

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 Kizskurno^{2,1*}, Norbert Grunwald^{1*}, Thomas Fischer^{1*}, 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](https://doi.org/10.xxxxxx/draft)

Software

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

Editor: [Open Journals](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

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

Summary

OGSTools (OpenGeoSys Tools) is a Python library for pre- and post-processing of OpenGeoSys 6 (OGS) — a software package for simulating thermo-hydro-mechanical-chemical (THMC) processes in porous and fractured media (Bilke, Naumov, et al., 2025; Kolditz et al., 2012). OGSTools (Meisel 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 for OGS users and developers. The library's functionalities are designed to be used in the OGS benchmark gallery, the OGS test suite, and for automating repetitive tasks in the model development cycle — from simple daily tasks to complex automated workflows.

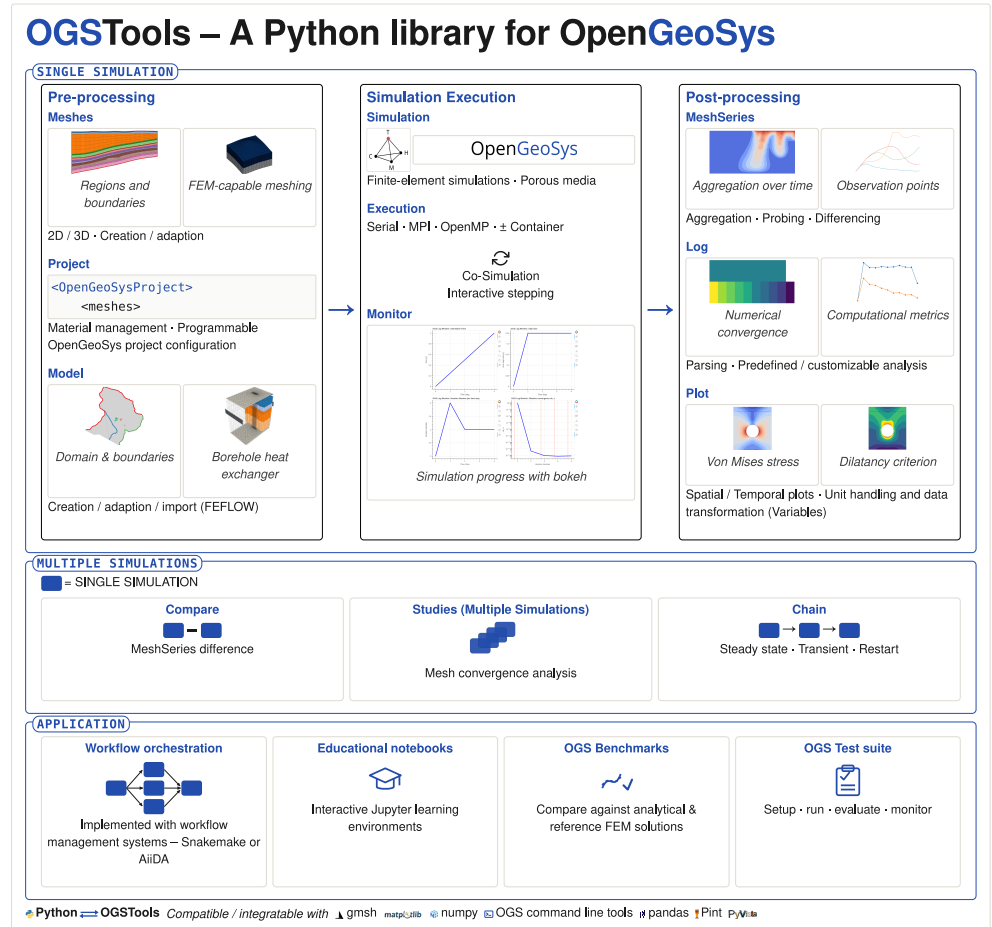


Figure 1: OGSTools graphical abstract: pre-processing, simulation execution, and post-processing for a single simulation; combining multiple simulations; and application areas.

Statement of need

Development efficiency

Modellers of OGS iteratively run simulations, analyse results, and refine their models. To improve efficiency, repetitive steps in the model development cycle are formalised in Python, matching the existing expertise of the user base. The Python library introduced here serves as a central platform to collect and improve common functionalities needed by modellers of OGS.

