RESEARCH ARTICLE

VOLCANICA

pyMelt: An extensible Python engine for mantle melting calculations

Simon Matthews*[α], Kevin Wong[β], and Matthew Gleeson[γ]

[α] Institute of Earth Sciences, University of Iceland, Reykjavík, Iceland.
[β] School of Earth and Environment, University of Leeds, Leeds, UK.
[γ] Dept. Earth and Planetary Sciences, University of California Berkeley, California, USA.

Abstract

Modelling the melting of Earth's mantle is crucial for understanding the distribution of volcanic activity on Earth and for testing models of mantle convection and mantle lithological heterogeneity. pyMelt is a new open-source Python library for calculating the melting behaviour of multi-lithology mantle and can be used to predict a number of geophysical and petrological observations, including melt productivity, spreading centre crustal thickness, lava trace element concentrations, and olivine crystallisation temperatures. The library is designed to be easily extensible, allowing melting models to be added, different methods for calculating lava chemistry to be applied, and new melting dynamics and properties to be incorporated.

Keywords: Mantle melting; Open-source software; Geochemistry; Crustal thickness; Magmatic productivity.

1 Introduction

Models for melting of Earth's convecting mantle can provide quantitative constraints on its thermal variability [e.g. Ball et al. 2021], compositional variability [e.g. Brown and Lesher 2014; Gleeson et al. 2021], and primary melt compositions [e.g. Jennings et al. 2016]. These melting models calculate mantle melting behaviour either by minimising thermodynamic potentials at each calculation step [e.g. Smith and Asimow 2005], or by using expressions parameterised directly from melting experiments [e.g. Lambart et al. 2016; Krein et al. 2020]. The parameterised approach is particularly useful for calculations requiring many runs of a melting model, for example when inverting for mantle properties from geochemical or geophysical observations [e.g. McKenzie and O'Nions 1991; Matthews et al. 2021].

Here, we present and describe pyMelt, an open-source extensible Python library that employs the parameterised approach, providing a powerful and flexible tool for calculating the melting behaviour of lithologically heterogeneous mantle. This library allows users to calculate melt productivity during isobaric melting or adiabatic decompression melting at an assigned mantle potential temperature T_p. Lithologies and melting parameterisations employed to calculate melt productivity can be user-defined or chosen from a list of published models (Section 3). In addition, calculated melt pathways can be used to estimate igneous crustal thickness, lava trace element concentrations, and olivine crystallisation temperatures at spreading centres or intra-plate settings (Section 6 and 7). More complex melting scenarios and other geochemical and geophysical predictions can be built from the existing library.

A number of software tools are available that can perform similar, though in some respects more limited, calculations, including INVMEL [McKenzie and O'Nions 1991], REEBOX-PRO [Brown and Lesher 2016], MELT-PX [Lambart et al. 2016], Petrogen [Krein et al. 2020], and BDD21 [Ball et al. 2022]. While these packages have been used extensively in studies of mantle melting, pyMelt offers a number of advantages. pyMelt is the only package that is simultaneously open-source, incorporates mantle lithological heterogeneity, and can be easily integrated with other Python libraries (e.g. Monte Carlo inversion tools).

Open-source software is an essential part of open, transparent, and reproducible science, and it provides the basis for the development of more advanced codes and integration with other libraries. The importance of modelling the effects of lithological heterogeneity on mantle melting behaviour for accurately predicting magmatic productivity, compositions, and temperatures is becoming increasingly clear [Phipps Morgan 2001; Pertermann and Hirschmann 2003; Ito and Mahoney 2005; Shorttle et al. 2014; Matthews et al. 2021]. pyMelt therefore occupies an important niche in mantle melting calculations and will provide the basis for solving new problems related to melt generation and magmatism for many years to come.

In this manuscript we review the main features of pyMelt, its library structure, its underlying mathematical formulation, and the computational approaches it takes. Users are directed to the pyMelt documentation[†] for a comprehensive guide to using the library, and to the interactive tutorials available online via myBinder[‡]. The pyMelt repository is hosted on GitHub[§], where the code can be obtained, bugs reported, new features requested, and new contributions made. pyMelt can also be installed directly using the pip package manager.

