RESEARCH ARTICLE

VOLCANICA

PySulfSat: An open-source Python3 tool for modeling sulfide and sulfate saturation

Penny E. Wieser* and Matthew Gleeson

Department of Earth and Planetary Science, UC Berkeley, California, U.S.A.

Abstract

We present PySulfSat, an Open-Source Python3 tool for modeling sulfide and anhydrite saturation in magmas. PySulfSat supports a variety of input data types (spreadsheets, Petrolog3 outputs, MELTS tbl files), and can be directly integrated with alphaMELTS for Python infrastructure to track sulfur solubility during fractional crystallization within a single Jupyter Notebook. PySulfSat allows easy propagation of uncertainty using Monte Carlo methods, and far more customization of calculations than existing tools. For example, the $SCSS^{2-}$ could be calculated with one model using the sulfide composition from a parameterization released with a different $SCSS^{2-}$ model. There are also functions for calculating the proportion of S^{6+}/S_{Tot} (allowing modeled SCSS and SCAS values to be converted into total S solubility to compare to natural data), and for modeling mantle melting in the presence of sulfides using a variety of SCSS and K_D models. Extensive documentation and worked examples are available at ReadTheDocs (https://bit.ly/PySulfSatRTD) along with narrated YouTube videos (https://bit.ly/PySulfSatYouTube).

Keywords: Python3; Sulfide; FAIR; Open-source; SCAS; Anhydrite; Mantle melting.

Submitted: 2023-02-09 | Accepted: 2023-04-04 | Published: 2023-05-15

Editor: J. Farquharson | Typesetter: J. Farquharson

1 Introduction

Modeling the solubility of sulfur in a silicate melt provides vital insights into the evolution of sulfur and other S-loving (chalcophile) elements during mantle melting and crustal processes such as fractional crystallization and crustal contamination [Ding and Dasgupta 2018; Reekie et al. 2019; Wieser et al. 2020; Wieser and Jenner 2021; Iacono-Marziano et al. 2022; Muth and Wallace 2022; Virtanen et al. 2022]. Modeling the removal of sulfide and sulfate phases is particularly vital to understand the formation of economical deposits of chalcophile elements, as well as the sulfur and metal flux emitted to the atmosphere during volcanic eruptions [Edmonds et al. 2018; Wieser et al. 2020; Mason et al. 2021]. A number of different models have been proposed over the years to calculate the sulfide content at sulfide saturation ($SCSS^{2-}$), which describes the amount of sulfide (S^{2-}) that can dissolve in a silicate melt saturated in a sulfide phase [e.g. Li and Ripley 2009; Fortin et al. 2015; Smythe et al. 2017; O'Neill 2021]. Numerous models also exist to quantify the sulfate content at anhydrite saturation (SCAS), which describes the amount of sulfate (S^{6+}) that dissolves in a silicate melt when saturated in anhydrite [e.g. Li and Ripley 2009; Baker and Moretti 2011; Masotta and Keppler 2015; Chowdhury and Dasgupta 2019; Zajacz and Tsay 2019]. In many magmas with intermediate oxygen fugacity (e.g. in volcanic arcs), S is present as a mixture of S^{2-} and S^{6+} species [Muth and Wallace 2021]. O'Neill and Mavrogenes [2022], Nash et al. [2019], and Jugo et al. [2010] present models to quantify the proportion of these two species as a function of melt redox. These speciation models can be used alongside $SCSS^{2-}$ and $SCAS^{6+}$ calculations to obtain the total amount of S that is dissolved in the melt (to compare to measured S contents in volcanic systems).

1.1 Previously-available tools

At the moment, $SCSS^{2-}$ and $SCAS^{6+}$ calculations are performed in spreadsheets accompanying each publication [e.g. Fortin et al. 2015; Smythe et al. 2017; O'Neill 2021]. These spreadsheets have a limited number of rows for performing calculations (e.g. $N = 50$ for Smythe et al. [2017], $N = 194$ for O'Neill [2021]), making it difficult to apply them to thousands of natural compositions, or outputs of fractional crystallization models with a small temperature step. The prevalence of Excel-based tools also makes it difficult to propagate uncertainty using Monte Carlo methods.

Available tools also make it time consuming and difficult to compare different models. Existing spreadsheets require users to paste in their melt compositions with oxides in a specific order, and the order differs between spreadsheets. After reformatting the input structure for each model, users would then have to extract outputs and compile these into a single format and location for plotting. There are also tools for which no published spreadsheets exist [e.g. Blanchard et al. 2021], requiring users to contact the author team to obtain such a tool, or individually interpret the equations (which may contain typographical errors, or ambiguities, particularly regarding which units to use).

The most recent $SCSS^{2-}$ models have a term accounting for the composition of the sulfide [Smythe et al. 2017; Blanchard et al. 2021; Liu et al. 2021; O'Neill 2021; Li and Zhang 2022], because melts in equilibrium with a sulfide containing Ni and Cu have a substantially lower SCSS compared with melts in equilibrium with pure Fe-S sulfides. However, the spreadsheets for these different models use a variety of approaches to account for the composition of the sulfide, making it hard to directly compare model outputs. The Smythe et al. [2017] Excel workbook has two sheets; one is designed for users to enter a sulfide composition in wt.%, while the other sheet calculates a sulfide composition using partition coeffi-

* penny_wieser@berkeley.edu

cients from Kiseeva and Wood [2015] and an estimate of the Ni and Cu content in the melt. In contrast, the spreadsheet of O'Neill [2021] calculates the Fe/(Fe+Cu+Ni) content of the sulfide using a simple regression based on the FeO_t, Ni, and Cu content of the melt (calibrated on MORB—mid-ocean ridge basalt), although the user can overwrite this and paste in a fixed value of Fe/(Fe+Cu+Ni). The spreadsheets of Li and Zhang [2022] and Liu et al. [2021] require users to input an estimate of Fe/(Fe+Cu+Ni). To be able to robustly compare the calculated $SCSS^{2-}$ values, it would be preferable to use the same routine for calculating sulfide composition. At the moment, this would require substantial tweaking of spreadsheets by each user.

1.2 PySulfSat: An open-source approach

The tedium associated with performing $SCSS^{2-}$ and $SCAS^{6+}$ calculations in existing spreadsheets, and difficulties associated with comparing models, motivated us to produce PySulfSat. This is an open-source package written in the popular programming language Python3. PySulfSat is designed to be accessible to people with no coding experience. All users must do is install Python on their machine (e.g. through Anaconda). Then, PySulfSat can be installed onto any computer using PyPI (an online software repository) using the following command line prompt:

```
pip install PySulfSat
```

Or, if installation is performed in a Jupyter notebook directly, an exclamation mark is added:

```
!pip install PySulfSat
```

Once PySulfSat is installed on a given computer, it must be loaded into each Jupyter Notebook (or other Python file) using any combination of letter users wish (here we use ss):

```
import PySulfSat as ss
```

Any function is then called from PySulfSat using ss.function_name.

In addition, we encourage users to import pandas [The pandas development team 2020], NumPy [Harris et al. 2020], and matplotlib [Hunter 2007] at the start of each script, for ease of plotting and data manipulation after performing PySulfSat calculations:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

We include numerous narrated worked examples on the PySulfSat YouTube channel to make this package more accessible to non coders*. Some relevant terminology for Python and S modeling is shown in Table 1.

*https://bit.ly/PySulfSatYouTube

Table 1: List of abbreviations

Geological Abbreviations	
SCSS	Sulfide content at sulfide saturation
SCAS	Sulfate content at anhydrite saturation
MELTS	A thermodynamic tool for modelling phase equilibrium in magmatic systems
Petrolog3	A popular software tool for modelling fractional crystallization, reverse fractional crystallization, and post-entrapment crystallization corrections of olivine-hosted melt inclusions.
Python Jargon	
pandas (pd.)	A Python library allowing handling of spreadsheet-like data structures
pandas Series	A 1D column of data with a column heading. Like a single column in an Excel spreadsheet
pandas DataFrame	A 2D data structure (labelled column headings, rows). Can visualize as a collection of pandas series (like a single sheet in an Excel spreadsheet)
NumPy (np.)	A Python library that handles the math used in PySulfSat (e.g. log, exp)
Matplotlib (plt.)	A Python library used for plotting
String (str)	A piece of text
Float (float)	A single number that is not an integer
Integer (int)	A single number that is an integer

1.3 Importing data

Users can import data from any Excel spreadsheet using the import_data function. The input spreadsheet should have the following column headings with oxide contents in wt.%:

1. SiO2_Liq
2. TiO2_Liq
3. Al2O3_Liq
4. FeOt_Liq
5. MnO_Liq
6. MgO_Liq
7. CaO_Liq
8. Na2O_Liq
9. K2O_Liq

Certain models also require users to input the following parameters (Figure 1):

1. P2O5_Liq
2. H2O_Liq
3. Fe3Fet_Liq

The `import_data` function returns a `pandas` dataframe (see Table 1). The order of the columns in the input spreadsheet does not matter, as columns are identified based on their column heading rather than position. If any column headings are missing in the input spreadsheet, they will be filled with zeros. Any additional columns entered by the user (e.g. temperature, pressure, sulfide composition) are appended onto the end of the outputted dataframe, for easy access for calculations. For example, the O'Neill [2021] and Smythe et al. [2017] models require the Ni and Cu content of the liquid in ppm. These can be stored in a column with any heading the user wishes (e.g. `Ni_Liq_ppm`, `Cu_Liq_ppm`), and then obtained from the outputted dataframe (df) using `df['column_name']` to input into the function of interest.

For example, to import generic data (perhaps whole-rock, matrix glass or melt inclusion compositions) from a spreadsheet named "`Liquids1.xlsx`" stored in "`Sheet3`":

```
df_out=ss.import_data(filename='Liquids1.xlsx',
sheet_name='Sheet3')
```

This function also supports specific output files from other petrological modelling programs. For example, users can load in the default spreadsheet-based output from `Petrolog3.1.1.3` [Danyushevsky and Plechov 2011]. Here, the `Petrolog` output is saved to an `Excel` file named "`Petrolog_Model1.xlsx`":

```
df_out=ss.import_data(filename='Petrolog_Model1.xlsx',
Petrolog=True)
```

Similarly, the standard liquid ".tbl" output from `MELTS` [Ghiorso and Sack 1995; Asimow and Ghiorso 1998; Gualda et al. 2012] can be imported:

```
df_out=ss.import_data(filename='melts-liquid.tbl',
MELTS=True)
```

In these examples, the `import_data` function has identified the appropriate column headings in each default structure, and has changed the column names into the format required by `PySulfSat` (e.g. converting `SiO2_melt` from `Petrolog3` into `SiO2_Liq`).

1.4 Units

All temperatures should be entered in Kelvin, all pressures in kbar, and all melt oxides in wt.%, apart from Ni and Cu contents in the liquid which are entered in ppm. All ratios are atomic (e.g. Fe/(Fe+Ni+Cu) in the sulfide).

1.5 Available functions

`PySulfSat` implements the most recent $SCSS^{2-}$ and $SCAS^{6+}$ models (Figure 1). The open-source nature of `PySulfSat` means we anticipate continuining to add models as they are published, so users should check the 'Available Functions' tab at ReadTheDocs*.

*https://bit.ly/PySulfSatRTD

1.6 Calibration datasets

Many SCSS and SCAS models are empirical. Thus, it is not recommended that they are extrapolated too far beyond the compositional range of the calibration dataset. We have compiled available calibration datasets, and incorporated them into `PySulfSat` (see Figure 1 for available datasets). This means that users can easily plot their melt compositions, and estimates of the pressures and temperatures of their system alongside the dataset used to calibrate each model, to assess its suitability. The function **`return_cali_dataset`** returns the calibration dataset for a given model. For example, to obtain the calibration dataset for the Smythe et al. [2017] SCSS model as a pandas.DataFrame:

```
df_S2017=ss.return_cali_dataset(model='S2017_SCSS')
```

Figure 2 shows how these different calibration datasets can be plotted in TAS (total alkali silica) space for visual inspection.

1.7 Worked examples

Example `Jupyter` Notebooks showing a number of workflows are available at ReadTheDocs page†. This list is not exhaustive, and we anticipate that we will continue adding examples in the future:

- Notebooks showing how to import different data types (e.g. measured oxide contents, `Petrolog3` files, and `MELTS` tbl outputs).

- Notebooks showing how to calculate the SCSS and SCAS using a variety of models during fractional crystallization from a `Petrolog3` output [Danyushevsky and Plechov 2011]. This example also shows how to calculate the trajectory of S if a sulfide phase was not present, and how to calculate the mass fraction of sulfide which has formed during crystallization.

- Notebooks showing how to run a `MELTS` fractional crystallization paths at a single pressure and at multiple pressures using `PyMELTScalc` [Gleeson et al. 2023], and then calculate the SCSS and SCAS within the same `Jupyter` Notebook.

- Notebooks showing how to model the SCSS from a `Petrolog3` path, and compare models of S contents and sulfide composition to natural melt inclusion and sulfide data.

- Notebooks showing how to calculate the proportion of S^{6+} using the models of Jugo et al. [2010], Nash et al. [2019], and O'Neill and Mavrogenes [2022].

- Notebooks showing how to perform calculations of trace element evolution during mantle melting in the presence of sulfide using different SCSS, SCAS, and sulfide-silicate partition coefficient (K_D) models.

- Notebooks showing how to propagate uncertainty in input parameters using Monte Carlo methods to obtain 1σ errors for different calculations.

