A geometric mesh quality improvement algorithm with guarantees

R. Rangarajan and A. Lew

Abstract We discuss an iterative algorithm called *directional vertex relaxation* that optimally perturbs vertices in a mesh along prescribed directions without altering element connectivities. Each vertex update in the algorithm maximizes the poorest quality among the elements around it. With a reasonable set of assumptions on the mesh and on the element quality metric, the algorithm is well defined and the mesh quality is guaranteed to improve with each vertex update. Implementing the algorithm reduces to one-dimensional root finding, rendering it easy to incorporate within existing mesh smoothing codes. We include representative numerical experiments examining the performance of the algorithm for improving triangle and tetrahedral mesh qualities.

1 Introduction

Directional vertex relaxation (dvr) is an algorithm for geometric mesh quality improvement that consists in iteratively optimizing the locations of vertices in a mesh along a prescribed set of directions. The idea behind the algorithm is illustrated in fig. 1, where for the sake of simplicity, we consider relaxing just a single vertex v along vertical and horizontal directions during odd and even iterations, respectively. With the understanding that more regularly-shaped triangles are assigned higher qualities, notice 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 λ^{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 perturbation $v \mapsto v'$ visibly improves

Ramsharan Rangarajan
Department of Mechanical Engineering, Indian Institute of Science Bangalore, India. e-mail: rram@iisc.ac.in

Adrian Lew
Department of Mechanical Engineering, Stanford University, USA. e-mail: lewa@stanford.edu

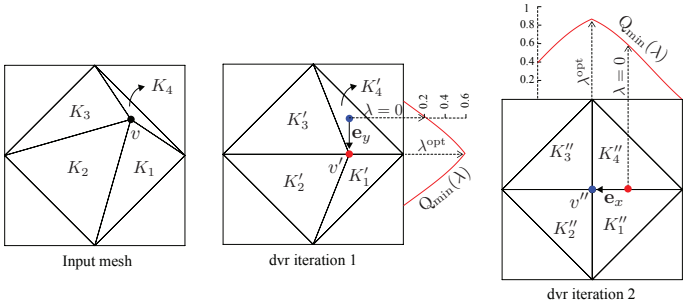


Fig. 1: An illustration of how dvr improves triangle qualities around a vertex v. During the first iteration, v is relocated to v' along the vertical direction by maximizing the function $\lambda \mapsto \mathrm{Q}_{\min}(\lambda)$, which computes the poorest quality among the elements $K_1, \ldots, K_4$ when 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.

the shape of K_4. At the next iteration, dvr computes λ^{opt} such that relaxing v' to $v'' = v' + \lambda^{\text{opt}}\, \mathbf{e}_x$ maximizes the poorest quality among the triangles $K_1'', \ldots, K_4''$. In this particular example, no further improvement is possible. The dvr algorithm essentially consists in repeating the recipe illustrated in the figure— given a mesh, an ordered subset of its vertices that can be relaxed, and the directions along which to relax them, each vertex location is updated by maximizing the minimum quality realized over elements in its 1-ring.

In addition to succinctly describing the dvr algorithm, our goal here is to briefly discuss a few key questions. First, under what conditions are vertex updates in the algorithm well defined? Second, in what sense do these updates improve the mesh quality overall? And third, how can vertex updates be computed in practice? These questions assume additional intrigue owing to the fact that the max-min problems defining vertex updates in dvr are necessarily nonlinear, nonsmooth and nonconvex. Thanks to the deliberate assumption of prescribed relaxation directions however, these max-min problems are one-dimensional. Then we find that exploiting reasonable assumptions on the mesh being relaxed and on properties of the element quality metric yields useful answers to the above questions and facilitates a easy implementation of the algorithm without resorting to any heuristics. For the sake simplicity and brevity, we restrict our discussion of dvr, the statements of related mathematical results and the numerical experiments examining its performance to triangle and tetrahedral mesh types and adopt the mean ratio metric to define element qualities.

