac{\mathrm{sgn}(\mu(K))|\mu^{2/d}(K)|}{\displaystyle\sum_{1\le p<q\le d+1}\ell_{pq}^2(K)}, \quad \text{where } \begin{cases} C(2) & = 4\sqrt{3}, \\ C(3) & = 12\sqrt[3]{9}. \end{cases} \tag{4}$$

The metric (4) is dimensionless, independent of the size of the simplex (invariant under homogeneous dilations), invariant under isometric coordinate transformations, equals zero if and only if the simplex is degenerate, has the same sign as the orientation of the simplex, and has a unique extremum equal to a maximum when the simplex is regular [5, 36, 42]. The dimension-dependent normalization factor $C(d)$ is chosen such that regular simplices are assigned value 1. Hence equilateral triangles and regular tetrahedra have unit quality. The triangle quality defined by (4) is related to other commonly used quality metrics. For instance, it is equivalent[1] to the metric defined as the minimum interior angle of $K$ and weakly equivalent to the aspect ratio metric which is defined as the ratio of the inradius to the circumradius of the triangle. Further

[1]The element quality metrics $\mathrm{Q}_1$ and $\mathrm{Q}_2$ are weakly equivalent if there exist positive constants $c_1, c_2, \alpha$, and $\beta$ such that $c_1\mathrm{Q}_1^{\alpha}(K) \le \mathrm{Q}_2(K) \le c_2\mathrm{Q}_2^{\beta}(K)$ for arbitrary $K$ with $\mathrm{Q}_1(K) > 0$. They are equivalent if $\alpha = \beta = 1$.

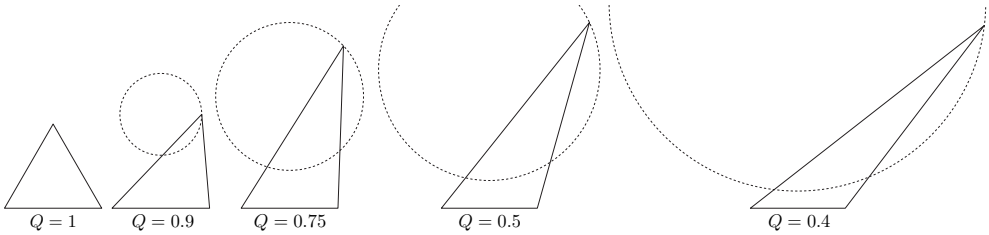



FIG. 3. *Triangles with qualities defined using the metric in* (4). *With two vertices held fixed, the locus of the third vertex to achieve a prescribed quality is a circle. Analogously, the locus of the free vertex in a tetrahedron with prescribed quality is a sphere (not shown here).*

examples of triangle/tetrahedron quality metrics and their relationships can be found in [34, 37, 42]. Figure 3 shows examples of triangles with a few different qualities, as defined by (4). The figure also shows the circular level sets of $\mathrm{Q}(K)$ as a function of the position of one of the vertices of triangle $K$.

**3.2. Computing maximizers in dvr with the mean ratio metric.** Following the discussions in section 2.5.2, let us reexamine the problem of identifying maximizers of $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$ for the example in Figure 2, with Q given by (4) this time. Without loss of generality, we assume that the maps $\{\lambda \mapsto \mathrm{Q}(K_\ell^\lambda) : 1 \le \ell \le 4\}$ are distinct. When this is not the case, i.e., when one or more of these maps coincide, we simply retain the set of unique ones, since these will suffice for defining $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$ and in turn for identifying its maximizer. We are now concerned with computing $\mathrm{L} = \arg\max_{\lambda \in \mathbb{R}} \min_{1 \le \ell \le 4} \mathrm{Q}(K_\ell^\lambda)$.

Straightforward algebraic computations using the expressions for Q in (4) reveal that extrema of the curve $\lambda \mapsto \mathrm{Q}(K_\ell^\lambda)$ are roots of a quadratic polynomial in $\lambda$ while intersection points of a pair of distinct curves $\lambda \mapsto \mathrm{Q}(K_{\ell_1}^\lambda)$ and $\lambda \mapsto \mathrm{Q}(K_{\ell_2}^\lambda)$ are roots of a cubic polynomial. In particular,

$$\mathrm{L}_{\max}^{\alpha} \triangleq \{\eta \in \mathbb{R} : \mathrm{Q}(K_\alpha^\lambda) \text{ has a local maximum at } \lambda = \eta\} \text{ for } 1 \le \alpha \le 4, \tag{5a}$$

$$\mathrm{L}_{\mathrm{int}}^{\alpha\beta} \triangleq \{\lambda \in \mathbb{R} : \mathrm{Q}(K_\alpha^\lambda) = \mathrm{Q}(K_\beta^\lambda)\} \text{ for } 1 \le \alpha < \beta \le 4; \tag{5b}$$

each contain finitely many points (at most two and three, respectively, to be precise). Then successive points in $\mathrm{L}_{\mathrm{int}} \triangleq \{\pm\infty\} \cup_{1 \le \alpha < \beta \le 4} \mathrm{L}_{\mathrm{int}}^{\alpha\beta}$ define intervals over which $\lambda \mapsto q_{1,1}(\lambda, \mathcal{T})$ equals one of the smooth functions $\{\lambda \mapsto \mathrm{Q}(K_\ell^\lambda)\}_{\ell=1}^4$. With $\mathrm{L}_{\max} \triangleq \cup_{\alpha=1}^{p} \mathrm{L}_{\max}^{\alpha}$, it is therefore evident that

$$\mathrm{L} \subseteq \mathrm{L}_{\max} \cup \mathrm{L}_{\mathrm{int}}. \tag{6}$$

In addition to showing that L contains finitely many points, (6) provides a practical method of computing L. We highlight that contrary to what (6) suggests, the points $\pm\infty$ are in fact not candidates for the optimizer and can therefore be discarded from L. This will be evident from the conditions on the quality metric and the input mesh discussed in section 4.

The above discussions referring to the example in Figure 2 apply verbatim to general triangle meshes, as well as to tetrahedral meshes. In the latter case, the only distinction worth noting is that computing the set of pairwise intersections of distinct quality curves, i.e., the set $\mathrm{L}_{\mathrm{int}}^{\alpha\beta}$, requires the resolution of an octic polynomial.

Nevertheless, $\mathrm{L}_{\max} \cup \mathrm{L}_{\mathrm{int}}$ remains a finite set and is straightforward to compute.

**Algorithm 2:** Candidate perturbation coordinates

**Input:**
$\mathcal{T} = (\mathrm{V}, \mathrm{I}, \mathrm{C})$ : Triangulation
$i \in \mathrm{I}_{\mathrm{R}}$ : Vertex to be relaxed
$\mathbf{d}_{k,i}$ : Prescribed relaxation direction for $i$ during the $k$th relaxation iteration

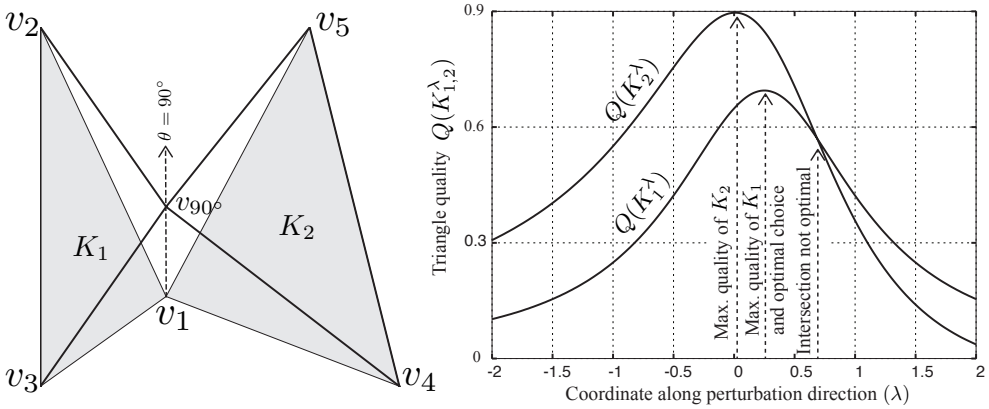
Set $\{K_{i_\alpha}\}_{\alpha=1}^{p} \triangleq \overset{\star}{\mathrm{e}}\,(i, (\mathrm{V}, \mathrm{I}, \mathrm{C}))$ ▷ `Elements in the 1-ring of` $i$
Reduce $p$ and reindex until $\{\lambda \mapsto \mathrm{Q}(K_{i_\alpha}^{\lambda})\}_{\alpha=1}^{p}$ are distinct curves
Set $\mathrm{L}_{\mathrm{cand}} \triangleq \emptyset$ ▷ `Set of candidate optimizers`
**for** $\alpha = 1$ **to** $p$ **do**
  ▷ `Append local maximizers`
  $\mathrm{L}_{\mathrm{cand}} \leftarrow \mathrm{L}_{\mathrm{cand}} \cup \{\eta : \lambda \mapsto \mathrm{Q}(K_{i_j}^{\lambda})$ has a local maximum at $\lambda = \eta\}$
  **for** $\beta = \alpha + 1$ **to** $p$ **do**
    ▷ `Append pairwise intersections`
    $\mathrm{L}_{\mathrm{cand}} \leftarrow \mathrm{L}_{\mathrm{cand}} \cup \{\lambda : \mathrm{Q}(K_{i_\alpha}^{\lambda}) = \mathrm{Q}(K_{i_\beta}^{\lambda})\}$
  **end**
**end**
**return** $\mathrm{L} = \arg\max_{\lambda \in \mathrm{L}_{\mathrm{cand}}} q_{i,i}(\lambda, \mathcal{T})$

Algorithm 2 provides a summary of the steps involved in identifying the set L in step 4 of Algorithm 1, provided that $\#(\mathrm{L}_{\max} \cup \mathrm{L}_{\mathrm{int}}) < \infty$. While the simple algebraic form of the mean ratio metric helped us to confirm this rather easily, the requirement needs to be independently checked for alternative choices of quality metrics. The main point of Algorithm 2 is that the computing $\mathrm{L} = \arg\max_{\lambda \in \mathbb{R}} q_{i,i}(\mathcal{T}, \lambda)$ required in Algorithm 1 is replaced by the simpler calculation $\mathrm{L} = \arg\max_{\lambda \in \mathrm{L}_{\mathrm{cand}}} q_{i,i}(\mathcal{T}, \lambda)$ in Algorithm 2. The latter only requires comparing the qualities of elements in the 1-ring of $i$ at the finitely many points in the set $\mathrm{L}_{\mathrm{cand}}$. In [45], we describe a more efficient alternative to Algorithm 2 for computing L that exploits properties of the quality metric discussed in section 4. We in fact use the algorithm from [45] for some of the examples in section 5 that require a large number of vertices to be relaxed.

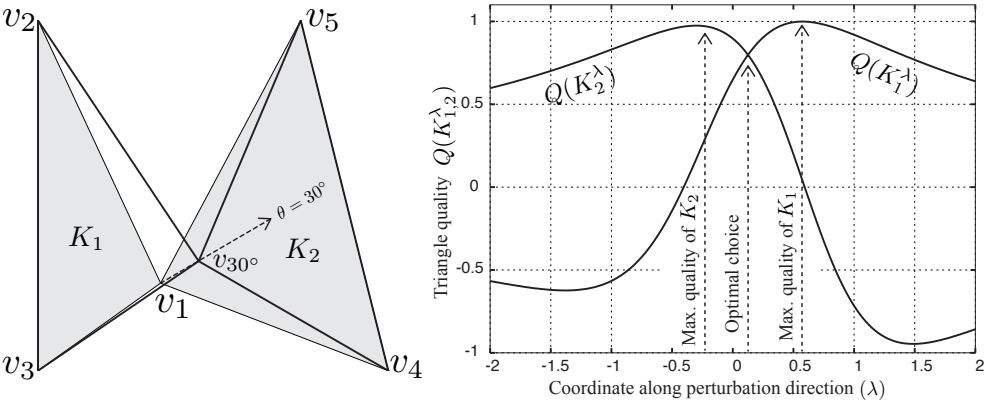
**3.3. Vertex updates: A 2-element example.** Figure 4 provides a graphical illustration of Algorithm 2 in a simple example where a vertex $v_1$ having just two triangles $K_1$ and $K_2$ in its 1-ring is perturbed along a prescribed direction $\mathbf{d}_1$. Let $K_1^{\lambda}$ and $K_2^{\lambda}$ denote the triangles with vertices $\{v_1 + \lambda\,\mathbf{d}_1, v_2, v_3\}$ and $\{v_1 + \lambda\,\mathbf{d}_1, v_4, v_5\}$, respectively. Algorithm 2 identifies (i) the set of local maxima $\mathrm{L}_{\max}^{1}$ and $\mathrm{L}_{\max}^{2}$ of the curves $\lambda \mapsto \mathrm{Q}(K_1^{\lambda})$, $\lambda \mapsto \mathrm{Q}(K_2^{\lambda})$, respectively, by solving a pair of quadratic equations and (ii) the set of points $\mathrm{L}_{\mathrm{int}}^{12}$ where the two curves intersect by finding roots of a cubic polynomial. Then the desired set of maximizers $\mathrm{L} = \min\{\mathrm{Q}(K_1^{\lambda}), \mathrm{Q}(K_2^{\lambda})\}$ is necessarily contained in $\mathrm{L}_{\mathrm{cand}} = \mathrm{L}_{\max}^{1} \cup \mathrm{L}_{\max}^{2} \cup \mathrm{L}_{\mathrm{in}}^{12}$.

Figure 4 explains the identification of the optimal perturbation $\lambda^{\mathrm{opt}}$ in $\mathrm{L}_{\mathrm{cand}}$ for two different choices of $\mathbf{d}_1$. In Figure 4a, where $\mathbf{d}_1$ is vertical, we observe that $\lambda^{\mathrm{opt}}$ equals the coordinate at which $\lambda \mapsto \mathrm{Q}(K_1^{\lambda})$ is maximized. When $\mathbf{d}_1$ is inclined at $30^{\circ}$ to the horizontal as shown in Figure 4b, $\lambda_{\mathrm{opt}}$ equals the coordinate at which the two quality curves intersect. The final triangle shapes following the vertex update $v_1^{\mathrm{new}} = v_1 + \lambda^{\mathrm{opt}}\,\mathbf{d}_1$ are also shown. Observe that for both choices of $\mathbf{d}_1$, we found a unique optimal coordinate in this example. We will see in section 4.3 why this is to be expected when using the mean ratio metric.

