The Journal of Open Source Software

pyGeo: A geometry package for multidisciplinary design optimization

Hannah Hajdik [1], **Anil Yildirim** [1], **Neil Wu** [1], **Ben Brelje** [1], **Sabet Seraj** [1], **Marco Mangano** [1], **Josh Anibal** [1], **Eirikur Jonsson** [1], **Eytan Adler** [1], **Charles A. Mader** [1], **Gaetan Kenway** [1], and **Joaquim R. R. A. Martins** [1]

1 Department of Aerospace Engineering, University of Michigan

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

In shape optimization, an algorithm modifies a body's geometry to improve its performance. A common shape optimization example is adjusting the geometry of an aircraft wing to minimize aerodynamic drag computed via computational fluid dynamics (CFD). Multidisciplinary design optimization (MDO) couples multiple disciplines, such as aerodynamics and structural mechanics, to optimize them simultaneously. In MDO, the geometry must be represented consistently across multiple disciplines.

pyGeo is a geometry package for three-dimensional shape manipulation tailored for aerodynamic and multidisciplinary design optimization. It provides several methods for geometry parameterization, geometric constraints, and utility functions for geometry manipulation. pyGeo computes derivatives for all parameterization methods and constraints, facilitating efficient gradient-based optimization.

Features

Integrations

pyGeo is the geometry manipulation engine within the MDO of Aircraft Configurations at High Fidelity (MACH) framework (Kenway et al., 2014; Kenway & Martins, 2014), which specializes in high-fidelity aerostructural optimization. pyGeo is also integrated into MPhys[1], a more general framework for high-fidelity multiphysics problems built with OpenMDAO (Gray et al., 2019). Both MACH and MPhys use pyOptSparse (Wu et al., 2020) to interface with optimization algorithms.

pyGeo's interface for design variables and constraints is independent of which solvers are accessing the geometry. This means that pyGeo geometries can interact with different types of solvers, such as structures and aerodynamics, in the same way. This also allows direct comparison of the behavior or performance of two different solvers within the same discipline using the same geometric parameterization for each, such as two different flow solvers (Adler et al., 2022).

Geometry Parameterization with pyGeo

pyGeo contains several options for parameterizing geometry: variations on the FFD method, interfaces to external parametric modeling tools, and an analytic parameterization. Because

[1] https://github.com/OpenMDAO/mphys

each parameterization method uses a common interface for interacting with the rest of the MACH framework, any surface parameterization can be used in place of another within an optimization setup (Hajdik et al., 2023). The choice of parameterization depends on the user's experience, the geometry details, and whether the user needs the final design in a specific format.

Free-form Deformation

The free-form deformation (FFD) method (Sederberg & Parry, 1986) is one of the most popular three-dimensional geometry parameterization approaches (Zhang et al., 2018). This approach embeds the entire reference geometry in a parameterized volume. The set of control points that determine the shape of the volume are displaced to manipulate the points inside. The user can have a high degree of control over the geometry by selecting different control point densities and locations.

Individual control points can be moved to obtain local shape modifications. In pyGeo, these are referred to as *local* design variables because a single control point is affected. Conversely, it is also common to define geometric operations involving a collection of control points across the entire FFD block. These are referred to as *global* design variables in pyGeo. For example, wing twist variables can be defined as rotations of the control points about a reference axis that runs along the wing. Figure 1 shows a few common planform design variables for an aircraft wing.

Design variables formulated from groupings of FFD control points often exhibit ill conditioning. To alleviate this, a parameterization based on singular value decomposition is also possible within pyGeo (Wu et al., 2022).

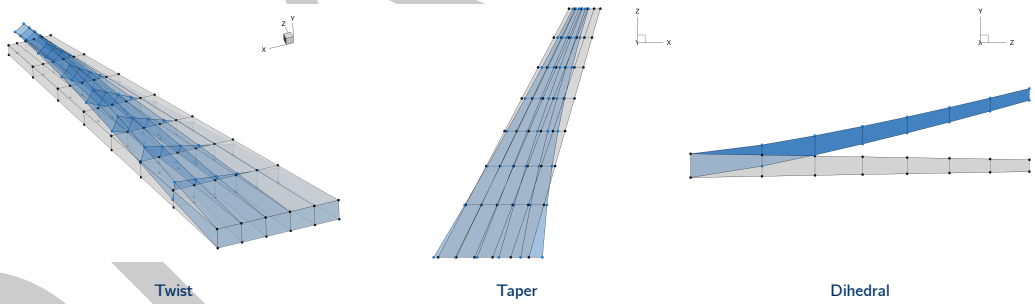

Figure 1: Examples of common wing planform design variables.

