Groundwater

Methods Note/

ogs5py: A Python-API for the OpenGeoSys 5 Scientific Modeling Package

by Sebastian Müller[1], Alraune Zech[2], and Falk Heße[3]

Abstract

High-performance numerical codes are an indispensable tool for hydrogeologists when modeling subsurface flow and transport systems. But as they are written in compiled languages, like C/C++ or Fortran, established software packages are rarely user-friendly, limiting a wider adoption of such tools. OpenGeoSys (OGS), an open-source, finite-element solver for thermo-hydro-mechanical–chemical processes in porous and fractured media, is no exception. Graphical user interfaces may increase usability, but do so at a dramatic reduction of flexibility and are difficult or impossible to integrate into a larger workflow. Python offers an optimal trade-off between these goals by providing a highly flexible, yet comparatively user-friendly environment for software applications. Hence, we introduce ogs5py, a Python-API for the OpenGeoSys 5 scientific modeling package. It provides a fully Python-based representation of an OGS project, a large array of convenience functions for users to interact with OGS and connects OGS to the scientific and computational environment of Python.

Introduction

The scientific-computing project OpenGeoSys (OGS) provides methods for the simulation of subsurface processes like groundwater flow, soil moisture dynamics, transport of contaminants, and energy extraction (Kolditz et al. 2012). It features state-of-the-art finite-element solvers with an open-source approach. Yet potential users without a strong background in computer science may find themselves confronted with a steep learning curve. While providing a Graphical User Interface may lower the threshold for some and therefore widen the potential user base, it would significantly reduce flexibility in the development process, require the allocation of manpower needed elsewhere and make OGS far less platform-independent.

In order to strike a balance between usability, flexibility, effort and efficiency, the use of a Python-based interface for OGS has emerged as the best choice. There are a number of reasons that make Python such a dominant candidate for this purpose. It provides a powerful environment for scientific and engineering applications. It is easy to learn, easy to code and easy to communicate. It has strong momentum in the highly competitive market of programming languages. It is a perfect glue language, meaning that complex workflows can be created and maintained. Finally, it is open source with a huge user community and community-supported infrastructure.

These reasons combine to make Python an almost uniquely suited tool for scientific and engineering applications due to its mediating position between

[1]Department of Computational Hydrosystems, Helmholtz Centre for Environmental Research—UFZ, Permoserstr. 15, 04318 Leipzig, Germany; sebastian.mueller@ufz.de

[2]Department of Earth Sciences, Utrecht University, Princetonlaan 8a, 3584CB Utrecht, The Netherlands; a.zech@uu.nl

[3]Corresponding author: Institute of Earth and Environmental Sciences, University of Potsdam, Karl-Liebknecht-Str. 24–25, 14476 Potsdam, Germany; falkhesse@uni-potsdam.de [Helmholtz-Zentrum für Umweltforschung GmbH - UFZ]

Article impact statement: This article introduces ogs5py a Python package to set up, control, and interact with the OGS 5 modeling project.

Correction added on 26 October 2020, after first online publication: [Helmholtz-Zentrum für Umweltforschung GmbH - UFZ] was added for [Falk Heße] where 26 October 2020 is the date the correction was made

Received August 2019, accepted May 2020.

doi: 10.1111/gwat.13017

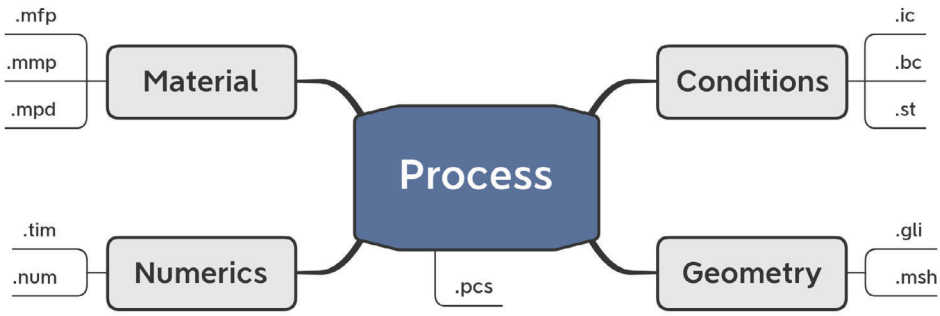

Figure 1. Schematic of the different input files associated with an OGS project.

