

¹ PySEMTTools: A library for post-processing hexahedral spectral element data.

³ **Adalberto Perez**  ¹, **Siavash Toosi**  ², **Tim Felle Olsen**  ⁴, **Stefano Markidis**  ³, and **Philipp Schlatter**  ^{1,2}

⁵ 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.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

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

Editor: [Henrik Finsberg](#)  

Reviewers:

- [@kris-rowe](#)
- [@nchristensen](#)

Submitted: 21 March 2025

Published: unpublished

License

Authors of papers retain copyright and release the work under a ²¹ Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))²²

⁹ Summary

¹⁰ PySEMTTools 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 PySEMTTools provides tools to researchers to accelerate scientific discovery and reduce the entry requirements for the use of advanced methods in computational fluid dynamics.

²⁰ Statement of need

²¹ 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.

²² 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 PySEMTTools 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.

²³ PySEMTTools 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

42 in low dissipation and dispersive errors.

43 Nek5000 ([Fischer et al., 2008](#)), written in Fortran 77, is a successful implementation of SEM
44 that has been used for several studies on the field, such as simulations of vascular flows by
45 Fischer et al. ([2006](#)), turbulent pipe flow by El Khoury et al. ([2013](#)), flow around wings by
46 Mallor et al. ([2024](#)) and even nuclear applications as shown in the overview by Merzari et al.
47 ([2020](#)). In general, the post-processing pipeline has been somewhat complicated, as when
48 the data is needed in the SEM format, for example, to calculate derivatives of velocity fields,
49 the solver itself has been used in a “post-processing” mode. This mode uses the solver and
50 additional Fortran code that needs to be compiled to produce smaller files that can be used in
51 Matlab or Python with e.g. PyMech by Mohanan et al. ([2022](#)), to perform signal processing,
52 create plots, etc. NekRS by Fischer et al. ([2022](#)), a GPU version of Nek5000, and Neko
53 ([Jansson et al., 2023, 2024](#)), a modern Fortran implementation of SEM have followed the
54 same approach, motivating the necessity of our PySEMTTools for the future.

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

62 Features

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

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

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

88 Acknowledgements

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

95 References

- 96 Dalcín, L., Paz, R., & Storti, M. (2005). MPI for python. *Journal of Parallel and Distributed
97 Computing*, 65(9), 1108–1115. <https://doi.org/10.1016/j.jpdc.2005.03.010>
- 98 El Khoury, G. K., Schlatter, P., Noorani, A., Fischer, P. F., Brethouwer, G., & Johansson, A.
99 V. (2013). Direct numerical simulation of turbulent pipe flow at moderately high reynolds
100 numbers. *Flow, Turbulence and Combustion*, 91(3), 475–495. <https://doi.org/10.1007/s10494-013-9482-8>
- 102 Fischer, P., Kerkemeier, S., Min, M., Lan, Y.-H., Phillips, M., Rathnayake, T., Merzari,
103 E., Tomboulides, A., Karakus, A., Chalmers, N., & Warburton, T. (2022). NekRS, a
104 GPU-accelerated spectral element navier-stokes solver. *Parallel Computing*, 114, 102982.
105 <https://doi.org/doi.org/10.1016/j.parco.2022.102982>
- 106 Fischer, P., Loth, F., Lee, S.-W., Smith, D., Tufo, H., & Bassiouny, H. (2006). Parallel
107 simulation of high reynolds number vascular flows. In A. Deane, A. Ecer, J. McDonough, N.
108 Satofuka, G. Brenner, D. R. Emerson, J. Periaux, & D. Tromeur-Dervout (Eds.), *Parallel
109 computational fluid dynamics 2005* (pp. 219–226). Elsevier. <https://doi.org/10.1016/B978-044452206-1/50026-4>
- 111 Fischer, P., Lottes, J. W., & Kerkemeier, S. G. (2008). *Nek5000 web page*.
- 112 Godoy, W. F., Podhorszki, N., Wang, R., Atkins, C., Eisenhauer, G., Gu, J., Davis, P., Choi,
113 J., Germaschewski, K., Huck, K., & others. (2020). Adios 2: The adaptable input output
114 system. A framework for high-performance data management. *SoftwareX*, 12, 100561.
115 <https://doi.org/10.1016/j.softx.2020.100561>
- 116 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,
117 Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,
118 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,
119 T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 121 Jansson, N., Karp, M., Perez, A., Mukha, T., Ju, Y., Liu, J., Páll, S., Laure, E., Weinkauf, T.,
122 Schumacher, J., Schlatter, P., & Markidis, S. (2023). Exploring the ultimate regime of
123 turbulent rayleigh–bénard convection through unprecedented spectral-element simulations.
124 *Proceedings of the International Conference for High Performance Computing, Networking,
125 Storage and Analysis*. <https://doi.org/10.1145/3581784.3627039>
- 126 Jansson, N., Karp, M., Podobas, A., Markidis, S., & Schlatter, P. (2024). Neko: A modern,
127 portable, and scalable framework for high-fidelity computational fluid dynamics. *Computers
128 & Fluids*, 275, 106243. <https://doi.org/10.1016/j.compfluid.2024.106243>
- 129 Mallor, F., Vinuesa, R., Örlü, R., & Schlatter, P. (2024). High-fidelity simulations of the flow
130 around a NACA 4412 wing section at high angles of attack. *International Journal of Heat
131 and Fluid Flow*, 110, 109590. <https://doi.org/10.1016/j.ijheatfluidflow.2024.109590>
- 132 Merzari, E., Fischer, P., Min, M., Kerkemeier, S., Obabko, A., Shaver, D., Yuan, H., Yu, Y.,
133 Martinez, J., Brockmeyer, L., Fick, L., Busco, G., Yıldız, A., & Hassan, Y. (2020). Toward

- 134 exascale: Overview of large eddy simulations and direct numerical simulations of nuclear
135 reactor flows with the spectral element method in Nek5000. *Nuclear Technology*, 206(9),
136 1308–1324. <https://doi.org/10.1080/00295450.2020.1748557>
- 137 Mohanan, A. V., Chauvat, G., Kleine, V. G., Fabbiane, N., & Canton, J. (2022). Pymech:
138 A python software suite for Nek5000 and SIMSON. In *GitHub repository*. GitHub. <https://doi.org/10.5281/zenodo.7358961>
- 140 Patera, A. T. (1984). A spectral element method for fluid dynamics: Laminar flow in a channel
141 expansion. *Journal of Computational Physics*, 54(3), 468–488. [https://doi.org/10.1016/0021-9991\(84\)90128-1](https://doi.org/10.1016/0021-9991(84)90128-1)

DRAFT