The general case in which $p$ triangles $\{K_\alpha\}_{\alpha=1}^{p}$ are incident at $v_1$ is now easily




(a) When $v_1$ is perturbed along the vertical direction, the qualities $\mathrm{Q}(K_1^\lambda)$ and $\mathrm{Q}(K_2^\lambda)$ of the perturbed triangles as a function of the coordinate $\lambda$ along the perturbation direction are shown on the right. The optimal perturbation $\lambda^{\mathrm{opt}} = 0.25$ occurs at the maximum of the curve $\lambda \mapsto \mathrm{Q}(K_1^\lambda)$.




(b) When the perturbation direction is inclined at 30° to the horizontal, the optimal coordinate $\lambda^{\mathrm{opt}} = 0.125$ occurs at the intersection of the two quality curves. As a result, the qualities of triangles $K_1$ and $K_2$ are altered from 0.65 and 0.90, respectively, to a common optimal value of 0.8.

FIG. 4. *The vertex $v_1$ common to triangles $K_1$ and $K_2$ is perturbed along a prescribed direction to maximize the smaller among the qualities of the resulting triangles.*

handled. We identify the maxima of each quality curve and their pairwise intersections; the optimal perturbation necessarily equals one of these computed coordinates. Noting that we have a unique optimizer when using the mean ratio metric, $\lambda^{\mathrm{opt}}$ is identified by evaluating element qualities at the computed points and picking the coordinate at which the poorest element quality in the 1-ring of $v_1$ is maximized.

**3.4. Vector-valued mesh quality.** With the intention of monitoring qualities of mesh iterates computed in dvr, it is straightforward to define a measure for mesh quality once an element quality metric Q is chosen (the mean ratio metric (4) is just one example for Q). Common definitions for mesh qualities are predominantly as $\ell^p$-norms of the sequence of its element qualities. An alternate and more stringent metric relevant to finite element methods consists in defining the mesh quality as the minimum among the qualities of all its elements.

In contrast to the different scalar-valued measures of mesh quality possible, we

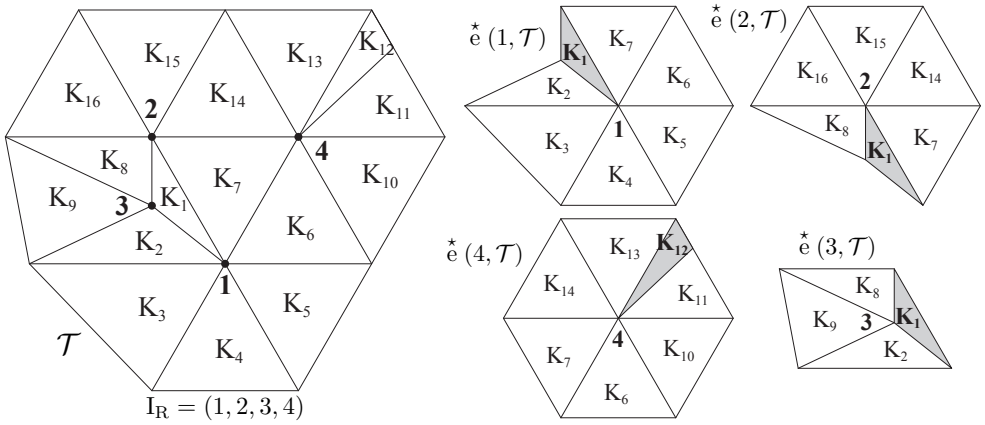



FIG. 5. *An illustrated explanation of the definition of the mesh quality* $\mathbf{Q}(\mathcal{T})$. *For the choice* $\mathrm{I_R} = (1, 2, 3, 4)$, *triangles in the* 1*-ring of each vertex in* $\mathrm{I_R}$ *are shown on the right. In each* 1*-ring, the triangle possessing the poorest quality is highlighted in gray. For vertices* $1, 2, 3,$ *and* $4$, *these triangles are* $K_1, K_1, K_1,$ *and* $K_{12}$, *respectively. Assuming* $\mathrm{Q}(K_1) \leq \mathrm{Q}(K_{12})$, *we have* $\mathbf{Q}(\mathcal{T}) = (\mathrm{Q}(K_1), \mathrm{Q}(K_1), \mathrm{Q}(K_1), \mathrm{Q}(K_{12}))$. *Notice that* $\mathbf{Q}(\mathcal{T})$ *is very different from just a list of all the element qualities in* $\mathcal{T}$. *For example,* $\mathrm{Q}(K_1)$ *appears thrice in* $\mathbf{Q}(\mathcal{T})$.

introduce a vector-valued mesh quality metric $\mathbf{Q}$ that facilitates evaluating the performance and analyzing the properties of the dvr algorithm. To this end, note that for a given mesh $\mathcal{T} = (\mathrm{V}, \mathrm{I}, \mathrm{C})$ to be improved, the meshes computed in the dvr algorithm necessarily belong to the family

$$\mathrm{Pert}(\mathcal{T}) = \{(\mathrm{V}', \mathrm{I}, \mathrm{C}) \,:\, \mathrm{V}' \in \left[\mathbb{R}^d\right]^N, \mathrm{V}'_j = \mathrm{V}_j \text{ if } j \in \mathrm{I} \setminus \mathrm{I_R}\} \tag{7}$$

irrespective of the prescribed relaxation directions. Each mesh in $\mathrm{Pert}(\mathcal{T})$ has the same set of vertices and the same connectivity as $\mathcal{T}$, but their vertices in $\mathrm{I_R}$ can assume arbitrary locations. Then we define $\mathbf{Q} : \mathrm{Pert}(\mathcal{T}) \to \mathbb{R}^m$ as

$$\mathbf{Q}(\mathcal{S}) \triangleq \mathbf{asc}\left(\uplus_{i \in \mathrm{I_R}} \{\min\{\mathrm{Q}(K) : K \in \overset{\star}{e}(i; \mathcal{S})\}\}\right), \ \mathcal{S} \in \mathrm{Pert}(\mathcal{T}). \tag{8}$$

Hence the vector $\mathbf{Q}(\mathcal{S})$ contains the minimum element quality in the 1-ring of each vertex in $\mathrm{I_R}$ and has its components sorted in ascending order. We refer to $\mathbf{Q}(\mathcal{S})$ as the vector-valued quality of the mesh $\mathcal{S}$, or simply as the quality of $\mathcal{S}$. We have intentionally omitted the dependence of $\mathrm{Pert}(\mathcal{T})$ and $\mathbf{Q}$ on $\mathrm{I_R}$ in favor of notational simplicity, since the choice of $\mathrm{I_R}$ will always be clear from the context. Figure 5 shows an example to further clarify the interpretation of (8).

Equipped with (8), we can now compare the qualities of any pair of triangulations in $\mathrm{Pert}(\mathcal{T})$ using the ordering $\geq$ on $\mathbb{R}^m$. We say that the meshes $\mathcal{S}_1, \mathcal{S}_2 \in \mathrm{Pert}(\mathcal{T})$ have identical qualities if $\mathbf{Q}(\mathcal{S}_1) = \mathbf{Q}(\mathcal{S}_2)$ as vectors in $\mathbb{R}^m$ and that $\mathcal{S}_1$ has better quality than $\mathcal{S}_2$ if $\mathbf{Q}(\mathcal{S}_1) > \mathbf{Q}(\mathcal{S}_2)$.

We highlight a few significant features of the mesh quality vector $\mathbf{Q}$.

(i) The first component $\mathrm{Q}_1(\mathcal{S})$ of the mesh quality $\mathbf{Q}(\mathcal{S})$ equals the minimum among the qualities of all simplices that can be perturbed in $\mathcal{S}$ (i.e., simplices having at least one vertex in $\mathrm{I_R}$). We have already alluded to the importance of $\mathrm{Q}_1(\mathcal{S})$ in finite element methods in section 1.

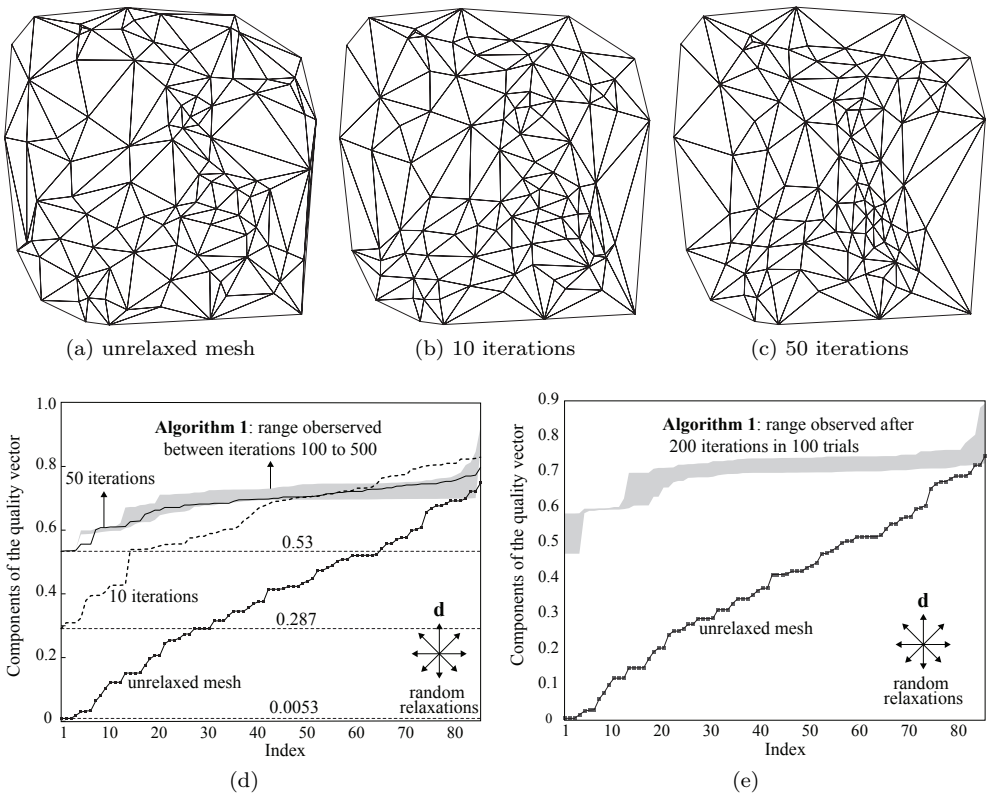



FIG. 6. *The Delaunay triangulation of a random set of points is improved by relaxing the interior vertices using* dvr. *Qualities of a few mesh iterates are plotted in* (d). *Since relaxation directions are randomly generated,* (e) *examines mesh qualities at the end of a fixed number of iterations in* 100 *different trials. Observe the consistent improvement in mesh quality and that* $\mathrm{Q}_1$ *is improved by factors close to* 100.

(ii) It is possible, and is indeed regularly the case, that the quality of the same element in $\mathcal{S}$ appears multiple times in the mesh quality vector $\mathbf{Q}(\mathcal{S})$. In fact, definition (8) naturally accentuates the influence of elements with poor qualities on the mesh quality vector $\mathbf{Q}(\mathcal{S})$. This should be contrasted with definitions for mesh quality that deliberately introduce "weighting" factors associated with each element [48].

(iii) In general, equality of $\mathbf{Q}$ only defines an equivalence relation over $\mathrm{Pert}(\mathcal{T})$, since it is possible that $\mathbf{Q}(\mathcal{S}_1) = \mathbf{Q}(\mathcal{S}_2)$ even though $\mathcal{S}_1$ and $\mathcal{S}_2$ have different lists of coordinates. This observation implies that convergence of mesh qualities in the dvr algorithm may not imply convergence of triangulations themselves.

(iv) We emphasize the distinction between $\mathbf{Q}(\mathcal{S})$ and the list of element qualities sorted in ascending order that is sometimes considered in the literature [32]. That $\mathbf{Q}(\mathcal{S})$ is different from the latter vector should be evident simply from recognizing that the two have different lengths in general. Similarly, the quality of the same element may recur multiple times in $\mathbf{Q}(\mathcal{S})$, which is not the case with the sorted list of element qualities.

**3.5. Example with a triangle mesh.** Figure 6a shows the Delaunay triangulation of a set of 99 randomly distributed points in a unit square. The distribution

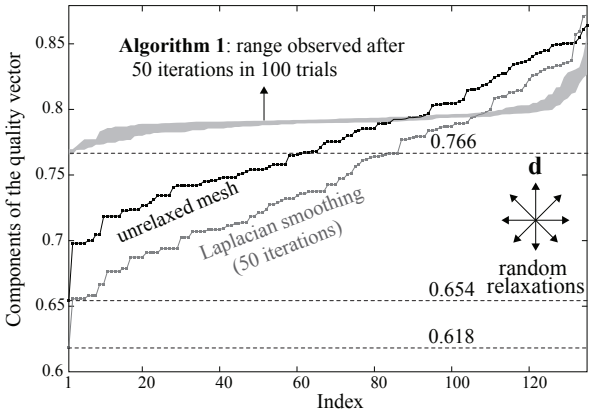



FIG. 7. *Optimizing a tetrahedral mesh by relaxing its vertices along randomly generated directions.*

of vertices results in many triangles with poor qualities. Since the triangulation is Delaunay, their qualities (as measured by the minimum interior angle) cannot be improved further by altering the connectivity of the mesh alone. Instead, we seek to maintain its connectivity and improve the mesh by perturbing its vertices using dvr.

In this example, all 86 interior vertices in the mesh are relaxed while those along the boundary are held fixed. The relaxed vertices are ordered according to their node numbers assigned arbitrarily by the mesh generator to create the list $\mathrm{I_R}$. For lack of a more natural alternative, a new relaxation direction $\mathbf{d}_{k,i}$ is randomly generated for each vertex $i \in \mathrm{I_R}$ at each relaxation iteration $k$.

Figures 6b and 6c show the meshes computed at the end of 10 and 50 iterations, respectively, for one set of relaxation directions realized in Algorithm 1. Inspecting the components of $\mathbf{Q}$ plotted in Figure 6d reveals that at the end of 50 iterations, the first component $\mathrm{Q}_1$ of $\mathbf{Q}$, which equals the minimum among the qualities of all triangles in the mesh, is improved by a factor of 100. The figure also highlights an important feature of dvr—to improve triangles with poorer qualities, the qualities of better-shaped ones are intentionally sacrificed. Between the 10th and 50th iterations, we see that the first 55 components of $\mathbf{Q}$ are improved significantly at the expense of the remaining components.