compiled languages, like C/C++, Fortran, and Java, on the one hand and tightly integrated software suites with a GUI on the other (Perez et al. 2011). The most recent ranking from IEEE Spectrum substantiates this notion, calling Python the best programming language overall (Cass 2019), as do the increasing number of scientific and engineering software tools that rely on Python, rather than a fully fledged GUI, to achieve a better user experience (O'Boyle et al. 2008; Chaudhury et al. 2010; Dalcin et al. 2011; Cornelis et al. 2012; Müller et al. 2012; Zambelli et al. 2013; Roest et al. 2014; Bakker et al. 2016; Schlömer 2018).

Therefore we introduce `ogs5py`, a Python package that provides a Python representation of the full modeling process of OGS 5. Here, a full modeling process means the preparation of the OGS input files, initiating and inspecting the run of an OGS model and, finally, the import and visualization of the results. In addition, `ogs5py` can import already existing OGS 5 applications and turn them into a fully Python-based workflow. Many OGS 5 applications are provided in the related `ogs5py_bechmarks` project, which we will also detail below. We chose OGS 5 as the target platform, since the newest version OGS 6, although already released, is still under development and therefore in flux. Using the latest version of OGS 5 guarantees a stable and mature numerical engine for any modeling effort. As examples, we cite such modeling efforts as the simulation of a groundwater system (Jing et al. 2018), its use as a data-generating forward model in a Bayesian inference (Savoy et al. 2017) or as part of a decision-making tool for water management services. There are `ogs5py` scripts for nearly all benchmarks of OGS 5 as well as a number of convenience functions in particular for working with mesh generation, inspection and manipulation.

OpenGeoSys 5

OpenGeoSys 5 is a scientific computing project, comprising numerical methods for the simulation of thermo-hydro-mechanical-chemical processes in porous and fractured media. OGS has found application in diverse fields like the modeling of groundwater contaminants, water resources, geothermal energy systems and energy storage. At its core, OGS 5 uses the finit-element method to solve the partial differential equations that govern flow and transport processes in the subsurface.

The first versions of OGS were known as RockFlow. They were created in the late 1980ies and early 1990ies, together with the early development of FEFLOW (Trefry and Muffels 2007). Both tools were initially written in FORTRAN but later development switched to C and eventually C++ as the main language. For a detailed overview of the development of OGS, we refer to Kolditz et al. (2012).

An OGS 5 project is specified by a number of ASCII files, which can be grouped according to their function (see Figure 1). At the center is the process file `.pcs`. It contains the processes being simulated and consequently determines the equation(s) that need(s) specification. Additional information is provided by the other input files, depending on the type of problem to be solved. Initial conditions, boundary conditions, and external source terms are specified in the `.ic`, `.bc`, and `.st` files, respectively. Properties are defined in the `.mfp`, `.mcp`, and `.mmp` files for the fluid, component, and material properties, respectively. The `.num` and `.tim` files contain specifications for the numerical solvers as well as time stepping in case of transient problems. The numerical mesh is given in the `.msh` file, whereas the `.gli` file serves to define spatial positions, for example, for the sinks and sources from the `.st` file.

OGS 5 provides a number of tangible benefits for potential users. Compared to FEFLOW (Trefry and Muffels 2007), OGS 5 is fully open-source. As regards MODFLOW (Hughes et al. 2017), OGS 5 is a finite-element solver and can handle complex geometries naturally by irregular Finite Element Method meshes. Although the newest version MODFLOW6 has mitigated some on this limitation, its meshes do still fall short when compared with the full flexibility of a FEM mesh. Such meshes, built using the Delaunay triangulation, use triangles or tetrahedra of appropriate sizes that can conform to the often irregular geometry of geological structures. In addition, the FEM is a general paradigm for partial-differential equations and OGS 5 can consequently solve a series of subsurface problems, like saturated and unsaturated flow, solute transport and heat transport; all using the same numerical paradigm.