†bit.ly/PySulfSatRTD

Reference	Name in PySulfSat	Melt composition?	T-sens?	P-sens?	H_2O-sens?	*Fe^{3+} sensitive?*	Sulfide/Sulfate comp?	Cali dataset available?
SCAS models								
Chowdhury & Dasgupta (2019)	"calculate_CD2019_SCAS"	✓	✓	✗	✓	✗	✗	✓
Zajacz & Tsay (2019)	"calculate_ZT2022_SCAS"	✓	✓	✗	✓	✗	✗	✓
Masotta & Keppler (2015)	"calculate_MK2015_SCAS"	✓	✓	✗	✓	✗	✗	✓
SCSS models								
Li and Zhang (2022)	"calculate_LiZhang2022_SCSS"	✓	✓	✓	✓	✓	✓	✓
Blanchard et al. (2021)	"calculate_B2021_SCSS"	✓	✓	✓	✓	✗	✓	✓
O'Neill (2021)	"calculate_O2021_SCSS"	✓	✓	✓	✗	✓	✓	✗
O'Neill and Mavrogenes (2022)[*1]	"calculate_OM2022_SCSS"	✓	✓	✓	✗	✓	✓	✓
Liu et al. (2021)	"calculate_Liu2021_SCSS"	✗	✓	✓	✓	✗	✓	✓
Smythe et al. (2017)	"calculate_S2017_SCSS"	✓	✓	✓	✓	✓	✓	✓
Fortin et al. (2015)	"calculate_F2015_SCSS"	✓	✓	✓	✓	✗	✗	✓
Sulfide composition models								
O'Neill (2021)	"Calc_ONeill"	✓	✗	✗	✗	✓		
Smythe et al. (2017) using Kiseeva et al. (2015)	"Calc_Smythe"	✓	✓	✗	✗	✓		

Calculating Proportion of S^{6+} using empirical approaches

Reference	Name in PySulfSat	Input parameters
Jugo et al. (2010)	"calculate_S6St_Jugo2010_eq10"	ΔQFM
Nash et al. (2019)	"calculate_S6St_Nash2019"	T, Fe^{3+}/Fe_T
O'Neill and Mavrogenes (2022)	"calculate_OM2022_S6St"	Melt comp, T, log(fo_2) **or** Fe3/Fe_T

Correcting $SCSS^{2-}$ and $SCAS^{6+}$ calculations for S_T

Name in PySulfSat	Input arguments
"calculate_SCSS_Total"	$SCSS^{2-}$, S^{6+}/S_T
"Calculate_SCAS_Total"	$SCAS^{6+}$, S^{2-}/S_T
"Calculate_S_Total_SCSS_SCAS"	$SCSS^{2-}$, $SCAS^{6+}$, S^{6+}/S_T, or model ('Nash', 'Jugo', 'OM2022', 'Kleinsasser')

Other functions

"crystallize_S_incomp"	Calculates S left in the melt for a given F_{melt} (assuming S is entirely incompatible
"calculate_mass_frac_sulf"	Calculates mass fraction of sulfide removed for a model of changes in SCSS with fractional crystallization
"convert_d34_to_3432S", "convert_3432S_to_d34"	Converts $\delta^{34}S$ to $^{34}S/^{32}S$ and vice versa
"Lee_Wieser_sulfide_melting"	Modelling of S and chalcophile element behaviour during mantle melting.
For Monte Carlo simulations	
'add_noise_2_dataframes'	Generate duplicated rows in df1 based on errors present in df2
'add_noise_series', 'duplicate_dataframe'	Used to simulate uncertainty in specific variables
'av_noise_samples_series'	Average outputs from Monte Carlo simulations per sample

Figure 1: Models currently available in PySulfSat. $SCAS^{6+}$ models: Chowdhury and Dasgupta [2019], Zajacz and Tsay [2019] and Masotta and Keppler [2015]. $SCSS^{2-}$ models: Li and Zhang [2022], Blanchard et al. [2021], O'Neill [2021], O'Neill and Mavrogenes [2022], Liu et al. [2021], Smythe et al. [2017], and Fortin et al. [2015]. [*1]The SCSS model of O'Neill [2021], and O'Neill and Mavrogenes [2022] are extremely similar, differing only with regard to a 7.2*Fe*Si term in 2021, and a 7.2*(Mn+Fe)*Si term in 2022. S^{6+} corrections from Jugo et al. [2010], Nash et al. [2019], and O'Neill and Mavrogenes [2022]. We suggest readers check the ReadTheDocs page for a complete list as we will add new models as they become available.

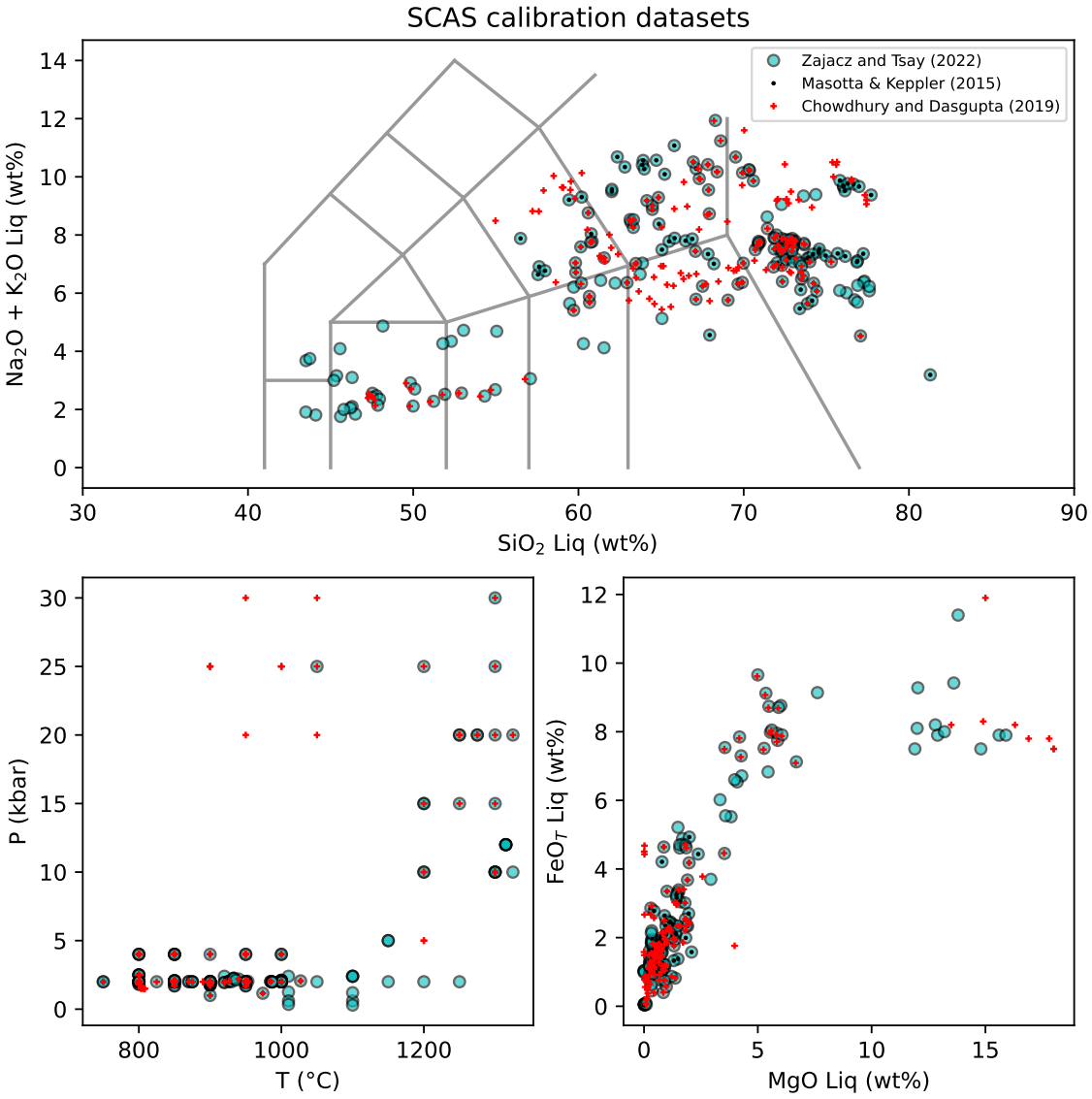

Figure 2: Plots of SCAS calibration datasets in P-T-X space. An example notebook to produce these plots and overlay user data is available at ReadTheDocs. Similar plots can easily be made for SCSS models.

- Notebooks showing other useful features, including calculating values of K_D using various models, converting between S isotope ratios and delta notation, and abundances of different S-bearing species.

2 $SCSS^{2-}$ MODELS

There are a number of ways to perform SCSS calculations, with various options discussed below (worked examples are available at ReadTheDocs).

2.1 Using measured sulfide compositions

The newest SCSS models [e.g. Smythe et al. 2017; Blanchard et al. 2021; O'Neill 2021; Li and Zhang 2022] contain terms for the composition of the sulfide. In some situations, the sulfide composition may have been directly measured in the samples of interest (e.g. using Energy Dispersive Spectroscopy [Wieser et al. 2020]). If so, the function `calculate_sulf_FeFeNiCu` can be used to convert measured elemental abundances in wt.% into the atomic Fe/(Fe+Ni+Cu) ratio used by SCSS models. In some systems, the Fe/(Fe+Ni+Cu) may remain approximately constant during fractional crystallization [Wieser et al. 2020], meaning that a fixed value for this ratio can be used

for simplicity. Figure 3 shows a worked example calculating the $SCSS^{2-}$ using the models of Smythe et al. [2017], O'Neill [2021], and Li and Zhang [2022] for Fe/(Fe+Ni+Cu)=0.65. The expected increase in the S content of the melt with fractional crystallization in the absence of a S-bearing phase is also calculated using the function `crystallize_S_incomp` for comparison (black dashes), and these different S trajectories are plotted using `matplotlib` (where they can be compared to natural melt inclusion or quenched submarine glass data).

2.2 Calculating sulfide compositions

While using a measured sulfide composition is the simplest and most reliable method to perform $SCSS^{2-}$ calculations, direct measurements of sulfide compositions do not exist in many systems. `PySulfSat` allows users to calculate sulfide composition from Ni and Cu contents of the liquid using the approaches implemented in the supporting spreadsheets of O'Neill [2021] and Smythe et al. [2017]. The O'Neill [2021] method is the simplest, calculating the atomic Fe/(Fe+Ni+Cu) ratio using the following empirical expression:

$$\left(\frac{\text{Fe}}{\text{Fe+Ni+Cu}}\right)_{\text{sulf}} = 1/\left[1 + 0.031\left(\frac{\text{Ni}_{\text{Liq, ppm}}}{\text{FeO}_{\text{Liq, wt}}}\right) + 0.025\left(\frac{\text{Cu}_{\text{Liq, ppm}}}{\text{FeO}_{\text{Liq, wt}}}\right)\right], \quad (1)$$

where:

$$\text{FeO}_{\text{Liq, wt}} = \text{FeOt}_{\text{Liq, wt}} \times (1 - \text{Fe}^{3+}/\text{Fe}_{\text{T}}). \quad (2)$$

If the sulfide composition is not known, the spreadsheet of Smythe et al. [2017] has a sheet which will iteratively calculate the sulfide composition based on the partition coefficients of Cu and Ni in the sulfide from Kiseeva and Wood [2015]. These partition coefficients are sensitive to temperature, liquid FeO content, and the Ni and Cu content of the sulfide. Starting with a first estimate of the sulfide Ni and Cu content, the temperature, and the FeO content of the liquid, a partition coefficient can be calculated. Using this partition coefficient along with the initial estimate of the Ni and Cu content in the sulfide, the amount of Cu and Ni in a melt in equilibrium with this sulfide can be calculated. Smythe et al. [2017] define a residual between this calculated value and the measured Ni and Cu contents of the melt:

$$\text{residual} = \left(\text{Ni}_{\text{Liq}}^{\text{Calc}} - \text{Ni}_{\text{Liq}}^{\text{Meas}}\right)^2 + \left(\text{Cu}_{\text{Liq}}^{\text{Calc}} - \text{Cu}_{\text{Liq}}^{\text{Meas}}\right)^2. \quad (3)$$

The `Excel` solver function varies the Ni and Cu in the sulfide to obtain the values which best minimise this residual. Then, the equation of Kiseeva and Wood [2015] is used to calculate the Fe content of the sulfide for these best fit sulfide Ni and Cu contents, and these 3 parameters are used to calculate the sulfide Fe/(Fe+Ni+Cu) ratio. In `PySulfSat`, this convergence routine is performed using the `scipy` optimize minimize function [Virtanen et al. 2020]. In `Excel`, for many compositions, the result obtained can depend slightly on the starting value of the Ni and Cu contents in the sulfide provided by the user. By default, the `PySulfSat` minimisation starts with initial Ni and Cu contents of 5 wt.%, but these parameters can be overwritten using `Cu_Sulf_init=10` and `Ni_Sulf_init=5`. These parameters are allowed to vary between 0–30 wt.%. In general, we find our python implementation of this solver method is stable and gives identical results to the `Excel` version for the same starting composition (and the vast majority of samples converge regardless of the starting Ni and Cu contents).

To perform SCSS calculations with modeled sulfide compositions, a string should be entered into the `Fe_FeNiCu_Sulf` argument. For example, to use the Smythe et al. [2017] $SCSS^{2-}$ model with the O'Neill [2021] calculated sulfide composition, enter `Fe_FeNiCu_sulf='Calc_ONeill'`. Users must also specify the Cu and Ni content in the liquid. In the example below, `Ni_Liq (ppm)` and `Cu_Liq (ppm)` are columns in the loaded dataframe `df_out` containing estimated Ni and Cu contents of the melt in ppm:

```
S17_SCSS_S17_Sulf=ss.calculate_S2017_SCSS(df=df_out,
Fe_FeNiCu_Sulf="Calc_ONeill",
T_K=df_out['T_K'], P_kbar=df_out['P_kbar'],
Fe3Fet_Liq=df_out['Fe3Fet_Liq'],
Ni_Liq=df_out['Ni_Liq (ppm)'],
Cu_Liq=df_out['Cu_Liq (ppm)'])
```

Similarly, to use the O'Neill [2021] $SCSS^{2-}$ model with the Smythe et al. [2017] calculated sulfide composition, specify `Fe_FeNiCu_Sulf='Calc_Smythe'`:

```
O21_SCSS_S17_Sulf=ss.calculate_O2021_SCSS(df=df_out,
Fe_FeNiCu_Sulf="Calc_Smythe",
T_K=df_out['T_K'], P_kbar=df_out['P_kbar'],
Fe3Fet_Liq=df_out['Fe3Fet_Liq'],
Ni_Liq=df_out['Ni_Liq (ppm)'],
Cu_Liq=df_out['Cu_Liq (ppm)'])
```

2.3 H_2O-sensitivity

Unlike the $SCSS^{2-}$ model of O'Neill [2021] which contains no term for H_2O, the $SCSS^{2-}$ models of Fortin et al. [2015], Smythe et al. [2017], Liu et al. [2021], Blanchard et al. [2021], and Li and Zhang [2022] are sensitive to the amount of H_2O in the liquid. By default, the $SCSS^{2-}$ functions for each of these models (Figure 1) use the H_2O content stored in the data loaded by the user in the column `H2O_Liq`. However, this can also be overwritten in the function itself, to allow investigation of the sensitivity of calculations to melt water content. For example, to perform all calculation at 3 wt% H_2O using the Fortin et al. [2015] model:

```
F2015_3H=ss.calculate_F2015_SCSS(df=df_out,
T_K=df_out['T_K'], P_kbar=df_out['P_kbar'],
H2O_Liq=3)
```

The argument `H2O_Liq` could also be set to a `pandas` series (e.g. any other column in the loaded data), which would allow calculations to be performed using several different water contents (e.g. `df_out['Raman_H2O']` for Raman spectroscopy measurements vs. `df_out['SIMS_H2O']` for SIMS measurements in the same samples).