Since relaxation directions are chosen randomly, we examine multiple realizations for them while retaining the same definition of $\mathrm{I_R}$. Inspecting the results from 100 different trials shown in Figure 6e reveals consistent improvement in mesh qualities by factors exceeding 80 for the first component. We emphasize that such improvement is despite the arbitrary choice of relaxation directions.

**3.6. Example with a tetrahedral mesh.** Figure 7 shows results from an analogous experiment to improve a mesh of tetrahedra by relaxing its vertices. The input mesh of unit cube has 295 vertices and 1184 tetrahedra and can be retrieved from [52, mesh: cube1k]. We choose $\mathrm{I_R}$ to be the set of all 135 interior vertices while leaving the vertices on the boundary of the cube fixed. At each iteration of Algorithm 1, the relaxation direction for each vertex is randomly generated.

Comparing the components of $\mathbf{Q}$ for the unrelaxed mesh and the range of values observed at the end of 50 iterations in 100 different trials, we observe consistent improvement in mesh qualities. The plot also shows the result of 50 iterations of Laplacian smoothing. In this example, Laplacian smoothing in fact yields a mesh

with a poorer quality compared to the unrelaxed one.

**4. Analysis of the directional vertex relaxation algorithm.** The examples in the previous section provide encouraging evidence of the improvement in mesh qualities possible with dvr. Now we ask what guarantees of mesh improvement are extended by the algorithm. Our main result to this end is Theorem 4.2, which shows that the sequence of mesh iterates computed by dvr has monotonic (nondecreasing) qualities, provided that the initial mesh has no degenerate/inverted elements and the element quality metric Q satisfies a few reasonable assumptions. We start by identifying conditions for the algorithm to be well defined in the first place.

**4.1. Existence of optimal perturbations.** Evidently, Algorithm 1 is well defined if the set $\mathrm{L} = \arg\max_{\lambda\in\mathbb{R}} q_{i,i}(\lambda, \mathcal{T})$ is nonempty for each $i \in \mathrm{I_R}$ at each iteration. To this end, we identify sufficient conditions on Q. Let $K(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_d)$ denote the simplex with vertices $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_d \in \mathbb{R}^d$, and let us denote the Euclidean norm of $\mathbf{x} \in \mathbb{R}^d$ by $\|\mathbf{x}\|_d$. We consider the following restrictions:

**A**1. *Degeneracy:* $\mu(K) = 0 \iff \mathrm{Q}(K) = 0$ if the vertices of $K$ do not all coincide.

**A**2. *Continuity:* $\mathrm{Q}(K(\mathbf{x}_0, \ldots, \mathbf{x}_d))$ is continuous function on $[\mathbb{R}^d]^{d+1} \setminus \mathbb{I}^d$, where $\mathbb{I}^d \triangleq \{(\mathbf{x}, \ldots, \mathbf{x}) : \mathbf{x} \in \mathbb{R}^d\}$.

**A**3. *Decay:* $\mathrm{Q}(K(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_d)) \to 0$ as $\|\mathbf{x}\|_d \to \infty$ for any $(\mathbf{x_1}, \ldots, \mathbf{x_d}) \in [\mathbb{R}^d]^d$.

These conditions are straightforward to interpret. In particular, **A**3 states that the quality of a simplex should approach zero when one of its vertices is moved far enough while holding all the others fixed. Thus if the quality of a simplex is initially positive, moving any one of its vertices far enough will result in a simplex with lower quality.

Let $\mathcal{Z}(\mathcal{T})$ consist of triangulations in $\mathrm{Pert}(\mathcal{T})$ that have at least one simplex in which all $d+1$ vertices coincide, and set $\mathrm{RegPert}(\mathcal{T}) \triangleq \mathrm{Pert}(\mathcal{T}) \setminus \mathcal{Z}(\mathcal{T})$.

PROPOSITION 4.1. *If* Q *satisfies* **A**1–**A**2 *and* $\mathrm{Q}_1(\mathcal{T}) > 0$*, then the following hold:*

(i) $\lambda \mapsto q_{i,j}(\lambda, \mathcal{T})$ *is continuous for each* $i, j \in \mathrm{I_R}$.

(ii) $\lambda \mapsto \mathbf{Q}(\mathcal{T}^{i,\lambda})$ *is continuous for each* $i \in \mathrm{I_R}$.

(iii) $\mathbf{Q} : \mathit{RegPert}(\mathcal{T}) \to \mathbb{R}^m$ *is continuous.*

(iv) *If* Q *additionally satisfies* **A**3*, then the set* $\{\lambda \in \mathbb{R} : q_{i,j}(\lambda, \mathcal{T}) \geq q_{i,j}(0, \mathcal{T})\}$ *is compact.*

We omit a detailed proof of the proposition but provide a sketch of the steps involved. Inferences (i) to (iii) follow from noting that $\mathrm{Q}_1(\mathcal{T}) > 0$ precludes the possibility of degenerate simplices in $\mathcal{T}$ and from recognizing $\mathbf{Q}$ as a composition of continuous maps (both Q and **asc** are continuous). To prove (iv), observe that since $\mathrm{Q}(K) \geq \mathrm{Q}_1(\mathcal{T}) > 0$ for each $K \in \mathcal{T}$, conditions **A**2 and **A**3 imply that the set $\{\lambda \in \mathbb{R}\colon \mathrm{Q}(K(\mathbf{x} + \lambda\mathbf{d}, \mathbf{x}_1, \ldots, \mathbf{x}_d)) \geq \mathrm{Q}_1(\mathcal{T})\}$ is bounded and hence compact (irrespective of the choice of perturbation direction $\mathbf{d}$). Therefore its closed subset $\{\lambda \in \mathbb{R} : q_{i,j}(\lambda, \mathcal{T}) \geq q_{i,j}(0, \mathcal{T})\}$ is compact as well.

Inferences (i) and (iv) of the proposition show that the first vertex update in the dvr algorithm is well defined. To infer the same about subsequent updates, however, we will need to show $\mathrm{Q}_1$ remains positive in the algorithm. We will show this in Theorem 4.2.

**4.2. Main result: Monotonicity of mesh qualities.** Our main result of this section is the following.

THEOREM 4.2 (mesh qualities in dvr). *Given* $\mathcal{T}$*, let* $\mathcal{T}_{k,i}$ *denote the triangulation computed upon updating the position of vertex* $i \in \mathrm{I_R}$ *during the* $k$*th iteration of*

*Algorithm* 1, *and set* $\mathcal{T}_{k+1,0} = \mathcal{T}_{k,m}$. *If* Q *satisfies* **A**1–**A**3 *and if* $\mathrm{Q}_1(\mathcal{T}) > 0$, *then the following hold:*

(i) $\mathcal{T}_{k,i}$ *is well defined for each* $k \in \mathbb{N}$ *and* $i \in \mathrm{I_R}$.

(ii) *For each* $k \in \mathbb{N}$, $i \in \mathrm{I_R}$, *and* $\lambda \in \mathbb{R}$,

$$\mathbf{Q}(\mathcal{T}_{k,i}) \geq \mathbf{Q}(\mathcal{T}_{k,i-1}^{i,\lambda}). \tag{9}$$

(iii) *The sequence of mesh qualities computed in* dvr *is nondecreasing, i.e.,*

$$\mathbf{Q}(\mathcal{T}) \leq \mathbf{Q}(\mathcal{T}_{1,1}) \cdots \leq \mathbf{Q}(\mathcal{T}_{1,m}) \leq \cdots \mathbf{Q}(\mathcal{T}_{k,1}) \cdots \leq \mathbf{Q}(\mathcal{T}_{k,m}) \cdots \tag{10}$$

(iv) *The sequence* $(\mathrm{Q}_1(\mathcal{T}), \mathrm{Q}_1(\mathcal{T}_{1,1}), \ldots, \mathrm{Q}_1(\mathcal{T}_{1,m}), \ldots, \mathrm{Q}_1(\mathcal{T}_{k,1}), \ldots, \mathrm{Q}_1(\mathcal{T}_{k,m}), \ldots)$ *is nondecreasing and hence convergent.*

Before commencing the proof, we highlight a few observations.

(i) Equation (10) reveals that each vertex update in dvr can only improve the mesh quality, just as observed in the examples in sections 3.5 and 3.6.

(ii) By definition in Algorithm 1, updating the position of a vertex in $\mathrm{I_R}$ can only improve the quality of the poorest element in its 1-ring. Then the claim of monotonicity of mesh qualities in (10) seems reasonable. It is, however, not trivial, because improvement in the quality of the poorest element in a 1-ring may be achieved at the expense of reducing the qualities of some of the remaining elements. Specifically, a vertex update alters not just one but multiple components of the mesh quality vector in general. For example, observe in Figure 2 that updating the location of vertex 1 alters not only the contribution of vertex 1 to the mesh quality but may change the contributions of vertices $2, \ldots, 5$ as well. Examining such changes to the mesh quality with vertex updates is the main intent of the proof.

(iii) The assumption $\mathrm{Q}_1(\mathcal{T}) > 0$ requires that the qualities of all perturbable simplices in the initial mesh be positive. As stated therefore, Theorem 4.2 precludes meshes with degenerate/inverted simplices. Nevertheless, we emphasize that $\mathrm{Q}_1(\mathcal{T}) > 0$ is a conservative sufficient condition and that often the conclusions of the theorem hold even when $\mathrm{Q}_1(\mathcal{T}) < 0$. The example discussed in section 5.3 in fact has $\mathrm{Q}_1(\mathcal{T}) < 0$; see Figure 13.

(iv) Though claim (iv) is an immediate consequence of (10), we make a special note of it owing to the significance of $Q_1$ in finite element computations.

(v) Property (10) can be guaranteed even with suboptimal choices of perturbations. We present one possibility in Algorithm 3 in section 4.4. Property (9), however, is a feature of the optimal choice in Algorithm 1.

We commence the proof of Theorem 4.2 by introducing a couple of useful properties of the ordering $\geq$. We omit their proofs, which are straightforward and follow from exercising definition (1).

PROPOSITION 4.3. *Let* $p, q \in \mathbb{N}$, *and let multisets* $\mathbf{u} \in \mathbb{R}^p$ *and* $\mathbf{v}, \mathbf{w} \in \mathbb{R}^q$. *Then*

$$\mathbf{asc}(\mathbf{v}) = \mathbf{asc}(\mathbf{w}) \iff \mathbf{asc}(\mathbf{u} \uplus \mathbf{v}) = \mathbf{asc}(\mathbf{u} \uplus \mathbf{w}), \tag{11a}$$

$$\mathbf{asc}(\mathbf{v}) > \mathbf{asc}(\mathbf{w}) \iff \mathbf{asc}(\mathbf{u} \uplus \mathbf{v}) > \mathbf{asc}(\mathbf{u} \uplus \mathbf{w}). \tag{11b}$$

PROPOSITION 4.4. *Let* $p \in \mathbb{N}$, $\Lambda \subset \mathbb{R}$ *be nonempty, and let* $\mathbf{f} : \Lambda \to \mathbb{R}^p$. *If* $\Lambda$ *is compact and* $\mathbf{f}$ *is continuous, then* $\arg\max_{\lambda \in \Lambda} \mathbf{f}(\lambda) \neq \emptyset$.

In the following intermediate results, we extensively employ the shorthand

(12)
$$\mathbf{Q}(\mathcal{T}^{i,\lambda}) = \mathbf{asc}(\mathrm{A}_i(\mathcal{T}) \uplus \mathrm{B}_{i,\lambda}(\mathcal{T})), \quad \text{where } \begin{cases} \mathrm{A}_i(\mathcal{T}) \triangleq \{q_{i,j}(0,\mathcal{T}) \,:\, j \in \mathrm{I_R} \setminus \mathrm{J_i}\}, \\ \mathrm{B}_{i,\lambda}(\mathcal{T}) \triangleq \{q_{i,j}(\lambda,\mathcal{T}) \,:\, j \in \mathrm{J}_i\}, \\ \mathrm{J}_i \triangleq (\overset{\star}{\mathrm{I}}\,(i,\mathrm{C}) \cap \mathrm{I_R}) \cup \{\mathrm{i}\} \end{cases}$$

for $i \in \mathrm{I_R}$ and $\lambda \in \mathbb{R}$. The motivation behind introducing (12) is that $\mathrm{B}_{i,\lambda}(\mathcal{T})$ collects those components of $\mathbf{Q}(\mathcal{T}^{i,\lambda})$ that can change when vertex $i$ is perturbed by a coordinate $\lambda$, while the multiset $\mathrm{A}_i(\mathcal{T})$ identifies the remaining unaffected components.