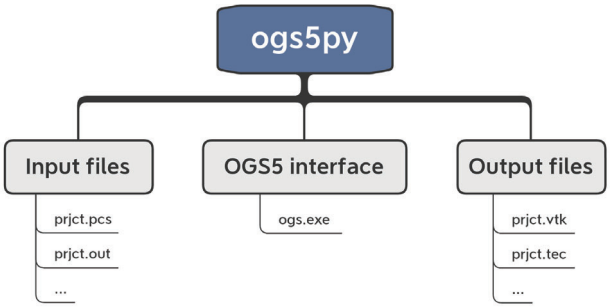

Figure 2. Schematic of the different steps of running an OGS 5 project and how they are controlled by `ogs5py`.

The `ogs5py` Package

Structure

Figure 2 shows how `ogs5py` runs an OGS 5 project in a three-step process: writing the input files, passing them on to the OGS 5 executable, and importing the resulting output files.

First, the structure of the OGS 5 project is represented by creating an OGS class. Its attributes correspond to the OGS 5 file structure as outlined in Figure 1. Then, the corresponding input files are written into the project folder and the project can be run. In a third step, the user can import the results into `ogs5py` and use Python functions for visualization and post-processing.

Users familiar with the `FloPy` package, which provides a similar Python-based interface for MODFLOW (Bakker et al. 2016), will recognize the similarities in this interaction. This is due to the shared design goals, i.e., providing a Python interface for a numerical modeling suite, as well as the fact that both modeling suites use ASCII-based input files. We provide and discuss two example scripts in the Supporting Information to exemplify `ogs5py` and familiarize potential users with it.

Installation

`ogs5py` is compatible with Python versions 2.7 and 3.4 or higher. Since Python 2 is on its way out, we strongly encourage the use of Python 3. The source code is maintained under a GitHub account to optimize team efforts. Furthermore, we allow users to raise issues or improving the code by making pull requests with the aid of GitHub's infrastructure. It can be found under https://github.com/GeoStat-Framework/ogs5py.

The package can be installed via the command line on Linux, MacOS and Windows using `conda` or `pip`. The latest version is installed by typing one of the commands:

```
pip install ogs5py
# or
conda install ogs5py
```

While `ogs5py` can run on any platform given the correct Python version and libraries are installed, the same is not necessarily true for OGS 5. `ogs5py` provides tools to automatically download a platform-specific version of the OGS executable for many popular Linux and Windows systems. If you have a Macintosh computer, an unusual CPU or do not want to use the prebuild binary; the source code of the newest version of OGS 5 can be downloaded from the website and compiled locally. Instructions can be found on the GitHub page of the project https://github.com/ufz/ogs5.

To obtain an OGS5 executable, you can run the following script:

```
from ogs5py import download_ogs
download_ogs ()
```

This command will store a system-dependent executable in the `ogs5py` config path and will be called automatically when a model is run. One can pass a version statement to the download routine (e.g. "5.7" [default] or "5.8"). Also "latest" and "stable" versions can be downloaded from the OGS 5 Jenkins CI system. OGS "5.7" provides executables for Windows, Linux, and MacOS. For "5.8," "latest," and "stable" there are no MacOS prebuilds.

Resources

Self-compiled versions of OGS 5 can be added to the `ogs5py` config path with the following script:

```
from ogs5py import add_exe
add_exe ("path / to / your / ogs / exe")
```

The source code is formatted by the "Black" code formatter. This tool guarantees a code style that is transparent and unified, thus making it easier for users to understand, review and contribute to the code. The documentation of `ogs5py`, as well as tutorials explaining the features, can be found at https://geostat-framework.readthedocs.io/projects/ogs5py.

Continuous Integration is established through Travis-CI. Python wheels are prebuilt to make installation as easy as possible. Also, unit tests run Travis-CI for each commit to the GitHub repository and code coverage is determined. Every new release on GitHub is directly deployed to the Python package index and gets a DOI by Zenodo. Version 1.0, discussed here, can be found under Müller (2019).

Licensing

We selected the MIT license for `ogs5py`, due to it being a well established and very permissive license. Since the benchmarks found in the `ogs5py_bechmarks` project are derived from OGS, we use its license. This license is unique to OGS but shows strong similarities to the LGPL.

