




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

Rushil Kukreja <sup>1</sup>, Edward J. Oughton <sup>1</sup>, Giovanni Lavezzi <sup>2</sup>, Enrico M. Zucchelli <sup>2</sup>, Daniel Jang <sup>3</sup>, and Richard Linares <sup>2</sup>

<sup>1</sup> Department of Geography and Geoinformation Science, George Mason University, Fairfax, VA, USA  
<sup>2</sup> Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA  
<sup>3</sup> Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, USA

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

## Software

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

Editor: 

Submitted: 05 September 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Published: unpublished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](https://creativecommons.org/licenses/by/4.0/)).

## Summary

The growth of satellite deployments and large-scale constellations has made orbital sustainability an urgent concern ([Bastida Virgili et al., 2016](#); [Lavezzi et al., 2025](#); [Lemmens et al., 2020](#); [Lewis, 2020](#)), and also has downstream economic impacts ([Oughton et al., 2024](#)). Simulating the space environment requires computationally efficient tools 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 restricts accessibility due to licensing requirements and limits integration with Python-based machine learning and data science workflows.

PyMOCAT-MC is a complete Python reimplementa-tion of MOCAT-MC, maintaining full compatibility while improving performance and accessibility. The translation involves converting over 150 MATLAB functions into Python, preserving core algorithms while restructuring for Python's scientific computing ecosystem. The codebase comprises over 8,600 lines and leverages NumPy, SciPy, pandas, and Matplotlib.

Benchmarking shows PyMOCAT-MC achieves a mean relative error of 0.63% and maximum error of 1.96% across all scenarios. Performance tests demonstrate up to 4× speed gains, with the most demanding scenario completing in 30 seconds versus 75 seconds in MATLAB, enabling more extensive simulation campaigns.

## Statement of need

As the orbital environment becomes more congested ([Kukreja et al., 2025](#)), modeling collisions and debris evolution has significant implications for policy and engineering. The original MOCAT-MC toolbox ([ARCLab-MIT, 2025](#)) is widely used for Monte Carlo-based orbital capacity assessments, yet its MATLAB implementation limits accessibility for open-source users and hinders integration with Python-based workflows, restricting adoption in the broader space sustainability community.

PyMOCAT-MC addresses these barriers with a functionally equivalent, open-source Python version. The implementation improves runtime performance with a modular design that integrates easily with packages like Astropy, poliastro, and MOCAT-pySSEM ([Brownhall et al., 2025](#)). The open-source nature supports reproducibility, community contributions, and extensions to new modeling approaches. PyMOCAT-MC fills a unique niche alongside established models like ESA's MASTER ([Flegel et al., 2012](#)), offering researchers an accessible, high-performance tool for Monte Carlo-based orbital capacity assessments.

## Methodology

Development began with analysis of the MATLAB source to understand data structures and algorithms. Each function was translated individually, maintaining algorithmic logic while adopting Pythonic conventions. Vectorized operations were introduced to enhance performance without altering results. Atmospheric modeling considerations (Ding et al., 2023; Emmert, 2015) were preserved.

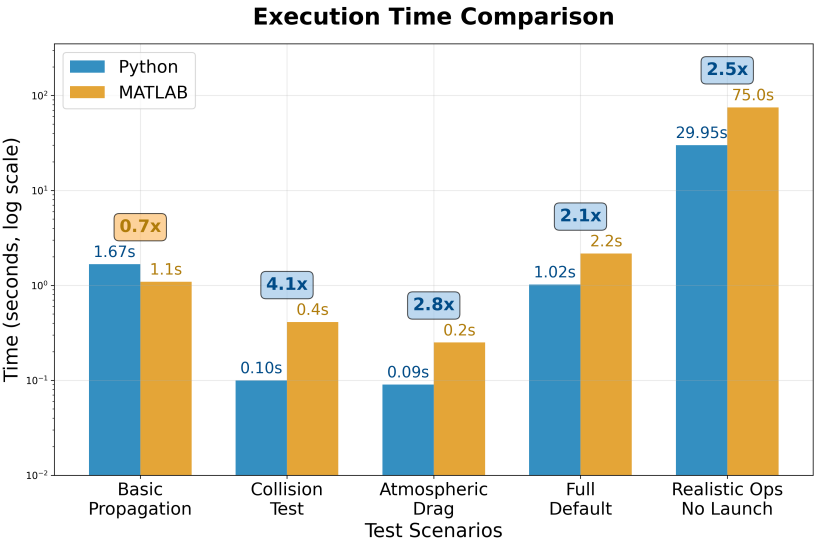
Validation followed a rigorous multi-tiered approach: unit tests verified individual function outputs against MATLAB equivalents, integration tests compared intermediate results at each simulation timestep, and end-to-end validation assessed complete scenario outputs across multiple random seeds. Side-by-side comparisons of orbital element distributions, collision events, and fragmentation patterns ensured differences remained within numerical tolerances (typically  $< 1\%$ ), consistent with established approaches for robust uncertainty quantification in orbit determination (Zucchelli et al., 2021).