1.1 Terminology

Throughout this manuscript we use Python terminology in our description of the library's implementation. Here we give a short description of these terms, but it is not necessary to understand this terminology to use pyMelt:

- Class: A collection of methods and properties which serves as a template for creating an object. For example, the properties of the mantle are contained in a class.

[†]https://pyMelt.readthedocs.io
[‡]https://mybinder.org/v2/gh/simonwmatthews/pyMelt/HEAD?labpath=docs%2Ftutorial%2Ftutorial1.ipynb
[§]https://www.github.com/simonwmatthews/pyMelt

*✉ simonm@hi.is

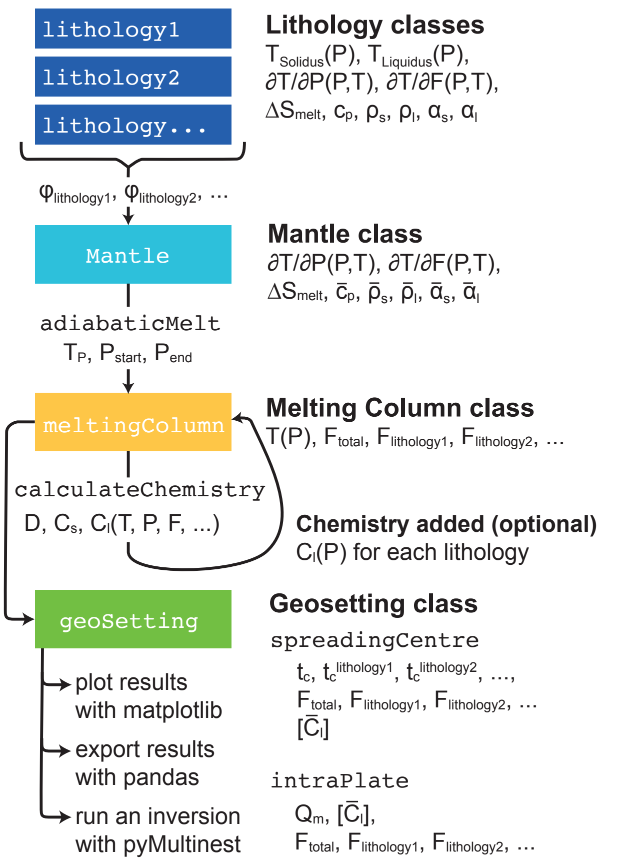
Editor: J. Farquharson | Typesetter: J. Farquharson

- Method: An algorithm that belongs to a class that may use user-input, properties of the class, or other methods of the class to produce a result. For example, the solidus temperature of the mantle is calculated using a method of the mantle class.
- Object: An instance of a class which is used by the library to perform calculations. For example, `pyMelt` calculations are performed with a mantle object, created from the mantle class.

More extensive definitions can be found in the `Python` documentation.

2 pyMelt STRUCTURE

Figure 1 summarises the modular structure of the `pyMelt` library and its workflow. The melting behaviours of individual lithologies (e.g. lherzolite or pyroxenite) are contained within `lithology` objects (Section 3), which can be combined in specified mass fractions ϕ to form a `Mantle` object (Section 5). Any `lithology` object in `pyMelt` can be converted to a `hydrousLithology` object where the effect of water on its solidus and melt productivity is estimated (Section 4). An adiabatic decompression calculation can then be performed on a `Mantle` object at a specified mantle potential temperature T_p (the temperature a parcel of mantle would have following decompression to 0 GPa while undergoing no chemical changes) using its `adiabaticMelt` method (Section 5) which returns a `meltingColumn` object. Alternatively, an isobaric melting calculation can be performed at a specified temperature using the `isobaricMelt` method. If desired, the trace element concentrations in these melts can be calculated by calling the `calculateChemistry` method of the `meltingColumn` object (Section 6). To calculate geological setting specific properties, such as crustal thickness at a mid-ocean ridge, a `geoSetting` object can be created using the `meltingColumn` object (Section 7). The results can then be extracted from the `geoSetting` object and plotted, used in further calculations, or saved (see the tutorial notebooks).



