

DVRlib: A C++ library for geometric mesh improvement using Directional Vertex Relaxation

Ramsharan Rangarajan¹ and Adrian Lew²

¹ Department of Mechanical Engineering, Indian Institute of Science Bangalore, India ² Department of Mechanical Engineering, Stanford University, USA

DOI: [10.21105/joss.02372](https://doi.org/10.21105/joss.02372)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Kevin M. Moerman](#) ↗

Reviewers:

- [@hugoledoux](#)
- [@jhale](#)
- [@jonwong12](#)

Submitted: 20 May 2020

Published: 25 August 2020

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

The success of finite element simulations in physical sciences, applied mathematics, and computer graphics relies on generating high-quality unstructured meshes. In general, well-shaped elements are essential to ensure accurate numerical approximations, to improve the conditioning of systems of equations to be resolved, and to choose reasonable step sizes in time integration schemes.

DVRlib is a C++ library that implements the directional vertex relaxation algorithm introduced in Rangarajan & Lew (2017) to improve qualities of unstructured meshes. DVRlib improves qualities of triangular and tetrahedral elements by *iteratively* and *optimally* perturbing a selected set of vertices along prescribed directions. During the process, the connectivity of the mesh remains unaltered; just the locations of vertices may change. DVRlib is straightforward to couple with existing libraries for unstructured mesh generation by invoking the functionalities it provides as a simple post-processing step. Equally significantly, DVRlib enables engineers and researchers to simulate a challenging class of moving boundary problems by helping to maintain good element qualities in deforming meshes.

What DVRlib does (and doesn't)

DVRlib provides robust and efficient functionalities for improving element qualities in triangular and tetrahedral meshes by perturbing locations of a specified set of vertices along a prescribed set of directions. Despite relying solely on vertex perturbations for (geometric) mesh relaxation, DVRlib generally achieves significant improvement in mesh quality. The documentation accompanying the library provides numerous examples demonstrating improvement in mesh quality achieved using DVRlib when relaxing meshes produced by commonly used mesh generators (Alliez et al., 2020; Altair Engineering, 2020; Geuzaine & Remacle, 2009; Shewchuk, 1996; Si, 2015).

A large number of software libraries provide a variety of tools for generating, improving and manipulating unstructured meshes (Altair Engineering, 2020; Geuzaine & Remacle, 2009; Shewchuk, 1996; Si, 2015; The CGAL Project, 2020; Schöberl, 2020). In this context, it is important to note what DVRlib is *not*:

- DVRlib is not a mesh generator but can be coupled to one.
- DVRlib is not a tool for adaptive mesh refinement but can improve an adaptively refined mesh.
- DVRlib is not a mesh untangling tool but can improve an untangled mesh.
- DVRlib is not a mesh simplification tool.

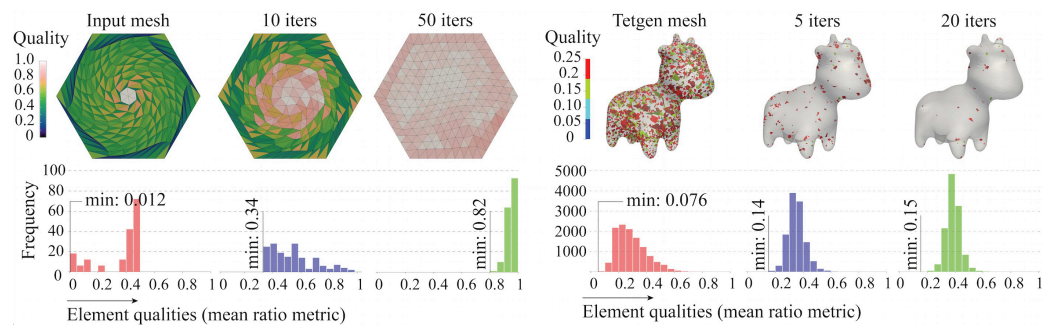


Figure 1: Examples illustrating connectivity-preserving mesh improvement with DVRlib.