In addition to the basic FFD implementation, pyGeo offers two additional features: hierarchical FFD and multi-component FFD.

Hierarchical FFD

FFD objects can be organized in a hierarchical structure within pyGeo. Dependent, "child" FFD blocks can be embedded in the main, "parent" FFD block to enable modifications on a subset of the full geometry. pyGeo first propagates the parent deformations to both the geometry and the child control points and then propagates the deformations of the child control points to their subset of the geometry. One of the advantages of using this approach is that each FFD block can have its own independent reference axis to be used for global design variables such as rotations and scaling. This has facilitated the definition of control surface deflections (Lyu & Martins, 2014) and hydrofoil design (Liao et al., 2021). Figure 2 from the former paper shows a case where the parent FFD block is used to manipulate the shape of an entire wing of a blended wing body aircraft while the control surface on that wing is deformed using a child FFD block.

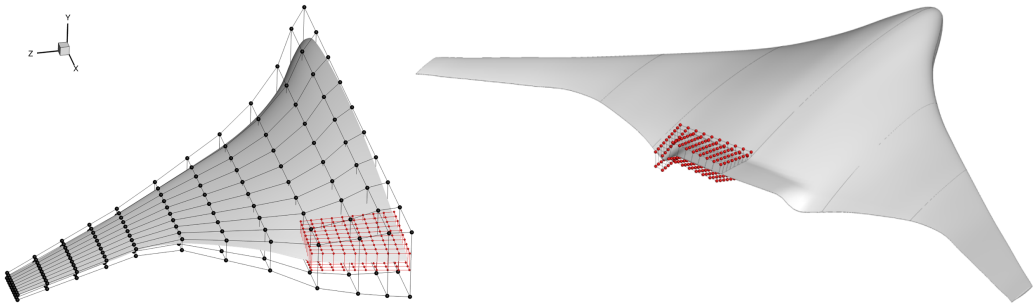

Figure 2: Example of parameterization through parent-child FFD blocks (Lyu & Martins, 2014).

Multi-component FFD

The basic FFD implementation lacks flexibility when the geometry has intersecting components. In such cases, pyGeo can parameterize each component using FFD and ensure a watertight surface representation at the component intersections using an inverse-distance surface deformation method (Yildirim et al., 2021). Figure 3 shows an example of a component-based FFD setup for a supersonic transport aircraft.

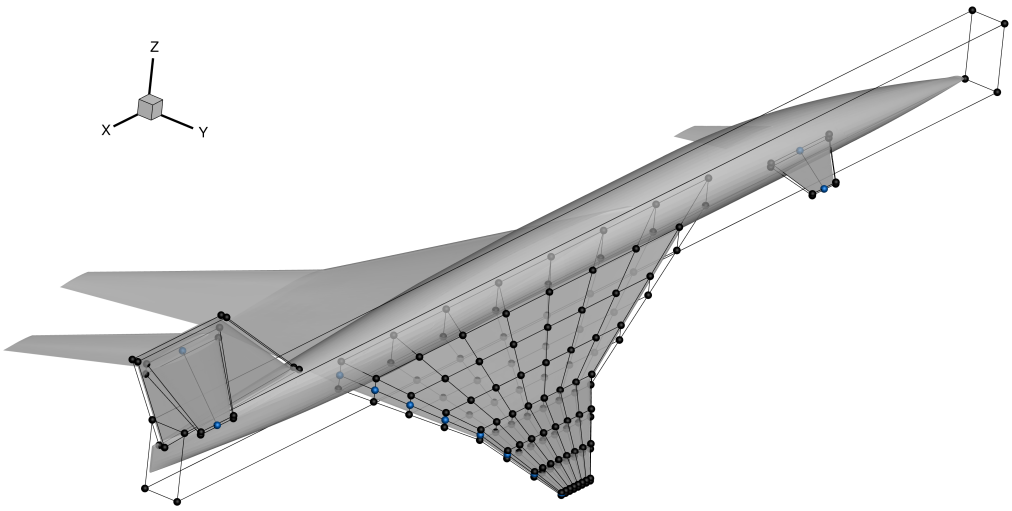

Figure 3: Example of FFD parameterization with intersecting components (Seraj & Martins, 2022).