lithology1
lithology2
lithology...
Lithology classes
$T_{Solidus}(P)$, $T_{Liquidus}(P)$, $\partial T/\partial P(P,T)$, $\partial T/\partial F(P,T)$, ΔS_{melt}, c_p, ρ_s, ρ_l, α_s, α_l
$\varphi_{lithology1}$, $\varphi_{lithology2}$, ...
Mantle
Mantle class
$\partial T/\partial P(P,T)$, $\partial T/\partial F(P,T)$, ΔS_{melt}, $\bar{c}_p$, $\bar{\rho}_s$, $\bar{\rho}_l$, $\bar{\alpha}_s$, $\bar{\alpha}_l$
adiabaticMelt
T_P, P_{start}, P_{end}
meltingColumn
Melting Column class
$T(P)$, F_{total}, $F_{lithology1}$, $F_{lithology2}$, ...
calculateChemistry
D, C_s, $C_l(T, P, F, ...)$
Chemistry added (optional)
$C_l(P)$ for each lithology
geoSetting
Geosetting class
spreadingCentre
t_c, $t_c^{lithology1}$, $t_c^{lithology2}$, ...,
F_{total}, $F_{lithology1}$, $F_{lithology2}$, ...
$[\bar{C}_l]$
intraPlate
Q_m, $[\bar{C}_l]$,
F_{total}, $F_{lithology1}$, $F_{lithology2}$, ...
plot results with matplotlib
export results with pandas
run an inversion with pyMultinest

Figure 1: Summary of the structure of `pyMelt`. Each box represents an instance of a `pyMelt` object, either created directly by the user (with user-defined variables as specified on the arrows) or returned by a method call. The properties and methods of each class of objects are shown on the right hand side (symbols as defined in the text).

3 MANTLE LITHOLOGIES

The experiments that are used to parameterise melting models are performed on particular bulk compositions, or lithologies. This means that each melting parameterisation, unless parameterised also for bulk composition [e.g. Lambart et al. 2016], represents a particular lithology, with its own melting behaviour. At a minimum, a `lithology` object has methods defined for the solidus temperature $T_{\text{solidus}}(P)$ (`TSolidus`), liquidus temperature $T_{\text{liquidus}}(P)$ (`TLiquidus`), and for the melt fraction $F(P, T, T_{\text{solidus}}, T_{\text{liquidus}})$ (`F`), where P and T are pressure and temperature. The models contained within `pyMelt` already have methods for $(\partial T/\partial F)_P$ (`dTdF`) and $(\partial T/\partial P)_F$ (`dTdP`), but `pyMelt` can also calculate their values numerically from the `TSolidus`, `TLiquidus`, and `F` methods. Additionally, each lithology object has properties associated with it for the density of the solid lithology ρ_s and its melt ρ_l, their thermal expansivities α_s and α_l respectively, the heat capacity C_p, and the entropy change on melting ΔS.

Table 1 lists the melting parameterisations for which `lithology` classes are defined in `pyMelt`. The pyroxenitic lithologies are categorised as having a relative excess or deficit in silica, with the silica-excess parameterisations in `pyMelt` representing a mid-ocean ridge basalt like composition, and the silica-deficient pyroxenites representing a mixture of basalt and lherzolite. The lithologies included already in `pyMelt` (with the exception of McKenzie and Bickle [1988]) have simple analytical expressions for $F(P, T)$; however, more complex models could be included, providing a method for $F(P, T)$ can be created. For example, the widely used parameterisations in the `Petrogen` software [Krein et al. 2020] could be added to `pyMelt` in the future, providing a method is created for numerically solving the more complex algorithms they use; however, calculations using such a model would likely be much slower to compute.

