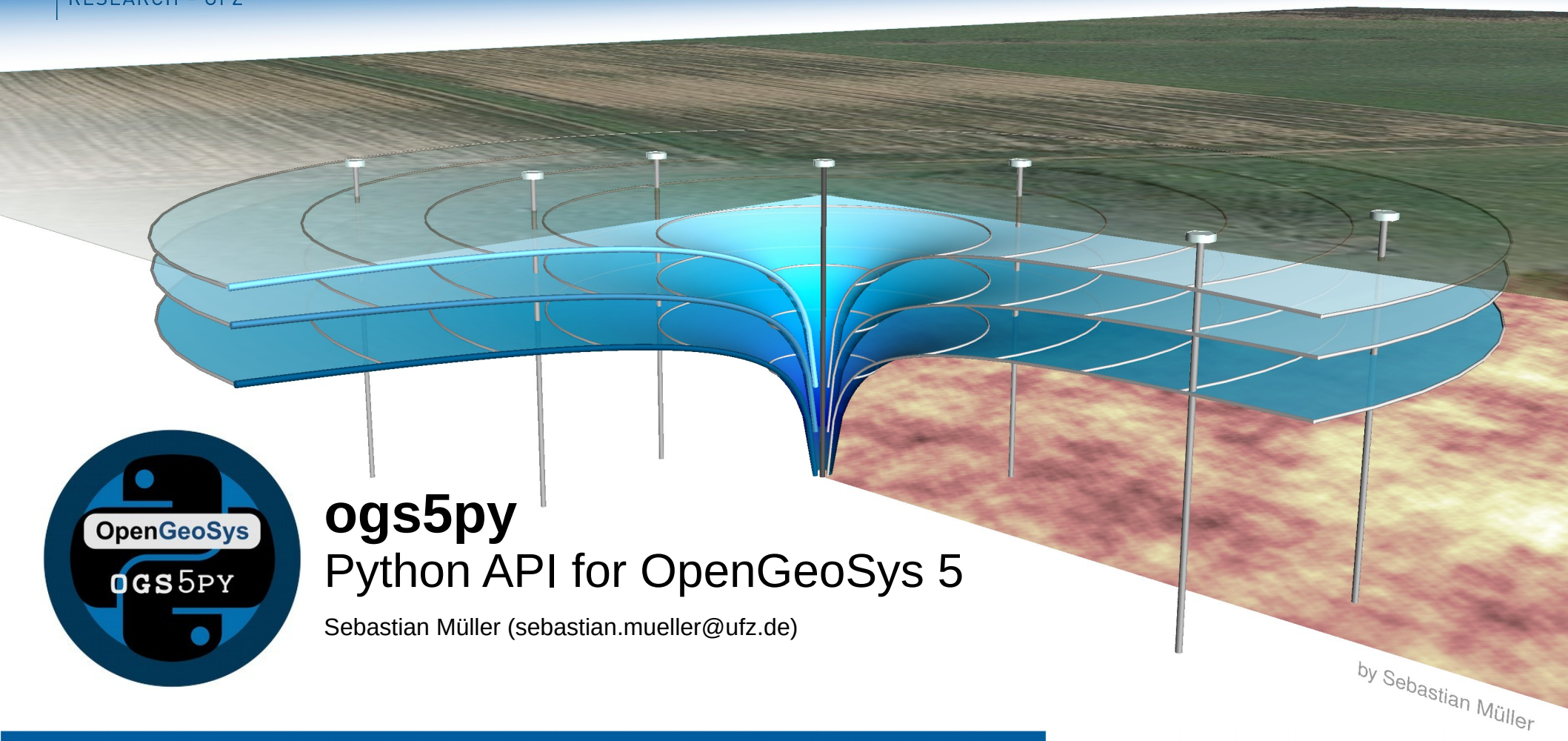
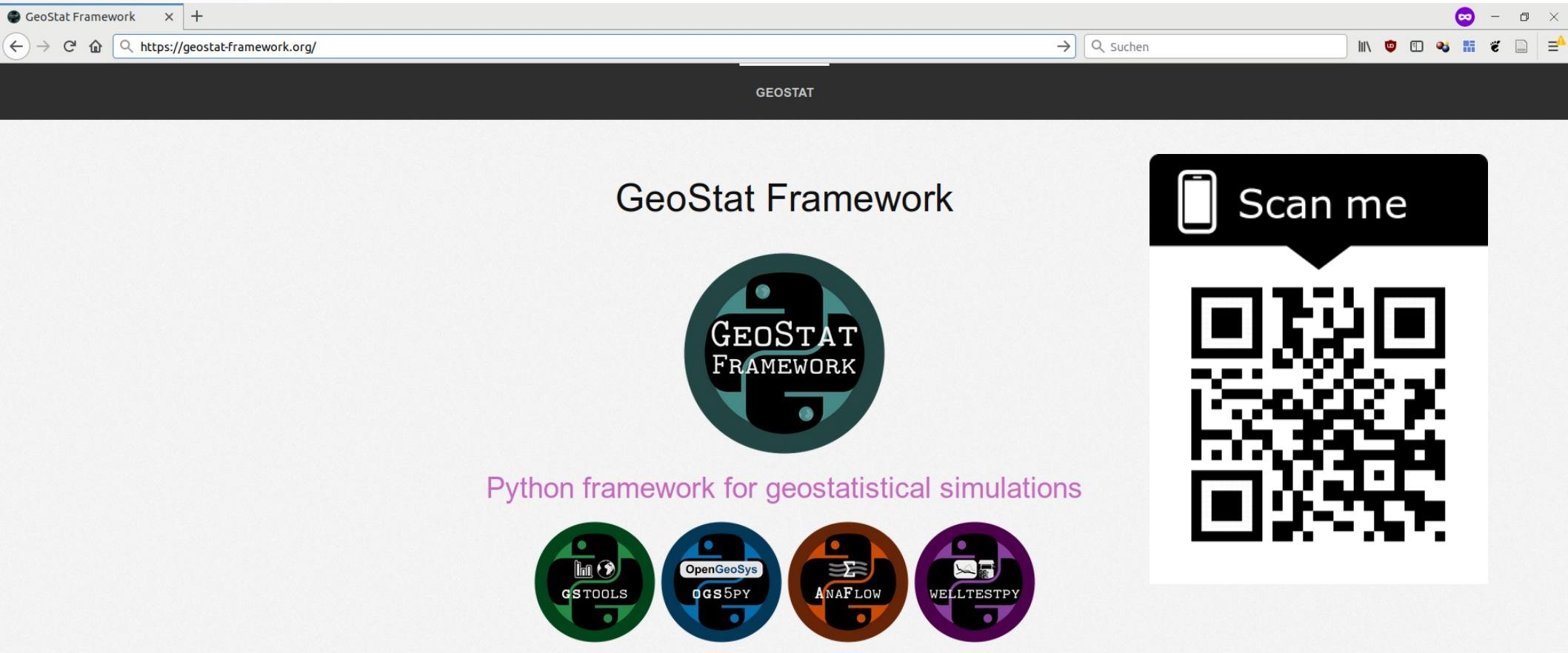


