




ogs6py and VTUinterface: streamlining OpenGeoSys workflows in Python

Jörg Buchwald^{*1, 2}, Olaf Kolditz^{1, 3, 4}, and Thomas Nagel^{2, 4}

1 Helmholtz Center for Environmental Research - UFZ, Leipzig, Germany **2** Technische Universität Bergakademie Freiberg, Germany **3** Technische Universität Dresden, Germany **4** TUBAF-UFZ Center for Environmental Geosciences, Germany

DOI: [10.21105/joss.03673](https://doi.org/10.21105/joss.03673)

Software

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

Editor: [Leonardo Uieda](#) 

Reviewers:

- [@cpgr](#)
- [@TobbeTripitaka](#)
- [@akaszynski](#)

Submitted: 31 May 2021

Published: 21 November 2021

License

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

Summary

We introduce two new Python modules that facilitate the pre- and post-processing of finite element calculations. [ogs6py](#) is a Python interface for the open-source package OpenGeoSys ([Bilke et al., 2019](#)), a finite element code for simulation of multi-field processes in fractured porous media. Modeling workflows can be further streamlined in Jupyter Notebooks ([Kluyver et al., 2016](#)) using the newly developed [VTUinterface](#). The use of the modules is demonstrated with common workflow operations, including parameter variations, setting boundary conditions, changing solver settings, verification of simulation results by comparison to analytical solutions, set-up and evaluation of ensemble runs, and convenient analysis of results by line plots, time series, or transient contour plots.

Statement of need

Python has become a widely used framework for scientific data analysis and modeling. The development is driven by ease of use and flexibility, the vast modular ecosystem including powerful plotting libraries, and the Jupyter Notebook technology. The attractiveness of Python is not limited to post-processing; pre-processing tasks can be simply conducted, using packages such as the Python wrapper for GMSH ([Geuzaine & Remacle, 2009](#)) or the tool meshio ([Schlömer et al., 2021](#)). While many existing open-source tools force the user to learn a new syntax for interacting with the software, Python bindings allow control in a general language and thus are more accessible for a wider community of users.

In this contribution, we address interaction with the open-source code OpenGeoSys (OGS) ([Bilke et al., 2019](#)) version 6, aiming to facilitate both pre- and post-processing workflows with Python. This aim was partly inspired by the desire to design, control and evaluate ensemble runs ([Buchwald et al., 2020](#); [Chaudhry et al., 2021](#)) but has now taken on a wider perspective for general usability. A similar Python interface “ogs5py” exists for OGS version 5 ([Müller et al., 2021](#)); however, conceptual differences between the versions, for example, the use of XML input files, required an entirely new package to be built from scratch.

The standard output format of OpenGeoSys is VTK ([W. Schroeder et al., 2006](#)) unstructured grid files (VTU) as time slices stacked together by a PVD file. These can be analyzed using Paraview ([Ahrens et al., 2005](#)), a Python wrapper for VTK ([W. J. Schroeder et al., 2000](#)), or visualization tools like PyVista ([Sullivan & Kaszynski, 2019](#)) or Mayavi ([Ramachandran & Varoquaux, 2011](#)). However, a finite-element-modeller’s *bread and butter* business often include extracting single- or multiple point time-series data. The direct use of the VTK library

^{*}corresponding author

is quite cumbersome for such tasks, especially when interpolation is required. The mentioned Python packages focus on visualization aspects, and except for Paraview, to our knowledge, do not have file support for PVD files or time-series data (Aboufirass, 2020; Sullivan, 2019).

Features

ogs6py allows creating complete OGS configuration files from scratch, altering existing files, running simulations and parsing OGS log files. The following example demonstrates some basic functionalities. The complete example demonstrating a typical ogs6py/VTUinterface workflow on a coupled thermo-hydro-mechanical (THM) problem of a tunnel excavation followed by the emplacement of a heat-emitting canister can be found in a [Jupyter notebook](#) located in the project repository.

An instance of OGS is created, an existing project file is imported, and an output file is specified:

```
model = OGS(INPUT_FILE="tunnel.prj", PROJECT_FILE="tunnel_exc.prj")
```

A project file can be altered by commands for adding blocks, removing or replacing parameters:

```
model.replace_phase_property(mediumid=0, phase="Solid",  
                             name="thermal_expansivity", value=a_s)
```

or

```
model.replace_text("tunnel_exc", xpath="./time_loop/output/prefix")
```

The project file can be written to disk:

```
model.write_input()
```

and OGS can be executed by calling the `run_model()` method:

```
model.run_model(path="~/github/ogs/build_mkl/bin",  
                logfile="excavation.log")
```