4 HYDROUS MELTING

Any lithology in `pyMelt` can be turned into a hydrous lithology using the `hydrousLithology` class. To approximate the effect of hydrous melting a similar formulation to that developed by Katz et al. [2003] is used. In this formulation the solidus temperature is depressed according to their Equation 16:

$$T_{\text{solidus}}^{\text{hydrous}} = T_{\text{solidus}} - KX_{\text{H}_2\text{O}}^{\gamma} \tag{1}$$

Table 1: The pure lithology melting parameterisations built into `pyMelt`. See Lambart et al. [2016] for a description of the pyroxenite classification.

Reference	Class name	Lithology type
McKenzie and Bickle [1988]	`mckenzie.lherzolite`	Lherzolite
Pertermann and Hirschmann [2003]	`pertermann.g2`	Pyroxenite (silica-excess)
Katz et al. [2003]	`katz.lherzolite`	Lherzolite
Shorttle et al. [2014]	`shorttle.kg1`	Pyroxenite (silica-deficient)
	`shorttle.harzburgite`	Harzburgite (non-melting)
Matthews et al. [2021]	`matthews.klb1`	Lherzolite
	`matthews.kg1`	Pyroxenite (silica-deficient)
	`matthews.eclogite`	Pyroxenite (silica-excess)
Ball et al. [2022]	`ball.depleted_mantle`	Lherzolite
	`ball.primitive_mantle`	Lherzolite
	`ball.mixed_mantle`	Lherzolite

where K and γ are constants and X_{H_2O} is the water concentration in the melt in wt.%. This differs slightly from the equations developed by Katz et al. [2003] as they did not apply the $-KX_{H_2O}^{\gamma}$ term to every instance of T_{solidus} in their expressions (e.g. the denominator of their Equation 19). The amount of water that can be dissolved in magmas before a H_2O vapour phase is exsolved is limited, but increases with pressure. Katz et al. [2003] model this effect using:

$$X_{H_2O}^{\text{sat}} = \chi_1 P^{\lambda} + \chi_2 P, \quad 0 < \lambda < 1 \tag{2}$$

where χ_1, χ_2, and λ are constants. This equation is also implemented in `pyMelt`.

The concentration of H_2O present in the melt decreases as melting proceeds, owing to H_2O partitioning favourably into the melt and being continually diluted by new additions of magma. Katz et al. [2003] modelled this change by using the batch melting equation:

$$X_{H_2O} = \frac{X_{H_2O}^{\text{bulk}}}{D_{H_2O} + F(1 - D_{H_2O})} \tag{3}$$

where D_{H_2O} is the partition coefficient of H_2O during melting. When a vapour phase is saturated, i.e. $X_{H_2O} > X_{H_2O}^{\text{sat}}$, X_{H_2O} is replaced by $X_{H_2O}^{\text{sat}}$ in Equation 1 (which is then used to calculate the values of the other melting functions).

We extend this formulation to consider also the removal of H_2O from the system by near-fractional melting, as modelled previously by Asimow et al. [2003]; however, since the melting models in `pyMelt` are implicitly expressions of batch (or equilibrium) melting, modelling the effect of H_2O extraction by fractional melting cannot be done entirely self-consistently. This effect is approximated in `pyMelt` by replacing Equation 3 with an expression for near-fractional melting:

$$X_{H_2O} = \frac{X_{H_2O}^{\text{bulk}}}{(1-\Phi)D_{H_2O} + \Phi}(1-F)^{\frac{(1-\Phi)(1-D_{H_2O})}{(1-\Phi)D_{H_2O}}} \tag{4}$$

where Φ is the porosity during melting.

When a `hydrousLithology` object is created, the supporting methods from the original `lithology` object are copied, along with the `TLiquidus` method (which is not changed by the hydrous melting extension). A new method for `TSolidus` is defined, which applies Equation 1 to the original `TSolidus` method. Since the value of F depends on X_{H_2O}, which itself depends on F, a new `F` method is created, which solves the equation:

$$F_{\text{calc}}(P, T, F_{\text{guess}}) - F_{\text{guess}} = 0 \tag{5}$$

where $F_{\text{calc}}()$ is the original `F` method (which will provide the hydrous melt fraction as it calls the modified `TSolidus` method) and F_{guess} is the value changed by the root finding method. `pyMelt` uses the `brentq` algorithm implemented in the `SciPy.optimize.root_scalar` method [Virtanen et al. 2020]. Hydrous melting also changes the other melting functions, so new methods for $\left(\frac{\partial T}{\partial F}\right)_P$ and $\left(\frac{\partial T}{\partial P}\right)_F$ are created, which calculate the values numerically using `SciPy.misc.differentiate`, alongside `SciPy.optimize.root_scalar` to find the T-P curve at constant F.