ogs5py

Python API for OpenGeoSys 5

Sebastian Müller (sebastian.mueller@ufz.de)

GeoStat Framework



Why an API?

- **Data from different sources**
  - Boundary Conditions
  - Meshes/Geometries
  - Conductivity Fields
- **Dynamical model setup**
  - Single script
  - Ensemble runs (mpi4py)
  - Modify existing models
- **Post-processsing**
  - Access output with Python
  - Reader for VTK/TecPlot
  - Sorted by PCS/primary-var
- **Why Python?**
  - Easy to use
  - Open source
  - Fully flexible language

How to get ogs5py?

- Installation: `pip install ogs5py`
- Source-Code: github.com/GeoStat-Framework/ogs5py
- Documentation: geostat-framework.readthedocs.io/projects/ogs5py
- PyPI release: pypi.org/project/ogs5py
- Zenodo: doi.org/10.5281/zenodo.2546767
- **Works on:**
  - Python 2 + 3
  - Unix + Windows

Workflow

```python
from ogs5py import OGS              # base class for the model
from ogs5py.reader import readpvd   # reader for PVD output
model = OGS(
    task_root="test_folder",        # folder for input files
    task_id="model",                # model name
    output_dir="output",            # output directory
)

### setup input ########################################

model.write_input()                 # write the input files
success = model.run_model()         # run ogs (return state)

out = readpvd(                      # get a dictionary
    task_root="output",             # with all pvd output
    task_id="model",                # sorted by process
    pcs="ALL",
)
```