Parametric Geometry Tools

pyGeo contains interfaces to two parametric geometry tools, the Engineering Sketch Pad (ESP) (Haimes & Dannenhoffer, 2013) and OpenVSP (McDonald & Gloudemans, 2022). ESP is a CAD software, while OpenVSP is a conceptual design tool. The two packages have different capabilities, but both directly define the geometry with design variables, and the created geometry can be used in external analysis tools.

pyGeo interfaces with ESP and OpenVSP in similar ways. In both cases, pyGeo takes an instance of the model and its points are associated with coordinates in a mesh from a solver in the MACH framework. For ESP and OpenVSP models, the pyGeo interface to the respective software stores the model in a form usable within the MACH framework and updates it as design variables are changed throughout the optimization. The pyGeo interface to ESP was used by (Brelje & Martins, 2021) to parameterize hydrogen tanks (Figure 4) that were packaged

within an aircraft wing as part of an aerostructural optimization. pyGeo's OpenVSP interface was used to parameterize the full aircraft configuration (Figure 5) studied in the aeropropulsive optimization work in (Yildirim et al., 2022).

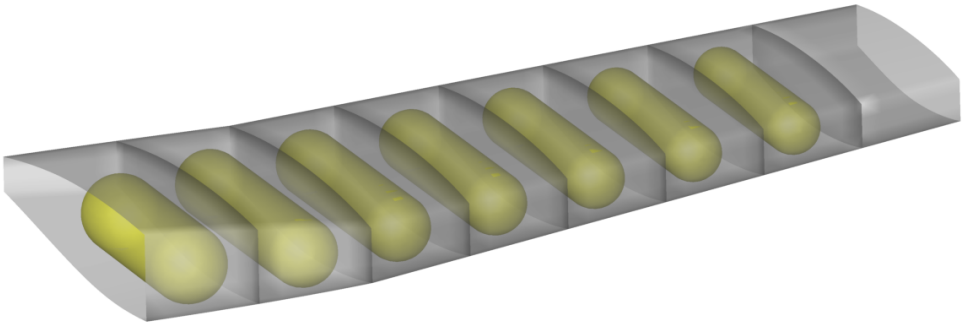

Figure 4: Example of ESP models of hydrogen tanks used through pyGeo (Brelje & Martins, 2021).

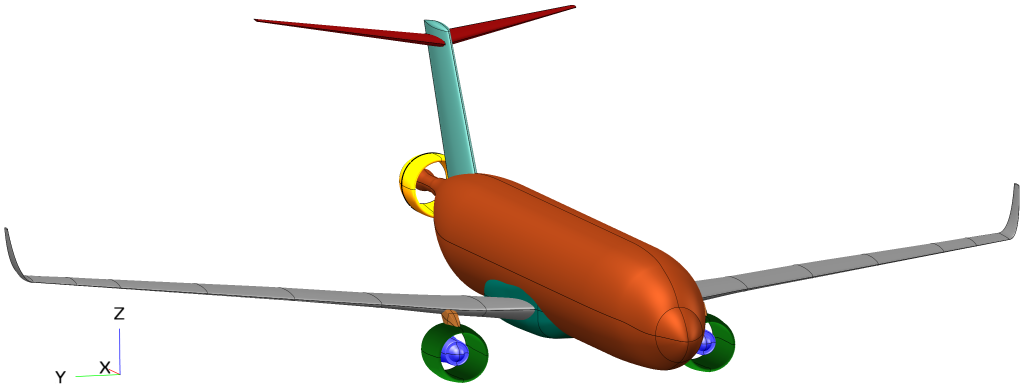

Figure 5: Example of a VSP aircraft model used through VSP's pyGeo interface (Yildirim et al., 2022).

