VESIcal manuscript text

**Key Points**

1. The first comprehensive volatile solubility tool capable of processing large datasets automatically
2. Eight built-in solubility models, with automatic calculation and plotting functionality
3. Build in python but easily usable by scientists with any level of coding skill

**Abstract**

Thermodynamic modeling has been the backbone of interpreting geologic data and modelling geologic systems for decades. However, more recent advancements in computational capabilities and a marked increase in researchers' accessibility to computing tools has outpaced the functionality and extensibility of currently available modeling tools. Here we present the first comprehensive modelling tool for H22O, CO22, and mixed (H22O-CO22) solubility in silicate melts that: a) allows users access to all commonly used models, inlcuding easy intercomparison between models; b) provides universal functionality for all models (e.g., functions for calculting saturation pressures, degassing paths, etc.); c) can process large datasets (1,000's of samples) automatically; d) outputs computed data into an excel spreadsheet for simple post-modelling analysis; e) integrates advanced plotting capabilities directly within the tool; and f) provides all of these within the framework of a python library, making the tool extensible by the user and allowing any of the model functions to be incorporated into any other code capable of calling python. The tool is presented within this manuscript, which is a Jupyter notebook containing worked examples accessible to python users with a range of skill levels. The VESIcal python library is open-source and available for download at <https://github.com/kaylai/VESIcal>.

# Plain Language Summary

To be written after first full draft is complete.

# 1. Introduction

The most abundant volatile components found in terrestrial magmatic systems are H22O and CO22. It has been known for nearly a century (Bowen, 1924; Tuttle & Bowen, 1952) that these components have profound effects on the chemical and material properties of magmas (e.g. phase equilibria, melting temperatures, magma viscosity and density), and thus significantly affect their geochemical and dynamical behavior (e.g. eruption and degassing style, role of magmatism in terrestrial C/H budgets, etc). Therefore, in order to accurately understand the behavior of magmas in both plutonic and volcanic environments, it is critical to assess how H22O and CO22 behave in magmatic systems as a function of intensive variables such as pressure, temperature, and melt/fluid composition.

One specific and critically important behavior of volatiles in magmas is their solubility. The volatile solubility of a magma varies significantly as it rises from depth to the surface, as it is particularly dependent on the pressure, melt composition, and more weakly on temperature. Volatile solubility of a magma is defined at a given pressure and temperature as the volatile content at which a magma becomes volatile saturated; i.e., the conditions where the melt can no longer hold the volatile in solution, and there is a resulting change in the number of magmatic phases present (e.g. from melt + crystals to melt + crystals + fluid). This change from volatile under-saturated to volatile-saturated is significant in terms of both the chemical and physical evolution of a rising magma, as 1.) the formation of a volatile fluid phase comes with a large change in total volume due to bubble formation and the large molar volume of H22O-CO22 fluids, 2.) the bubbles themselves may increase the bulk magma viscosity (refs?) and become nucleation sites for crystals (refs?), 3.) the volatile content of the silicate melt portion of the magma is no longer constant and begins to decrease with pressure, thereby increasing the viscosity of the melt (ref’s?), and thus bulk magma viscosity, and 4.) with decreasing volatile content in the melt, mineral phase boundaries may be crossed, resulting in the stabilization (crystallization) or destabilization (reaction melting) of phenocryst phases (ref’s?).

## 1.1 Experimental studies determining volatile solubility in magmas

Given the critical importance of volatile solubility to our understanding of magmatic systems discussed above, it is no surprise that there is a great deal of past work to quantify it, starting with the classic treatise of Tuttle & Bowen (1952?). This study led to the wider recognition of the importance of volatiles, and initiated a significant effort in the development of both the experimental and analytical approaches to determining volatile solubilities as a function of pressure, temperature, and melt composition. Hamilton et al (1964) was one of the first studies that compared H22O solubilities for differing melt compositions (basalt and andesite), while also investigating the effect of dissolved H22O and oxidation state on the magmatic phase equilibria. This study was followed by the groundbreaking experimental measurements of the Burnham group on the dissolution of H22O in albite melts (Burnham et al, 197?; 197??; Holloway et al, 19??). Many subsequent studies have continued since, aided significantly by developments in analytical techniques to determine dissolved volatile contents, such as infra-red spectroscopy (FTIR; e.g., Stolper, 1980; Stolper et al, 198??, Fine & Stolper, 1980; etc), and secondary ion mass spectrometry (SIMS; Hervig, Hauri, etc, need more ref’s here). The fact that both the FTIR and SIMS techniques allow micro-determinations of volatile contents on sample sizes at the ~10 micron scale, their development also facilitated a new approach to determining pre-eruptive magmatic volatile contents by allowing precise measurement of volatiles in trapped glass/melt inclusions (Wallace, Roggensack, Plank, etc). This explosion of new information from melt inclusion measurements about pre-eruptive volatile contents made understanding volatile solubility in magmas as a function of pressure and temperature even more critical for the interpretation of the data, and highlighted the need for flexible and precise solubility models that allowed estimation of solubility conditions for a broad range of pressure, temperature, and melt/fluid compositions

