The Journal of Open Source Software

OGSTools: A Python package for OpenGeoSys

Tobias Meisel [1,4], **Florian Zill** [2,1*], **Julian Heinze** [1*], **Lars Bilke** [1*], **Max Jäschke** [3,1,4*], **Feliks Kiszkurno** [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: 10.21105/joss.10973

Software

- Review
- Repository
- Archive

Editor: Andrew Walker

Reviewers:

- @daavid00
- @jlarrahondo

Submitted: 13 February 2026
Published: 29 September 2026

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 et al., 2026; Kolditz et al., 2012). `OGSTools` (Meisel et al., 2026) provides an interface between OGS-specific data and well-established data structures of the Python ecosystem, as well as domain-specific solutions and 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 summarises these capabilities graphically.

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 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.

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 well suited for 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 visualise the outcomes. OGSTools reduces the boilerplate and keeps notebooks focused on the learning objective.

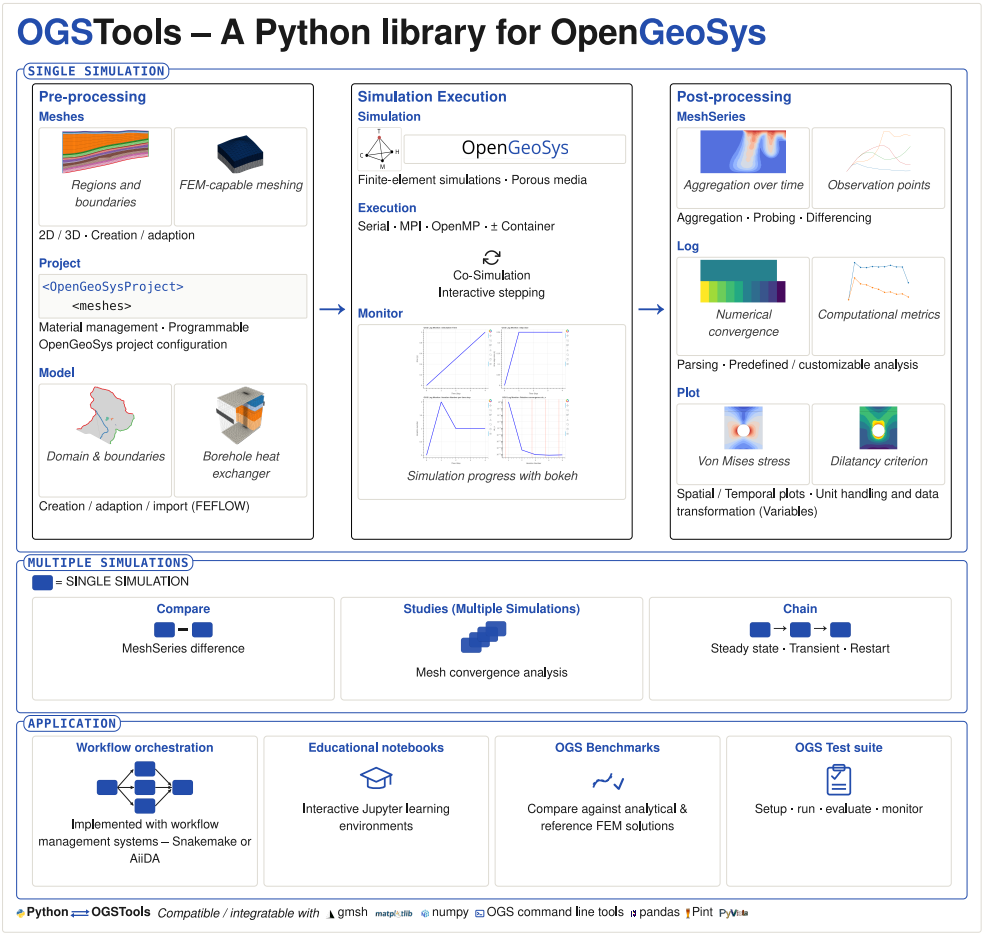

Figure 1: OGSTools graphical abstract: pre-processing, simulation execution, and post-processing for a single simulation; combining multiple simulations; and application areas.

State of the field

Simulator-specific companion libraries have emerged as a recurring pattern across scientific computing domains. These software packages connect a domain-specific simulator with the data structures and tooling of a general-purpose programming ecosystem (e.g., Python), typically covering pre-processing, execution, and post-processing on a single programmatic platform. OGSTools follows this pattern, in this case with OGS as the simulator.

