**Monte Carlo Analysis of GEOPHIRES Inputs
Using Parallel Computations**

Jared D. Smith

July 16, 2018
Updated for the GDR on October 26th, 2019

**Description**

A Python script called ParallelGEOPHIRES.py is used to run GEOPHIRES v2.0 in parallel, save results, and make some figures of the results. An R script called PlottingFunctions.R is used to make more figures of the results and summarize results as Excel workbooks with worksheets for each output variable that summarize quantiles, mean, and standard deviation. A Python script called TimestepStabilityCheck.py is used to evaluate the stability of selected GEOPHIRES results with respect to the number of timesteps. These functions currently save all direct-use heat information, but are not yet programmed to save power information.

The ParallelGEOPHIRES.py script steps are as follows:

1. Read in the user-specified input file. Default name is ExampleParameterSpecificationTable.csv
	1. The script automatically checks that all input values are defined properly. Any errors for the user to fix are printed in the Python console window.
2. Generate standard uniform random samples for each of the variables in the input file.
3. Convert standard uniform samples into their specified distributions.
4. Run GEOPHIRES in parallel over the number of samples.
5. Write output files and some figures.
6. Call Rscript module to run PlottingFunctions.R directly within Python (requires Anaconda installation with Rscript installed – more details below).
7. Move all output files to a unique storage directory.

The PlottingFunctions.R script steps are as follows:

1. Load results file from step 5 above.
2. Write figures and summarize results into Excel workbooks.

The TimestepStabilityCheck.py script steps are as follows:

1. Read in the user-specified input file. Default name is ExampleParameterSpecificationTable.csv
	1. The script automatically checks that all input values are defined properly. Any errors for the user to fix are printed in the Python console window.
2. Convert the data into point values based on their specified distributions. Point values are used to run the stability check, keeping everything constant except for the number of timesteps per year in GEOPHIRES.
3. Run GEOPHIRES in parallel over the different number of timesteps.
4. Write output files and figures.
5. Move all output files to a unique storage directory.

**Warning**: I’ve programmed all of these scripts to work on Windows 10 OS. I’m not sure how well they will work on Mac, Linux or earlier versions of Windows.

**Input File Format for ParallelGEOPHIRES and TimestepStabilityCheck**

*Row names*

Rows are the input parameters to GEOPHIRES. These rows may be in any order. A full list and descriptions of these parameters are available in the file “InputParameterDescriptionsTables.xlsx.” Every parameter must be listed in the input file, but not every parameter will be used in every run. The GEOPHIRES user manual provides a description of what parameters are used for each model type selected.

***It is important that these row names remain unchanged from the InputParameterDescriptionsTables.xlsx file. They are called in the code.***

*Additional Input Parameters*

There are new GEOPHIRES input parameters for the Monte Carlo simulation:

‘TestID’ is an integer value that is used as a unique identifier for the whole run. It is also used to set the random seed for the Monte Carlo analysis. You should change the ‘TestID’ for each case that you run to avoid overwriting results, and to use different random seeds.

‘TestID’ should always have a point distribution (p), and ‘arg1’ specifies the ID number.

‘TIMESTEPS’ is the number of timesteps per year in GEOPHIRES.

‘TIMESTEPS’ should always have a point distribution (p), and ‘arg1’ specifies the value.

‘MCreps’ is the number of Monte Carlo replicates to run.

‘MCreps’ should always have a point distribution (p), and ‘arg1’ specifies the value.

There are also new GEOPHIRES parameters for the multiple parallel fractures model:

 ‘FracHalfWidth’ is the half-width of the fractures in meters.

*Column names*

‘dist’ specifies the probability distribution, and the ‘args’ are the arguments of the distribution (see table). I have uniform (u), beta (b), triangular (t), normal (n), lognormal (l), and point (p) distributions programmed. The table below describes what is needed for each distribution. Some values must be strictly greater than 0, and some other values must be greater than 0, as indicated in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **dist** | **arg1** | **arg2** | **arg3** | **arg4** |
| Point: specify as p | Value of parameter |  |  |  |
| Triangular: specify as t | Minimum | Maximum | Mode (triangle peak location) |  |
| Uniform: specify as u | Minimum | Maximum |  |  |
| Beta: Specify as b | Minimum | Maximum | Alpha parameterlower tail (> 0) | Beta parameterupper tail (> 0) |
| Normal: Specify as n | Mean | Standard Deviation (> 0) |  |  |
| Lognormal: Specify as l | Real space mean (≥ 0) | Real space standard dev. (> 0) | Real space lower bound (≥ 0) |  |

Use Normal and Lognormal with caution. Infinite tails can cause problems. Common problems with the Normal are generating negative values when negatives are impossible, and generating values below lower bounds. Lognormal and bounded distributions can prevent these problems. An “infinite tails” problem with normal and lognormal has potential to generate physically or technically infeasible values. Bounded distributions may be a better choice for some variables. Beta, for example, is bounded and can take a wide variety of shapes.