Example Applications for `ogs5py`

In general, an `ogs5py` script is a succession of block-wise definitions of the different OGS5 input files. A generic model script looks as follows.

```
from ogs5py import OGS
# create the base class
```

```
model = OGS (task_root = "project",
task_id = "model")
# define the model settings
model.pcs.add_block ( # set the pro-
cess type
   PCS_TYPE = "GROUNDWATER_FLOW",
   NUM_TYPE= "NEW",
)
model.msh. generate (...) # define mesh
model.gli.generate (...) # define geom-
etry
model.mmp.add_block (...) # medium prop-
erties
model.bc.add_block (...) # bound-
ary condition (if any)
model.st.add_block (...) # source
term (if any)
model.ic.add_block (...) # ini-
tial condition (transient)
model.tim.add_block (...) # timestep-
ping (transient)
model.out.add_block (...) # out-
put settings
model.num.add_block (...) # numeri-
cal solver settings
# list of other inputs, set-
tings and interfaces
...
# write input, run OGS5, read output
model.write_input ()
success = model.run_model ()
output = model.readvtk ()
```

Within an `add_block` call, the subkeywords of the associated OGS5 input file can be addressed and provided with information.

We exemplify the workflow and application of `ogs5py` by virtue of two examples. They are part of the `ogs5py` package itself and can be found at https://github.com/GeoStat-Framework/ogs5py/tree/master/examples/. They also serve as tutorials for the `ogs5py` documentation.

The first example is a simple two-dimensional numerical model of a pumping test. It allows potential users to become familiar with the general workflow and introduces the main features of the package. The second example provides a three-dimensional numerical model of a pumping test in a heterogeneous porous medium. Heterogeneities are generated by another Python package called `GSTools`. This example shows the abilities of Python to connect different tools into a larger workflow. Both examples are presented and discussed in detail in the Supporting Information.

Additionally, we provide a large number of OGS 5 benchmarks transformed into their equivalent `ogs5py` projects. They can be found under https://github.com/GeoStat-Framework/ogs5py_benchmarks. Users can obtain them using `git` and clone them directly from the GitHub repository `git clone https://github.com/GeoStat-Framework/ogs5py_benchmarks.git`

These benchmarks cover a wide range of applications. Examples are saturated and unsaturated flow, sorption using Freundlich and Langmuir isotherms, solute transport, density-dependent flow, reactive transport, multiphase flow, ion exchange and many more. In total, several hundreds of benchmarks can be found. Depending on the specific needs of a practitioner, they can serve as a starting point for building models for specific case studies.

Additional Advantages of `ogs5py`

In addition to the general advantages of a Python environment, `ogs5py` offers a number of specific benefits for this project.

Mesh Handling

Discretizing the spatial domain to a numerical mesh is usually the first step and thus of great importance. Accordingly, `ogs5py` provides a range of internal methods to generate, manipulate and import meshes. It also provides close integration with the `pygmsh` package (Schlömer 2018), which is a powerful Python tool for mesh generation. For example, the user can quickly generate a grid adapter, to refine a rectangular grid in a given domain by:

```
from ogs5py import MSH

mesh = MSH ()
mesh.generate (
  " grid_adapter2D", # adapter
  generator
  out_dim=(100.0, 100.0), # outer
  dimension
  in_dim=(50.0, 50.0), # inner
  dimension
  out_res=(10.0, 10.0), # outer
  x-y resolution
  in_res=(1.0, 1.0), # inner
  x-y resolution
  out_pos=(0.0, 0.0), # outer
  block position
  in_pos =(25.0, 25.0), # inner
  block position
  out_mat=1, # outer
  material ID
  in_mat=0, # inner
  material ID
  fill=True, # fill the inner
block
)
mesh.show (show_material_id=True)
```

Here, a transition zone is created with `gmsh` by triangulation to adapt the grid resolutions. The adapter was filled with a rectangular mesh. Furthermore, different material IDs for both mesh parts were set. The resulting

