CaveCalc Installation & Operation Details

1. Installation

CaveCalc is a Python module that may be installed on Windows, Mac OS X and Linux systems. Installation of CaveCalc requires users to execute some simple commands using the Terminal (Mac/Linux) or Command Prompt (Windows). To install CaveCalc:

1. Install Python 3
2. Install CaveCalc module
3. Install IPhreeqc COM (Windows only)

For users that are new to using the Terminal window (Mac/Linux) or Command Prompt (Windows), a limited number of useful commands and file locations are provided in table 1.

Command	**Operating System**	**Description**
pwd	Linux/Mac	Prints the current location in the file structure.
ls	Linux/Mac	Lists the files within the current directory.
cd mydir	Linux/Mac/Windows	Changes directory, in this case into a directory called 'mydir'.
cd ..	Linux/Mac/Windows	Changes directory one level back (towards the root of the tree) in the directory structure.
help cd	Linux/Mac/Windows	Invokes the help system, in this case to learn about the 'cd' command.
cd	Windows	Displays the current drive and directory.
dir	Windows	Lists the contents of the current directory.
conda create -n py35 python=3.5	Linux/Mac/Windows	Anaconda command for installing Python 3.5 within an environment called py35
source activate py35	Linux/Mac/Windows	Anaconda command for activating the Python 3.5 environment. Within this environment the Python version is 3.5 by default
Folder	**Operating System**	**Description**
cavecalc-1.0/	Linux/Mac/Windows	location of file setup.py for installing CaveCalc with command 'python setup.py install'
cavecalc-1.0/examples	Linux/Mac/Windows	location of example files for testing the CaveCalc installation. First navigate into the folder with e.g. 'cd examples'. Then install CaveCalc with command 'python example1.py'
cavecalc-1.0/scripts	Linux/Mac/Windows	location of script cc_input_gui.py for opening the CaveCalc graphical user interface. Run command 'python cc_input_gui.py' within this folder.

Table 1: Some helpful commands and file locations for users new to Terminal (Mac/Linux) or Command Prompt (Windows) environments.

1.1. Install Python 3

Python is the programming language used by CaveCalc. It must be installed as a pre-requisite. CaveCalc has been tested and developed using Python 3.5, and is likely compatible with more recent Python 3 versions. Python 2.7 is not supported.

It is advised that users download a scientific Python distribution (e.g. Anaconda), as this will come with various dependencies (e.g. numpy, scipy & matplotlib) pre-installed. Users more familiar with Python may install Python 3 as they wish, although a virtual environment is recommended.

To check that Python has been installed correctly, open the Terminal (Mac/Linux) or Command Prompt (Windows) and enter the command:

```
python -V
```

If successful, this command will return the Python version detected. On Mac/Linux systems, the command 'python3' may be needed instead as 'python' refers to the system version (Python 2.7 by default).

1.2. Install CaveCalc

CaveCalc is distributed as a zip archive. Extract the files. This archive contains the source code for CaveCalc, scripts used to load the GUI, the PHREEQC database used for calculations and several pieces of example code. Run the following command from the extracted directory:

```
python setup.py install
```

This command will execute the CaveCalc installation script. Depending on which third party dependencies need to be installed, this may take a while.

1.3. Install IPhreeqc COM (Windows only)

CaveCalc requires an interface with PHREEQC to perform geochemical calculations. On Windows systems, this is provided by the IPhreeqc COM server, available for download from the USGS website (`http://wwwbrr.cr.usgs.gov/projects/GWC_coupled/phreeqc/`).

On Mac OSX and Linux systems, the interface with PHREEQC is handled by the phreeqpy module, installed with CaveCalc. On 64-bit Mac OSX systems and

Table 2: Additional CaveCalc input parameters. These inputs control file IO, modify model output and provide debugging options. * - R(13C) R(18O)_HCO3- R(13C)_HCO3- R(18O)_CO3-2 R(13C)_CO3-2 R(44Ca)_Calcite R(18O)_Calcite R(13C)_Calcite.

