The Journal of Open Source Software

Condor, a mathematical modeling framework for engineers with deadlines

Benjamin W. L. Margolis [1] **and Kenneth R. Lyons** [1]

1 NASA Ames Research Center, Systems Analysis Office

DOI: 10.21105/joss.08859

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @HectorMozo3110
- @amcandio

Submitted: 21 August 2025
Published: 15 September 2025

Summary

Numerical modeling is an important part of the engineering workflow, providing understanding of engineering systems without the often prohibitive cost of fabricating physical models and prototypes. Condor is a mathematical modeling framework in Python that enables the rapid deployment of models for analysis and design. Condor uses metaprogramming to provide a mathematical domain-specific language (DSL) that reduces the software development effort needed to deploy models for solving engineering analysis and design problems. It also features a modular model-solver-backend architecture, analogous to the model-view-controller (MVC) pattern in web-application development, facilitating usage of off-the-shelf solvers.

Statement of Need

Condor was developed at NASA Ames Research Center's Systems Analysis Office to solve a variety of analysis and design problems in aeronautics (Listgarten et al., 2025; B. Margolis et al., 2024; B. W. L. Margolis & others, 2026a; Park et al., 2025; Pham et al., 2025; Zlinski et al., 2025), orbital trajectory design (Koehler et al., 2024; B. W. L. Margolis & Woffinden, 2024; B. W. Margolis & Woffinden, 2024), and subsystem design (B. W. L. Margolis & others, 2026c, 2026d). Condor's modular framework makes it feasible to develop algorithms using existing models as test examples like gradient methods for solutions to ordinary differential equation with events (B. W. L. Margolis, 2023) or uncertain differential equations (B. W. L. Margolis & others, 2026b).

A variety of existing libraries work towards unifying solvers and optimization tools under a single interface. After assessing the available tools, we found that no existing tool provided an interface that we felt improved the engineering workflow. Condor is unique, to the best of the authors' knowledge, for providing a mathematically-focused DSL to facilitate rapid prototyping of engineering models. Condor's modular architecture supports rapid deployment of new or existing solvers for engineering problems. These features improve the engineering workflow by providing a single engineering-focused interface to any existing solver, facilitating the creation of new models as conceptual analysis demands arise.

We built Condor to provide an interface for engineers that used computational tools from the AI/ML community as the "backend" of the framework in the Python programming language. Using Python as the base language offers access to a huge community of practice, especially in scientific computing and numerical methods, where the majority of the AI/ML tools originated. This means that even with a small development team, we could provide features like parallel computing, file interfaces, and more, by leveraging the open-source ecosystem.

Description

Condor is a general-purpose engineering-mathematical modeling framework used to build conceptual design and analysis capability. It is related to multi-disciplinary analysis (MDA) frameworks (e.g., Simulink (The MathWorks, Inc., n.d.), SimuPy (B. W. Margolis, 2017), ModelCenter (Phoenix Integration, n.d.), NPSS (Curlett & Felder, 1995), Modelica (Fritzson & Bunus, 2002)), computational tools developed by the Artificial Intelligence/Machine Learning (AI/ML) community (e.g., CasADi (Andersson et al., 2018), JAX (Bradbury et al., 2018), Aesara (Willard et al., 2023), PyTorch (Paszke et al., 2017), TensorFlow (Abadi et al., 2015)), and specific numerical solvers (e.g., IPOPT (Wächter & Biegler, 2005) / SLSQP (Kraft, 1994) for optimization, SUNDIALS (Hindmarsh et al., 2005) for trajectory integration, etc.) The goal was to provide a single, easy-to-use interface to the best in-class numerical solvers so engineers can focus on engineering rather than learning solver-specific interfaces or spend their time with intensive coding or algorithm tuning for new models. The computational tools from the AI/ML communities took some steps to address these challenges, but their interfaces did not lend themselves to general engineering approaches to modeling physical systems. Similarly, the existing MDA frameworks did not satisfy the system analysis needs, either because they were too domain focused (e.g., dynamical systems), or because they used their own language so had limited community of practice and could not benefit from recent advancements in best practices and computational techniques.