1.2 Model availability prior to VESIcal

Until now, calculating saturation pressures, dissolved volatile contents, and isobars and isopleths has been a time-consuming endeavour. In many cases, the authors of solubility models provided no accompanying tool to perform these calculations, requiring users to correctly combine and interpret the relevant equations (e.g., Dixon et al., 1995; Dixon, 1997; Liu et al., 2005; Shishkina et al., 2014). This is problematic, given that typesetting can sometimes result in errors such as typos within model equations (See Table ? for corrected versions). For some models, an excel spreadsheet is provided, or available at request from the authors. For example, Newman and Lowenstern (2002) included a simplified version of the Dixon (1997) model as part of “VolatileCalc”, which is written in Visual Basic for Excel. Due to its simplicity, allowing users to calculate saturation pressures, degassing paths, isobars and isopleths with a few button clicks and pop-up boxes, this tool has proved extremely popular (with 741 citations at the time of writing). However, to calculate saturation pressures using VolatileCalc, the user must individually enter the SiO2, H2O, CO2 content and temperature of every single sample into pop-up boxes. Similarly, the excel spreadsheet for the Moore et al. (1998) model calculates dissolved H2O contents based on the concentration of 9 oxides, temperature, and the fraction of XH2O in the vapour, which must be pasted in for every sample. Finally, Allison et al. (2019) provide an excel spreadsheet that allows users to calculate fugacities, partial pressures, isobars, isopleths and saturation pressures. Again, parameters for each sample must be entered individually, with no way to calculate large numbers of samples automatically.

More recently, authors have provided web-hosted interfaces to calculating saturation pressures and dissolved volatile contents (e.g., Iacono-Marziano et al., 2012; <http://calcul-isto.cnrs-orleans.fr/>, and Ghiorso and Gualda, 2015; <http://melts.ofm-research.org/CORBA_CTserver/GG-H2O-CO2.html>). Ghiorso and Gualda (2015) also provide a Mac Application~~. Interestingly, the web-hosted version of Magmasat returns the error message “no conv” when calculating saturation pressures for many samples which do return an output using the Mac Application.~~ The main limitation of these tools is that they require manual entry of the concentrations of 8-10 major oxides, temperature, as well as CO2 and H2O concentrations to calculate saturation pressures, or XH2O to calculate dissolved volatile contents. Unlike the Moore et al. (1998) spreadsheet, the oxide concentrations for a single sample cannot be pasted in en-mass, but must be individually typed or pasted into each box. This is particularly time consuming for melt inclusion studies, where saturation pressures are calculated for hundreds of inclusions, each with different entrapment temperatures, CO2, H2O, and major element concentrations. For example, the saturation pressures from 105 Gakkel ridge melt inclusions calculated in Magmasat by Bennet et al. (2019) required the manual entry of 1365 values! The potential for user error in this data entry stage should not be overlooked. Another issue with these web-hosted tools is their longevity; the link provided in the Iacono-Marziano et al. (2012) manuscript returns an error “this site cannot be reached”, although email contact with the author directed us towards the newer link given above. Similarly, the web tool for the popular Papale et al. (2006) H2O-CO2 solubility model no longer works (due to updates in MELTS?), and the link to H2O-CO2 equation of state web tool of Duan and Zhang (2006) provided in their manuscript returns a 404 error.

In contrast, VESIcal is an open source tool and as such is far less prone to the preservation issues discussed above. Because the VESIcal code is hosted on GitHub, every change to the code is tracked publicly (Perkel, 2016). VESIcal’s current release is also archived on Zenodo, which provides a static citable DOI (DOI here) for the current version of the code, along with a snapshot of the GitHub repository at the time of release.

# 2. Research Methodology

Here we present a thermodynamic volatile calculation model engine capable of performing a wide array of calculations using any one of eight of the most popular volatile solubility models:

* MagmaSat (MELTS v. 1.2.0; Ghiorso and Gualda, 2015; the default model)
* Dixon (1997)
* Moore et al. (1998; water only, but H22O fluid concentration can be specified)
* Liu et al (2005; H22O-CO22, rhyolitic compositions only)
* Iacono-Marziano et al. (2002)
* Shishkina et al. (2014)
* Eguchi and Dasgupta (2018; carbon only)
* Allison et al. (2019, carbon only).

The combination of these models produces a single tool with an extensive pressure-temperature-composition calibration range (Figure calibrange). VESIcal represents the first volatile solubility tool with the ability to perform calculations for multiple samples at once, with built-in functionality for extracting data from an Excel file. In addition, the code is written such that it is flexible (sample, calculation type, and model type can be chosen discretely) and extensible (VESIcal code can be easily imported for use in python scripts, and the code is formatted such that new volatile models can be easily added).

As discussed below, any individual model is only valid within its calibrated range, and each model is parameterized and expressed differently, making it impractical to simply combine them into one large model. Instead, VESIcal provides one set of tools that can access and utilize all of these models.