Parameter	Model Name	Default Value
Log PHREEQC input	phreeqc_log_file	FALSE
Output Directory	out_dir	
PHREEQC Database Filename	database	oxotope.dat
PHREEQC Log Filename	phreeqc_log_file_name	log_{}.phr
Totals	totals	
Molalities	molalities	HCO3- CO3-2
Isotopes	isotopes	R(44Ca) R(18O) ...*

32-bit Linux systems no further action is required. Users on other systems (e.g. 64-bit Linux) are advised to follow the phreeqpy installation instructions (available at: `http://www.phreeqpy.com/`) to compile the system-specific .dll/.so file required by phreeqpy and copy this file to the phreeqpy installation directory. It is possible to run CaveCalc using phreeqpy on Windows systems (avoiding use of the IPhreeqc COM server), however this is not advised as interaction via the COM server is more memory efficient.

To test the installation, attempt to run one of the examples included in the CaveCalc zip archive. To do this, navigate to the location cavecalc/examples and run the following:

```
python ./example1.py
```

If successful, a plot will appear.

2. Model Input

The main text describes all scientifically-relevant input options for Cavecalc. A few other options exist to handle file IO, manipulate model output and help with debugging. These are summarised in Table 2.

'Log PHREEQC input' controls whether to log commands passed from CaveCalc to IPhreeqc. If set to TRUE, CaveCalc will write a log file of PHREEQC input for each model calculation performed. The log files generated are valid PHREEQC input and may be run through PHREEQC to provide further information about a model run. This is particularly useful for debugging failed model runs, and may also help users better understand the inner workings of Cavecalc. ‘PHREEQC Log Filename’ determines the file naming convention for any log files created; {} is replaced with an arbitrary number when more than one model is run.

‘PHREEQC Database Filename’ gives the name of the PHREEQC database to be used. By default this is oxotope.dat (the database distributed with cavecalc). If another suitable database is available, its name may be given here. If the file is located outside the cavecalc/data directory, the full path should be given. ‘Output Directory’ specifies the location where model output files and any log files will be saved; if left blank the current directory is used.

‘Totals’, ‘Molalities’ and ’Isotopes’ input parameters are used to customise the chemical data returned by Cavecalc. These inputs are passed to the SELECTED_OUTPUT PHREEQC input block (see PHREEQC manual).

2.1. Graphical User Interface (GUI)

The CaveCalc GUI consists of two parts: the input GUI and output GUI. The scripts for these GUIs are included in the caveCalc archive (in cavecalc/scripts). If the Python scripts directory is on the system path, the GUIs may be run using the commands:

```
cc_input_gui.py
cc_output_gui.py
```

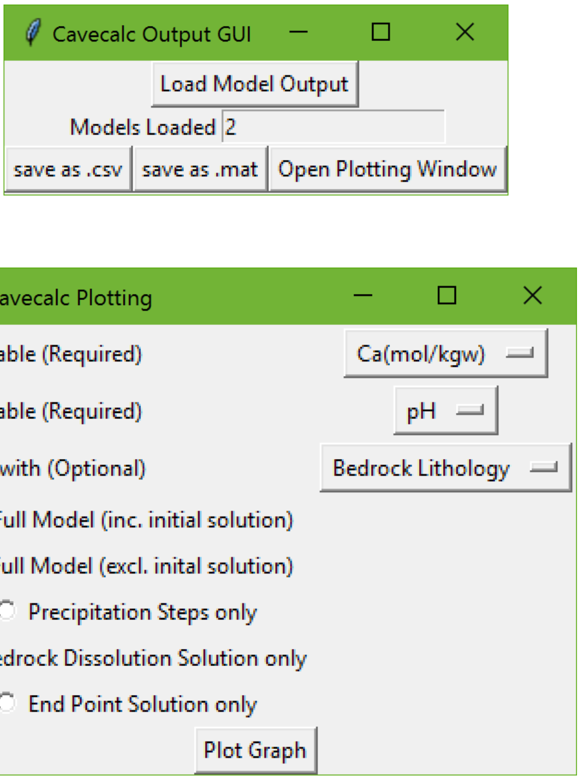
Alternatively they may be run directly from the cavecalc/scripts directory using the commands:

```
python ./cc_input_gui.py
python ./cc_output_gui.py
```