DVRlib is neither intended to be nor is a comprehensive mesh improvement toolbox. DVRlib performs a very specific type of vertex relaxation to improve mesh qualities. Nevertheless, DVRlib can be coupled with libraries providing a complementary suite of operations, including topological operations such as edge/face swapping, vertex insertion/removal and local remeshing (The CGAL Project, 2020; Vartziotis & Wipperfurth, 2018; Freitag-Diachin, 2020; Schlömer, 2020).

Statement of need

Despite the large volume of literature on the topic and the extensive arsenal of operations available for mesh relaxation, mesh improvement remains a challenging problem. The state of affairs is well summarized in the words of Geuzaine & Remacle (2009): “*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.*”

Mesh relaxation with guarantees. The directional vertex relaxation (DVR) algorithm implemented by DVRlib enjoys the distinctive feature of being provably robust. DVR guarantees that the mesh quality, and in particular, the poorest quality among elements being perturbed improves monotonically with each vertex perturbation at each iteration (Rangarajan & Lew, 2017). These guarantees hold independently of the choice of vertices being relaxed and the choice of relaxation directions, and even when computing with finite precision. These features of DVR, which are inherited by its implementation in DVRlib, are notable when compared with commonly used mesh smoothing techniques that are often based on heuristic analogies with mechanical systems (Degand & Farhat, 2002; Persson & Strang, 2004). Furthermore, unlike mesh relaxation algorithms that require resolving systems of equations (Budd, Huang, & Russell, 2009; Freitag & Knupp, 2002), DVRlib relaxes vertices by resolving single-variable optimization problems.

Simulating moving boundary problems. Simulating problems involving domains that evolve with time present algorithmic challenges stemming from deterioration in mesh quality due to element distortion. Fluid-structure interaction and shape optimization problems, which are of a broad interest in research and engineering, fall in this category. Practical simulation strategies invariably rely on remeshing the domain once the mesh is deemed to be sufficiently distorted (Dapogny, Dobrzynski, Frey, & Froehly, 2020). The functionalities provided by DVRlib are well-suited to aid such simulations, see Figure 1. First, a deforming mesh can be relaxed periodically with DVRlib and hence used for longer durations (either in time or number of iterations) before remeshing entirely. Second, since DVRlib retains element connectivities during mesh relaxation, the sparsity of data structures involved in a simulation are preserved (e.g., mass and stiffness matrices) so that expensive memory reallocation is avoided.

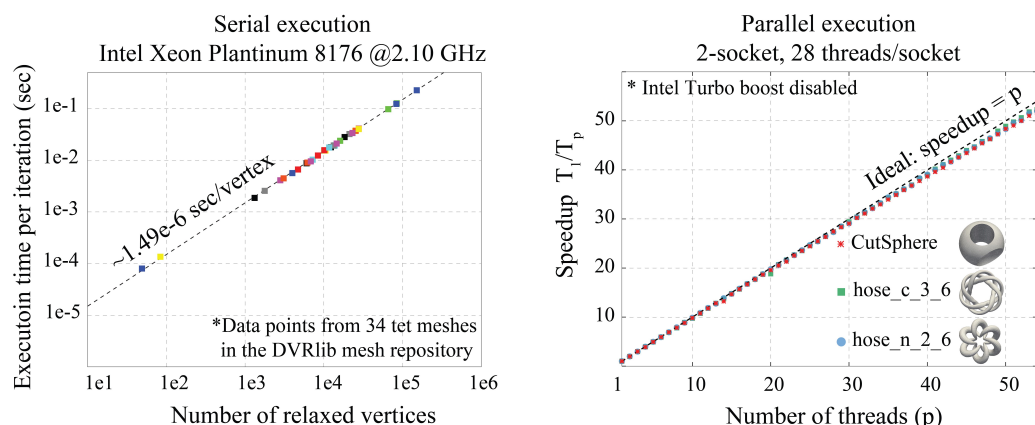


Figure 2: Performance benchmarks for serial and thread-parallel mesh optimization routines in DVRlib.

