The Journal of Open Source Software

SeismicMesh: Triangular meshing for seismology

Keith J. Roberts[1], **Rafael dos Santos Gioria**[1], **and William J. Pringle**[2]

1 Research Center for Gas Innovation, Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil. **2** Dept. of Civil and Environmental Engineering and Earth Sciences, University of Notre Dame, 156 Fitzpatrick Hall, Notre Dame, IN, U.S.A.

DOI: 10.21105/joss.02687

Software

- Review
- Repository
- Archive

Editor: Marie E. Rognes

Reviewers:

- @nschloe
- @jorgensd

Submitted: 21 August 2020
Published: 18 January 2021

Summary

SeismicMesh is a Python package for simplex mesh generation in two or three dimensions. As an implementation of DistMesh (Persson & Strang, 2004), it produces high-quality meshes at the expense of speed. For increased efficiency, the core package is written in C++, works in parallel, and uses the Computational Geometry Algorithms Library (Hert & Seel, 2020). SeismicMesh can also produce mesh-density functions from seismological data to be used in the mesh generator.

Background

Generating a high-quality graded mesh for a geophysical domain represents a challenge for seismological modeling using the finite element method (FEM). In these applications, a domain is discretized typically with triangular/tetrahedral elements that vary widely in size around features of interest. These meshes are commonly used with the FEM to solve partial differential equations that model acoustic or elastic waves, which are used in seismic velocity model building algorithms such as full waveform inversion (FWI) (Tarantola, 1984; Virieux & Operto, 2009) and reverse time migration (Modave et al., 2015).

Statement of Need

Despite the fact that many mesh generation programs exist such as Gmsh (Geuzaine & Remacle, 2009) and CGAL (Alliez et al., 2020; Rineau, 2020), it is uncommon to find capabilities that incorporate geophysical data into the mesh generation process to appropriately size elements. This in part contributes to the reality that automatic mesh generation for geophysical domains is not user-friendly.

Some packages have been created to script mesh generation from geophysical datasets such as in coastal ocean modeling (Gorman et al., 2008; K. J. Roberts et al., 2019) and reservoir modeling (Cacace & Blöcher, 2015). In a similar manner, the aim of this package is to provide a straightforward Python package to script mesh generation directly from seismic velocity models. This is accomplished first by building a mesh density function using seismic velocity data and then supplying these inputs to a mesh generator that can use these inputs and operate at scale.

The mesh density function can be used as input other mesh generators. However, the usage of a sizing function can have significant impact on the mesh generation performance. For example, Gmsh's advancing front and Delaunay refinement methods construct the mesh incrementally and do not permit vectorization, which leads to reduced performance at scale in

2D/3D. In contrast, the DistMesh algorithm takes advantage of vectorization when querying a complex mesh density function making it efficient and competitive to Gmsh for this kind of meshing problem.

Core functionality

1. The creation of 2D/3D graded mesh size functions defined on axis-aligned regular Cartesian grids. These mesh sizing functions encode mesh resolution distributions that conform to the variations from inputted seismic velocity model data and are distributed according to several heuristics (see K. Roberts, 2020 for further details). Mesh size function grading is accomplished using (Persson, 2006).
2. Distributed memory parallelism. The generation of potentially large ($>$ 10 million cells) high-quality triangular or tetrahedral meshes using distributed memory parallelism with mesh resolution following sizing functions.
3. An implementation of a 3D degenerate (i.e., sliver) tetrahedral element removal technique (Tournois et al., 2009) to bound a mesh quality metric. Note that 2D mesh generation does not suffer from the formation of degenerate elements.

Similar to other meshing programs such as Gmsh, SeismicMesh (K. Roberts, 2020) enables generation of simplex meshes through a Python application programming interface.

The mesh's domain geometry is defined as the 0-level set of a signed distance function (SDF), which avoids the need to have explicit geometry information defining the boundary and can be particularly useful in geophysical domains.

Performance Comparison

We compare the 2D/3D serial performance in terms of cell quality and mesh creation time between SeismicMesh, Gmsh (Geuzaine & Remacle, 2009) and CGAL (Alliez et al., 2020; Rineau, 2020). The cell quality is defined as the product of the topological dimension of the mesh (2 or 3) and the incircle radius divided by the circumcircle radius and ranges between 0 and 1, where 1 is a perfectly symmetrical simplex. In mesh generation, there is always a trade-off between generation speed and mesh quality. We find that Gmsh produces high-quality meshes by far the fastest, SeismicMesh will produce meshes with the best quality, but much slower. Gmsh becomes comparatively slow when a user-defined mesh-density function is involved, which is SeismicMesh's primary use case.

For the two seismic domains (e.g., BP2004 and EAGE), SeismicMesh is faster than Gmsh for the 2D BP2004 benchmark but slightly slower for the 3D EAGE benchmark at scale. CGAL is not competitive for the 3D benchmark and is therefore not shown. Interpolant-based mesh sizing functions significantly slow the mesh generation time of Gmsh by a factor of ~ 3 as Gmsh calls the sizing function for each point individually (e.g., 95,756 times) whereas SeismicMesh does it for all points at once each meshing iteration (e.g., 26 times).

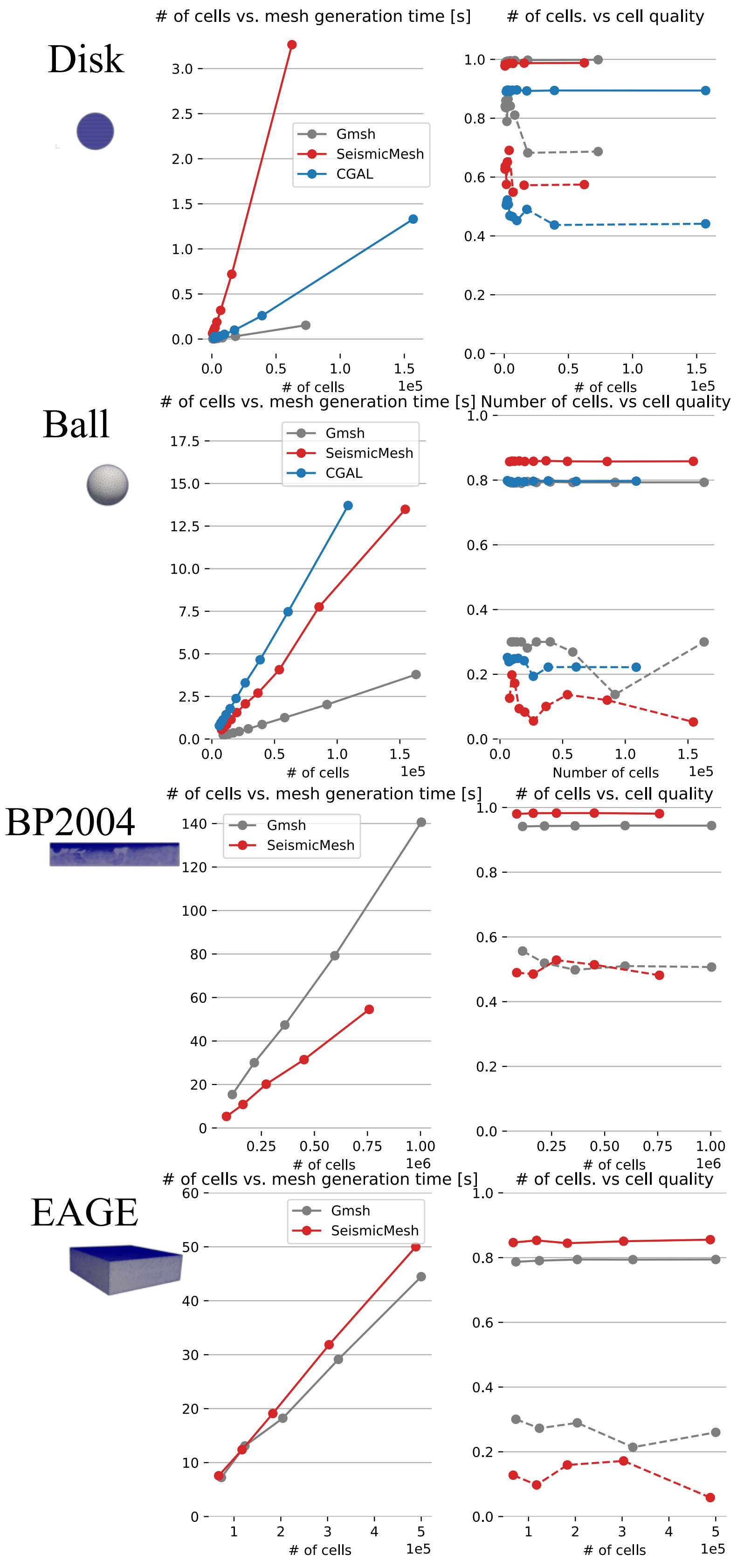

Figure 1: Using SeismicMesh V3.2.0, the mesh creation time (left columns) and resulting cell quality (right columns) for the four benchmarks studied over a range of problem sizes. For the panels that show cell quality, solid lines indicate the mean and dashed lines indicate the minimum cell quality in the mesh.

Parallelism

A simplified version of the parallel Delaunay algorithm proposed by (Peterka et al., 2014) is implemented inside the DistMesh algorithm, which does not consider sophisticated domain decomposition or load balancing yet. Figure 2 shows a peak speed-up of approximately 6 times using 11 cores when performing 50 meshing iterations to generate the 33M cell mesh of the EAGE P-wave velocity model. While the parallel performance is not perfect at this stage of development, the capability reduces the generation time of this relatively large example (e.g., 33 M cells) from 91.0 minutes to approximately 15.6 minutes. Results indicate that the simple domain decomposition approach inhibit perfect scalability. The machine used for this experiment was an Intel Xeon Gold 6148 machine clocked at 2.4 GHz with 192 GB of RAM connected together with a 100 Gb/s InfiniBand network.

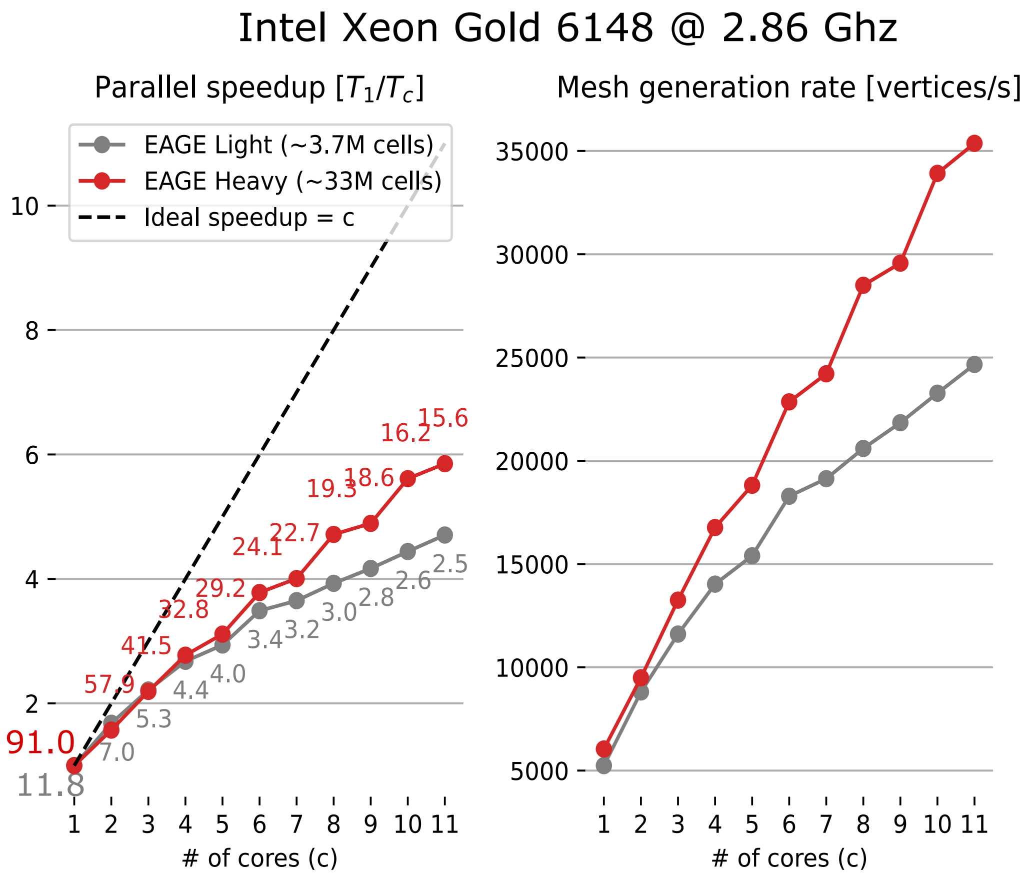

Figure 2: The speedup (left-panel) as compared to the serial version of SeismicMesh V3.2.0 for a relatively light and heavy mesh each adapted to P-wave data from the EAGE Salt seismic velocity model. The total mesh generation wall-clock time is annotated in decimal minutes next to each point. The panel on the right hand side shows the mesh generation rate normalized by the number of total number of cells in the mesh.

Ongoing and future applications

Some future applications for this software:

- SeismicMesh is being used by a group of researchers to build 2D/3D meshes for a seismological FEM model that has been developed in the Firedrake computing environment

(Rathgeber et al., 2017).

- The usage of SDF to implicitly define the meshing domain presents potential use cases in a topology-optimization framework (Laurain, 2018) for modeling the sharp interface of salt-bodies in seismological domains. In these applications, the 0-level set of a SDF is used to demarcate the boundary of the feature. Each inversion iteration, an optimization problem is solved to produce modifications to the location of the 0-level set. In this framework, SeismicMesh can be used within the inversion algorithm to generate and adapt meshes.
- Much like how the original DistMesh program has been used, SeismicMesh can be adapted for other domain-specific applications besides seismology (e.g., fluid dynamics, astrophysics, and oceanography). An open source project project is already under way to use the same mesh generation technology for a Python version of OceanMesh2D to build industrial-grade meshes of coastal oceans (K. J. Roberts et al., 2019).

We expect future extensions of the program to introduce better domain decomposition algorithms to improve parallel performance.

Acknowledgements

This research was carried out in association with the ongoing R&D project registered as ANP 20714-2, "Software technologies for modelling and inversion, with applications in seismic imaging" (University of São Paulo / Shell Brasil / ANP). We would like to thank the two reviewers who helped improve the presentation and quality of this manuscript and package greatly.

References

Alliez, P., Jamin, C., Rineau, L., Tayeb, S., Tournois, J., & Yvinec, M. (2020). 3D mesh generation. In *CGAL user and reference manual* (5.1 ed.). CGAL Editorial Board. https://doc.cgal.org/5.1/Manual/packages.html#PkgMesh3

Cacace, M., & Blöcher, G. (2015). MeshIt—a software for three dimensional volumetric meshing of complex faulted reservoirs. *Environ Earth Sci*, *74*(6), 5191–5209. https://doi.org/10.1007/s12665-015-4537-x

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities: THE GMSH PAPER. *Int. J. Numer. Meth. Engng.*, *79*(11), 1309–1331. https://doi.org/10.1002/nme.2579

Gorman, G. J., Piggott, M. D., Wells, M. R., Pain, C. C., & Allison, P. A. (2008). A systematic approach to unstructured mesh generation for ocean modelling using GMT and terreno. *Comput. Geosci-Uk.*, *34*(12), 1721–1731. https://doi.org/10.1016/j.cageo.2007.06.014

Hert, S., & Seel, M. (2020). Convex hulls. In *Lectures on convex sets* (5.1.1 ed., pp. 123–147). WORLD SCIENTIFIC. https://doi.org/10.1142/9789811202124_0004

Laurain, A. (2018). A level set-based structural optimization code using FEniCS. *Struct Multidisc Optim*, *58*(3), 1311–1334. https://doi.org/10.1007/s00158-018-1950-2

Modave, A., St-Cyr, A., Mulder, W. A., & Warburton, T. (2015). A nodal discontinuous galerkin method for reverse-time migration on GPU clusters. *Geophys. J. Int.*, *203*(2), 1419–1435. https://doi.org/10.1093/gji/ggv380

Persson, P.-O. (2006). Mesh size functions for implicit geometries and PDE-based gradient limiting. *Eng. Comput-Germany.*, *22*(2), 95–109. https://doi.org/10.1007/s00366-006-0014-1

Persson, P.-O., & Strang, G. (2004). A simple mesh generator in MATLAB. *SIAM Rev.*, *46*(2), 329–345. https://doi.org/10.1137/s0036144503429121

Peterka, T., Morozov, D., & Phillips, C. (2014). High-performance computation of distributed-memory parallel 3D voronoi and delaunay tessellation. *Sc14: International Conference for High Performance Computing, Networking, Storage and Analysis*, 997–1007. https://doi.org/10.1109/sc.2014.86

Rathgeber, F., Ham, D. A., Mitchell, L., Lange, M., Luporini, F., Mcrae, A. T. T., Bercea, G.-T., Markall, G. R., & Kelly, P. H. J. (2017). Firedrake: Automating the finite element method by composing abstractions. *ACM Trans. Math. Softw.*, *43*(3), 1–27. https://doi.org/10.1145/2998441

Rineau, L. (2020). 2D conforming triangulations and meshes. In *CGAL user and reference manual* (5.1 ed.). CGAL Editorial Board. https://doc.cgal.org/5.1/Manual/packages.html#PkgMesh2

Roberts, K. (2020). Tutorial: Seismicmesh. In *ReadTheDocs website*. https://seismicmesh.readthedocs.io/en/master/tutorial.html; ReadTheDocs.

Roberts, K. J., Pringle, W. J., & Westerink, J. J. (2019). OceanMesh2D 1.0: Matlab-based software for two-dimensional unstructured mesh generation in coastal ocean modeling. *Geosci. Model Dev.*, *12*(5), 1847–1868. https://doi.org/10.5194/gmd-12-1847-2019

Tarantola, A. (1984). Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, *49*(8), 1259–1266. https://doi.org/10.1190/1.1441754

Tournois, J., Srinivasan, R., & Alliez, P. (2009). Perturbing slivers in 3D delaunay meshes. In *Proceedings of the 18th international meshing roundtable* (pp. 157–173). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04319-2_10

Virieux, J., & Operto, S. (2009). An overview of full-waveform inversion in exploration geophysics. *Geophysics*, *74*(6), WCC1–WCC26. https://doi.org/10.1190/1.3238367