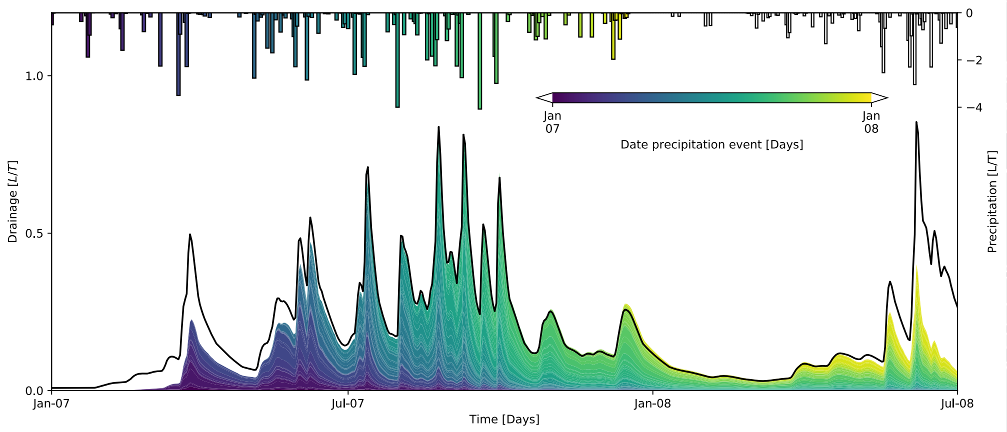
DOCTORAL PROGRAMME CLIMATE CHANGE

KARL-FRANZENS-UNIVERSITÄT GRAZ
UNIVERSITY OF GRAZ
FWF-DK Climate Change

Interfacing FORTAN Code with Python: an example for the Hydrus-1D model



Raoul Collenteur
Matevz Vremec
Giuseppe Brunetti

EGU General Assembly 2020

Session EOS7.10. Open Hydrology: Advances towards fully reproducible, re-usable and collaborative research methods in Hydrology

Motivation for Phydrus

"The wish/need to create a Python interface to capture the entire modelling Hydrus-1D[1] process in scripts to ensure reproducibility and simplify more complex analyses"

Advantages:

- Make use of legacy code that has a proven record, not reinventing the wheel
- Fortran / C code is often faster than interpreted languages (e.g., Python or R)
- Make accessing and using the legacy code through scripts a community effort
- No need to use a GUI to create a model, all steps are recorded in a Python script
- Enable more complex model analysis (e.g., uncertainty analysis, climate projections, sensitivity analyses) with little extra work and higher reproducibility

[1] Šimůnek, J. and M. Th. van Genuchten (2008) Modeling nonequilibrium flow and transport with HYDRUS, Vadose Zone Journal.

Phydrus Overview

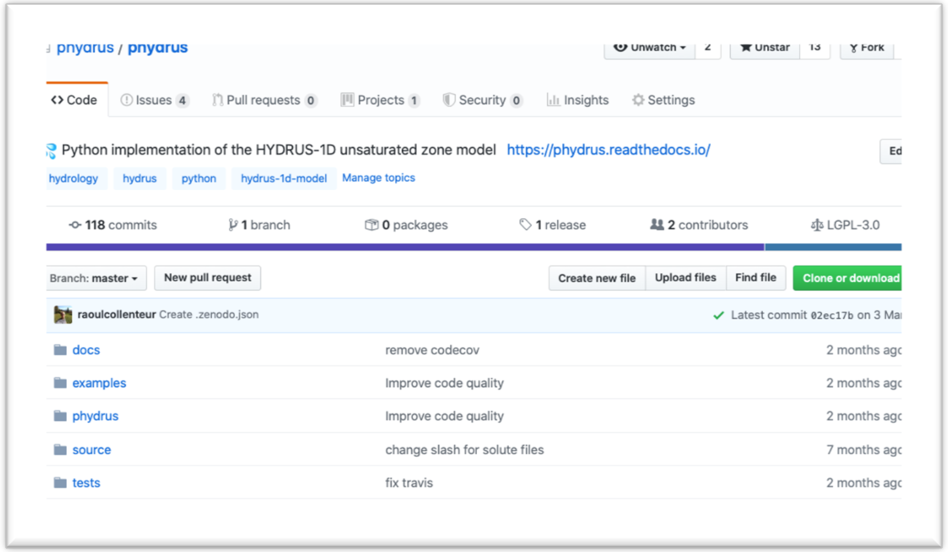
Description: Phydrus (pronounced as "Pie-drus") is an open source Python package to create, run, optimize, and visualize Hydrus-1D models. The entire modelling process is performed through Python scripts; from creating the soil profile, to adding (time-varying) boundary conditions and different flow and transport processes, to running the Hydrus-1D model and visualizing the results.