The input GUI provides an interface for CaveCalc inputs. For many inputs, clicking the asterisk allows users to quickly generate linearly spaced numerical arrays of multiple input values — useful when you want to run multiple models with systematic variations in one (or more) inputs. If more than one value is given for any parameter, CaveCalc will automatically run a series of models — one for each unique combination of input parameters. When all parameters are set as desired, click 'Run' to run the models. Runs may take some time, depending on the inputs chosen, the Run Mode and the number of models to be calculated. When complete, output files (settings.pkl and results.pkl) will be saved to the specified output directory. These files may be accessed using the Python pickle library, or via the output GUI. The 'Load Output GUI' button provides a shortcut to launch the output GUI.

The output GUI (Figure 1) provides a few simple tools to help visualise and convert the output data between formats. To use, click 'Load Model Output' and select a directory that contains model output (settings.pkl and results.pkl). Once loaded, the number of model runs loaded is displayed in the window; a single directory may contain multiple model runs. Multiple directories may be selected in sequence is you wish to combine their output or visualise them together. Loaded model output may be saved for use in MATLAB or spreadsheet programmes (e.g. Excel) by clicking the 'save as .mat' and 'save as .csv' buttons, respectively. If .csv files are requested, one file will be saved for each model run. If .mat is requested, a single file will be saved containing all settings and results data. Files are saved in the current directory.

The output GUI also contains options for plotting models. This tool is intended to allow quick visualisation of model output — it is not intended to provide a fully functional plotting programme! To use, once data have been loaded, click 'Open Plotting Window' (Figure 1). Here you may select model output variables X and Y to be plotted, along with an optional third parameter to label them with. The radio buttons allow users to select which model steps should be plotted.



Models Loaded 2

X variable (Required) Ca(mol/kgw)

Y variable (Required) pH

Label with (Optional) Bedrock Lithology

( ) Full Model (inc. initial solution)

[✓ Full Model (excl. inital solution)]

( ) Precipitation Steps only

( ) Bedrock Dissolution Solution only

( ) End Point Solution only

Figure 1: CaveCalc output GUI. Left: main window. Right: plotting window.

2.2. Python API

All functionality exposed through the GUI, and more, is available via the Python API. Various examples of how to use CaveCalc within Python are given in the cavecalc/examples directory. In particular, Example 4 demonstrates how users may define their own series of reactions without relying on using a pre-defined Run Mode.

Users interested in fully understanding the workings and capabilities of CaveCalc are advised to read the documented source code. A brief description of each file and the code it contains is given here.

caves.py This is the main file. It includes all code for running a single model, including geochemical reactions and interaction with IPhreeqc. Notably, the Solution class provides the API for all available geochemical reactions used by the Run Modes. The Simulator class co-ordinates all interactions with IPhreeqc. Users who wish to understand the operation of CaveCalc are advised to begin by reading the Simulator.run() method. The Simulator.run() method is the starting point for all model runs that use a Run Mode; it demonstrates the high level scripting of a CaveCalc model.

forward_models.py This file includes the ForwardModels class, which is used to handle the sequential running of multiple models and handle their output. This class provides a clean interface to caves.py that is often used in preference to interacting with caves.py directly; the main exception to this is when running models without a Run Mode (e.g. Example 4).

analyse.py This file contains all code for handling model output files, including plotting, filtering and converting file types. It is not an essential component of the model but may be useful. Some useage of analyse.py is demonstrated in the examples provided.

setter.py This file contains code for handling of model input parameters: auto-population of default values, checking entries for type or value errors, and generating suites of model runs where applicable.

util.py This file contains utility code to help with a variety of tasks. This includes code for parsing the PHREEQC database, saving log files and performing common unit conversions.

data/... This subdirectory contains all static data used by Cavecalc. This includes the PHREEQC database, default and allowed values for model input parameters and various template strings used by caves.py to construct input strings for IPhreeqc.

gui/... This subdirectory includes all code for the GUI.