Condor was designed and built following modern software development best-practices. We leveraged CasADi as our backend, but thanks to Condor's modular architecture, changing backends (e.g., to JAX or Aesara) would require providing a few interface files for the backend-shim infrastructure. We also leverage existing best-in-class solvers for optimization (IPOPT, other CasADi-enabled optimizers, sequential least squares SLSQP), differential equations (Runge-Kutta implementations in SciPy (Hairer et al., 1993; Virtanen et al., 2020), Lawrence Livermore National Lab's SUNDIALS), and efficient array-arithmetic and linear algebra from the backend. Since Condor makes it easy to wrap external solvers into the modeling framework, we have also used NASA-developed solvers like CBAero (D. Kinney, 2007) (for hypersonic aerodynamics and aerothermal modeling), VSPAero (D. J. Kinney, 2025) (for subsonic and supersonic aerodynamics modeling), and NPSS (for propulsion).

Since Condor uses standard Python data structures, we also leverage general purpose scientific tools and libraries. Virtually all utility functions come from the larger Python ecosystem:

- NumPy's (Harris et al., 2020) built-in file format to read and store model results
- Compatible with Python parallel processing libraries such as the built-in multiprocessing or the third-party joblib (Joblib Development Team, 2020) and Dask (Dask Development Team, 2016) projects
- For most low-level operations (creating identity matrix, concatenating vectors, etc.), we follow the Python array API standard to reduce user and development burden
- Compatible with any plotting capability like Matplotlib (Hunter, 2007), seaborn (Waskom, 2021), and PyQtGraph (Moore et al., 2023)

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2018). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. https://doi.org/10.1007/s12532-018-0139-4

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., & Wanderman-Milne, S. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.1.46). http://github.com/google/jax

Curlett, B. P., & Felder, J. L. (1995). *Object-oriented approach for gas turbine engine simulation* (Technical Report No. 106970). NASA.

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. http://dask.pydata.org

Fritzson, P., & Bunus, P. (2002). Modelica - a general object-oriented language for continuous and discrete-event system modeling and simulation. *Proceedings 35th Annual Simulation Symposium. SS 2002*, 365–380. https://doi.org/10.1109/SIMSYM.2002.1000174

Hairer, E., Wanner, G., & Nørsett, S. P. (1993). *Solving ordinary differential equations i: Nonstiff problems*. Springer.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., & Woodward, C. S. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, *31*(3), 363–396. https://doi.org/10.1145/1089014.1089020

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Joblib Development Team. (2020). *Joblib: Running python functions as pipeline jobs*. https://joblib.readthedocs.io/

Kinney, D. (2007, January). Aerothermal anchoring of CBAERO using high fidelity CFD. *45th AIAA Aerospace Sciences Meeting and Exhibit*. https://doi.org/10.2514/6.2007-608

Kinney, D. J. (2025). *Using VSPAero*. https://openvsp.org/wiki/doku.php?id=vspaerotutorial

Koehler, H. M., Hawkins, M. J., Neuhaus, J. R., Couch, J., Bossinger, R. L., Brazukas, K., Fiebelkorn, T., Crues, E. Z., Margolis, B. W., Hasseler, T. D., & others. (2024). *Expansion of check-cases for 6DOF simulation*. National Aeronautics; Space Administration.

