

## Using THINCARB

Stephen King  
STFC Rutherford Appleton Laboratory  
Harwell Campus  
Didcot  
Oxfordshire  
OX11 0QX

[stephen.king@stfc.ac.uk](mailto:stephen.king@stfc.ac.uk)

### What is THINCARB?

THINCARB stands for Thermodynamics of Inorganic CARBon and is a computational model for estimating the excess partial pressure of CO<sub>2</sub> (EpCO<sub>2</sub>), and the equilibrium constants, activities, concentrations, and fractionation, of the hydroxyl and principal carbon-containing ions present in natural waters (ie, rivers and lakes) from basic water quality determinands.

THINCARB is not suitable for marine environments.

At the heart of THINCARB are the equations developed by Neal *et al* (1998)<sup>[1]</sup> and which have proved to be extremely useful. However, in the intervening years some minor corrections became apparent, and the platform on which that original model was deployed became obsolete! In addition, it was noted that there was a straightforward extension of the model that would also provide estimates of Dissolved Inorganic Carbon (DIC) concentrations, something that is not routinely measured experimentally. So against this background it was decided to make a corrected, enhanced, model available to a new generation of water quality scientists. This enhanced model is THINCARB!<sup>[2]</sup>

The new DIC calculations have been validated using literature data and seem to provide quantitative agreement with experimental measurements *except in samples having both low alkalinity and high levels of Dissolved Organic Carbon (DOC)*.

### What do I need to run THINCARB?

You can run THINCARB in one of two ways: as a macro embedded in a Microsoft Excel™ (Excel Version 14.0, 'Excel 2010') or later spreadsheet, or by using the Python language (Versions 2.7.x and 3.x.x are supported). There are pros and cons of both approaches:

#### Excel

##### Pros

- It is usually installed on most organisational PCs and laptops running Windows™
- It is familiar to use to most people and doesn't require any programming
- It is tolerant of missing inputs
- It is fine for processing small (eg, low temporal resolution) datasets

##### Cons

- It is not free and requires a license
- It only works on Windows™ machines
- Many organisations regard embedded macros with suspicion because they are a potential security risk
- It is not as fast as Python
- It is very tedious to use for processing large (eg, high temporal resolution) datasets

## Python

### Pros

- It is free
- It is cross-platform (ie, it can be used with Windows™, Unix, Linux, Mac™ OSX)
- It is faster than using Visual Basic (that runs Excel™ macros)
- It is ideal for processing large (eg, high temporal resolution) datasets
- It can be used in conjunction with the 'Cloud'

### Cons

- It needs to be downloaded and installed (which may require administrator permissions)
- It requires a little programming aptitude to use it effectively
- It requires values for all inputs, whether they are known or not

## What input data does THINCARB require?

The absolute minimum input data are the pH and Gran alkalinity of a water sample, though temperature has a big effect on the calculations, so if this is not known then an 'educated guess' will be necessary! However, all three are usually routinely measured water quality determinands.

Optional additional inputs, which can be used to refine the calculations if available, are the altitude at which the water sample was taken and the calcium ion concentration.

<i>Required units for input data</i>			
Gran alkalinity	Temperature	Altitude	[Ca]
μEq/l	°C	m	mg/l

## What does THINCARB calculate?

The full list of output values calculated is shown below.

- Excess partial pressure of dissolved CO<sub>2</sub> (EpCO<sub>2</sub>) {5 successive approximations}
- CaCO<sub>3</sub> saturation
- Equilibrium constants for:
  - Formation of H<sub>2</sub>CO<sub>3</sub> from CO<sub>2</sub> (k<sub>0</sub>)
  - Dissociation of H<sub>2</sub>CO<sub>3</sub> (k<sub>1</sub>)
  - Dissociation of HCO<sub>3</sub><sup>-</sup> (k<sub>2</sub>)
  - Formation of CaHCO<sub>3</sub><sup>+</sup> (k<sub>3</sub>)
  - Formation of CaCO<sub>3</sub> (k<sub>4</sub>)
  - Formation of CaOH<sup>+</sup> (k<sub>5</sub>)
  - Formation of H<sub>2</sub>O (k<sub>6</sub>)
  - Dissolution of solid CaCO<sub>3</sub> (k<sub>7</sub>)
- Chemical activities of:
  - OH<sup>-</sup>
  - H<sub>2</sub>CO<sub>3</sub>
  - HCO<sub>3</sub><sup>-</sup>
  - CO<sub>3</sub><sup>2-</sup>
  - Ca<sup>2+</sup>
  - CaHCO<sub>3</sub><sup>+</sup>
  - CaCO<sub>3</sub>
  - CaOH<sup>+</sup>
- Concentrations of:
  - Total Ca
  - HCO<sub>3</sub><sup>-</sup>
  - CO<sub>3</sub><sup>2-</sup>
  - H<sub>2</sub>CO<sub>3</sub>
  - CaHCO<sub>3</sub><sup>+</sup>