Class Shape Transformation

The class shape transformation (CST) methodology (Kulfan, 2008) is a popular airfoil parameterization. It generates a shape using Bernstein polynomials to scale a class function, which is most often a base airfoil shape. pyGeo contains an implementation of this airfoil parameterization that supports design variables for the Bernstein polynomial weightings, the class function parameters, and the airfoil chord length. pyGeo's CST implementation can only be used for 2D problems, such as airfoil optimization (Figure 6).

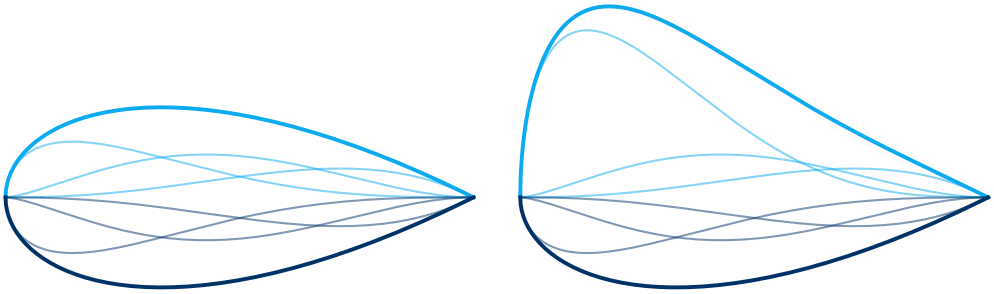

Figure 6: Airfoil defined by three CST coefficients on each surface undergoing a perturbation in one Bernstein polynomial.

Constraints

pyGeo also includes geometric constraints to prevent geometrically undesirable designs. The most commonly-used class of geometry constraints in pyGeo involves tracking one or more linear dimensions on the optimized object's surface. These constraints are created by specifying a single point, a line, or an array of points, along with a normal direction, then computing two line-surface intersection points. Some commonly used geometric constraints in shape optimization, such as thickness, area, and volume constraints (Figure 7) can be computed using variations on this approach, which is computationally cheap and robust (Brelje et al., 2020).

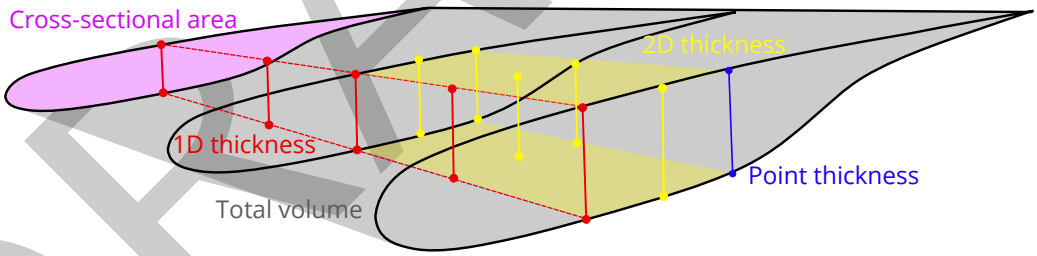

Figure 7: Thickness and volume constraints demonstrated on an wing section (Brelje et al., 2020).

If a more complex geometry needs to be integrated into an optimized surface, pyGeo supports an alternative geometric constraint formulation based on arbitrary triangulated surfaces as described in (Brelje et al., 2020) and illustrated in Figure 8.

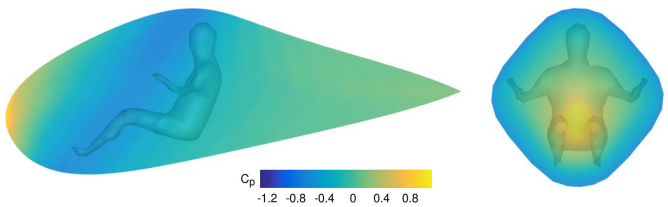

Figure 8: Triangulated surface constraint used to optimize an aeroshell around a complex geometry (Brelje et al., 2020).