The default values for K, γ, χ_1, χ_2, λ, and D_{H_2O} are taken from Katz et al. [2003] who calibrated them for hydrous-lherzolite melting. While `pyMelt` provides the opportunity to model hydrous-pyroxenite melting in the same way, the user must choose appropriate constant values for pyroxenite.

5 Mantle melting

Before melting calculations can be performed, the `lithology` objects must be assembled in a `Mantle` object, with their relative mass fractions specified. `pyMelt` does not limit the number of lithologies that can be assembled in a `Mantle` object, though in most situations one each of a lherzolite, pyroxenite, and harzburgite lithology are sufficient. The `Mantle` objects replicate many of the properties of the `lithology` objects (Figure 1), with methods returning either the mass-weighted average properties, or an array with the value of each lithology. Implicit in our treatment of the lithology objects, and our application of the melting formulation by Phipps Morgan [2001] for adiabatic decompression melting, is an assumption of complete thermal equilibrium but complete chemical disequilibrium between lithologies.

5.1 Point melting

The simplest melting calculation that `pyMelt` can perform is at a single set of conditions; either at a particular P and T using the `mantle.F()` method, or at a particular P and entropy (expressed by the mantle potential temperature T_p) using the `mantle.isobaricMelt()` method. When calculating the melt fraction for a given P and T, `pyMelt` calls the `F()` methods for each lithology, returning each in an array. Isentropic melting at a particular P are performed according to the method for isobaric melting in Matthews et al. [2021] in two steps. First the entropy change on cooling the mantle to its solidus is calculated:

$$\Delta S_{\text{cooling}} = \overline{C}_p \ln\left(\frac{T_{\text{solidus}}}{T}\right) \tag{6}$$

where $\overline{C}_p$ is the C_p averaged over each lithology. Melting then proceeds by isobaric heating in small temperature increments δT until:

$$\sum \delta S_{\text{melting}} = \Delta S_{\text{cooling}} \tag{7}$$

where:

$$\delta S_{\text{melting}} = \sum^{i} (\phi_i \Delta S_i^m \delta F_i) + \frac{\overline{C}_p}{T}\delta T \tag{8}$$

summing over each lithology i, where δF_i is the increment of melt fraction corresponding to the increment δT.

5.2 Adiabatic decompression melting

Adiabatic decompression melting calculations are performed by the `adiabaticMelt` method of the `Mantle` object, requiring only that a value for T_p is specified. By default the calculation will begin at the solidus and end at 0.01 GPa with a pressure decrement of 0.004 GPa at each decompression step.

The calculation proceeds by simultaneously integrating dF/dP for each lithology i, and dT/dP for the melting assemblage, to obtain the melt fractions (F_i) of each lithology and the mantle temperature (T) at each step. The value of dF/dP is determined for each melting lithology using Equation 29 of Phipps Morgan [2001] (the mass-weighted average values of c_p, α, and density ρ are indicated with a bar):

$$\frac{dF_i}{dP} = -\frac{\frac{\overline{c_p}}{T}\frac{\partial T_i}{\partial P} - \frac{\overline{\alpha}}{\overline{\rho}} \sum\limits_{n \neq i} \left[\phi_n \Delta S_n^m \frac{\frac{\partial T_i}{\partial P} - \frac{\partial T_n}{\partial P}}{\frac{\partial T_n}{\partial F_n}} \right]}{\phi_i \Delta S_i^m + \sum\limits_{n \neq i} \left[\phi_n \Delta S_n^m \frac{\frac{\partial T_i}{\partial F_i}}{\frac{\partial T_n}{\partial F_n}} \right] + \frac{\overline{c_p}}{T}\frac{\partial T_i}{\partial F_i}}. \tag{9}$$