- **OGS class attributes**
  - Files: `bc, cct, ddc, fct, gem, gli, ic, krc, mcp, mfp, mmp, msh, msp, num, out, pcs, pct, pqc, pqcdat, rei, rfd, st, tim`
  - Lists: `mpd, gli_ext, rfr, gem_init, asc, copy_files`
  - Strings: `task_root, task_id, output_dir, top_com, bot_com`
- **OGS class methods**
  - `add_/del_[mpd, gli_ext, …]()`
  - `load_model()`
  - `reset()`
  - `run_model()`
  - `write_input()`

OGS5 input files

- **Block files** (main case, data handled block-wise)
  - ```
    #<main_key>   or   #<main_key>
     $<sub_key>          <content>
      <content>          …
      …                  …
    #STOP                #STOP
    ```
    } one block
  - BC, CCT, DDC, FCT, GEM, IC, KRC, MCP, MFP, MMP, MPD, MSP, NUM, OUT, PCS, REI, RFD, ST, TIM
- **Line-wise files** (additional files handled as list of lines)
  - ASC (from TIM, PCS, GEM …), PQC + phreeqc.dat, GEMS3K files (dch, ipm, dbr)
- **Special files**
  - GLI, MSH (extended functionality provided)
  - PCT, RFR (other handling)
    ```

Modify block files

```python
from ogs5py import OGS
model = OGS(
    task_root="test_folder",
    task_id="model",
    output_dir="output",
)
model.ic.add_block(
    PCS_TYPE="GROUNDWATER_FLOW",
    PRIMARY_VARIABLE="HEAD",
    GEO_TYPE="DOMAIN",
    DIS_TYPE=["CONSTANT", 0.0],
)
model.ic.write_file()
```

```
|------------------ Written with ogs5py -----------------|
#INITIAL_CONDITION
  $PCS_TYPE
   GROUNDWATER_FLOW
  $PRIMARY_VARIABLE
   HEAD
  $DIS_TYPE
   CONSTANT 0.0
  $GEO_TYPE
   DOMAIN
#STOP
|-- Written with ogs5py (0.6.1) on: 2019-03-18_13-12-22 --|
```

- **BlockFile class attributes**
  - MKEYS (possible main keywords)
  - SKEYS (possible sub keywords)
  - is_empty (state if file is empty)
- **BlockFile class methods**
  - add_block()
  - update_block() (modify existing)
  - get_block() (access data)
  - read_file() (load existing file)
  - write_file() (to model folder)
  - add_copy_link()(copy existing file)
  - save() (arbitrary location)

The MSH file

```python
model.msh.generate(
    "rectangular",
    dim=2,
    element_no=(200, 200),
    element_size=(1.0, 1.0),
)
# or load an unstructured mesh
model.msh.import_mesh("mesh.vtk")
```

- **MSH class methods**
  - combine_mesh() (with another ogs-mesh)
  - [import/export]_mesh() (multiple types)
  - generate()
  - load() / save() (ogs meshes IO)
  - rotate() / shift() / transform()

- **MSH class attributes**
  - AXISYMMETRY
  - CROSS_SECTION
  - ELEMENTS (sorted by type)
  - ELEMENT_[ID/NO]
  - GEO_[NAME/TYPE]
  - LAYER
  - MATERIAL_ID[_flat]
  - NODES[_NO]
  - PCS_TYPE
  - centroids[_flat] (centers of elem.)
  - volumes[_flat] (volumes of elem.)

The GLI file

```python
model.gli.generate(
    "rectangular",
    dim=2,
    ori=(-100.0, -100.0),
    size=(200.0, 200.0),
    name="boundary",
)
# add the pumping well
model.gli.add_points([0.0, 0.0, 0.0], "pwell")
# add observations
model.gli.add_points([10.0, 0.0, 0.0], "owell")
# add line
model.gli.add_polyline(
    points=["pwell", "owell"],
    name="line"
)
```

- **GLI class methods**
  - add_[points/polyline/surface/volume]
  - generate()
  - load() / save()
  - rotate() / shift() / transform()

- **GLI class attributes**
  - POINTS[_MD/_NANES/_NO]
  - POLYLINES[_NAMES/_NO]
  - SURFACES[_NAMES/_NO]
  - VOLUMES[_NAMES/_NO]