VESIcal has been designed for practicality and ease of use. It is designed to be used by anyone, from someone who is completely unfamiliar with coding to an adept programmer. The non-coder user can interact with VESIcal through this manuscript, and can upload a file with data, execute the tutorial below, and save the results to an Excel file to work with outside of VESIcal. The programmer user may wish to use the more advanced functionality provided by VESIcal, including the ability to hybridize models (e.g., use one model for H22O and another for CO22) or write their own routines and code that calls VESIcal methods. Specific instructions on how to interact with this code are provided in section X.

All of the calculations implemented in VESIcal can be performed using any of the models included. The code is structured by calculation rather than by model, which provides an intuitive way for users to interact with the code and compare outputs from multiple models. A python class is definied for each calculation. These classes can be called as python methods, take the model name and any applicable data as arguments, and return the results of the calculation. Each class performs five key functions: 1) creates the requested model object and performs any necessary pre-processing (e.g., ensuring relevant data are present; normalizing data); 2) takes user input and performs the mathematical calculation; 3) does any necessary processing of the output (e.g., normalizing totals); 4) checks that the model is being used within its calibrated range; and 5) returns calcualted outputs in an intuitive and manipulatable format (e.g., a python dictionary, a figure, or a pandas DataFrame). Results of calculations can easily be saved to one or more Excel files (.xslx).

## 2.1 Models included in VESIcal

Below we detail all eight volatile solubility models implemented in VESIcal, including information about their calibration ranges and notes on the sample types appropriate to each model (and, importantly, where caution should be taken when extrapolating these models). To ensure the proper implementation of these models into VESIcal, we have tested the predictions of VESIcal versus each model, using the calibration dataset for each (or a close approximation of that dataset when complete information was unavailable). The comparisons show the VESIcal reproduces all model predictions well (can we say how well?) and are included in the Supplement.

Navigating this array of models can be challenging. How can a user determine which model best suits their needs? MagmaSat (the default model in VESIcal) is the mostly widely calibrated (in P-T-X space), and so we recommend it for the majority of cases. Where a user wishes to use the other implemented models, we provide some tools to help choose the most appropriate model (see Supplement). These tools are described in more detail in Section X.X.

**Dixon, 1997**

The Dixon (1997) model calculates the solubility of H2O and CO2 solubility for basaltic silicate melts, combining thermodynamic expressions as a function of pressure and temperature from Dixon and Stolper (1995), and an empirical compositional parameter, (Dixon, 1997):

A screenshot of a cell phone

Description automatically generated

Based on the strong correlation between and SiO2 in a suite of lavas from the North Arch Volcanic Field, Dixon (1997) simplify this compositional parameter by expressing their  and A picture containing drawing, table

Description automatically generated terms as a function of melt SiO2 content, to aid their investigations of volatile solubility in this specific suite of lavas (Dixon et al., 1997).

{Do we want these equations? Also do we need full thermodynamic expression these terms go into?}

The popular excel-based workbook VolatileCalc (Newman and Lowernstern, 2002) uses these simplified expressions rather than the full -expression, requiring users to only specify melt temperature, SiO2, CO2 and H2O contents. This approach has been widely implemented in the literature (perhaps add refs?) and is implemented in VESIcal for consistency, allowing users to directly compare VESIcal results to those of VolatileCalc.

However, care must be taken with this approach. Firstly, the North Arch lavas span SiO2 contents of 40-49 wt%, and due to the rapid drop in with increasing SiO2, extrapolation beyond 51.2 wt% SiO2 yields a negative value for . To avoid this issue, VolatileCalc returns an error, and will not perform the calculation if users enter a SiO2 content>49 wt%, or <40 wt%. Some studies extrapolate beyond this by simply entering SiO2=49 wt% into VolatileCalc if their melts have slightly higher silica contents (e.g., Sides et al. 2014a, b; Tucker et al. 2019). If a user enters a composition with SiO2>49 wt% or <40 wt%, VESIcal performs the calculation for an SiO2 content of 49 wt% or 40 wt% respectively. Newman and Lowenstern (2002) suggest that this approach will be “generally applicable for other basaltic rocks with <52 wt% SiO2”. However, if a large proportion of a sample suite has SiO2 contents outside this range, this simplified parameterization will neglect variations in solubility that may exist within that suite. Additionally, if samples have SiO2 contents between 40-49 wt%, this simplification is only valid for melts which follow similar trends in -SiO2 space to the North Arch lavas. We encourage users to check this relationship using the Jupyter notebook provided in the supplement. If samples follow an offset trend, or a display a different gradient, we recommend users apply one of the more recent models, which include more complete compositional terms calibrated on a wider range of basaltic compositions (e.g., Iacono-Marziano et al., 2012; Shishkina et al., 2014; Ghiorso and Gualda, 2015; Allison et al., 2019).