Load data from a Petrolog3 output file

	A	B	C	D	E	F	G	H	I	J
1	SiO2_mag	TiO2_mag	Al2O3_ma	Fe2O3_ma	FeO_mag	MnO_ma	MgO_ma	CaO_mag	Na2O_ma	K2O_mag
2	49.901	0.9981	14.9715	0.9839	8.0964	0.0998	9.9763	11.9772	2.4953	0.1996
3	49.9978	1.0081	15.122	0.9743	8.0754	0.1008	9.6064	12.0976	2.5203	0.2016
4	50.0982	1.0185	15.277	0.9649	8.0492	0.1018	9.2279	12.2216	2.5462	0.2037
5	50.2003	1.0289	15.4337	0.9561	8.0178	0.1029	8.8486	12.3469	2.5723	0.2058

AY	AZ	BA
density	Ln(viscosit	Melt_%_m
2.683	6.25	99.99
2.682	6.38	98.995
2.681	6.52	97.9904
2.68	6.67	96.9959

```
df_out=ss.import_data('PetrologCalculations.xlsx', Petrolog=True)
df_out.head()
```

Specifying this is a Petrolog3 file

prints reformatted data for inspection

```
We have replaced all missing liquid oxides and strings with zeros.
```

	SiO2_Liq	TiO2_Liq	Al2O3_Liq	FeOt_Liq	MnO_Liq	MgO_Liq	CaO_Liq	Na2O_Liq	K2O_Liq	P2O5_Liq	H2O_Liq	Fe3Fet_Liq
0	49.9010	0.9981	14.9715	8.980926	0.0998	9.9763	11.9772	2.4953	0.1996	0.0998	0.0	0.098489
1	49.9978	1.0081	15.1220	8.951296	0.1008	9.6064	12.0976	2.5203	0.2016	0.1008	0.0	0.097851
2	50.0982	1.0185	15.2770	8.916645	0.1018	9.2279	12.2216	2.5462	0.2037	0.1018	0.0	0.097284

Option 1: Calculate Smythe et al. (2017) SCSS (measured sulf comp)

```
Smythe_FixedSulf=ss.calculate_S2017_SCSS(df=df_out,
T_K=df_out['T_K'], P_kbar=df_out['P_kbar'],
Fe3Fet_Liq=df_out['Fe3Fet_Liq'],
Fe_FeNiCu_Sulf=0.65)
Smythe_FixedSulf.head()
```

Reading melt composition, T, P from dataframe extracted from Petrolog3

Measured sulfide composition

Inspect calculations

All calculation steps returned

```
Using inputted Fe_FeNiCu_Sulf ratio for calculations.
You havent entered a value for Ni_FeNiCu_Sulf and Cu_FeNiCu_Sulf so we cant calculate the non-ideal SCSS
```

	SCSS2_ppm_ideal_Smythe2017	SCSS2_ppm_ideal_Smythe2017_1sigma	Si_XA_ideal	Ti_XA_ideal	Al_XA_ideal	Mg_XA_ideal
0	1163.632126	317.869126	-12643.917846	-77.425761	-2992.954217	-1910.042747
1	1132.187652	309.279446	-12681.596545	-78.282676	-3026.179002	-1841.131776
2	1099.523987	300.356720	-12720.511949	-79.173986	-3060.433090	-1770.461624

Scroll

Option 2: Calculate ONeill (2021) SCSS (meas sulf comp)

```
ONeill_FixedSulf=ss.calculate_O2021_SCSS(df=df_out,
T_K=df_out['T_K'], P_kbar=df_out['P_kbar'],
Fe3Fet_Liq=df_out['Fe3Fet_Liq'],
Fe_FeNiCu_Sulf=0.65)

ONeill_FixedSulf.head()
```

Identical inputs to above, only difference is the function name!

```
Using inputted Fe_FeNiCu_Sulf ratio for calculations.
```

	SCSS2_ppm	LnS	Ln_a_FeO	Ln_a_FeS	DeltaG	LnCS2_calc	SiO2_Liq	TiO2_Liq	Al2O3_Liq	FeOt_Liq	MnO_Liq	MgO_Liq	CaO_Liq
0	1117.435082	7.018791	-2.405572	-0.495104	7.309272	-2.200950	49.9010	0.9981	14.9715	8.980926	0.0998	9.9763	11.9772
1	1085.704562	6.989984	-2.401368	-0.495001	7.373234	-2.289616	49.9978	1.0081	15.1220	8.951296	0.1008	9.6064	12.0976
2	1053.015445	6.959413	-2.397406	-0.494856	7.441489	-2.384626	50.0982	1.0185	15.2770	8.916645	0.1018	9.2279	12.2216

Option 3: Calculate Li & Zhang (2022) SCSS (meas sulf comp)

```
LZ2022_FixedSulf=ss.calculate_LiZhang2022_SCSS(df=df_out,
T_K=df_out['T_K'], P_kbar=df_out['P_kbar'],
Fe3Fet_Liq=df_out['Fe3Fet_Liq'],
Fe_FeNiCu_Sulf=0.65)
LZ2022_FixedSulf.head()
```

Identical inputs to above, only difference is the function name!

Calculate trajectory if no sulfide (S behaving incompatibly)

```
FC=ss.crystallize_S_incomp(S_init=1000, F_melt=df_out['Fraction_melt'])
```

Plot modelled SCSS vs. incompatible FC trajectory with MI data

```
MI_data=pd.read_excel('MI_Data.xlsx')
fig, (ax1) = plt.subplots(1, 1, figsize=(4,3.5))
ax1.plot(df_out['MgO_Liq'], FC,
':k', label='Incompatible Behavoir')
ax1.plot(Smythe_FixedSulf['MgO_Liq'],
Smythe_FixedSulf['SCSS2_ppm_ideal_Smythe2017'], '-r',
label='S2017 SCSS')
ax1.plot(LZ2022_FixedSulf['MgO_Liq'],
LZ2022_FixedSulf['SCSS_Tot'], '-c', label='LZ22 SCSS')
ax1.plot(ONeill_FixedSulf['MgO_Liq'],
ONeill_FixedSulf['SCSS2_ppm'], '-b', label='O2021 SCSS')
ax1.plot(MI_data['MgO_Liq'], MI_data['S_ppm'],
'^k', mfc='yellow', label='MI')
ax1.set_ylabel('S (ppm)')
ax1.set_xlabel('MgO Liq (wt%)')
ax1.legend(fontsize=8)
plt.xlim([4, 10])
plt.ylim([200, 1300])
fig.savefig('SCSS_Models.png', dpi=200, bbox_inches='tight')
```

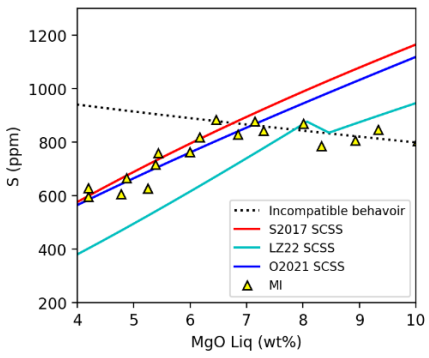

Figure 3: Annotated worked example showing how to calculate $SCSS^{2-}$ for a `Petrolog3` fractional crystallization path using a fixed Fe/(Fe+Ni+Cu) ratio in the sulfide. Hypothetical melt inclusion data is overlain. The data initially follows the incompatible fractional crystallization trend, followed by a prominent downturn, indicating the onset of sulfide saturation at ~ 6–7 wt.% MgO.

2.4 Redox sensitivity

A number of SCSS models are also sensitive to the ratio of Fe^{3+}, because they contain a term for only Fe^{2+} species in the melt (see Figure 1). The input argument `Fe3Fet_Liq` should be supplied when using these models. If no value is entered, calculations are performed assuming $Fe^{3+} = 0$. Alternatively, users can specify a single value in the function (e.g. `Fe3Fet_Liq=0.15`), or refer to a column in the input dataframe. Another option is to use the `Python` package `Thermobar` [Wieser et al. 2022] to convert a log fO_2 value or buffer position into a `Fe3Fet_Liq` ratio. For models which are not redox-sensitive [e.g. Blanchard et al. 2021; Liu et al. 2021], entering a non-zero value for `Fe3Fet_Liq` will not affect the SCSS (except through secondary dependencies, e.g. if the model of Smythe et al. [2017] or O'Neill [2021] is used to calculate the sulfide composition).

2.5 Calculating sulfide proportions

The difference between the fractional crystallization trajectory and the predicted $SCSS^{2-}$ can be used to calculate the cumulative mass proportion of sulfide forming over the fractionation interval [after Kiseeva and Wood 2015]:

$$X_{sulf} = \frac{S_{init} - F_{melt} \times S_{model}}{S_{sulf}}, \quad (4)$$

where S_{init} is the initial S content at the start of the fractional crystallization sequence ($F_{melt} = 1$), F_{melt} is the melt fraction remaining at each step, S_{model} is the modeled solubility of S in the melt, and S_{sulf} is the S content of the sulfide (all concentrations in ppm).

In `PySulfSat`, this is calculated as follows for the example shown in Figure 3:

```
S_Frac=ss.calculate_mass_frac_sulf(
S_model=ONeill_FixedSulf['SCSS2_ppm'],
S_sulf=320000, S_init=800,
F_melt=df_out['Fraction_melt']/100)
```

This calculates the mass fraction of sulfide formed for a magma with 800 ppm S initially, a S content in the sulfide of 32 wt.%, and a melt fraction from the `Petrolog3` file (column heading `Fraction_melt`, obtained from the column `Melt_%_magma` in the `Petrolog3` file by the `PySulfSat` import function).

3 $SCAS^{6+}$ MODELS

In `PySulfSat`, $SCAS^{6+}$ calculations are performed in a very similar way to $SCSS^{2-}$ calculations. For example, to calculate $SCAS^{6+}$ for the Petrolog3 model loaded in as `df_out` using the model of Chowdhury and Dasgupta [2019]:

```
CD19_SCAS=ss.calculate_CD2019_SCAS(df=df_out,
T_K=df_out['T_K'])
```

The calculation could also be performed using the $SCAS^{6+}$ model of Zajacz and Tsay [2019]:

```
ZT22_SCAS=ss.calculate_ZT2022_SCAS(df=df_out,
T_K=df_out['T_K'])
```

As for $SCSS^{2-}$ models, these functions return the calculated $SCAS^{6+}$, all intermediate calculations, and the originally loaded compositions. The main simplification relative to SCSS models is the fact that none of the existing SCAS models have a term for the composition of the sulfate-bearing phase, pressure, or the Fe^{3+}/Fe_T ratio (Figure 1).

4 MAGMAS WITH A MIX OF S^{2-} AND S^{6+}

Silicate melts undergo a relatively abrupt transition in S speciation from sulfide (S^{2-}) to sulfate (S^{6+}) dominated with increasing oxygen fugacity [Fincham and Richardson 1954; Wallace and Carmichael 1994; Jugo et al. 2010; Kleinsasser et al. 2022, ; cyan line in Figure 4B]. In systems where both S^{2-} and S^{6+} are present, the calculated $SCSS^{2-}$ will underestimate the total solubility of S, because this parameter only accounts for the solubility of S^{2-} species. Similarly, in systems dominated by S^{6+} with some S^{2-}, the total solubility of S will exceed the $SCAS^{6+}$ [Jugo 2009].

4.1 Correcting for S^{6+} and S^{2-}

4.1.1 *Demonstrating the importance of S^{2-} and S^{6+} corrections*

To demonstrate the importance of accounting for both S^{2-} and S^{6+} species when modeling total S solubility, lets consider a melt with an $SCSS^{2-}$ of 1000 ppm, and an $SCAS^{6+}$ of 5000 ppm. Equation 10 of Jugo et al. [2010] can be used to calculate the proportion of S^{6+}/S_T as a function of ΔQFM between −1 and +3:

$$\frac{S^{6+}}{S_T} = \frac{1}{1 + 10^{2.1-2\Delta FMQ}}. \quad (5)$$

This equation can be implemented in `PySulfSat` for a single ΔQFM value as follows:

```
S6St_03=ss.calculate_S6St_Jugo2010_eq10(deltaQFM=0.3)
= 0.030653430031715508
```

To produce the cyan line on Figure 4B, we input a linearly-spaced `numpy` array of 10,001 points between ΔQFM = −1 and ΔQFM = 3 generated using the `np.linspace` function, and calculate S^{6+}/S_T for every value in this array (cyan line, Figure 4B).

```
deltaQFM=np.linspace(-1, 3, 10001)
S6St=ss.calculate_S6St_Jugo2010_eq10(
deltaQFM=deltaQFM)
```

At ΔQFM = −1 (point 1 on Figure 4B), the melt is sufficiently reduced that only S^{2-} is dissolved in meaningful quantities (S^{6+}/S_T= 0.00008). Thus, the total solubility of sulfur is well approximated by the $SCSS^{2-}$ (1000 ppm for this specific example, horizontal magenta line on Figure 4A).

For a moderately oxidized melt at ΔQFM = 1 (Point 2), S^{6+}/S_T = 0.442, so the presence of S^{6+} species substantially increases the total amount of S that is dissolved. Thus, the

$SCSS^{2-}$ must be corrected to obtain the $SCSS_T$ using the equation of Jugo et al. [2010]:

$$SCSS_T = \frac{SCSS^{2-}}{1 - \left(\frac{S^{6+}}{S_T}\right)}. \quad (6)$$

In `PySulfSat` this is implemented as follows:

```
S6=ss.calculate_S6St_Jugo2010_eq10(
deltaQFM=1)
SCSS_Tot=ss.calculate_SCSS_Total(SCSS=1000,
S6St_Liq=S6)
= 1794
```

The $SCSS_T$ comprises with 1000 ppm of S^{2-}, and 794 ppm of S^{6+} (see red and grey lines on Figure 4B).

At ΔQFM = 1.39 (Point 3), S^{6+}/S_T = 0.827. Using Equation 6, the $SCSS_T$ is 5786 ppm, with 1000 ppm of S^{2-}, and 4786 ppm of S^{6+}. However, if ΔQFM (and therefore S^{6+}/S_T) increases slightly more, Equation 6 becomes invalid, because the amount of predicted S^{6+} exceeds the $SCAS^{6+}$ (dashed magenta line, Figure 4A). For example, at point 4 (ΔQFM = 2), S^{6+}/S_T = 0.9876. Equation 6 would predict that the $SCSS_T$ is 80,433 ppm, with 1000 ppm of S^{2-}, and 79,433 ppm of S^{6+}. However, this much S^{6+} cannot dissolve, because the $SCAS^{6+}$ is only 5000 ppm.