The repository includes the main `mocat_mc.py` simulation engine with modules for orbital propagation, collision detection, atmospheric drag, and debris generation. Continuous integration ensures numerical fidelity across benchmark scenarios. Example scripts, comparison tools, and supporting data (historical TLEs, JB2008 atmospheric model, megaconstellation launch schedules) are bundled for immediate use.

## Results

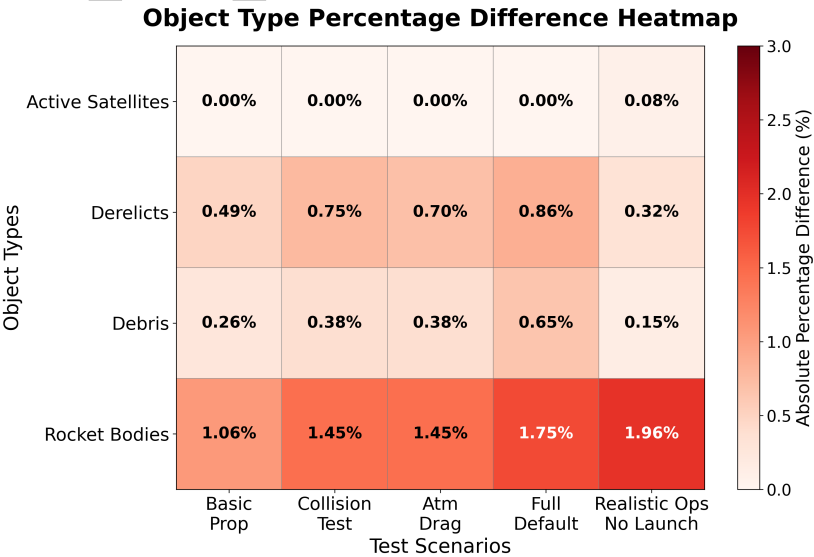
Across all tested scenarios, PyMOCAT-MC reproduces the results of the MATLAB implementation with high fidelity. Testing included five benchmark scenarios (Basic Propagation, Collision Test, Atmospheric Drag, Full Default, and Realistic Operations No Launch) using identical random seeds between implementations to ensure deterministic comparison. All simulations were conducted over 100 years to assess long-term orbital evolution. Differences in final total object counts between the two implementations are small, with a maximum deviation of 123 objects out of approximately 13,700 objects (Full Default scenario after 1 year), representing less than 1% error.

In addition to matching the accuracy of MATLAB, the Python version delivers substantial performance gains, as shown in Figure 1. The most computationally demanding scenario, which includes realistic launch patterns for megaconstellations, runs in less than 29.95 seconds with PyMOCAT-MC compared to 75.02 seconds in MATLAB, representing a speed-up larger than a factor of two. These improvements reduce the time required for large simulation batches, enabling exploration of a wider range of parameters and more detailed sensitivity analyses.