The value of dT/dP is then obtained from Equation 28 of Phipps Morgan [2001] using the values for one lithology j (arbitrarily, the one with the most negative dF/dP in `pyMelt`):

$$\frac{dT}{dP} = \frac{dT_j}{dP} + \frac{dT_j}{dF_j}\frac{dF_j}{dP} \tag{10}$$

The integration is performed using a fourth-order Runge-Kutta routine. The results of the melting calculation are returned as a `meltingColumn` object, which records the melt fractions of each lithology, in addition to the aggregate melt fraction and the temperature at each pressure step.

When the calculation is started at a specified pressure, high mantle T_p may mean the mantle has already exceeded its solidus. In this case an interval of isobaric melting will occur before decompression starts, such that entropy is conserved (Equations 6–8). Decompression melting then proceeds, as described above.

The Phipps Morgan [2001] melting formulation assumes batch melting, whereby the melt is not separated from the solid residue. Therefore, in cases where a lithology j is in thermal equilibrium with a more fusible lithology, the heat extracted by melting of the more fusible lithology can cause lithology j to refreeze, i.e. $dF_j/dP > 0$. By default `pyMelt` will prevent freezing from occurring by setting $dF_j/dP = 0$, thereby more closely representing continual melt extraction. This is set as the default behaviour because the `chemistry` module requires monotonically increasing melt fractions.

5.3 Melting hydrous lithologies

`pyMelt` allows any lithology to be used in decompression melting calculations; however, when hydrous lithologies are modelled by assuming their H_2O is not extracted it is unlikely to correspond to a realistic melting scenario. Nevertheless, `pyMelt` can calculate the evolution of melt fraction (including vapour exsolution and the accompanying melt freezing) that such a hydrous lithology would undergo during adiabatic decompression. A demonstration of such calculations is provided in the `pyMelt` documentation. `pyMelt` could be extended in the future with new melting methods and geosettings classes (Section 7) to make use of the hydrous lithologies for modelling subduction zone melting.

6 TRACE ELEMENTS

Following the creation of a `meltingColumn` object by the `adiabaticMelt` method, the `pyMelt chemistry` module can be used to calculate the concentration of each trace element k within the melt $C_{k,l}$ (Figure 1). The calculation is performed by the `meltingColumn.calculateChemistry` method and requires the concentration of each trace element in each lithology $C_{k,s}$, in addition to the parameters required by the chemical model (e.g. the partition coefficients D_i). There are four built in chemical models: batch melting, near-fractional melting (instantaneous and accumulated melts), and the `INVMEL` forward model [McKenzie and O'Nions 1991; White et al. 1992]. For the batch and near-fractional melting models the partition coefficient can either be a constant, or a user-defined function of F, P, and T.

Each element (in each lithology) to be included in the calculation is defined as a `species` object that contains its solid concentration c_0 and a `composition` method for calculating the melt composition as a function of F (and possibly P and T). Defining each element separately permits the incorporation of more complex partitioning behaviour for some elements alongside simpler models for other elements. Generally, users will be unaware of `species` objects: the `meltingColumn.calculateChemistry` method can assemble them automatically.

The `INVMEL` model for lherzolite melting incorporates the effects of phase changes and phase exhaustion on the partioning of trace elements, in particular the effects of garnet- and clinopyroxene-present melting, but requires many more parameters to be defined. The partition coefficients used by default are those compiled by Gibson and Geist [2010], and other parameters are set to the values used by Ball et al. [2021].

For convenience, the chemistry module has a number of estimates for partition coefficients and mantle trace element concentrations built in (see the documentation for more details).

7 Geological settings

In many cases the information provided by the meltingColumn object is sufficient; in other cases a user may be interested in derived properties for a particular geological setting, aggregate melt compositions, for example. The pyMelt geoSetting classes (spreadingCentre and intraPlate) provide this facility, taking a meltingColumn object as an input (Figure 1).