Complex workflows

In our scientific research, workflows integrate multiple steps — geological data preprocessing, ensemble simulations with OGS, domain-specific analysis and visualisation — into fully automated, reproducible sequences. Workflow-based approaches adhere to the FAIR principles (Goble et al., 2020; Wilkinson et al., 2025), using workflow management software for dependency management, execution control, and data provenance (Bilke, Fischer, et al., 2025). Building on Python-based workflow managers like Snakemake (Köster & Rahmann, 2012) and AiIDA (Huber et al., 2020), OGSTools provides reusable, domain-specific functionality as the shared foundation across workflow components.

32 Test suite

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

36 Educational Jupyter notebooks

37 OGS is well suited for academic courses and teaching environments. With Jupyter Notebooks,
38 students can explore interactive learning environments where they directly modify parameters,
39 material laws, and other influencing factors, and instantly visualise the outcomes. OGSTools
40 reduces the boilerplate and keeps notebooks focused on the learning objective.

41 State of the field

42 Simulator-specific companion libraries have emerged as a recurring pattern across scientific
43 computing domains. These software packages bridge a domain-specific simulator to a general-
44 purpose programming language ecosystem (e.g. Python), typically to cover pre-processing,
45 execution, and post-processing conducted on a single programmatic platform.

46 In subsurface hydrology, FloPy ([Hughes et al., 2024](#)) wraps the MODFLOW family of
47 groundwater flow and transport models, supporting model creation, execution, and result
48 analysis. pyGSFLOW ([Larsen et al., 2022](#)) provides equivalent functionality for the GSFLOW
49 integrated hydrologic model. toughio ([Luu, 2020](#)) covers pre- and post-processing for the
50 TOUGH simulator family. In energy systems modelling, otoole ([Barnes & Usher, 2023](#)) supports
51 users of OSeMOSYS to formalise pre- and post-processing tasks. DOLFINx ([Baratta et al.,
52 2023](#)) is worth noting despite a fundamental architectural difference: rather than wrapping
53 an external solver such as OGS, it exposes FEM assembly and solving directly through a
54 Python API. It partially shares the same tooling ecosystem as OGSTools — gmsh, PyVista,
55 and VTK/XDMF.

56 OGSTools follows the simulator-specific companion library pattern, here for OpenGeoSys.

57 An alternative to scripting-based companion libraries is a dedicated companion GUI — as with
58 ModelMuse ([Winston, 2019](#)) for MODFLOW and the DataExplorer ([Rink et al., 2012](#)) for
59 OpenGeoSys.

60 Build vs. contribute

61 OGSTools contains only functionality that is explicitly specific to [OpenGeoSys](#) — domain-
62 specific data structures, OGS input/output formats, and process-specific defaults. General-
63 purpose functionality is deliberately left to established libraries (PyVista, Pandas, NumPy,
64 Pint), which OGSTools relies on.

65 Previously, without any centralisation to contribute OGS-specific pre- and postprocessing
66 code, the code base for Python-related tasks in OGS was fragmented, with components often
67 developed for specific use cases and varying degrees of standardisation, quality and maintenance
68 efforts. Further, OGSTools enables the transfer of years of experience in maintaining the OGS
69 core ([Bilke et al., 2019](#)) to the pre- and post-processing code. For the centralised approach,
70 preceding work on msh2vtu ([Kern & Bilke, 2022](#)), ogs6py and VTUInterface ([Buchwald et
71 al., 2021](#)) and further not yet published functionalities have been adapted and integrated into
72 OGSTools.

73 Software Design

74 Design choices

75 The functionality is grouped thematically into sub-libraries. Beyond general software engineering
76 best practices, the following design principles deserve particular attention.

77 **Open interfaces to common Python libraries:** Each sub-library either transforms OpenGeoSys
 78 specific data into common Python data structures (e.g. PyVista, Pandas, NumPy, Matplotlib,
 79 Pint), or vice versa. Users can use any subset of the library without lock-in, including when
 80 preferring to run OpenGeoSys from the command line.

81 **Reuse OGS command line tools:** The new functionality combines the OGS command line
 82 tools ¹ to cover more complex tasks than any single tool can handle alone.

83 **Fail loudly:** Silently producing wrong results is the highest risk in our simulation workflows.
 84 OGSTools therefore raises errors immediately when constraints or plausibility checks are violated,
 85 prioritising early failure over silent pass-through.

86 **Large data awareness:** The top-level API is designed to be approachable for small datasets at
 87 entry level, while experts working with large data can tune performance via lower-level API
 88 access or fall back to command-line tools and custom code.

89 **Infrastructure:** Code development is centralised on a self-hosted GitLab instance. A multi-
 90 platform CI pipeline (Linux, Windows, macOS) enforces code quality and reports test coverage.
 91 Examples are organised as plain Python scripts automatically converted to Jupyter notebooks,
 92 with Binder integration for every release. To address the need for a versioned analysis
 93 environment (Blomer et al., 2014), OGSTools ships a pinned dependency environment updated
 94 with every release, while continuously tested against the latest dependencies.