Instead of correcting the SCSS for S^{6+}, in more oxidising magmas, we can also correct the $SCAS^{6+}$ for the presence of S^{2-}:

$$SCAS_T = \frac{SCAS^{6+}}{1 - \left(\frac{S^{2-}}{S_T}\right)}. \quad (7)$$

For example, at point 4:

```
SCAS_Tot=ss.calculate_SCAS_Total(SCAS=5000,
S6St_Liq=0.9876)
=5063
```

This total dissolved S comprises 5000 ppm of S^{6+} and 63 ppm of S^{2-}. However, if this equation were applied to point 2, it would predict more dissolved S^{2-} than the SCSS. These worked examples demonstrate that at certain proportions of S^{6+} to S^{2-}, Equation 7 and Equation 6 are invalid for predicting the total solubility of S. For the specific $SCSS^{2-}$ and $SCAS^{6+}$ values used in this example, ΔQFM = ~1.4 is the oxygen fugacity where the maximum amount of S dissolves in the system, because at this ΔQFM value, the ratio of S^{6+}/S_T is such that the amount of S^{2-} dissolved is equal to the $SCSS^{2-}$, and the amount of S^{6+} is equal to the $SCAS^{6+}$ (yielding the maximum possible sum of these two values).

The total amount of dissolved S in ΔQFM space that does not violate the calculated $SCSS^{2-}$ and $SCAS^{6+}$ is defined by the section of the $SCSS_T$ curve where S^{6+} does not exceed the $SCAS^{6+}$ (magenta solid line, Figure 4A), and the section of the $SCAS_T$ curve where S^{2-} does not exceed the $SCSS^{2-}$ (black solid line, Figure 4A). The combined curve meeting these requirements is shown as a green line in Figure 4B.

In `PySulfSat`, for any calculated $SCSS^{2-}$ and $SCAS^{6+}$ values, the total amount of S can be calculated using the function `calculate_S_Total_SCSS_SCAS`. This can be used to produce plots of changing S speciation with fO_2 (e.g. Figure 4).

For example, using 11 equally spaced ΔQFM values between −1 and 3 (−1, −0.6, −0.2...), we can calculate the total solubility of S using the model of Jugo et al. [2010], for a fixed $SCSS^{2-}$ (1000 ppm) and $SCAS^{6+}$ value (5000 ppm):

```
deltaQFM_lin=np.linspace(-1, 3, 10)
df_S_Jugo=ss.calculate_S_Total_SCSS_SCAS(
deltaQFM=deltaQFM_lin,
SCSS=1000, SCAS=5000, model='Jugo')
```

This function returns a `pandas` dataframe:

	Total_S_ppm	S2_Tot_ppm	S6_Tot_ppm	deltaQFM	S6St_Liq	SCSS_2_ppm	SCAS_6_ppm
0	1000.079433	1000.000000	0.079433	-1.000000	0.000079	1000	5000
1	1000.615020	1000.000000	0.615020	-0.555556	0.000615	1000	5000
2	1004.761873	1000.000000	4.761873	-0.111111	0.004739	1000	5000
3	1036.869451	1000.000000	36.869451	0.333333	0.035558	1000	5000
4	1285.466766	1000.000000	285.466766	0.777778	0.222072	1000	5000

The column `Total_S_ppm` shows the total amount of S dissolved in ppm, with an amount of S^{2-} indicated by the column `S2_Tot_ppm` and S^{6+} by the column `S6_Tot_ppm`. The input SCSS and SCAS are also shown in columns `SCSS_2_ppm` and `SCAS_6_ppm`; the values in the columns `S2_Tot_ppm` and `S6_Tot_ppm` will always be less than or equal to these values.

In addition to the Jugo et al. [2010] model which calculates S^{6+}/S_T in terms of ΔQFM, calculations can also be performed in `PySulfSat` using the Nash et al. [2019] model, which parameterizes S^{6+}/S_T in terms of the ratio of Fe^{3+} to Fe^{2+} and temperature (in Kelvin):

$$\log\left(\frac{S^{6+}}{S^{2-}}\right) = 8\log\left(\frac{Fe^{3+}}{Fe^{2+}}\right) + \frac{8.7436 \times 10^6}{T^2} - \frac{27703}{T} + 20.273. \quad (8)$$

To calculate S^{6+}/S_T using this model, the temperature in Kelvin and the ratio of Fe^{3+}/Fe_T must be input:

```
Calc_Nash_S6=ss.calculate_S6St_Nash2019(
T_K=df_out['T_K'], Fe3Fet_Liq=df_out['Fe3Fet_Liq'])
```

When calculating the Total S content, specify `model='Nash'` rather than `model='Jugo'` in the function `calculate_S_Total_SCSS_SCAS`:

```
deltaQFM_lin=np.linspace(-1, 3, 11)
df_S_Nash=ss.calculate_S_Total_SCSS_SCAS(
SCSS=1000, SCAS=5000,
model='Nash', T_K=df_out['T_K'],
Fe3Fet_Liq=df_out['Fe3Fet_Liq'])
```

Kleinsasser et al. [2022] note that the transition predicted by models primarily calibrated on mafic melts [e.g. Jugo et al. 2010; Nash et al. 2019] is not a good match for dacitic melt compositions, where the transition occurs at higher fO_2 values (ΔQFM = +1.81 ± 0.56). They provide two expressions for correcting the $SCSS^{2-}$ and $SCAS^{6+}$:

$$SCSS_T^{dacitic} = SCSS^{2-} * (1 + 10^{2\Delta QFM - 3.05})$$
$$SCAS_T^{dacitic} = SCAS^{6+} * (1 + e^{1.26 - 2\Delta QFM}). \quad (9)$$

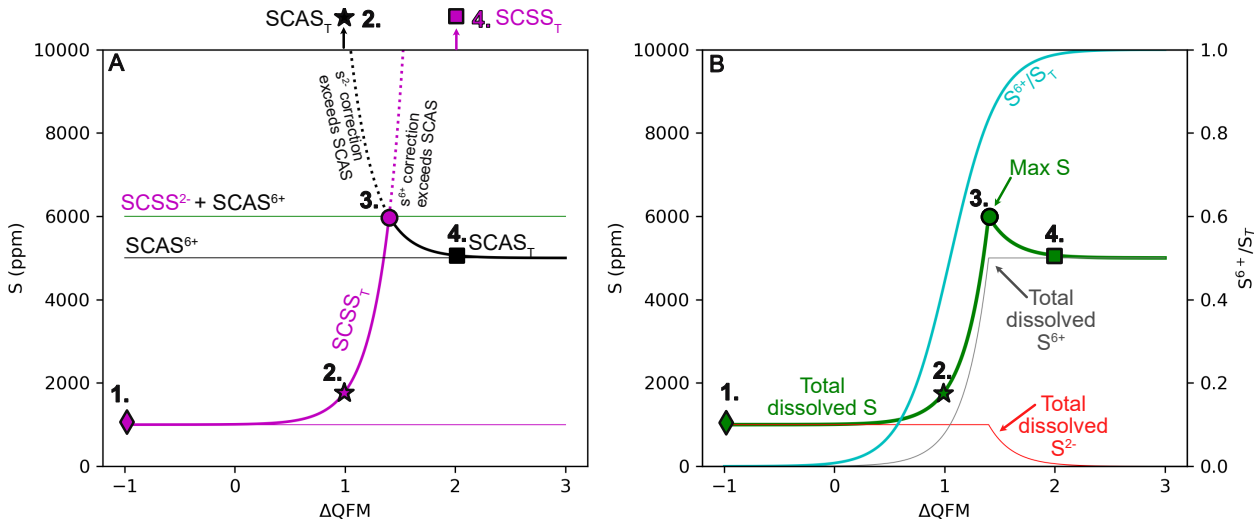

Figure 4: Calculating the total amount of dissolved S by applying corrections for the presence of both S species using the model of Jugo et al. [2010] in the function `calculate_S_Total_SCSS_SCAS`. These graphs were drawn for $SCSS^{2-}$ = 1000 ppm and $SCAS^{6+}$ = 5000 ppm, although these numbers could be calculated using any SCSS and SCAS model in `PySulfSat`.

This parameterization can also be used in `PySulfSat`, by specifying `model='Kleinsasser'`:

```
deltaQFM_lin=np.linspace(-1, 3, 11)
df_S_Klein=ss.calculate_S_Total_SCSS_SCAS(
deltaQFM=deltaQFM_lin,
SCSS=1000, SCAS=5000,
model='Kleinsasser')
```

4.1.2 *Calculating S^{6+}/S_T from the Sulfate and Sulfide capacity*

In addition to the methods described above where the proportion of S species is estimated from oxygen fugacity or Fe^{3+}/Fe_T, the ratio of S^{6+}/S_T can also be calculated using the method of O'Neill and Mavrogenes [2022]. This approach calculates the sulfide capacity ($C_{S^{2-}}$) using the parameterization of O'Neill [2021], and the sulfate capacity ($C_{S^{6+}}$) using O'Neill and Mavrogenes [2022]. The equilibrium constant for the gas-phase equilibrium, lnK, is then calculated using T in Kelvin:

$$\ln(K) = -55921/T + 25.07 - 0.6465 \times \ln(T). \quad (10)$$

These values are then used to calculate S^{6+}/S^{2-}, which can be easily converted into a S^{6+}/S_T ratio:

$$\ln\left(\frac{S^{6+}}{S^{2-}}\right) = \ln\left(C_{S^{6+}}\right) - \ln(K) - \ln\left(C_{S^{2-}}\right) + 2\ln(10) \times \log fO_2, \quad (11)$$

and:

$$\frac{S^{6+}}{S_T} = 1 - \frac{1}{1 + e^{\ln\left(\frac{S^{6+}}{S^{2-}}\right)}}. \quad (12)$$

The O'Neill and Mavrogenes [2022] supporting spreadsheet also provides an option to input Fe^{3+}/Fe_T ratio instead of a value for $\log fO_2$. The spreadsheet uses this ratio to calculate ΔQFM using an adapted version of Eq9a of O'Neill et al. [2018] (missing the term for P_2O_5, as this oxide is not included in their capacity models):

$$\Delta QFM = 4\left(\log\left(\frac{\frac{Fe^{3+}}{Fe_T}}{1 - \frac{Fe^{3+}}{Fe_T}}\right) + 1.36 - 2 * X_{Na} - 3.7X_K - 2.4X_{Ca}\right), \quad (13)$$

Where X_{Na}, X_K, and X_{Ca} are the cation fractions of Na, K, and Ca in the melt. This ΔQFM value is then converted into $\log fO_2$ using Eq8 of O'Neill et al. [2018] based on O'Neill [1987] to input into Equation 12:

$$\log_{10} fO_2 = \Delta QFM - 25050/T + 8.58. \quad (14)$$

Where T is in Kelvin.

These equations are all implemented in `PySulfSat` through the function `calculate_OM2022_S6St`. For example, to perform calculations using a known $\log fO_2$ value:

```
Calc_OM2022=ss.calculate_OM2022_S6St(df=df_out,
T_K=df_out['T_K'], logfo2=df_out['logfo2'])
```

Alternatively, if the Fe^{3+}/Fe_T ratio is stored in a column in the input dataframe:

```
Calc_OM2022=ss.calculate_OM2022_S6St(df=df_out,
T_K=df_out['T_K'], Fe3Fet_Liq=df_out['Fe3Fet_Liq'])
```

which returns a pandas dataframe:

	S6St_Liq	LnCS2_calc	LnKSO2S2	LnS6S2	deltaQFM_calc	Sample ID
0	0.009061	-2.252326	-18.676809	-4.694696	0.385382	VG175
1	0.018402	-2.187329	-18.757549	-3.976697	0.549923	180
2	0.016171	-2.276202	-18.567275	-4.108246	0.468580	183
3	0.013897	-2.376010	-18.501180	-4.262089	0.417146	186
4	0.051541	-2.345865	-18.557150	-2.912471	0.721690	187

Boulliung and Wood [2022] also publish an equation to calculate log $C_{S^{6+}}$. While related to the ln $C_{S^{6+}}$ value of O'Neill and Mavrogenes [2022], this is not simply a log–ln conversion. Boulliung and Wood [2022] express their S content in wt percent, rather than ppm, and their equilibrium constant refers to a different equation. These values can be converted from one form to another (see ReadTheDocs for a derivation). In `PySulfSat`, the function `calculate_BW2022_CS6` returns a dataframe for columns named `'LogCS6_calc_BW22_format'` which uses the Boulliung and Wood [2022] format, and `'LnCS6_calc_OM22_format'` which uses the format of O'Neill and Mavrogenes [2022]. This allows direct comparison between models. We also include the function `calculate_BW2022_OM2022_S6St` to calculate S^{6+}/S_T using $C_{S^{6+}}$ from Boulliung and Wood [2022] and $C_{S^{2-}}$ from O'Neill [2021].

4.1.3 *Calculations for natural samples*

When calculating the total solubility of S in a natural system with a non negligible proportion of both S species, using the function `calculate_S_Total_SCSS_SCAS` ensures that the correction has not exceeded the solubility of either species, unlike functions correcting the SCSS for S^{6+} using `calculate_SCSS_Total`, or SCAS for S^{2-} using `calculate_SCAS_Total`.

When comparing measured S contents to total S solubility obtained from SCSS and SCAS models, it is most reliable to use measured S^{6+}/S_T ratios (e.g. using X-ray absorption near edge structure—XANES [Lerner et al. 2021]). In this ideal scenario, users can enter the measured ratio directly in the `calculate_S_Total_SCSS_SCAS` function. For example, after calculating the SCSS using Smythe et al. [2017] (saved in df=S2017) and the SCAS using Zajacz and Tsay [2019] (saved in df=Z2019), the total amount of dissolved S can be calculated using a fixed S^{6+}/S_T ratio of 0.2:

```
Tot_S_S17_Z19=ss.calculate_S_Total_SCSS_SCAS(
SCSS=S2017['SCSS2_ppm_ideal_Smythe2017'],
SCAS=Z2019['SCAS6_ppm'],
S6St_Liq=0.2)
```

Alternatively, it is more common that Fe^{3+}/Fe_T has been constrained using XANES. Using the Nash et al. [2019] correction, this Fe^{3+}/Fe_T ratio can be entered directly to calculate the S^{6+}/S_T ratio, and thus the maximum amount of S that can dissolve:

```
Tot_S_S17_Z19_Nash=ss.calculate_S_Total_SCSS_SCAS(
SCSS=S2017['SCSS2_ppm_ideal_Smythe2017'],
SCAS=Z2019['SCAS6_ppm'],
Fe3Fet_Liq=0.15, model='Nash', T_K=df['T_K'])
```

For consistency, in this example, the S2017 dataframe should also have been calculated using the same input value for `Fe3Fet_Liq=0.15`.