LEMMA 4.5. *Given $\mathcal{T}$, $i \in \mathrm{I_R}$, and $\lambda, \eta \in \mathbb{R}$,*

(13)
$$q_{i,i}(\lambda,\mathcal{T}) > q_{i,i}(\eta,\mathcal{T}) \Rightarrow \mathbf{asc}(B_{i,\lambda}(\mathcal{T})) > \mathbf{asc}(B_{i,\eta}(\mathcal{T})) \iff \mathbf{Q}(\mathcal{T}^{i,\lambda}) > \mathbf{Q}(\mathcal{T}^{i,\eta}).$$

*Proof.* The equivalence in (13) is a consequence of Proposition 4.3 and (12):

$$\begin{aligned} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T})) > \mathbf{asc}(\mathrm{B}_{i,\eta}(\mathcal{T})) &\iff \mathbf{asc}(\mathrm{A}_i(\mathcal{T}) \uplus \mathrm{B}_{i,\lambda}(\mathcal{T})) > \mathbf{asc}(\mathrm{A}_i(\mathcal{T}) \uplus \mathrm{B}_{i,\eta}(\mathcal{T})) \\ &\iff \mathbf{Q}(\mathcal{T}^{i,\lambda}) > \mathbf{Q}(\mathcal{T}^{i,\eta}). \end{aligned}$$

Let us prove the first implication claimed in (13). Without loss of generality, let us assume that $\eta = 0$, $\mathrm{J}_i \setminus \{i\} = \{1, 2, \ldots, p\}$ and

$$q_{i,1}(0) \le q_{i,2}(0) \le \cdots \le q_{i,p}(0). \tag{14}$$

Noting that the case $p = 0$ is trivial, we additionally suppose that $p > 0$ in the following. Denote $\mathbf{u} = \mathbf{asc}(\mathrm{B}_{i,0}(\mathcal{T}))$ and $\mathbf{w} = \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T}))$. For $\zeta \in \mathbb{R}$, consider the minimum among the qualities of simplices in $\mathcal{T}^{i,\zeta}$ that belong to the 1-rings of both vertices $i$ and $j$, namely

$$q^{ij}(\zeta,\mathcal{T}) \triangleq \min\{\mathrm{Q}(K) \,:\, K \in \overset{\star}{\mathrm{e}}\,(i,\mathcal{T}^{i,\zeta}) \cap \overset{\star}{\mathrm{e}}\,(j,\mathcal{T}^{i,\zeta})\} \quad \text{for } 1 \le j \le p. \tag{15}$$

The function $\zeta \mapsto q^{ij}(\zeta,\mathcal{T})$ in (15) is well defined, because $\overset{\star}{\mathrm{e}}\,(i,\mathcal{T}^{i,\zeta}) \cap \overset{\star}{\mathrm{e}}\,(j,\mathcal{T}^{i,\zeta})$ necessarily contains at least one simplex. Observe that

$$q_{i,j}(\zeta,\mathcal{T}) = \min\{q_{i,j}(0,\mathcal{T}), q^{ij}(\zeta,\mathcal{T})\} \quad \text{for } 1 \le j \le p. \tag{16}$$

Using $q^{ij}(\lambda,\mathcal{T}) \ge q_{i,i}(\lambda,\mathcal{T}) > q_{i,i}(0,\mathcal{T})$ and (16), we have the following:

(17a)
If $q_{i,j}(0,\mathcal{T}) \le q_{i,i}(0,\mathcal{T})$, then $q_{i,j}(\lambda,\mathcal{T}) = \min\{\underbrace{q_{i,j}(0,\mathcal{T})}_{\le q_{i,i}(0,\mathcal{T})}, \underbrace{q^{ij}(\lambda,\mathcal{T})}_{> q_{i,i}(0,\mathcal{T})}\} = q_{i,j}(0,\mathcal{T})$.

(17b)
If $q_{i,j}(0,\mathcal{T}) > q_{i,i}(0,\mathcal{T})$, then $q_{i,j}(\lambda,\mathcal{T}) = \min\{\underbrace{q_{i,j}(0,\mathcal{T})}_{> q_{i,i}(0,\mathcal{T})}, \underbrace{q^{ij}(\lambda,\mathcal{T})}_{> q_{i,i}(0,\mathcal{T})}\} > q_{i,i}(0,\mathcal{T})$.

In the following, we repeatedly use (14), (17), and the assumption $q_{i,i}(\lambda,\mathcal{T}) > q_{i,i}(0,\mathcal{T})$ to show that $\mathbf{w} > \mathbf{u}$.

First, suppose that $q_{i,i}(0,\mathcal{T}) < q_{i,1}(0,\mathcal{T})$. Then

$$\left.\begin{aligned} u_1 &= q_{i,i}(0,\mathcal{T}) \\ w_1 &= \min\{q_{i,j}(\lambda,\mathcal{T}) \,:\, j \in \mathrm{J}_i\} > q_{i,i}(0,\mathcal{T}) \end{aligned}\right\} \Rightarrow w_1 > u_1 \Rightarrow \mathbf{w} > \mathbf{u}. \tag{18}$$

Next, suppose that $q_{i,p}(0,\mathcal{T}) \le q_{i,i}(0,\mathcal{T})$. Then

$$\left.\begin{aligned} w_j = u_j = q_{i,j}(0,\mathcal{T}) \text{ for } 1 \le j \le p \\ w_{p+1} = q_{i,i}(\lambda,\mathcal{T}) > q_{i,i}(0,\mathcal{T}) = u_{p+1} \end{aligned}\right\} \Rightarrow \mathbf{w} > \mathbf{u}. \tag{19}$$

Otherwise, we can find index $\ell$ such that $1 \le \ell < p$ and $q_{i,\ell}(0,\mathcal{T}) \le q_{i,i}(0,\mathcal{T}) < q_{i,\ell+1}(0,\mathcal{T})$. Then

$$\left.\begin{aligned} w_j = u_j = q_{i,j}(0,\mathcal{T}) \text{ for } 1 \le j \le \ell \\ w_{\ell+1} = \min\{q_{i,i}(\lambda,\mathcal{T}), q_{i,\ell+1}(\lambda,\mathcal{T}), \ldots, q_{i,p}(\lambda,\mathcal{T})\} > q_{i,i}(0,\mathcal{T}) = u_{\ell+1} \end{aligned}\right\} \Rightarrow \mathbf{w} > \mathbf{u}. \tag{20}$$

Equations (18)–(20) prove the implication claimed in (13). ∎

PROPOSITION 4.6. *Given* $\mathcal{T}$ *and* $i \in \mathrm{I_R}$,

$$\arg\max_{\lambda\in\mathbb{R}} \mathbf{asc}(B_{i,\lambda}(\mathcal{T})) = \arg\max_{\lambda\in\mathbb{R}} \mathbf{Q}(\mathcal{T}^{i,\lambda}) \subseteq \arg\max_{\lambda\in\mathbb{R}} q_{i,i}(\lambda,\mathcal{T}). \tag{21}$$

*Proof.* From Proposition 4.3 and Lemma 4.5, we have

$$\mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T})) \ge \mathbf{asc}(\mathrm{B}_{i,\eta}(\mathcal{T})) \iff \mathbf{Q}(\mathcal{T}^{i,\lambda}) \ge \mathbf{Q}(\mathcal{T}^{i,\eta}),$$

which implies that

$$\arg\max_{\lambda\in\mathbb{R}} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T})) = \arg\max_{\lambda\in\mathbb{R}} \mathbf{Q}(\mathcal{T}^{i,\lambda}). \tag{22}$$

Moreover, from the contrapositive of (13), namely

$$\lambda,\eta \in \mathbb{R},\ \mathbf{Q}(\mathcal{T}^{i,\lambda}) \ge \mathbf{Q}(\mathcal{T}^{i,\eta}) \Rightarrow q_{i,i}(\lambda,\mathcal{T}) \ge q_{i,i}(\eta,\mathcal{T}),$$

we get that

$$\arg\max_{\lambda\in\mathbb{R}} \mathbf{Q}(\mathcal{T}^{i,\lambda}) \subseteq \arg\max_{\lambda\in\mathbb{R}} q_{i,i}(\lambda). \tag{23}$$

Equations (22) and (23) prove (21). ∎

PROPOSITION 4.7. *Given* $\mathcal{T}$ *and* $i \in \mathrm{I_R}$ *and* Q *satisfying* **A**1–**A**3,

$$\mathrm{Q}_1(\mathcal{T}) > 0 \Rightarrow \arg\max_{\lambda\in\mathbb{R}} \mathbf{Q}(\mathcal{T}^{i,\lambda}) \ne \emptyset. \tag{24}$$

*Proof.* Let $j \in \mathrm{J}_i$ be an index such that $q_{i,j}(0,\mathcal{T})$ equals the minimum among the elements in $\mathrm{B}_{i,0}(\mathcal{T})$. Let $\Lambda \triangleq \{\lambda : q_{i,j}(\lambda,\mathcal{T}) \ge q_{i,j}(0,\mathcal{T})\}$. From (21) and the choice of $j$, we know that

$$\arg\max_{\lambda\in\mathbb{R}} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T})) \subseteq \arg\max_{\lambda\in\mathbb{R}} q_{i,i}(\lambda) \subseteq \Lambda,$$

which implies that

$$\arg\max_{\lambda\in\mathbb{R}} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T})) = \arg\max_{\lambda\in\Lambda} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T})). \tag{25}$$

Using $\mathrm{Q}_1(\mathcal{T}) > 0$ in Proposition 4.1 shows that $\Lambda$ is compact. Additionally, $\Lambda$ is nonempty, because $0 \in \Lambda$. Hence we conclude from Proposition 4.4 that

$$\arg\max_{\lambda\in\Lambda} \mathbf{asc}(\mathrm{B}_{i,\lambda}) \ne \emptyset.$$

Equation (24) now follows from (21) and (25). ∎

*Proof of Theorem* 4.2. Fix $k \in \mathbb{N}$ and $i \in \mathrm{I_R}$. Suppose we can show that

$$\begin{cases} \mathcal{T}_{k,i-1} \text{ is well defined,} \\ \mathrm{Q}_1(\mathcal{T}_{k,i-1}) > 0 \end{cases} \implies \begin{cases} \mathcal{T}_{k,i} \text{ is well defined,} \\ \mathbf{Q}(\mathcal{T}_{k,i}) \geq \mathbf{Q}(\mathcal{T}_{k,i-1}^{i,\lambda}) \ \forall \lambda \in \mathbb{R}, \\ \mathrm{Q}_1(\mathcal{T}_{k,i}) > 0. \end{cases} \tag{26}$$

Then, since we have assumed $\mathrm{Q}_1(\mathcal{T}_{1,0}) = \mathrm{Q}_1(\mathcal{T}) > 0$, claims (i) and (ii) in the statement will follow from (26) by induction. Choosing $\lambda = 0$ in (10) shows that $\mathbf{Q}(\mathcal{T}_{k,i}) \geq \mathbf{Q}(\mathcal{T}_{k,i-1})$ for each $k \in \mathbb{N}$ and $i \in \mathrm{I_R}$. Claim (iii) then follows from choosing indices $k$ and $i$. Claim (iv) is an immediate consequence of (10).

Let us assume that $\mathcal{T}_{k,i-1}, \mathrm{Q}_1(\mathcal{T}_{k,i-1}) > 0$ and prove (26). Let

$$\mathrm{L} \triangleq \arg\max_{\lambda \in \mathbb{R}} q_{i,i}(\lambda, \mathcal{T}_{k,i-1}) \text{ and } \mathrm{S} \triangleq \arg\max_{\lambda \in \mathbb{R}} \mathbf{Q}(\mathcal{T}_{k,i-1}^{i,\lambda}).$$

From Proposition 4.7, we know that

$$\emptyset \neq \arg\max_{\lambda \in \mathbb{R}} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T}_{k,i-1})) = \mathrm{S} \subseteq \mathrm{L}. \tag{27}$$

By definition in Algorithm 1, $\mathcal{T}_{k,i} = \mathcal{T}_{k,i-1}^{i,\lambda^{\mathrm{opt}}}$, where

$$\lambda^{\mathrm{opt}} \in \arg\max_{\lambda \in \mathrm{L}} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T}_{k,i-1})).$$

Since (27) implies $\arg\max_{\lambda \in \mathrm{L}} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T}_{k,i-1})) = \arg\max_{\lambda \in \mathbb{R}} \mathbf{asc}(\mathrm{B}_{i,\lambda}(\mathcal{T}_{k,i-1})) \neq \emptyset$, we know that $\lambda^{\mathrm{opt}}$ exists and therefore $\mathcal{T}_{k,i}$ is well defined. Additionally, (27) shows that $\lambda^{\mathrm{opt}} \in \mathrm{S}$, from where we conclude that

$$\mathbf{Q}(\mathcal{T}_{k,i}) = \mathbf{Q}(\mathcal{T}_{k,i-1}^{i,\lambda^{\mathrm{opt}}}) \geq \mathbf{Q}(\mathcal{T}_{k,i-1}^{i,\lambda}) \ \forall \, \lambda \in \mathbb{R}. \tag{28}$$

Choosing $\lambda = 0$ in (28) yields $\mathbf{Q}(\mathcal{T}_{k,i}) \geq \mathbf{Q}(\mathcal{T}_{k,i-1})$, which in turn implies that $\mathrm{Q}_1(\mathcal{T}_{k,i}) \geq \mathrm{Q}_1(\mathcal{T}_{k,i-1}) > 0$. Equation (26) follows. ∎

*Remark* 4.8. Following the steps in the proof of Theorem 4.2 reveals that the result is applicable in a more general context in which vertex updates need not be restricted to prescribed directions; cf. [22, 20]. To this end, let $q_{i,j}(\boldsymbol{\lambda}, \mathcal{T})$ be defined analogous to (3) by replacing $\lambda \mathbf{d}_{k,i}$ with $\boldsymbol{\lambda} \in \mathbb{R}^d$. Consider any well defined vertex update algorithm $\mathrm{V}_i \mapsto \mathrm{V}_i + \boldsymbol{\lambda}^{\mathrm{opt}}$ with $\boldsymbol{\lambda}^{\mathrm{opt}} \in \mathbb{R}^d$ satisfying the condition $q_{i,i}(\boldsymbol{\lambda}^{\mathrm{opt}}, \mathcal{T}) > q_{i,i}(\boldsymbol{\eta}, \mathcal{T}) \, \forall \, \boldsymbol{\eta} \neq \boldsymbol{\lambda}^{\mathrm{opt}}$. Then the sequence of mesh iterates computed by such an algorithm also enjoy properties (ii)–(iv) of Theorem 4.2. In particular, properties (ii)–(iv) are not specific to the dvr algorithm, which assumes a prescribed relaxation direction. A similar remark applies also to Theorem 4.10 concerning suboptimal perturbations.

**4.3. Uniqueness of optimal perturbation coordinates.** In general, $\lambda \mapsto q_{i,i}(\lambda, \mathcal{T})$ may not have a unique maximizer. More crucially, some of the maximizers of $q_{i,i}(\lambda, \mathcal{T})$ may not be maximizers of $\mathbf{Q}(\mathcal{T}^{i,\lambda})$. Indeed, depending on the choice of Q, the inclusion in (21) can be strict. When $\lambda \mapsto q_{i,i}(\lambda, \mathcal{T})$ has multiple maximizers, choosing $\lambda^{\mathrm{opt}}$ arbitrarily as one of them may in fact result in violating property (10) or even yield a mesh with a lower quality than the unperturbed one. This distinction between maximizers of $\lambda \mapsto \mathbf{Q}(\mathcal{T}^{i,\lambda})$ and $\lambda \mapsto q_{i,i}(\lambda, \mathcal{T})$, despite the inclusion shown by (21), is precisely the reason behind the check in step 9 in Algorithm 1. The check identifies the coordinate(s) that best improve the mesh quality.