```
|------------------ Written with ogs5py ------------------|
#POINTS
0 -100.0 -100.0 0.0
1 100.0 -100.0 0.0
2 100.0 100.0 0.0
3 -100.0 100.0 0.0
4 0.0 0.0 0.0 $NAME pwell
5 10.0 0.0 0.0 $NAME owell
#POLYLINE
  $NAME
   boundary
  $POINTS
   0
   1
   2
   3
   0
#POLYLINE
  $NAME
   line
  $POINTS
   4
   5
#STOP
|-- Written with ogs5py (0.6.1) on: 2019-03-18_12-22-14 --|
```

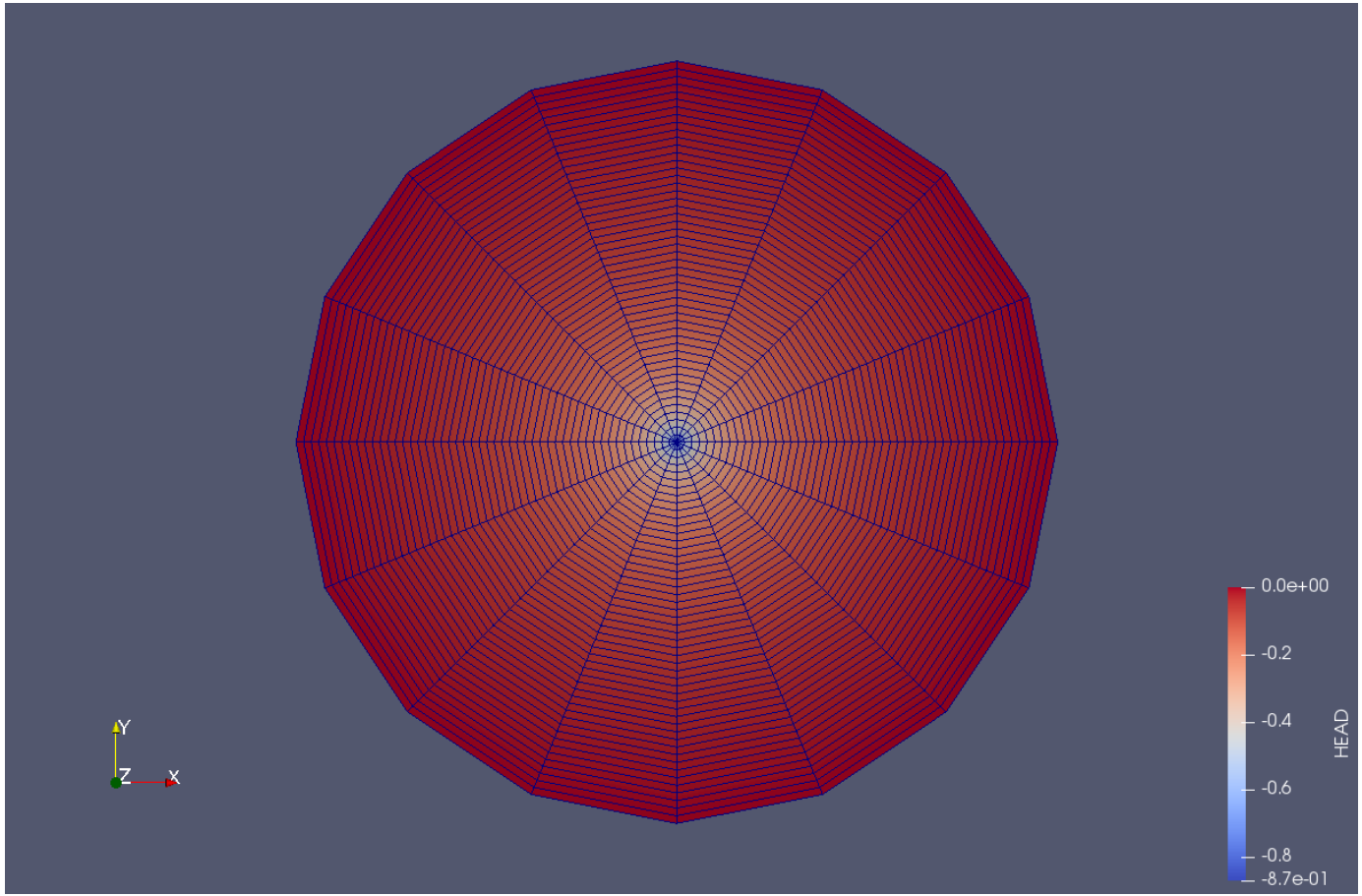
Pumping test example

