# CaveCalcV2.0 - Manual

**Note yet complete! So any queries do not hesitate to reach out at samuelhollowood.bnc.ox.ac.uk**

## Contents

# 1 Installation

Step-by-step code for installation can be found in the CaveCalcV2.0 GitHub Repository (https://github.com/Samhollowood/CaveCalcV2.0/tree/main). Here, I outline a more thorough installation guide. CaveCalcV2.0 is a Python module that can be installed on Windows, Mac OS X, and Linux systems. Installation of CaveCalcV2.0 requires users to execute some simple commands using the terminal (Mac/Linux) or Command Prompt (Windows). To install CaveCalcV2.0:

1. Install Python 3
2. Install CaveCalcV2.0 module

Notice there is now no need to install IPhreeqc COM for Windows users. This configuration of the IPhreeqc COM server is now done automatically when running the setup.py file (see section 1.2).

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 the original manual (https://github.com/Rob-Owen/cavecalc) in Table 1.

## 1.1 Install Python 3

Python is the programming language used by CaveCalc. It must be installed as a prerequisite. Python 2.7 (the pre-installed default on most systems) is not supported. CaveCalcV2.0 has been tested on Python 3.10.9, and is likely compatible with any version above 3. Python can be downloaded from https://www.python.org/downloads.

It is recommended to use Python with Anaconda (https://www.anaconda.com/download). Anaconda provides an integrated development environment (IDE) like Spyder, which makes it easier to run and debug models. For users who prefer not to install Anaconda, Python can still be used directly, as the dependencies are installed as part of the setup file.

To check the Python version, run the following command in the Terminal (Mac/Linux) or Command Prompt (Windows):

```
python -V
```

On some Mac/Linux systems, the command `python` may refer to Python 2.x. Use `python3` to ensure Python 3.x is invoked.

If you encounter any issues:
- Ensure Python is installed and added to your PATH
- Confirm that you are using Python 3.x (not Python 2.x).

## 1.2 Installing CaveCalcV2.0 module

To download the source code for CaveCalcV2.0, you have two options:

1. **Using Git**

(requires Git to be installed on your system). If Git is not installed, you can download it from https://git-scm.com. Open a terminal (Mac/Linux) or command prompt (Windows) and run:

```
git clone https://github.com/Samhollowood/CaveCalcV2.0.git
```

2. **Downloading as a ZIP File**

Go to the CaveCalcV2.0 GitHub repository and click the green 'Code' button, then select 'Download ZIP'. Extract the ZIP file to your desired directory.

Once you have downloaded the source code, you need to navigate to the directory containing the CaveCalcV2.0 folder. If you used Git or extracted the ZIP file, you can navigate to the directory using the following command:

```
cd CaveCalcV2.0
```

This goes to the working directory of the CaveCalcV2.0 folder, where you can proceed with the installation steps. If you are using Anaconda, it is recommended to activate your environment before running the installation script. To activate the base environment, run:

```
conda activate base
```

This ensures that CaveCalcV2.0 and its dependencies are installed in your Anaconda base environment. If you prefer to install it into a local anaconda/python environment, you can create one using:

```
conda create -n cavecalc_env
conda activate cavecalc_env
```

or if you do not have Anaconda:

```
python3 -m venv cavecalc_env
source cavecalc_env/bin/activate
```

Then, to install the CaveCalcV2.0 module, run the following command to install CaveCalcV2.0 and its dependencies:

```
python setup.py install
```

This command runs the CaveCalc installation script, which installs the necessary dependencies (e.g., NumPy, SciPy, Matplotlib) and makes CaveCalcV2.0 available as an executable package.

## 1.3 Testing Installation

After completing the installation, it is recommended to verify that CaveCalcV2.0 has been installed successfully. Navigate to the examples directory within the CaveCalcV2.0 folder:

```
cd CaveCalcV2.0/examples
```

Then, run the following command to execute a sample script:

```
python example1.py
```

If the script runs without errors and produces the expected output, the installation is complete. Any errors/debugging, please contact samuel.hollowood@bnc.ox.ac.uk.

## 1.4 Accessing the Graphical User Interface (GUI)

To launch the CaveCalcV2.0 graphical user interface (GUI), navigate to the scripts directory within the CaveCalcV2.0 folder:

```
cd CaveCalcV2.0/scripts
```

Then, run the following command:

```
python cc_input_gui.py
```

This will open the CaveCalcV2.0 GUI, where you can interact with the software's features through a user-friendly interface. Here, users can define inputs to run CaveCalcV2.0 models, along with the new Carbonate Data Analyser (CDA) mode.

# 2 Model Inputs

Once installation is complete, users can begin running forward models. Due to the large number of model inputs, it may be challenging for new users to determine which parameters are most relevant. To address this, updates have been made to improve awareness of each input's impact on speleothem chemistry, as well as the corresponding model name (see Table 3). Before running models, there should be a few things that may be of use to the user.

**Important Notes**

- **Atmospheric End-Member Inputs:** These are only applicable if atmospheric exchange is set to a value greater than 0. In the GUI, this parameter can be found under the *Mix Gas* drop-down heading as the input *Amtospheric Gas Fraction (0–1)*. Via the Python API, the model name is `atmo_exchange`.
- **Trace Elements:** If you decide to include these, please define them in either the soil (mmol/kgw) or bedrock (mmol/mol). Otherwise, default values are set to 0.
- **Dolomite Override:** If you set the `bedrock_mineral` to Dolomite, it will override your `bedrock_MgCa` value and set it to 1000 mmol/mol.

Here, I outline how to run forward models either in the GUI or via the Python API.

## 2.1 Graphical User Interface (GUI)

1. **General:**
    - Open the GUI (see section 1.4). The GUI should look something like Figure 1.