OGS produces PVD and VTU files that can be handled with VTUinterface:

```
pvdfile = vtuiO.PVDIO("tunnel_exc.pvd", dim=2)
```

One of the most powerful features of VTUinterface is the ability to deal with PVD files as time-series data. For example, the following command reads in the VTU point field “pressure” at point “pt0,” defined in a dictionary, using nearest neighbour interpolation.

```
excavation_curve = pvdfile.read_time_series("pressure",  
                                             interpolation_method="nearest", pts={"pt0": (0.0,0.0,0)})
```

The result can directly be plotted using matplotlib ([Figure 1](#)). The time axis can be retrieved from the PVD file as well.

```
plt.plot(pvdfile.timesteps, excavation_curve["pt0"] / 1e6)
plt.xlabel("$t$ / d")
plt.ylabel("$p$ / MPa");
```

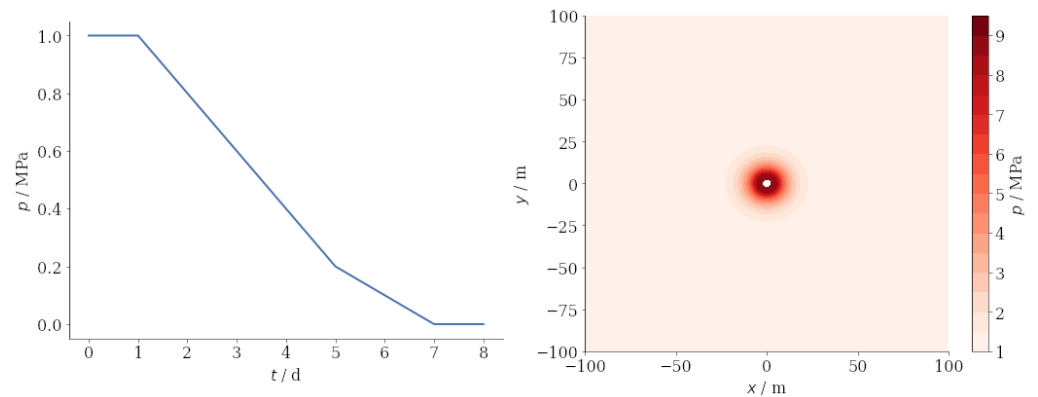


Figure 1: Plots demonstrating the usage of VTUinterface: Deconfinement curve extracted as time series from a PVD file of excavation simulation (left). Contour plot of pressure distribution generated with VTUinterface and matplotlibs `tricontourf()` shows thermal pressurization during the heating phase (right).

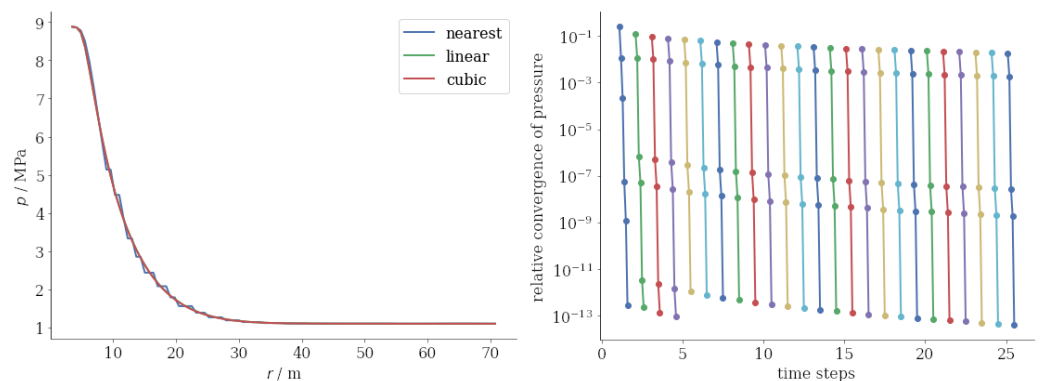


Figure 2: Spatial pressure distribution generated with VTUinterface from a linear point set array using three different grid interpolation methods (left). Relative convergence plot showing the numerical behaviour over ten time steps extracted using the log file parser of ogs6py (right).

This brief overview shows only some of the functionalities coming with ogs6py and VTUinterface. Further developments will focus on extending functionalities focusing on built-in checks to ensure that only valid input files are generated.

Technical Details

ogs6py requires python 3.8 or above and uses `lxml` (Behnel et al., 2005) to process OGS6 input files and uses the subprocess module to run OGS. Furthermore, `pandas` (McKinney & others, 2010) is required for holding OGS log file data. VTUinterface requires python 3.8 or above and uses the python wrapper for `VTK` to access VTU files and `lxml` for PVD files. In addition to VTK's own interpolation functionalities, we use `pandas` and `SciPy` (Virtanen et al., 2020) for interpolation.