**Moore et al., 1998**

The Moore et al. (1998) model calculates the solubility of H2O for a wide range of silicate melt compositions using an empirical expression valid between 700-1200˚C and 0-3000 bars. The model calibration dataset combines their experiments with literature data, and spans sub-alkaline basaltic to rhyolitic compositions, also incorporating some more alkaline compositions (ref TAS diagram). {Details about expression and coefficients?}

A close up of a clock

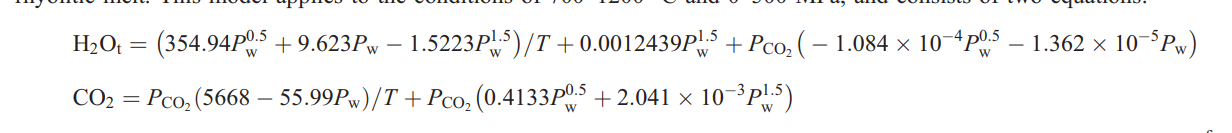
Description automatically generated

Where XH2O melt is…As their equation includes the term , this model can be integrated with CO2-only models (e.g., that of Allison et al., 2019) to investigate mixed CO2-H2O fluids. The empirical nature of the Moore et al. (1998) model means that extreme care should be taken when extrapolating it outside its calibration range in terms of melt composition, temperature, and pressure. In particular, the authors warn against extrapolating this model to higher pressures, due to the complexities of the critical behavoir of fluids at these compositions. {Do we want to include a sentence at the end of each saying how, untill now, the model has been available, e.g., a spreadsheet for Moore, a web-app for Magmasat, and maybe discuss some of the limitations of each of these for each model? – e.g., magmasat now ddoesnt work on PC}

**Liu et al., 2005**

The Liu et al. (2005) model calculates the solubility of H2O and CO2 in metaluminous, high-silica rhyolitic melts using an empirical expression, valid between 700-1200˚C, and 0-5000 bars. The model calibration dataset combines their experiments of H2O solubility between ~1 and 250 bars with literature data for the solubility of H2O and CO2 spanning a range of pressures, extending up to 5000 bars. Unlike the Moore et al. (1998) model, their empirical expressions do not have a term dependent upon the composition of the silicate melt, so extreme care should be taken when applying this model to melts with compositions different to those in the calibration dataset.

The equations implemented in VESIcal are:



where H2Ot is total dissolved H2O content in wt%, CO2 is in ppm by mass, T is temperature in Kelvin, Pw and PCO2 are the mole fraction of H2O or CO2 in the fluid times pressure in MPa (Pw=X$^f$P).

**Iacono-Marziano et al., 2012**

The Iacono-Marziano et al. (2012) model calculates the solubility of H2O and CO2 in mafic melts by combining thermodynamic expressions with empirical formulations accounting for melt composition. The calibration dataset for CO2 combines their experiments with those from a variety of literature studies for mixed H2O-CO2 fluids, spanning temperatures between 1100 and 1400 ˚C, and pressures between 100 and 10,000 bars. The expression for H2O incorporates pure H2O experiments from the literature (spanning 163-6067 bars, and 1000-1250 ˚C), as well as the experiments on H2O-CO2 fluids used to calibrate the CO2 expression. Melt compositions are predominantly mafic (SiO2 < X wt%), spanning subalkaline-alkaline basalts (total alkalis X-Y wt%) to basaltic andesites.

{Do we want equations? Quite long….}

A screenshot of a cell phone

Description automatically generated

The expression for CO2 solubility incorporates an compositional term parameterized in terms of melt Al2O3, FeO, MgO, Na2O, K2O, MgO and FeO contents. Due to the empirical nature of this term, care must be taken when applying this model to melts which have compositions lying outside the compositional range of their calibration dataset. In particular, the authors warn that the effects of melt MgO and FeO content on CO2 solubility are poorly constrained, due to the restricted variations of these parameters in their database. Additionally, while their compositional term gives equal weight to Na2O and K2O, the calibration dataset only includes K-rich melts with a range of pressures, so the effect of substituting Na and K is poorly constrained. Finally, the authors warn that their regressions were performed assuming all Fe was Fe2+, so the dependence of oxidation state on CO2 solubility identified by Papale et al. (2006) is not incorporated in this model (see also Brooker et al., 2001b).

Their expression for H2O solubility has no dependence on melt composition; they suggest that due to the poor sensitivity of this parameter to melt composition in the calibration dataset, such fitting terms are statistically unjustified. While the calibration dataset for H2O is more chemically diverse than for CO2, the lack of a compositional parameter means that extreme caution is required when applying this model outside the calibration range.

**Shishkina et al., 2014**

The Shishkina et al. (2014) model calculates the solubility of H2O and CO2 using empirical expressions. The expression for CO2 was calibrated on a dataset of experiments on predominantly mafic compositions between 1200-1300˚C, and 500-5000 bars:

lnCO2 (ppm)=1.15ln(P(MPa))+6.71 (2)

The parameter accounts for the compositional dependence of CO2 solubility in terms of 7 cation species:

(1)

The expression for H2O solubility was calibrated on a dataset of experiments with mafic to intermediate compositions between 1200-1250˚C, and 485-5009 bars. Unlike the expression of Iacono-Marziano et al. (2012), it incorporates a composition parameter expressed in terms of the anhydrous cation fractions of Na and K.

A picture containing indoor, bird

Description automatically generated

In general, the compositional range of the Shishkina et al. (2014) dataset includes a larger variety of mafic compositions than that of Iacono-Marziano et al. (2012), particularly with respect to alkali-rich, or highly depolyerized melt compositions.