- Define model inputs. Inputs are now organized within drop-down headings. Clicking on a heading reveals all inputs associated with that category.
- Hover over an input to see its impact on speleothem chemistry.
- All model inputs and realistic values can be found in Table 3

2. **Multiple values for a single input variable:**

   - Click the asterisks (`*`) next to certain inputs can generate linearly spaced arrays of input values, allowing you to run multiple models (Figure 2).

3. **Define Output Directory:**

   - In the bottom left of the GUI (Figure 1), you are able to chose an output directory
   - If no output directory is specified, one will be created automatically as `cavecalc_output` in your working directory.

4. **Run Model:**

   - Click the `Run CaveCalc Only!` button to initiate simulations. Model progress is displayed in the console.

5. **Outputs:**

   - Three output files are generated within the output directory:
     - `settings.pkl` and `results.pkl` (standard CaveCalc outputs).
     - `settings_and_results.csv` (a new .csv file, storing all settings and results from the .pkl files and translating it into a single, readable format).
     - Model output keys can be found in Table 4.

6. **Access Output GUI for Post Processing:**

   - You can access the output via the `Load Output GUI` button in the GUI, or by running the following command:

```
python cc_output_gui.py
```

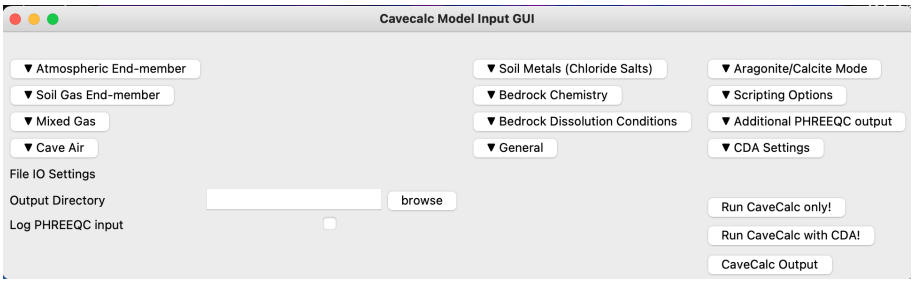


Figure 1: CaveCalcV2.0 GUI. Inputs are grouped under drop-down headings for ease of navigation.

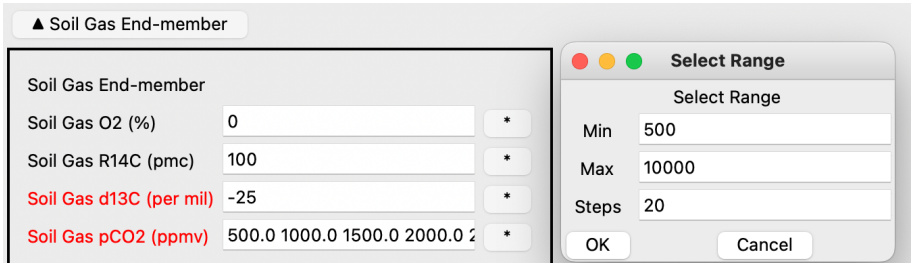


Figure 2: Example of varying an input in steps. This functionality facilitates the generation of multiple models.

## 2.2 Python API

A new folder, `CaveCalcV2.0/API_Model`, contains Python scripts (`.py` files) that allow users to run models through an IDE environment such as Spyder or Jupyter Notebooks. Among these scripts, `run_models.py` enables users to execute forward models by defining inputs in a settings dictionary `s = {}`. For further guidance on the model inputs, refer to Table 3, which provides information on all model names, realistic input values, and their impact on speleothem chemistry.

Any input can be added, modified, or removed based on the requirements of the study. If an input is removed, the model will use the default value. A list of the default values can be found in `./CaveCalcV2.0/cavecalc/data/defaults.py`.

The general framework of the `run_models.py` script is as follows:

```
# Import modules
from cavecalc.forward_models import ForwardModels

```

```
# STEP 1: Define the non-default settings and run models
s = {# Define input values (see Table 2).
}

# STEP 2: Run forward models
model = ForwardModels(settings=s)
model.run_models()
model.save()
```

In the `run_models.py` script within the source code, the settings dictionary (`s = {}`) is already populated and all available model names are explicitly defined as their default values. This is apart from the soil_pCO2, which is defined as multiple values in a list, showing the user that some inputs can be defined as multiple values. Advanced users may want to create their own scripts for more flexibility, using `run_models.py` as a template. Running this script will then run the forward models with progress in the console.

## 2.3 Simulating aragonite precipitation

In the **GUI**, users can define the precipitate mineralogy through the drop-down heading `Aragonite/Calcite Mode` (Fig. 1). This option allows the user to choose either `Calcite` or `Aragonite` as the precipitating mineral.

**Note**

There is no need to explicitly define the database for this functionality. Defining the precipitate mineralogy will feed into CaveCalcV2.0 that a certain database needs loading.

Alternatively, users can simulate mineralogy directly via the API by including the model_name 'precipitate_mineralogy' in the settings dictionary. The input can be set to either '`Calcite`' or '`Aragonite`', depending on the desired simulation:

```
# Import modules
from cavecalc.forward_models import ForwardModels

# STEP 1: Define the non-default settings and run models
s = {  # Define input values (see Table 2).
    'precipitate_mineralogy': 'Aragonite',
}

```

```
# STEP 2: Run forward models
model = ForwardModels(settings=s)
model.run_models()
model.save()
```

**Output Variations:** The output keys will vary slightly depending on the selected precipitate mineralogy (see Table 4). Specifically:

- If `Calcite` is simulated, the output key for a trace element `X` will be `X/Ca(mol/mol)_Calcite`.
- If `Aragonite` is simulated, the output key for the same trace element will be `X/Ca(mol/mol)_Aragonite`.