In computational geoscience, several simulators have companion libraries of this kind. FloPy (Hughes et al., 2024) wraps the MODFLOW family of groundwater flow and transport models, supporting model creation, execution, and result analysis. pyGSFLOW (Larsen et al., 2022) provides equivalent functionality for the GSFLOW integrated hydrologic model, and toughio (Luu, 2020) covers pre- and post-processing for the TOUGH simulator family. Outside this group, DOLFINx (Baratta et al., 2023) takes a different approach: it exposes FEM assembly

and solving directly through a Python API rather than wrapping an external solver.

An alternative to these scripting-based libraries is a dedicated companion GUI — as with ModelMuse (Winston, 2019) for MODFLOW and the Data Explorer (Rink et al., 2012) for OpenGeoSys.

Build vs. contribute

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

Previously, without any centralisation to contribute OGS-specific pre- and postprocessing code, the code base for Python-related tasks in OGS was fragmented, with components often developed for specific use cases and varying degrees of standardisation, quality and maintenance efforts. Further, OGSTools 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 centralised approach, preceding work on `msh2vtu` (Kern & Bilke, 2022), `ogs6py` and `VTUinterface` (Buchwald et al., 2021) and further not yet published functionalities have been adapted and integrated into `OGSTools`.

Software Design

Design choices

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

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

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

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

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

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

Example

The following example shows a complete `0GS Liquid Flow` simulation workflow, adapted to 2D from an OGS benchmark. First, an OGS-capable mesh is generated and pressure boundary conditions are assigned to the boundary meshes (Figure 2), using standard `PyVista` (Sullivan & Kaszynski, 2019) functionality. After execution of the simulation, convergence metrics (Figure 3) and the final pressure distribution (Figure 4) are visualised. An annotated version of

this example is available in the OGSTools documentation. The example is deliberately kept minimal to keep the code listing short. OGSTools handles considerably more elaborate examples (e.g., complex geometries or coupled physical processes), as shown by the OGS benchmarks for the GREAT cell benchmark suite, excavation under two-phase flow, and Kirsch's problem.

```python
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()

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

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

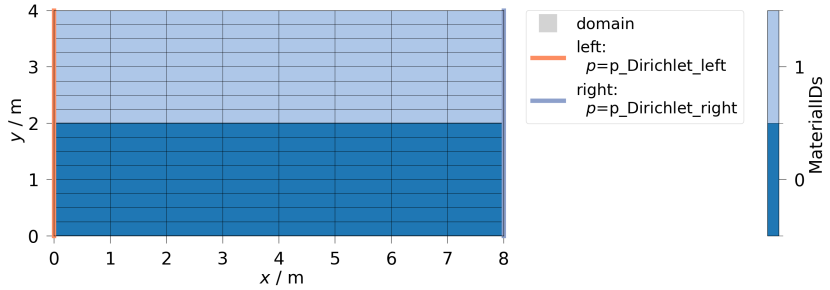

Figure 2: Initial boundary conditions.

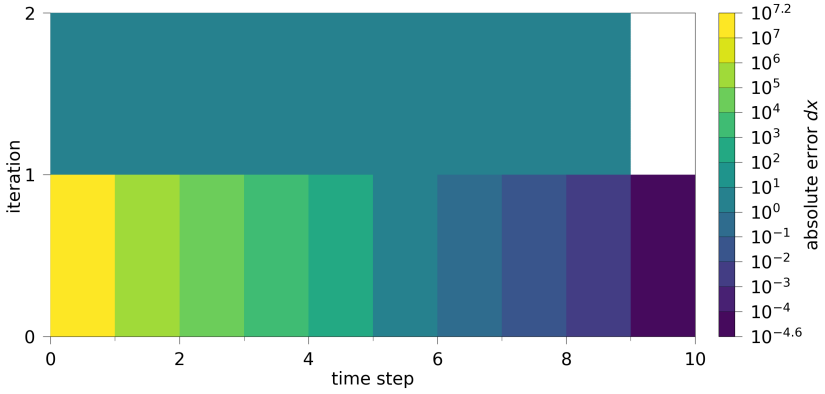

Figure 3: Resulting convergence metrics.

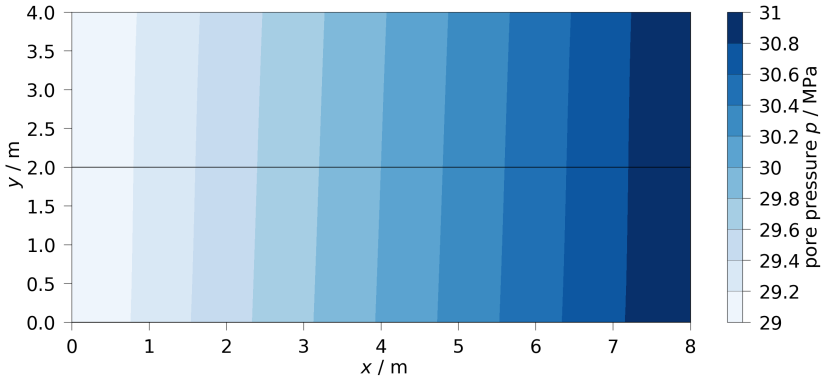

Figure 4: Resulting pressure distribution.

Features

The implemented features cover pre-processing, setup and execution of simulations, and 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 documentation. Containers are provided for reproducibility, benefiting both developers and users (Bilke et al., 2025). Like `OpenGeoSys`, `OGSTools` is available via `PyPI` and `conda-forge`.

Research impact

Workflows

OGSTools emerged from and is used in several 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., 2024) 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 for 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.

Acknowledgements

This work has been supported by multiple funding sources, including `SUTOGS` (Streamlining Usability and Testing of OpenGeoSys) under grant 528785032 by `Deutsche Forschungsgemeinschaft (DFG)`, `OpenWorkFlow` under grant STAFuE-21-05-Klei by `Bundesgesellschaft für Endlagerung (BGE)`, `AREHS` under grant 4719F10402 by `Bundesamt für die Sicherheit der nuklearen Entsorgung (BASE)`, and `DigBen` under grant 03G0927A by `Bundesministerium für Forschung, Technologie und Raumfahrt (BMFTR)`.

AI usage disclosure

In preparing this manuscript, the authors used Anthropic's Claude and OpenAI's ChatGPT, predominantly for grammar and spelling corrections, which the authors reviewed. For contributions to the software itself, the use of LLMs is permitted. Contributors are currently not required to disclose it; all contributions instead pass through a review process by the developers, maintainers, and authors.

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

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., Berger, F., Meisel, T., Lehmann, C., Buchwald, J., Shao, H., Jäschke, M., Grabow, L., Zill, F., Dörnbrack, M., Mollaali, M., Kiszkurno, F. K., Kolditz, O., Keese, H., Bernt, M., Hasan, N., & Grunwald, N. (2026). *OpenGeoSys* (Version 6.5.8). Zenodo. https://doi.org/10.5281/zenodo.20269146

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

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

Larsen, J. D., Alzraiee, A., & Niswonger, R. G. (2022). Integrated hydrologic model development and postprocessing for GSFLOW using pyGSFLOW. *Journal of Open Source Software*, *7*(72), 3852. https://doi.org/10.21105/joss.03852

Lehmann, C., Bilke, L., Buchwald, J., Graebling, N., Grunwald, N., Heinze, J., Meisel, T., Lu, R., Naumov, D., Rink, K., Sen, O. Ö., Selzer, P., Shao, H., Wang, W., Zill, F., Nagel, T., & Kolditz, O. (2024). OpenWorkFlow – development of an open-source synthesis-platform for safety investigations in the site selection process. *Grundwasser*, *29*(1), 31–47. https://doi.org/10.1007/s00767-024-00566-9

Luu, K. (2020). toughio: Pre- and post-processing Python library for TOUGH. *Journal of Open Source Software*, *5*(51), 2412. https://doi.org/10.21105/joss.02412

Meisel, T., Zill, F., Heinze, J., Bilke, L., Jäschke, M., Kiszkurno, F. K., Grunwald, N., Fischer,

T., & Buchwald, J. (2026). *OGSTools* (Version 0.8.2). Zenodo. https://doi.org/10.5281/zenodo.22274697

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

Rink, K., Kalbacher, T., & Kolditz, O. (2012). Visual data exploration for hydrological analysis. *Environmental Earth Sciences*, *65*(5), 1395–1403. https://doi.org/10.1007/s12665-011-1230-6

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

Sullivan, C., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, *4*(37), 1450. https://doi.org/10.21105/joss.01450

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

Winston, R. B. (2019). *ModelMuse version 4 — a graphical user interface for MODFLOW 6* (Scientific Investigations Report No. 2019-5036). U.S. Geological Survey. https://doi.org/10.3133/sir20195036

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