**MagmaSat: Ghiorso and Gualda, 2015**

Ghiorso and Gualda, (2015) present a comprehensive thermodynamic model for mixed H2O-CO2 solubility, calibrated on a very wide range of silicate melt compositions, called MagmaSat. MagmaSat is the most thermodynamically robust model available in VESIcal and as such is the default model. The model is internally consistent with Duan and Zhang (2006) and with rhyolite-MELTS and MELTS so can be integrated with a variety of calculations performed in these programs. The calibration dataset for H2O compositions spans ~550-1420˚C, and pressures of 0-20,000 bars, and for CO2 spans ~1140-1400 ˚C and 0-30,000 bars. MagmaSat is typically accessed via its Mac application, or through a web server, although the latter frequently fails to converge for basaltic compositions.

In recent years, the MELTS family of models (among others) has been integrated and rewritten in python as ENKI (Enabling Knowledge Integration; <http://enki-portal.org>), a recently introduced and continuously in-development model engine. The ENKI thermoengine package hosts MELTS (including MagmaSat) and allows a coder to use pieces of MELTS piecemeal. In VESIcal, the MagmaSat model is implemented via the ENKI thermoengine directly. Although the capability exists within it, thermoengine does not have pre-made built-in functions for volatile solubility calculations, and so VESIcal uses the underlying thermoengine code to create these functions. MagmaSat calculations through VESIcal can be relatively slow, however, due to the computational requirements of each calculation. That is, the equilibrium state of the system must be solved iteratively, with the relevant volatile information extracted at each step. Due to the nature in which both VESIcal and ENKI thermoengine are written (e.g., both in python and with extensibility in mind), they are somewhat interoperable. Because all of thermoengine is imported for use in VESIcal, any thermoengine calculation can be performed within, for example, this jupyter notebook. For example, one might use thermoengine to compute a fractional crystallization sequence of a melt during cooling and/or ascent, and then use VESIcal to compute the evolution of the composition of the fluid phase in tandem (either with MagmaSat or any other VESIcal model).

**Allison et al. 2019**

Allison et al. (2019) present thermodynamic models to calculate CO2 solubility for six different basaltic compositions from Stromboli (alkali basalt), Etna (trachybasalt), Vesuvius (phonotephrite), Erebus(phonotephrite), Sunset Crater (alkali basalt), and the San Francisco Volcanic Field (basaltic andesite). Specifically, this study performed experiments at 1200˚C, and ~400- 600 Mpa to address the paucity of experiments examining CO2 solubility in alkali systems at mid crustal pressures. In addition to these experiments, the models for Vesuvius, Etna and Stromboli compositions incorporate others from the literature (references), extending the calibration range of these three models to upper crustal pressures (see Table X). While Allison et al. (2019) only parameterize CO2 solubility, their experiments were run on melts with 2.1-5.3 wt% H2O, so the behaviour of mixed H2O-CO2 fluids must be taken into account when using this model. In the case studies in their paper, and the supplementary spreadsheet they provide for users to calculate solubilities or saturation pressures, the authors integrate their CO2 model with the H2O solubility model of Lesne et al. (2011). In VESIcal, you can do XYZ. As for the Iacono-Marziano model, the effect of iron speciation on CO2 solubility is not included.

## 2.2 Format of the python library

In this section, the basic organization and use cases of VESIcal are discussed. Specific details on how to perform model calculations are discussed in Section 3 and include worked examples. The VESIcal library is written so that users can interact first and foremost with the calculation they want to perform. Five standard calculations can be performed with any model in the library:

1. calculate\_dissolved\_volatiles()
2. calculate\_equilibrium\_fluid\_composition()
3. calculate\_saturation\_pressure()
4. calculate\_isobars\_and\_isopleths() (plus functionality for plotting; only for mixed volatiles models)
5. calculate\_degassing\_path() (plus functionality for plotting; only for mixed volatiles models)

Figure 1 illustrates the basic organization of the code. First, the user determines which calculation they wish to perform by accessing one of the five core calculation classes (listed above). In this step, the user specifies any input parameters needed for the calculation (e.g., sample composition, pressure, temperature) as well as the model they wish to use. The default model is MagmaSat, but the user may specify any model in the library. As an example, the code to calculate the saturation pressure of some sample using the MagmaSat model would be written as:

calculate\_saturation\_pressure(sample=mysample, temperature=850.0)

where mysample is a dictionary or pandas Series containing the composition of the sample in oxide wt%, and the temperature is given in ∘∘C. If a different model is desired, for example Shishkina et al. (2014), it can be passed as:

calculate\_saturation\_pressure(sample=mysample, temperature=850.0, model='Shishkina')

The core calculation classes each perform two functions: 1) a check is performed to ensure that the user input is within the model's recommended calibration range; 2) the calculate() method sends the user input to the appropriate model. Each model in the library is defined as a python class and contains model-specific methods for performing the five standard calculations. Standard pre-processing of the data (e.g., normalizing the composition to 100%, ignoring extranneous compositional information) is performed before the calculation and is uniquely defined for each model.