Additionally, the outputs for `d18O`, `d13C`, and `d44Ca` will vary based on the selected mineralogy:

- For `Aragonite` simulation, the output will include: `d18O_Aragonite`, `d13_Aragonite`, and `d44Ca_Aragonite`.
- For `Calcite` simulation, the respective outputs will use the `Calcite` suffix (e.g., `d18O_Calcite`).

# 3 Model Outputs

## 3.1 File outputs

The new `settings_and_results.csv` file offers a more user-friendly format compared to the original `.pkl` files in CaveCalc. This file combines the data from `settings.pkl` and `results.pkl` into a single, comprehensive table. Users can easily view input settings alongside corresponding model outputs, including results for each step of speleothem chemistry per model (e.g., soil-water equilibriation, bedrock dissolution and $CO_2$-degassing).

For advanced users who prefer working with pickle files, the original `settings.pkl` and `results.pkl` files are still available for custom analyses.

Additionally, there are options to convert the output pickle files into individual .csv (one per model output) and .mat files for post-processing.

**Note**

These .csv files differ from `settings_and_results.csv`. In these files, each output has its own .csv and only stores the output values (not input settings).

Conversion of .pkl files can be done in the Output GUI, after loading the data from the output directory. Here is an example of converting to a .csv and .mat file via scripting below:

```
# Import modules
import cavecalc.analyse as cca

# Initialize the Evaluate class
e = cca.Evaluate()
out_dir = 'path/to/output'  # Define the output directory
e.load_data(out_dir)  # Load model data from the output directory

# Save in different formats
e.save_csvs(out_dir)
e.save_mat(out_dir)
```

## 3.2 Plotting Model Results

CaveCalc offers two types of plots for visualizing model results:

- **Plot_models:** Displays model results, with a line representing each model.
- **Plot_points:** Generates scatter points at specific indices for selected outputs.

### 3.2.1 Plotting via the GUI

Plots can be generated via the Output GUI (Figure 3). On the Output GUI, load the data by clicking `Load Data` and then browse to the output directory in which the `settings.pkl` and `results.pkl` are stored. The number of models loaded should then be displayed.

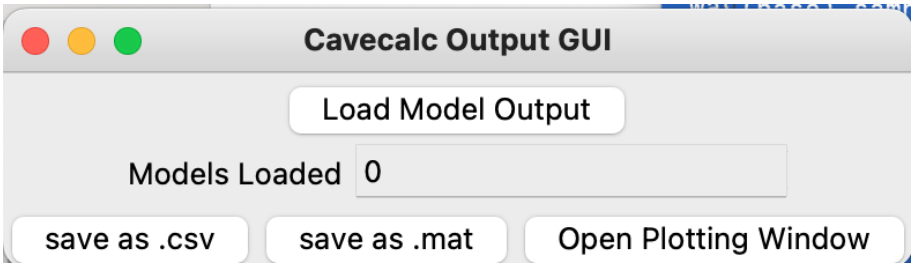


Figure 3: CaveCalcV2.0 Output GUI.

Then clicking `Open Plotting Window` will open the plotting functionality. The plotting window should look like Figure 4.

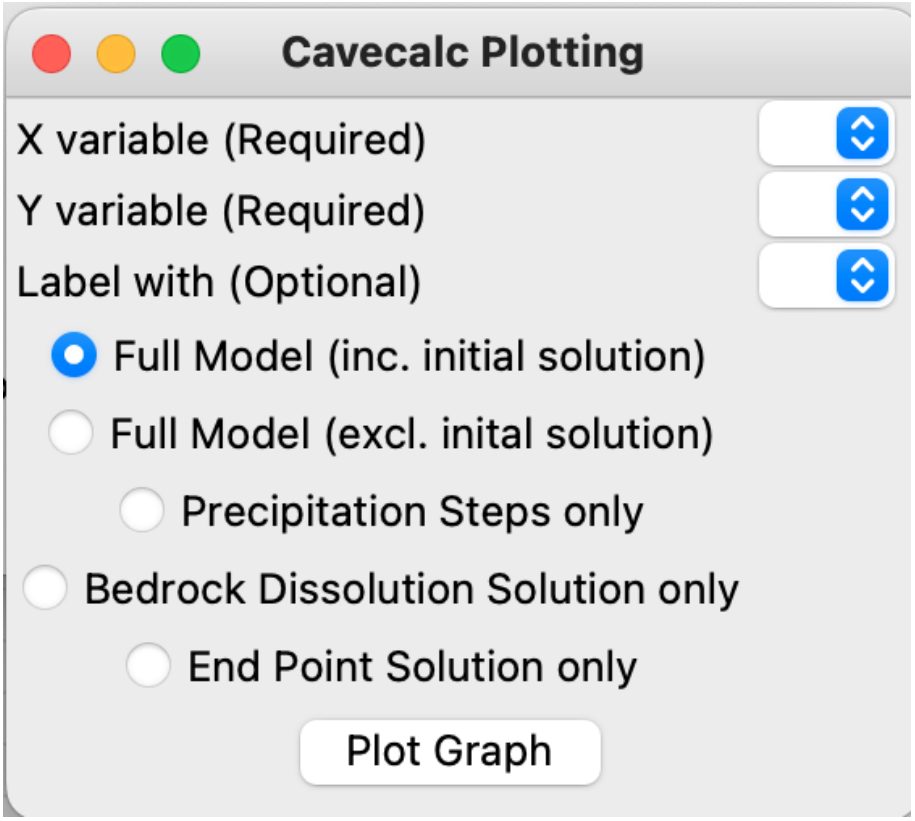


Figure 4: CaveCalcV2.0 Plotting Window

Users can then click the drop-down to plot model outputs as X and Y variables. Users also have the option to include the full model, including or excluding the initial solution, the precipitation steps (i.e. Prior Carbonate Precipitation, the bedrock dissolution value, or the equilibrium end-point value). The label with provides an option to add an input setting, determining how different models are labelled in the plot.

