The Journal of Open Source Software

OGSTools: A Python package for OpenGeoSys

Tobias Meisel [1*], **Florian Zill** [2,1*], **Julian Heinze** [1*], **Lars Bilke** [1*], **Max Jäschke** [3,1,4*], **Feliks Kiszkurno** [2,1*], **Dominik Kern** [2*], and **Jörg Buchwald** [1*]

1 Helmholtz Centre for Environmental Research, Germany **2** TU Bergakademie Freiberg, Germany **3** Leipzig University of Applied Sciences, Germany **4** Technische Universität Dresden, Germany * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

`OGSTools` (`OpenGeoSys` Tools) is a Python library for streamlined usage of `OpenGeoSys 6` (OGS) - a software for simulating thermo-hydro-mechanical-chemical (THMC) processes in porous and fractured media [Bilke et al. (2025)] [Kolditz et al. (2012)]. `OGSTools` [Zill et al. (2025)] provides an interface between OGS-specific data and well-established data structures of the Python ecosystem, as well as domain-specific solutions, examples, and tailored defaults for OGS users and developers. By connecting OGS to the ecosystem of Python, the entry threshold to the OGS platform is lowered for users with different levels of expertise. The libraries' functionalities are designed to be used in complex automated workflows (including pre- and post-processing), the OGS benchmark gallery, the OGS test-suite, and in automating repetitive tasks in the model development cycle.

Statement of need

Development efficiency

Modellers of OGS iteratively run simulations, analyse results, and refine their models with the goal to enhance the accuracy, efficiency and reliability of the simulation results. To improve efficiency, repetitive steps in the model development cycle should be formalized and implemented - in the case of OGSTools as a Python library. The Python library introduced here serves as a central platform to collect and improve common functionalities needed by modellers of OGS.

Complex workflows

A workflow is a structured sequence of steps, that processes data and executes computations to achieve a specific goal [Diercks et al. (2022)]. In our scientific research, workflows need to integrate multiple steps—such as geological data preprocessing, ensemble simulations with OGS, domain-specific analysis and visualization—into complex and fully automated and therefore reproducible sequences. Typically, one specific workflow is implemented to answer one specific scientific question. Workflow-based approaches have been proven to adhere to the `FAIR principles` [Goble et al. (2020)], [Wilkinson et al. (2025)]. The typical approach is to use existing workflow management software that covers domain-independent parts like dependency graph description, computational efficiency, data management, execution control, multi-user collaboration and data provenance [(**?**)]. When building upon the Python ecosystem, Python-based workflow managers such as `Snakemake` [Köster & Rahmann (2012)] and `AiiDA` [Huber et al. (2020)] are a natural choice.

The usage of workflow managers shifts the actual development to the workflow components. Common and frequently used functionality found within workflow components is made reusable and provided in this Python library. It focuses on functionalities directly related to (1) the OGS core simulator and its specific input and output data, (2) domain-specific definitions in geo-science, (3) finite element modelling (FEM), and (4) numerical computation. The workflow components are constructed from generic Python libraries, OGSTools, and integration code tailored to the respective workflow manager chosen. To ensure integration of the library's functionalities with the workflow management software, a list of functional and non-functional requirements (e.g., thread safety), imposed by workflow management software, is maintained and regularly validated through continuous testing.

Test suite

`OGSTools` provides functionality for (1) setting up test environments, (2) executing OGS under specified conditions, (3) evaluating OGS results against defined test criteria, and (4) monitoring the testing process.

Educational Jupyter notebooks

OGS is already being used in academic courses and teaching environments. With Jupyter Notebooks, students can explore interactive learning environments where they directly modify parameters, material laws, and other influencing factors, and instantly visualize the outcomes. `OGSTools` serves as an interface between OGS and Jupyter Notebooks. It supports the creation of input data—such as easily configurable meshes or ready-to-use project files.

Centralization of Python-related development

Previously, the code base for Python-related tasks in OGS was fragmented, with components that were often developed for specific use cases, with varying degrees of standardization and sharing. The lack of centralization led to inefficiencies, inconsistent quality, and challenges in maintaining and extending the code. With `OGSTools`, reusable Python code is now centralized under the professional maintenance of the core developer team of OGS. It ensures better collaboration, standardized practices, and improved code quality. Further, it enables the transfer of years of experience in maintaining the OGS' core [Bilke et al. (2019)] to the pre- and post-processing code. For the centralized approach, preceding work from `msh2vtu` [Kern & Bilke (2022)], `ogs6py` and `VTUInterface` [Buchwald et al. (2021)] and extracted functionalities from the projects (1) `AREHS` [Meisel et al. (2024)], and (2) `OpenWorkFlow - Synthesis Platform` [Environmental Research UFZ (2023)] have been adapted and integrated into `OGSTools`.