The VESIcal code has been designed to facilitate easy use by a user with any skill level in coding or in the python language. To facilitate this, we have structured the code as two basic end-member computational paths: a batch processing path (simpler) and a single sample path (more advanced options). The level of simplicity of either path is indirectly proportional to the level of customization and extensibility (Figure 2).

The "simplest" way to interact with the VESIcal code is via the batch processing path. Here, the user provides input data in the form of a Microsoft Excel spreadsheet (.xlsx file) and instructs the model to perform whatever calculation is desired. The model returns data formatted like a Microsoft Excel spreadsheet, which contains the user's original input data plus whatever model outputs were calculated. This path always operates in a "verbse" output style; that is, calculations return not only the specific calculated values requested but also any other relevant data calculated along the way. For example, a call to calculate\_saturation\_pressure() in the batch processing path returns, for every sample, the saturation pressure in bars, the mass of fluid present in grams, and the composition of the fluid present. After the calculation is performed, the user can continue to work with the returned data in their jupyter notebook or terminal window, or they can simply export the data to an excel file with one simple command save\_excel\_file().

The more advanced calculation path is actually the most fundamental as it allows the user access to the most basic model functions and even to hybridize models. Calculations are performed on a single sample; iterating over large datasets is left to the user to implement. However, this allows the user to input information in a variety of ways without being constrained to formatting the input data in a particular way or naming scheme. This also gives the user more flexibility in integrating any VESIcal model function into some other python code.

## 2.3 Extensibility[¶](http://localhost:8888/notebooks/manuscript/Manuscript.ipynb#2.3-Extensibility)

Some text on how awesome this tool is because it's written in Python and can easily be combined with other libraries.

### Penny do you have some initial thoughts for this section? Do we need this section?

## 2.4 Integration within the ENKI framework

Introduce ENKI

**some older text** MagmaSat+ is a volatile-focused tool capable of producing outputs relevant to volatile solubility and degassing processes. MagmaSat determines the saturation conditions for a mixed H2O-CO2 fluid in natural composition silicate melts (Ghiorso & Gualda, 2015). MagmaSat+ is a rewrite of the MagmaSat app and based on the thermodynamic model of Ghiorso & Gualda (2015). Our implementation in the ENKI framework based on Mark’s MELTS-v.1.2.0-equilibrium example notebook (ENKI Equilibrate package).

## 2.5 Running the code

This tool can be used in a number of ways, via this jupyter notebook or by directly accessing the code in any python script. The first and simplest way to use VESIcal is within this very manuscript, in the form of a jupyter notebook. Because this manuscript and VESIcal python library files are hosted on the ENKI server, code can be manipulated and executed in the code cells below. Making changes won't affect the public version of this manuscript. Second, if jupyter notebook is installed on your local machine, this manuscript notebook can be downloaded along with the VESIcal python library and run locally. Python3 and ENKI thermoengine must be installed, plus the dependencies specified below. Once dependencies are installed, simply ensure that both the notebook (manuscript.ipynb) and library (VESIcal.py) are in the same directory. Instructions on installing and running jupyter notebook can be found on the jupyter website (<https://jupyter.org/>). Third, the VESIcal python library can be downloaded via the github repository (<https://github.com/kaylai/MagmaSatPlus>) or installed directly via the python package manager pip using the command pip install VESIcal. The same dependencies, including Python3 and ENKI thermoengine, are required for this code to run.

**To run the code in this notebook**, nothing needs to be installed. Simply execute the code cells below, changing parameters as necessary. Custom data may be processed by uploading an excel file into the same folder containing this notebook and then changing the filename in Section 3.1.1.

**To run this code locally**, some python packages must first be installed:

* Python3 (<https://www.python.org/downloads/>)
* ENKI thermoengine (<https://gitlab.com/ENKI-portal/ThermoEngine>)
* Python pandas (<https://pandas.pydata.org/>)
* Numpy (<https://numpy.org/>)
* SciPy (<https://www.scipy.org/>)
* Matplotlib (<https://matplotlib.org/>)

Pandas, Numpy, SciPy, Matplotlib, and jupyter all come standard in Anaconda (<https://www.anaconda.com/products/individual>), a python package manager for use in data science.

## 2.6 Documentation

This manuscript serves as documentation of the VESIcal library aimed at python users of all levels. However, the code itself is documented with explanations of each method and its input parameters and returned values. This documentation can be accessed after VESIcal has been imported in Section 3.0.1 by typing help(v), which returns all of the documentation text directly from the code in one large block. An easier way to look through the documentation is at our readthedocs website (<https://vesical.readthedocs.io/>).

Video tutorials are also available on the VESIcal YouTube (link-to-youtube). Currently, tutorials cover: blah blah blah.

## 2.7 Generic methods for calculating mixed-fluid properties

VESIcal provides a set of methods for calculating the properties of mixed CO22-H22O fluids, which can be used with any combinations of H22O and CO22 solubility model. This allows additional models to be added to VESIcal by defining only the (simpler) expressions describing pure fluid solubility. Non-ideality of mixing in the fluid or magma phases can be incorporated by defining activity and fugacity models, which are then accessed by the pure-fluid models. We provide a summary here of how we approach calculating the properties of mixed-fluid systems.