### 3.2.2 Plotting via Python API

The following example demonstrates how to plot model outputs using Python scripting:

```python
# Import modules
import cavecalc.analyse as cca

# Initialize the Evaluate class
e = cca.Evaluate()
out_dir = 'path/to/output'  #Scan to output directory in which models are stored
e.load_data(out_dir)  # Load model data from the output directory

# Example 1: Plot model results (e.g., degassing slope against calcite d13C,
# with each line representing a different soil d13C value)
plot = e.plot_models(x_key='f_ca', y_key='d13C_Calcite',
    label_with='soil_d13C')

# Example 2: Plot scatter points (e.g., equilibrium value of f_ca vs calcite
    d13C)
plot_2 = e.plot_points(x_key='f_ca', y_key='d13C_Calcite',
                       plot_index=-1, label_with='soil_d13C')
```

The `x_key` and `y_key` parameters define the output variables plotted on the x-axis and y-axis, respectively. The `label_with` parameter determines which input setting is being used to distinguish the different models in the plot.

- `x_key`: Specifies the variable for the x-axis. Example: `x_key='f_ca'` sets the x-axis to the fraction of calcium remaining in the solution. Changing this value (e.g., `x_key='pH'`) will plot pH instead of `f_ca` on the x-axis.
- `y_key`: Specifies the variable for the y-axis. Example: `y_key='d13C_Calcite'` plots calcite d13C on the y-axis. Changing this to `y_key='Mg/Ca(mol/mol)_Calcite'` would plot the calcite Mg/Ca instead.
- `label_with`: Determines how different models are labelled in the plot. Example: `label_with='soil_d13C'` means each line in `plot_models` corresponds to a different model input `soil_d13C` value. Changing this to `label_with='temperature'` would distinguish models by the model input temperature instead.

A comprehensive guide to all model output keys is provided in Table 4. Users can modify these parameters to visualize different model relationships and trends.

The `plot_index` parameter determines which stage of the speleothem chemistry process is visualized:

- `plot_index=0`: Solution after soil water equilibration.
- `plot_index=1`: Solution after bedrock dissolution.

- `plot_index=-1`: Final equilibrium values for solid and solution phases.

This feature allows users to analyze both intermediate and final model outputs.

# 4 The Carbonate Data Analyser (CDA)

The only difference between the CDA and standard CaveCalcV2.0 model runs is that users must import their measured data and optionally define tolerance intervals for each proxy inside the measured data. The operating procedure can be found in the manuscript ().

## 4.1 User's Measured Data

To use the CDA, users need to import their measured speleothem data in a compatible format. An example `.csv` file is provided in the supplementary information of the manuscript or is available directly in the CaveCalcV2.0 GitHub Repository.

The file should contain the following columns:

- **Age:** The age of the sample in years or kyr BP.
- **Measured Data:** Measured values such as $\delta^{18}O$, $\delta^{13}C$, or other isotopic/elemental values.

Isotopic values should be defined as ‰, with a specific reference standard. It is recommended to provide measured data using $\delta^{13}C$ (‰, VPDB) and $\delta^{18}O$ (‰, VPDB), as this is the most popular reference standard. $\delta^{44}Ca$ can be reported relative to NIST SRM915a or BSE, provided the reference standard is consistent when placed in model inputs. Elemental values should be stored as mmol/mol.

**Important Note for $\delta^{18}$O**

- The original CaveCalcV2.0 reported the carbonate output of $\delta^{18}O$ relative to **VSMOW**. However, **CaveCalcV2.0** now reports it as the most common reference standard: **VPDB**. Therefore, the measured $\delta^{18}O$ should be in **VPDB**. Despite this, the model input `atm_d18O` should remain relative to **VSMOW** as this is this most common reference standard for water.
- **CaveCalcV2.0** now handles the conversion of VSMOW to VPDB

The provided `.csv` file serves as a template to organize your data in the required format. Columns can be added or removed depending on the number of proxies you wish to incorporate. Users have complete flexibility regarding the number and combination of proxies included. The CDA can process any selection of proxies, so you are not required to include all possible proxies. Simply include the ones relevant to your study.

**Note**

- Providing a high-resolution dataset will increase the processing time of the CDA.
- High-resolution data may distort the automated plotted output, but will still provide tabular outputs for independent post-processing.

## 4.2 Model inputs

Here, I outline how to run the CDA with CaveCalcV2.0, either via the GUI (see section 4.2.1) or Python API (see section 4.2.2).

**Important Notes**

- If you decide to include trace elements or calcium isotopes, please define them in either the soil (mmol/kgw) or bedrock (mmol/mol). Otherwise, default values are set to 0. If trace elements inputs are 0 mmol/mol, their carbonate value will be 0.
- Once you have ran the CDA under a batch of model runs, adding another batch of runs will append the results of the CDA, not replace, provided the output directory is left unchanged between runs.

### 4.2.1 GUI

It may be difficult to narrow down the input values you wish to investigate for the CDA. As general guidance, environmental model inputs highlighted in red in the GUI denote their high influence on speleothem chemistry and are the inputs within the default plots that are generated. These inputs would be a good place to start. With time and more experience, users may wish to test the CDA with different inputs (e.g. including inputs to test pyrite oxidation). All model inputs, and their impact on speleothem chemistry, can be found in Table 3.

The required inputs for the CDA are located under the `CDA Settings` field (Fig. 5). Here, you can browse for the file path to your measured speleothem data. Optionally, you may also define tolerance intervals for the proxies within the measured data.