```python
from ogs5py import OGS
model = OGS(task_root="pump_test", task_id="model")
# generate a radial mesh + gli
model.msh.generate("radial", dim=2, rad=range(51))
model.gli.generate("radial", dim=2, rad_out=50.)
model.gli.add_points([[0., 0., 0.], [1., 0., 0.]], ["pwell", "owell"])
# define input files
model.bc.add_block(  # boundary condition
    PCS_TYPE='GROUNDWATER_FLOW',
    PRIMARY_VARIABLE='HEAD',
    GEO_TYPE=['POLYLINE', "boundary"],
    DIS_TYPE=['CONSTANT', 0.0],
)
model.st.add_block(  # source term
    PCS_TYPE='GROUNDWATER_FLOW',
    PRIMARY_VARIABLE='HEAD',
    GEO_TYPE=['POINT', "pwell"],
    DIS_TYPE=['CONSTANT_NEUMANN', -1.0e-04],
)
model.ic.add_block(  # initial condition
    PCS_TYPE='GROUNDWATER_FLOW',
    PRIMARY_VARIABLE='HEAD',
    GEO_TYPE='DOMAIN',
    DIS_TYPE=['CONSTANT', 0.0],
)
model.mmp.add_block(  # medium properties
    GEOMETRY_DIMENSION=2,
    STORAGE=[1, 1.0e-04],
    PERMEABILITY_TENSOR=['ISOTROPIC', 1.0e-4],
    POROSITY=0.2,
)
```

...

```python
model.num.add_block(  # numerical solver
    PCS_TYPE='GROUNDWATER_FLOW',
    LINEAR_SOLVER=[2, 5, 1.0e-14, 1000, 1.0, 100, 4],
)
model.out.add_block(  # domain output
    PCS_TYPE='GROUNDWATER_FLOW',
    NOD_VALUES='HEAD',
    GEO_TYPE='DOMAIN',
    DAT_TYPE='PVD',
    TIM_TYPE=['STEPS', 1],
)
model.out.add_block(  # point observation
    PCS_TYPE='GROUNDWATER_FLOW',
    NOD_VALUES='HEAD',
    GEO_TYPE=['POINT', "owell"],
    DAT_TYPE='TECPLOT',
    TIM_TYPE=['STEPS', 1],
)
model.pcs.add_block(  # set the process type
    PCS_TYPE='GROUNDWATER_FLOW',
    NUM_TYPE='NEW',
)
model.tim.add_block(  # set the timesteps
    PCS_TYPE='GROUNDWATER_FLOW',
    TIME_START=0,
    TIME_END=600,
    TIME_STEPS=[[10, 30], [5, 60]],
)
model.write_input()
success = model.run_model()
```

Pumping test example



Pumping test example

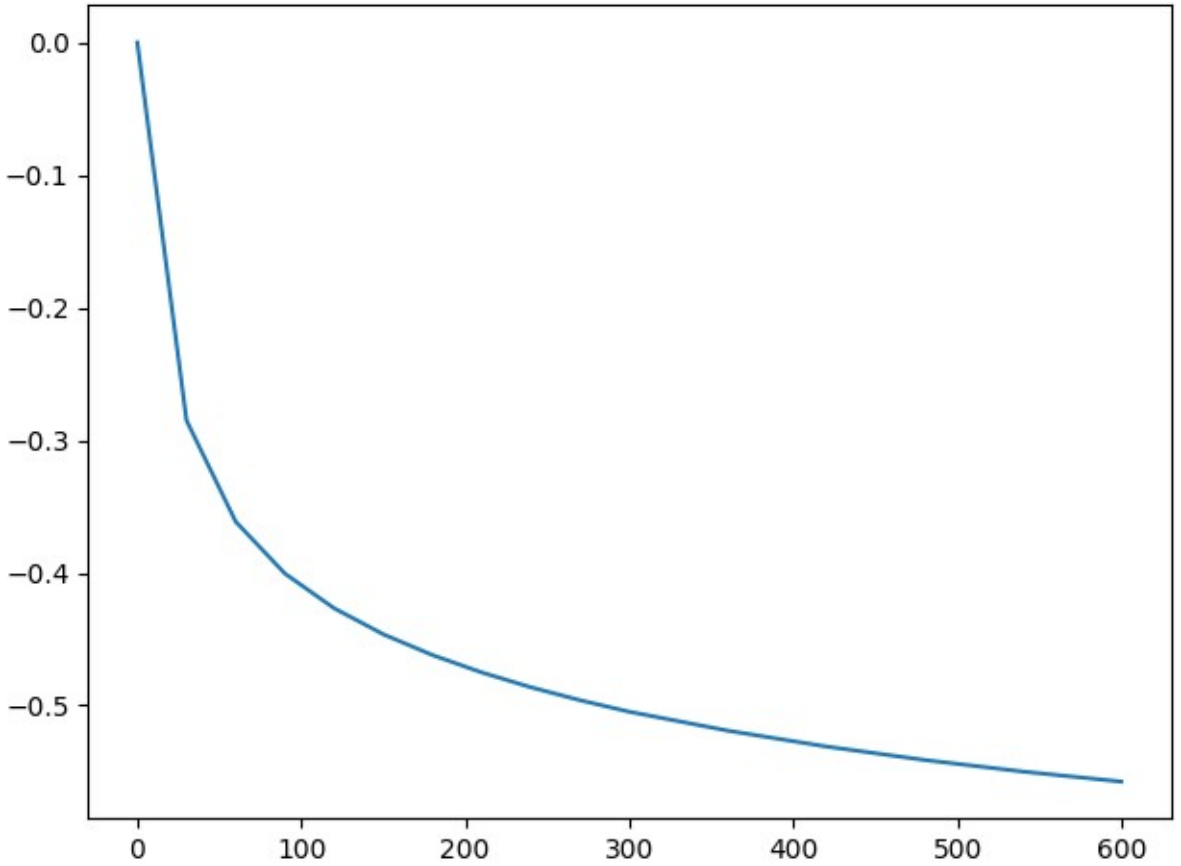
- **Log-File**
  - OGS output saved as log
  - Can be shown during computation
  - Will be saved in the specified output folder
  - File name contains actual data by default e.g.:

    `model_2019-03-18_12-16-10_log.txt`