### 2.7.1 Pure fluid solutions

Whilst the variables required for each pure-fluid solubility model are different, we take a very similar approach to calculating fluid properties for each. In general pure-fluid models provide analytical expressions for the concentration of a volatile species dissolved in a magma as a function of either its partial-pressure or fugacity in the vapour phase. To provide a homogeneous framework, each model must have a fugacity model associated with it. Where a solubility model uses partial-pressure in place of fugacity, the fugacity model is set to being an ideal gas.

In general, an analytical expression does not exist for the inverse calculation, i.e., calculating the saturation pressure given the concentration of dissolved volatile species. The calculation must then be solved numerically. Numerical methods obtain the answer by making guesses for the saturation pressure, checking whether the predicted concentration of the dissolved volatile species matches the measured value, and refining the guesses as appropriate. When the saturation pressure is found, this equality holds:

𝑉𝐿measured−𝑉𝐿predicted(𝑃,...)=0VmeasuredL−VpredictedL(P,...)=0

When the guessed pressure is too high the left hand side of the equation will be negative, and if the guessed pressure is too low it will be positive, We therefore require an algorithm to find the root of the equation. To this end we employ the scipy.optimize.root\_scalar() method. The advantage of using a generic scipy algorithm, in preference to creating a bespoke routine, is that it affords us flexibility in how the problem is solved, and the algorithm is underlain by robust, well-tested, efficient code.

### 2.7.2 Finding saturation pressure of mixed-fluid

Describe how scipy.optimize is used here

### 2.7.3 Finding the equilibrium fluid composition

When calculating the dissolved volatile concentrations in a progressively degassing magma, we break the calculation into steps, each with a small decrement in pressure. As pressure decreases the mass of fluid in equilibrium with the magma increases and its composition will change. At each stage we must, therefore, solve for the mass and composition of the fluid in equilibrium with the magma, given the total mass of volatiles in the system. This calculation is implemented in MixedFluid.calculate\_equilibrium\_fluid\_comp(), which in turn is called by MixedFluid.calculate\_degassing\_path(). Here we summarise the methodology for the calculation.

Mass balance in the system can be expressed in terms of the mole fractions of H22O (𝑋H2OXH2O) and CO22 (𝑋CO2XCO2) in the silicate-liquid (𝐿L), the H22O-CO22O fluid (𝐹F), and the system total (𝑇T), and the fraction of fluid present (𝑓f).

𝑋𝐹CO2+𝑋𝐹H2O=1XCO2F+XH2OF=1

(1−𝑓)𝑋𝐿CO2+𝑓𝑋𝐹CO2=𝑋𝑇CO2(1−f)XCO2L+fXCO2F=XCO2T

(1−𝑓)𝑋𝐿H2O+𝑓𝑋𝐹H2O=𝑋𝑇H2O(1−f)XH2OL+fXH2OF=XH2OT

The fraction of fluid (𝑓f) can be eliminated by substitution, obtaining Equation --NUMBER-- of --CITATION--:

𝑋𝑇CO2−𝑋𝐿CO2𝑋𝐹CO2−𝑋𝐿CO2−𝑋𝑇H2O−𝑋𝐿H2O𝑋𝐹H2O−𝑋𝐿H2O=0XCO2T−XCO2LXCO2F−XCO2L−XH2OT−XH2OLXH2OF−XH2OL=0

In principle, this equation allows us to numerically solve for mass balance; however, the equation is valid only for both 𝑋𝐹CO2≠𝑋𝐿CO2XCO2F≠XCO2L and 𝑋𝐹H2O≠𝑋𝐿H2OXH2OF≠XH2OL. When 𝑋𝐹CO2→𝑋𝐿CO2XCO2F→XCO2L or 𝑋𝐹H2O→𝑋𝐿H2OXH2OF→XH2OL the equation tends to ±∞±∞, which the scipy.optimize.root\_scalar() routine will mistake for a root of the equation. The regions over which the expression is not infinite can be found; however we find a more reliable approach when using the scipy.optimize library is to use the equation in a different form.

Rearranging the CO22 mass-balance expression allows the fraction of fluid (𝑓f) to be calculated:

𝑓=𝑋𝑇CO2−𝑋𝐿CO2𝑋𝐹CO2−𝑋𝐿CO2f=XCO2T−XCO2LXCO2F−XCO2L

This is valid for 𝑋𝐹CO2≠𝑋𝐿CO2XCO2F≠XCO2L. It is much more unlikely that a solution will exist near 𝑋𝐹CO2=𝑋𝐿CO2XCO2F=XCO2L than near 𝑋𝐹H2O=𝑋𝐿H2OXH2OF=XH2OL. By not directly eliminating 𝑓f by dividing through by both (𝑋𝐹CO2−𝑋𝐿CO2XCO2F−XCO2L) and (𝑋𝐹H2O−𝑋𝐿H2OXH2OF−XH2OL) we significantly increase the reliability of using this expression with scipy.optimize.root\_scalar(). In this expression, the only known value is 𝑋𝑇CO2XCO2T. However, if a guess is provided for 𝑋𝐹CO2XCO2F, the partial pressure of CO22 can be calculated and the corresponding value for 𝑋𝐿CO2XCO2L obtained. In general the guess for 𝑋𝐹CO2XCO2F will not satisfy mass balance when the same parameters are applied to both CO22 and H22O.