To address `The Need for a Versioned Data Analysis Software Environment` [Blomer et al. (2014)] `OGSTools` provides additionally a pinned environment, updated at least once per release. While reproducibility requires environments with pinned dependencies, `OGSTools` is additionally tested with the latest dependencies, to receive early warnings of breaking changes and to support the long-term sustainability of the codebase.

To support broad adoption within the OGS user community, the library is deliberately integrated at key points of interest, such as the official OGS benchmarks, executable test cases, and further contexts where previously used libraries were employed.

Features

The implemented features are covering (1) pre-processing, (2) setup and execution of simulations, and (3) post-processing.

Preprocessing (1) for OGS includes mesh creation, adaptation, conversion, and refinement, as well as defining boundary conditions, s ource terms, and generating project files (OGS specific XML-Files). Building upon this, a `FEFLOW` converter (from `FEFLOW` models to OGS models) is integrated [Heinze et al. (2025)]. The converter uses the geometric and material data of FEFLOW models to generate OGS-suitable meshes and definitions for H, HT and HC processes.

Postprocessing (3) includes domain specific evaluation and visualization of simulation results, for temporal and spatial distribution analysis. `OGSTools` helps to create detailed plots by defining sensible defaults and OGS-specific standards. It offers functionalities for the comparison of numerical simulation results with experimental data or analytical solutions. Just as preprocessing and analysis are essential for single simulations, tooling becomes critical for efficiently handling ensemble runs. Ensemble runs enable evaluation of model robustness, parameter sensitivity, numerical behaviour, and computational efficiency, with spatial and temporal grid conversion currently supported.

A more complete list of examples covering a significant part of the feature set is found in the online documentation of OGSTools [1]. Containers are provided for reproducibility, benefiting both developers and users ([(**?**)]). Like `OpenGeoSys`, `OGSTools` is available on `PyPI` and `Conda`.

Applications

`OGSTools` library is designed to aid users in the implementation of complex workflows and has been an integral part of several scientific projects utilizing them.

AREHS

The AREHS-Project (effects of changing boundary conditions on the development of hydro-geological systems: numerical long-term modelling considering thermal–hydraulic–mechan-ical(–chemical) coupled effects) project [Kahnt et al. (2021)] is focused on modelling the effects of the glacial cycle on hydro-geological parameters in potential geological nuclear waste repositories in Germany. [Zill et al. (2024)] and [Silbermann et al. (2025)] highlighted the importance of automated workflow to efficiently develop models to answer the scientific question and to ensure the reproducibility of the results. This workflow covers all necessary steps from a structured layered model and geological parameters over the simulation with OGS to the resulting figures shown in [Zill et al. (2024)] and [Silbermann et al. (2025)]. It is composed as a Snakemake workflow and all material is available on [Meisel et al. (2024)].

OpenWorkflow

`OpenWorkFlow` [Environmental Research UFZ (2023)], is a project for an open-source, modular synthesis platform designed for safety assessment in the nuclear waste site selection procedure of Germany. Automated workflows as a piece of the planned scientific computational basis for investigating repository-induced physical and chemical processes in different geological setting are essential for transparent and reproducible simulation results. OGS together with `OGSTools` has been used in a study of thermal repository dimensioning - named `ThEDi`. `ThEDi` focuses on determining the optimal packing of disposal containers in a repository to ensure temperature limits are not exceeded. The fully automated workflow generates the simulation models based on geometric and material data, runs and analyses the simulations. For scalability and parallelization the workflow is embedded optionally within the workflow management `Snakemake`. The workflow components are implemented reusing `OGSTools` functionalities.

OpenGeoSys benchmarks

The OGS benchmark gallery is a collection of web documents (mostly generated from `Jupyter Notebooks`) that demonstrate, how users can set up, adjust, execute, and analyse simulations. They can be downloaded, executed, and adapted in an interactive environment for further exploration. With `OGSTools` code complexity and code duplication has been reduced, and it allows especially inexperienced users to focus on the important part of the notebook.