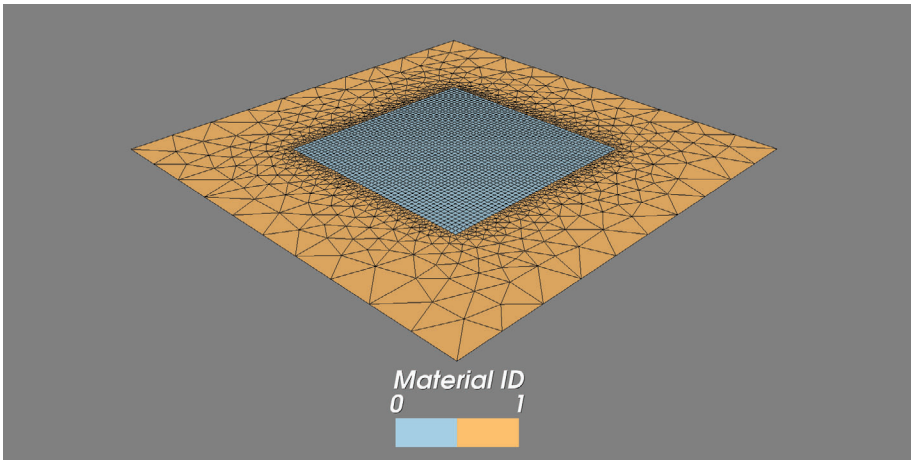

Figure 3. A generated adapter to locally refine a rectangular grid from a coarse to a fine resolution.

mesh in Figure 3 shows the adaptive mesh behavior. Other examples utilizing `pygmsh` can be found in the package documentation.

High-Performance Computing

Flow and transport simulations are based on partial-differential equations and thus computationally demanding. Realistic models can have in excess of 10^6 mesh cells, which can quickly exceed the limits of regular desktop computers. Uncertainty assessment demands the simulation of a large number of scenarios, dramatically increasing the computational costs.

`ogs5py` meets this challenge of computational efficiency by using OGS 5 for the high-performance computing and Python for user interaction. Python supports the use of parallelization techniques, e.g. for uncertainty assessment studies, as well as the use of supercomputers for the simulation of large numerical models.

Import and Conversion of Existing OGS Projects

The associated `ogs5py_benchmarks` project does not only provide the OGS benchmarks but also has the ability to convert most existing OGS 5 projects into their `ogs5py` counterparts. To do so, the `OGS` class provides `gen_script`, that creates `ogs5py` python script for the specified model, which is loaded first:

```
from ogs5py import OGS
model = OGS (task_root = " root ",
task_id = " model ")
model.load_model ("path / to / your /
old / model")
model.gen_script ()
```

Through `gen_script`, users can easily convert their existing projects with minimal effort to benefit from the abilities of `ogs5py`.

Conclusions

The `ogs5py` package provides a Python-based interface for the OGS 5 software suit. In both design and functionality, `ogs5py` is similar to the popular `FloPy` suite, which provides a Python-based interface for the MODFLOW software package. The salient features of `ogs5py` are determined by OGS 5. OGS 5 provides a large collection of open-source, numerical tools for solving subsurface problems and these functionalities are the main assets of `ogs5py`. Compared to MODLFOW, OGS 5 provides a number of well-tested FEM solvers. This enables the use of flexible grids as well as the joint modeling of coupled subsurface processes, like flow and transport or reactive transport systems.

The `ogs5py` package also provides a close integration with the `pygmsh` package. This allows for a seamless interaction with its meshing abilities. Finally, the `ogs5py_benchmarks` package, developed together with `ogs5py` provides a number of OGS benchmarks, as well as the ability to convert existing OGS 5 projects into their Python counterparts.

Acknowledgments

For this study, Falk Heße was financially supported by Deutsche Forschungsgemeinschaft via grant number: HE 7028/2-1. Sebastian Müller was funded by Deutsche Forschungsgemeinschaft via grant number HE 7028/2-1 as well as by the German Federal Environmental Foundation. We also thank two anonymous reviewers for their comments and feedback.

Authors' Note

The authors do not have any conflicts of interest or financial disclosures to report.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of

the article. Supporting Information is generally *not* peer reviewed.