**User Interface with ParallelGEOPHIRES.py and TimestepStabilityCheck.py**

There are several lines of code that you may want to change. All are at the top.

|  |  |  |
| --- | --- | --- |
| Approximate line number | Description | How |
| Parallel: 21Timestep: 25 | Set working directory | Enter the folder location on your computer where the scripts and input files are located. Output folders are generated within this folder. |
| Parallel: 25Timestep: 28 | Set input file name | Enter name of csv file in fileName with the .csv extension (e.g. ‘file.csv’) |
| Parallel: 28Timestep: 31 | Set number of cores to use in parallel computations | Set num\_cores variable. |
| Parallel: NATimestep: 36 | Set number of timesteps per year vector | Change, add, or delete list elements in Timesteps vector. |
| Parallel: 137Timestep: 112 | Figure names / plots | Change any plot component. |

**Output Results**

For each Monte Carlo replicate, the results that are scalar values are appended to the input file, and that whole file is saved to a csv file called Run#\_Results.csv, where the # sign is replaced by the integer you specify in the ‘TestID’ field of the input file. The results by time step and by year for each replicate are saved as separate csv files called Run#\_\*\_AllTimesteps.csv and Run#\_\*\_YearlyInfo.csv, respectively, where the \* represents the Monte Carlo replicate number.

If you would like to save more output from GEOPHIRES, you can add more in the GEOPHIRES\_i function at the bottom. The current values saved are only for direct use heating.

Note: the standard output file with extension .out is also saved for every replicate. These files are best viewed in a software like Notepad++. These .out files are not used in further processing, but do provide a simple summary of the results. Note that some rounding is used when writing these files, so values reported in this file may not be as accurate as the other results files provide.

**Figures and Summary Data using PlottingFunctions.R**

Histogram images (png files) are plotted in Python for all scalar value results that are currently in the output files. An R script called PlottingFunctions.R is provided to plot additional graphics of these same output values. More graphs are provided by the R script than are provided in the Python script. For example, yearly data and timestep data are plotted in the R script.

***Within the R script you have to set the working directory at the top of the script.***This working directory should contain ONLY the files for a particular TestID. This is important so that results plotted are only the results for that run, and not multiple runs. Nothing else should need to be changed for figures to write. You may want to adjust some of the plotting parameters after viewing the results.

The R script also provides summary data for all variables. Quantiles, Mean, and Standard Deviation of all yearly and timestep data are reported and summarized as Excel files. Each tab is a different variable. The first column of each tab is a template that shows what information is contained within each row of the spreadsheets (quantiles followed by mean and sd).

**Tips for Running ParallelGEOPHIRES.py**

All Python scripts and input files must be in the same directory for them to run correctly. Parallel computations will hang and not run (and no warning issued) when files are not in the same directory.

For parallel computation, if you’re using your personal computer and trying to do other tasks at the same time as this code is running, you may want to reduce the allocated number of cores. I have it set to use (number of cores – 1). Changing to –2 may be beneficial if working simultaneously and you notice lags when using -1.

If you do not want yearly or timestep summary plots and data, you can comment out that section of the code to speed up parallel processing (not recommended).

**Tips for Running PlottingFunctions.R using Python**

Currently, the ParallelGEOPHIRES.py script is set up to call a module named Rscript and run the PlottingFunctions.R script. To do this, the following are required:

1. Anaconda with Rscript module installed (default options install this)
2. Install R-essentials using Anaconda command prompt and command:

conda install R-essentials

1. Install two additional R packages that are used to run the R script.
	1. Find the directory of the R packages that are referenced by Rscript. For me on Windows 10, it is located in: C:\Users\jsmif\Anaconda2\Lib\R\library
	2. Open RStudio or R on your computer
	3. Copy the directory of a) to your computer’s clipboard.
	4. In the R or RStudio command line, type the following and hit enter:
	install.packages(c(“abind”, “dataframes2xls”), readClipboard())
	5. Now you should have those two packages within the directory in a) and the script should be callable and run in Python. An error will be issued if it doesn’t run.
	6. You may need to tell Windows to look for a Python path for the R script to work. If so, this video worked for me on Windows: <https://www.youtube.com/watch?v=Y2q_b4ugPWk>

**Warning**: With joblib version 0.12 I receive a BrokenProcessPool error when running in parallel. So, use joblib version 0.11 to run these scripts. That may mean reverting to older versions of Python (as I had to do).

If you’re unable to run PlottingFunctions.R through Python, simply run it using R in the same directory.

**Timestep Stability Analysis**

**Note**: If timestep stability analysis is run without the automatic output file directory generation, all results should be placed in a separate folder than the output from ParallelGEOPHIRES to avoid making summary plots in R with the timestep stability check data.

A useful first step before running any GEOPHIRES run is to see how many timesteps are needed to provide results that are accurate. The script TimeStepStabilityCheck.py can be used with the same input as ParallelGEOPHIRES.py to evaluate stability of the LCOH with different number of timesteps. The values used for every parameter are either the mean or most likely value (mode). Practically, the selection of these values for each parameter should not affect the stability much if the values are physically reasonable.

Within this script, you may want to change the number of timesteps per year to include other values than are currently used (see above table).

Output from the stability analysis include:

1. Plots of the LCOH versus timesteps, and runtime versus number of timesteps.
2. TimestepPointValues.csv – the values used for each of the different number of timesteps.
3. TimestepResults.csv that summarizes the scalar output from GEOPHIRES for each of the different number of timesteps evaluated.
4. YearlyInfo and AllTimesteps csv files, same as for ParallelGEOPHIRES, but the index number is now -TimestepVals instead of the Monte Carlo replicate.
For example, Run1\_-5\_YearlyInfo.csv would be the yearly output for TestID #1 run with 5 timesteps.