**Note**

Tolerance intervals determine whether a match occurs. If

$$|\text{measured proxy} - \text{CaveCalc proxy}| < \text{tolerance interval}$$

it is a match. If no tolerance intervals are specified, default values are applied.

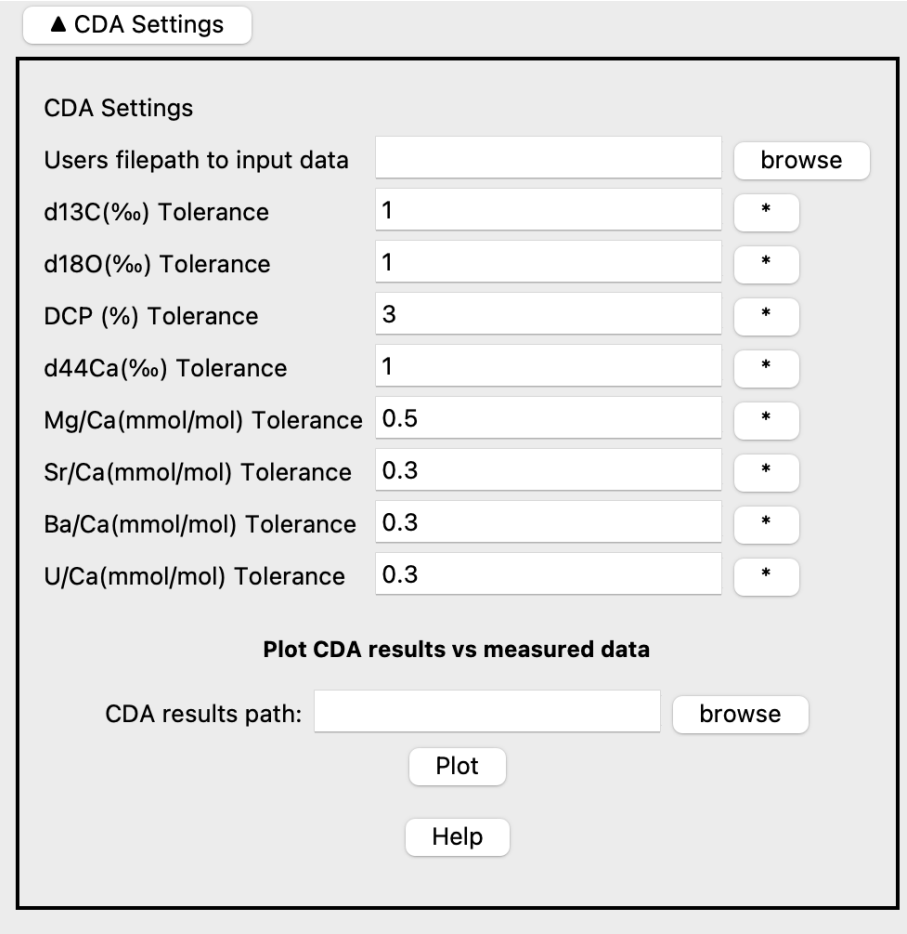


Figure 5: CaveCalcV2.0 GUI

To initialize a CDA run:

1. Define the necessary model inputs under the other headings on the GUI.
2. Under the **CDA Settings** field, browse to the file path of your measured speleothem data.
3. Optionally, define tolerance intervals for proxies within the measured data.
4. Click the **Run CaveCalc with CDA!** button.

If successful, the terminal (Mac/Linux) or command prompt (Windows) will display:

```
CDA is initialized
```

The output will be stored in a folder named `CDA_Results`, created within the output directory. When the first match with the measured data is found, the following will be displayed:

```
Created new file path/to/out_dir/CDA_Results/Matches.
    csv and saved results.
```

After the model runs are complete:

- If there are matches, the CDA will generate plots (see Section 4.3).
- Tabular outputs will also be stored in the `CDA_Results` in which the data for the plots (and other inputs/outputs) are stored

### 4.2.2 Python API

The CaveCalcV2.0 GitHub directory includes an `API_model` folder containing a `run_CDA.py` script for running the CDA. The main difference between a standard CaveCalc run and a CDA run is the need to define the path to your measured data (`user_filepath`) and optionally define tolerance intervals for proxies (e.g., `tolerance_d13C`, `tolerance_d18O`, `tolerance_d44Ca`).

Below is an example framework for running the CDA:

```python
# Import modules
from cavecalc.forward_models import ForwardModels
import cavecalc.analyse as cca

# STEP 1: Define the non-default settings and run models
settings = {
    # Environmental model inputs
    'soil_d13C': [-11, -18, -25],
    'soil_pCO2': [260, 600, 1000, 2000, 5000, 8000],
    'gas_volume': [0, 100, 200, 300, 400, 500],
    'bedrock_d13C': 0,
    'temperature': [5, 10, 15],
    'cave_pCO2': [260, 800, 4500],
    'precipitate_mineralogy': 'Calcite',

    # Path to measured data
    'user_filepath': 'path/to/data.csv',  # Define for CDA
```

```
    # Optional tolerance intervals
    'tolerance_d13C': 0.5,
    'tolerance_d18O': 0.5,
    'tolerance_d44Ca': 0.5,
    'tolerance_MgCa': 0.3,
    'tolerance_SrCa': 0.3,
    'tolerance_BaCa': 0.3,
    'tolerance_UCa': 0.3,
}

# STEP 2: Run CDA models
CDA_model = ForwardModels(settings=settings)
CDA_model.run_models()
CDA_model.save()
```

In this example, the settings include parameters such as `soil_d13C`, `soil_pCO2`, which are plotted in the default selection of figures for the CDA (see section 4.3). These are also the inputs which are highlighted in red within the GUI. Users can modify these or add other parameters as needed. A full list of model inputs and their realistic ranges is provided in Table 3.

