The Journal of Open Source Software

PySEMTools: A library for post-processing hexahedral spectral element data.

Adalberto Perez [1¶], **Siavash Toosi** [2], **Tim Felle Olsen** [4], **Stefano Markidis** [3], **and Philipp Schlatter** [1,2]

1 FLOW Dept. Engineering Mechanics, KTH Royal Institute of Technology **2** Institute of Fluid Mechanics (LSTM), Friedrich–Alexander–Universität (FAU) **3** Division of Computational Science and Technology (CST), KTH Royal Institute of Technology **4** Department of Civil and Mechanical Engineering Solid Mechanics, Technical University of Denmark ¶ Corresponding author

DOI: 10.21105/joss.08767

Software

- Review
- Repository
- Archive

Editor: Henrik Finsberg

Reviewers:

- @nchristensen
- @finsberg

Submitted: 21 March 2025
Published: 01 April 2026

Summary

PySEMTools is a Python-based library for post-processing simulation data produced with high-order hexahedral elements in the context of the spectral element method in computational fluid dynamics. It aims to minimize intermediate steps typically needed when analyzing large files. Specifically, the need to use separate codebases (like the solvers themselves) at post-processing. For this effect, we leverage the use of message passing interface (MPI) for distributed computing to perform typical data processing tasks such as spectrally accurate differentiation, integration, interpolation, and reduced order modeling, among others, on a spectral element mesh. All the functionalities are provided in self-contained Python code and do not depend on the use of a particular solver. We believe that PySEMTools provides tools to researchers to accelerate scientific discovery and reduce the entry requirements for the use of advanced methods in computational fluid dynamics.

Statement of need

The motion of fluids around objects is fundamental for many industrial and natural systems, from aerodynamics and cooling to the behavior of weather systems. Particularly relevant applications generally exist in the turbulent flow regime, where the fluid is subject to unsteady motions that are characterized by the interactions of eddies of multiple sizes and where increased levels of fluctuations and mixing exist.

A popular method to study these phenomena is to use computers to simulate their governing physics. The multi-scale characteristic of turbulence and the high Reynolds numbers (ratio between inertial and viscous forces) that are typically of interest require that the numerical grids are fine enough to capture the motion of the smallest eddies. While this has implied that the computational cost of simulations is high, the advent of graphics processing units (GPUs) has opened the doors to perform simulations that would not have been possible in the past. This increase in capability has made managing the data produced on a typical simulation campaign more challenging. Our work in PySEMTools aims to streamline the data management and post-processing of the results obtained from a particular numerical method often used to study turbulent flows while keeping high-order accuracy.

PySEMTools aims to help post-processing data from solvers that use the spectral element method (SEM) originally proposed by Patera (1984), which is a high-order variant of the finite element method (FEM). In SEM, the computational domain is divided into a finite set of elements in which a Gauss-Lobatto-Legendre (GLL) grid of a given degree N is embedded. Inside each element, the solution is expanded using polynomials of order $P = N - 1$, resulting

in low dissipation and dispersive errors.

Nek5000 (Fischer et al., 2008), written in Fortran 77, is a successful implementation of SEM that has been used for several studies on the field, such as simulations of vascular flows by Fischer et al. (2006), turbulent pipe flow by El Khoury et al. (2013), flow around wings by Mallor et al. (2024) and even nuclear applications as shown in the overview by Merzari et al. (2020). In general, the post-processing pipeline has been somewhat complicated, as when the data is needed in the SEM format, for example, to calculate derivatives of velocity fields, the solver itself has been used in a "post-processing" mode. This mode uses the solver and additional Fortran code that needs to be compiled to produce smaller files that can be used in Matlab or Python with e.g. PyMech by Mohanan et al. (2024), to perform signal processing, create plots, etc. NekRS by Fischer et al. (2022), a GPU version of Nek5000, and Neko (Jansson et al., 2023, 2024), a modern Fortran implementation of SEM have followed the same approach, motivating the necessity of our PySEMTools for the future.

The motivation behind using the solvers themselves with the data in its raw format is understandable, as these large files need to be processed in parallel due to their sheer size. Still, we believe that the process has become very cumbersome as multiple code bases need to be maintained for post-processing the data. With `PySEMTools` we have brought a solution to this, as we have included all the functionalities that are typically needed from the solvers while ensuring that the codes perform efficiently in parallel while also taking advantage of the rich library ecosystem present in Python.