95 Example

96 The following example shows a complete OGS Liquid Flow ² simulation workflow, adapted to
 97 2D from ³. First, an OGS-capable mesh is generated and pressure boundary conditions are
 98 assigned to the boundary meshes (Figure 2), using standard PyVista [Sullivan & Kaszynski
 99 (2019)] functionality. After execution of the simulation, convergence metrics (Figure 3) and
 100 the final pressure distribution (Figure 4) are visualised. An extended version of this example is
 101 available in the OGSTools documentation ⁴.

```
import numpy as np
import ogstools as ot
from ogstools.examples import load_project_simple_lf

# 1. Pre-processing: Load example Project and construct input meshes
project = load_project_simple_lf()
meshes = ot.Meshes.from_gmsh(ot.gmsh_tools.rect((8,4),8,2))
# Set boundary conditions on the pyvista meshes
meshes["left"].point_data["pressure"] = 2.9e7
meshes["right"].point_data["pressure"] = 3.1e7
model = ot.Model(project=project, meshes=meshes)
# Visualize setup with boundary conditions (Figure 2)
model.plot_constraints()

# 2. Run: Execute Simulation
sim = model.run()

# 3. Post-processing: Analyse results

# Plot convergence behaviour (Figure 3)
sim.log.plot_convergence()
```

¹<https://www.opengeosys.org/6.5.8/docs/tools/getting-started/overview/>

²<https://www.opengeosys.org/6.5.8/docs/processes/liquid-flow/liquidflow/>

³<https://www.opengeosys.org/6.5.8/docs/benchmarks/liquid-flow/primary-variable-constrain-dirichlet-boundary-condition/>

⁴https://ogstools.opengeosys.org/0.8.1/auto_examples/howto_quickstart/plot_framework.html

```
# Plot final pressure distribution (Figure 4)
ot.plot.contourf(sim.meshseries[-1], "pressure")
```

```
# 4. Store: Save Simulation
sim.save(id = "mysim", archive=True)
```

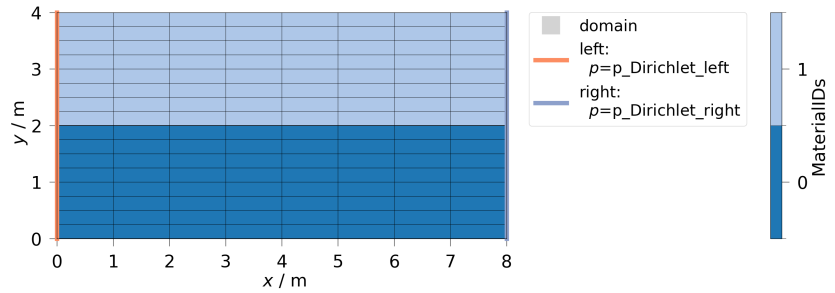


Figure 2: Initial boundary conditions.

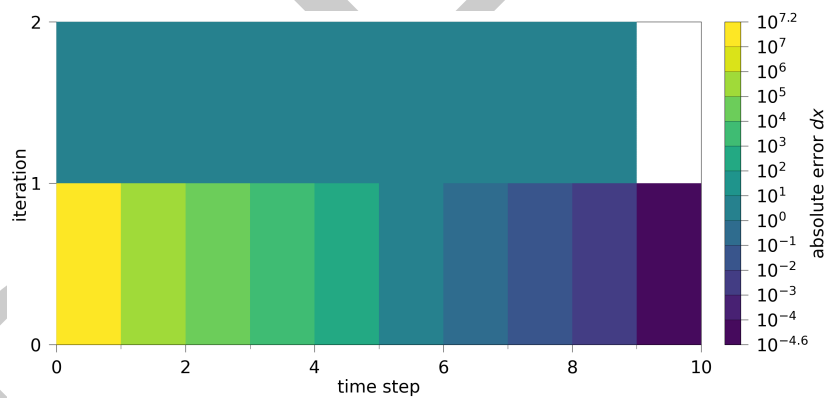


Figure 3: Resulting convergence metrics.