**Note:** Remove tolerance intervals for proxies not included in your measured data.

## 4.3 CDA Outputs and Plotting

The CDA creates a folder within the model output directory named `CDA_Results`. This folder contains the following files:

`Matches.csv` A CSV file that stores the inputs, chosen outputs, and residuals of all matches with the measured data.

`All_outputs.csv` A CSV file that stores all inputs, chosen outputs, and residuals from the CDA model runs, regardless of a match.

`Tolerance.csv` A CSV file that stores the tolerance intervals used in the CDA.

`Input_ranges.csv` A CSV file that stores the range of model inputs used in the CDA.

The chosen model outputs in the Matches.csv and All_outputs are fCa (the fraction of calcium remaining in solution), $d13C_{initial\_solution}$, and $Ca^{2+}{}_{initial\_solution}$. These are added due to their significant enhancement of the environmental interpretations that can be drawn after running the CDA (Table 1).

| Parameter | Description |
|---|---|
| fCa | An indicator for the amount of Prior Carbonate Precipitation (PCarbP) along the flow-path. Lower values are indicative of a greater amount of PCP/PAP. |
| $d13C_{initial\_solution}$ | The DIC d13C value directly after bedrock dissolution and has suffered no alteration from degassing. Can be used to isolate the impact of the soil and flow-path on d13C. |
| $Ca^{2+}{}_{initial\_solution}$ | The concentration of Ca immediately after bedrock dissolution. Can be used as an indicator for the extent of bedrock dissolution. |

Table 1: Description of Chosen Model Outputs in the Matches.csv and All_outputs.csv

We are open to community feedback to provide additional outputs that could be stored within the `Matches.csv` and `All_outputs.csv` files. The definition of all model outputs is found in Table 4.

Plots are generated automatically after completing all model runs with the CDA. The aim of these plots is to aid the environmental interpretation, and the code for these plots is based on the .csv files in the CDA_Results folder. The plots only contain a sub-section of model inputs and outputs (i.e. the variables highlighted in red in the GUI (Figure 5). This means that there are certain model inputs that are not part of the default plot but will still be archived in the tabular files `Matches.csv` and `All_outputs.csv`.

**Important Note**

The CDA plotting feature does not plot all input variables, only a selected amount. Nonetheless, the Matches.csv file stores every input variable.

Plots can also be generated manually, using either the GUI or the Python API.
To plot CDA results using the GUI:

1. Open the **CDA Settings** field
2. Navigate to your .csv file with speleothem measured data under user filepath
3. Under plot CDA vs measured data, navigate to the `CDA_Results` folder by browsing to the archived output directory.
4. Click Plot

The code for plotting CDA results can also be executed using the Python API. Below is an example script:

```python
# Import modules
import cavecalc.analyse as cca

# Initialize the Evaluate class
e = cca.Evaluate()

# Define path to measured data and archived output directory
user_filepath = 'path/to/data.csv'  # Path to the measured data used in the CDA run
out_dir = 'path/to/out_dir'         # Path to the archived output directory containing CDA results

# Plot CDA results
plot = e.plot_CDA(user_filepath, out_dir)
```

This script requires the `path/to/data.csv`, which contains the measured data used in the CDA run, and the `path/to/out_dir`, which contains the path to the CDA_results.

# 5 Advanced manual

## 5.1 Editing database files

In order to edit the database files, the user will need to know the location of the files. Run the following in the terminal (MacOS/Linux) or command prompt (Windows):

```
pip show cavecalc
```

This gives the location (path) where CaveCalcV2.0 was installed. The database files are then found under cavecalc/data within this location. They can be edited here.

Alternatively, they may be edited in the locally downloaded source code, and copied over to the installation location. After editing, run in the terminal (MacOS/Linux) or command prompt (Windows):

```
sudo cp /path/to/cavecalc/data/calcite.dat /path/to/
    cavecalc/data/aragonite.dat "$(python3 -m pip show
    cavecalc | awk '/Location/ {print $2}')/cavecalc/
    data/"
```

Where /path/to/cavecalc/data/calcite.dat and/or /path/to/cavecalc/data/aragonite.dat are the paths to these database files in the locally downloaded source code.

With this in, the following sections provide a guide on how to edit fractionation factors and partition coefficients within these databases. The databases can be edited with any text editor. It is recommended to make a copy of the original version so you can always refer back to it if need be.

## 5.2 Changing stable oxygen fractionation factors within the database files

In the calcite.dat database, the stable oxygen fractionation factor between the precipitate and water is defined as:

```
Log_alpha_18O_Calcite/H2O(l)           # 1000ln(alpha(25C)) = 28.05
# Kim and O'Neil (1997)
    -ln_alpha1000    -32.42     0   18.03e3
```

and for the aragonite.dat:

```
Log_alpha_18O_Aragonite/H2O(l)           # 1000ln(alpha(25C)) = 28.05
    # Kim et al., (2007)
    -ln_alpha1000    -31.14     0   17.88e3
```

In both cases, the fractionation factor is given as $1000\ln(\alpha_{CaCO3-H2O})$) as a function of temperature in Kelvin (K). It is the same as writing:

$$1000ln(\alpha_{CaCO3-H2O}) = 18.03e3/T(K) - 32.42 \quad (1)$$

For the calcite.dat, where T is the temperature in Kelvin. The databases are large, and the relevant fractionation factors and partition coefficients can be found via a search within the document:

```
Log_alpha_18O_Calcite/H2O(l)   Log_alpha_18O_Aragonite/H2O(l)
```

## 5.3 Changing D(X) and $\alpha_{44/40}$ within the database files

Users may want to use their own D(X) and $\alpha_{44/40}$ values that are more applicable to their cave system or study (e.g. values derived from drip sites within caves).

