cmmpy

Alessandro Comunian

Feb 23, 2023

CONTENTS:

Welcome to the *cmmpy*'s documentation, a Python implementation of the comparison model method (CMM), a **direct** inversion method to obtain the hydraulic transmissivity T of a confined aquifer.

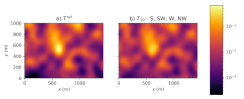

Fig. 1: Synthetic reference T field (left) and CMM inversion result, 10th iteration (right)

This is the accompanying software of the manuscript "Improving the robustness of the Comparison Model Method for the identification of hydraulic transmissivities", by A.Comunian and M.Giudici, published on Computers and Geosciences - DOI: 10.1016/j.cageo.2021.104705 -

CHAPTER

ONE

PURPOSE

This is a python implementation of the Comparison Model Method (CMM), a direct method to solve inverse problems in hydrogeology, and in particular to compute the hydraulic conductivity T of a confined aquifer given an initial tentative value of T and one or more interpolated hydraulic head fields h. This implementation of the CMM heavily relies on the USGS engines of the Modflow family (and Modflow6 in particular) to solve the forward problem, facilitated by the use of the Python module flopy. Nevertheless, it can be adapted to make use of other engines for the solution of the forward problem.

CHAPTER

TWO

INSTALLATION

Installing and using *cmmpy* should be relatively easy, and thanks to Python's flexibility, that should be feasible on many operative systems (OSs), including MS Windows, Mac OS X and Linux.

2.1 Requirements

To use the *cmmpy* module you need a standard Python3.X installation, together with the main mathematical and plotting libraries (`numpy`, `pandas`, `matplotlib` etc), and the USGS package `flopy`. If `pip` is used for the installation, all there requirement should be automatically.

Clearly, if you have installed `flopy` then you should also have installed the binaries related to the corresponding MODFLOW package.

Note: The current version of `cmmpy` is based on MODFLOW6 compiled with the *double* option.

2.2 Installation

The suggested way is to use `pip` (which should be also already available with *Anaconda*).

`cmmpy` is available at the Python Package Index repository. Therefore, in can be easily installed (together with its dependencies) with the command:

```
pip install cmmpy
```

Alternatively, if you prefer to download the sources from https://bitbucket.org/alecomunian/cmmpy, you can:

1) Clone or download this repository on your hard drive.
2) If required, unpack it and `cd cmmpy`.
3) Inside the project directory, from the command line:

```
pip install -e .
```

4) To check if it worked, open a Python terminal and try:

```
import cmmpy
```

CHAPTER

THREE

TUTORIAL

This is a brief tutorial to illustrate how to run a simple application that makes use of the `cmmpy`. For more details and more options, one can have a look at many examples available in the bitbucket/github repositories in the folder *test*. The files provided in the test folder cover almost all the examples explored in the companion paper published in Computers and Geosciences by A.Comunian and M.Giudici, DOI: 10.1016/j.cageo.2021.104705 (hereinafter, the *companion paper*).

3.1 Example - Tomographic approach

This example is based on the implementation of the CMM with a tomographic approach. It should allow to obtain part e) of Fig.9 of the companion paper.

Hereinafter we will briefly describe the main components required to run the CMM.

3.1.1 Setting up the forward problem

As many other inversion tools, the CMM requires to set up a tool to solve the forward problem (FP). Here the set up is implemented with `flopy` and the `Modflow6` version. Nevertheless, you can customize the code to use different tools for the solution of the FP, as it was for example done in the preliminary part of this study by using `Modflow2005`, or by Comunian and Giudici (DOI: 10.1007/s11004-018-9727-0) who applied the CMM with Parflow as flow simulation engine.

In the provided Python code, this is done in the function `run_fp` of the module `tool`. You can customize this function to suit your particular needs. However, if your have modeling assumption akin to the one stated in the companion paper, then you will not need to customize this function too much, maybe even not. In fact, the code is written in order to delegate as much as possible all the parameterization and a big part of the model settings to an editable `json` file, to avoid as much as possible edits on the function `run_fp` function. The content of this `json` file is described in the next section.

3.1.2 Parameter file

As mentioned in the previous section, the main modifications that a user should do to apply the CMM to his case study should be made on the data files and to the input parameter file (JSON file). An example of this file is provided for many test cases in the bitbucket/github repositories in the aforementioned *test* folder. For example, hereinafter it is provided a list of the entries of the file `test.json` contained in the folder `test/test07_md_paper/04_S-SW-W-NW`, explained one by one. In general, the parameters are grouped into four categories:

`general`
: some "general" parameters.

fwd
parameters related to the forward problem run.

t_gen
parameters related to the generation of the synthetic T distribution which is used as reference.

cmm
parameters related to the Comparison Model Method (CMM).

noise
parameters related to the addition of noise to the input head fields.