Applications

Both of the packages introduced are relatively new, being only 1 to 2 years old. However, the adoption process in the OpenGeoSys community is gearing up. For example, a [YouTube video](#) was published explaining their use; both tools are also used for teaching at the TU Bergakademie Freiberg and they were also extensively utilized in two recent peer-reviewed publications ([Buchwald et al., 2020](#), [2021](#)).

Acknowledgements

We acknowledge contributions from Tom Fischer, Dmitry Yu. Naumov, Dominik Kern and Sebastian Müller during the genesis of this project. The funding through the iCROSS-Project (Integrity of nuclear waste repository systems – Cross-scale system understanding and analysis) by the Federal Ministry of Research and Education (BMBF, grant number 02NUK053E) and Helmholtz Association (Helmholtz-Gemeinschaft e.V.) through the Impulse and Networking Funds (grant number SO-093) is greatly acknowledged. This work was in part funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number NA1528/2-1.

References

- Aboufirass, A. (2020). PyVista issue 294: Time series data support in pyvista. In *GitHub repository issue*. GitHub. <https://github.com/pyvista/pyvista-support/issues/294>
- Ahrens, J., Geveci, B., & Law, C. (2005). Paraview: An end-user tool for large data visualization. *The Visualization Handbook*, 717(8). <https://doi.org/10.1016/b978-012387582-2/50038-1>
- Behnel, S., Faassen, M., & Bicking, I. (2005). *Lxml: XML and HTML with python*. Lxml.
- 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>
- Buchwald, J., Chaudhry, A. A., Yoshioka, K., Kolditz, O., Attinger, S., & Nagel, T. (2020). DoE-based history matching for probabilistic uncertainty quantification of thermo-hydro-mechanical processes around heat sources in clay rocks. *International Journal of Rock Mechanics and Mining Sciences*, 134(May), 104481. <https://doi.org/10.1016/j.ijrmms.2020.104481>
- Buchwald, J., Kaiser, S., Kolditz, O., & Nagel, T. (2021). Improved predictions of thermal fluid pressurization in hydro-thermal models based on consistent incorporation of thermo-mechanical effects in anisotropic porous media. *International Journal of Heat and Mass Transfer*, 172, 121127. <https://doi.org/10.1016/j.ijheatmasstransfer.2021.121127>
- Chaudhry, A. A., Buchwald, J., & Nagel, T. (2021). Local and global spatio-temporal sensitivity analysis of thermal consolidation around a point heat source. *International Journal of Rock Mechanics and Mining Sciences*, 139(June 2020), 104662. <https://doi.org/10.1016/j.ijrmms.2021.104662>
- Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11), 1309–1331. <https://doi.org/10.1002/nme.2579>

- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., & others. (2016). *Jupyter notebooks-a publishing format for reproducible computational workflows*. (Vol. 2016). <https://doi.org/10.3233/978-1-61499-649-1-87>
- McKinney, W., & others. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, 445, 51–56. <https://doi.org/10.25080/Majora-92bf1922-00a>
- Müller, S., Zech, A., & Heße, F. (2021). ogs5py: A python-API for the OpenGeoSys 5 scientific modeling package. *Groundwater*, 59(1), 117–122.
- Ramachandran, P., & Varoquaux, G. (2011). Mayavi: 3D visualization of scientific data. *Computing in Science & Engineering*, 13(2), 40–51. <https://doi.org/10.1109/mcse.2011.35>
- Schlömer, N., McBain, G. D., Luu, K., Tsolakis, C., Li, T., Keilegavlen, E., Ferrándiz, V. M., Barnes, C., Lukeš, V., Dalcin, L., Jansen, M., Wagner, N., Gupta, A., Müller, S., Woodsend, B., Krande, Schwarz, L., Blechta, J., Christovasilis, I. P., ... Cereijo, I. (2021). *Nschloe/meshio: none* (Version v0.1.5) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.4745399>
- Schroeder, W. J., Avila, L. S., & Hoffman, W. (2000). Visualizing with VTK: A tutorial. *IEEE Computer Graphics and Applications*, 20(5), 20–27. <https://doi.org/10.1109/38.865875>
- Schroeder, W., Martin, K., & Lorensen, B. (2006). *The Visualization Toolkit—An Object-Oriented Approach To 3D Graphics* (Fourth). Kitware, Inc.
- Sullivan, C. (2019). PyVista issue 414: Add .pvd reader. In *GitHub repository issue*. GitHub. <https://github.com/pyvista/pyvista/issues/414>
- 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>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>