To use the Jugo et al. [2010] correction, the redox state of the magma must be calculated relative to the QFM buffer position of Frost [1991] (see Section 4.2. If Fe^{3+}/Fe_T is known, this can be converted into a log fO_2 value using Kress and Carmichael [1988] using the `Python` package `Thermobar` [Wieser et al. 2022]. Once a log fO_2 value is calculated, Thermobar can then be used to calculate the offset from the QFM buffer position (i.e. ΔQFM). Alternatively, the log fO_2 may be known independently without having to do a conversion based on Fe^{3+}/Fe_T first. For example, the `Petrolog3` output in Figure 3 has a column for the log of the fO_2 value, the temperature, and the pressure.

```
!pip install Thermobar
import Thermobar as pt
Buffer_calc=pt.convert_fo2_to_buffer(
fo2=10**df_out['Lg(fO2)'],
T_K=df_out['T_K'],  P_kbar=df_out['P_kbar'])
```

	deltaNNO_Frost1991	deltaQFM_Frost1991	QFM_equation_Choice	T_K	P_kbar	fo2	Cut off T (K)
0	-0.777890	-0.085829	High T	1526.431	1	1.905461e-08	871.15
1	-0.781999	-0.089502	High T	1516.580	1	1.479108e-08	871.15
2	-0.779077	-0.086116	High T	1506.214	1	1.148154e-08	871.15
3	-0.770841	-0.077393	High T	1495.511	1	8.912509e-09	871.15
4	-0.774373	-0.080406	High T	1484.230	1	6.606934e-09	871.15

The different buffers stored in the `Buffer_calc` dataframe can then be input into the `PySulfSat` function:

```
ss.calculate_S_Total_SCSS_SCAS(
deltaQFM=Buffer_calc['DeltaQFM_Frost1991'],
SCSS=S2017['SCSS2_ppm_ideal_Smythe2017'],
SCAS=Z2019['SCAS6_ppm'],
T_K=df_out['T_K'],
model='Jugo')
```

Alternatively, if you have an estimate of fO_2 you can use the O'Neill and Mavrogenes [2022] method:

```
ss.calculate_S_Total_SCSS_SCAS(
df=df_out,
logfo2=df_out['Lg(fO2)'],
SCSS=S2017['SCSS2_ppm_ideal_Smythe2017'],
SCAS=Z2019['SCAS6_ppm'],
T_K=df_out['T_K'],
model='OM2022')
```

This function can also take Fe^{3+}/Fe_T as input, although our code (and the published spreadsheet) convert this into a log(fO_2) value using Equation 13.

4.2 Different buffer positions and melt redox models

It is important to recognize the uncertainty introduced into calculations of S^{6+} proportions as a result of different definitions of buffer positions, melt redox models, and XANES data processing strategies. For example, the ΔQFM values for the Jugo et al. [2010] S^{6+} correction should be relative to the QFM

buffer position of Frost [1991]. Petrolog3 uses the expression of Myers and Eugster [1983] for its QFM buffer. alphaMELTS (including MELTS for MATLAB and Python [Antoshechkina and Ghiorso 2018]) and MELTS for Excel [Gualda and Ghiorso 2015] also use Myers and Eugster [1983], with an additional pressure correction from Frost [1991]. Expressing all these different QFM buffer positions in terms of $\log(fO_2)$ values at QFM yields the following equations:

$$\log fO_{2\ \text{at QFM [Frost 1991]}} = \frac{-25,096.3}{T} + 8.735 + 0.11\frac{P-1}{T}; \quad (15)$$

$$\log fO_{2\ \text{at QFM [O'Neill et al. 2018]}} = \frac{-25,050}{T} + 8.58; \quad (16)$$

$$\log fO_{2\ \text{at QFM (Petrolog3)}} = \frac{-24,442}{T} + 8.29; \quad (17)$$

$$\log fO_{2\ \text{at QFM (MELTS)}} = \frac{-24,442}{T} + 8.29 + 0.11 * \frac{(P-1)}{T}. \quad (18)$$

where P is in bars and T is in Kelvin. If users have a ΔQFM value relative to a buffer which is not Frost [1991], they need to convert that into a value relative to the Frost [1991] prior to using Jugo et al. [2010]. To demonstrate the importance of performing these conversions, lets consider a melt at 1050 °C and 200 MPa. Say a user has obtained a buffer position of ΔQFM+1 relative to Equation 16. If this ΔQFM value was entered directly into Jugo et al. [2010], it would yield 45 % S^{6+}. However, this buffer position should first be used to calculate the $\log(fO_2)$ value (−9.35), to then calculate the ΔQFM relative to Frost [1991] (ΔQFM = 0.71). This yields only 18 % S^{6+}. This shows the importance of maintaining consistency with the buffer position used to calibrate Jugo et al. [2010].

There are a variety of methods to convert $\log(fO_2)$ values into Fe^{3+}/Fe_T ratios [see Putirka 2016], which can introduce uncertainty when using the Nash et al. [2019] method, or when inputting ratios directly into the O'Neill and Mavrogenes [2022] method. For example, Petrolog3 allows users to choose between the models of Sack et al. [1981], Kilinc et al. [1983], Kress and Carmichael [1988], and Borisov and Shapkin [1990]. For the default Petrolog3 composition at ΔQFM=0, atmospheric pressure and the liquidus position, these 4 models return Fe^{3+} proportions between 10 and 14 %, which corresponds to S^{6+} proportions using Nash et al. [2019] of 1.2–24 %. Of course, offsets between the selected definition of the QFM buffer (O'Neill-Petrolog3-Frost) will also affect the Fe^{3+} proportions calculated by different melt redox models (through influencing the $\log(fO_2)$ value).

The O'Neill and Mavrogenes [2022] method for calculating S^{6+} proportions is parameterized directly in terms of $\log(fO_2)$, so when pairing this model with various petrology modeling software tools, the easiest way to avoid mixing and matching buffer definitions/melt redox models is to directly input this parameter. If Fe^{3+}/Fe_T is entered, this is converted to $\log(fO_2)$ using O'Neill et al. [2018] Eq8 and 9b (Equation 13 and 14 in this paper). This will return a different $\log(fO_2)$ to that outputted directly by MELTS/Petrolog3 (which use models other than O'Neill et al. [2018] to convert $\log(fO_2)$ to Fe^{3+}/Fe_T). When Fe^{3+}/Fe_T is measured directly by XANES, it is worth considering additional compilations resulting from different calibration strategies. For example, O'Neill and Mavrogenes [2022] make sure to correct the Fe XANES measurements of Brounce et al. [2017] and Muth and Wallace [2021] using the method of Berry et al. [2018] prior to performing calculations of S^{6+} proportions. However, we find that the S XANES measurements of Muth and Wallace [2021] are best matched by the O'Neill and Mavrogenes [2022] if measured Fe^{3+}/Fe_T ratios are input, rather than ratios corrected using Berry et al. [2018]. More comparisons are clearly required to see if this is a one-off occurrence. In many instances, offsets between different Fe^{3+}/Fe_T XANES reduction methods, and Fe^{3+}/Fe_T–$\log(fO_2)$ conversion strategies should perhaps be considered as true error on these methods, given the lack of community consensus [Anenburg and O'Neill 2019].

5 Monte Carlo error propagation

In addition to simplifying calculations and aiding model comparisons, PySulfSat also allows users to propagate uncertainty in input parameters for all calculation types using Monte Carlo methods. There are two main workflows that can be used. First, if errors are known for every input variable, users should load in two dataframes. The first dataframe (df1) should contain the preferred value for each input parameter (e.g. columns MgO_Liq, FeOt_Liq, H2O_Liq). The second dataframe (df2) should have exactly the same column headings with the addition of the suffix _Err. These columns can contain absolute or percentage errors. Additional columns (e.g. temperatures calculated using Thermobar [Wieser et al. 2022]) can be appended onto df1 in the Jupyter Notebook itself, along with an appropriate error in df2. The function add_noise_2_dataframes can then be used to duplicate each input row in the input dataframe df_values N_dups times, adding noise based on the value in the dataframe with errors (df_err). For example, to add normally distributed errors using absolute 1σ values from df2, and create 5000 duplicates for each sample:

```
df_noisy=ss.add_noise_2_dataframes(
df_values=df1, df_err=df2,
error_type="Abs", error_dist="normal",
N_dups=5000)
```

This new dataframe is then entered into any of the functions.

In Figure 5 we use the add_noise_2_dataframes function to generate 5000 synthetic compositions for each melt inclusion, with errors from quoted 1σ values for each variable from Muth and Wallace [2021]. These synthetic compositions were then input into the various functions to calculate S^{6+}/S_T ratios. Finally, the function av_noise_samples_series is used to calculate statistics for each melt inclusion. Users should input a panda.Series containing the variable of interest into this function as the arguement **var** (in this case, the calculated S^{6+}/S_T ratio stored in the dataframe ONeill_S6ST), and a second pandas.Series with the sample names to average over (as sampleID).

```
Stats_S6=pt.av_noise_samples_series(
calc=ONeill_S6ST['S6St_Liq'],
sampleID=ONeill_S6ST['Sample_ID_Liq'])
Stats_S6.head()
```

For example, the first row in the output averages all 5000 simulations for rows with the sample name `BBL-5-32`.

	Sample	# averaged	Mean_calc	Median_calc	St_dev_calc	Max_calc	Min_calc
0	BBL-5-32	5000	0.031770	0.015473	0.046489	0.642037	9.926852e-06
1	BBL-5-33	5000	0.449445	0.443940	0.212362	0.949757	1.100763e-02
2	BBL-5-34	5000	0.143123	0.099256	0.134041	0.813120	4.726367e-04
3	BBL-5-43	5000	0.131442	0.086405	0.129857	0.839532	1.181191e-04
4	BBL-5-44	5000	0.018558	0.007207	0.031935	0.403228	7.200104e-08

This function calculates the mean, median, max, min and standard deviation of all 5000 simulations for each melt inclusion (which are used to plot symbols and error bars on Figure 5F–H). Simulations can be conducted for any of the calculations available in `PySulfSat` (e.g. SCSS, SCAS, K_D, etc.).

A second set of functions can be useful when you want to explore noise in a smaller number of input variables (e.g. just T, T and H_2O), or where some errors are absolute, some are percentages, some are normal and some are uniformly distributed. The function `duplicate_dataframe` takes a dataframe and duplicates the values in each row `N_dup` times (row1-row2-row3 goes to row1-row1-row1..., row2-row2-row2 ...):

```
Dupdf=ss.duplicate_dataframe(df=df1, N_dup=5000)
```

Then the function `add_noise_series` can be used to create a panda.Series of noise for one specific variable with the same length as this larger dataframe. For example, EDS measurements in a suite of lavas may reveal the sulfide composition for sample 1 is Fe/(Fe+Ni+Cu) = 0.65, and sample 2 is 0.8, with an error of ± 0.05 (stored in the column `df1['Sulf_X']`. Here, we add normally distributed noise, with 5000 duplicates for each input (to match the dataframe above).

```
sulf_comp_err=ss.add_noise_series(var=df1['Sulf_X'],
error_var=0.05, error_type="Abs",
error_dist="normal", N_dup=5000)
```

For example, the first 5000 rows in this new pandas.Series may read 0.64, 0.65, 0.67, 0.65...N_{5000}, and the next 5000 rows may read 0.81, 0.8, 0.79, 0.82....N_{5000}. The total length is the number of rows input multiplied by the number of duplicates, which is the same as the duplicated dataframe. Thus, this new pandas.Series can be appended onto this dataframe as a column:

```
Dupdf['Sulf_MC']=sulf_comp_err
```

As many 'noisy' columns can be added as the user wishes, with different error types and distributions. This dataframe where some columns have noise added and some do not can then be input into any of the `PySulfSat` functions.

6 Integration with MELTS

While `PySulfSat` can load the results from a `MELTS` calculation as a .tbl file, recent advances in the `MELTS` computing infrastructure means that `MELTS` fractional crystallization calculations can be performed directly in `Python` in the same `Jupyter` Notebook as `PySulfSat` calculations. There are currently two options for performing `MELTS` calculations in `Python`; `Thermoengine` [Johnson et al. 2022] and `alphaMELTS` for `Python` [Antoshechkina and Ghiorso 2018]. We make use of the `PyMELTScalc Python` package [Gleeson et al. 2023], which provides neatly wrapped functions for fractional crystallization using `alphaMELTS` for `Python`, and returns output structures consistent with the required inputs for `PySulfSat`.

After installing `PyMELTScalc` (see example on ReadTheDocs), this package must be imported into the notebook:

```
import pyMELTScalc as M
```

After loading data using the `ss.import_data` function as `df_out`, a specific melt composition can be selected as a starting composition (here, we select the first row):

```
sample=df_out.iloc[0]
```

Then, a `MELTS` fractional crystallization model can be initiated at a single pressure using the `multi_path` function:

```
MELTS_FC=M.multi_path(
model="MELTSv1.0.2",
comp = sample,
P_bar = 1000,
find_liquidus = True,
T_end_C = 750,
dt_C = 5,
Fe3Fet_Liq=0.1,
Frac_solid = True,
Frac_fluid = True)
```

This runs a fractional crystallization model at 1000 bars (`P_bar`), starting at the wet liquidus (`find_liquidus=True`), and runs until 750 °C (`T_end_C`). If the `MELTS` calculation does not converge after 100 quadratic minimisation attempts, the simulation may end at a higher temperature. The temperature step is 5 °C (`dt_C`), the initial `Fe3Fet_Liq` ratio is set at 0.1, and both fluids and solids are fractionated.