Third, the guarantee of robustness underlying the mesh relaxation algorithm implemented by DVRlib is crucial when simulating moving boundary problems, since user-interactivity in guiding mesh improvement at each time step or each iteration is impossible.

In this context, we mention that DVRlib is an integral component of on-going research on universal meshes for discretizing evolving domains (Rangarajan, Kabaria, & Lew, 2019). We refer to Sharma & Rangarajan (2019) for an application that uses DVRlib to adapt triangle meshes to conform to moving contact interfaces that arise when simulating elastic contact.

Non-smooth max-min optimization. The mathematical problem defining vertex perturbations in the DVR algorithm is of the form:

$$\text{Find } \lambda^{\text{opt}} \in \arg \max_{\lambda \in \mathbb{R}} F(\lambda), \quad \text{where } F(\lambda) \triangleq \min\{f_1(\lambda), f_2(\lambda), \dots, f_n(\lambda)\}. \quad (1)$$

Each univariate scalar-valued function $\lambda \mapsto f_\alpha(\lambda)$ appearing in (1) represents the variation in quality of an element as one of its nodes is perturbed along a prescribed direction by the coordinate $\lambda \in \mathbb{R}$. The vertex perturbation λ^{opt} is optimal in the sense that it maximizes the minimum among the qualities of all elements incident at the vertex.

Computing λ^{opt} , however, is a challenge. Since element quality metrics generally depend smoothly on vertex locations, each function f_α is smooth. As the minimum among the functions $\{f_\alpha\}_\alpha$, however, F is continuous but not differentiable. Hence, eq. (1) represents a non-smooth max-min optimization problem for which straightforward Newton-type methods do not apply. Ad hoc techniques are ill-advised because mesh relaxation requires resolving one such problem for each vertex perturbation at each iteration of the algorithm. Heuristic methods are also likely to be inefficient, since the number of functions n , which equals the valence of the vertex being relaxed, can be large. In tetrahedral meshes, for example, n ranges from 25 to 30 for commonly used mesh generators. One of the main functionalities provided by DVRlib is an efficient implementation of the algorithm introduced and analyzed in Rangarajan (2017) to resolve (1) for a specific class of element quality metrics.

Functionalities in DVRlib

- *Mean ratio element quality metrics* for 2D triangles and 3D tetrahedra.
- *Max-min solvers* to compute optimal unconstrained vertex perturbations and sampling-based suboptimal constrained vertex perturbations.
- *Relaxation direction generators* for vertex relaxation along Cartesian and random directions, and along tangent planes to curvilinear manifolds.

- *Mesh optimization routines.* Overloaded vertex and mesh optimization routines for serial and thread-parallel execution, see [Figure 2](#).
- *Mesh data structures.* Recognizing the diversity in commonly used mesh data structures, DVRLib retains the mesh type as a template parameter throughout. This also facilitates integrating DVRLib into high-performance computing codes. A rudimentary mesh data structure is included with the library for testing.

The documentation for DVRLib provides a detailed set of tutorial-style examples for users to get started. The library also provides a mesh repository to help test its functionalities and to reproduce the benchmark examples discussed in the documentation.

We expect future extensions of the library to introduce new mesh types (e.g., quadrilaterals, bricks) and to incorporate a broader class of element quality metrics.

Acknowledgements

RR thanks the Defence Research and Development Organization's (DRDO, India) research grant JATP/P-VIII/P-2019/164 through the Joint Advanced Technology Program with the Indian Institute of Science Bangalore, the Science and Engineering Research Board's (SERB, India) early career award ECR/2017/000346, and Ms. Varshini Subash for assistance with code profiling and testing as part of her undergraduate thesis.