- $\text{CaCO}_3$
- C in  $\text{HCO}_3^-$
- C in  $\text{CO}_3^{2-}$
- C in  $\text{H}_2\text{CO}_3$
- C in  $\text{CaHCO}_3^+$
- C in  $\text{CaCO}_3$
- DIC as  $\text{HCO}_3^-$
- DIC as  $\text{CO}_3^{2-}$
- DIC as  $\text{H}_2\text{CO}_3$
- DIC as  $\text{CaHCO}_3^+$
- DIC as  $\text{CaCO}_3$
- Total DIC
- Ionic strength
- Monovalent activity coefficient
- Divalent activity coefficient

### How does THINCARB work?

For the physical chemistry behind the model the reader is referred to the references <sup>[1],[2]</sup>.

The algorithmic approach of both the Excel<sup>TM</sup> and Python implementations of THINCARB is the same: an initial approximation for  $\text{EpCO}_2$  is gradually refined (optimised) by minimising the overall charge balance in the system. In Excel<sup>TM</sup> this optimisation is performed with the *Goal Seek* function. In Python a simple, but tenacious, bisection algorithm is used. The latter usually achieves a better residual charge balance\*, but both approaches will give essentially the same outputs for the same residual charge balance. For this reason care must be exercised if comparing results from the Excel<sup>TM</sup> implementation with results from the Python implementation!

*\*An interesting philosophical debate is whether a residual charge balance of, say,  $10^{-9}$  e is actually physically more important or realistic than, say, a residual charge balance of  $10^{-4}$  e. However, this difference is sufficient to produce variations in  $\text{EpCO}_2$ ,  $[\text{HCO}_3^-]$ , and the total [DIC] of almost 4%! So beware!*

### Speed

How quickly THINCARB processes datasets depends on several factors: the specification of the computer, the acceptable residual charge balance, how close to that acceptable residual charge balance the initial estimate of  $\text{EpCO}_2$  puts the starting charge balance, and the efficiency of the optimisation algorithm.

The Python implementation has been coded with human-readability (not speed) in mind, and the bisection algorithm is very robust but not particularly fast. Consequently there is undoubtedly scope for future speed improvements if needed. But as a guide, during real use of THINCARB on a harmonised dataset, the Python 2.7 implementation processed 8300 readings from file in 88 mins (ie, roughly 1.5/sec) on an Intel quad-core 3.2 GHz CPU based PC with 12 GB RAM running Python from a Windows 7<sup>TM</sup> command line interface.

## References

[1] 'An assessment of excess carbon dioxide partial pressures in natural waters based on pH and alkalinity measurements'. Colin Neal, W Alan House, Kevin Down. *The Science of the Total Environment*. 210/211 (1998) 173-185.

[2] 'Inorganic Carbon Dominates Dissolved Carbon Concentrations and Fluxes in British Rivers: Application of the THINCARB Model - Thermodynamics of Inorganic Carbon'. Helen Jarvie, Stephen King, Colin Neal. *Manuscript in preparation*.

## Running THINCARB from Excel 2010™

From the THINCARB repository \src folder, download the vba sub-folder. Open the file **THINCARB-excel2010.xlsm** in Excel™.

This file comes with an example dataset (taken from [1]) and is ‘ready to go’. The input values are in green, and the model calculations are in red. Above the input values are a button and three parameters: the ‘target’ value that the *Goal Seek* function should attempt to get the charge balance to, and the first and last lines of the dataset that are to be processed.