This `multi_path` function outputs a dictionary containing a series of dataframes. There is a dataframe for each phase, but most relevant for this work is the dataframe named 'All'. This contains all the relevant outputs stitched together, and can be obtained from the overall output as follows:

```
MELTS=MELTS_FC['All']
```

This dataframe named `MELTS` contains system properties (T, P, enthalpy, entropy, volume) and the composition of each phase with the phase name as an underscore (e.g. `SiO2_Liq`, `SiO2_Plag` etc.). It can be fed directly into the `PySulfSat` code. For example, lets use the model of Li and Zhang [2022] for a specified sulfide composition:

```
LiZhang22=ss.calculate_LiZhang2022_SCSS(df=MELTS,
T_K=MELTS['T_C']+273.15,
P_kbar=MELTS['P_bar']/1000,
H2O_Liq=MELTS['H2O_Liq'],
Fe_FeNiCu_Sulf=0.6,
Fe3Fet_Liq=MELTS['Fe3Fet_Liq'])
```

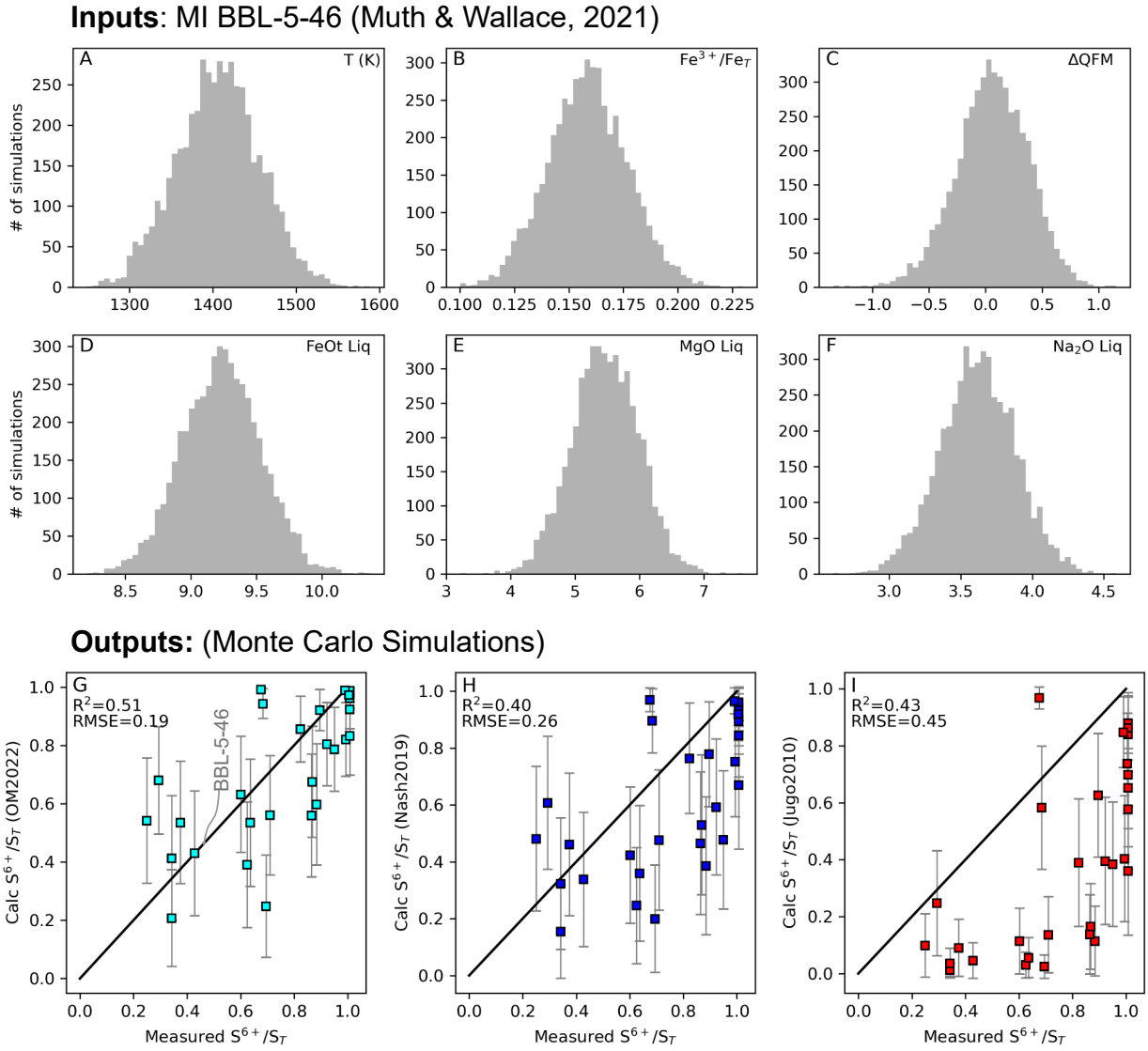

Figure 5: Using Monte Carlo simulations to investigate errors associated with different methods of calculating S^{6+}/S_T ratios. For each melt inclusion, 5000 synthetic compositions were generated using quoted 1σ values from Muth and Wallace [2021] (distributions for MI BBL-5-46 are shown in [A]–[F]). In [H]–[I], we show 1σ errors for each method of calculating S^{6+}/S_T. A detailed worked example showing how to produce this figure can be found at ReadTheDocs.

`PyMELTScalc` can also be used to investigate a wide range of different fractional crystallization paths using parallel processing for computational efficiency, with hundreds to thousands of different fractional paths initiated with a single function call. For example, coupling of `PyMELTScalc` and `PySulfSat` would allow users to investigate S behavior during fractional crystallization for a single melt or range of melt compositions over a wide variety of different starting pressures, oxygen fugacities, and melt water contents. Figure 6 shows the $SCSS^{2-}$ calculated for fractional crystallization models run at 4 different pressures from a single call to the `PyMELTScalc multi_path` function. `PyMELTScalc` can run calculations at a redox buffer or unbuffered, so calculations can be implemented with the various options for the treatment of S^{6+} to investigate changes in S speciation during fractional crystallization.

7 Mantle melting calculations

Modeling the concentrations of S, Cu and other chalcophile elements during mantle melting is complicated by the fact that these elements are held in silicate minerals and mantle sulfides. Because mantle melts contain higher S contents than the mantle residue, the mantle becomes more and more depleted in sulfide during progressive melting until the sulfide phase is eventually exhausted [Lee et al. 2012; Ding and Dasgupta 2018; Wieser et al. 2020]. Exhaustion of sulfide in the mantle

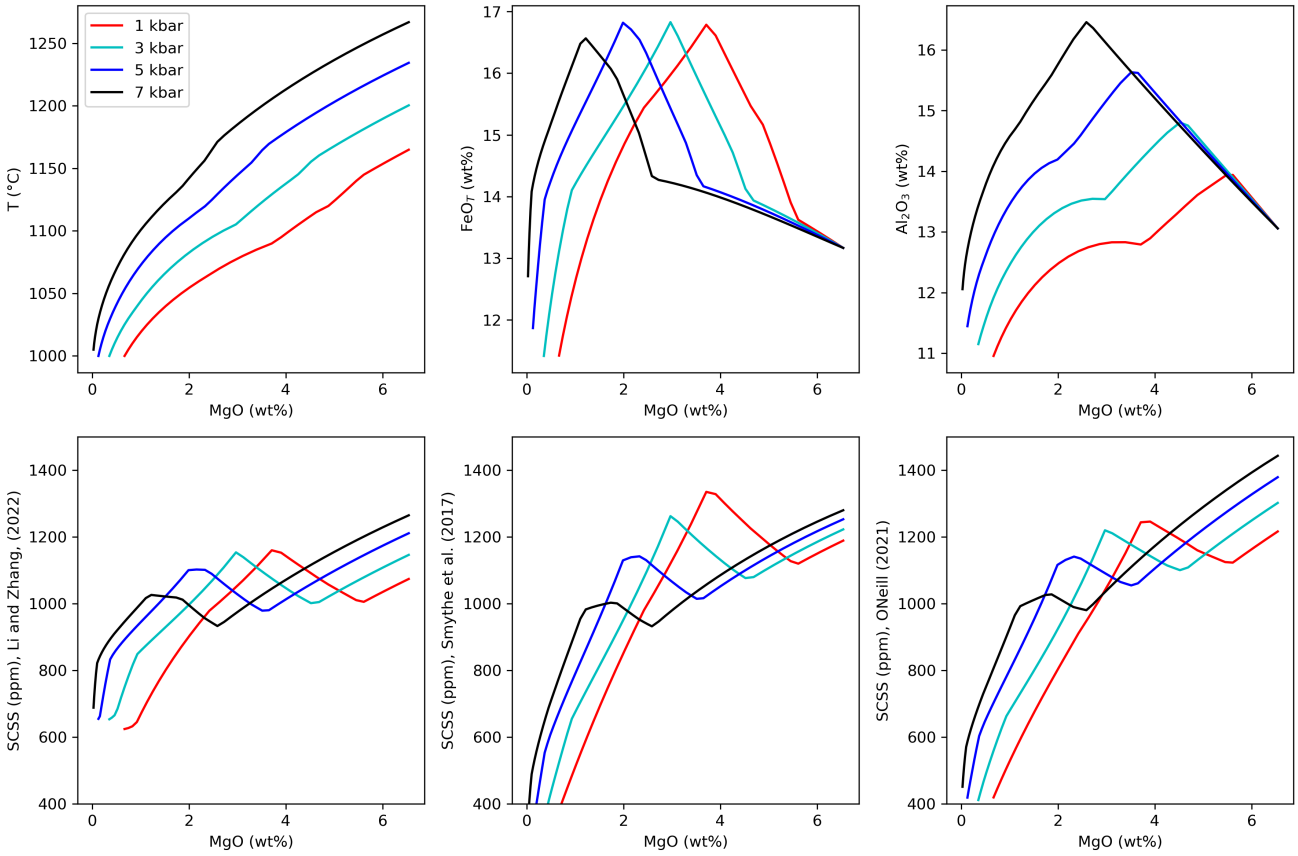

Figure 6: Integrating `PyMELTScalc` and `PySulfSat` to model the SCSS for a fractional crystallization at 4 different pressures. Worked examples showing how to produce this and other similar plots are available on the ReadTheDocs page.

residue drives a large change in the bulk partition coefficient of chalcophile elements during the melting interval.

Lee et al. [2012] provide an `Excel` spreadsheet for calculating the concentration of Cu during near-fractional melting. This model removes small batch melts, updating the composition of the remaining mantle residue before the next melting step proceeds. The equation for batch melting is as follows:

$$\frac{C_{\text{melt}}}{C_{\text{source}}} = \frac{1}{D_0 + F(1-P)} \tag{19}$$

Where C_{melt} is the concentration in the melt, C_{source} is the concentration in the mantle source, D_0 is the bulk partition coefficient (sulfide+silicate) at the start of that melting step, F is the degree of melt produced in that melt step, and P is the bulk partition coefficient weighted for the proportion that each component enters the melt. For simplicity, Lee et al. [2012] assume that $D_0 = P$ (e.g. sulfide and silicate minerals melt at the same rate). Wieser et al. [2020] update this model to account for non-modal melting behavior, because the sulfide preferentially melts, so contributes more to the partition coefficient of highly chalcophile elements such as Cu than the silicates. It should be noted that at a small enough step size (i.e. small enough ΔF), the results from these two approaches converge. However, using the limited number of columns supplied in the spreadsheet of Lee et al. [2012], the divergence can be several 10s of ppm at a given extent of melting (F).

We implement the non-modal melting version of Wieser et al. [2020] in `PySulfSat` with the function `Lee_Wieser_sulfide_melting`. This function can be used to model the concentration of any element during near fractional batch melting, and allows the contrasting behavior of chalcophile and lithophile elements to be modeled (e.g. Ba vs. Cu [Wieser et al. 2020]). The user must supply a dataframe with partition coefficients for silicate and sulfide phases, and the mass proportion of each phase. In Figure 7A–B, we calculate the concentration of Cu and Ba in aggregated melts for different melt extents. First, we specify the silicate modes:

```
Modes=pd.DataFrame(data={'ol': 0.6, 'opx': 0.2,
       'cpx': 0.18, 'sp': 0.02, 'gt': 0}, index=[0])
```

and the partition coefficients:

```
KDs_Cu=pd.DataFrame(data={'element': 'Cu',
        'ol': 0.048, 'opx': 0.034,
        'cpx': 0.043, 'sp': 0.223,
        'gt': 0, 'sulf': 800}, index=[0])

KDs_Ba=pd.DataFrame(data={'element': 'Ba',
        'ol': 0.000005, 'opx': 0.000006,
        'cpx': 0.0004, 'sp': 0.223,
        'gt': 0.00007, 'sulf': 0 }, index=[0])
```

For simplicity in this example, we assume that the silicate modes stay fixed throughout the melting interval. This assumption makes very little difference for Cu, as the partition coefficient is substantially higher for sulfides than any silicate phases. Even for Ba, this is a reasonable first-order assumption because it is extremely incompatible in all high abundance silicate phases. The other required inputs are:

1. The number of iterative steps (`N=3000`)
2. The S content of the mantle source in ppm (`S_Mantle=200`)
3. The concentration of S in mantle sulfides in ppm (`S_Sulf=360000`)
4. The initial concentration of the element of interest in the mantle prior to melting in ppm (`elem_Per=30`)
5. The S^{2-} concentration of the melt in ppm (`S_Melt_SCSS_2=1000`).
6. The proportion of S^{6+} (here `Prop_S6=0`), which will be used alongside the S^{2-} concentration to calculate the total amount of S in the melt using Equation 6:

These inputs are then used as follows for Cu:

```
df_Cu_200S=ss.Lee_Wieser_sulfide_melting(N=3000,
Modes=Modes, KDs=KDs_Cu, S_Mantle=200,
S_Sulf=360000, S_Melt_SCSS_2_ppm=1000,
elem_Per=30, Prop_S6=0)
```

and Ba:

```
df_Ba_200S=ss.Lee_Wieser_sulfide_melting(N=3000,
Modes=Modes, KDs=KDs_Ba, S_Mantle=200,
S_Sulf=360000, S_Melt_SCSS_2_ppm=1000,
elem_Per=6.85, Prop_S6=0)
```

These calculations were run at `S_Mantle` contents of 100 ppm, 200 ppm, and 300 ppm to produce Figure 7A–B).