Appendix S1. Supporting information

References

Bakker, M., V. Post, C.D. Langevin, J.D. Hughes, J.T. White, J.J. Starn, and M.N. Fienen. 2016. Scripting modflow model development using python and flopy. *Groundwater* 54, no. 5: 733–739.

Cass, S. 2019. The Top Programming Languages 2019. Python remains the big kahuna, but specialist languages hold their own. *IEEE Spectrum*. https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019 (accessed March 5, 2020).

Chaudhury, S., S. Lyskov, and J.J. Gray. 2010. PyRosetta: A script-based interface for implementing molecular modeling algorithms using Rosetta. *Bioinformatics* 26, no. 5: 689–691.

Cornelis, H., A.L. Rodriguez, A.D. Coop, and J.M. Bower. 2012. Python as a federation tool for GENESIS 3.0. *PLOS One* 7, no. 1.

Dalcin, L.D., R.R. Paz, P.A. Kler, and A. Cosimo. 2011. Parallel distributed computing using Python. *Advances in Water Resources* 34, no. 9: 1124–1139.

Hughes, J.D., C.D. Langevin and E.R. Banta. 2017. Documentation for the MODFLOW 6 framework. *U.S. Geological Survey Techniques and Methods*, book 6, chap. A57, 40 p., https://doi.org/10.3133/tm6A57.

Jing, M., F. Heße, R. Kumar, W. Wang, T. Fischer, M. Walther, M. Zink, A. Zech, L. Samaniego, O. Kolditz, and S. Attinger. 2018. Improved regional-scale groundwater representation by the coupling of the mesoscale Hydrologic Model (mHM v5.7) to the groundwater model OpenGeoSys (OGS). *Geoscientific Model Development* 11, no. 5: 1989–2007.

Kolditz, O., S. Bauer, L. Bilke, N. Böttcher, J.-O. Delfs, T. Fischer, U.J. Görke, T. Kalbacher, G. Kosakowski, I. McDermott, C.-H. Park, F.A. Radu, K. Rink, H. Shao, H. Shao, F. Sun, Y.Y. Sun, A.K. Singh, J. Taron, M. Walther, W. Wang, N. Watanabe, Y. Wu, M. Xie, W. Xu, and B. Zehner. 2012. Newblock Opengeosys: An open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (thm/c) processes in porous media. *Environmental Earth Sciences* 67, no. 2: 589–599.

Müller, M., D. Parkhurst, and S. Charlton. 2012. Programming phreeqc calculations with C++ and python—A comparative study. In *Integrated Hydrological Modeling*, ed. R. Maxwell, E. Poeter, M. Hill, and C. Zheng, 632–636.

Müller, S. 2019. ogs5py v1.0.5. *Zenodo* https://doi.org/10.5281/zenodo.3546035

O'Boyle, N.M., C. Morley, and G.R. Hutchison. 2008. Pybel: A Python wrapper for the OpenBabel cheminformatics toolkit. *Chemistry Central Journal* 2.

Perez, F., B.E. Granger, and J.D. Hunter. 2011. Python: An ecosystem for scientific computing. *Computing in Science & Engineering* 13, no. 2: 13–21.

Roest, H.L., U. Schmitt, R. Aebersold, and L. Malmstroem. 2014. pyOpenMS: A Python-based interface to the OpenMS mass-spectrometry algorithm library. *Proteomics* 14, no. 1: 74–77.

Savoy, H., T. Kalbacher, P. Dietrich, and Y. Rubin. 2017. Geological heterogeneity: Goal-oriented simplification of structure and characterization needs. *Advances in Water Resources* 109, no. Suppl. C: 1–13.

Schlömer, N. 2018. pygmsh. *Zenodo*. https://doi.org/10.5281/zenodo.1173106

Trefry, M.G., and C. Muffels. 2007. Feflow: A finite-element ground water flow and transport modeling tool. *Groundwater* 45, no. 5: 525–528.

Zambelli, P., S. Gebbert, and M. Ciolli. 2013. Pygrass: An object oriented python application programming Interface (API) for geographic resources analysis support system (GRASS) geographic information system (GIS). *ISPRS International Journal of Geo-Information* 2, no. 1: 201–219.