```
          ##################################################
          ##                                              ##
          ##              OpenGeoSys-Project              ##
          ##                                              ##
          ##  Helmholtz Center for Environmental Research ##
          ##    UFZ Leipzig - Environmental Informatics   ##
          ##                  TU Dresden                  ##
          ##              University of Kiel              ##
          ##            University of Edinburgh           ##
          ##         University of Tuebingen (ZAG)        ##
          ##       Federal Institute for Geosciences      ##
          ##         and Natural Resources (BGR)          ##
          ##  German Research Centre for Geosciences (GFZ) ##
          ##                                              ##
          ##    Version 5.7(WH/WW/LB)  Date 07.07.2015    ##
          ##                                              ##
          ##################################################

          File name (without extension): model

------------------------------------------
Data input:
GEOLIB::readGLIFile open stream from file model.gli ... done
read points from stream ...  ok, 18 points read
read polylines from stream ...  ok, 1 polylines read
tag #SURFACE not found or input stream error in GEOObjects
PCSRead ... done, read 1 processes
...
```

Postprocessing

```python
from ogs5py.reader import readtec_point
from matplotlib import pyplot as plt

point = readtec_point(
    task_root="pump_test",
    task_id="model",
    pcs='GROUNDWATER_FLOW',
)
time = point['owell']["TIME"]
head = point['owell']["HEAD"]

plt.plot(time, head)
plt.show()
```



- **Reader**
  - PVD / VTK / TEC-PLOT
  - PCS can be specified
  - Output as Dictionaries
  - Plotting ready

Script generation

```python
from ogs5py import OGS
# create empty model
model = OGS(task_root="theis", task_id="model")
# load existing OGS5 model
model.load_model("H/Theis/GWF_Theis_1-5D")
# generate ogs5py script in "script_dir"
model.gen_script("script_dir")
```