Click the *CommandButton*. A number of the red calculated values will change. Once all the measurement sets have been processed a dialog box will pop-up and say *Finished!*

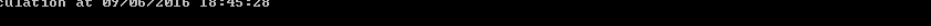
To use this spreadsheet for another dataset:

- simply **replace the existing green input values** with the new input values
- **carefully select Cells H15 > AV15** (ie, all the red values in Row 15)
- **drag this selection down** as far as the new input values extend
- **update the start and end rows in Cells E5 & E6**
- **Click the *CommandButton!***

[illegible]

**(above) The THINCARB spreadsheet**

*(below) THINCARB running from Python*



```

C:\windows\system32\cmd.exe
E:\Data\Helen\CO2Therm\THINCARB\CS data>python call_THINCARB_from_file.py
Starting calculation at 09/06/2016 18:40:02
Processed 100 readings so far at 18:40:59
Processed 200 readings so far at 18:42:08
Processed 300 readings so far at 18:44:00
Processed 400 readings so far at 18:44:41
Processed 500 readings so far at 18:45:03
Processed 600 readings so far at 18:45:17
Processed 676 readings in total
Finishing calculation at 09/06/2016 18:45:28

Done!

```

## Running THINCARB from Python

First, if you do not already have Python installed (ArcGIS 10.x, for example, ships with a Python interpreter), download a version of Python appropriate to your computer operating system from <https://www.python.org/> and install it. (At the time of writing (May 2016), THINCARB has been tested with 32-bit versions of Python 2.7.10, and Python 3.4.2)

Then, from the THINCARB repository \src folder, download the sub-folder corresponding to your chosen version of Python.

There are two ways to run the Python version of THINCARB: you can either edit your measurement data by hand into the file `call_THINCARB_pyX_from_list.py` (where X indicates the Python version), or you can have `call_THINCARB_pyX_from_file.py` read measurement data from a separate file (eg, as output by a data logger or extracted from a database).

### From LIST Method

Using a text editor (eg, Notepad, vi) open the file `call_THINCARB_pyX_from_list.py`.

Scroll down to the section headed `##### ENTER YOUR OBSERVATIONS/READINGS HERE #####` and locate the lines beginning `inputZ=`, etc.

Enter your measurement data for each input parameter as a comma-separated list of values between the [] brackets. For example:

```
# These test data are from Neal et al (1998), Table 5. A, B, C and G are dummy values.
inputZ=['Neal_Table5_Line_1','Neal_Table5_Line_2','Neal_Table5_Line_3','Neal_Table5_Line_4','Neal_Table5_Line_5','Neal_Table5_Line_6','Neal_Table5_Line_7','Neal_Table5_Line_8','Neal_Table5_Line_9','Neal_Table5_Line_10']
inputA=['09/09/14','09/09/14','09/09/14','09/09/14','09/09/14','09/09/14','09/09/14','09/09/14','09/09/14','09/09/14']
inputB=['09:00','10:00','11:00','12:00','13:00','14:00','15:00','16:00','17:00','18:00']
inputC=[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
inputD=[8.29,8.38,8.21,8.31,8.24,8.42,8.32,8.17,8.01,8.20]
inputE=[3930.0,3880.0,3970.0,4010.0,3850.0,3890.0,3940.0,3930.0,3680.0,3970.0]
inputF=[12.2,10.6,13.6,18.0,16.6,15.2,17.0,17.8,16.6,15.3]
inputG=[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
```

Note that *location* (**inputZ**), *date* (**inputA**) and *time* (**inputB**) values are text strings and must be enclosed by single quote (') characters. These values are reference metadata only and so can be empty strings (ie, '') or just assigned 'dummy' values.

Unknown measurement values (most often things like the *altitude* (**inputC**) or the calcium concentration (**inputG**)) must still be assigned a number. If in doubt, zero is probably the safest choice for these! The *temperature* (**inputF**) can usually be estimated with some degree of certainty if it has not actually been recorded.

Make sure each input list has the same number of entries!

Also make certain that your input values are in the correct units (see the comments in the file and earlier in this document)!

Finally, just below `##### END OF OBSERVATIONS/READINGS #####` enter a name for the file to contain the results of the THINCARB calculations.

Save the file!

*(How you do this next part depends on the computer operating system you are using)*

From your operating system command line interface, type:

```
python call_THINCARB_pyX_from_list.py
```

THINCARB should process your data. When it has finished it will say *Done!* and tell you how many readings it processed in total. It will also have created your output file. You can open the output file in a text editor, or easily import the contents (it is space-delimited) into a spreadsheet.

#### From FILE Method

THINCARB expects to read an ASCII text file of measurement values, one set of readings to a line, where each value is separated by a space. This is important!