References

- Alliez, P., Jamin, C., Rineau, L., Tayeb, S., Tournois, J., & Yvinec, M. (2020). 3D mesh generation. In *CGAL user and reference manual* (5.0.2 ed.). CGAL Editorial Board. Retrieved from https://doc.cgal.org/latest/Mesh_3/group__PkgMesh3Ref.html
- Altair Engineering, I. (2020). HyperMesh. Retrieved from <https://altairhyperworks.com/product/HyperMesh>
- Budd, C., Huang, W., & Russell, R. (2009). Adaptivity with moving grids. *Acta Numer.*, 18(1), 111–241. doi:[10.1017/S0962492906400015](https://doi.org/10.1017/S0962492906400015)
- Degand, C., & Farhat, C. (2002). A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers & Structures*, 80(3-4), 305–316. doi:[10.1016/S0045-7949\(02\)00002-0](https://doi.org/10.1016/S0045-7949(02)00002-0)
- Freitag, L., & Knupp, P. (2002). Tetrahedral mesh improvement via optimization of the element condition number. *Internat. J. Numer. Methods Engrg.*, 53(6), 1377–1391. doi:[10.1002/nme.341](https://doi.org/10.1002/nme.341)
- Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *Internat. J. Numer. Methods Engrg.*, 79(11), 1309–1331. doi:[10.1002/nme.2579](https://doi.org/10.1002/nme.2579)
- Persson, P.-O., & Strang, G. (2004). A simple mesh generator in matlab. *SIAM Review*, 46(2), 329–345. doi:[10.1137/S0036144503429121](https://doi.org/10.1137/S0036144503429121)
- Rangarajan, R. (2017). On the resolution of certain discrete univariate max–min problems. *Comput. Optim. Appl.*, 68(1), 163–192. doi:[10.1007/s10589-017-9903-z](https://doi.org/10.1007/s10589-017-9903-z)
- Rangarajan, R., Kabaria, H., & Lew, A. (2019). An algorithm for triangulating smooth three-dimensional domains immersed in universal meshes. *Internat. J. Numer. Methods Engrg.*, 117(1), 84–117. doi:[10.1002/nme.5949](https://doi.org/10.1002/nme.5949)

- Rangarajan, R., & Lew, A. J. (2017). Provably robust directional vertex relaxation for geometric mesh optimization. *SIAM J. Sci. Comput.*, 39(6), A2438–A2471. doi:[10.1137/16M1089101](https://doi.org/10.1137/16M1089101)
- Sharma, A., & Rangarajan, R. (2019). A shape optimization approach for simulating contact of elastic membranes with rigid obstacles. *Internat. J. Numer. Methods Engrg.*, 117(4), 371–404. doi:[10.1002/nme.5960](https://doi.org/10.1002/nme.5960)
- Shewchuk, J. (1996). Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *Workshop on applied computational geometry* (pp. 203–222). Springer. doi:[10.1007/BFb0014497](https://doi.org/10.1007/BFb0014497)
- Si, H. (2015). TetGen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Software*, 41(2), 1–36. doi:[10.1145/2629697](https://doi.org/10.1145/2629697)
- The CGAL Project. (2020). *CGAL user and reference manual* (5.0.2 ed.). CGAL Editorial Board. Retrieved from <https://doc.cgal.org/5.0.2/Manual/packages.html>
- Vartziotis, D., & Wipperfurth, J. (2018). *The getme mesh smoothing framework: A geometric way to quality finite element meshes*. CRC Press. ISBN: 9780429680106
- Dapogny, C., Dobrzynski, C., Frey, P., & Froehly, A. (2020). Mmg platform: Robust, open-source and multidisciplinary software for remeshing. *GitHub repository*. GitHub. Retrieved from <https://github.com/MmgTools/mmg>
- Freitag-Diachin, L. (2020). Mesquite: Mesh quality improvement toolkit. Retrieved from <https://trilinos.github.io/mesquite.html>
- Schlömer, N. (2020). OPTIMESH: Triangular mesh optimization. *GitHub repository*. GitHub. Retrieved from <https://github.com/nshloe/optimesh>
- Schöberl, J. (2020). Netgen mesh generator. *GitHub repository*. GitHub. Retrieved from <https://github.com/NGSolve/netgen>