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.03327

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Submitted: 31 May 2021
Published: 06 June 2021

Summary

ogs6py is a Python interface for the open-source package OpenGeoSys (Bilke et al., 2019), a finite element code for the simulation of multi-field processes in fractured porous media. In conjunction with VTUinterface it is possible to streamline modeling workflows in Jupyter Notebooks using Python. With this article, we wish to introduce two new Python modules that facilitate the pre- and post-processing of finite element calculations of OpenGeoSys and thus make this code more accessible to the community. Their use is demonstrated along workflows typically encountered by the modeler, including the variation of parameters, boundary conditions, or solver settings, the verification of simulation results by comparison to analytical solutions, the set-up and evaluation of ensemble runs, the rapid analysis of results by line plots, time series, or transient contour plots.

Statement of need

Driven by its ease-of-use and flexibility as an open-source dynamic language, its vast modular ecosystem, the development of powerful plotting libraries and the Jupyter Notebook technology, Python became a widely used common framework for scientific data analysis in the modeling community during the past decade. However, the attractiveness of Python is not just limited to post-processing. E.g., with the Python wrapper for GMSH (Geuzaine & Remacle, 2009) or the tool meshio (Schlömer et al., 2021) also pre-processing tasks can be easily conducted without leaving the IPython command prompt. The usability of any modeling package is therefore greatly enhanced if Python bindings are provided. In fact, while many open-source tools effectively forced the user to learn a singular syntax for interacting with the software, Python bindings allow control over such tools from within the Python world and thus open them up for a wider community of users.

Here, we are particularly addressing the open-source code OpenGeoSys (OGS) (Bilke et al., 2019) version 6. It is our aim to facilitate both pre- and post-processing workflows utilizing the Python ecosystem. This aim was not the least inspired by the desire to facilitate setting up, controlling and evaluating ensemble runs (Buchwald et al., 2020; Chaudhry et al., 2021) but has now taken on a wider perspective of general software usability. There exists already a similar Python interface "ogs5py" for OGS version 5 (Müller et al., 2021). However, the differences in many concepts, like the use of XML input files, required an entirely new package to be built from scratch.

As standard output format, OpenGeoSys uses VTK unstructured grid files (VTU) as time slices stacked together by a PVD file. These can be analyzed typically using Paraview (Ahrens

[*] corresponding author

et al., 2005). For interactive use the Python wrapper for VTK (Schroeder et al., 2000) and some other tools like PyVista (Sullivan & Kaszynski, 2019) or Mayavi (Ramachandran & Varoquaux, 2011) are available facilitating an easier access to the VTK library. While the direct use of the VTK library is quite cumbersome for basic tasks, like reading data for a given point set, especially when interpolation between grid points is also required. The latter packages focus mainly on 3D visualization. However, the *bread and butter* business of a finite-element-modeler often consists of the extraction of single- or multiple point time-series data. To our knowledge the named packages (with the exception of Paraview) don't have file support for PVDs or time series data, yet (awa5114, 2020; banesullivan, 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 common 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.

To read in an existing project file and to specify an output name an instance of OGS needs to be created.

```
model = OGS(INPUT_FILE="tunnel_ogs6py.prj", PROJECT_FILE="tunnel_exc.prj")
```

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

```
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 with

```
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 significant features of VTUinterface is the ability to deal with PVD files as time series data. E.g., the following command reads in the VTU point field "pressure" at point "pt0," defined in a dictionary, using nearest neighbor 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");
```

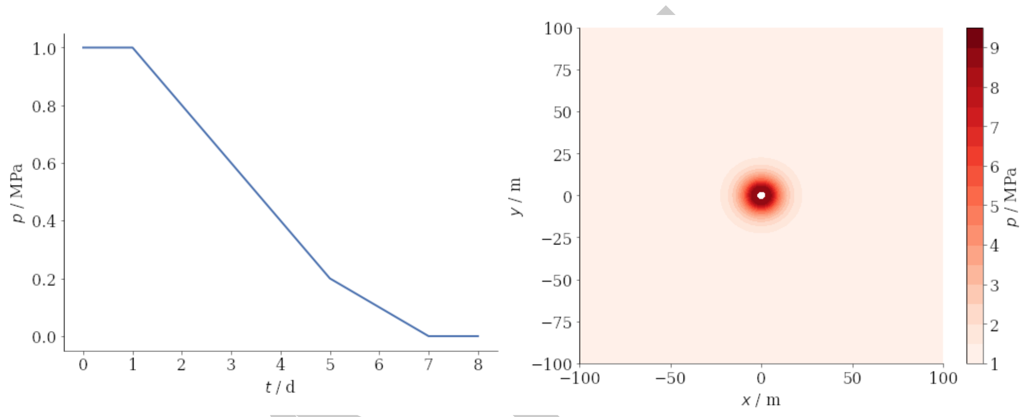

Figure 1: Plots demonstrating the usage of VTUinterface: Deconfinement curve extracted as time-series from a PVD file of excavation simulation (left). Contour plot of pressure distribution generated with VTUinterface and matplotlibs `tricontourf()` showing the thermal pressurization during the heating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

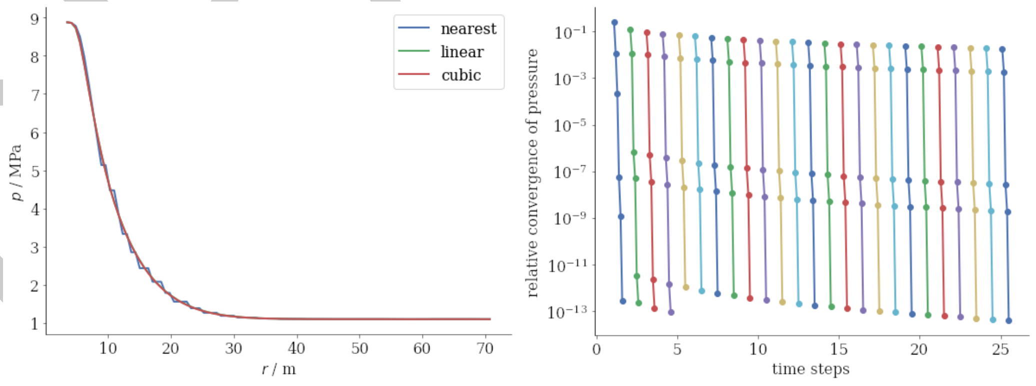

Figure 2: Spatial pressure distribution generated with VTUinterface from a linear point set array using three different grid interpolation methods (left). Relative convergence plot showing the numerical behaviour over ten time steps extracted using the log file parser of ogs6py (right).

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

Applications

Both introduced packages are with 1-2 years of age relatively new. However, the adoption process in the OpenGeoSys community is gearing up. E.g., 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

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

awa5114. (2020). Pyvista issue: Time series data support in pyvista. In *GitHub repository issue*. GitHub. https://github.com/pyvista/pyvista-support/issues/294

banesullivan. (2019). Pyvista issue: Add .pvd reader. In *GitHub repository issue*. GitHub. https://github.com/pyvista/pyvista/issues/414

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

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

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