License: LGPL-3.0 Licensed

Source code: https://github.com/phydrus/phydrus

Docs: https://phydrus.readthedocs.io

Installation: >>> pip install phydrus

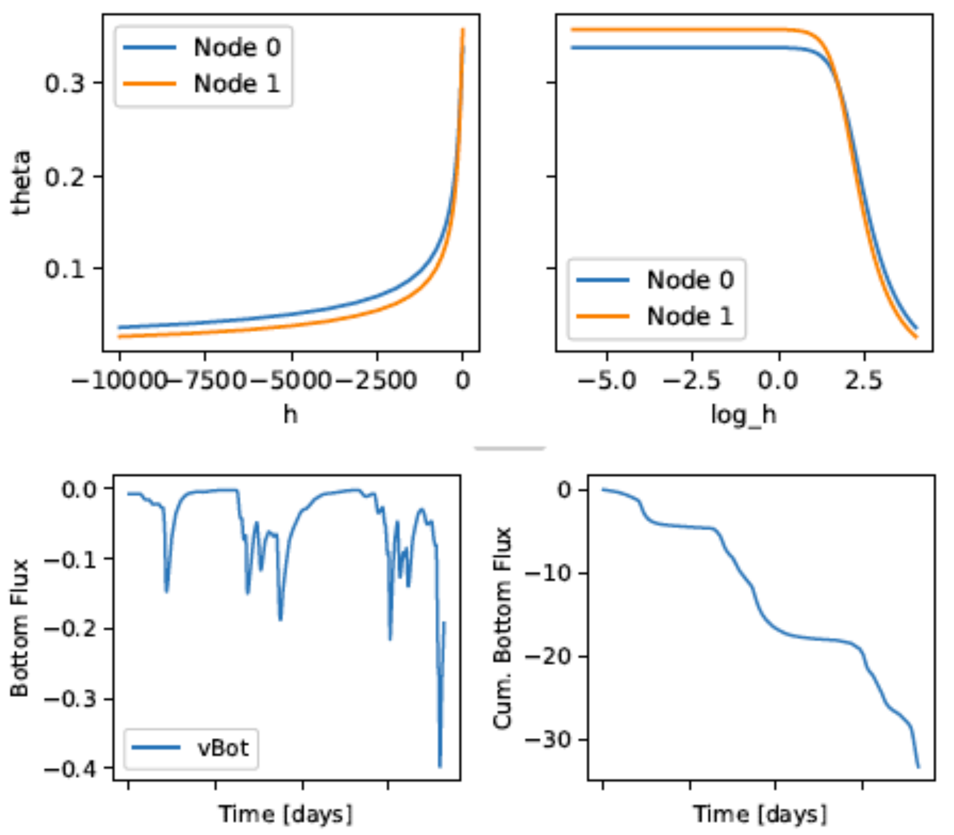


Simple Example Script

```
In[0]: import phydrus as ps

In[1]: ml = ps.Model(exe_name=exe, ws_name=ws, name="model",
                     time_unit="days", length_unit="cm")
In[2]: ml.add_time_info(tinit=0, tmax=730)

In[3]: ml.add_waterflow(top_bc=3, bot_bc=4)
In[4]: m = ml.get_empty_material_df(n=2)
In[5]: m.loc[0:2] = [[0.0, 0.3382, 0.0111, 1.4737, 13, 0.5],
                     [0.0, 0.3579, 0.0145, 1.5234, 50, 0.5]]
In[6]: ml.add_material(m)

In[7]: profile = ps.create_profile(top=0, bot=[-30, -100],
                                   dx=1, h=-500, mat=[1,2] )
In[8]: ml.add_profile(profile)

In[9]: atm = pd.read_csv("atmosphere.csv", index_col=0)
In[10]: ml.add_atmospheric_bc(atm, hcrits=0)
In[11]: ml.add_obs_nodes([-10, -50])
In[12]: ml.write_files()
In[13]: ml.simulate()
```

python powered

Example output figures



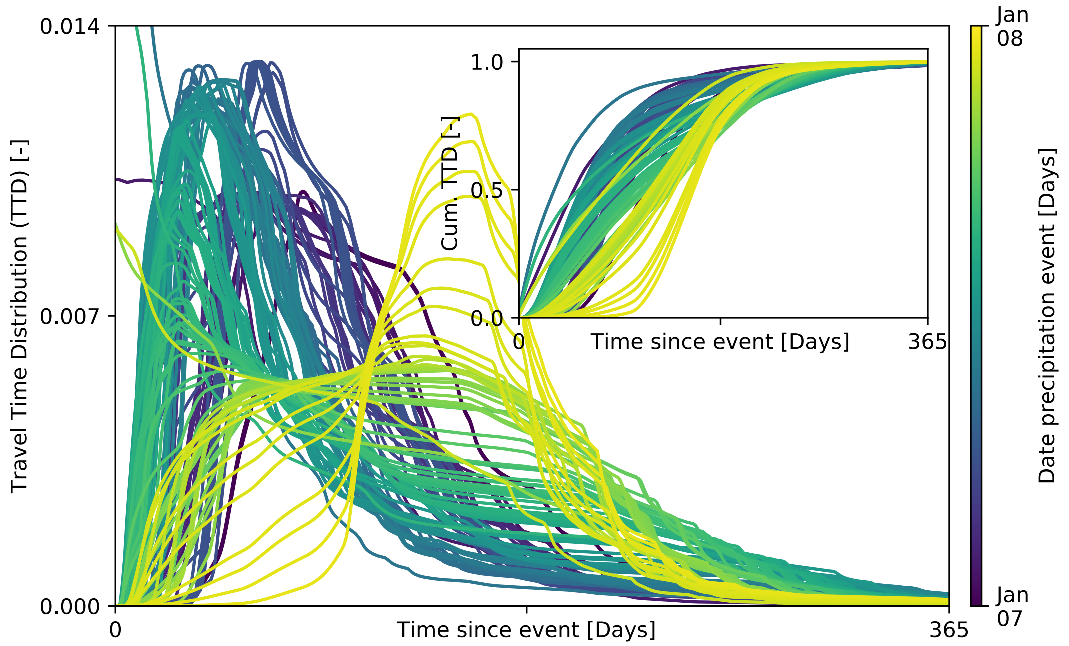
Check out the full example here: https://github.com/phydrus/phydrus/tree/master/examples/phydrus_paper/