Parallelism

pyGeo can optionally work under distributed memory parallelism using MPI, which is a requirement when interfacing with large-scale CFD applications. For example, the computational mesh may be partitioned and distributed among many processors by the CFD solver, and each processor may be aware of only its portion of the mesh. pyGeo can handle such scenarios by independently manipulating the geometry on each processor and aggregating the constraints across all processors when communicating with the optimizer.

Derivative Computation

In addition to geometry manipulation and constraints, pyGeo can compute derivatives of these operations with respect to design variables. For the geometric deformation, pyGeo can compute the Jacobian

$$\frac{\mathrm{d}X_s}{\mathrm{d}x},$$

where X_s is the vector of surface mesh coordinates, and x is the vector of geometric design variables.

Similarly, pyGeo can compute the constraint Jacobian

$$\frac{\mathrm{d}g}{\mathrm{d}x},$$

where g is the vector of geometric constraints.

For the FFD parameterization, these derivatives are computed using a combination of analytic methods (Martins & Ning, 2021) and the complex-step method (Martins et al., 2003). For the interfaces to OpenVSP and ESP, the derivatives are computed with parallel finite differences. The CST derivatives are computed analytically.

Statement of Need

Few open-source packages exist with comparable functionalities. To the authors' best knowledge, the only other open-source CFD-based optimization framework that contains geometry parameterization is SU2 (Economon et al., 2016). It supports Hicks–Henne bump functions (Hicks & Henne, 1978) for airfoil optimizations and the FFD method for three-dimensional cases. However, it cannot be used with other solvers because it is tightly integrated into the CFD solver.

While both OpenVSP and ESP can be used directly in optimization without using pyGeo, they lack capabilities needed for high-fidelity MDO when used as stand-alone tools. pyGeo fills in these gaps through parallelism, efficient gradients, and geometric constraints. It keeps OpenVSP and ESP in the optimization loop and provides a standard interface to these tools for their use with external solvers.

pyGeo has been used extensively in aerodynamic and aerostructural optimizations in aircraft, hydrofoil, and wind turbine applications. The different parameterizations within pyGeo have all been necessary for different optimization problems, depending on the geometry involved. The interface to ESP made it possible to parameterize hydrogen tanks within a combined aerostructural and packaging optimization (Brelje & Martins, 2021). pyGeo's OpenVSP interface was used in aeropropulsive optimizations (Yildirim et al., 2022).

The implementation of CST airfoil parameterization was used to compare methods for airfoil optimization (Adler et al., 2022). The method for using multiple FFD volumes has been used to optimize a conventional aircraft (Yildirim et al., 2021), a T-shaped hydrofoil (Liao et al., 2022), and a supersonic transport aircraft (Seraj & Martins, 2022).

pyGeo is maintained and developed by the MDO Lab[2] at the University of Michigan and is actively used for MDO applications in both research and industry. The geometry parameterization capabilities provided by pyGeo have facilitated the development of environmentally sustainable aircraft through design optimization.

Acknowledgements

We are grateful to the numerous pyGeo users who have contributed their time to the code and its maintenance over the years.

References

Adler, E. J., Gray, A. C., & Martins, J. R. R. A. (2022). To CFD or not to CFD? Comparing RANS and viscous panel methods for airfoil shape optimization. *33rd Congress of the International Council of the Aeronautical Sciences*.

Brelje, B. J., Anibal, J., Yildirim, A., Mader, C. A., & Martins, J. R. R. A. (2020). Flexible formulation of spatial integration constraints in aerodynamic shape optimization. *AIAA Journal*, *58*(6), 2571–2580. https://doi.org/10.2514/1.J058366

Brelje, B. J., & Martins, J. R. R. A. (2021). Aerostructural wing optimization for a hydrogen fuel cell aircraft. *Proceedings of the AIAA SciTech Forum*. https://doi.org/10.2514/6.2021-1132

Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., & Alonso, J. J. (2016). SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, *54*(3), 828–846. https://doi.org/10.2514/1.j053813

Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., & Naylor, B. A. (2019). OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, *59*(4), 1075–1104. https://doi.org/10.1007/s00158-019-02211-z

Haimes, R., & Dannenhoffer, J. (2013). The engineering sketch pad: A solid-modeling, feature-based, web-enabled system for building parametric geometry. *21st AIAA Computational Fluid Dynamics Conference*. https://doi.org/10.2514/6.2013-3073

Hajdik, H. M., Yildirim, A., & Martins, J. R. R. A. (2023). Aerodynamic shape optimization with CAD-based geometric parameterization. *AIAA SciTech Forum*. https://doi.org/10.2514/6.2023-0726

Hicks, R. M., & Henne, P. A. (1978). Wing design by numerical optimization. *Journal of Aircraft*, *15*, 407–412.

