VOLATILECALC FOR PYTHON

A3.1. Credit

This python script is a direct port of the visual basic script developed for the following publication:

Newman, Sally, and Jacob B. Lowenstern. "VolatileCalc: a silicate melt–H_2O–CO_2 solution model written in Visual Basic for excel." Computers & Geosciences 28.5 (2002): 597-604.

Please cite the above publication for use of this script.

A3.2. Running python script

The script is written for Python 2.X and requires numpy package.

The script is written as a function. To run it, place "VolatileCalc.py" in your working directory and import.

All the same functions as the excel version are included. They are detailed below.

It should be noted that the python script performs the calculations significantly slower than the excel version. This is most notable for calculations of degassing paths and when looping over VolatileCalc functions numerous times.

A3.3. Functions

Examples of how to use this script are given below. Lines in blue are those input by the user. Lines in green that start with ">>>" indicate output.

All functions take the same three arguments. However, each argument is different depending on the desired function. The general form is:

VolatileCalc(*func, var1, var2*)

The first argument (*func*) is a string that describes the desired function. The values of *var1* and *var2* will be described in the following sections.

All functions output two variables. The first contains the output data. The second variable contains error messages. If no errors have occurred, the second variable will be an empty list.

A3.3.1. Fugacity (func = 'fug')

Fugacity arguments

var1 = T (in °C)
var2 = P (in MPa)

These two arguments indicate the desired T and P of the calculation. They can be float or integer numbers. Limits: T ≥ 450 °C and P ≥ 20 MPa

Fugacity output

The first output is a list. The first element is the fugacity of CO_2 (in MPa) and the second is the fugacity of H_2O (in MPa).

The second output is a list of errors.

Fugacity example

```
import numpy as np
import VolatileCalc

VolatileCalc('fug',1000,100)

>>> ([126.9, 93.9],[])
```

*Tip: Because all VolatileCalc functions output two variables (the first is the output data, the second is a list of strings that indicate errors that have occurred), a good way to save the output would be as follows:

```
output_data, errors = VolatileCalc('fug',1000,100)
```

A3.3.2. Vapor saturation pressure (func = 'sp')

Vapor saturation pressure arguments

var1 = 'basalt' or 'rhyolite'
This variable indicates the composition of the system.

var2 is a list that is different for 'basalt' and 'rhyolite' options.
If *var1* is set to 'basalt':
var2 = [H_2O (in wt%), CO_2 (in ppm), SiO_2 (in wt%), temperature (in °C)]
If *var1* is set to 'rhyolite':
var2 = [H_2O (in wt%), CO_2 (in ppm), temperature (in °C)]

If *var1* == 'basalt', SiO_2 must be ≥40 and ≤49 wt%.

Vapor saturation pressure output

The first output is a list. The list includes the following:
[Saturation pressure (in MPa),
H_2O (in wt%),
CO_2 (in ppm),

molecular H_2O in melt (in wt%),
OH in melt (in wt%),
percent of H_2O in the vapor,
percent of CO_2 in the vapor]

The second output is a list of errors.

Vapor saturation pressure example

```
import numpy as np
import VolatileCalc

VolatileCalc('sp','basalt',[4,1800,49,1200])

>>> ([479.1, 4.0, 1800.0, 2.1, 1.9, 35.6, 64.4], [])

VolatileCalc('sp','rhyolite',[6,2500,800])

>>> ([508.4,  6.0,  2500.0,  4.2,  1.8,  39.5,  60.5], ['VolatileCalc is not recommended for P > 500 MPa.'])
```

A3.3.3. Degassing path (func = 'dp')

Degassing path arguments

var1 = 'basalt' or 'rhyolite'
This variable indicates the composition of the system.

var2 is a list that is different for 'basalt' and 'rhyolite' options.
If *var1* is set to 'basalt':
var2 = [starting H_2O (in wt%), starting CO_2 (in ppm), SiO_2 (in wt%), temperature (in °C), [style], steps]
If *var1* is set to 'rhyolite':
var2 = [starting H_2O (in wt%), starting CO_2 (in ppm), temperature (in °C), [style], steps]

[style] is a list. In the case of open system degassing: *[style]* = [0]. In the case of closed system degassing: *[style]* = [1, wt% of excess vapor].

steps is an integer that indicates the number of steps in the degassing calculation.

In the case of *var1* == 'basalt', SiO_2 must be $\geq$40 and $\leq$49 wt%.

Degassing path output

The first output is a list of lists. Each of the nested lists corresponds to a step in the degassing calculation (i.e., [[step 1 list], [step 2 list], etc.]) and has the same variables as the output for the vapor saturation pressure output:
[Saturation pressure (in MPa),
H_2O (in wt%),
CO_2 (in ppm),
molecular H_2O in melt (in wt%),
OH in melt (in wt%),
percent of H_2O in the vapor,
percent of CO_2 in the vapor]

The second output is a list of errors.