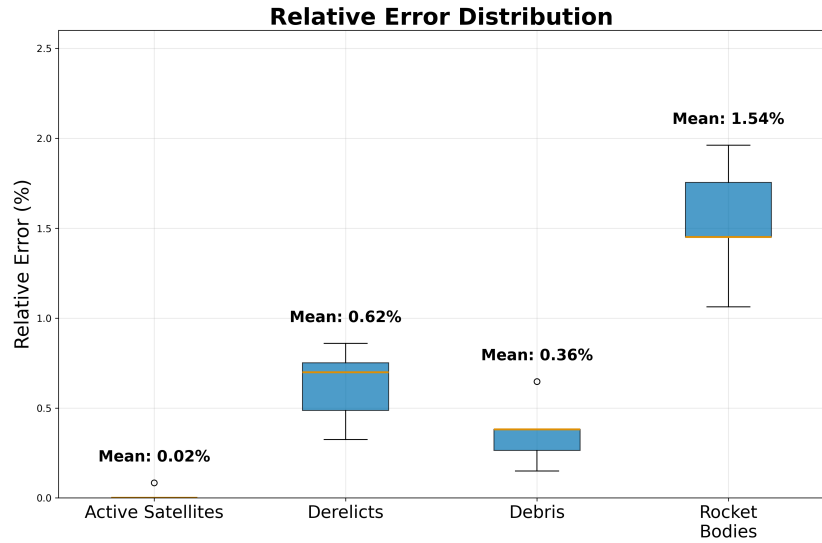


**Figure 1:** Runtime comparison between MATLAB and Python implementations for multiple scenarios, showing up to a 4× improvement in computational speed for PyMOCAT-MC.

74 Error analysis confirms that the differences between MATLAB and Python remain minimal  
75 across test scenarios and object types. Figure 2 shows that end-of-simulation relative errors  
76 are uniformly low across all object types and test scenarios, with a mean relative error of  
77 0.63% and a maximum error of 1.96%. Figure 3 provides additional detail through box plots,  
78 demonstrating that errors for active satellites and debris are consistently lower than 0.65%.  
79 These sub-1% differences are negligible compared to the variance typically observed between  
80 different orbital evolution models (Jang et al., 2025).

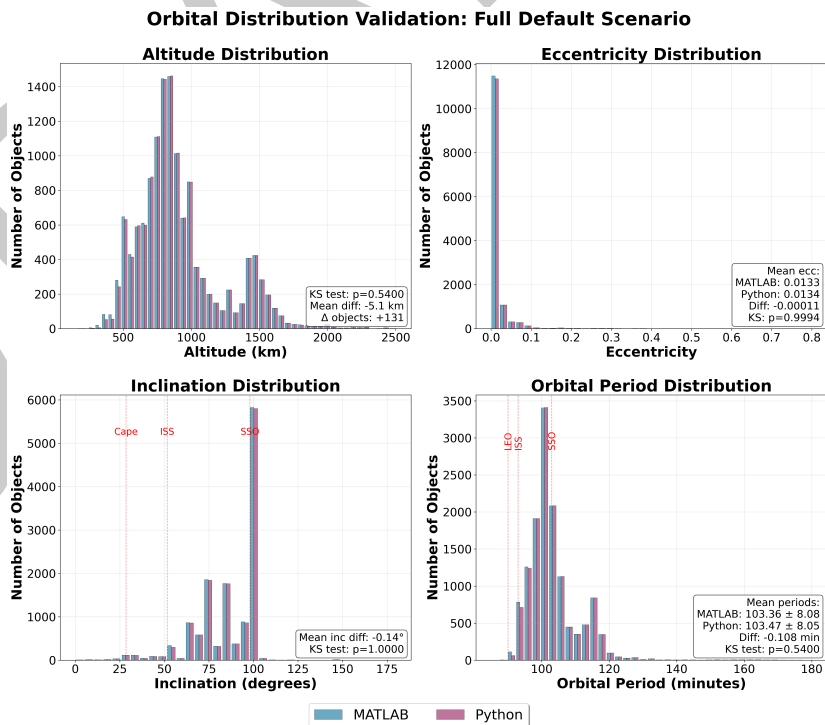


**Figure 2:** Heatmap of relative error (%) across test scenarios and object types, showing end-of-simulation accuracy of the Python implementation compared to MATLAB.



**Figure 3:** Relative error distribution box plots across object classes, showing PyMOCAT-MC achieves near-zero error for active satellites and consistently low error across all categories.

81 Figure 4 compares orbital element distributions between MATLAB and Python implementations.  
 82 Statistical validation using Kolmogorov-Smirnov tests shows excellent agreement (all p-values  
 83 > 0.05), confirming identical orbital mechanics implementation.



**Figure 4:** MATLAB vs Python orbital element distribution comparison showing altitude, eccentricity, inclination, and orbital period distributions.