Kraft, D. (1994). Algorithm 733: TOMP–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, *20*(3), 262–281. https://doi.org/10.1145/192115.192124

Listgarten, N. S., Pham, D., Natividad, C. A. D., Margolis, B. W., Lyons, K. R., Garcia, J. A., Bowles, J. V., & Russell, C. E. (2025, January). Parallel hybrid turboprop performance modeling and optimization. *AIAA SCITECH 2025 Forum*. https://doi.org/10.2514/6.2025-1435

Margolis, B. W. (2017). SimuPy: A Python framework for modeling and simulating dynamical systems. *J. Open Source Software*, *2*(17), 396. https://doi.org/10.21105/joss.00396

Margolis, B. W. L. (2023). A sweeping gradient method for ordinary differential equations with events. *Journal of Optimization Theory and Applications*, *199*(2), 600–638. https://doi.org/10.1007/s10957-023-02303-3

Margolis, B. W. L., & others. (2026a). A component-assembly architecture for conceptual aircraft design. *AIAA SciTech Forum*.

Margolis, B. W. L., & others. (2026b). A systems approach to modeling uncertain ODE

systems. *AIAA SciTech Forum*.

Margolis, B. W. L., & others. (2026c). Outer mold LineDesign ofa blended-wing-body using gradient-based optimization of a vortex-lattice model. *AIAA SciTech Forum*.

Margolis, B. W. L., & others. (2026d). Useful derivatives for numerical propulsion system simulation models. *AIAA SciTech Forum*.

Margolis, B. W. L., & Woffinden, D. (2024). Co-optimization of navigation system requirements and trajectory design using a sweeping gradient method and linear covariance analysis. *AIAA and AAS Astrodynamics Specialist Conference*.

Margolis, B. W., & Woffinden, D. (2024). Robust trajectory optimization techniques using a sweeping gradient method and linear covariance analysis. *AAS/AIAA Astrodynamics Specialist Conference*.

Margolis, B., Lyons, K., Garcia, J. A., Pham, D., Bowles, J. V., Bridges, N., & Chang, E. (2024, July). General aviation synthesis program advancements with symbolic computations, optimization, and decoupled numerical methods. *AIAA AVIATION FORUM AND ASCEND 2024*. https://doi.org/10.2514/6.2024-3628

Moore, O., Jessurun, N., Chase, M., Nemitz, N., & Campagnola, L. (2023). PyQtGraph - high performance visualization for all platforms. *Proceedings of the 22nd Python in Science Conference*, 106–113. https://doi.org/10.25080/gerudo-f2bc6f59-00e

Park, J., Listgarten, N., Buescher, H., & Pham, D. D. (2025). Framework for technical performance uncertainty in hybrid-electric aircraft development. *IEEE Transportation Electrification Conference & Expo+ Electric Aircraft Technologies Symposium*. https://doi.org/10.1109/ITEC63604.2025.11097944

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). *Automatic differentiation in PyTorch*.

Pham, D. D., Naik, A. M., Recine, C., Joseph, J., Listgarten, N. S., Natividad, C. A., Park, J., & Wishart, J. (2025). Performance scaling of multi-class parallel hybrid-electric regional turboprop airliner concepts. *2025 IEEE Transportation Electrification Conference & Expo+ Electric Aircraft Technologies Symposium*. https://doi.org/10.1109/ITEC63604.2025.11097987

Phoenix Integration. (n.d.). *ModelCenter*. https://www.ansys.com/products/connect/ansys-modelcenter?intcid=phoenixintreferral

The MathWorks, Inc. (n.d.). *Simulink - simulation and model-based design*. https://www.mathworks.com/products/simulink.html

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wächter, A., & Biegler, L. T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1). https://doi.org/10.1007/s10107-004-0559-y

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Willard, B. T., Louf, R., Chaudhari, K., Mares, B., Lunagariya, S., Foreman-Mackey, D., & Gerlanc, D. (2023). *Aesara*. Aesara Developers. https://github.com/aesara-devs/aesara

Zlinski, S., Recine, C. J., Natividad, C. A., Pham, D. D., Margolis, B. W., Listgarden, N. S., & Phillips, J. (2025, July). Assessing national airspace system impact of the

transonic truss-braced wing aircraft. *AIAA AVIATION FORUM AND EXPOSITION*. https://doi.org/10.2514/6.2025-3101