

Scikit-Topt: A Python Library for Algorithm Development in Topology Optimization


Kohei Watanabe ¹

¹ AI Technology Office, Data Analytics R&D Dept., JTEKT

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

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

Editor: [Mojtaba Barzegari](#) 

Reviewers:

- [@aslan-ng](#)
- [@HaoLI-KU](#)

Submitted: 11 June 2025

Published: unpublished

License

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

Summary

Topology optimization is a computational method for optimizing the shape and material layout of structures and components, typically using the finite element method (FEM). It automatically generates optimal structural configurations by adjusting the distribution of material to achieve target performance—for example, minimizing compliance under a volume constraint. Specifically, the method optimizes an objective function by adjusting design variables that represent the presence or absence of material in each region.

This technique can be applied to a wide range of physical simulations, including structural analysis, heat conduction, fluid dynamics, and electromagnetic field analysis, making it a powerful tool for optimal design.

Scikit-topt is a Python library that implements topology optimization algorithms. It provides tools for visualizing parameter transitions and optimization results, thereby facilitating algorithm tuning and comparative analysis of optimization strategies. The library is tested via GitHub Actions and includes documentation generated with Sphinx. Installation and usage instructions are available in the [GitHub repository](#).

This article describes version **v0.2.5** of *scikit-topt*.

Statement of Need

There is a growing demand for accessible and flexible tools for topology optimization that support unstructured meshes and can be easily integrated into Python-based workflows. While many existing libraries focus on specific problems with rigid architectures or heavy dependencies, Scikit-topt offers a lightweight, modular, and fully Pythonic framework suitable for both academic research and industrial applications.

In topology optimization, the evolution of material density fields and the scheduling of algorithmic parameters (e.g., via continuation methods) play a crucial role in achieving high-quality results. However, most existing libraries provide limited or no support for managing and visualizing such transitions, making it difficult to interpret the optimization process.

Scikit-topt addresses this gap by enabling users to track and visualize the progression of density distributions and parameter schedules throughout the optimization process. This feature is particularly valuable for algorithm development, comparative studies, and educational purposes.

The project is under active development and welcomes community contributions.

Purpose and Prior Art

There are numerous open-source projects for topology optimization. Examples include FEni-Top(Jia et al., 2024), TopOpt.jl(Tarek, 2019), Topology optimization using PETSc(Smit et al., 2021), DL4TO(Erzmann et al., 2023), and pytopo3d(Kim & Kang, 2025). However, many projects are specialized implementations targeted at specific problems and are not well-suited for solving general topology optimization tasks. Common limitations include:

- Parameters and problem settings are hard-coded, limiting flexibility.
- They rely on grid- or voxel-based finite element analysis, making it difficult to handle arbitrary geometries.
- They are not packaged properly or depend on outdated software, making installation and usage difficult.
- They are not designed as standard experimental platforms, making it hard to compare algorithms across different parameters and case studies.

Scikit-topt avoids these issues by not including compiled code and depending only on Scipy(Virtanen et al., 2020), a widely used and stable Python library, making packaging and installation straightforward. For finite element matrix assembly, it uses Scikit-FEM(Gustafsson & McBain, 2020), which also contains no compiled components.

Furthermore, Scikit-topt visualizes parameter transitions and material density evolution through Matplotlib(Hunter, 2007) and PyVista(Sullivan & Kaszynski, 2019).

Scikit-topt implements topology optimization algorithms based on the density method, and currently supports the following features:

- Structural analysis and topology optimization using unstructured meshes
- Optimization based on the density method (SIMP / RAMP)
- Binartization acceleration using Heaviside projection
- Smoothing using the Helmholtz filter
- Gradient computation via backpropagation, combining the density method, Heaviside projection, and Helmholtz filter
- Optimization under multiple load cases
- Continuation for scheduling parameter transitions
- Includes FEA solvers powered by:
 - Scipy(Virtanen et al., 2020)
 - PyAMG(Bell et al., 2023)
 - scikit-fem(Gustafsson & McBain, 2020)
- Support for multiple optimization algorithms
 - Optimality Criteria Method
 - Modified Optimality Criteria (MOC) Variants
 - Log-space Update
 - Linear-space Update
 - Lagrangian Method

Usage Example

The following example demonstrates how to use Scikit-topt to perform topology optimization on an unstructured mesh using the Optimality Criteria (OC) method. First, a mesh is loaded using scikit-fem, and a TaskConfig object is defined with material properties and boundary conditions. Next, a configuration object for the optimizer is created, and the optimization process is launched. In this example, a TaskConfig is created from a mesh file using scikit-fem, which defines the FEM model, material properties, and boundary conditions. Then, OC_Config sets the optimization parameters such as volume fraction and iteration limits. Finally, the optimizer is executed.

```
import skfem, sktopt

mesh_path = "./data/model.msh"
basis = sktopt.mesh.loader.basis_from_file(mesh_path, intorder=2)

task = sktopt.mesh.task.TaskConfig.from_defaults(
    210e9, 0.30, basis, dirichlet_nodes...
)

cfg = sktopt.core.OC_Config(
    dst_path="./result",
    vol_frac=0.4,
    max_iter=100...
)

optimizer = sktopt.core.OC_Optimizer(cfg, task)

optimizer.parameterize()
optimizer.optimize()
```

83 During the optimization process, intermediate density fields are automatically visualized and
 84 saved using PyVista and Matplotlib.
 85 This enables users to monitor the convergence behavior and spatial evolution of the material
 86 distribution. An example is shown below. The left figure shows the boundary conditions, while
 87 the right shows the optimized density distribution:

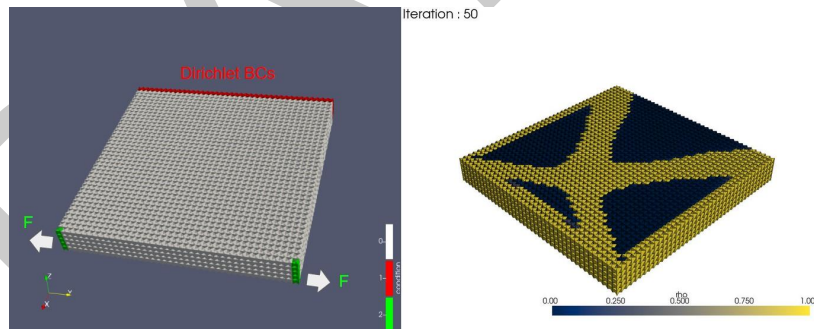


Figure 1: Topology optimization setup under multiple load cases (left) and resulting optimized density distribution (right)

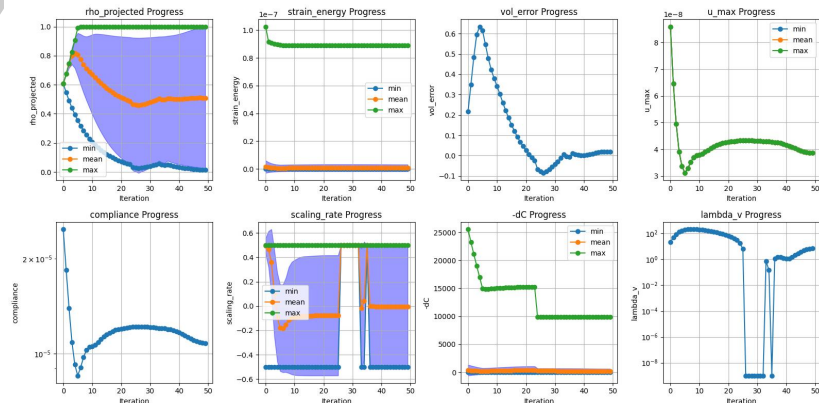


Figure 2: Parameter progression during optimization

88 As an extensible and open framework, Scikit-topo aims to facilitate reproducible experimentation
89 and accelerate the development of new topology optimization strategies.

90 Acknowledgements

91 I acknowledge the use of open-source tools and libraries that made this research possible.

92 References

- 93 Bell, N., Olson, L. N., Schroder, J., & Southworth, B. (2023). PyAMG: Algebraic multigrid
94 solvers in python. *Journal of Open Source Software*, 8(87), 5495. <https://doi.org/10.21105/joss.05495>
95
- 96 Erzmann, D., Dittmer, S., Harms, H., & Maaß, P. (2023). DL4TO: A deep learning library for
97 sample-efficient topology optimization. In *Geometric science of information* (pp. 543–551).
98 Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-38271-0_54
- 99 Gustafsson, T., & McBain, G. D. (2020). *kinnala/scikit-fem: Simple finite element assemblers*
100 *for Python* (Version 1.1.0). Zenodo. <https://doi.org/10.5281/zenodo.3862391>
- 101 Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/mcse.2007.55>
102
- 103 Jia, Y., Wang, C., & Zhang, X. S. (2024). FEniTop: A simple FEniCSx implementation
104 for 2D and 3D topology optimization supporting parallel computing. *Structural and*
105 *Multidisciplinary Optimization*, 67(8). <https://doi.org/10.1007/s00158-024-03818-7>
- 106 Kim, J., & Kang, N. (2025). PyTopo3D: A python framework for 3D SIMP-based topology
107 optimization. *arXiv Preprint arXiv:2504.05604*.
- 108 Smit, T., Aage, N., Ferguson, S. J., & Helgason, B. (2021). Topology optimization using
109 PETSc: A python wrapper and extended functionality. *Structural and Multidisciplinary*
110 *Optimization*, 64(6), 4343–4353. <https://doi.org/10.1007/s00158-021-03018-7>
- 111 Sullivan, C., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a
112 streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*,
113 4(37), 1450. <https://doi.org/10.21105/joss.01450>
- 114 Tarek, M. (2019). TopOpt.jl: An efficient and high-performance package for topology
115 optimization of continuum structures in the julia programming language. *Proceedings of*
116 *the 13th World Congress of Structural and Multidisciplinary Optimization*.
- 117 Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D.,
118 Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M.,
119 Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E.,
120 ... Vázquez-Baeza, Y. (2020). SciPy 1.0: Fundamental algorithms for scientific computing
121 in python. *Nature Methods*, 17(3), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>