The Journal of Open Source Software

PyMOCAT-MC: A Python Implementation of the MIT Orbital Capacity Assessment Toolbox Monte Carlo Module

Rushil Kukreja [1], **Edward J. Oughton** [2], **Giovanni Lavezzi** [3], **Enrico M. Zucchelli** [3], **Daniel Jang** [4], **and Richard Linares** [3]

1 Thomas Jefferson High School for Science and Technology, Alexandria, VA, USA **2** Department of Geography and Geoinformation Science, George Mason University, Fairfax, VA, USA **3** Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA **4** Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

The growth of satellite deployments and large-scale constellations has made orbital sustainability an urgent concern (Bastida Virgili et al., 2016; Lemmens et al., 2020; Lewis, 2020). Simulating the evolution of the space environment requires tools that are both computationally efficient and compatible with modern research workflows. The MIT Orbital Capacity Assessment Toolbox Monte Carlo module (MOCAT-MC) (Jang et al., 2025) is a leading framework for modeling space traffic and debris risk, but its MATLAB implementation limits accessibility due to licensing requirements and hinders integration with state-of-the-art machine learning algorithms and Python-based data science pipelines central to contemporary space sustainability research.

PyMOCAT-MC is a complete Python reimplementation of MOCAT-MC, designed to maintain full functional compatibility with the original toolbox while improving performance, modularity, and accessibility. The translation involved converting more than 150 MATLAB functions into Python, preserving the core algorithms while restructuring them for clarity and efficiency within Python's scientific computing ecosystem. The resulting codebase comprises over 8,600 lines in Python and leverages open-source libraries including NumPy, SciPy, pandas, and Matplotlib.

Benchmarking against the MATLAB version shows that PyMOCAT-MC achieves a mean relative error of 0.96% and a maximum error of 2.38% across all tested scenarios. Performance tests demonstrate speed gains of up to four times, with the "Realistic Launch" scenario completing in under 30 seconds in Python compared to over 75 seconds in MATLAB. This combination of high accuracy and faster execution enables more extensive simulation campaigns and facilitates integration into a wider range of research and policy workflows.

Statement of need

As the orbital environment becomes more congested (Kukreja et al., 2025), the ability to model collisions, debris evolution, and the long-term sustainability of satellite operations has significant implications for both policy-making and engineering design. The original MOCAT-MC toolbox (ARCLab-MIT, 2025) has been used by researchers to perform Monte Carlo–based assessments of orbital capacity, yet its MATLAB implementation limits accessibility for researchers and organizations that rely on open-source tools. It also presents barriers to integrating with state-of-the-art machine learning frameworks and Python-based data analysis workflows, which are increasingly central to advancing space situational awareness modeling. These constraints

can hinder collaboration, slow down simulation work, and restrict the adoption of the software in the broader space sustainability community.

PyMOCAT-MC addresses these barriers by delivering a functionally equivalent, open-source Python version that is free to use and modify. The Python implementation not only replicates the original capabilities but also improves runtime performance and introduces a modular design that makes it easier to integrate with other orbital mechanics packages such as Astropy or poliastro, as well as related tools like MOCAT-pySSEM (Brownhall et al., 2025). The open-source nature of the project supports reproducibility, encourages community contributions, and creates opportunities for extending the toolbox to new modeling approaches, such as agent-based simulations of satellite behavior. In the broader landscape, PyMOCAT-MC complements established environment models such as ESA's MASTER and related frameworks (Flegel et al., 2012).

Methodology

The development of PyMOCAT-MC began with a careful analysis of the MATLAB source code to understand the data structures, algorithms, and dependencies used in MOCAT-MC. Each function was translated individually into Python, maintaining the mathematical and algorithmic logic while adopting Pythonic conventions for readability and maintainability. Where appropriate, vectorized operations and optimized data handling were introduced to enhance computational performance without altering the results. Atmospheric modeling considerations, critical for accurate orbit propagation (Ding et al., 2023; Emmert, 2015), were preserved in the translation.

Validation was an integral part of the translation process. Unit tests were created to verify the correctness of individual functions, while side-by-side comparisons of MATLAB and Python simulation outputs were performed for each standard scenario. These comparisons ensured that differences in results remained within acceptable numerical tolerances. Manual code reviews were also conducted to confirm fidelity to the original design.