Navigate to the database (cavecalc/data/..), and open them with a text editor (or any app with text editor capabilities). The databases are large, and the relevant fractionation factors and partition coefficients can be found via a search within the document:

```
Log_alpha_MgCa_Calcite/H2O(l) or Log_alpha_MgCa_Aragonite/H2O(l)
Log_alpha_SrCa_Calcite/H2O(l) or Log_alpha_SrCa_Aragonite/H2O(l)
Log_alpha_BaCa_Calcite/H2O(l) or Log_alpha_BaCa_Aragonite/H2O(l)
Log_alpha_SrCa_Calcite/H2O(l) or Log_alpha_UCa_Aragonite/H2O(l)
Log_alpha_44Ca_Calcite/Ca(aq) or Log_alpha_44Ca_Aragonite/Ca(aq)
```

Under these variables, you will find the reference and the temperature-dependent (in kelvin) relationship with each trace element (existing as 1000*ln(X/Ca)):

$$-ln\_1000(alpha) = C + xT(K) \tag{2}$$

Where C is the constant in the relationship (the first number after the -ln_1000(alpha)), and x is the slope (the second number after the -ln_1000(alpha)). Ca fractionation factors have a single value because there is not yet an established temperature-dependent relationship. Users may define their own temperature dependence or replace an original D(X) with a single value. For example, a D(X) of 3 would be 1000*ln(D(X)) = 1098.612 within the database file, which would look like:

```
Log_alpha_XCaCalcite/H2O(l) or Log_alpha_XCa_Aragonite/H2O(l)
       #D(X) = 3 in 1000*ln(D(X)) form
       -ln_alpha1000   1098.612
```

Note that the PHREEQC databases only deal with temperature dependence, and there is currently no way to incorporate a growth rate dependence among other factors.

## 5.4 Non-environmental model inputs

A few input 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 V2.0. '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 calcite.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).

| Parameter | Model | Default Value |
|---|---|---|
| Log PHREEQC Input | phreeqc_log_file | FALSE |
| Output Directory | out_dir | " |
| PHREEQC Database Filename | database | calcite.dat |
| PHREEQC Log Filename | phreeqc_log_file_name | log_phr |
| Totals | totals | |
| Molalities | molalities | HCO3- CO3-2 |
| Isotopes | isotopes | R(44Ca) R(18O) ... |

Table 2: Other Inputs

# 6 Tables

| **Model Name** | **Realistic Range of Values** | **Impact on speleothem chemistry** |
|---|---|---|
| Amtospheric Gas end-member | | |
| atm_O2 | 0.21 (%) | If the atmosphere and soil is exchanged, this value impacts gaseous oxygen in the soil, with implications for pyrite oxidation. |
| atm_d18O | -25 to 5 (‰, VSMOW) | The oxygen isotopic composition of rainwater. Impacts . speleothem d18O (‰, VPDB |

| | | |
|---|---|---|
| atm_pCO2 | 150 to 480 ppmv | The atmospheric CO2 concentration. If the atmosphere and soil is exchanged, this value impacts gaseous CO2 in the soil. |
| atm_d13C | -9 to -6 (‰, VPDB) | The carbon isotopic composition of the atmosphere. If the atmosphere and soil is exchanged, this value impacts the carbon isotopic composition in the soil. |
| atm_14C | IntCal | The radiocarbon activity in the atmosphere. If the atmosphere and soil is exchanged, this impact speleothem DCP. |
| Soil Gas end-member | | |
| soil_O2 | 0-0.21 (%) | The O2 percentage in the soil, with implications for pyrite oxidation |
| soil_R14C | less than or equal to atm_R14C | The radiocarbon activity in the soil. Impacts speleothem DCP. |
| soil_d13C | -30 to -5 (‰) | The carbon isotopic composition of the soil-gas $pCO_2$. Impacts speleothem d13C. |
| soil_pCO2 | 260 to 10,000 (ppmv) | The soil-gas CO2 concentration. Impacts amount of bedrock dissolution. and degassing slope, including the amount of prior carbonate precipitation. Impacts speleothem d13C, d44Ca, DCP and X/Ca |
| Atmospheric and soil gas mixing | | |

| | | |
|---|---|---|
| atmo_exchange | 0 to 1 | Sets the atmospheric contribution (see atmospheric gas end-members) to the soil |

| Model Name | Realistic Range of Values | Impact on speleothem chemistry |
|---|---|---|
| Soil metals | | |
| soil_Ba | Site-specific (mol/kgw) | Sets the terrigenous input of Barium |
| soil_Ca | Site-specific (mol/kgw) | Sets the terrigenous input of Calcium |
| soil_Mg | Site-specific (mol/kgw) | Sets the terrigenous input of Magnesium |
| soil_Sr | Site-specific (mol/kgw) | Sets the terrigenous input of Strontium |
| soil_U | Site-specific (mol/kgw) | Sets the terrigenous input of Uranium |
| soil_d44Ca | $\sim$0 | Sets the calcium isotopic composition of terrigenous inputs |
| Bedrock Chemistry | | |
| bedrock_BaCa | Site-specific (mmol/mol) | Sets the Ba/Ca of the bedrock |
| bedrock_MgCa | Site-specific (mmol/mol) | Sets the Mg/Ca of the bedrock |
| bedrock_SrCa | Site-specific (mmol/mol) | Sets the Sr/Ca of the bedrock |
| bedrock_UCa | Site-specific (mmol/mol) | Sets the U/Ca of the bedrock |
| bedrock_d44Ca | Site-specific (‰) | Sets the calcium isotopic composition of the bedrock |
| bedrock_d13C | -3 to 4 (‰, VPDB) | Sets the carbon isotopic composition of the bedrock |
| bedrock_d18O | -5 to 3 (‰, VPDB) | Sets the oxygen isotopic composition of the bedrock |
| bedrock_mineral | 'Dolomite", 'Calcite" or 'Aragonite" | Sets the mineralogy of the bedrock |
| Bedrock Conditions | | |
| bedrock (moles) | 10 (default) (moles) | Moles of bedrock. Defaults to excess moles |
| bedrock_pyrite | 0 to (moles) | Moles of pyrite in bedrock. Initialises pyrite oxidation in the prescence of oxygen. |