```
script_dir/
  |
  +- model.gli
  |
  +- model.msh
  |
  +- model.py
```

```python
from ogs5py import OGS

model = OGS(
    task_root='theis',
    task_id='model',
)
model.bc.add_block(
    main_key='BOUNDARY_CONDITION',
    PCS_TYPE='GROUNDWATER_FLOW',
    PRIMARY_VARIABLE='HEAD',
    GEO_TYPE=['POINT', 'INIFINIT'],
    DIS_TYPE=['CONSTANT', 0.0],
)
model.gli.read_file('model.gli')
model.ic.add_block(
    main_key='INITIAL_CONDITION',
    PCS_TYPE='GROUNDWATER_FLOW',
    PRIMARY_VARIABLE='HEAD',
    GEO_TYPE='DOMAIN',
    DIS_TYPE=['CONSTANT', 0.0],
)
model.mfp.add_block(
    main_key='FLUID_PROPERTIES',
    FLUID_TYPE='LIQUID',
    PCS_TYPE='HEAD',
    DENSITY=[1, 0.0],
    VISCOSITY=[1, 1.0],
)
model.mmp.add_block(
    main_key='MEDIUM_PROPERTIES',
    GEOMETRY_DIMENSION=1,
    GEOMETRY_AREA=1,
    STORAGE=[1, 0.001],
    PERMEABILITY_SATURATION=[1, 1.0],
    PERMEABILITY_TENSOR=['ISOTROPIC', 0.000929036],
)
model.msh.read_file('model.msh')
model.num.add_block(
    main_key='NUMERICS',
    PCS_TYPE='GROUNDWATER_FLOW',
    LINEAR_SOLVER=[2, 5, 1e-14, 1000, 1.0, 100, 4],
    RENUMBER=[2, -1],
)
```