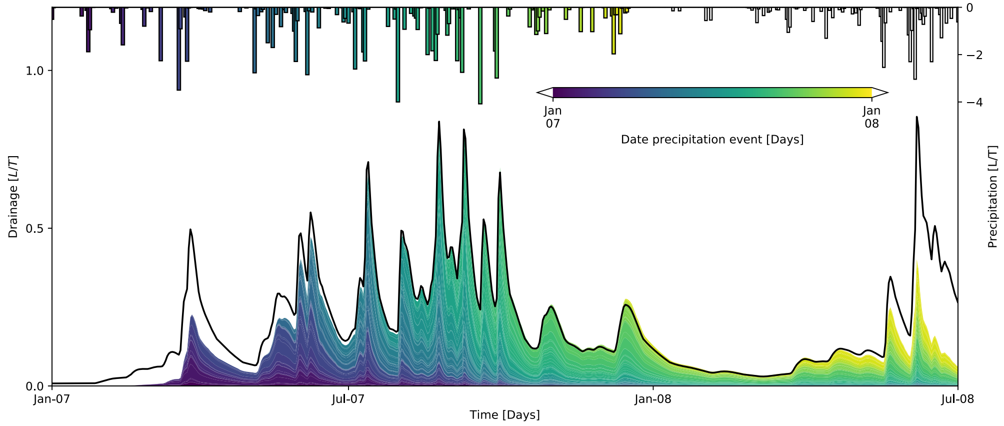
The power of scripts

Example Usage: Estimating travel time distributions

In this example the model created on the previous slide is used to estimate travel time distributions for the water leaving the bottom of the soil column. For each precipitation event we record when that exact water leaves the soil column. This requires us to run the model N number of times, where N is the number of precipitation events. This analysis can be done in only a few lines of code shown below:

```
data = {}

for i in atm.loc[:365].loc[atm.loc[:365, "Prec"] > 0].index:
    ml.atmosphere.loc[:, "cTop"] = 0.0
    ml.atmosphere.loc[i, "cTop"] = 1.0
    ml.write_atmosphere()
    ml.simulate()
    s = ml.read_solutes()
    data[i] = s.loc[:, "cBot"]
```





Check out the full example here: https://github.com/phydrus/phydrus/tree/master/examples/phydrus_paper/

A few tips for interfacing Fortan Code and making Python Packages

Start on paper, don't start programming:

1. **Think about how users will finally use the software.**
   For example, in Hydrus-1D the steps of the modelling process were already clearly defined. It can help to make a minimal example script of how you think it should be used.

2. **Design the software structure on paper or whiteboard**
   While it is tempting to start programming right away, it is good practice to first "design" the program on paper. Define the classes and methods before programming, for example by drawing UML diagrams.

3. **Get in contact with the original Authors of the code (if not yours).**
   Even while the original Fortran code may have been published under an Open Source license, (LGPL in case of Hydrus-1D) it is generally a nice idea to contact the original authors about your package and plans ☺.

Supported modules & Future plans

- Most standard modules are now supported, heat transport module which will be added shortly.

- Inverse estimation of model parameters and parameter sensitivity analysis is possible through numerous Python packages that are freely available.

Near Future Plans

- Compile Fortan code automatically using Conda Forge
- Provide more example applications
- Extend documentation
- Write a Technical description paper

We welcome contributions at GitHub and future collaborations to improve the package and its capabilities!

Table 1
Overview of the HYDRUS-1D modules and the modules supported in Phydrus at time of publication. *Inverse optimization is possible through Python.

Hydrus 1-D Module	Phydrus Method	Supported
Processes		
water flow	add_waterflow	Yes
solute transport	add_solute_transport	Yes
root water uptake	add_root_uptake	Yes
root growth	add_root_growth	Yes
ion exchange	-	No
heat transport	-	No
CO_2 transport	-	No
Physical properties		
soil profile	add_profile	Yes
materials	add_materials	Yes
solutes	add_solutes	Yes
atmospheric BC	add_atmosphere	Yes
Observation nodes	add_obs_nodes	Yes
Drains	-	No
Inverse Optimization	-	No*

For more information contact us on GitHub, Twitter or send an email (raoul.collenteur@uni-graz.at)