solve_nivp: A Python toolkit for integrating nonsmooth dynamical systems

David Riley[1] **and Ioannis Stefanou**[1]

1 IMSIA (UMR 9219), CNRS, EDF, CEA, ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France

Summary

solve_nivp is a Python library for time integration of *nonsmooth* ODE/DAE models: systems with abrupt changes such as impacts, switching, or inequality constraints. Such models are widespread: frictional contact in mechanics, piecewise and switching behaviour in circuits, sliding–mode control, and discontinuous rules in finance and energy markets (Acary & Brogliato, 2008; Bernardo et al., 2008; Brogliato, 2016; Gabriel et al., 2012; Shtessel et al., 2014; Tramontana et al., 2010). Classical solvers, which assume smoothness, often require heavy regularisation or very small steps due to the inherent stiffness of these models. **solve_nivp** builds nonsmooth rules directly into the implicit time-stepping scheme, enabling users to encode constraints and advance the state robustly. Documentation, examples, and tests accompany the code for reproducible use.

Statement of need

Many models in mechanics, circuits, biology, control, and quantitative markets exhibit discontinuities or set-valued relations (ideal diodes, genetic switching, finance applications, stick–slip friction, unilateral contact, switching, saturation). While nonsmooth time-stepping schemes are well established (Acary & Brogliato, 2008; Stewart & Trinkle, 1996), practitioners working in Python often use interfaces oriented to smooth right-hand sides and event detection. In practice, this pushes users toward ad hoc regularisation or very small time steps to cope with kinks and set-valued relations, thereby increasing computational cost and complicating reproducibility.

Existing libraries such as *Siconos* provide C++ with a Python interface for complementarity-based and rigid-body contact dynamics (Acary et al., 2019). At the other end of the spectrum, general-purpose DAE solvers such as *SUNDIALS/IDA* deliver robust variable-order BDF methods for smooth problems (Hindmarsh et al., 2005). Between these options, there is a gap for a lightweight, Python-native library that lets users encode nonsmooth rules directly and couple them with implicit integrators and globalisation strategies—without adopting a full multibody framework or rewriting models in another language.

solve_nivp addresses this need with a minimal projector interface for expressing set-valued relations, together with nonlinear solvers and implicit time-stepping tailored to piecewise-smooth behaviour. The design targets research and prototyping workflows common in computational mechanics and mathematics, robotics, and control: small-to-moderate problem sizes, clear separation of model and solver components, and emphasis on reproducible examples and tests. By bringing projection-based nonsmooth techniques into a familiar scientific Python workflow, the package lowers the barrier to building, comparing, and sharing nonsmooth models alongside existing tools.

Functionality and design

solve_nivp is organised around three interchangeable components. Users encode the nonsmooth rule as a constraint (i.e., a projection onto a convex set), choose a nonlinear solver, and select an implicit integrator. The library wires these components and integrates the ODE/DAE in time while enforcing the constraint at each step, helping keep models readable and making it straightforward to swap algorithms during experimentation.

Projectors are small classes exposing `project()` and an optional generalised derivative via `tangent_cone()`. The distribution includes several built-in projection methods; users can add new ones by implementing the same two methods. For the nonlinear step, the package provides a semismooth Newton method (Qi & Sun, 1993) with Armijo line search (Armijo, 1966) and a variational-inequality fixed-point iteration (Fortin & Glowinski, 1983; Uzawa, 1958), both aimed at nonsmooth problems and offering standard tolerances, safeguards, and iteration diagnostics.

Time integration of the dynamical system is implicit and includes the Backward Euler (Hairer et al., 1993), Trapezoidal (Hairer et al., 1993), and θ methods (Hairer et al., 1993), as well as a composite scheme based on the Bathe method (Bathe & Noh, 2012). An adaptive controller is available and can exclude algebraic components from the local-error norm, which is useful for mixed ODE/DAE models. An optional reinforcement-learning add-on exposes the time integrator as an environment for learning adaptive step-size policies. In a companion study (Riley et al., 2025), RL-based policies achieved substantial speedups while maintaining accuracy, demonstrating the approach's potential for hard, nonsmooth problems. Linear-algebra routines operate on dense or sparse arrays in the SciPy ecosystem (Virtanen et al., 2020).