[1] https://ogstools.opengeosys.org

The code transformation of the Mandel-Cryer effect benchmark, based on a simplified merge request[2], demonstrates that using predefined plotting utilities reduces technical overhead.

Sections of the benchmark without OGSTools :

```
# Mandel-Cryer effect benchmark
# Load the result
pvdfile = vtuIO.PVDIO("results_MandelCryerStaggered.pvd", dim=3)
fields = pvdfile.get_point_field_names()
time = pvdfile.timesteps

# Define observation points
observation_points = {"center": (0, 0, 0)}
pressure = pvdfile.read_time_series("pressure", observation_points)

# Plot pressure over time
fig, ax1 = plt.subplots(figsize=(10, 8))
ax1.plot(time, pressure["center"], color="tab:orange")
ax1.set_ylabel("Pressure (Pa)", fontsize=20)
ax1.set_xlabel("Time (s)", fontsize=20)
ax1.set_xlim(0, t_end)
ax1.set_ylim(0, 1500)
ax1.grid()
fig.supxlabel("Pressure at center of sphere")
plt.show()
```

By abstracting away lower-level plotting code, users can focus more directly on the physical interpretation of the benchmark results. Sections of the benchmark with OGSTools:

```
# Mandel-Cryer effect benchmark without OGSTools
import ogstools as ot

ms = ot.MeshSeries("results_MandelCryerStaggered.pvd")

# Plot pressure over time
center_point = [0, 0, 0]
fig = ms.plot_probe(center_point, ot.variables.pressure, labels=["Center"])
```

OGS-GIScape

`OGS-GIScape` is a `Snakemake`-based workflow for creating, simulating and analysing numerical groundwater models (NGM). OGS-GIScape enables scientists to investigate complex environmental models or conduct scenario analyses to study the groundwater flow and the associated environmental impact due to changes in groundwater resources. Furthermore, the outcome of the models could be used for the management of groundwater resources.

An important part of the NGM creation is the geometric model (mesh). It is build using geographic information system (GIS) tools at the landscape scale and combining various meshing tools. The workflow also comprises the parametrization of the geometric model with physical parameters as well as defining boundary conditions, for instance groundwater recharge at the top of the computational domain or the integration of rivers. For these workflow steps it is mainly necessary to change parts of the OGS project file which is done with `OGSTools`.

[2]https://gitlab.opengeosys.org/ogs/ogs/-/merge_requests/5171

Acknowledgements

This work has been supported by multiple funding sources, including AREHS under grant 4719F10402 by Bundesamt für die Sicherheit der nuklearen Entsorgung(BASE), and OpenWorkflow under grant STAFuE-21-05-Klei by Bundesgesellschaft für Endlagerung (BGE). The authors also acknowledge ongoing support from SUTOGS (Streamlining Usability and Testing of OpenGeoSys) under (grant [Grant Number]) by Deutsche Forschungsgemeinschaft (DFG)

Bilke, L., Flemisch, B., Kalbacher, T., Kolditz, O., Helmig, R., & Nagel, T. (2019). Development of Open-Source Porous Media Simulators: Principles and Experiences. *Transport in Porous Media*, *130*(1), 337–361. https://doi.org/10.1007/s11242-019-01310-1

Bilke, L., Naumov, D., Wang, W., Fischer, T., Kiszkurno, F. K., Lehmann, C., Max, J., Zill, F., Buchwald, J., Grunwald, N., Kessler, K., Aubry, L., Dörnbrack, M., Nagel, T., Ahrendt, L., Kaiser, S., & Meisel, T. (2025). *OpenGeoSys* (Version 6.5.4). Zenodo. https://doi.org/10.5281/zenodo.14672997

Blomer, J., Berzano, D., Buncic, P., Charalampidis, I., Ganis, G., Lestaris, G., & Meusel, R. (2014). The need for a versioned data analysis software environment. *CoRR*, *abs/1407.3063*. http://arxiv.org/abs/1407.3063

Buchwald, J., Kolditz, O., & Nagel, T. (2021). ogs6py and VTUinterface: Streamlining OpenGeoSys workflows in python. *Journal of Open Source Software*, *6*(67), 3673. https://doi.org/10.21105/joss.03673

Diercks, P., Gläser, D., Lünsdorf, O., Selzer, M., Flemisch, B., & Unger, J. F. (2022). *Evaluation of tools for describing, reproducing and reusing scientific workflows*. https://arxiv.org/abs/2211.06429

