

Efficient Polyhedral Gravity Modeling in Modern C++ and Python

Jonas Schuhmacher¹, Emmanuel Blazquez², Fabio Gratl¹, Dario Izzo², and Pablo Gómez²

¹ Chair for Scientific Computing, Technische Universität München, Arcisstraße 21, 80333 München, Germany ² Advanced Concepts Team, European Space Agency, European Space Research and Technology Centre (ESTEC), Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands

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

Software

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

Editor: [Dan Foreman-Mackey](#)

Reviewers:

- [@mikegrudic](#)
- [@santisoler](#)

Submitted: 23 October 2023

Published: unpublished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

Polyhedral gravity models are essential for modeling the gravitational field of irregular bodies, such as asteroids and comets. We present an open-source C++ library for the efficient, parallelized computation of a polyhedral gravity model following the line integral approach by Tsoulis (Tsoulis, 2012). A slim, easy-to-use Python interface using *pybind11* accompanies the library. The library is particularly focused on delivering high performance and scalability, which we achieve through vectorization and parallelization with *xsimd* and *thrust*, respectively. For example, the average evaluation of 1 out of 1000 randomly sampled points took 253 microseconds on a M1 Pro chip for the mesh of Eros consisting of 7374 vertices and 14744 faces (see downscaled to 10% in Figure 1 (Gaskell, 2008)). The library supports many common formats, such as *.stl*, *.off*, *.ply*, *.mesh* and *tetgen's .node* and *.face* (Hang, 2015). These properties make the application of this implementation straightforward to (re-)use in an arbitrary context.

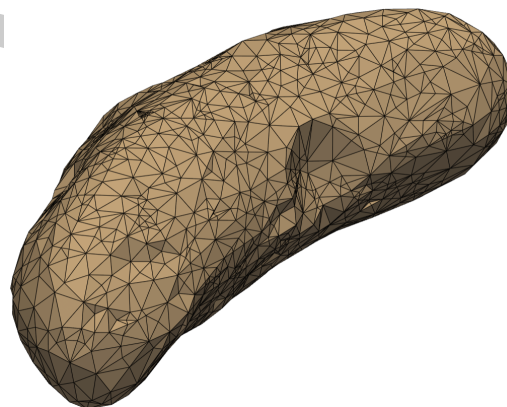


Figure 1: Downscaled mesh of (433) Eros to 10% of its original vertices and faces.

Statement of Need

The complex gravitational fields of irregular bodies, such as asteroids and comets, are often modeled using polyhedral gravity models since alternative approaches like mascon models or spherical harmonics struggle with these bodies' irregular geometry. The former struggles with convergence close to the surface (Šprlák & Han, 2021), whereas the latter requires a

26 computationally expensive amount of mascons (point masses of which the target body is
27 composed) to model fine-granular surface geometry (Wittick & Russell, 2017).

28 In contrast, polyhedral gravity models provide an analytic solution for the computation of the
29 gravitational potential, acceleration (and second derivative) given a mesh of the body (Tsoulis,
30 2012; Tsoulis & Gavrilidou, 2021) with the only assumption of homogeneous density. The
31 computation of the gravitational potential and acceleration is a computationally expensive
32 task, especially for large meshes, which can however benefit from parallelization either over
33 computed target points for which we seek potential and acceleration or over the mesh. Thus,
34 a high-performance implementation of a polyhedral gravity model is desirable.

35 While some research code for these models exists, they are not focused on usability and are lim-
36 ited to FORTRAN¹ and proprietary software like MATLAB². There is a lack of well-documented,
37 actively maintained open-source implementations, particularly in modern programming lan-
38 guages, and with a focus on scalability and performance.

39 This circumstance and the fact that polyhedral models are often used in studying gravitational
40 fields, e.g., for Eros (Zhang et al., 2010), or as a reference for creating new neural models
41 (Martin & Schaub, 2023) make an easy-to-install implementation necessary.

42 The presented software has already seen application in several research works. It has been used
43 to optimize trajectories around the highly irregular comet 67P/Churyumov-Gerasimenko with
44 the goal of maximizing the gravity signal (Marák et al., 2023) using pygmo (Biscani & Izzo,
45 2020). In the context of that work, the presented implementation was extended to enable
46 caching and even serialization to persistent memory on the C++ side. A change that enables
47 researchers to, e.g., efficiently propagate an orbit since the computation points can be given
48 apiece and do not need to be all known from the beginning.

49 Further, it has been used to study the effectiveness of so-called neural density fields (Izzo &
50 Gómez, 2022), where it served as ground truth to (pre-)train neural networks representing the
51 density distribution of an arbitrarily shaped body (Schuhmacher et al., 2023).

52 Thus, this model is highly versatile overall due to its easy-to-use API. It can be used in a
53 wide range of applications, especially due to the availability on major platforms like Windows,
54 macOS, and Linux for ARM64 and x86_64. We hope it will enable further research in the
55 field, especially related to recent machine-learning techniques, which typically rely on Python
56 implementations.

57 Polyhedral Model

58 On a mathematical level, the implemented model follows the line integral approach by Petrović
59 (Petrović, 1996) as refined by Tsoulis and Petrović (Tsoulis & Petrović, 2001). The associated
60 student report gives a comprehensive description of the mathematical foundations of the model
61 and how the gravitational triple integral is resolved to a double summation over the faces and
62 line segments of a polyhedron (Schuhmacher, 2022).

63 Implementation-wise, it makes use of the inherent parallelization opportunity of the approach
64 as it iterates over the mesh elements. This parallelization is achieved via *thrust*, which allows
65 utilizing *OpenMP* and *Intel TBB*. On a finer scale, individual, costly operations have been
66 investigated, and, e.g., the arctan operations have been vectorized using *xsimd*. On the
67 application side, the user can choose between the functional interface for evaluating the full
68 gravity tensor or the object-oriented *GravityEvaluable*, providing the same functionality while
69 implementing a caching mechanism to avoid recomputing mesh properties that can be shared
70 between multipoint evaluation, such as the face normals.

¹<https://software.seg.org/2012/0001/index.html>

²<https://github.com/Gavrilidou/GPolyhedron>

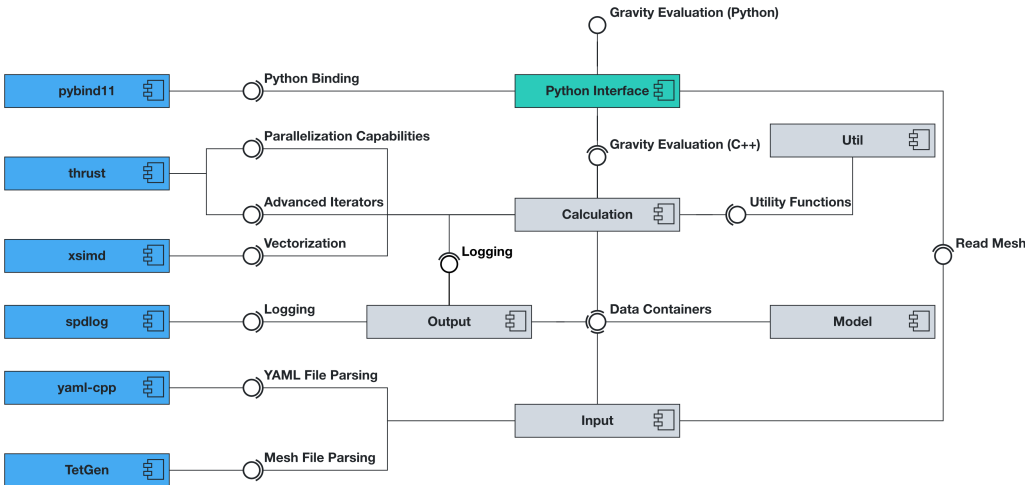


Figure 2: UML Component Diagram of the implementation. External dependencies are depicted in blue. Internal components are colored in grey.

Extensive tests using GoogleTest for the C++ side and pytest for the Python interface are employed via GitHub Actions to ensure the (continued) correctness of the implementation. Figure 2 summarizes the modular implementation and its dependencies in a UML component diagram.

Installation & Contribution

The library is available on GitHub³ and can be installed with `pip` (PyPi)⁴ or from `conda`⁵. Build instructions using `CMake` are provided in the repository. The library is licensed under a GPL license. The project is open to contributions via pull requests, with instructions on how to contribute provided in the repository.

Usage Instructions

We provide detailed usage instructions in the technical documentation on ReadTheDocs⁶. Additionally, a minimal working example is given in the repository readme, and more extensive examples, including a walkthrough over the available options as a `Jupyter` notebook⁷.

References

Biscani, F., & Izzo, D. (2020). A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53), 2338. <https://doi.org/10.21105/joss.02338>

Gaskell, R. W. (2008). *Eros polyhedral model*. <https://arcnv.psi.edu/urn:nasa:pds:gaskell.ast-eros.shape-model>.