*It also means that the name of any site/location/station for that set of readings must not contain any spaces! Spaces in site names should be removed or replaced with underscores.*

As a first step, therefore, the proposed input data file should be examined in a text editor (eg, Notepad, vi) and the editors *find-replace* or *macro* functions used to change delimiting commas, tabs, etc, to spaces. Any resulting spaces at the start of a line of readings also then need to be removed, as do any column headings or trailing spaces. It is also a good idea to remove any blank lines after the last line of data.

If the dataset is presented as a spreadsheet workbook it will be necessary to use the spreadsheet *Save as...* option to write the dataset to a text format suitable for editing. For example, Excel™ has a *Formatted Text (Space Delimited)* output format.

Here is an example of a suitably formatted set of measurement values (in this case comprising of the data origin, date, time, pH, alkalinity and temperature; in this example the date and time are actually fictitious 'dummy' values!):

```
Neal_Table5_Line_1 09/09/2014 09:00 8.29 3930 12.2
Neal_Table5_Line_2 09/09/2014 10:00 8.38 3880 10.6
Neal_Table5_Line_3 09/09/2014 11:00 8.21 3970 13.6
Neal_Table5_Line_4 09/09/2014 12:00 8.31 4010 18.0
Neal_Table5_Line_5 09/09/2014 13:00 8.24 3850 16.6
Neal_Table5_Line_6 09/09/2014 14:00 8.42 3890 15.2
Neal_Table5_Line_7 09/09/2014 15:00 8.32 3940 17.0
Neal_Table5_Line_8 09/09/2014 16:00 8.17 3930 17.8
Neal_Table5_Line_9 09/09/2014 17:00 8.01 3680 16.6
Neal_Table5_Line_10 09/09/2014 18:00 8.20 3970 15.3
```

Notice that the fields do not have to line up with one another; they just need to be separated by a space.

Save the file!

Then, using the text editor open the file `call_THINCARB_pyX_from_file.py`.

Scroll down to the line beginning **InputFilename=** and enter the name of the file containing the measurements for THINCARB to process.

Below this, enter a name for the file to contain the results of the THINCARB calculations.

Now scroll down to the section headed **#### NOW MATCH THE COLUMNS ####** and locate the lines containing **=columns[n]** and **=float(columns[n])**.

Here you need to match the THINCARB input variable (ie, SITE, A, B, C, D, E, F, G) to the appropriate column of measurement values in your input data file or, if a set of measurement values are not in the file, set the corresponding input value to a fixed 'dummy' value (see below).

So in this example

```
SITE=columns[0]
A=columns[1]
B=columns[2]
D=float(columns[3])
E=float(columns[4])
F=float(columns[5])
```

```
C=0.0
G=0.0
```

the input file contains 6 columns of data (location, date, time, pH, alkalinity & temperature) and 2 fields (altitude & [Ca]) are being set to zero.

Note that because *location* (**SITE**), *date* (**A**) and *time* (**B**) are text fields they do not need to be read as 'float' values. Also note that Python array indices start at 0 and not 1!

Unknown measurement values (most often things like the *altitude* (**C**) or the calcium concentration (**G**)) must still be assigned a number. If in doubt, zero is probably the safest choice for these! The *temperature* (**F**) can usually be estimated with some degree of certainty if it has not actually been recorded.

Make certain that your input values are in the correct units (see the comments in the file and earlier in this document)!

Save the file!

*(How you do this next part depends on the computer operating system you are using)*

From your operating system command line interface, type:

```
python call_THINCARB_pyX_from_file.py
```

THINCARB should process your data. As it proceeds it will give you an update on how many 100 measurements it has processed. When it has finished it will say *Done!* and tell you how many readings it processed in total. It will also have created your output file. You can open the output file in a text editor, or easily import the contents (it is space-delimited) into a spreadsheet.

The call handler routines in the sub-folder you downloaded are 'ready to go' and can be used for test purposes. Example input (data taken from [1]) and output files are provided.



## **Acknowledging THINCARB**

If you use THINCARB in your work please acknowledge as much in your publications/presentations!

For example:

*This work benefitted from THINCARB, software developed by the NERC Centre for Ecology & Hydrology and the STFC. See <https://smk78.github.io/thincarb/>*

## **Legal Stuff**

THINCARB has been developed to be used, and perhaps even enhanced, by the water quality community, be they academic, government, or commercial. For that reason it has been given a permissive licence. Basically the only thing you are prevented from doing with THINCARB is selling it!

Please see the file LICENSE.txt for more details.