Environmental Research UFZ, H. C. for. (2023). *OpenWorkFlow - synthesis platform*. https://www.ufz.de/index.php?en=48378

Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M. R., Peters, K., & Schober, D. (2020). FAIR computational workflows. *Data Intelligence*, *2*(1-2), 108–131. https://doi.org/10.1162/dint_a_00033

Heinze, J., Lehmann, C., Meisel, T., Rink, K., Kreye, P., Renz, A., Zeunert, S., & Rühaak, W. (2025). Combining FEFLOW and OpenGeoSys for interoperable workflows in environmental geotechnics. *Environmental Earth Sciences*, *84*(16), 457. https://doi.org/10.1007/s12665-025-12380-4

Huber, S. P., Zoupanos, S., Uhrin, M., Talirz, L., Kahle, L., Häuselmann, R., Gresch, D., Müller, T., Yakutovich, A. V., Andersen, C. W., Ramirez, F. F., Adorf, C. S., Gargiulo, F., Kumbhar, S., Passaro, E., Johnston, C., Merkys, A., Cepellotti, A., Mounet, N., … Pizzi, G. (2020). AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. *Scientific Data*, *7*(1). https://doi.org/10.1038/s41597-020-00638-4

Kahnt, R., Konietzky, H., Nagel, T., Kolditz, O., Jockel, A., Silbermann, C., Tiedke, F., Meisel, T., Rink, K., Wang, W., Zill, F., Carl, A., Gabriel, A., Schlegel, M., & Abraham, T. (2021). AREHS: Effects of changing boundary conditions on the development of hydrogeological systems: Numerical long-term modelling considering thermal–hydraulic–mechanical(–chemical) coupled effects. *Safety of Nuclear Waste Disposal*, *1*, 175–177. https://doi.org/10.5194/sand-1-175-2021

Kern, D., & Bilke, L. (2022). msh2vtu. In *GitHub repository*. GitHub. https://github.com/dominik-kern/msh2vtu

Kolditz, O., Bauer, S., Bilke, L., Grunwald, N., Delfs, J.-O., Fischer, T., Görke, U., Kalbacher, T., Kosakowski, G., Mcdermott, C., Park, C.-H., Radu, F., Rink, K., Shao, H., Sun, F., Sun, Y., Singh, A., Taron, J., Walther, M., & Zehner, B. (2012). OpenGeoSys:

An open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/c) processes in porous media. *Environmental Earth Sciences*, *67*, 589–599. https://doi.org/10.1007/s12665-012-1546-x

Köster, J., & Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, *28*(19), 2520–2522. https://doi.org/10.1093/bioinformatics/bts480

Meisel, T., Zill, F., Silbermann, C. B., Wang, W., Bilke, L., & Kern, D. (2024). *AREHS - OpenGeoSys workflow* (Version 1.0). Zenodo. https://doi.org/10.5281/zenodo.11367280

Silbermann, C., Zill, F., Meisel, T., Kern, D., Kolditz, O., Magri, F., & Nagel, T. (2025). Automated thermo-hydro-mechanical simulations capturing glacial cycle effects on nuclear waste repositories in clay rock. *Geomechanics and Geophysics for Geo-Energy and Geo-Resources*, *11*. https://doi.org/10.1007/s40948-025-00960-4

Wilkinson, S. R., Aloqalaa, M., Belhajjame, K., Crusoe, M. R., Paula Kinoshita, B. de, Gadelha, L., Garijo, D., Gustafsson, O. J. R., Juty, N., Kanwal, S., Khan, F. Z., Köster, J., Peters-von Gehlen, K., Pouchard, L., Rannow, R. K., Soiland-Reyes, S., Soranzo, N., Sufi, S., Sun, Z., … Goble, C. (2025). Applying the FAIR principles to computational workflows. *Scientific Data*, *12*(1). https://doi.org/10.1038/s41597-025-04451-9

Zill, F., Bilke, L., Meisel, T., Heinze, J., Kiszkurno, F. K., Kern, D., Jäschke, M., & Lehmann, C. (2025). *OGSTools* (Version 0.7.0). Zenodo. https://doi.org/10.5281/zenodo.15804988

Zill, F., Silbermann, C., Meisel, T., Magri, F., & Nagel, T. (2024). Far-field modelling of THM processes in rock salt formations. *Open Geomechanics*, *5*, 1–16. https://doi.org/10.5802/ogeo.20