| | | |
|---|---|---|
| gas_volume | 0 to 500 (L/kg) | Controls the amount of gas present during dissolution. Impacts speleothem d13C and DCP |
| reprecip | False or True | Allows for carbonate re-precipitation Impacts speleothem d13C, d44Ca and DCP . |
| Aragonite/Calcite Mode | | |
| precipitate_mineralogy | 'Calcite' or 'Aragonite' | Sets the speleothem mineralogy |
| Cave air | | |
| cave_O2 | 0 to 0.21 (%) | Initialised only in 'kinetics_mode' = 'single_step_degassing' Sets the oxygen composition of the cave air |
| cave_pCO2 | Site-specific ppmv | The CO2 concentration of the cave air. Determines the end-point of degassing/carbonate precipitation |
| cave_d13C | Site-specific ‰ | Initialised only in 'kinetics_mode' = 'single_step_degassing' Sets the carbon isotopic composition of the cave air that equilibriates with the solution |
| cave_R14C | Site-specific ‰ | Initialised only in 'kinetics_mode' = 'single_step_degassing' Sets the radiocarbon activity of the cave air that equilibriates with the solution |
| cave_d18O | Site-specific ‰ | Initialised only in 'kinetics_mode' = 'single_step_degassing'. Sets the oxygen isotopic composition of the cave air that equilibriates with the solution |

| | | |
|---|---|---|
| cave_air_volume | Litres | Initialised only in 'kinetics_mode' = 'single_step_degassing'. Sets the cave air volume that can equilibriate with the solution |
| General | | |
| temperature | 0 to 35 (Celsius) | Sets in-cave temperature. Impacts the kinetics, thermodynamics, partitioning coefficients and fractionation factors. |
| kinetics_mode | ' ' | Sets how the model is ran. See original publication. |
| Scripting Options | | |
| co2_decrement | 0.5 | Sets the fraction of CO2 removed at each degassing step |
| calcite_sat_limit | 1 | Sets the index at which CaCO3 precipitation occurs |

Table 3: A summary of the model inputs, with a realistic range of values

| Model Outputs (Key) | Definition |
|---|---|
| Ba(mol/kgw) | The Barium concentration within the solution |
| Ba/Ca(mol/mol) | The ratio of Barium to Calcium within the solution |
| Ba/Ca(mol/mol)_Calcite / Ba/Ca(mol/mol)_Aragonite | Speleothem Ba/Ca value |
| C(mol/kgw) | The Carbon concentration within the solution |
| Ca(mol/kgw) | The Calcium concentration within the solution |
| CO2(mol/kgw) | The CO2 concentration within the solution |
| CO3-2(mol/kgw) | The carbonate ion concentration within the solution |
| d13C | The Dissolved Inorganic Carbon (DIC) d13C |
| d13C_Calcite / d13C_Aragonite | Speleothem d13C |
| d13C_CO2(aq) | The d13C of CO2 in aqueous form within the solution |
| d13C_CO3-2 | The d13C of the carbonate ion within the solution |
| d13C_HCO3- | The d13C of the bicarbonate ion within the solution |
| d18O | The d18O of the H2O resevoir (VSMOW) |
| d18O_Calcite / d18O_Aragonite | Speleothem d18O (VSMOW) |
| d18O_CO2(aq) | The d18O of CO2 in aqueous form within the solution (VSMOW) |
| d18O_CO3-2 | The d18O of the carbonate ion within the solution (VSMOW) |
| d18O_HCO3- | The d18O of the bicarbonate ion within the solution (VSMOW) |
| d18O_PDB | Speleothem d18O (VPDB) |
| d44Ca | The d44Ca within the solution |
| d44Ca_Calcite / d44Ca_Aragonite | Speleothem d44Ca |
| DCP | Speleothem Dead Carbon Percentage |
| f_c | Fraction of carbon remaining in solution |
| f_ca | Fraction of calcium remaining in the solution |
| HCO3-(mol/kgw) | Concentration of the bicarbonate ion within the solution |
| Mg(mol/kgw) | The concentration of Magnesium within the solution |
| Mg/Ca(mol/mol) | The ratio of Magnesium to Calcium within the solution |
| Mg/Ca(mol/mol)_Calcite / Mg/Ca(mol/mol_Aragonite | Speleothem Mg/Ca |
| O(mol/kgw) | Concentration of oxygen within the solution |
| pH | pH of the solution |
| R14C | Speleothem R14C |
| si_Calcite | Saturation index with respect to calcite |
| Sr(mol/kgw) | Concentration of Strontium within the solution |
| Sr/Ca(mol/mol) | The ratio of Srontium to Calcium within the solution |
| Sr/Ca(mol/mol)_Calcite / Sr/Ca(mol/mol)_Aragonite | Speleothem Sr/Ca |
| step_desc | The step description (e.g soil-water equilibriation, bedrock dissolution or degassing?) |
| U(mol/kgw) | Concentration of Uranium within the solution |
| U/Ca(mol/mol) | The ratio of Uranium to Calcium within the solution |
| U/Ca(mol/mol)_Calcite / U/Ca(mol/mol)_Aragonite | Speleothem U/Ca |

Table 4: