




DelaunayTriangulation.jl: A Julia package for Delaunay triangulations and Voronoi tessellations in the plane

Daniel J. VandenHeuvel ¹✉

¹ Department of Mathematics, Imperial College London, UK ✉ Corresponding author

DOI: [10.21105/joss.07174](https://doi.org/10.21105/joss.07174)

Software

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

Editor: [Vissarion Fisikopoulos](#)  

Reviewers:

- [@PieterjanRobbe](#)
- [@mtsch](#)

Submitted: 27 June 2024

Published: 27 September 2024

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

DelaunayTriangulation.jl is a feature-rich Julia ([Bezanson et al., 2017](#)) package for computing Delaunay triangulations and Voronoi tessellations. The package, amongst many other features, supports unconstrained and constrained triangulations, mesh refinement, clipped and centroidal Voronoi tessellations, power diagrams, and dynamic updates. Thanks to the speed and genericity of Julia, the package is both performant and robust—making use of [AdaptivePredicates.jl](#) ([Churavy & VandenHeuvel, 2024](#); [Shewchuk, 1997](#)) and [ExactPredicates.jl](#) ([Lairez, 2024](#)) for computing predicates with robust arithmetic—while still allowing for generic representations of geometric primitives.

Given a set of points \mathcal{P} , a *Delaunay triangulation* is a subdivision of the convex hull of \mathcal{P} into triangles, with the vertices of the triangles coming from \mathcal{P} , constructed such that no triangle's circumcircle contains any point from \mathcal{P} in its interior ([Aurenhammer et al., 2013](#); [Cheng et al., 2013](#)). A *constrained Delaunay triangulation* extends this definition to additionally allow for edges \mathcal{E} and piecewise linear boundaries \mathcal{B} to be included, ensuring that each segment from \mathcal{E} and \mathcal{B} is an edge of some triangle and the boundaries of the domain come from \mathcal{B} ([Cheng et al., 2013](#)). For constrained Delaunay triangulations, the triangles must still obey the empty circumcircle property above, except for allowing a point p to be in a triangle T 's circumcircle if all line segments from T 's interior to p intersect a segment from \mathcal{E} or \mathcal{B} ([Cheng et al., 2013](#)). The boundaries \mathcal{B} may also be given as parametric curves, in which case they are discretised until they accurately approximate the curved boundary ([Gosselin, 2009](#)). A related geometric structure is the *Voronoi tessellation* that partitions the plane into convex polygons for each $p \in \mathcal{P}$ such that, for a given polygon, each point in that polygon is closer to the associated polygon's point than to any other $q \in \mathcal{P}$ ([Aurenhammer et al., 2013](#); [Cheng et al., 2013](#)). Weighted triangulations and power diagrams are generalisations of these structures that allow for the inclusion of weights associated with the points ([Cheng et al., 2013](#)). A more detailed description of these mathematical details can be found in the package's [documentation](#).

Statement of Need

Delaunay triangulations and Voronoi tessellations have applications in a myriad of fields. Delaunay triangulations have been used for point location ([Mücke et al., 1999](#)), solving differential equations ([Goliás & Dutton, 1997](#); [Ju et al., 2006](#)), path planning ([Yan et al., 2008](#)), etc. Voronoi tessellations are typically useful when there is some notion of *influence* associated with a point, and have been applied to problems such as geospatial interpolation ([Bobach, 2009](#)), image processing ([Du et al., 1999](#)), and cell biology ([Meyer-Hermann, 2008](#); [Wang et al., 2024](#)).

