The Journal of Open Source Software

Efficient Polyhedral Gravity Modeling in Modern C++ and Python

Jonas Schuhmacher [1], **Emmanuel Blazquez** [2], **Fabio Gratl** [1], **Dario Izzo** [2], **and Pablo Gómez** [2]

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

Software

- Review
- Repository
- Archive

Editor:

Submitted: 23 October 2023
Published: unpublished

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.

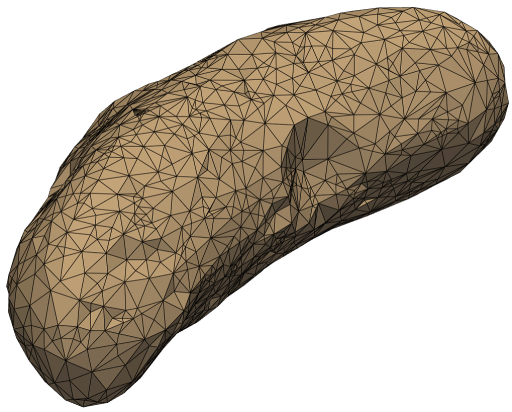

Figure 1: Downscaled mesh of (433) Eros to 10% of its original vertices and faces.

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

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

While some research code for these models exists, they are not focused on usability and are limited to FORTRAN[1] and proprietary software like MATLAB[2]. There is a lack of well-documented, actively maintained open-source implementations, particularly in modern programming languages, and with a focus on scalability and performance.

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

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

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

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

Polyhedral Model

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

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

[1] https://software.seg.org/2012/0001/index.html
[2] https://github.com/Gavriilidou/GPolyhedron

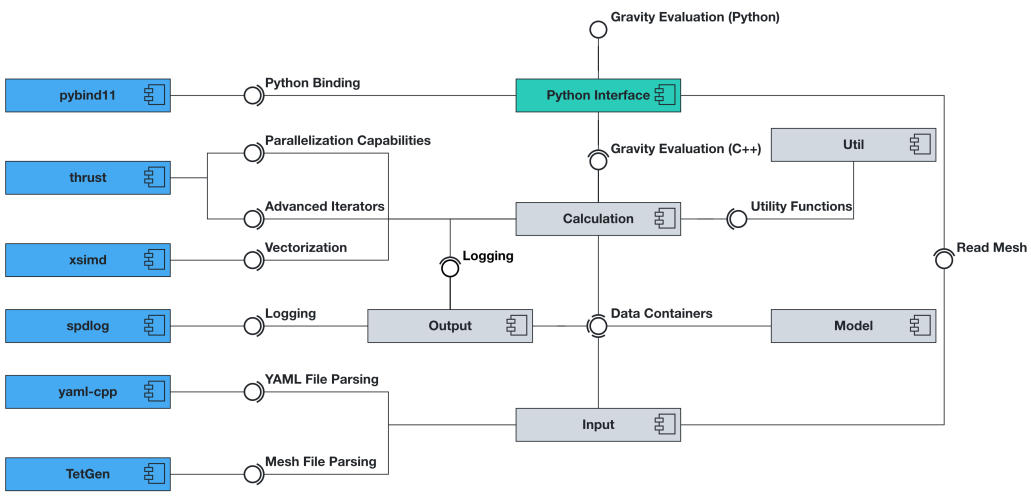

Figure 2: UML Component Diagram of the implementation. External dependencies are depicted in blue. Internal components are colored in grey.

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 GitHub[3] 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 ReadTheDocs[6]. 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* notebook[7].

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://arcnav.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

Hang, S. (2015). TetGen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw*, *41*(2), 11. https://doi.org/10.1145/2629697

Izzo, D., & Gómez, P. (2022). Geodesy of irregular small bodies via neural density fields. *Communications Engineering*, *1*(1), 48. https://doi.org/10.1038/s44172-022-00050-3

Marảk, R., Blazquez, E., & Gómez, P. (2023). Trajectory optimization of a spacecraft swarm orbiting around 67P/Churyumov-Gerasimenko. *Proceedings of the 9th International Conference on Astrodynamics Tools and Techniques, ICATT*. https://az659834.vo.msecnd.net/eventsairwesteuprod/production-atpi-public/6f457b8c46dd40ab826a9160c3110b56

Martin, J., & Schaub, H. (2023). The physics-informed neural network gravity model revisited: Model generation III. *33rd AAS/AIAA Space Flight Mechanics Meeting, Austin, United States*.

Petrović, S. (1996). Determination of the potential of homogeneous polyhedral bodies using line integrals. *Journal of Geodesy*, *71*, 44–52. https://doi.org/10.1007/s001900050074

Schuhmacher, J. (2022). *Efficient Polyhedral Gravity Modeling in Modern C++*. Technische Universität München. https://mediatum.ub.tum.de/doc/1695208/1695208.pdf

Schuhmacher, J., Gratl, F., Izzo, D., & Gómez, P. (2023). Investigation of the robustness of neural density fields. *Proceedings of the 12th International Conference on Guidance, Navigation & Control Systems (GNC)*. https://az659834.vo.msecnd.net/eventsairwesteuprod/production-atpi-public/b1e566f6284f4814b1c733ec08e4d136

Šprlák, M., & Han, S.-C. (2021). On the use of spherical harmonic series inside the minimum brillouin sphere: Theoretical review and evaluation by GRAIL and LOLA satellite data. *Earth-Science Reviews*, *222*, 103739. https://doi.org/10.1016/j.earscirev.2021.103739

Tsoulis, D. (2012). Analytical computation of the full gravity tensor of a homogeneous arbitrarily shaped polyhedral source using line integrals. *Geophysics*, *77*(2), F1–F11. https://doi.org/10.1190/geo2010-0334.1

Tsoulis, D., & Gavriilidou, G. (2021). A computational review of the line integral analytical formulation of the polyhedral gravity signal. *Geophysical Prospecting*, *69*(8-9), 1745–1760. https://doi.org/10.1111/1365-2478.13134

Tsoulis, D., & Petrović, S. (2001). On the singularities of the gravity field of a homogeneous polyhedral body. *Geophysics*, *66*(2), 535–539. https://doi.org/10.1190/1.1444944

Wittick, P. T., & Russell, R. P. (2017). Mascon models for small body gravity fields. *AAS/AIAA Astrodynamics Specialist Conference*, *162*, 17–162.

Zhang, Z., Cui, H., Cui, P., & Yu, M. (2010). Modeling and analysis of gravity field of 433Eros using polyhedron model method. *2010 2nd International Conference on Information Engineering and Computer Science*, 1–4. https://doi.org/10.1109/iciecs.2010.5677738