84 With this validated accuracy and improved performance, PyMOCAT-MC can be applied to  
 85 test how different launch strategies influence orbital sustainability, evaluate debris mitigation

86 policies, and generate scenario libraries that inform both engineering design and regulatory  
87 decision-making.

## 88 References

- 89 ARCLab-MIT. (2025). *MOCAT-MC*. <https://github.com/ARCLab-MIT/MOCAT-MC>.
- 90 Bastida Virgili, B., Dolado, J.-C., Lewis, H. G., Radtke, J., Krag, H., Meadows, P., Rossi,  
91 A., Valsecchi, G. B., & Colombo, C. (2016). Risk to space sustainability from large  
92 constellations in low earth orbit. *Acta Astronautica*, 126, 154–162. [https://doi.org/10.](https://doi.org/10.1016/j.actaastro.2016.03.034)  
93 [1016/j.actaastro.2016.03.034](https://doi.org/10.1016/j.actaastro.2016.03.034)
- 94 Brownhall, I., Lifson, M., Hall, S., Constant, C., Lavezzi, G., Ziebart, M., Linares, R.,  
95 & Bhattarai, S. (2025). MOCAT-pySSEM: An open-source python library and user  
96 interface for orbital debris and source sink environmental modeling. *SoftwareX*, 30, 102062.  
97 <https://doi.org/10.1016/j.softx.2025.102062>
- 98 Ding, Y., Li, Z., Liu, C., Kang, Z., Sun, M., Sun, J., & Chen, L. (2023). Analysis of the  
99 impact of atmospheric models on the orbit prediction of space debris. *Sensors*, 23(21),  
100 8993. <https://doi.org/10.3390/s23218993>
- 101 Emmert, J. T. (2015). Thermospheric mass density: A review. *Advances in Space Research*,  
102 56(5), 773–824. <https://doi.org/10.1016/j.asr.2015.05.038>
- 103 Flegel, S., Gelhaus, J., M"ockel, M., S"utterlin, P., Wiedemann, C., & Kempf, D. (2012).  
104 The MASTER-2009 space debris environment model. *Acta Astronautica*, 81, 65–71.  
105 <https://doi.org/10.1016/j.actaastro.2012.06.009>
- 106 Jang, D., Gusmini, D., Siew, P. M., D'Ambrosio, A., Servadio, S., Machuca, P., & Linares,  
107 R. (2025). New monte carlo model for the space environment. *Journal of Spacecraft and*  
108 *Rockets*, 1–22. <https://doi.org/10.2514/1.A36137>
- 109 Kukreja, R., Oughton, E. J., & Linares, R. (2025). Greenhouse gas (GHG) emissions poised  
110 to rocket: Modeling the environmental impact of LEO satellite constellations. *arXiv*.  
111 <https://doi.org/10.48550/arXiv.2504.15291>
- 112 Lavezzi, G., Jang, D., & Linares, R. (2025). Space sustainability and large constellations. *Acta*  
113 *Astronautica*. <https://doi.org/10.1016/j.actaastro.2025.03.662>
- 114 Lemmens, S., Hoots, F. R., Krag, H., & Flohrer, T. (2020). Collision risk in low earth  
115 orbit in the presence of large constellations. *Acta Astronautica*, 170, 501–510. <https://doi.org/10.1016/j.actaastro.2020.02.003>
- 117 Lewis, H. G. (2020). The kessler syndrome: 40 years on. *Acta Astronautica*, 170, 421–435.  
118 <https://doi.org/10.1016/j.actaastro.2020.01.009>
- 119 Oughton, E. J., Renton, A., Mac Marnus, D., & Rodger, C. J. (2024). Assessing the economic  
120 benefits of space weather mitigation investment decisions: Evidence from aotearoa new  
121 zealand. *arXiv*. <https://doi.org/10.48550/arXiv.2507.12495>
- 122 Zucchelli, E. M., Delande, E. D., Jones, B. A., & Jah, M. K. (2021). Uncertainty quantification  
123 in orbit determination. *The Journal of the Astronautical Sciences*. [https://doi.org/10.](https://doi.org/10.1007/s40295-021-00267-y)  
124 [1007/s40295-021-00267-y](https://doi.org/10.1007/s40295-021-00267-y)