Several software packages with support for computing Delaunay triangulations and Voronoi

tessellations in two dimensions already exist, such as *Triangle* in C (Shewchuk, 1996), *MATLAB* (The MathWorks Inc., 2024), *SciPy* (Virtanen et al., 2020) in Python, *CGAL* (The CGAL Project, 2024) in C++, and *Gmsh* (Geuzaine & Remacle, 2009) which has interfaces in several languages. There are also other Julia packages supporting some of these features, although none is as developed as *DelaunayTriangulation.jl*; a more detailed comparison with these other software packages is given in *DelaunayTriangulation.jl*'s [README](#). *DelaunayTriangulation.jl* supports many features not present in most of these other software packages, such as power diagrams and the triangulation of curve-bounded domains, and benefits from the high-performance of Julia to efficiently support many operations. Julia's multiple dispatch (Bezanson et al., 2017) is leveraged to allow for complete customisation in how a user wishes to represent geometric primitives such as points and domain boundaries, a useful feature for allowing users to represent primitives in a way that suits their application without needing to sacrifice performance. The [documentation](#) lists many more features, including the package's ability to represent a wide range of domains, even those that are disjoint and with holes.

DelaunayTriangulation.jl has already seen use in several areas. *DelaunayTriangulation.jl* was used for mesh generation in VandenHeuvel et al. (2023) and is used for the *tricontourf*, *triplot*, and *voronoiplot* routines inside *Makie.jl* (Danisch & Krumbiegel, 2021). The packages *FiniteVolumeMethod.jl* (VandenHeuvel, 2024a) and *NaturalNeighbours.jl* (VandenHeuvel, 2024b) are also built directly on top of *DelaunayTriangulation.jl*.

Example

We give one example of how the package can be used, focusing on Delaunay triangulations rather than Voronoi tessellations. Many more examples are given in the [documentation](#), including [tutorials](#) and [fully detailed applications](#) such as cell simulations. To fully demonstrate the utility of the package, we consider a realistic application. We omit code used for plotting with *Makie.jl* (Danisch & Krumbiegel, 2021) in the example below for space reasons. The complete code is available [here](#).

We consider a domain motivated by mean exit time, relating to the time taken for a particle to reach a certain target, with applications to areas such as diffusive transport (Carr et al., 2022) and economics (Li, 2019). For example, mean exit time can be used to describe the expected time for a stock to reach a certain threshold (Li, 2019; Redner, 2001). Denoting the mean exit time of a particle at a point (x, y) by $T(x, y)$, one model describing the mean exit time of a particle exiting Ω with diffusivity D starting at (x, y) is given by (Carr et al., 2022; Redner, 2001)

$$\begin{aligned} D\nabla^2 T(x, y) &= -1 & (x, y) \in \Omega, \\ T(x, y) &= 0 & (x, y) \in \Gamma_a, \\ T(x, y) &= 0 & (x, y) = (x_s, y_s), \\ \nabla T(x, y) \cdot \hat{n}(x, y) &= 0 & (x, y) \in \Gamma_r. \end{aligned}$$

Here, $\hat{n}(x, y)$ is the unit normal vector field on Γ_r , $(x_s, y_s) = (0, 0)$, and the domain Ω with boundary $\partial\Omega = \Gamma_a \cup \Gamma_r$ is shown in [Figure 1\(a\)](#). This setup defines a mean exit time where the particle can only exit through Γ_a or through the sink (x_s, y_s) , and it gets reflected off of Γ_r .

The code to generate a mesh of the domain is given below. We use *CircularArcs* to define the boundary so that curve-bounded refinement can be applied using the algorithm of Gosselin (2009). The resulting mesh is shown in [Figure 1](#), together with a solution of the mean exit time problem with $D = 6.25 \times 10^{-4}$; *FiniteVolumeMethod.jl* (VandenHeuvel, 2024a) is used to solve this problem, and the code for this can be found [here](#).

```
# The outer circle
θ = 5π / 64
cs = θ -> (cos(θ), sin(θ))
p1, q1 = cs(-π / 2 - θ), cs(θ) # Absorbing
```

```
p2, q2 = q1, cs( $\pi$  / 2 -  $\theta$ )      # Reflecting
p3, q3 = q2, cs( $\pi$  +  $\theta$ )         # Absorbing
p4, q4 = q3, p1                  # Reflecting
c0 = (0.0, 0.0)
C01 = CircularArc(p1, q1, c0) # first, last, center
C02 = CircularArc(p2, q2, c0)
C03 = CircularArc(p3, q3, c0)
C04 = CircularArc(p4, q4, c0)
C0 = [[C01], [C02], [C03], [C04]]
# Inner circles
c1, p5 = (-0.4, -0.4), (-0.65, -0.65)
c2, p6 = (0.4, 0.4), (0.65, 0.65)
C1 = CircularArc(p5, p5, c1, positive=false) # Reflecting
C2 = CircularArc(p6, p6, c2, positive=false) # Reflecting
# Triangulate and refine
sink = (0.0, 0.0)
tri = triangulate([sink], boundary_nodes=[C0, [[C1]], [[C2]]])
refine!(tri; max_area=1e-3get_area(tri))
```

