




# scikit-finite-diff, a new tool for PDE solving

Nicolas Cellier<sup>1</sup> and Christian Ruyer-Quil<sup>1</sup>

<sup>1</sup> Université Savoie Mont-Blanc

DOI: [10.21105/joss.01356](https://doi.org/10.21105/joss.01356)

## Software

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

**Submitted:** 18 March 2019

**Published:** 03 June 2019

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

## Summary

Scikit-FDiff is a new tool for Partial Differential Equation (PDE) solving, written in pure Python, that focuses on reducing the time between the development of the mathematical model and the numerical solving.

It allows easy and automated finite difference discretization, thanks to symbolic processing (using the SymPy library (Meurer et al., 2017)) which can deal with systems of multi-dimensional partial differential equations and complex boundary conditions.

Using finite differences, and the method of lines, Scikit-FDiff allows for the transformation of the original PDE into an Ordinary Differential Equation (ODE), providing fast computation of the temporal evolution vector and the Jacobian matrix. The latter is pre-computed in a symbolic way and is sparse by nature. An efficient vectorization allows one to formulate the numerical system in such a way as to facilitate the numerical solver work, even for complex multi-dimensional coupled cases. Systems can be evaluated with as few computational resources as possible, enabling the use of implicit and explicit solvers at a reasonable cost.

Scikit-FDiff stands out in comparison to other competitors, such as the FEniCS Project, Clawpack, or the Dedalus Project, in terms of its simplicity. Despite the fact that Scikit-FDiff uses a relatively simple method (finite-differences), which allows one to code in a way which is close to the mathematical model, it is able to solve real-world problems. It is however less suited than other packages for building specialized solvers.

Scikit-FDiff also contains several classic ODE solver implementations (some of which have been made available from dedicated python libraries), including the backward and forward Euler scheme, Crank-Nicolson, and explicit Runge-Kutta. More complex approaches, like the improved Rosenbrock-Wanner schemes (Rang, 2015) up to the 6th order, are also available in Scikit-FDiff. The time-step is managed by a built-in error computation, which ensures the accuracy of the solution.

The main goal of this software is to minimize the time spent writing numerical solvers allowing one to focus on model development and data analysis. Therefore, Scikit-FDiff has been formulated such that it can solve both simple examples and complex models in only a few line of code. In addition, extra tools are provided, such as data saving during the simulation, real-time plotting, and post-processing.

Scikit-FDiff can be used for a wide range of applications including thermal diffusion, heterogeneous chemical reactor modelling, wave propagation, and a large number of non-linear phenomena modelled as systems of PDEs. To date the authors have used it for solving heated falling-films (Cellier, 2018), droplet spread on windshield and simple moisture flow in a porous medium. Furthermore, Scikit-Fdiff has been validated for solving the shallow-water equation on dam-breaks (LeVeque, 2002) (and the steady-lake case).

## Acknowledgements

This software development was partially supported by the Agence Nationale de la Recherche (project FRAISE). We thank Matteo Ravasi and Jack Poulson for their insights during the review process and Kevin Mattheus Moerman for his work as editor.

## References

- Cellier, N. (2018). *Optimisation d'échangeurs à films ruisselants* (PhD Thesis). Université de Savoie Mont-Blanc.
- LeVeque, R. J. (2002). *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press.
- Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., et al. (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*, 3, e103. doi:[10.7717/peerj-cs.103](https://doi.org/10.7717/peerj-cs.103)
- Rang, J. (2015). Improved traditional Rosenbrock-Wanner methods for stiff ODEs and DAEs. *J. Comput. Appl. Math.*, 286, 128–144. doi:[10.1016/j.cam.2015.03.010](https://doi.org/10.1016/j.cam.2015.03.010)