With additional restrictions on Q, it is possible to ensure that $\lambda \mapsto q_{i,i}(\lambda, \mathcal{T})$ has a unique maximizer; see also [1]. Then, from (21) and (24), we can conclude that the maximizer of $\lambda \mapsto q_{i,i}(\lambda, \mathcal{T})$ is the unique maximizer of $\lambda \mapsto \mathbf{Q}(\mathcal{T}^{i,\lambda})$ as well. To facilitate stating these conditions, some additional notation is helpful. Recalling that $K(\mathbf{x}_0, \ldots, \mathbf{x}_d)$ denotes the simplex with vertices $\mathbf{x}_0, \ldots, \mathbf{x}_d \in \mathbb{R}^d$, for $\alpha \in \mathbb{R}$, define

$$S_\alpha^{\mathrm{ge}}(\mathbf{x}_1, \ldots, \mathbf{x}_d) \triangleq \{\mathbf{x} \in \mathbb{R}^d : \mathrm{Q}(K(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_d)) \geq \alpha\}, \tag{29a}$$

$$S_\alpha^{\mathrm{g}}(\mathbf{x}_1, \ldots, \mathbf{x}_d) \triangleq \{\mathbf{x} \in \mathbb{R}^d : \mathrm{Q}(K(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_d)) > \alpha\}. \tag{29b}$$

We consider the following restrictions on Q:

**A**4. *Boundedness:* Q is bounded from above. Without loss of generality, $\mathrm{Q} \leq 1$.

**A**5. *Unique global maximizer:* $S_1^{\mathrm{ge}}(\mathbf{x}_1, \ldots, \mathbf{x}_d)$ is a singleton.

**A**6. *Strict convexity of level sets:* For $0 < \alpha < 1$, $S_\alpha^{\mathrm{ge}}(\mathbf{x}_1, \ldots, \mathbf{x}_d)$ is a strictly convex set in $\mathbb{R}^d$ and $\mathrm{int}(S_\alpha^{\mathrm{ge}}(\mathbf{x}_1, \ldots, \mathbf{x}_d)) = S_\alpha^{\mathrm{g}}(\mathbf{x}_1, \ldots, \mathbf{x}_d)$.

PROPOSITION 4.9. *Let $\mathcal{T}$ be given and $i \in \mathrm{I_R}$. If $\mathrm{Q}_1(\mathcal{T}) > 0$ and* Q *satisfies* **A**1–**A**6*, then $\lambda \mapsto q_{i,i}(\lambda, \mathcal{T})$ has a unique maximizer. Consequently, $\lambda \mapsto \mathbf{Q}(\mathcal{T}^{i,\lambda})$ has a unique maximizer and* $\arg\max_{\lambda \in \mathbb{R}} q_{i,i}(\lambda, \mathcal{T}) = \arg\max_{\lambda \in \mathbb{R}} \mathbf{Q}(\mathcal{T}^{i,\lambda})$.

*Proof.* Let us show that $\mathrm{L} \triangleq \arg\max_{\lambda \in \mathbb{R}} q_{i,i}(\lambda, \mathcal{T})$ is a singleton. The subsequent claims then follow from (21).

Noting assumptions **A**1–**A**3 and $\mathrm{Q}_1(\mathcal{T}) > 0$ in Proposition 4.7 shows that L is nonempty. Let $\alpha = \max_{\lambda \in \mathbb{R}} q_{i,i}(\lambda, \mathcal{T})$. Assumption **A**4 and $q_{i,i}(0, \mathcal{T}) \geq \mathrm{Q}_1(\mathcal{T}) > 0$ imply that $0 < \alpha \leq 1$. If $\alpha = 1$, the L is a singleton by virtue of assumption **A**5.

It remains to consider the possibility $0 < \alpha < 1$. For $\mathbf{x} \in \mathbb{R}^d$, let $\mathcal{T}^{i,\mathbf{x}}$ denote the triangulation obtained by positioning vertex $i$ in $\mathcal{T}$ at $\mathbf{x}$. Denote $\{K_{i_j}^{\mathbf{x}}\}_{j=1}^p = \overset{\star}{\mathrm{e}}(i; \mathcal{T}^{i,\mathbf{x}})$. From assumptions **A**3 and **A**6 and $\alpha < 1$, we know that $\rho_{\alpha,j} \triangleq \{\mathbf{x} \in \mathbb{R}^d : \mathrm{Q}(K_{i_j}^{\mathbf{x}}) \geq \alpha\}$ is strictly convex and bounded for each $1 \leq j \leq p$. Consequently, each set $\Lambda_{\alpha,j} \triangleq \{\lambda \in \mathbb{R} : \mathrm{V}_i + \lambda\, \mathbf{d}_i \in \rho_{\alpha,j}\}$ is a compact interval in $\mathbb{R}$, where $\mathbf{d}_i$ is the relaxation direction prescribed for $\mathrm{V}_i$.

Now notice that if $\lambda \in \mathrm{int}(\Lambda_{\alpha,j})$, then $\mathbf{x}_\lambda \triangleq \mathrm{V}_i + \lambda\, \mathbf{d}_i \in \mathrm{int}(\rho_{\alpha,j})$ and in turn $\mathrm{Q}(K_{i_j}^{\mathbf{x}_\lambda}) > \alpha$ by virtue of assumption **A**6. The choice of $\alpha$ therefore implies that $\mathrm{int}(\Lambda_{\alpha,j}) = \emptyset$ for at least one index $j \in \{1, \ldots, p\}$. Without loss of generality, assume that $\mathrm{int}(\Lambda_{\alpha,1}) = \emptyset$. Since we also know that $\Lambda_{\alpha,1}$ is a compact interval, we conclude that $\Lambda_{\alpha,1}$ is a singleton. The choice of $\alpha$ implies that $\mathrm{L} = \cap_{j=1}^p \Lambda_{\alpha,j}$, which shows that $\mathrm{L} \subseteq \Lambda_{\alpha,1}$. Since $\mathrm{L} \neq \emptyset$ and $\Lambda_{\alpha,1}$ is a singleton, we conclude that $\mathrm{L}(= \Lambda_{1,\alpha})$ is a singleton, i.e., $\lambda \mapsto q_{i,i}(\lambda, \mathcal{T})$ has a unique maximizer. ∎

It is an easy exercise to verify that the mean ratio metric (4) satisfies **A**1–**A**6. In particular, $S_\alpha^{\mathrm{ge}}$ is the region bounded by a circle (sphere) for $d = 2$ ($d = 3$) for $\alpha > 0$; see Figure 3. Conditions **A**1–**A**6 will have to be verified independently for other metrics; cf. [34, 42].

**4.4. Monotonic mesh qualities with suboptimal perturbations.** Whether unique or not, computing optimal perturbations requires finding roots of nonlinear scalar equations. In the case of the mean ratio metric, we argued in section 3.2 that these equations are polynomials and are hence easily resolvable. In general, however, it may only be possible to compute these coordinates approximately, which are therefore suboptimal. Alternatively, it may not be feasible to identify all the roots of the relevant nonlinear equations, or suboptimal perturbations may be efficiently computable, for instance, by performing an incremental sweep using small coordinate increments along the prescribed relaxation direction to sample the objective. Algorithm 3 below

outlines a modification of Algorithm 1 using possibly suboptimal choices for vertex perturbations while exploiting Lemma 4.5 to guarantee that the sequence of resulting mesh qualities are nondecreasing.

**Algorithm 3:** Suboptimal directional vertex relaxation

**Input:**
$\mathcal{T} = (\mathrm{V}, \mathrm{I}, \mathrm{C})$ : Input triangulation
$\mathrm{I_R}$ : Ordered set of vertices to relax
$\{\mathbf{d}_{k,i}\}_{i\in \mathrm{I_R}, \mathrm{k}\in\mathbb{N}}$ : Relaxation directions
$N_\mathrm{R}$ : Number of relaxation iterations

```
for iter = 1 to N_R do
    for i = 1 to m do
        if found λ ∈ ℝ such that q_{i,i}(λ, T) > q_{i,i}(0, T) then
            Update: V_i ← V_i + λ d_{k,i}
        end
    end
end
return (V, I, C)
```

THEOREM 4.10 (mesh qualities with suboptimal dvr). *Given $\mathcal{T}$, let $\mathcal{T}_{k,i}$ denote the triangulation computed upon updating the position of vertex $i \in \mathrm{I_R}$ during the kth iteration of Algorithm* 3*, and set $\mathcal{T}_{k+1,0} = \mathcal{T}_{k,m}$. Then*

$$\mathbf{Q}(\mathcal{T}) \le \mathbf{Q}(\mathcal{T}_{1,1}) \cdots \le \mathbf{Q}(\mathcal{T}_{1,m}) \le \cdots \mathbf{Q}(\mathcal{T}_{k,1}) \cdots \le \mathbf{Q}(\mathcal{T}_{k,m}) \cdots . \tag{30}$$

*Consequently, the sequence*

$$\mathrm{Q}_1(\mathcal{T}), \mathrm{Q}_1(\mathcal{T}_{1,1}), \ldots, \mathrm{Q}_1(\mathcal{T}_{1,m}), \ldots, \mathrm{Q}_1(\mathcal{T}_{k,1}), \ldots, \mathrm{Q}_1(\mathcal{T}_{k,m}), \ldots$$

*is nondecreasing and hence convergent.*

The assumptions on the quality metric and the requirement $\mathrm{Q}_1(\mathcal{T}) > 0$ in Theorem 4.2 are conspicuously absent in Theorem 4.10. This relief, however, comes at the expense of property (9) and, more crucially, an indeterminate search for a perturbation for each vertex update (step 4 of Algorithm 3). We expect that strategies for such a search will inevitably entail some restrictions on Q and on the input mesh. Owing to the algebraic simplifications possible using the mean ratio metric, in all our numerical experiments, we compute "correct" vertex perturbations as roots of polynomials using the GSL library [26] with algorithmic tolerances set to default values.

**4.5. Termination criteria.** The monotonicity of quality vectors computed by Algorithms 1 and 3, and expressed in (10) and (30), respectively, help in identifying a termination criterion for relaxation iterations. In particular, Theorems 4.2 and 4.10 guarantee that the first component $\mathrm{Q}_1$ of the mesh quality converges as $N_\mathrm{R} \to \infty$. Therefore, instead of relying on a fixed number of iterations, the algorithms can be terminated once $\mathrm{Q}_1(\mathcal{T}_{k,m}) - \mathrm{Q}_1(\mathcal{T}_{k,0}) < \varepsilon$, where $\varepsilon > 0$ is a given tolerance. Such a criterion is meaningful when improving meshes for use in finite element computations.

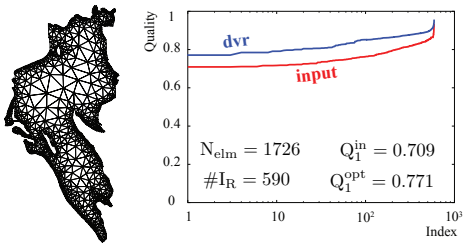
Admittedly, termination criteria based on a fixed number of iterations or on convergence of the component $\mathrm{Q}_1$ are both heuristic. Neither guarantee convergence of meshes or of mesh qualities. We caution that despite the monotonic nature of mesh qualities, we cannot infer their convergence (owing to the definition of the ordering (2)). Nevertheless, it may be possible to identify convergent subsequences instead.

**5. Demonstrative applications of dvr.** Next, we compile a list of examples using dvr to improve triangle/tetrahedral meshes followed by a set of representative applications where dvr will be useful. The intent behind the examples in section 5.1 is quite different from that behind the applications presented subsequently. In the former, we are interested mainly in examining the magnitude of mesh improvement possible with dvr. To this end, we pick meshes from open-source repositories and meshes generated using open-source mesh generators, and provide these as inputs to the dvr algorithm. Similar (and often the same) meshes have been used for testing other mesh improvement algorithms in the literature [22, 32], which will be helpful for readers interested in comparing the performance of dvr with some of these alternative algorithms.

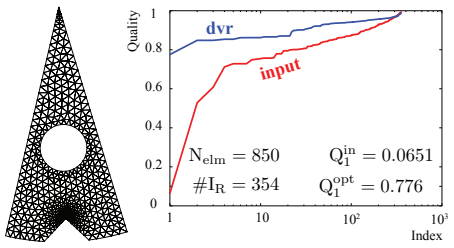
The applications discussed in sections 5.2–5.4 serve to exemplify the role of dvr in simulating moving boundary problems. In the interest of keeping the discussions brief and self-contained, we retain just the quintessential details concerning the evolution of the boundary and intentionally omit pondering over how these evolutions are actually computed in practice. The meshes used in these examples are quite simple—they have a relatively small number of vertices/elements and have not been refined based on any specific criterion. Our singular goal in these examples is to demonstrate that dvr is useful in maintaining good element qualities by optimizing deformed meshes realized during the course of simulating moving boundary problems. An application of dvr to perturb nonconforming meshes to conform to cracks propagating in a linearly elastic brittle solid can be found in [46].

**5.1. Mesh improvement with dvr.** Figures 8 and 9 show a collection of examples using dvr to improve the qualities of triangle and tetrahedral meshes. In all these examples, vertices lying on the boundary are left undisturbed while the remaining set of interior vertices enumerated in ascending order of the indices assigned by the mesh generator constitutes the list $\mathrm{I_R}$. The relaxation direction for each vertex in each dvr iteration (i.e., $\mathbf{d}_{k,i}$ for $i \in \mathrm{I_R}, \mathrm{k} \in \mathbb{N}$) is generated at random. Alongside the images of the meshes themselves, the figures show plots of the quality vectors of the input meshes and the final optimized meshes computed after a fixed number of iterations. A visual inspection revealed that the components of the quality vectors converged within a couple of dozen iterations in all the examples. For definiteness, we have used 25 iterations for the triangles meshes shown in Figures 8a and 8b and 40 iterations for all the remaining tetrahedral meshes. The termination criteria mentioned in section 4.5 could have been used as well. While examining plots of the quality vectors, note the logarithmic scale used for the horizontal axes. This has been done to help visualize the improvement in the first few components of the quality vector, which are certainly more significant than the later ones. In each plot, we note the first component of the quality vectors (the poorest element quality) for the input and optimized meshes, denoted by $\mathrm{Q}_1^{\mathrm{in}}$ and $\mathrm{Q}_1^{\mathrm{opt}}$, respectively.

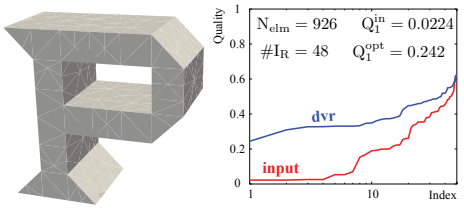
Since the examples are mostly self-explanatory, we mention just a few noteworthy details. We observe that the poorest element quality is consistently improved in all the examples and typically by a factor of at least two. The obvious question is then about what limits improving these qualities any further. A few different factors are apparent from our numerical experiments. Vertices in $\mathrm{I_R}$ having a large number of elements in their 1-rings (i.e., a high valence) are severely restricted in their motion irrespective of the choice of relaxation directions. In some of the tetrahedral meshes shown, the valence even exceeds 70. It is difficult to envision improving element qualities around such vertices without altering the mesh connectivity in some way—using face/edge



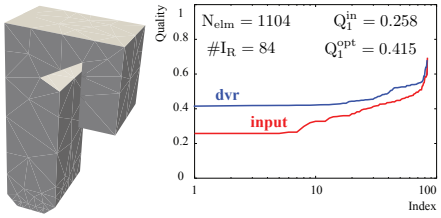

(a) Input mesh: *"superior"* from [53]



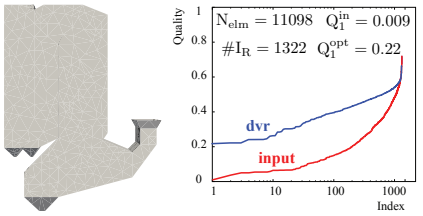

(b) Input mesh from [43]



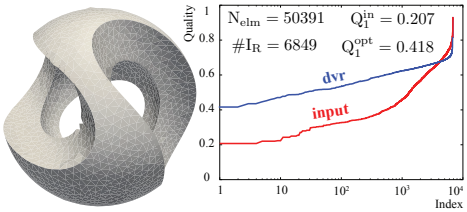

(c) Input mesh: *"P"* from [52]




(d) Input mesh: *"tfire"* from [52]




(e) Input mesh: *"tire"* from [52]

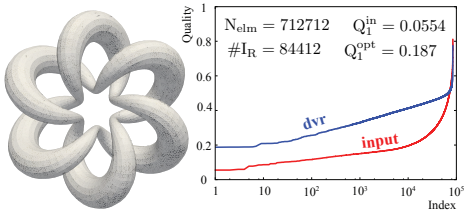



(f) Input mesh: *"sculpt10kv"* from [52]

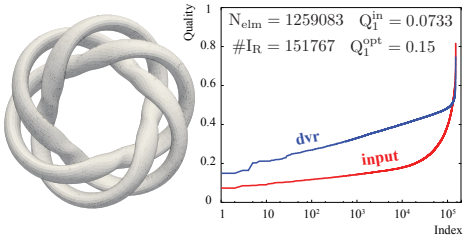
FIG. 8. *Examples demonstrating the mesh improvement possible with* dvr *in triangle and tetrahedral meshes. In each example, we have noted the poorest element qualities in the input and optimized meshes, the number of elements in the mesh, the number of vertices relaxed, and the source of the input mesh. Notice the logarithmic scale used for the horizontal axes in the plots of the quality vectors. This has been done to facilitate a closer inspection of the elements with poor qualities in the meshes. The optimized qualities correspond to meshes computed at the end of* 25 *and* 40 *iterations for the triangle and tetrahedral meshes, respectively.*

collapse or vertex insertion/deletion operations, for instance. Similarly, vertices in $I_R$ whose 1-rings contain a significant number of elements having edges/faces lying on the boundary are also severely restricted during relaxation iterations. Furthermore, a poorly shaped triangle on the boundary clearly limits the quality of the tetrahedron to which it belongs. Quite often, we find that the poorest element quality in the optimized mesh is in fact realized over elements having an edge or a face lying on the boundary. These observations reinforce the need for relaxing vertices lying on the boundary of the mesh as well. Finally, we mention that the examples in Figures 8 and 9 provide useful evidence that the paradigm of relaxing vertices one by one adopted in dvr, though inherently limited in scope when compared to relaxing multiple vertices simultaneously, is nevertheless able to effect considerable mesh improvement.

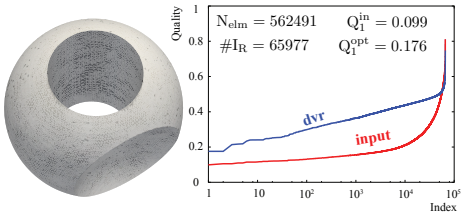
Next, we inspect the execution times recorded while optimizing the tetrahedral meshes in Figures 8 and 9. Computed as an average over 40 iterations, the run times per dvr iteration are plotted in Figure 10. Each data point in the plot corresponds to one of the tetrahedral meshes. We have used the algorithm described in [45],



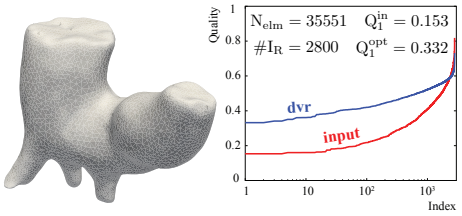

(a) Input mesh generated using Tetgen [55]



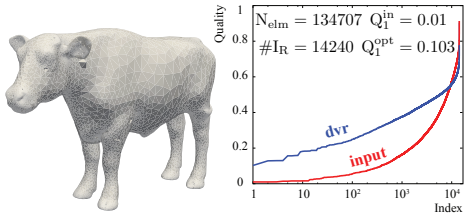

(b) Input mesh generated using Tetgen [55]



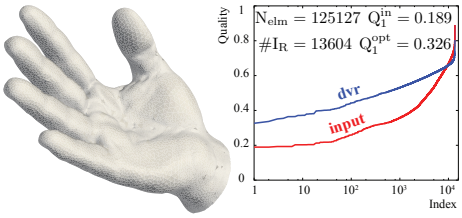

(c) Input mesh generated using Tetgen [55]



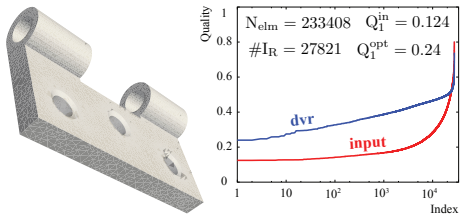

(d) Input mesh: *"aorta"* from [57]



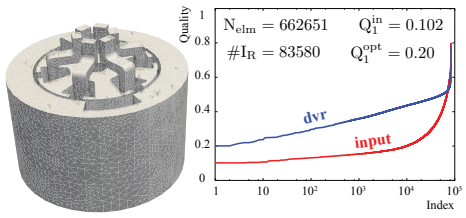

(e) Input mesh: *"3D cow"* from [57]



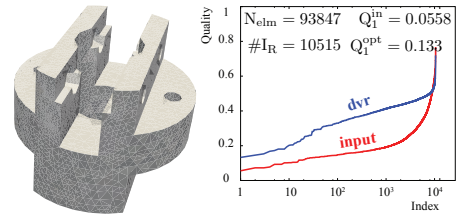

(f) Input mesh: *"Hand man"* from [57]



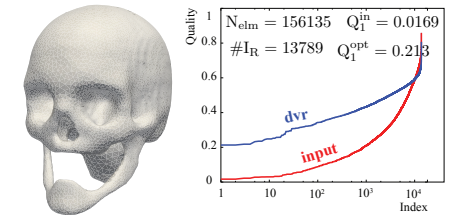

(g) Input mesh generated using Tetgen [55]




(h) Input mesh generated using Tetgen [55]




(i) Input mesh generated using Tetgen [55]




(j) Input mesh: *"Skull"* from [57]

FIG. 9. *More examples illustrating the improvement in tetrahedral mesh qualities possible with* dvr.

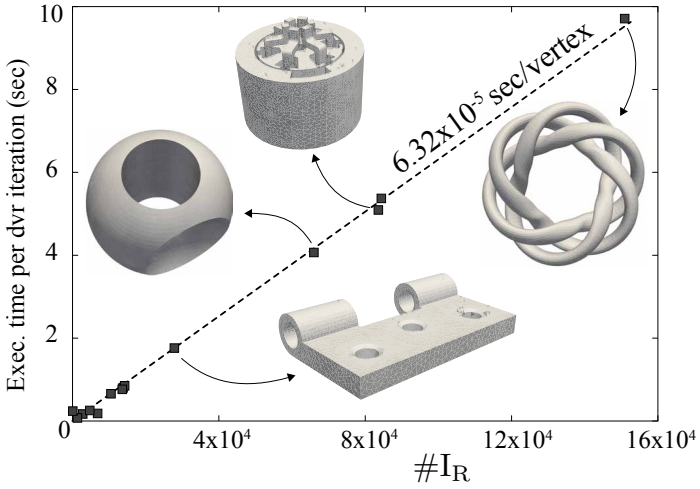



FIG. 10. *Execution time per* dvr *iteration for optimizing the tetrahedral meshes shown in Figures* 8 *and* 9 *while using the algorithm described in* [45] *to compute optimal perturbations (instead of Algorithm* 2*). The times reported are computed as an average over* 40 *iterations. Our serialized implementation of the* dvr *algorithm was run on a Mac Pro with a* 3.5*GHz Intel Xeon E*5 *processor and using the* gcc *compiler (version* 5.4.0*, -O*2 *optimization flag). Notice the linear scaling of the execution time with the number of vertices being relaxed.*

henceforth referred to as Algorithm 2′, in place of Algorithm 2, to compute vertex updates while recording these run times. Similar to Algorithm 2, Algorithm 2′ also computes optimal vertex perturbations *correctly*, at least up to the tolerances set (by default) while resolving polynomials in the GNU Scientific Library [26]. Both Algorithms 2 and 2′ compute identical meshes; the latter simply avoids many of the extraneous computations inherent in the former by exploiting properties of the quality metric discussed in section 4. The main point to be noted from Figure 10 is the linear scaling between the execution time and the number of vertices being relaxed. The plot reflects that in our serialized implementation, each vertex update in a tetrahedral mesh requires about 63.2 microseconds on average; see the caption of the figure for more details. Algorithm 2 also yields a linear scaling for the run time with the number of relaxed vertices but with the caveat that the time taken per vertex update when using it is much larger when compared to Algorithm 2′. We refer the reader to [45] for a detailed comparison of the two algorithms. Since the flop count and hence the time required to compute a vertex update are necessarily data-dependent with both Algorithms 2 and 2′, the times reported in Figure 10 should be understood as being representative averages only.

We conclude our discussion by briefly examining the influence of the vertex ordering chosen for $\mathrm{I_R}$ on the final mesh qualities. In Figure 11, we revisit the examples shown in Figures 8a and 9f and inspect the range of mesh qualities observed in 100 different trials, with the vertices in $\mathrm{I_R}$ randomly shuffled at the beginning of each trial. All other details of the mesh optimization procedure remain the same as before. Both results in Figure 11 suggest that the effect of vertex ordering in $\mathrm{I_R}$ on the mesh quality is small. Nevertheless, it is possible that the influence may be more significant in other examples, as well as during early iterations. Without quantitative estimates for the mesh quality, we are unable to draw general conclusions at this time. A couple of remarks are, however, worth mentioning. First, as an algorithm that relaxes vertices

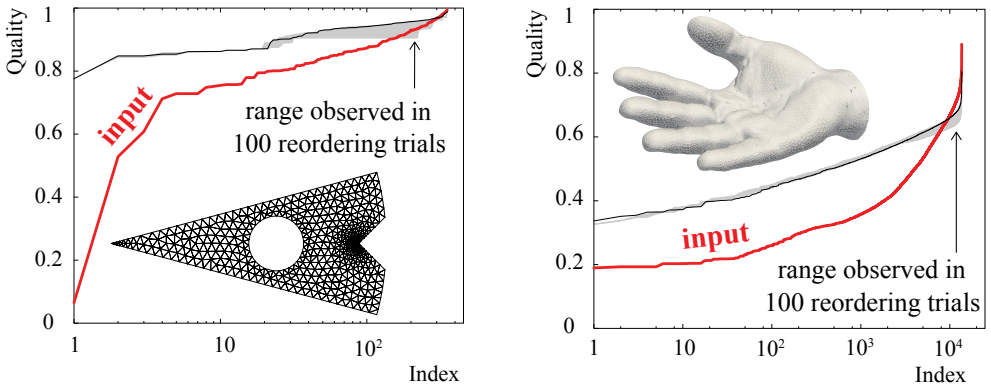


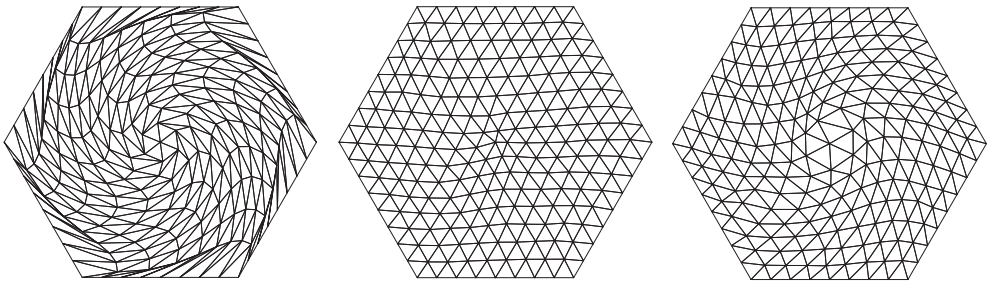

FIG. 11. *Inspecting the influence of vertex ordering in* $\mathrm{I_R}$ *on the final mesh quality. Figures* (a) *and* (b) *show the range of mesh qualities observed when using* 100 *different (random) orderings for vertices in* $\mathrm{I_R}$ *for the meshes previously shown in Figures* 8a *and* 9f*, respectively. The mesh qualities from the previous figures are shown by solid black lines, while the region enveloping the range of mesh qualities observed with different vertex orderings is shaded in gray. In both experiments, the influence of vertex ordering appears to be inconsequential.*

one by one, consideration of vertex orderings in dvr is only to be expected. In contrast, this is not the case in algorithms that permit relaxing multiple (or all) vertices simultaneously. Second, enforcing a prescribed vertex ordering in dvr is not natural in parallel implementations of the algorithm, since vertex updates across multiple processors cannot be synchronized without compromising efficiency.

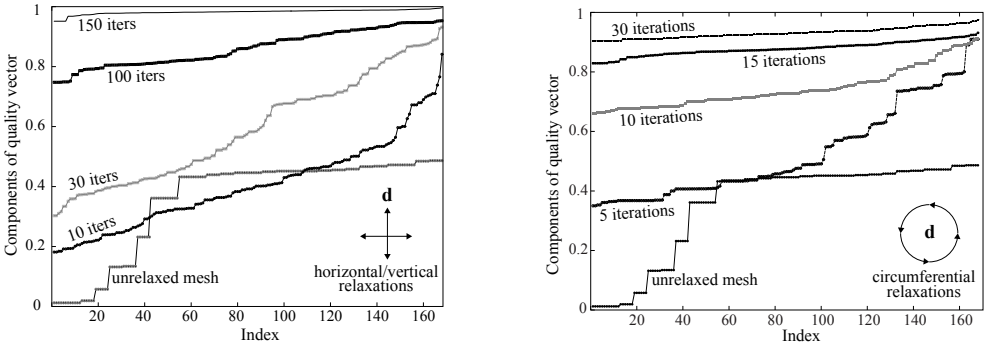
**5.2. Application: Accommodating mesh motion.** This example is aimed at demonstrating how dvr can be useful in numerical simulations involving reasonably large mesh motion, such as in ALE methods. Figure 12a shows a mesh over a hexagonal domain that for the sake of discussion we may think of as the result of advecting vertices according to the fluid velocity in a flow simulation. We seek to improve it using dvr, say to simulate the flow at subsequent times.

Choosing a Cartesian coordinate system $(x, y)$ with the origin at the center of the mesh, Figures 12b–12e show the results from two experiments: (i) when vertices are relaxed along horizontal and vertical directions during even and odd iterations, respectively, and (ii) when vertices are relaxed along the circumferential direction, i.e., a vertex at position $(x, y)$ during a dvr iteration is relaxed along the direction $(-y, x)/\sqrt{x^2 + y^2}$. Vertices along the boundary and at the center of the mesh are held fixed, while all the remaining ones are relaxed. The ordering of relaxed vertices is chosen arbitrarily at the start of the algorithm and is identical in both experiments.

The results shown in Figure 12 help highlight two points. First, the connectivity of the input mesh $\mathcal{T}$ implies that the best possible mesh in $\mathrm{Pert}(\mathcal{T})$ is the mesh of all equilateral triangles having quality $\mathbf{Q} = (1, \ldots, 1)$. However, we do not know a priori which among the candidate solutions in $\mathrm{Pert}(\mathcal{T})$ is computed by the dvr algorithm or the magnitude of improvement in mesh quality that is possible. These will depend on, for instance, the initial positions of vertices in $\mathcal{T}$, the choice of relaxation directions, and the ordering of vertices in $\mathrm{I_R}$. We find that the meshes computed by dvr indeed approach the best solution. The improvement compared to the given distorted mesh is especially noteworthy. That arbitrary choices for the ordering and relaxation di-



(a) unrelaxed mesh (b) 150 iterations, horizontal/vertical relaxation directions (c) 15 iterations, circumferential relaxation directions




(d) (e)

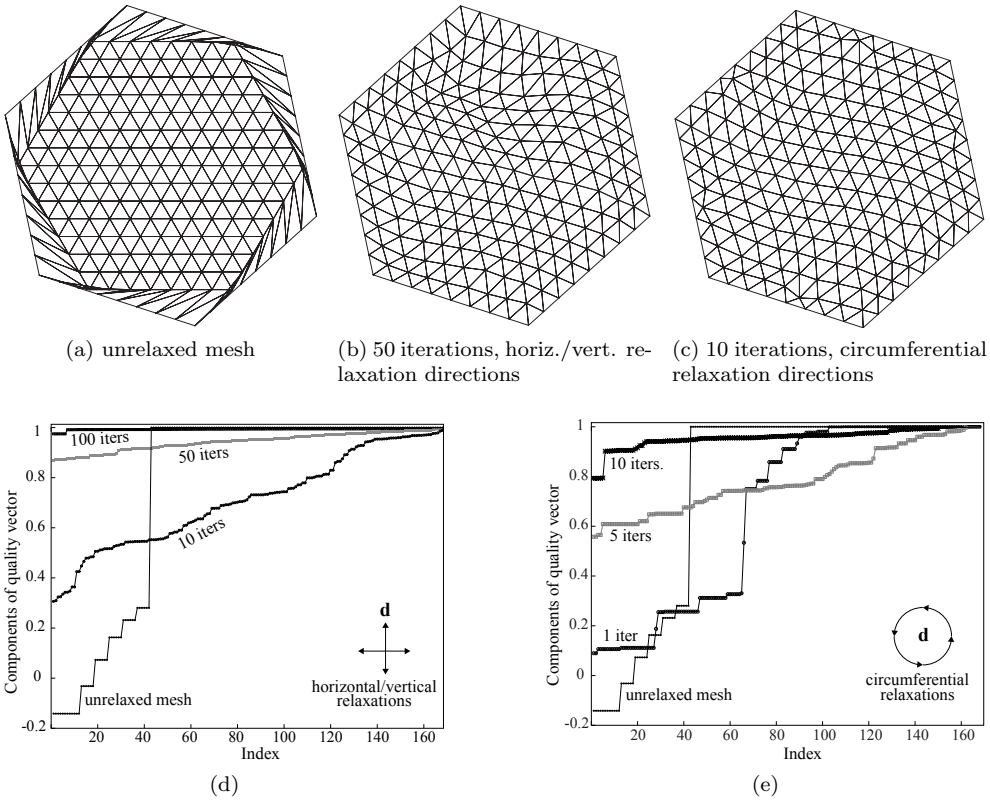
FIG. 12. *Improvement of the mesh in* (a) *using* dvr. *Figures* (d) *and* (e) *show the mesh qualities resulting from relaxing vertices along the directions indicated in the plots. Observe the improvement in mesh qualities in both cases, as well as the difference in the number of iterations required to achieve comparable qualities.*

rections yield meshes approaching the best possible solution is encouraging evidence of the performance of dvr.

Second, comparing Figures 12d and 12e underscores the fact that good choices for relaxation directions can drastically reduce the number of iterations required to achieve a desired mesh quality. While alternating relaxations along horizontal/vertical directions is an "uninformed" choice, using circumferential relaxation directions is motivated by the specific deformations imposed on an equilateral mesh used to construct the input distorted mesh in the first place.

**5.3. Application: Accommodating boundary motion.** In numerical methods for moving boundary problems, we frequently encounter the need to perturb vertices in a mesh to accommodate the prescribed motion of nodes that are advected to follow evolving boundaries/discontinuities such as material interfaces, shocks, or cracks. Figure 13 shows a simple example using dvr to accommodate such boundary motion.

The mesh in Figure 13a is obtained by rotating the boundary of a mesh of equilateral triangles by 18°. We use dvr to improve the quality of the resulting deformed mesh. In practice, only vertices in the vicinity of the perturbed boundary will need to be relaxed. Since boundary displacements are comparable to the size of the domain in this example, we relax all interior vertices while holding the ones along the boundary




(a) unrelaxed mesh (b) 50 iterations, horiz./vert. relaxation directions (c) 10 iterations, circumferential relaxation directions

(d) (e)

FIG. 13. *Optimizing a mesh using* dvr *to accommodate boundary motion.*

and at the center of the mesh fixed.

We repeat the two experiments performed in the previous example by relaxing vertices alternately along horizontal and vertical directions, and by relaxing vertices along circumferential directions. Our observations from section 5.2 are confirmed here as well. Figures 13d and 13e show that the qualities of the computed meshes approach that of the best solution (a mesh of equilateral triangles) and that a good choice for relaxation directions can substantially reduce the number of iterations required to achieve close to best mesh quality. An additional detail to be noted here is that a few elements in the initial mesh $\mathcal{T}$ are inverted, i.e., $Q_1(\mathcal{T}) < 0$, which violates the assumption made in Theorem 4.2. This example serves to highlight the fact that the assumption $Q_1(\mathcal{T}) > 0$ is only a sufficient condition for the conclusions derived there, not a necessary one.

**5.4. Application: Meshing evolving domains.** This final example concerns the application of dvr in an algorithm introduced in [47] for meshing evolving domains using only a nonconforming background mesh. Figure 14a shows the evolution of a hand-shaped domain whose boundary follows a mean curvature flow [14]. The exact details of the evolution are not important for the current discussion; it suffices to note that the boundary is described as a cubic spline and remains $C^2$-regular for the time duration considered. The meshing algorithm discussed in [47] recovers a mesh that conforms to the evolving geometry at each time instant by perturbing a few vertices

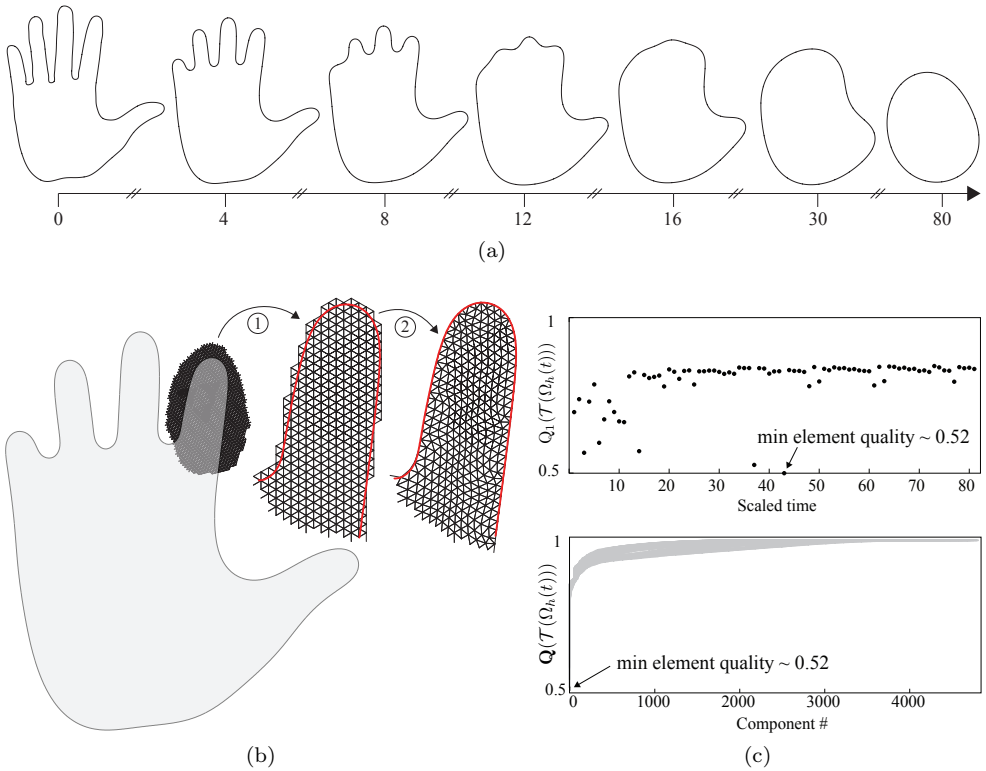



FIG. 14. *The evolution of a hand-shaped domain whose boundary follows a mean-curvature flow is shown in* (a). *At each instant, projecting a few vertices in a fixed background mesh of equilateral triangles onto the immersed boundary and relaxing a few others using* dvr *helps recover meshes that conform to the domain, as shown in* (b). *The meshes computed in this way at a couple of different time instants are shown in Figure* 15. *The plots in* (c) *reveal that despite the large domain deformations, the computed meshes* $\mathcal{T}(\Omega_h(t))$ *at* 80 *different time instants all have good qualities.*

in a fixed background mesh.

For this example, we choose a refined background mesh of all equilateral triangles (not shown). At each time instant, the meshing algorithm identifies elements in the *same* background mesh having at least one vertex inside the domain; see Figure 14b. Vertices on the boundary of this collection of triangles are projected onto their respective closest points on the immersed domain boundary. To accommodate the resulting deformations of triangles of the background mesh, it is necessary to relax a few of the remaining vertices away from the boundary—we adopt dvr for this purpose. In effect, therefore, dvr helps in extending the closest point projection map of the immersed boundary to the interior of the domain being meshed at each time instant. Meshes computed in this way at a couple of different time instants are shown in Figure 15. We refer the reader to [47] for further details on the meshing algorithm itself.

Figure 14c shows details of the qualities of optimized meshes computed at 80 different time instants. That meshes with good qualities are computed for a domain undergoing significant shape changes while using only a fixed background mesh and without resorting to cumbersome cutting/trimming/remeshing operations is noteworthy.

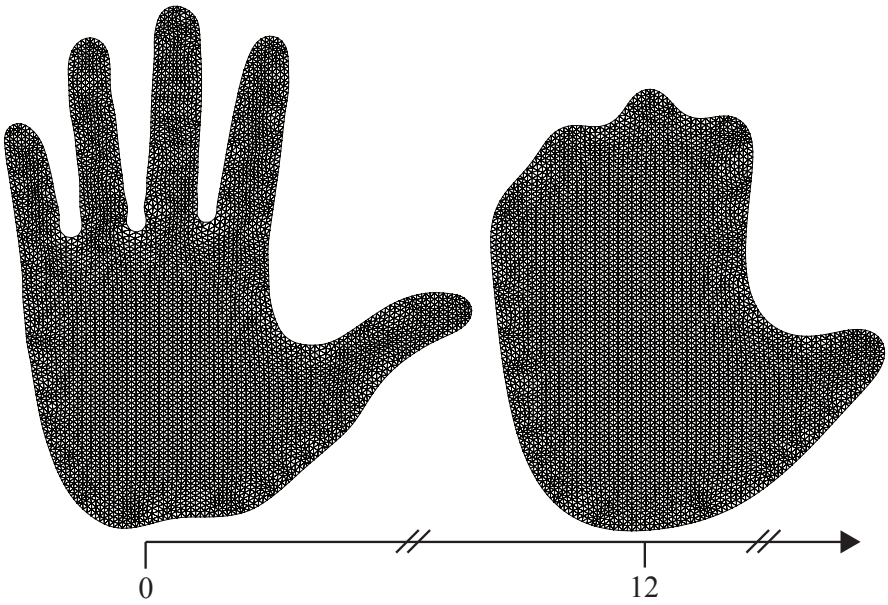



FIG. 15. *Meshes computed at particular time instants for the evolving domain shown in Figure* 14a *by perturbing a few vertices in a nonconforming background mesh* [47].

**6. Outlook.** We have introduced a simple yet robust algorithm called directional vertex relaxation for optimization-based mesh improvement. The guarantees of nondecreasing mesh qualities and the conspicuous absence of any heuristics in the algorithm distinguish it from alternative geometric mesh relaxation strategies proposed in the literature. Numerical experiments reveal good performance of the algorithm in varied representative applications and highlight it as a potent tool in simulation methods for moving boundary problems.

For dvr to be adopted in large scale codes and in challenging numerical simulations, important questions remain. First, it is not evident from the algorithm or from the analysis presented whether the sequence (or a subsequence) of mesh iterates computed by dvr converges to a limiting mesh. An affirmative answer to this question will help find robust termination criteria for the algorithm. It will also reveal the mesh optimization problem approximated by the dvr algorithm. Second, it is important to scrutinize the performance of dvr with a variety of mesh types and element quality metrics. The analysis of the dvr algorithm presented here applies verbatim to improving meshes with quads, hexes, and mixed element types as well. A systematic study identifying quality metrics satisfying the assumptions we have established here, accompanied by detailed numerical experiments examining the magnitude of improvement possible with quad/hex meshes, is still essential. Finally, it will be crucial to generalize/adapt the dvr algorithm to accommodate constraints requiring vertices to remain on specified curves/surfaces. Then dvr can be used to improve meshes with embedded cracks and interfaces, as well as to improve surface meshes.

## REFERENCES

[1] M. AIFFA AND J. FLAHERTY, *A geometrical approach to mesh smoothing*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 4497–4514, https://doi.org/10.1016/S0045-7825(03)00441-9.

[2] G. Allaire, F. Jouve, and A. Toader, *Structural optimization using sensitivity analysis and a level-set method*, J. Comput. Phys., 194 (2004), pp. 363–393, https://doi.org/10.1016/j.jcp.2003.09.032.
[3] N. Amenta, M. Bern, and D. Eppstein, *Optimal point placement for mesh smoothing*, J. Algorithms, 30 (1999), pp. 302–322, https://doi.org/10.1006/jagm.1998.0984.
[4] T. Baker, *Mesh movement and metamorphosis*, Eng. Comput., 18 (2002), pp. 188–198, https://doi.org/10.1007/s003660200017.
[5] R. E. Bank and R. K. Smith, *Mesh smoothing using a posteriori error estimates*, SIAM J. Numer. Anal., 34 (1997), pp. 979–997, https://doi.org/10.1137/S0036142994265292.
[6] F. Blom, *Considerations on the spring analogy*, Internat. J. Numer. Methods Fluids, 32 (2000), pp. 647–668, https://doi.org/10.1002/(SICI)1097-0363(20000330)32:6<647::AIDFLD979>3.0.CO;2-K.
[7] P. Bochev, G. Liao, and G. Pena, *Analysis and computation of adaptive moving grids by deformation*, Numer. Methods Partial Differential Equations, 12 (1996), pp. 489–506, https://doi.org/10.1002/(SICI)1098-2426(199607)12:4<489::AIDNUM5>3.0.CO;2-I.
[8] M. Brewer, L. Freitag, P. Knupp, T. Leurent, and D. Melander, *The mesquite mesh quality improvement toolkit*, in Proceedings of the 12th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM, 2003, pp. 239–259.
[9] C. Budd, W. Huang, and R. Russell, *Adaptivity with moving grids*, Acta Numer., 18 (2009), pp. 111–241, https://doi.org/10.1017/S0962492906400015.
[10] W. Cao, *On the error of linear interpolation and the orientation, aspect ratio, and internal angles of a triangle*, SIAM J. Numer. Anal., 43 (2005), pp. 19–40, https://doi.org/10.1137/S0036142903433492.
[11] L. Chen, P. Sun, and J. Xu, *Optimal anisotropic meshes for minimizing interpolation errors in $L^p$-norm*, Math. Comp., 76 (2007), pp. 179–204, https://doi.org/10.1090/S0025-5718-06-01896-5.
[12] J. Crank, *Free and Moving Boundary Problems*, Oxford Sci. Publ., The Clarendon Press, Oxford University Press, New York, 1987.
[13] V. F. De Almeida, *Domain deformation mapping: Application to variational mesh generation*, SIAM J. Sci. Comput., 20 (1999), pp. 1252–1275, https://doi.org/10.1137/S1064827594274760.
[14] K. Deckelnick, G. Dziuk, and C. Elliott, *Computation of geometric partial differential equations and mean curvature flow*, Acta Numer., 14 (2005), pp. 139–232, https://doi.org/10.1017/S0962492904000224.
[15] L. Diachin, P. Knupp, T. Munson, and S. Shontz, *A comparison of two optimization methods for mesh quality improvement*, Eng. Comput., 22 (2006), pp. 61–74, https://doi.org/10.1007/s00366-006-0015-0.
[16] J. Donea, A. Huerta, J. Ponthot, and A. Rodríguez-Ferran, *Arbitrary Lagrangian–Eulerian methods*, in Encyclopedia of Computational Mechanics, John Wiley & Sons, New York, 2004, pp. 413–437.
[17] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne, *Torsional springs for two-dimensional dynamic unstructured fluid meshes*, Comput. Methods Appl. Mech. Engrg., 163 (1998), pp. 231–245, https://doi.org/10.1016/S0045-7825(98)00016-4.
[18] L. Freitag, *On combining Laplacian and optimization-based mesh smoothing techniques*, in Trends in Unstructured Mesh Generation, ASME, New York, 1997, pp. 37–43.
[19] L. Freitag, M. Jones, and P. Plassmann, *An efficient parallel algorithm for mesh smoothing*, in Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM, 1995, pp. 47–58.
[20] L. Freitag, M. Jones, and P. Plassmann, *A parallel algorithm for mesh smoothing*, SIAM J. Sci. Comput., 20 (1999), pp. 2023–2040, https://doi.org/10.1137/S1064827597323208.
[21] L. Freitag and P. Knupp, *Tetrahedral mesh improvement via optimization of the element condition number*, Internat. J. Numer. Methods Engrg., 53 (2002), pp. 1377–1391, https://doi.org/10.1002/nme.341.
[22] L. Freitag and C. Ollivier-Gooch, *A comparison of tetrahedral mesh improvement techniques*, in Proceedings of the 5th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM, 1996.
[23] L. Freitag and P. Plassmann, *Local optimization-based simplicial mesh untangling and improvement*, Internat. J. Numer. Methods Engrg., 49 (2000), pp. 109–125, https://doi.org/10.1002/1097-0207(20000910/20)49:1/2<109::AIDNME925>3.0.CO;2-U.
[24] P. Frey and H. Borouchaki, *Geometric surface mesh optimization*, Comput. Vis. Sci., 1 (1998), pp. 113–121, https://doi.org/10.1007/s007910050011.
[25] P. J. Frey and H. Borouchaki, *Surface mesh quality evaluation*, Internat. J. Nu-

mer. Methods Engrg., 45 (1999), pp. 101–118, https://doi.org/10.1002/(SICI)1097-0207(19990510)45:1<101::AID-NME582>3.0.CO;2-4.
[26] M. Galassi et al, *GNU Scientific Library Reference Manual*, http://www.gnu.org/software/gsl, 2010.
[27] C. Geuzaine and J. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, Internat. J. Numer. Methods Engrg., 79 (2009), pp. 1309–1331, https://doi.org/10.1002/nme.2579.
[28] B. Helenbrook, *Mesh deformation using the biharmonic operator*, Internat. J. Numer. Methods Engrg., 56 (2003), pp. 1007–1021, https://doi.org/10.1002/nme.595.
[29] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle*, SIAM J. Numer. Anal., 31 (1994), pp. 709–730, https://doi.org/10.1137/0731038.
[30] B. Joe, *Construction of three-dimensional improved-quality triangulations using local transformations*, SIAM J. Sci. Comput., 16 (1995), pp. 1292–1307, https://doi.org/10.1137/0916075.
[31] J. Kennedy, *Particle swarm optimization*, in Encyclopedia of Machine Learning, Springer, New York, 2011, pp. 760–766.
[32] B. Klingner and J. Shewchuk, *Aggressive tetrahedral mesh improvement*, in Proceedings of the 16th International Meshing Roundtable, Springer, Berlin, Heidelberg, 2008, pp. 3–23.
[33] P. Knupp, *Updating meshes on deforming domains: An application of the target-matrix paradigm*, Int. J. Numer. Methods Biomed. Eng., 24 (2008), pp. 467–476, https://doi.org/10.1002/cnm.1013.
[34] P. M. Knupp, *Algebraic mesh quality metrics*, SIAM J. Sci. Comput., 23 (2001), pp. 193–218, https://doi.org/10.1137/S1064827500371499.
[35] P. Knupp, L. Margolin, and M. Shashkov, *Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods*, J. Comput. Phys., 176 (2002), pp. 93–128, https://doi.org/10.1006/jcph.2001.6969.
[36] A. Liu and B. Joe, *On the shape of tetrahedra from bisection*, Math. Comp., 63 (1994), pp. 141–154, https://doi.org/10.1090/S0025-5718-1994-1240660-4.
[37] A. Liu and B. Joe, *Relationship between tetrahedron shape measures*, BIT, 34 (1994), pp. 268–287, https://doi.org/10.1007/BF01955874.
[38] L. Liu, C. Tai, Z. Ji, and G. Wang, *Non-iterative approach for global mesh optimization*, Comput. Aided Design, 39 (2007), pp. 772–782, https://doi.org/10.1016/j.cad.2007.03.004.
[39] T. Munson, *Mesh shape-quality optimization using the inverse mean-ratio metric*, Math. Program., 110 (2007), pp. 561–590.
[40] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, *Laplacian mesh optimization*, in Proceedings of the 4th International ACM Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, 2006, pp. 381–389.
[41] J. Park and S. Shontz, *Two derivative-free optimization algorithms for mesh quality improvement*, Procedia Comput. Sci., 1 (2010), pp. 387–396.
[42] P. Pébay and T. Baker, *Analysis of triangle quality measures*, Math. Comp., 72 (2003), pp. 1817–1839, https://doi.org/10.1090/S0025-5718-03-01485-6.
[43] P.-O. Persson and G. Strang, *A simple mesh generator in* Matlab, SIAM Rev., 46 (2004), pp. 329–345, https://doi.org/10.1137/S0036144503429121.
[44] R. A. Polyak, *Smooth optimization methods for minimax problems*, SIAM J. Control Optim., 26 (1988), pp. 1274–1286, https://doi.org/10.1137/0326071.
[45] R. Rangarajan, *On the resolution of certain discrete univariate max–min problems*, Comput. Optim. Appl., 68 (2017), pp. 163–192, https://doi.org/10.1007/s10589-017-9903-z.
[46] R. Rangarajan, M. Chiaramonte, M. Hunsweck, Y. Shen, and A. Lew, *Simulating curvilinear crack propagation in two dimensions with universal meshes*, Internat. J. Numer. Methods Engrg., 102 (2015), pp. 632–670, https://doi.org/10.1002/nme.4731.
[47] R. Rangarajan and A. Lew, *Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes*, Internat. J. Numer. Methods Engrg., 98 (2014), pp. 236–264, https://doi.org/10.1002/nme.4624.
[48] R. J. Renka, *Mesh improvement by minimizing a weighted sum of squared element volumes*, Internat. J. Numer. Methods Engrg., 101 (2015), pp. 870–886, https://doi.org/10.1002/nme.4845.
[49] M. Rumpf, *A variational approach to optimal meshes*, Numer. Math., 72 (1996), pp. 523–540, https://doi.org/10.1007/s002110050180.
[50] S. Sastry, S. Shontz, and S. Vavasis, *A log-barrier method for mesh quality improvement*, in Proceedings of the 20th International Meshing Roundtable, Springer, Berlin, Heidelberg, 2011, pp. 329–346.

[51] S. Sastry, S. Shontz, and S. Vavasis, *A log-barrier method for mesh quality improvement and untangling*, Eng. Comput., 30 (2014), pp. 315–329, https://doi.org/10.1007/s00366-012-0294-6.

[52] J. Shewchuk, *Stellar: A tetrahedral mesh improvement program*, https://people.eecs.berkeley.edu/~jrs/stellar/input_meshes.zip (3-23-2017).

[53] J. Shewchuk, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, in Applied Computational Geometry towards Geometric Engineering, Springer, Berlin, Heidelberg, 1996, pp. 203–222.

[54] S. Shontz and S. Vavasis, *A mesh warping algorithm based on weighted Laplacian smoothing*, in Proceedings of the 12th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM, 2003, pp. 147–158.

[55] H. Si, *TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, http://http://wias-berlin.de/software/tetgen/ (3-23-2017).

[56] M. Staten, S. Owen, S. Shontz, A. Salinger, and T. Coffey, *A comparison of mesh morphing methods for 3d shape optimization*, in Proceedings of the 20th International Meshing Roundtable, Springer, Berlin, Heidelberg, 2011, pp. 293–311.

[57] VisionAir Project, *Aim@shape: Digital Shape Workbench v5.0*, http://visionair.ge.imati.cnr.it/ (3-23-2017).

[58] S. Walton, O. Hassan, and K. Morgan, *Reduced order mesh optimisation using proper orthogonal decomposition and a modified cuckoo search*, Internat. J. Numer. Methods Engrg., 93 (2013), pp. 527–550, https://doi.org/10.1002/nme.4400.

[59] H. Weizhang and R. Russell, *Adaptive Moving Mesh Methods*, Appl. Math. Sci. 174, Springer New York, 2011.

[60] X. Yang and S. Deb, *Engineering optimisation by cuckoo search*, Intl. J. Math. Model. Num. Opt., 1 (2010), pp. 330–343, https://doi.org/10.1504/IJMMNO.2010.035430.

[61] P. Zavattieri, E. Dari, and G. Buscaglia, *Optimization strategies in unstructured mesh generation*, Internat. J. Numer. Methods Engrg., 39 (1996), pp. 2055–2071, https://doi.org/10.1002/(SICI)1097-0207(19960630)39:12<2055::AIDNME942>3.0.CO;2-2.