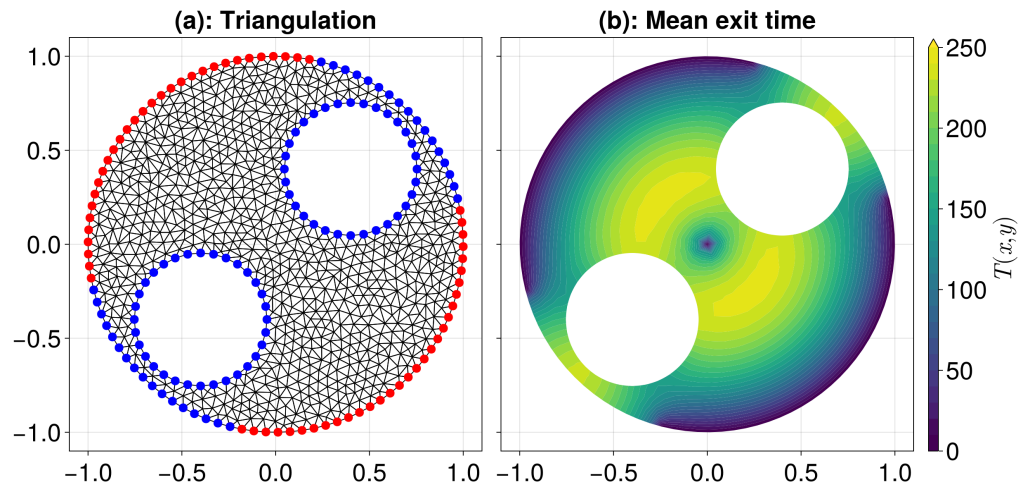


Figure 1: (a) The generated mesh using DelaunayTriangulation.jl for the mean exit time domain. The red dots along the boundary define the absorbing part of the boundary, Γ_a , and the blue dots define the reflecting part, Γ_r . (b) The solution to the mean exit time problem using the mesh from (a) together with FiniteVolumeMethod.jl (VandenHeuvel, 2024a).

References

- Aurenhammer, F., Klein, R., & Lee, D.-T. (2013). *Voronoi diagrams and Delaunay triangulations*. World Scientific. <https://doi.org/10.1142/8685>
- Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65–98. <https://doi.org/10.1137/141000671>
- Bobach, T. (2009). *Natural neighbor interpolation - critical assessment and new contributions* [PhD thesis]. Technische Universität Kaiserslautern.
- Carr, E., VandenHeuvel, D., Wilson, J., & Simpson, M. (2022). Mean exit time in irregularly-shaped annular and composite disc domains. *Journal of Physics A: Mathematical and Theoretical*, 55, 105002. <https://doi.org/10.1088/1751-8121/ac4a1d>