In addition to the ease of the above calculations vs. existing tools, the other substantial advantage of `PySulfSat` is that it allows integration of melting models with models for partition coefficients in sulfides, and models of the SCSS within a single calculation environment. This enables a more sophisticated modeling approach than existing studies, which assumed a fixed SCSS throughout the melting interval [e.g. Lee et al. 2012; Wieser et al. 2020; Muth and Wallace 2022]. In reality, the major element composition of instantaneous melts will change as melting proceeds, particularly for incompatible elements such as Na_2O and K_2O. Consequently, the SCSS will change during melting, rather than being set at a fixed value. `PySulfSat` can be used to calculate the SCSS for instantaneous melt compositions from melting models. For example, the cyan line in Figure 7C–E shows calculations using instantaneous melt compositions estimated from a `Thermocalc` melting model [Jennings and Holland 2015]. This model using a calculated SCSS has a higher S content in the initial melts than the model assuming S = 1000 ppm throughout, resulting in a lower sulfide mode, a lower bulk K_D, and thus a higher Cu concentration in mantle melts at low F values (cyan vs. dashed magenta line, Figure 7). Sulfide is also exhausted at a lower F (black star, Figure 7C). Changing silicate melt modes can also be used instead of a fixed modal abundance, which will create more realistic trajectories for elements with an affinity for both sulfide and silicate phases.

While both the cyan and magenta models on Figure 7 assume K_D for sulfide-melt is fixed at 800, `PySulfSat` can also be used to calculate K_D as a function of temperature, liquid FeO content, and the Ni and Cu content of the sulfide using the model of Kiseeva and Wood [2015]. This more rigorous K_D approach results in a substantially lower K_D, and thus higher Cu contents in the melt. Additional information on how to perform these more advanced melting calculations can be found at ReadTheDocs. Overall, `PySulfSat` gives substantially more flexibility to explore concentrations in instantaneous and aggregated melts for all elements during melting in the presence of sulfide phases.

8 Other useful functions

We also include a number of functions for other common workflows associated with S. For example, the functions `convert_d34_to_3432S` and `convert_3432S_to_d34` can be used to convert between $\delta^{34}S$ values and $^{34}S/^{32}S$ ratios. By default, these functions use the the Vienna-CDT value of 1/22.6436 from Ding et al. [2001], although this can be overwritten with any value of interest (using the input `st_ratio`). For example, if a dataframe is loaded in with a column for `d34S` the isotope ratio can be calculated as follows:

```
S3432=ss.convert_d34_to_3432S(d34S=df['d34S'])
```

We also include a function which allows users to enter the amount of S present in the melt as S in wt.%, ppm, or as SO_2, SO_3, or SO_4. It then converts this concentration into an equivalent concentration expressed as different species (useful when converting EPMA data measured as SO_2 into S in ppm for example):

```
df=ss.convert_S_types(S_ppm=df['S_ppm'])
```

	S_wt	S_ppm	SO2_wt	SO2_ppm	SO3_wt	SO3_ppm	SO4_wt	SO4_ppm
0	0.100	1000.0	0.199791	1997.910494	0.249687	2496.865741	0.299582	2995.820989
1	0.110	1100.0	0.219770	2197.701544	0.274655	2746.552316	0.329540	3295.403087
2	0.090	900.0	0.179812	1798.119445	0.224718	2247.179167	0.269624	2696.238890
3	0.050	500.0	0.099896	998.955247	0.124843	1248.432871	0.149791	1497.910494
4	0.040	400.0	0.079916	799.164198	0.099875	998.746297	0.119833	1198.328395
5	0.035	350.0	0.069927	699.268673	0.087390	873.903010	0.104854	1048.537346
6	0.020	200.0	0.039958	399.582099	0.049937	499.373148	0.059916	599.164198
7	0.100	1000.0	0.199791	1997.910494	0.249687	2496.865741	0.299582	2995.820989

Additionally, the studies of Kiseeva and Wood [2015] and Brenan [2015] parameterize K_Ds as a function of melt composition, and sulfide composition for Kiseeva and Wood [2015]. The function `calculate_sulfide_kds` can be used to calculate these partition coefficients.

9 Future work and citation

The open-source nature of `PySulfSat`, along with recent increase in interest in the behavior of S in magmas, means that

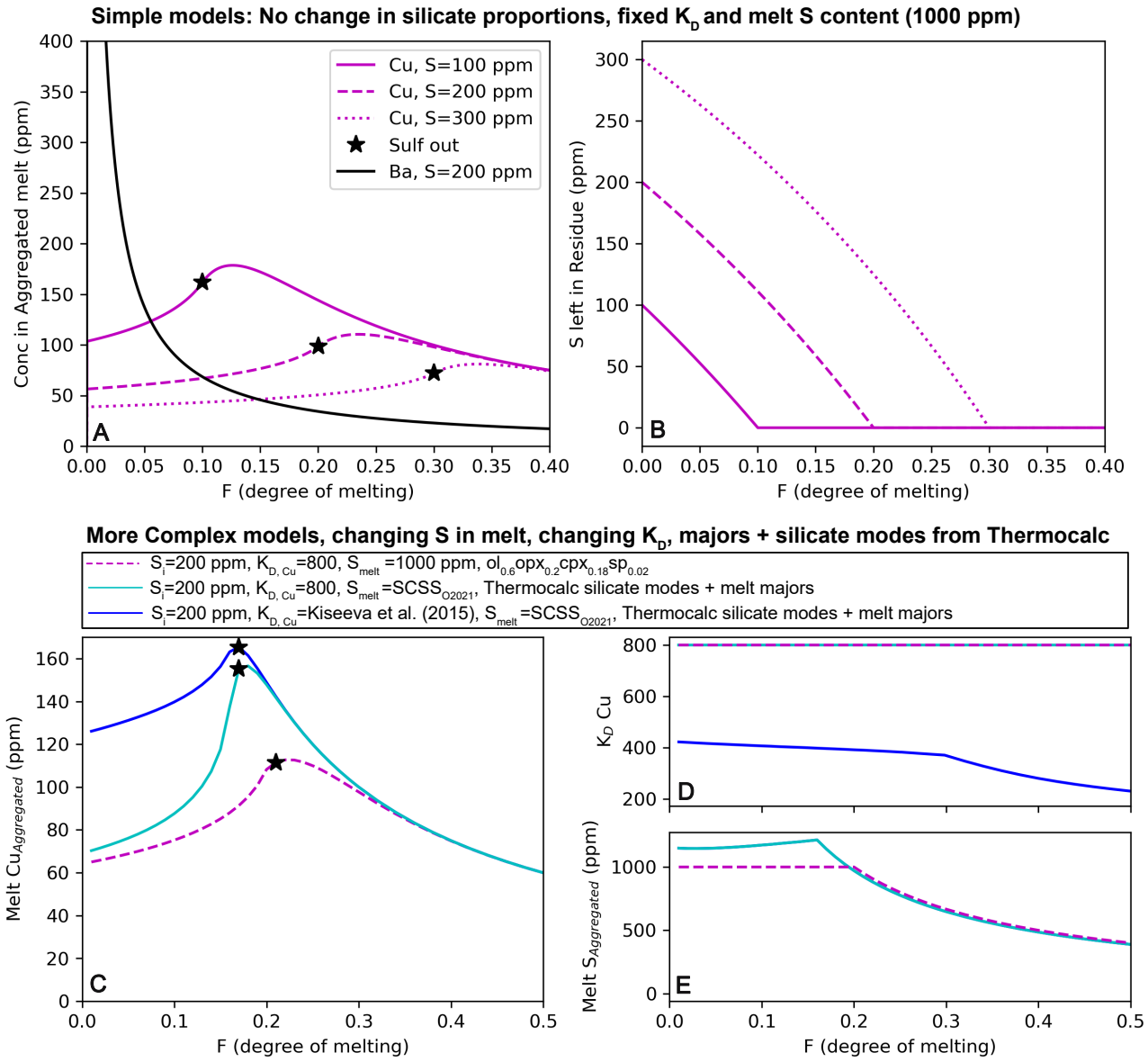

Figure 7: Modeling chalcophile elements during mantle melting. [A]–[B] Simple models following Lee et al. [2012] and Wieser et al. [2020] where the K_D in the sulfide, the modal proportion of silicate minerals and S in the melt is kept constant throughout the melting interval. Variation in elemental concentrations correlate with the initial S content of the mantle source. [C]–[E] More complex models combining melting models with K_D and SCSS functions within `PySulfSat`. For 200 ppm S in the mantle source, substantially different trajectories can be generated by varying the model for the amount of S in the melt, or the partition coefficient of Cu. The cyan and blue lines use a mantle melting model from Thermocalc to obtain the major element contents and temperature of instantaneous melts [Jennings and Holland 2015]. This allows the S content of these melts to be determined using the SCSS model of O'Neill [2021], assuming mantle sulfides contain 20 wt.% Ni and 5 wt.% Cu (after Ding and Dasgupta [2018]). The cyan line uses a fixed K_D for Cu (800, after Lee et al. [2012]). The blue line uses K_D calculated from the instantaneous silicate melt composition and an estimated mantle sulfide composition from Kiseeva and Wood [2015]. All models assume there is 30 ppm Cu in the mantle source.

this tool will continuously evolve. The current author team will endeavor to add new models as they are released, and anyone can submit new code using a pull request on GitHub (or by contacting the authors). Thus, users should check the ReadTheDocs page, where examples demonstrating new functionality beyond that described in this manuscript will be added in the future. New versions of `PySulfSat` can be obtained by running the following code in a `Jupyter` environment:

```
!pip install PySulfSat --upgrade
```

When citing calculations performed in `PySulfSat` in papers, users should be sure to specify which version they used, which can be obtained using:

```
ss.__version__
```

For example, the text may read "SCSS calculations were performed using the model of Smythe et al. [2017] implemented in PySulfSat v.1.0.3 (Wieser and Gleeson, 2023)." It is important to cite all the original papers used to perform calculations (e.g. the SCSS model, the model for S^{6+}), as well as citing PySulfSat.

At present, there is no open-source code that can model sulfide and sulfate saturation with all the most recent models, and the behavior of S during degassing from a a silicate melt. We hope that in future, the PySulfSat source code can be integrated with the wide variety of S degassing tools becoming available [e.g. Ding et al. 2023] to produce a single, coherent model engine for modeling S behavior in silicate melts.

10 Reporting bugs and requesting features

No software is free of bugs, particularly when new features are being constantly added. We have extensively benchmarked PySulfSat to existing spreadsheets, and before the package is published on PyPI, automatic unit tests are run through GitHub in the attempt to catch problems introduced by changing Python dependencies/updates. However, if users spot any bugs, or wish to request features, they should submit an 'issue' on the GitHub page. Alternatively, they can email the author team.

11 Conclusions

PySulfSat is a open-source Python3 tool motivated by the FAIR (Findable, Accessible, Interoperable, and Reusable) research framework [Bloemers and Montesanti 2020]. It will greatly speed up calculations, allow more intercomparison between models, and through its ease of implementation with Python, allow more detailed and robust investigations of the behavior of sulfur in magmatic systems (with a rigorous consideration of errors).

Author Contributions

PW conceived the project, wrote the S-based code and the manuscript. MG built the fractional crystallization MELTS functions allowing integrating of pyMELTScalc with PySulfSat, and contributed to manuscript editing and code testing.

Acknowledgements

We are grateful for help from Callum Reekie, who produced the O'Neill (2022) spreadsheet, as well as Nick Barber for motivation for this project. Thanks to Lee Saper and Ery Hughes who suggested implementing the method of O'Neill and Mavrogenes [2022] for calculating S^{6+}/S_T. We thank Lee Saper and Michelle Muth for very helpful reviews. We are extremely grateful for Hugh O'Neill for helping us understand the differences between CS6 in his paper and that of Boulliung and Wood, as well as various aspects of his S^{6+} models. Thanks to Kang Liu, Proteek Chowdhury, Julien Boulliung, Bernie Wood, and Ingrid Blanchard, for providing calibration data and/or spreadsheets/data to benchmark their models. We are enormously grateful to Paula Antoshechkina for her work building Matlab and Python tools for MELTS calculations. PW was supported by UC Berkeley start up funds.

Data Availability

All files are available on GitHub (https://github.com/PennyWieser/PySulfSat). YouTube videos explaining various aspects of the tool are available on the PySulfSat YouTube channel bit.ly/PySulfSatYouTube, and Jupyter Notebook examples are available on the ReadTheDocs page (bit.ly/PySulfSatRTD). The PyMELTScalc code is available on GitHub (https://github.com/gleesonm1/pyMELTScalc), archived on Zenodo (https://doi.org/10.5281/zenodo.7758493), and will be described in a follow up publication.

Copyright notice

References

Anenburg, M. and H. S. C. O'Neill (2019). "Redox in Magmas: Comment on a Recent Treatment of the Kaiserstuhl Volcanics (Braunger et al., Journal of Petrology, 59, 1731–1762, 2018) and Some Other Misconceptions". *Journal of Petrology* 60(9), pages 1825–1832. DOI: 10.1093/petrology/egz046.

Antoshechkina, P. M. and M. S. Ghiorso (2018). "MELTS for MATLAB: A new Educational and Research Tool for Computational Thermodynamics". *AGU Fall Meeting Abstracts*. Volume 2018, ED44B-23, ED44B–23.

Asimow, P. D. and M. S. Ghiorso (1998). "Algorithmic modifications extending MELTS to calculate subsolidus phase relations". *American Mineralogist* 83(9-10), pages 1127–1132. DOI: 10.2138/am-1998-9-1022.

Baker, D. R. and R. Moretti (2011). "Modeling the solubility of sulfur in magmas: a 50-year old geochemical challenge". *Reviews in Mineralogy and Geochemistry* 73(1), pages 167–213. DOI: 10.2138/rmg.2011.73.7.

Berry, A. J., G. A. Stewart, H. S. C. O'Neill, G. Mallmann, and J. F. W. Mosselmans (2018). "A re-assessment of the oxidation state of iron in MORB glasses". *Earth and Planetary Science Letters* 483, pages 114–123. DOI: 10.1016/j.epsl.2017.11.032.

Blanchard, I., S. Abeykoon, D. J. Frost, and D. C. Rubie (2021). "Sulfur content at sulfide saturation of peridotitic melt at upper mantle conditions". *American Mineralogist: Journal of Earth and Planetary Materials* 106(11), pages 1835–1843. DOI: 10.2138/am-2021-7649.

Bloemers, M. and A. Montesanti (2020). "The FAIR Funding Model: Providing a Framework for Research Funders to Drive the Transition toward FAIR Data Manage-

ment and Stewardship Practices". *Data Intelligence* 2(1-2), pages 171–180. DOI: 10.1162/dint_a_00039.

Borisov, A. A. and A. I. Shapkin (1990). "A new empirical equation rating Fe3+/Fe2+ in magmas to their composition, oxygen fugacity, and temperature". *Geochemistry International* 27(1), pages 111–116.

Boulliung, J. and B. J. Wood (2022). "SO2 solubility and degassing behavior in silicate melts". *Geochimica et Cosmochimica Acta* 336, pages 150–164. DOI: 10.1016/j.gca.2022.08.032.