The repository is organized to facilitate both research use and further development. The `mocat_mc.py` module contains the main Monte Carlo simulation engine, supported by additional modules for orbital propagation, collision detection, atmospheric drag modeling, and debris generation. Continuous integration workflows were implemented to automatically run tests and verify that code changes preserve numerical fidelity across all benchmark scenarios. Example scripts demonstrate common simulation scenarios, including baseline, no-launch, and realistic-launch cases. Comparison scripts and performance analysis tools are included to reproduce the accuracy and speed results reported here. Supporting data, such as historical two-line element (TLE) sets, the JB2008 atmospheric density model, and launch schedules for megaconstellations, are bundled with the software to enable immediate use.

Results

Across all tested scenarios, PyMOCAT-MC reproduces the results of the MATLAB implementation with high fidelity. Differences in total object counts between the two implementations are small, with a maximum deviation of 150 objects out of approximately 13,700.

In addition to matching the accuracy of MATLAB, the Python version delivers substantial performance gains. The most computationally demanding scenario, which includes realistic launch patterns for megaconstellations, runs in less than 29.95 with PyMOCAT-MC compared to 75.02 seconds in MATLAB, representing a speed-up larger than a factor of two.

Error analysis confirms that the differences between MATLAB and Python remain minimal across simulation years and object types. The heatmap highlights that relative errors are

uniformly low; the mean relative error is 0.96%, and the maximum error across object types and time horizons is 2.38%.

These improvements reduce the time required for large simulation batches, making it feasible to explore a wider range of parameters and run more detailed sensitivity analyses.

References

ARCLab-MIT. (2025). *MOCAT-MC*. https://github.com/ARCLab-MIT/MOCAT-MC.

Bastida Virgili, B., Dolado, J.-C., Lewis, H. G., Radtke, J., Krag, H., Meadows, P., Rossi, A., Valsecchi, G. B., & Colombo, C. (2016). Risk to space sustainability from large constellations in low earth orbit. *Acta Astronautica*, *126*, 154–162. https://doi.org/10.1016/j.actaastro.2016.03.034

Brownhall, I., Lifson, M., Hall, S., Constant, C., Lavezzi, G., Ziebart, M., Linares, R., & Bhattarai, S. (2025). MOCAT-pySSEM: An open-source python library and user interface for orbital debris and source sink environmental modeling. *SoftwareX*, *30*, 102062. https://doi.org/10.1016/j.softx.2025.102062

Ding, Y., Li, Z., Liu, C., Kang, Z., Sun, M., Sun, J., & Chen, L. (2023). Analysis of the impact of atmospheric models on the orbit prediction of space debris. *Sensors*, *23*(21), 8993. https://doi.org/10.3390/s23218993

Emmert, J. T. (2015). Thermospheric mass density: A review. *Advances in Space Research*, *56*(5), 773–824. https://doi.org/10.1016/j.asr.2015.05.038

Flegel, S., Gelhaus, J., M"ockel, M., S"utterlin, P., Wiedemann, C., & Kempf, D. (2012). The MASTER-2009 space debris environment model. *Acta Astronautica*, *81*, 65–71. https://doi.org/10.1016/j.actaastro.2012.06.009

Jang, D., Gusmini, D., Siew, P. M., D'Ambrosio, A., Servadio, S., Machuca, P., & Linares, R. (2025). New monte carlo model for the space environment. *Journal of Spacecraft and Rockets*, 1–22. https://doi.org/10.2514/1.A36137

Kukreja, R., Oughton, E. J., & Linares, R. (2025). Greenhouse gas (GHG) emissions poised to rocket: Modeling the environmental impact of LEO satellite constellations. *arXiv*. https://doi.org/10.48550/arXiv.2504.15291

Lemmens, S., Hoots, F. R., Krag, H., & Flohrer, T. (2020). Collision risk in low earth orbit in the presence of large constellations. *Acta Astronautica*, *170*, 501–510. https://doi.org/10.1016/j.actaastro.2020.02.003

Lewis, H. G. (2020). The kessler syndrome: 40 years on. *Acta Astronautica*, *170*, 421–435. https://doi.org/10.1016/j.actaastro.2020.01.009