- Cheng, S.-W., Dey, T., & Shewchuk, J. (2013). *Delaunay mesh generation*. CRC Press. <https://www.routledge.com/Delaunay-Mesh-Generation/Cheng-Dey-Shewchuk/p/book/9781584887300>
- Churavy, V., & VandenHeuvel, D. (2024). AdaptivePredicates.jl: Port of Shewchuk's robust predicates into Julia. In *GitHub repository*. <https://github.com/JuliaGeometry/AdaptivePredicates.jl>; GitHub.
- Danisch, S., & Krumbiegel, J. (2021). Makie.jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software*, 6, 3349. <https://doi.org/10.21105/joss.03349>
- Du, Q., Faber, V., & Gunzburger, M. (1999). Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41, 637–676. <https://doi.org/10.1137/S0036144599352836>
- Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79, 1309–1331. <https://doi.org/10.1002/nme.2579>
- Golias, N., & Dutton, R. (1997). Delaunay triangulation and 3D adaptive mesh generation. *Finite Elements in Analysis and Design*, 25, 331–341. [https://doi.org/10.1016/S0168-874X\(96\)00054-6](https://doi.org/10.1016/S0168-874X(96)00054-6)
- Gosselin, S. (2009). *Delaunay refinement mesh generation of curve-bounded domains* [PhD thesis, The University of British Columbia]. <https://doi.org/10.14288/1.0067778>
- Ju, L., Gunzburger, M., & Zhao, W. (2006). Adaptive finite element methods for elliptic PDEs based on conforming centroidal Voronoi-Delaunay triangulations. *SIAM Journal on Scientific Computing*, 28, 2023–2053. <https://doi.org/10.1137/050643568>
- Lairez, P. (2024). ExactPredicates.jl: Fast and exact geometrical predicates in the Euclidean plane. In *GitHub repository*. <https://github.com/lairez/ExactPredicates.jl>; GitHub.
- Li, L. (2019). *First passage times of diffusion processes and their applications to finance* [PhD thesis]. London School of Economics and Political Science.
- Meyer-Hermann, M. (2008). Delaunay-object-dynamics: Cell mechanics with a 3D kinetic and dynamic weighted Delaunay triangulation. *Current Topics in Developmental Biology*, 81, 373–399. [https://doi.org/10.1016/S0070-2153\(07\)81013-1](https://doi.org/10.1016/S0070-2153(07)81013-1)
- Mücke, E., Saias, I., & Zhu, B. (1999). Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. *Computational Geometry*, 12, 63–83. [https://doi.org/10.1016/S0925-7721\(98\)00035-2](https://doi.org/10.1016/S0925-7721(98)00035-2)
- Redner, S. (2001). *A guide to first passage processes*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511606014>
- Shewchuk, J. (1996). Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In M. C. Lin & D. Manocha (Eds.), *Applied computational geometry: Towards geometric engineering* (Vol. 1148, pp. 203–222). Springer-Verlag. <https://doi.org/10.1007/BFb0014497>
- Shewchuk, J. (1997). Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete Computational Geometry*, 18, 305–363. <https://doi.org/10.1007/PL00009321>
- The CGAL Project. (2024). *CGAL user and reference manual* (5.6.1 ed.). CGAL Editorial Board. <https://doc.cgal.org/5.6.1/Manual/packages.html>
- The MathWorks Inc. (2024). *MATLAB version: R2024a*. The MathWorks Inc. <https://www.mathworks.com>
- VandenHeuvel, D. (2024a). *FiniteVolumeMethod.jl* (Version v1.1.3). Zenodo. <https://doi.org/10.5281/zenodo.11178646>

- VandenHeuvel, D. (2024b). *NaturalNeighbours.jl* (Version v1.3.2). Zenodo. <https://doi.org/10.5281/zenodo.11176971>
- VandenHeuvel, D., Devlin, B., Buenzli, P., Woodruff, M., & Simpson, M. (2023). New computational tools and experiments reveal how geometry affects tissue growth in 3D printed scaffolds. *Chemical Engineering Journal*, 475, 145776. <https://doi.org/10.1016/j.cej.2023.145776>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Wang, X., Jenner, A., Salomone, R., Warne, D., & Drovandi, C. (2024). Calibration of agent based models for monophasic and biphasic tumour growth using approximate Bayesian computation. *Journal of Mathematical Biology*, 88. <https://doi.org/10.1007/s00285-024-02045-4>
- Yan, H., Wang, H., Chen, Y., & Dai, G. (2008). Path planning based on constrained Delaunay triangulation. *2008 7th World Congress on Intelligent Control and Automation*, 5168–5173. <https://doi.org/10.1109/WCICA.2008.4593771>