Let us start to describe in details the sub-categories of the aforementioned "main" categories:

general["wdir"]
The working directory where all the output files will be saved.

general["data"]
Directory containing some needed data sets. For example, the files containing the boundary conditions and the shape of the domain will be stored here. For examples of possible input files, you can have a look at the folder `test/data`

general["out"]
An output directory to save all the output needed by the flopy implementation.

fwd["ws"]
The workspace related to the Modflow problem

fwd["name"]
Name of the modflow problem

fwd["exe_name"]
The name of the modflow executable. In the example, the name `mf6dbl` is provided. However, in general, a more common name would be `mf6` (if you are working on Linux of macOS) or `mf6.exe`. Clearly, if the binary file of your modflow is not in the system PATH, you could also set here the explicit path.

fwd["bcs"]
A file that contains codes that indicate the type of the cells used for the boundary conditions. The next section *Input data files* contains a more detailed description of this input format.

fwd["nx"], fwd["ny"]
Numer of cells along the x and y coordinates

fwd["dx"], fwd["dy"]
Size of the cells side (in meters)

fwd["data_sets"]
This keyword contains the detailed description of all the multiple data sets that can be used within `cmmpy` to apply the CMM with a tomographic approach. The main components of this keyword is a list, containing one dictionary for each data set. The dictionary contain the following keywords: `name` defines a name for the corresponding data set, which at the moment is not explicitly used inside the code, but that is useful as reference to the data set; `h_BCs` is the name of the VTK file (contained in the folder defined by the previously mentioned keyword `data`) that contains the values of the fixed head boundary conditions.

fwd["rch"]
This contains the numerical value of a diffuse recharge term.

fwd["wells_loc"]
This should be a list containing three integers, that correspond to the cell location of the well (see for example the JSON files in the folder `test/test06_wells1a1`)

fwd["well_ID"]
A reference name for the well.

fwd["well_q"]
The abstraction rate of the well, with the same conventions used in *flopy*.

t_gen["seed"]
The seed for the pseudo-random generator

t_gen["dim"]
The dimension of the problem, 2D or 3D.

t_gen["var"]
Variance of the heterogeneous field.

t_gen["len_scale"]
The scale length of the simulated heterogeneity.

cmm["nb_iter"]
The maximum number of iterations required to run the CMM. For the case study analyzed in the companion paper, 10 iteration were OK.

cmm["cprop"]
The proportions of data to be rejected and where the low gradient values should be corrected.

cmm["eps_gradh"]
A threshold value for the hydraulic gradient (h).

cmm["mode"]
When using multiple data sets, this is the mode that is used to merge the T computed with the different input data. Allowed values are *arithmetic*, *geometric*, *harmonic*, *median*, and *mincorr* (there is also an "alternative" *mincorrX*). Default value is *geometric*. See the code and the companion paper for more details.

The same algorithm used for the generation of the synthetic field example is here used to generate a correlated noise field to be added to the input h data. If you do not need to add noise to your data, simply set the value of `std` to 0.0.

noise["seed"]
The seed for the pseudo-random generator

noise["dim"]
The dimension of the problem, 2D or 3D.

noise["var"]
Variance of the heterogeneous field.

noise["len_scale"]
The scale length of the simulated heterogeneity.

3.1.3 Input data files

This is a brief description of some of the input data files contained in the folder `test/data`.

shape of the domain and cells type
A matrix with the same shape of the domain should be provided, with the letter `I` for *internal cells*, `D` for *Dirichlet* fixed head BCs, `E` for external cells. See for example the file `bcs.txt` in the folder `test/data`.

files containing the head BCs values
These are VTK structured points files.

3.1.4 Run the test

Once you set up all the parameters in the JSON file and provided the required data files, you can run the inversion. If you take as reference the folder structures provided on Github or Bitbucket, you should first move to the folder `test`. Once there, from the shell, you can call the script `run_cmm.py` by providing the name of the JSON input file. For example, you could write:

```
./run_cmm.py test07_md_paper/04_S-SW-W-NW/test.json
```

At the end of the run, you can find the intermediate and the output files into the folder defined in the JSON file, for example in the folder `out/test07_md_paper/04_S-SW-W-NW/`.

CHAPTER

FOUR

INDICES AND TABLES

- genindex