Features

`PySEMTools` relies heavily on MPI for Python by Dalcín et al. (2005), given that it has been designed from the beginning to work on distributed settings. For computations, we rely on NumPy (Harris et al., 2020). It has been extensively tested on data produced by Nek5000 and Neko but, as mentioned before, the implemented methods and routines are consistent with any SEM-like data structure with hexahedral elements. Among its most relevant features are the following:

- **Parallel IO**: A set of routines to perform distributed IO on Nek5000/Neko field files and directly keep the data in memory on NumPy arrays or PyMech data objects.
- **Parallel data interfaces**: A set of objects that aim to facilitate the transfer of messages among processors. This is designed to ease the use of MPI functions for more inexperienced users.
- **Calculus**: Objects to calculate the derivation and integration matrices based on the geometry, which allows users to perform calculus operations on the spectral element mesh.
- **Mesh connectivity and partitioning**: Objects to determine the connectivity based on the geometry and mesh repartitioning tools for tasks such as global summation, among others.
- **Interpolation**: Routines to perform high-order interpolation from an SEM mesh into any arbitrary query point. A crucial functionality when performing post-processing.
- **Reduced-order modeling**: Objects to perform parallel and streaming proper orthogonal decomposition (POD).
- **Data compression/streaming**: Through the use of ADIOS2 (Godoy et al., 2020), a set of interfaces is available to perform data compression or to connect Python scripts to running simulations to perform in-situ data processing.
- **Visualization**: Given that the data is available in Python, visualizations can be performed from readily available packages.

We note that all of these functionalities are supported by examples in the software repository.

Acknowledgements

This work is partially funded by the "Adaptive multi-tier intelligent data manager for Exascale (ADMIRE)" project, which is funded by the European Union's Horizon 2020 JTI-EuroHPC research and innovation program under grant Agreement number: 956748. Computations for testing were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

References

Dalcín, L., Paz, R., & Storti, M. (2005). MPI for python. *Journal of Parallel and Distributed Computing*, *65*(9), 1108–1115. https://doi.org/10.1016/j.jpdc.2005.03.010

El Khoury, G. K., Schlatter, P., Noorani, A., Fischer, P. F., Brethouwer, G., & Johansson, A. V. (2013). Direct numerical simulation of turbulent pipe flow at moderately high reynolds numbers. *Flow, Turbulence and Combustion*, *91*(3), 475–495. https://doi.org/10.1007/s10494-013-9482-8

Fischer, P., Kerkemeier, S., Min, M., Lan, Y.-H., Phillips, M., Rathnayake, T., Merzari, E., Tomboulides, A., Karakus, A., Chalmers, N., & Warburton, T. (2022). NekRS, a GPU-accelerated spectral element navier–stokes solver. *Parallel Computing*, *114*, 102982. https://doi.org/10.1016/j.parco.2022.102982

Fischer, P., Loth, F., Lee, S.-W., Smith, D., Tufo, H., & Bassiouny, H. (2006). Parallel simulation of high reynolds number vascular flows. In A. Deane, A. Ecer, J. McDonough, N. Satofuka, G. Brenner, D. R. Emerson, J. Periaux, & D. Tromeur-Dervout (Eds.), *Parallel computational fluid dynamics 2005* (pp. 219–226). Elsevier. https://doi.org/10.1016/B978-044452206-1/50026-4

Fischer, P., Lottes, J. W., & Kerkemeier, S. G. (2008). *Nek5000 web page*. http://nek5000.mcs.anl.gov

Godoy, W. F., Podhorszki, N., Wang, R., Atkins, C., Eisenhauer, G., Gu, J., Davis, P., Choi, J., Germaschewski, K., Huck, K., & others. (2020). Adios 2: The adaptable input output system. A framework for high-performance data management. *SoftwareX*, *12*, 100561. https://doi.org/10.1016/j.softx.2020.100561

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

Jansson, N., Karp, M., Perez, A., Mukha, T., Ju, Y., Liu, J., Páll, S., Laure, E., Weinkauf, T., Schumacher, J., Schlatter, P., & Markidis, S. (2023). Exploring the ultimate regime of turbulent rayleigh–bénard convection through unprecedented spectral-element simulations. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. https://doi.org/10.1145/3581784.3627039

Jansson, N., Karp, M., Podobas, A., Markidis, S., & Schlatter, P. (2024). Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics. *Computers & Fluids*, *275*, 106243. https://doi.org/10.1016/j.compfluid.2024.106243

Mallor, F., Vinuesa, R., Örlü, R., & Schlatter, P. (2024). High-fidelity simulations of the flow around a NACA 4412 wing section at high angles of attack. *International Journal of Heat and Fluid Flow*, *110*, 109590. https://doi.org/10.1016/j.ijheatfluidflow.2024.109590

Merzari, E., Fischer, P., Min, M., Kerkemeier, S., Obabko, A., Shaver, D., Yuan, H., Yu, Y.,

Martinez, J., Brockmeyer, L., Fick, L., Busco, G., Yildiz, A., & Hassan, Y. (2020). Toward exascale: Overview of large eddy simulations and direct numerical simulations of nuclear reactor flows with the spectral element method in Nek5000. *Nuclear Technology*, *206*(9), 1308–1324. https://doi.org/10.1080/00295450.2020.1748557

Mohanan, A. V., Chauvat, G., Kleine, V. G., Fabbiane, N., & Canton, J. (2024). *Pymech: A python software suite for Nek5000 and SIMSON* (Version 2.0.1). Zenodo. https://doi.org/10.5281/zenodo.14194101

Patera, A. T. (1984). A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *Journal of Computational Physics*, *54*(3), 468–488. https://doi.org/10.1016/0021-9991(84)90128-1