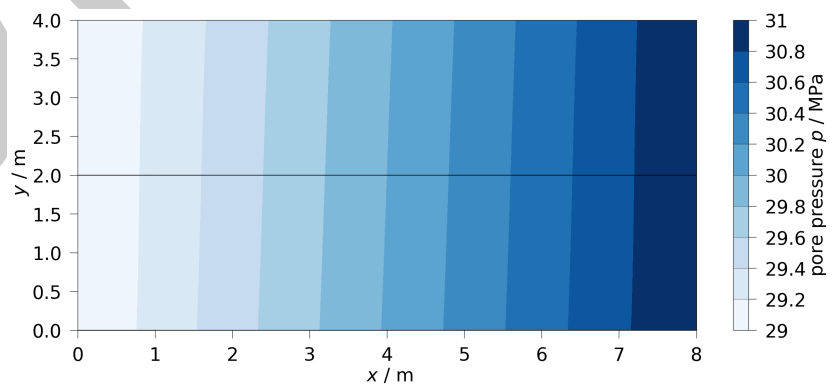


Figure 4: Resulting pressure distribution.

102 Features

103 The implemented features cover pre-processing, setup and execution of simulations, and
104 post-processing.

Pre-processing for OGS includes mesh creation, adaptation, conversion, as well as defining boundary conditions, source terms, and generating project files (OGS-specific XML files). OGSTools further provides a material management component that allows process-specific material definitions to be assembled from structured YAML sources and translated into OGS-compatible project file entries. This allows a consistent, database-like handling of material parameters across workflows, test cases, and educational examples, while separating physical model definitions from project file syntax. In addition, a FEFLOW converter (from FEFLOW models to OGS models) is integrated (Heinze et al., 2025).

The simulation execution part covers running simulations with the OGS core via the command line and Python-based co-simulation interfaces. Runtime features include monitoring, interactive stepping, and access to intermediate results for in-simulation analysis.

Post-processing includes domain-specific evaluation and visualisation of simulation results for temporal and spatial distribution analysis, with sensible defaults and OGS-specific standards for plotting, and comparison against experimental data or analytical solutions.

The complete feature list is found in the online documentation⁵. Containers are provided for reproducibility, benefiting both developers and users (Bilke, Fischer, et al., 2025). Like OpenGeoSys, OGSTools is available on PyPI and Conda.

Research impact

Workflows

OGSTools emerged from and is used in the following research projects. The AREHS-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. Within this project, Zill et al. (2024) and Silbermann et al. (2025) conducted their work using automated OGSTools workflows, with all material available at Meisel et al. (2024). OpenWorkflow (Lehmann et al., 2025) is a project for an open-source, modular synthesis platform designed for safety assessment in the nuclear waste site selection procedure of Germany. ThEDi, a completed study on optimal disposal container packing to meet repository temperature limits, is one of multiple studies within OpenWorkflow, mostly implemented using OGSTools.

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 are well-suited as a starting point of research, and can be downloaded, executed, and adapted interactively. With OGSTools, code complexity and code duplication have been reduced, allowing especially inexperienced users to focus on the important part of the notebook.

AI usage disclosure

In the writing of this manuscript, the authors used Anthropic's Claude and OpenAI's ChatGPT for grammar and spelling corrections. For all contributions to the software project the use of Large Language Models (LLMs) is in principle permitted, but all contributions must pass through a review process by the developers, maintainers, and authors.

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

⁵<https://ogstools.opengeosys.org>

and Testing of OpenGeoSys) under grant 528785032⁶ by Deutsche Forschungsgemeinschaft (DFG)

References

- Baratta, I. A., Dean, J. P., Dokken, J. S., Habera, M., Hale, J. S., Richardson, C. N., Rognes, M. E., Scroggs, M. W., Sime, N., & Wells, G. N. (2023). *DOLFINx: The next generation FEniCS problem solving environment*. Zenodo. <https://doi.org/10.5281/zenodo.10447666>
- Barnes, T., & Usher, W. (2023). Otoole: OSeMOSYS tools for energy work. *Journal of Open Source Software*, 8(92), 5511. <https://doi.org/10.21105/joss.05511>
- Bilke, L., Fischer, T., Naumov, D., & Meisel, T. (2025). Reproducible HPC software deployments, simulations, and workflows – a case study for far-field deep geological repository assessment. *Environmental Earth Sciences*, 84(17), 502. <https://doi.org/10.1007/s12665-025-12501-z>
- 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., Kiszkurko, 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. <https://doi.org/10.48550/arXiv.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>
- 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>
- Hughes, J. D., Langevin, C. D., Paulinski, S. R., Larsen, J. D., & Brakenhoff, D. (2024). FloPy workflows for creating structured and unstructured MODFLOW models. *Groundwater*, 62(1), 124–139. <https://doi.org/10.1111/gwat.13327>
- 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>