Brenan, J. M. (2015). "Se–Te fractionation by sulfide–silicate melt partitioning: implications for the composition of mantle-derived magmas and their melting residues". *Earth and Planetary Science Letters* 422, pages 45–57. DOI: 10.1016/j.epsl.2015.04.011.

Brounce, M., E. Stolper, and J. Eiler (2017). "Redox variations in Mauna Kea lavas, the oxygen fugacity of the Hawaiian plume, and the role of volcanic gases in Earth's oxygenation". *Proceedings of the National Academy of Sciences* 114(34), pages 8997–9002. DOI: 10.1073/pnas.1619527114.

Chowdhury, P. and R. Dasgupta (2019). "Effect of sulfate on the basaltic liquidus and sulfur Concentration at Anhydrite Saturation (SCAS) of hydrous basalts–Implications for sulfur cycle in subduction zones". *Chemical Geology* 522, pages 162–174. DOI: 10.1016/j.chemgeo.2019.05.020.

Danyushevsky, L. V. and P. Plechov (2011). "Petrolog3: Integrated software for modeling crystallization processes". *Geochemistry, Geophysics, Geosystems* 12(7). DOI: 10.1029/2011GC003516.

Ding, S. and R. Dasgupta (2018). "Sulfur inventory of ocean island basalt source regions constrained by modeling the fate of sulfide during decompression melting of a heterogeneous mantle". *Journal of Petrology* 59(7), pages 1281–1308. DOI: 10.1093/petrology/egy061.

Ding, S., T. Plank, P. J. Wallace, and D. J. Rasmussen (2023). "Sulfur_X: A Model of Sulfur Degassing During Magma Ascent". *Geochemistry, Geophysics, Geosystems* 24(4). DOI: 10.1029/2022gc010552.

Ding, T., S. l. Valkiers, H. Kipphardt, P. De Bievre, P. D. P. Taylor, R. Gonfiantini, and R. Krouse (2001). "Calibrated sulfur isotope abundance ratios of three IAEA sulfur isotope reference materials and V-CDT with a reassessment of the atomic weight of sulfur". *Geochimica et Cosmochimica Acta* 65(15), pages 2433–2437. DOI: 10.1016/s0016-7037(01)00611-1.

Edmonds, M., T. A. Mather, and E. J. Liu (2018). "A distinct metal fingerprint in arc volcanic emissions". *Nature Geoscience* 11(10), pages 790–794. DOI: 10.1038/s41561-018-0214-5.

Fincham, C. and F. D. Richardson (1954). "The behaviour of sulphur in silicate and aluminate melts". *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 223(1152), pages 40–62. DOI: 10.1098/rspa.1954.0099.

Fortin, M.-A., J. Riddle, Y. Desjardins-Langlais, and D. R. Baker (2015). "The effect of water on the sulfur concentration at sulfide saturation (SCSS) in natural melts". *Geochimica et Cosmochimica Acta* 160, pages 100–116. DOI: 10.1016/j.gca.2015.03.022.

Frost, B. (1991). "Chapter 1. Introduction to oxygen fugacity and its petrological importance". *Oxide minerals* (1), pages 1–10.

Ghiorso, M. S. and R. O. Sack (1995). "Chemical mass transfer in magmatic processes IV. A revised and internally consistent thermodynamic model for the interpolation and extrapolation of liquid-solid equilibria in magmatic systems at elevated temperatures and pressures". *Contributions to Mineralogy and Petrology* 119(2), pages 197–212. DOI: 10.1007/BF00307281.

Gleeson, M., A. Paula, and P. Wieser (2023). "PyMELTSCalc". *Zenodo*. DOI: 10.5281/zenodo.7758494. [Github Code repository /gleesonm1/pyMELTScalc].

Gualda, G. A. R. and M. S. Ghiorso (2015). "MELTS_Excel: AMicrosoft Excel-based MELTS interface for research and teaching of magma properties and evolution". *Geochemistry, Geophysics, Geosystems* 16(1), pages 315–324. DOI: 10.1002/2014gc005545.

Gualda, G. A. R., M. S. Ghiorso, R. V. Lemons, and T. L. Carley (2012). "Rhyolite-MELTS: a modified calibration of MELTS optimized for silica-rich, fluid-bearing magmatic systems". *Journal of Petrology* 53(5), pages 875–890. DOI: 10.1093/petrology/egr080.

Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. (2020). "Array programming with NumPy". *Nature* 585(7825), pages 357–362. DOI: 10.1038/s41586-020-2649-2.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". *Computing in Science & Engineering* 9(3), pages 90–95. DOI: 10.1109/MCSE.2007.55.

Iacono-Marziano, G., M. Le Vaillant, B. M. Godel, S. J. Barnes, and L. Arbaret (2022). "The critical role of magma degassing in sulphide melt mobility and metal enrichment". *Nature Communications* 13(1), page 2359. DOI: 10.1038/s41467-022-30107-y.

Jennings, E. S. and T. J. B. Holland (2015). "A simple thermodynamic model for melting of peridotite in the system NCFMASOCr". *Journal of Petrology* 56(5), pages 869–892. DOI: 10.1093/petrology/egv020.

Johnson, C. M., M. S. Ghiorso, M. Spiegelman, A. S. Wolf, J. Adams, and R. MyHill (2022). "ThermoEngine: Thermodynamic properties estimator and phase equilibrium calculator". *Astrophysics Source Code Library*, ascl–2208.

Jugo, P. J. (2009). "Sulfur content at sulfide saturation in oxidized magmas". *Geology* 37(5), pages 415–418. DOI: 10.1130/g25527a.1.

Jugo, P. J., M. Wilke, and R. E. Botcharnikov (2010). "Sulfur K-edge XANES analysis of natural and synthetic basaltic glasses: Implications for S speciation and S content as function of oxygen fugacity". *Geochimica et Cosmochimica Acta* 74(20), pages 5926–5938. DOI: 10.1016/j.gca.2010.07.022.

Kilinc, A., I. S. E. Carmichael, M. L. Rivers, and R. O. Sack (1983). "The ferric-ferrous ratio of natural silicate liquids equilibrated in air". *Contributions to Mineralogy*

and Petrology 83(1-2), pages 136–140. DOI: 10.1007/bf00373086.

Kiseeva, E. S. and B. J. Wood (2015). "The effects of composition and temperature on chalcophile and lithophile element partitioning into magmatic sulphides". *Earth and Planetary Science Letters* 424, pages 280–294. DOI: 10.1016/j.epsl.2015.05.012.

Kleinsasser, J. M., A. C. Simon, B. A. Konecke, M. J. Kleinsasser, P. Beckmann, and F. Holtz (2022). "Sulfide and sulfate saturation of dacitic melts as a function of oxygen fugacity". *Geochimica et Cosmochimica Acta* 326, pages 1–16. DOI: 10.1016/j.gca.2022.03.032.

Kress, V. C. and I. S. E. Carmichael (1988). "Stoichiometry of the iron oxidation reaction in silicate melts". *American Mineralogist* 73(11-12), pages 1267–1274.

Lee, C.-T. A., P. Luffi, E. J. Chin, R. Bouchet, R. Dasgupta, D. M. Morton, V. Le Roux, Q.-z. Yin, and D. Jin (2012). "Copper systematics in arc magmas and implications for crust-mantle differentiation". *Science* 336(6077), pages 64–68. DOI: 10.1126/science.1217313.

Lerner, A. H., M. J. Muth, P. J. Wallace, A. Lanzirotti, M. Newville, G. A. Gaetani, P. Chowdhury, and R. Dasgupta (2021). "Improving the reliability of Fe-and S-XANES measurements in silicate glasses: Correcting beam damage and identifying Fe-oxide nanolites in hydrous and anhydrous melt inclusions". *Chemical Geology* 586, page 120610. DOI: 10.1016/j.chemgeo.2021.120610.

Li, C. and E. M. Ripley (2009). "Sulfur contents at sulfide-liquid or anhydrite saturation in silicate melts: empirical equations and example applications". *Economic Geology* 104(3), pages 405–412. DOI: 10.2113/gsecongeo.104.3.405.

Li, H. and L. Zhang (2022). "A thermodynamic model for sulfur content at sulfide saturation (SCSS) in hydrous silicate melts: With implications for arc magma genesis and sulfur recycling". *Geochimica et Cosmochimica Acta* 325, pages 187–204. DOI: 10.1016/j.gca.2022.03.008.

Liu, K., L. Zhang, X. Guo, and H. Ni (2021). "Effects of sulfide composition and melt H2O on sulfur content at sulfide saturation in basaltic melts". *Chemical Geology* 559, page 119913. DOI: 10.1016/j.chemgeo.2020.119913.

Mason, E., P. E. Wieser, E. J. Liu, M. Edmonds, E. Ilyinskaya, R. C. Whitty, T. A. Mather, T. Elias, P. A. Nadeau, T. C. Wilkes, et al. (2021). "Volatile metal emissions from volcanic degassing and lava–seawater interactions at Kīlauea Volcano, Hawai'i". *Communications Earth & Environment* 2(1), pages 1–16. DOI: 10.1038/s43247-021-00145-3.

Masotta, M. and H. Keppler (2015). "Anhydrite solubility in differentiated arc magmas". *Geochimica et Cosmochimica Acta* 158, pages 79–102. DOI: 10.1016/j.gca.2015.02.033.

Muth, M. J. and P. J. Wallace (2021). "Slab-derived sulfate generates oxidized basaltic magmas in the southern Cascade arc (California, USA)". *Geology* 49(10), pages 1177–1181. DOI: 10.1130/g48759.1.

– (2022). "Sulfur recycling in subduction zones and the oxygen fugacity of mafic arc magmas". *Earth and Planetary Science Letters* 599, page 117836. DOI: 10.1016/j.epsl.2022.117836.

Myers, J. t. and H. P. Eugster (1983). "The system Fe-Si-O: Oxygen buffer calibrations to 1,500 K". *Contributions to Mineralogy and Petrology* 82, pages 75–90. DOI: 10.1007/bf00371177.

Nash, W. M., D. J. Smythe, and B. J. Wood (2019). "Compositional and temperature effects on sulfur speciation and solubility in silicate melts". *Earth and Planetary Science Letters* 507, pages 187–198. DOI: 10.1016/j.epsl.2018.12.006.

O'Neill, H. S. C. (1987). "Quartz-fayalite-iron and quartz-fayalite-magnetite equilibria and the free energy of formation of fayalite (Fe 2 SiO 4) and magnetite (Fe 3 O 4)". *American Mineralogist* 72(1–2), pages 67–75.

– (2021). "The thermodynamic controls on sulfide saturation in silicate melts with application to ocean floor basalts". *Magma Redox Geochemistry*, pages 177–213. DOI: 10.1002/9781119473206.ch10.

O'Neill, H. S. C., A. J. Berry, and G. Mallmann (2018). "The oxidation state of iron in Mid-Ocean Ridge Basaltic (MORB) glasses: Implications for their petrogenesis and oxygen fugacities". *Earth and Planetary Science Letters* 504, pages 152–162. DOI: 10.1016/j.epsl.2018.10.002.

O'Neill, H. S. C. and J. A. Mavrogenes (2022). "The sulfate capacities of silicate melts". *Geochimica et Cosmochimica Acta* 334, pages 368–382. DOI: 10.1016/j.gca.2022.06.020.

Putirka, K. (2016). "Rates and styles of planetary cooling on Earth, Moon, Mars, and Vesta, using new models for oxygen fugacity, ferric-ferrous ratios, olivine-liquid Fe-Mg exchange, and mantle potential temperature". *American Mineralogist* 101(4), pages 819–840. DOI: 10.2138/am-2016-5402.

Reekie, C. D. J., F. E. Jenner, D. J. Smythe, E. H. Hauri, E. S. Bullock, and H. M. Williams (2019). "Sulfide resorption during crustal ascent and degassing of oceanic plateau basalts". *Nature communications* 10(1), pages 1–11. DOI: 10.1038/s41467-018-08001-3.

Sack, R. O., I. S. E. Carmichael, M. Rivers, and M. S. Ghiorso (1981). "Ferric-ferrous equilibria in natural silicate liquids at 1 bar". *Contributions to Mineralogy and petrology* 75, pages 369–376.

Smythe, D. J., B. J. Wood, and E. S. Kiseeva (2017). "The S content of silicate melts at sulfide saturation: new experiments and a model incorporating the effects of sulfide composition". *American Mineralogist* 102(4), pages 795–803. DOI: 10.2138/am-2017-5800ccby.

The pandas development team (2020). "pandas-dev/pandas: Pandas". Version v2.0.1. *Zenodo*. DOI: 10.5281/zenodo.7857418. [Github Code repository /pandas-dev/pandas/tree/v2.0.1].

Virtanen, P., R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors (2020). "SciPy 1.0: Fundamental Algorithms for Scientific

Computing in Python". *Nature Methods* 17, pages 261–272. DOI: 10.1038/s41592-019-0686-2.

Virtanen, V. J., J. S. Heinonen, N. D. Barber, and F. Molnár (2022). "Complex Effects of Assimilation on Sulfide Saturation Revealed by Modeling with the Magma Chamber Simulator: A Case Study on the Duluth Complex, Minnesota, USA". *Economic Geology* 117(8), pages 1881–1899. DOI: 10.5382/econgeo.4917.

Wallace, P. J. and I. S. Carmichael (1994). "S speciation in submarine basaltic glasses as determined by measurements of S Kα X-ray wavelength shifts". *American Mineralogist* 79(1-2), pages 161–167.

Wieser, P. E. and F. Jenner (2021). "Chalcophile Elements: Systematics and Relevance". *Reference Module in Earth Systems and Environmental Sciences*, pages 67–80. DOI: 10.1016/b978-0-08-102908-4.00092-8.

Wieser, P. E., F. Jenner, M. Edmonds, J. Maclennan, and B. E. Kunz (2020). "Chalcophile elements track the fate of sulfur at Kilauea Volcano, Hawai'i". *Geochimica et Cosmochimica Acta* 282, pages 245–275. DOI: 10.1016/j.gca.2020.05.018.

Wieser, P. E., M. Petrelli, J. Lubbers, E. Wieser, S. Ozaydin, A. Kent, and C. Till (2022). "Thermobar: an open-source Python3 tool for thermobarometry and hygrometry". *Volcanica* 5(2), pages 349–384. DOI: 10.30909/vol.05.02.349384.

Zajacz, Z. and A. Tsay (2019). "An accurate model to predict sulfur concentration at anhydrite saturation in silicate melts". *Geochimica et Cosmochimica Acta* 261, pages 288–304. DOI: 10.1016/j.gca.2019.07.007.