Degassing path example

```
import numpy as np
import VolatileCalc

VolatileCalc('dp','basalt',[3.5,1200,45,1100,[1,2],20])

>>> ([[242.9, 3.5, 1200.0, 1.7, 1.8, 65.7, 34.3],…, [30.8, 1.5, 60.0, 0.35, 1.1, 85.0, 15.0]], [])
```

*Tip: It may be helpful to transpose the first output so that all the pressures, H_2O values, CO_2 values, etc. are together in lists. To do so, simply do the following:

```
output_data, errors = VolatileCalc('dp','basalt',[3.5,1200,45,1100,[1,2],20])
output_data = map(list, zip(*output_data))
```

A3.3.4. Isobar (func = 'ib')

Isobar arguments

var1 = 'basalt' or 'rhyolite'
This variable indicates the composition of the system.

var2 is a list that is different for 'basalt' and 'rhyolite' options.
If *var1* is set to 'basalt':
var2 = [P (in MPa), T (in °C), SiO_2 (in wt%), steps]
If *var1* is set to 'rhyolite':
var2 = [P (in MPa), T (in °C), steps]

steps is an integer that indicates the number of steps in the isobar calculation.

In the case of *var1* == 'basalt', SiO_2 must be ≥40 and ≤49 wt%.

Isobar output

The first output is a list of lists. Each of the nested lists corresponds to a step in the isobar calculation (i.e., [[step 1 list], [step 2 list], etc.]) and has the same variables as the output for the vapor saturation pressure output:
[Saturation pressure (in MPa),
H_2O (in wt%),
CO_2 (in ppm),
molecular H_2O in melt (in wt%),
OH in melt (in wt%),
percent of H_2O in the vapor,
percent of CO_2 in the vapor]

The second output is a list of errors.

Isobar example

```
import numpy as np
import VolatileCalc

VolatileCalc('ib','rhyolite',[450,800,5])

>>> ([[450.0, 11.4, 0, 9.5, 1.9, 100, 0.0],…, [450.0, 0, 3471.1 0, 0, 0.0, 100]], [])
```

A3.3.5. Isopleth (func = 'ip')

Isopleth arguments

var1 = 'basalt' or 'rhyolite'
This variable indicates the composition of the system.

var2 is a list that is different for 'basalt' and 'rhyolite' options.
If *var1* is set to 'basalt':
var2 = [CO_2 increment (in ppm), Molar percentage of H_2O in fluid phase (in %), steps, SiO_2 (in wt%), temperature (in °C)]
If *var1* is set to 'rhyolite':
var2 = [CO_2 increment (in ppm), Molar percentage of H_2O in fluid phase (in %), steps, temperature (in °C)]

steps is an integer that indicates the number of steps in the isobar calculation.

In the case of *var1* == 'basalt', SiO2 must be ≥40 and ≤49 wt%.

Isopleth output

The first output is a list of lists. Each of the nested lists corresponds to a step in the isopleth calculation (i.e., [[step 1 list], [step 2 list], etc.]) and has the same variables as the output for the vapor saturation pressure output:
[Saturation pressure (in MPa), H_2O (in wt%), CO_2 (in ppm), molecular H_2O in melt (in wt%), OH in melt (in wt%), percent of H_2O in the vapor, percent of CO_2 in the vapor]

The second output is a list of errors.

Isopleth example

```
import numpy as np
import VolatileCalc

VolatileCalc('ip','basalt',[50,50,20,47,1050])

>>> ([[0, 0, 0, 0, 0, 50.0, 50.0],…, [200.0, 2.8, 950, 1.2, 1.6, 50.0, 50.0]], [])
```

A3.3.6. Solubility vs. Pressure (func = 'svp')

Solubility vs. pressure arguments

var1 = 'basalt' or 'rhyolite'
This variable indicates the composition of the system.

var2 is a list that is different for 'basalt' and 'rhyolite' options.
If *var1* is set to 'basalt':
var2 = [variable, pressure interval (in MPa), SiO_2 (in wt%), temperature (in °C)]
If *var1* is set to 'rhyolite':
var2 = [variable, pressure interval (in MPa), temperature (in °C)]

variable is a string that indicates whether to perform the calculation for 'H2O' or 'CO2'.

In the case of *var1* == 'basalt', SiO_2 must be ≥40 and ≤49 wt%.

Solubility vs. pressure output

The first output is a list of lists. Each of the nested lists corresponds to a step in the solubility vs. pressure calculation (i.e., [[step 1 list], [step 2 list], etc.]) and has the same variables as the output for the vapor saturation pressure output:
[Saturation pressure (in MPa), H_2O (in wt%), CO_2 (in ppm), molecular H_2O in melt (in wt%), OH in melt (in wt%), percent of H_2O in the vapor, percent of CO_2 in the vapor]

The second output is a list of errors.

Solubility vs. pressure example

```
import numpy as np
import VolatileCalc

VolatileCalc('svp','rhyolite',['H2O',50,900])

>>> ([[0, 0, 0, 0, 0, 100, 0],…, [500.0, 12.2, 0, 10.3, 1.9, 100, 0.0]], [])
```