2 Directional vertex relaxation: algorithm and examples

We begin by introducing the notation required for a description of the dvr algorithm. We identify a triangulation $\mathscr{T}$ in $\mathbb{R}^d$ with the triple $(\mathrm{V}, \mathrm{I}, \mathrm{C})$, where the multiset V

contains the list of vertex coordinates, I is the list of vertex indices, and C is the list of element connectivities, i.e., $(d+1)-$tuples of indices in I. We shall assume $\mathscr{T}$ to be a mesh of planar triangles in the case $d = 2$, and a mesh of tetrahedra in the case $d = 3$. We denote the location of a vertex with index $i \in \mathrm{I}$ by $\mathrm{V}_i \in \mathrm{V}$, and refer to a simplex K in $\mathscr{T}$ by $K \in \mathscr{T}$. For $i \in \mathrm{I}$, the 1-ring of i is the set $\overset{\star}{e}(i;\mathscr{T})$ containing the list of simplices in $\mathscr{T}$ that are incident at i.

The point of departure for dvr is a mesh $\mathscr{T}$ to be improved, an ordered list $\mathrm{I_R} \subseteq \mathrm{I}$ of vertices that can be relaxed, a prescription $\mathbf{d}_{k,i} \in \mathbb{R}^d$ for the direction along which to relax vertex $i \in \mathrm{I_R}$ at the k-th iteration of the algorithm, and a choice Q for the metric to measure element qualities. Here we assume that Q is the mean ratio metric, which assigns the maximum possible value 1 to equilateral triangles and regular tetrahedra. It is convenient to introduce the shorthand $\mathscr{T}^{i,\lambda}$ to denote the mesh resulting from relocating vertex i from V_i to $\mathrm{V}_i + \lambda\,\mathbf{d}_{k,i}$, and set

$$q_i(\lambda,\mathscr{T}) \triangleq \min\{Q(K) : K \in \overset{\star}{e}(i,\mathscr{T}^{i,\lambda})\} \text{ for } \lambda \in \mathbb{R} \text{ and } i \in \mathrm{I_R}, \tag{1}$$

to denote the poorest quality among simplices in the 1-ring of i in the mesh $\mathscr{T}^{i,\lambda}$. Algorithm 1 now summarizes dvr.

Algorithm 1: Directional vertex relaxation (DVR)

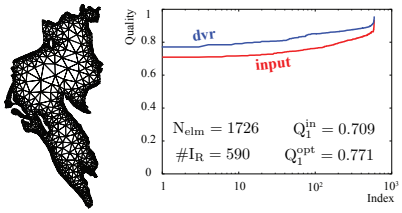
Input: $\mathscr{T} = (\mathrm{V},\mathrm{I},\mathrm{C})$: Input triangulation
$\mathrm{I_R} = (1,\ldots,m)$: Ordered list of vertices to relax
$\{\mathbf{d}_{k,i}\}_{i\in\mathrm{I_R},k\in\mathbb{N}}$: Relaxation directions
N_R : Number of relaxation iterations

for $k = 1$ **to** N_R **do**
  for $i = 1$ **to** m **do**
    Compute: $\lambda^{\mathrm{opt}} \triangleq \arg\max_{\lambda\in\mathbb{R}} q_i(\lambda,\mathscr{T})$ ▷ `nonlinear, nonsmooth & nonconvex max-min problem`
    Update: $\mathrm{V}_i \leftarrow \mathrm{V}_i + \lambda^{\mathrm{opt}}\mathbf{d}_{k,i}$
  end
end
return $(\mathrm{V},\mathrm{I},\mathrm{C})$

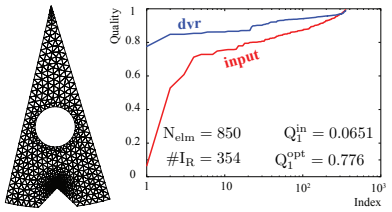
Figure 2 shows representative results from a few numerical experiments using Algorithm 1 to improve meshes accessed from publicly available repositories. In each case, vertices on the boundary of the input mesh are kept fixed while the remaining ones are relaxed along directions that are randomly generated for each vertex at each iteration.

The curves plotted in the figure are components of a *vector-valued mesh quality* $\mathbf{Q}$. In contrast to conventional definitions for mesh qualities as ℓ^p-norms of the list of element qualities, a vector-valued quality is natural in the context of dvr. Setting $q_i(\mathscr{T}) = q_i(0,\mathscr{T})$, the quality of a mesh $\mathscr{T}$ is defined as

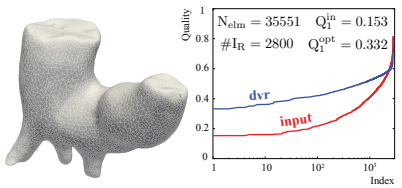
$$\mathbf{Q}(\mathscr{T}) \triangleq \mathbf{asc}\left(q_1(\mathscr{T}),\ldots,q_m(\mathscr{T})\right), \text{ where } \mathrm{I_R} = (1,2,\ldots,m), \tag{2}$$



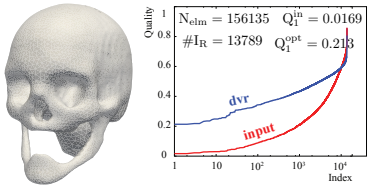
(a) Input mesh: *"superior"* from [4]



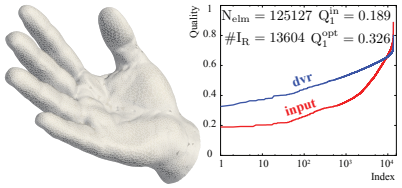
(b) Input mesh from [1]



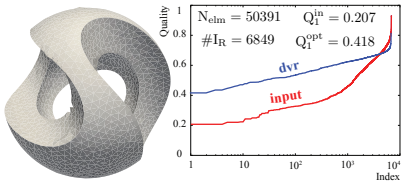
(c) Input mesh: *"aorta"* from [6]



(d) Input mesh: *"Skull"* from [6]



(e) Input mesh: *"Hand man"* from [6]



(f) Input mesh: *"sculpt10kv"* from [5]

Fig. 2: Examples demonstrating the mesh improvement possible with dvr for 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 case of triangle and tetrahedral meshes, respectively.

and **asc** is a permutation that rearranges components of a vector in $\mathbb{R}^m$ in ascending order, i.e.,

$$\mathbf{asc}(u_1 \leq u_2 \leq \ldots \leq u_m) \triangleq (u_1, u_2, \ldots, u_m), \quad m \in \mathbb{N}.$$

A couple of aspects of the definition of $\mathbf{Q}(\mathscr{T})$ are worth highlighting. Its first component $\mathrm{Q}_1(\mathscr{T})$ equals the minimum among the qualities of all simplices that can be perturbed, and is therefore particularly significant in finite element calculations. In the examples in fig. 2 where all interior vertices are relaxed and no element has all its vertices on the boundary, Q_1 equals the poorest element quality in the mesh.

Hence $\mathrm{Q}_1^{\mathrm{in}}$ and $\mathrm{Q}_1^{\mathrm{opt}}$ mentioned in the figures equal the poorest element quality in the input and optimized meshes, respectively. To emphasize the fact that $\mathbf{Q}(\mathscr{T})$ is not simply a lexicographical ordering of element qualities, we mention that the quality of the same element generally appears multiple times in $\mathbf{Q}(\mathscr{T})$. In this sense, $\mathbf{Q}(\mathscr{T})$ naturally accentuates the influence of elements with poor qualities.

In addition to $\mathbf{Q}$, we also need an ordering over the set of mesh qualities to compare the mesh iterates computed by dvr. To this end, we introduce a binary relation on $\mathbb{R}^m$ that is defined as

$$\mathbf{u},\mathbf{w}\in\mathbb{R}^m,\quad \mathbf{u}>\mathbf{w}\iff u_i>w_i \text{ where } i=\arg\min_{1\le j\le m}\{u_j\neq w_j\}. \tag{3}$$

With equality of vectors defined in the usual component-wise sense, we can now say that the mesh iterate $\mathscr{T}_\ell$ has better quality than $\mathscr{T}_k$ if $\mathbf{Q}(\mathscr{T}_\ell)\geq\mathbf{Q}(\mathscr{T}_k)$. In each of the examples in fig. 2, the quality of the mesh computed by dvr is better than the input mesh. This can be deduced by simply observing that the blue curves (qualities of dvr output meshes) start above the red ones (qualities of input meshes). It is however not the case that each component of the quality of the optimized mesh is larger than that of the input mesh, as evident from the fact that the quality curves intersect in some of the examples. Indeed, this observation reveals that dvr autonomously improves the qualities of poor elements by sacrificing qualities of some of the better shaped ones.

Inspecting Step 4 raises an important question of whether Algorithm 1 is well defined. Specifically, when can we expect $\lambda\mapsto q_i(\lambda,\mathscr{T})$ to have a unique maximizer? Next, how is the mesh quality $\mathbf{Q}$ affected by vertex updates in dvr? The issue here is a subtle one— it may seem that since optimizing the location of a vertex improves the quality of the poorest element in its 1-ring, the overall mesh quality ought to improve as well (roughly speaking). However, improvement in the qualities of some elements comes at the expense of reducing the qualities of a few others. There is hence a balancing act at play, where some element qualities are improved and others worsened during each dvr iteration. What can we say about the qualities of mesh iterates then? These questions are addressed by theorem 1.

3 Mathematical guarantees for DVR

Even though the conclusions of theorem 1 hold for a general class of quality metrics, we intentionally state it specifically for the case of the mean ratio metric. We do this to avoid distracting details concerning properties required of the quality metric, and instead focus on the guarantees offered by dvr. Besides, the mean ratio metric is widely used and is directly relevant to the examples in fig. 2.

Theorem 1 (Theorem 2 in [3]). *Given $\mathscr{T}$, let $\mathscr{T}_{k,i}$ 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 $\mathscr{T}_{k+1,0}=\mathscr{T}_{k,m}$. Let Q be the mean ratio element quality metric and assume that $\mathrm{Q}_1(\mathscr{T})>0$. Then:*

[(i)]

1. $\mathscr{T}_{k,i}$ *is well defined for each* $k \in \mathbb{N}$ *and* $i \in \mathrm{I_R}$.
2. $\mathbf{Q}(\mathscr{T}_{k,i}) \geq \mathbf{Q}(\mathscr{T}_{k,i-1}^{i,\lambda})$ *for each* $k \in \mathbb{N}$ *and* $\lambda \in \mathbb{R}$.
3. *The sequence of mesh qualities is nondecreasing, i.e.,*

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

4. *The sequence* $(\mathrm{Q_1}(\mathscr{T}), \mathrm{Q_1}(\mathscr{T}_{1,1}), \ldots, \mathrm{Q_1}(\mathscr{T}_{1,m}), \ldots, \mathrm{Q_1}(\mathscr{T}_{k,1}), \ldots, \mathrm{Q_1}(\mathscr{T}_{k,m}), \ldots)$ *is nondecreasing and therefore convergent.*

A key assumption on the input mesh $\mathscr{T}$, namely $\mathrm{Q_1}(\mathscr{T}) > 0$, implies that none of the elements in $\mathscr{T}$ can be inverted. This requirement is a sufficient condition and not a necessary one, and is intimately related to the convexity of certain positive level sets of the mean ratio metric. The same property, together with continuity, boundedness, degeneracy and decay conditions discussed in [3] and which are all satisfied by the mean ratio metric, helps to ensure that the optimizer λ^{opt} in the algorithm exists and is unique, and hence that each vertex update in dvr is well defined as claimed in (i). Point (ii) shows that the mesh $\mathscr{T}_{k,i}$ has better quality than any mesh that can be obtained by perturbing vertex i along $\mathbf{d}_{k,i}$ in $\mathscr{T}_{k,i-1}$. In particular, setting $\lambda = 0$ in (ii) reveals that $\mathscr{T}_{k,i}$ has better quality than $\mathscr{T}_{k,i-1}$, which is precisely claim (iii). Hence each vertex update in dvr can only improve the mesh quality. Inspecting just the first component of the qualities of mesh iterates shows that the component $\mathrm{Q_1}$ is nondecreasing. Notably, the poorest element quality (among elements that can be perturbed) in the mesh iterates is a monotonic sequence and therefore converges, which is claim (iv). This observation suggests a simple termination criterion for Algorithm 1— stop when the poorest element quality appears to have converged. In practice however, we use a fixed number of iterations because the remaining components of the mesh quality continue to improve even after the poorest quality has converged.

Finally we mention that resolving the max-min problem in Step 4 to compute optimal perturbations is a delicate matter because even if the quality metric Q is a smooth function of the element coordinates, the map $\lambda \mapsto q_i(\lambda, \mathscr{T})$ being a minimum among smooth functions is continuous but not differentiable. Consequently, λ^{opt} cannot be computed by simple derivative-based methods. Yet, an efficient and robust strategy for computing λ^{opt} is crucial because one such max-min problem has to be resolved for each vertex update in dvr. For the case of the mean ratio metric however, computing λ^{opt} turns out to be surprisingly simple and reduces to computing roots of scalar polynomials, see [3]. The execution times reported in fig. 3 correspond to using an efficient strategy for computing λ^{opt} that is discussed and analyzed in [2].

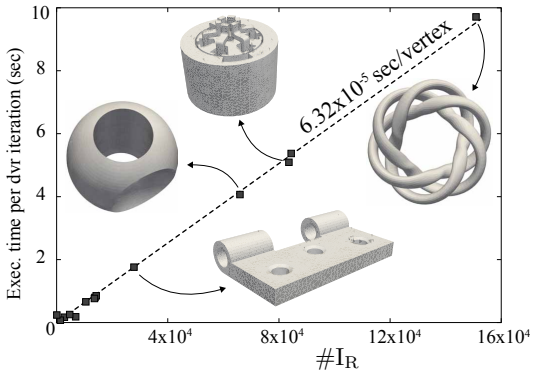


Fig. 3: Execution time per dvr iteration for optimizing tetrahedral meshes while using the algorithm described in [2] to compute λ^{opt}. The times reported are computed as an average over 40 iterations. Our serialized implementation of Algorithm 1 was run on a Mac Pro with a 3.5GHz Intel Xeon E5 processor and using the *gcc* compiler (version 5.4.0, -O2 optimization flag). Notice the linear scaling of the execution time with the number of vertices being relaxed.

4 Concluding remarks

We conclude this article mentioning that the dvr algorithm can serve as a useful tool in algorithms for simulating moving boundary problems. These simulations require mesh improvement at every iteration/time step, and offer no prospect of user intervention. In such a context, the lack of any heuristics in the dvr together with the guarantees it offers makes it stand out compared to alternatives in the literature.

References

1. P. Persson and G. Strang. A simple mesh generator in matlab. *SIAM Rev.*, 46(2):329–345, 2004.
2. R. Rangarajan. On the resolution of certain discrete univariate max–min problems. *Comput. Optim. Appl.*, pages 1–30, 2017.
3. R. Rangarajan and A. Lew. Provably robust directional vertex relaxation for geometric mesh optimization. *SIAM J. Sci. Comput.*, 39(6):A2438–A2471, 2017.
4. J. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Appl. Comput. Geom. towards Geometric Engng.*, pages 203–222. Springer, 1996.
5. J Shewchuk. Stellar: A tetrahedral mesh improvement program. https://people.eecs.berkeley.edu/ jrs/stellar/input_meshes.zip, Accessed on 03-23-2017.
6. VisionAir Project. Aim@shape: Digital shape workbench v5.0. http://visionair.ge.imati.cnr.it/, Accessed on 03-23-2017.