The public API exposes both a convenience entry point (`solve_nivp`) and low-level building blocks (projectors, solvers, integrators) to support rapid prototyping and comparative studies. The repository includes examples and notebooks demonstrating contact-rich mechanics and switching problems, together with unit tests and Sphinx documentation, so users can reproduce results with minimal setup.

Acknowledgements

References

Acary, V., Bonnefon, O., Brémond, M., Huber, O., Pérignon, F., & Sinclair, S. (2019). *An introduction to Siconos* (Rapport Technique RT-0340; Version v3, p. 97). Inria. https://hal.inria.fr/inria-00162911

Acary, V., & Brogliato, B. (2008). *Numerical methods for nonsmooth dynamical systems: Applications in mechanics and electronics* (Vol. 35). Springer. https://doi.org/10.1007/978-3-540-75392-6

Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, *16*(1), 1–3. https://doi.org/10.2140/pjm.1966.16.1

Bathe, K.-J., & Noh, G. (2012). Insight into an implicit time integration scheme for structural dynamics. *Computers & Structures*, *98*, 1–6. https://doi.org/10.1016/j.compstruc.2012.01.009

Bernardo, M. di, Budd, C. J., Champneys, A. R., & Kowalczyk, P. (2008). *Piecewise-smooth dynamical systems: Theory and applications* (Vol. 163). Springer. https://doi.org/10.

1007/978-1-84628-708-4

Brogliato, B. (2016). *Nonsmooth mechanics: Models, dynamics and control* (3rd ed.). Springer. https://doi.org/10.1007/978-3-319-28664-8

Fortin, M., & Glowinski, R. (1983). *Augmented Lagrangian methods: Applications to the numerical solution of boundary-value problems*. North-Holland.

Gabriel, S. A., Conejo, A. J., Fuller, J. D., Hobbs, B. F., & Ruiz, C. (2012). *Complementarity modeling in energy markets* (Vol. 180). Springer Science & Business Media.

Hairer, E., Nørsett, S. P., & Wanner, G. (1993). *Solving ordinary differential equations I: Nonstiff problems* (Vol. 8). Springer. https://doi.org/10.1007/978-3-540-78862-1

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.

Qi, L., & Sun, J. (1993). A nonsmooth version of Newton's method. *Mathematical Programming*, *58*(1), 353–367.

Riley, D., Stathas, A., Gutiérrez-Oribio, D., & Stefanou, I. (2025). *Reinforcement learning-based adaptive time-integration for nonsmooth dynamics*. https://doi.org/10.48550/arXiv.2501.08934

Shtessel, Y., Edwards, C., Fridman, L., & Levant, A. (2014). *Sliding mode control and observation*. Birkhäuser. https://doi.org/10.1007/978-0-8176-4893-0

Stewart, D. E., & Trinkle, J. C. (1996). An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction. *International Journal for Numerical Methods in Engineering*, *39*(15), 2673–2691. https://doi.org/10.1002/(SICI)1097-0207(19960815)39:15%3C2673::AID-NME972%3E3.0.CO;2-I

Tramontana, F., Westerhoff, F., & Gardini, L. (2010). On the complicated price dynamics of a simple one-dimensional discontinuous financial market model with heterogeneous interacting traders. *Journal of Economic Behavior & Organization*, *74*(3), 187–205. https://doi.org/10.1016/j.jebo.2010.02.008

Uzawa, H. (1958). Iterative methods for concave programming. In K. J. Arrow, L. Hurwicz, & H. Uzawa (Eds.), *Studies in linear and nonlinear programming* (pp. 154–165). Stanford University Press. https://doi.org/10.1017/cbo9780511664496.011

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272.