³<https://github.com/esa/polyhedral-gravity-model>
⁴<https://pypi.org/project/polyhedral-gravity/>
⁵<https://anaconda.org/conda-forge/polyhedral-gravity-model>
⁶<https://polyhedral-gravity-model-cpp.readthedocs.io/en/latest/>
⁷<https://github.com/esa/polyhedral-gravity-model/blob/main/script/polyhedral-gravity.ipynb>

- 91 Hang, S. (2015). TetGen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans.*
92 *Math. Softw.*, 41(2), 11. <https://doi.org/10.1145/2629697>
- 93 Izzo, D., & Gómez, P. (2022). Geodesy of irregular small bodies via neural density fields.
94 *Communications Engineering*, 1(1), 48. <https://doi.org/10.1038/s44172-022-00050-3>
- 95 Marák, R., Blazquez, E., & Gómez, P. (2023). Trajectory optimization of a spacecraft
96 swarm orbiting around 67P/Churyumov-Gerasimenko. *Proceedings of the 9th International*
97 *Conference on Astrodynamics Tools and Techniques, ICATT*. [https://az659834.vo.msecnd.](https://az659834.vo.msecnd.net/eventsairwesteuprod/production-atpi-public/6f457b8c46dd40ab826a9160c3110b56)
98 [net/eventsairwesteuprod/production-atpi-public/6f457b8c46dd40ab826a9160c3110b56](https://az659834.vo.msecnd.net/eventsairwesteuprod/production-atpi-public/6f457b8c46dd40ab826a9160c3110b56)
- 99 Martin, J., & Schaub, H. (2023). The physics-informed neural network gravity model revisited:
100 Model generation III. *33rd AAS/AIAA Space Flight Mechanics Meeting, Austin, United*
101 *States*.
- 102 Petrović, S. (1996). Determination of the potential of homogeneous polyhedral bodies using
103 line integrals. *Journal of Geodesy*, 71, 44–52. <https://doi.org/10.1007/s001900050074>
- 104 Schuhmacher, J. (2022). *Efficient Polyhedral Gravity Modeling in Modern C++*. Technische
105 Universität München. <https://mediatum.ub.tum.de/doc/1695208/1695208.pdf>
- 106 Schuhmacher, J., Gratl, F., Izzo, D., & Gómez, P. (2023). Investigation of the robustness of
107 neural density fields. *Proceedings of the 12th International Conference on Guidance, Navi-*
108 *gation & Control Systems (GNC)*. [https://az659834.vo.msecnd.net/eventsairwesteuprod/](https://az659834.vo.msecnd.net/eventsairwesteuprod/production-atpi-public/b1e566f6284f4814b1c733ec08e4d136)
109 [production-atpi-public/b1e566f6284f4814b1c733ec08e4d136](https://az659834.vo.msecnd.net/eventsairwesteuprod/production-atpi-public/b1e566f6284f4814b1c733ec08e4d136)
- 110 Šprlák, M., & Han, S.-C. (2021). On the use of spherical harmonic series inside the minimum
111 brillouin sphere: Theoretical review and evaluation by GRail and LOLA satellite data.
112 *Earth-Science Reviews*, 222, 103739. <https://doi.org/10.1016/j.earscirev.2021.103739>
- 113 Tsoulis, D. (2012). Analytical computation of the full gravity tensor of a homogeneous
114 arbitrarily shaped polyhedral source using line integrals. *Geophysics*, 77(2), F1–F11.
115 <https://doi.org/10.1190/geo2010-0334.1>
- 116 Tsoulis, D., & Gavrilidou, G. (2021). A computational review of the line integral analytical
117 formulation of the polyhedral gravity signal. *Geophysical Prospecting*, 69(8–9), 1745–1760.
118 <https://doi.org/10.1111/1365-2478.13134>
- 119 Tsoulis, D., & Petrović, S. (2001). On the singularities of the gravity field of a homogeneous
120 polyhedral body. *Geophysics*, 66(2), 535–539. <https://doi.org/10.1190/1.1444944>
- 121 Wittick, P. T., & Russell, R. P. (2017). Mascon models for small body gravity fields. *AAS/AIAA*
122 *Astrodynamics Specialist Conference*, 162, 17–162.
- 123 Zhang, Z., Cui, H., Cui, P., & Yu, M. (2010). Modeling and analysis of gravity field of 433Eros
124 using polyhedron model method. *2010 2nd International Conference on Information*
125 *Engineering and Computer Science*, 1–4. <https://doi.org/10.1109/iciacs.2010.5677738>