Since 𝑋𝐹H2O=1−𝑋𝐹CO2XH2OF=1−XCO2F, when we provide a guess for 𝑋𝐹CO2XCO2F we also fix 𝑋𝐹H2OXH2OF. This in turn fixes the partial pressure of H22O, allowing us to calculate 𝑋𝐿H2OXH2OL consistent with that guess. We can then test whether our guess for 𝑋𝐹CO2XCO2F satisfies mass balance by checking for the equality:

(1−𝑓)𝑋𝐿H2O+𝑓𝑋𝐹H2O−𝑋𝑇H2O=0(1−f)XH2OL+fXH2OF−XH2OT=0

The value of 𝑋𝐹CO2XCO2F that uniquely satisfies mass balance can then be found numerically by solving this equation using scipy.optimize.root\_scalar().

# 3. Workable example uses

[workable example stuff in notebook]

# 4. Discussion and Conclusion

some text

## 4.1 Model Comparisons

### We can use one sample composition to show comparisons between model calcs, user can change this sample to whatever they want! This becomes a useful tool for anyone to automatically compare model results for all calcs for any sample of interest.

In [ ]:

magmasat\_vols **=** myfile.calculate\_dissolved\_volatiles(temperature**=**900.0,

pressure**=**2000.0, X\_fluid**=**0.5, verbose**=False**)

​

*# shish\_vols = myfile.calculate\_dissolved\_volatiles(model='Shishkina', temperature=900.0,*

*# pressure=2000.0, X\_fluid=0.5)*

​

*# dix\_vols = myfile.calculate\_dissolved\_volatiles(model='Dixon', temperature=900.0,*

*# pressure=2000.0, X\_fluid=0.5)*

​

*# iac\_vols = myfile.calculate\_dissolved\_volatiles(model='IaconoMarziano', temperature=900.0,*

*# pressure=2000.0, X\_fluid=0.5)*

In [ ]:

models **=** v.get\_models(models**=**'mixed')

fig, ax **=** plt.subplots(len(models), figsize**=**(12,24))

fig.subplots\_adjust(hspace**=**.5)

​

**for** i **in** range(len(models)):

print(models[i])

ax[i].set\_ylabel('Dissolved H$\_2$O wt% \n' **+** str(models[i]))

ax[i].set\_xlabel('Dissolved H$\_2$O wt%, MagmaSat')

result **=** myfile.calculate\_dissolved\_volatiles(model**=**models[i], temperature**=**1200.0,

pressure**=**2000.0, X\_fluid**=**0.5)

ax[i].scatter(magmasat\_vols["H2O\_liq\_VESIcal"],

result["H2O\_liq\_VESIcal"])

xy\_1to1 **=** np.linspace(result["H2O\_liq\_VESIcal"].min()**-**0.5,result["H2O\_liq\_VESIcal"].max()**+**0.5,100)

ax[i].plot(xy\_1to1, xy\_1to1)

In [ ]:

​

In [ ]:

​

In [ ]:

*# MS\_isobars, MS\_isopleths = v.calculate\_isobars\_and\_isopleths(sample=my\_sample,*

*# temperature=900.0,*

*# pressure\_list=[2000.0],*

*# isopleth\_list=[0.5],*

*# print\_status=True).result*

​

Shish\_isobars, Shish\_isopleths **=** v.calculate\_isobars\_and\_isopleths(model**=**'Shishkina',

sample**=**my\_sample,

temperature**=**1200.0,

pressure\_list**=**[2000.0],

isopleth\_list**=**[0.5]).result

​

Dix\_isobars, Dix\_isopleths **=** v.calculate\_isobars\_and\_isopleths(model**=**'Dixon',

sample**=**my\_sample,

temperature**=**1200.0,

pressure\_list**=**[2000.0],

isopleth\_list**=**[0.5]).result

​

Iac\_isobars, Iac\_isopleths **=** v.calculate\_isobars\_and\_isopleths(model**=**'IaconoMarziano',

sample**=**my\_sample,

temperature**=**1200.0,

pressure\_list**=**[2000.0],

isopleth\_list**=**[0.5]).result

In [ ]:

v.plot\_isobars\_and\_isopleths([Shish\_isobars, Dix\_isobars, Iac\_isobars],

[Shish\_isopleths, Dix\_isopleths, Iac\_isopleths])

v.plot\_isobars\_and\_isopleths(Shish\_isobars, Shish\_isopleths)

## 4.2 Example degassing calculations -- and some warnings (need better title)

Take in an MI data-set (Bishop Tuff, Kilauea). Plot degassing curve, isobars/isopleths using one composition. Plot MI compositions on top of this. Give warnings that, although this is a common way to do this, it's problematic!

### Gordon can you take a crack at this text? I can help plot up the data!

# 5. Future Work

# Acknowledgements

# References