⁶<https://gepris.dfg.de/gepris/projekt/528785032>

- 195 Kern, D., & Bilke, L. (2022). msh2vtu. In *GitHub repository*. GitHub. [https://github.com/](https://github.com/dominik-kern/msh2vtu)
196 [dominik-kern/msh2vtu](https://github.com/dominik-kern/msh2vtu)
- 197 Kolditz, O., Bauer, S., Bilke, L., Grunwald, N., Delfs, J.-O., Fischer, T., Görke, U., Kalbacher,
198 T., Kosakowski, G., Mcdermott, C., Park, C.-H., Radu, F., Rink, K., Shao, H., Sun,
199 F., Sun, Y., Singh, A., Taron, J., Walther, M., & Zehner, B. (2012). OpenGeoSys:
200 An open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical
201 (THM/c) processes in porous media. *Environmental Earth Sciences*, 67, 589–599. <https://doi.org/10.1007/s12665-012-1546-x>
202
- 203 Köster, J., & Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine.
204 *Bioinformatics*, 28(19), 2520–2522. <https://doi.org/10.1093/bioinformatics/bts480>
- 205 Larsen, J. D., Alzaiee, A., & Niswonger, R. G. (2022). Integrated hydrologic model
206 development and postprocessing for GSFLOW using pyGSFLOW. *Journal of Open Source*
207 *Software*, 7(72), 3852. <https://doi.org/10.21105/joss.03852>
- 208 Lehmann, C., Bilke, L., Graebing, N., Heinze, J., Meisel, T., Naumov, D., Sen, Ö. O., &
209 Kolditz, O. (2025). *Software products from the OpenWorkFlow project*. EGU General
210 Assembly 2025, Vienna, Austria, 27 Apr–2 May 2025, EGU25-15544. <https://doi.org/10.5194/egusphere-egu25-15544>
211
- 212 Luu, K. (2020). Toughio: Pre- and post-processing python library for TOUGH. *Journal of*
213 *Open Source Software*, 5(51), 2412. <https://doi.org/10.21105/joss.02412>
- 214 Meisel, T., Zill, F., Jäschke, M., Bilke, L., Grunwald, N., Fischer, T., & Kiszkurno, F. K.
215 (2025). *OGSTools* (Version 0.7.1). Zenodo. <https://doi.org/10.5281/zenodo.17223939>
- 216 Meisel, T., Zill, F., Silbermann, C. B., Wang, W., Bilke, L., & Kern, D. (2024). *AREHS -*
217 *OpenGeoSys workflow* (Version 1.0). Zenodo. <https://doi.org/10.5281/zenodo.11367280>
- 218 Rink, K., Kalbacher, T., & Kolditz, O. (2012). Visual data exploration for hydrological analysis.
219 *Environmental Earth Sciences*, 65(5), 1395–1403. <https://doi.org/10.1007/s12665-011-1230-6>
220
- 221 Silbermann, C., Zill, F., Meisel, T., Kern, D., Kolditz, O., Magri, F., & Nagel, T. (2025).
222 Automated thermo-hydro-mechanical simulations capturing glacial cycle effects on nuclear
223 waste repositories in clay rock. *Geomechanics and Geophysics for Geo-Energy and Geo-*
224 *Resources*, 11. <https://doi.org/10.1007/s40948-025-00960-4>
- 225 Sullivan, C., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a
226 streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*,
227 4(37), 1450. <https://doi.org/10.21105/joss.01450>
- 228 Wilkinson, S. R., Aloqalaa, M., Belhajjame, K., Crusoe, M. R., Paula Kinoshita, B. de, Gadelha,
229 L., Garijo, D., Gustafsson, O. J. R., Juty, N., Kanwal, S., Khan, F. Z., Köster, J., Peters-von
230 Gehlen, K., Pouchard, L., Rannow, R. K., Soiland-Reyes, S., Soranzo, N., Sufi, S., Sun, Z.,
231 ... Goble, C. (2025). Applying the FAIR principles to computational workflows. *Scientific*
232 *Data*, 12(1). <https://doi.org/10.1038/s41597-025-04451-9>
- 233 Winston, R. B. (2019). *ModelMuse version 4 — a graphical user interface for MODFLOW*
234 *6* (Scientific Investigations Report No. 2019-5036). U.S. Geological Survey. <https://doi.org/10.3133/sir20195036>
235
- 236 Zill, F., Silbermann, C., Meisel, T., Magri, F., & Nagel, T. (2024). Far-field modelling of
237 THM processes in rock salt formations. *Open Geomechanics*, 5, 1–16. <https://doi.org/10.5802/ogeo.20>
238