```python
model.out.add_block(
    main_key='OUTPUT',
    PCS_TYPE='GROUNDWATER_FLOW',
    NOD_VALUES='HEAD',
    GEO_TYPE='DOMAIN',
    DAT_TYPE='TECPLOT',
    TIM_TYPE=['STEPS', 1],
)
model.out.add_block(
    main_key='OUTPUT',
    PCS_TYPE='GROUNDWATER_FLOW',
    NOD_VALUES='HEAD',
    GEO_TYPE=['POINT', 'OBS'],
    DAT_TYPE='TECPLOT',
    TIM_TYPE=['STEPS', 1],
)
model.pcs.add_block(
    main_key='PROCESS',
    PCS_TYPE='GROUNDWATER_FLOW',
    NUM_TYPE='NEW',
    PRIMARY_VARIABLE='HEAD',
)
model.st.add_block(
    main_key='SOURCE_TERM',
    PCS_TYPE='GROUNDWATER_FLOW',
    PRIMARY_VARIABLE='HEAD',
    GEO_TYPE=['POINT', 'WELL'],
    DIS_TYPE=['CONSTANT_NEUMANN', -194.69],
)
model.tim.add_block(
    main_key='TIME_STEPPING',
    PCS_TYPE='GROUNDWATER_FLOW',
    TIME_STEPS=[
        [10, 1e-05],
        [10, 9e-05],
        [10, 0.0009],
        [10, 0.009],
        [10, 0.09],
        [10, 0.9],
    ],
    TIME_END=10,
    TIME_START=0.0,
    TIME_UNIT='DAY',
)
model.write_input()
model.run_model()
```

- **Script generator**
  - Load your existing model
  - Convert to one ogs5py script
  - *.msh, *.gli separate
  - Converted Benchmarks: github.com/GeoStat-Framework/ogs5py_benchmarks

Ensemble runs with MPI4py

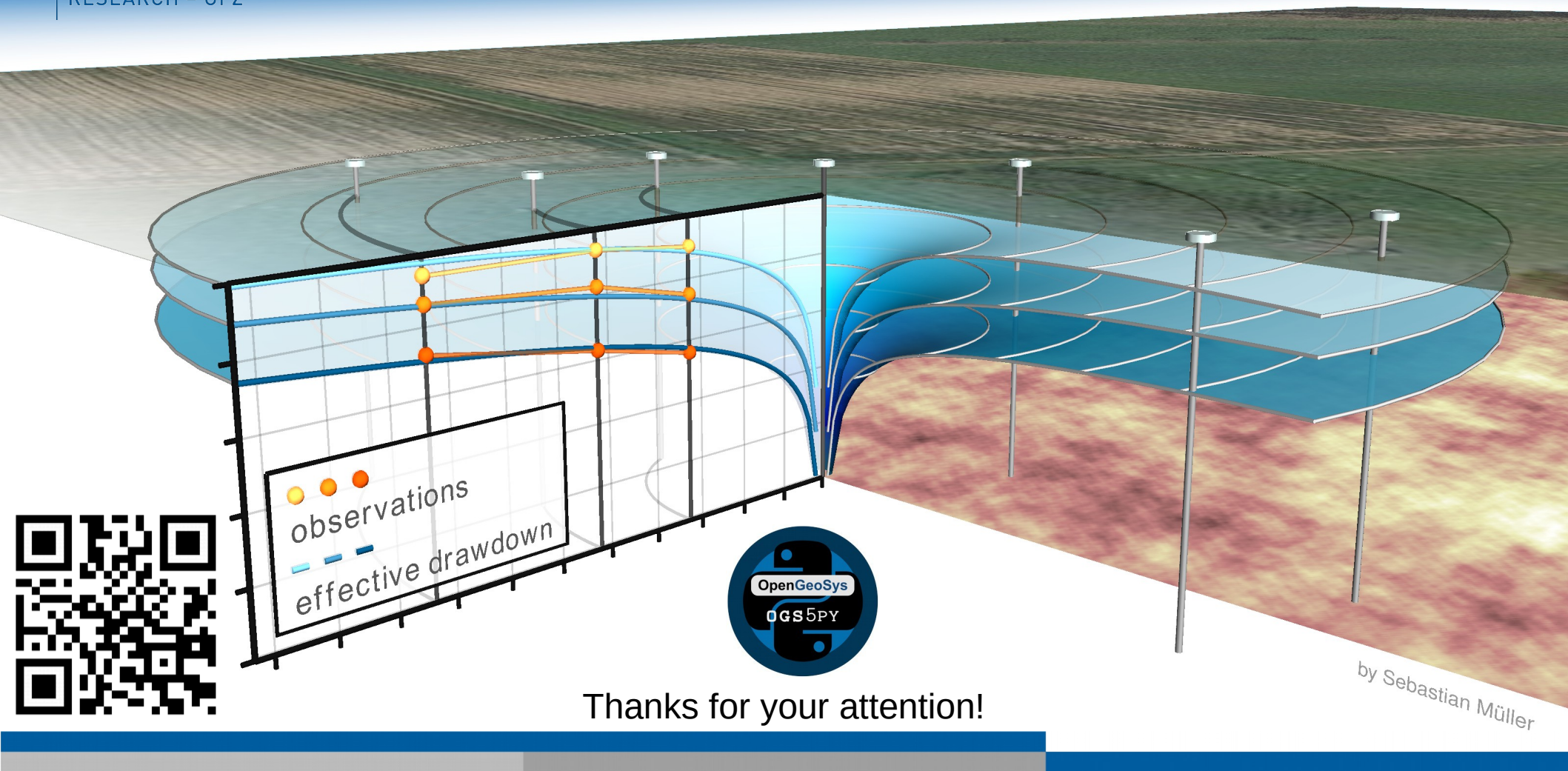
```python
from ogs5py import OGS
from mpi4py import MPI

# rank is the actual core-number, size is total number of cores
rank = MPI.COMM_WORLD.Get_rank()
size = MPI.COMM_WORLD.Get_size()

ogs = OGS(task_root="core_{}-{}".format(rank, size))
# ...configure ogs model

for seed in range(1000):
    # parallel running the right jobs on the right core
    if seed % size != rank:
        continue
    # generate the new transmissivity field
    trans = gen_trans(seed=seed)  # GSTools !!!
    # update the MPD file with the new transmissivity field
    ogs.mpd[0].update_block(DATA=zip(range(len(trans)), trans))
    ogs.mpd[0].write_file()
    # set the new output-directory
    ogs.output_dir = "seed{:04}".format(seed)
    # run this job
    ogs.run_model(print_log=False)
```

- **mpi4py**
  - Simply parallelize your ensemble runs in python
  - Can be done within the script

```
mpiexec -n 4 python3 model.py
```

HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ

OpenGeoSys Community Meeting 2019



Thanks for your attention!