Kenway, G. K. W., Kennedy, G. J., & Martins, J. R. R. A. (2014). Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. *AIAA Journal*, *52*(5), 935–951. https://doi.org/10.2514/1.J052255

Kenway, G. K. W., & Martins, J. R. R. A. (2014). Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration. *Journal of Aircraft*, *51*(1), 144–160. https://doi.org/10.2514/1.C032150

Kulfan, B. M. (2008). Universal parametric geometry representation method. *Journal of Aircraft*, *45*(1), 142–158. https://doi.org/10.2514/1.29958

[2] https://mdolab.engin.umich.edu

Liao, Y., Martins, J. R. R. A., & Young, Y. L. (2021). 3-D high-fidelity hydrostructural optimization of cavitation-free composite lifting surfaces. *Composite Structures*, *268*, 113937. https://doi.org/10.1016/j.compstruct.2021.113937

Liao, Y., Yildirim, A., Martins, J. R. R. A., & Young, Y. L. (2022). RANS-based optimization of a T-shaped hydrofoil considering junction design. *Ocean Engineering*, *262*, 112051. https://doi.org/10.1016/j.oceaneng.2022.112051

Lyu, Z., & Martins, J. R. R. A. (2014). Aerodynamic design optimization studies of a blended-wing-body aircraft. *Journal of Aircraft*, *51*(5), 1604–1617. https://doi.org/10.2514/1.C032491

Martins, J. R. R. A., & Ning, A. (2021). *Engineering design optimization*. Cambridge University Press. https://doi.org/10.1017/9781108980647

Martins, J. R. R. A., Sturdza, P., & Alonso, J. J. (2003). The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, *29*(3), 245–262. https://doi.org/10.1145/838250.838251

McDonald, R. A., & Gloudemans, J. R. (2022). Open vehicle sketch pad: An open source parametric geometry and analysis tool for conceptual aircraft design. In *AIAA SCITECH 2022 forum*. American Institute of Aeronautics; Astronautics. https://doi.org/10.2514/6.2022-0004

Sederberg, T. W., & Parry, S. R. (1986). Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, *20*(4), 151–160. https://doi.org/10.1145/15886.15903

Seraj, S., & Martins, J. R. R. A. (2022). Aerodynamic shape optimization of a supersonic transport considering low-speed stability. *AIAA SciTech Forum*. https://doi.org/10.2514/6.2022-2177

Wu, N., Kenway, G., Mader, C. A., Jasa, J., & Martins, J. R. R. A. (2020). pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems. *Journal of Open Source Software*, *5*(54), 2564. https://doi.org/10.21105/joss.02564

Wu, N., Mader, C., & Martins, J. R. R. A. (2022). Sensitivity-based geometric parameterization for aerodynamic shape optimization. *AIAA AVIATION 2022 Forum*. https://doi.org/10.2514/6.2022-3931

Yildirim, A., Gray, J. S., Mader, C. A., & Martins, J. R. R. A. (2022). Boundary layer ingestion benefit for the STARC-ABL concept. *Journal of Aircraft*, *59*(4), 896–911. https://doi.org/10.2514/1.C036103

Yildirim, A., Mader, C. A., & Martins, J. R. R. A. (2021). A surface mesh deformation method near component intersections for high-fidelity design optimization. *Engineering with Computers*. https://doi.org/10.1007/s00366-020-01247-w

Zhang, T., Wang, Z., Huang, W., & Yan, L. (2018). A review of parametric approaches specific to aerodynamic design process. *Acta Astronautica*, *145*, 319–331. https://doi.org/10.1016/j.actaastro.2018.02.011