When calculating aggregate properties of the melting region we must consider how each melt in the melting column should be weighted to account for mantle flow. For example, active upwelling in a mantle plume causes more mantle material to pass through the melting region at its base [Maclennan et al. 2001], meaning deeper melts should have a greater weighting in plume models. User-defined weighting functions may be specified for each calculation, but how they are implemented varies between geoSetting classes. An example weighting function built into pyMelt (geosettings.**weighting_expdecay**) has the form:

$$w(P) = \mu \exp\left(-\frac{1}{\lambda_w}\frac{P_{\max} - P}{P_{\max} - P_{\min}}\right) \tag{11}$$

where λ_w and μ are constants, and $P_{\max}$ and $P_{\min}$ are the maximum and minimum pressures from which melts are formed at the geological setting.

7.1 Spreading centres at steady state

When a spreadingCentre object is initialised, the crustal thickness t_c will be calculated, assuming passive corner-flow mantle upwelling [Plank and Langmuir 1992]. To account for the triangular melting region, the total melt fraction is integrated over the melting column, until the pressure exerted by the crust (calculated by stepwise integration with the trapezium rule) is equal to the pressure of the melting step:

$$t_c = \frac{1}{g\rho_c}\int_{P_{\text{start}}}^{P_{\text{crust}}} (1+w)\frac{\sum \phi_i F_i}{1-\sum \phi_i F_i} dP \tag{12}$$

where g is the acceleration due to gravity on Earth, ρ_c is the density of the crust, and P_{crust} is the pressure at the base of the crust. The term w is the optional user-defined weighting function, which takes $w = 0$ by default. Using the form $1 + w$ allows separation of the passive upwelling component and the active upwelling component. The $(1-\sum \phi_i F_i)$ term in the denominator accounts for compaction, i.e. mantle material will continuously replace the volume lost due to melt extraction [White et al. 1992]. The contributions of each lithology to the aggregate crust is calculated similarly:

$$t_c^n = \frac{1}{g\rho_c}\int_{P_{\text{start}}}^{P_{\text{crust}}} (1+w)\frac{\phi_n F_n}{1-\sum \phi_i F_i} dP. \tag{13}$$

When modelling continental rifts, the pressure exerted by the lithosphere can be imposed and the thickness of igneous crust calculated.

If the meltingColumn object used to generate the spreadingCentre object has chemistry, upon initialisation of the spreadingCentre object, the composition of the homogenised melt is calculated. Similarly to the crustal thickness calculations, homogenisation of chemistry takes into account the triangular melting region, compaction and any additional weighting function. The equation is modified from McKenzie and O'Nions [1991]:

$$\overline{C}_k = \frac{\int_{P_{\text{start}}}^{P_{\text{end}}} (1+w)\frac{\sum \phi_i F_i C_{k,l,i}}{1-\sum \phi_i F_i} dP}{\int_{P_{\text{start}}}^{P_{\text{end}}} (1+w)\frac{\sum \phi_i F_i}{1-\sum \phi_i F_i} dP} \tag{14}$$

and is evaluated using the trapezium rule. If the calculated melt compositions represent instantaneous and not batch melts, each column is first homogenised, with each melt weighted according to its lithology fraction and melt fraction, as above. The weighting function $w(P)$ is not applied in this step.

The crystallisation temperature of melts extracted from the top and base of the melting region can be calculated, using the method described by Matthews et al. [2016]. The olivine saturation temperature at the pressure of magma storage is found using the pressure dependence of the olivine saturation surface [39.16 K GPa^{-1}, Putirka 2008]. This method is available also in the intraPlate geoSetting class.

7.2 Intra-Plate settings at steady state

To initialise an intraPlate geoSetting object, the pressure at the base of the lithosphere must be provided. The calculation results stored in the meltingColumn will be truncated at that pressure. If the relative density of the mantle is provided ($\Delta\rho = \rho_{\text{ambient-mantle}} - \rho_{\text{plume-mantle}}$), the melt flux Q_m is calculated during initialisation using the equation:

$$Q_m = \frac{\pi}{8}\frac{\Delta\rho g r^4}{\mu}\sum \phi_i \int_0^{F_{i(\max)}} w\, dF_i \tag{15}$$

which is modified from the equation for volume flux through a deformable conduit [Turcotte and Schubert 2002]. $F_{i(\max)}$ is the melt fraction of lithology i at the top of the conduit, r is the conduit radius (default: 100 km), and μ is the viscosity of the plume (default: 10^{19} Pa s), with the default values taken from Shorttle et al. [2014].

If the meltingColumn object used to initialise the intraPlate object has chemistry and the melt compositions represent instantaneous melts, they will be homogenised during initialisation according to:

$$\overline{C_k} = \frac{\sum \phi_i \int_0^{F_{i(\max)}} w\, C_{k,l,i}\, dF_i}{\sum \phi_i \int_0^{F_{i(\max)}} w\, dF_i}. \tag{16}$$

If the melt compositions represent accumulated melts and there is a weighting function applied, no result will be returned.

8 Applications

pyMelt has already been used in three published studies, with more studies having been submitted for publication. Matthews et al. [2021] inverted pyMelt to extract estimates of mantle T_p, pyroxenite fraction, and harzburgite fraction for a number of igneous provinces by matching petrological estimates of crystallisation temperature and geophysically determined magmatic productivity. Gleeson et al. [2021] used pyMelt to estimate the contribution of pyroxenite melts to Galápagos lavas as a function of mantle T_p and proportion of pyroxenite in the mantle source. Harðardóttir et al. [2022] used pyMelt as the starting point for calculating magma compositions generated by melting a lherzolitic and pyroxenitic mantle, thereby testing models for the petrogenesis of Icelandic basalts.

The implementation of pyMelt in Python enables easy integration with other Python modules, models for running Bayesian inversion calculations for example, as done by Matthews et al. [2021]. Since numerical inversion techniques rely on running many forward models, the forward model should be fast. Running an adiabatic decompression melting calculation for a T_p of 1350 °C and a pure lherzolite mantle takes 455 ms in a Jupyterlab notebook running on a 2019 MacBook Pro with 2.6 GHz processor and 16 GB of RAM. A similar calculation, but using a hydrated lherzolite (0.1 wt.% H_2O modelled as continuous melting) takes much longer (59.6 s on the same machine), due to the need to solve for F numerically at each step. For efficient inversion using a hydrated lithology, further development will be required to increase the computational efficiency. Calculating the trace element composition of the magmas using the default settings takes 7.1 s.

Future development of the library could include methods for calculating melting behaviour at subduction zones, integration with geodynamical models, and other more complex melting scenarios. Work currently underway will provide an expanded set of tools for modelling the behaviour of trace elements during melting of lithologically heterogeneous mantle, and potentially the major element composition of lavas. We welcome contributions to pyMelt from the community and guidelines are available on the GitHub repository.

Author Contributions

SM conceived the idea of a modular mantle melting engine built in Python, and designed the overarching structure of the library. SM and KW wrote the code for the melting models, developed the documentation, and drafted the manuscript. KW led the development of the INVMEL chemistry code and many of the features in the chemistry module. MG developed the hydrous melting code, which was generalised by SM. All authors contributed to editing the manuscript.

Acknowledgements

This work was supported by Natural Environment Research Council studentship [NE/L002574/1] awarded to KW. We thank Patrick Ball and Christy Till for their constructive reviews which significantly improved the clarity of the manuscript.

Data Availability

The latest version of pyMelt can be obtained from github.com/simonwmatthews/pyMelt and the pip package manager. pyMelt version v2.00 is archived in a Zenodo repository with DOI 10.5281/zenodo.7101330.

Copyright notice

References

Asimow, P. D., Langmuir, and CH (2003). "The importance of water to oceanic mantle melting regimes". *Nature* 421(6925), pages 815–820. DOI: 10.1038/nature01429.

Ball, P., T. Duvernay, and D. Davies (2022). "A Coupled Geochemical-Geodynamic Approach for Predicting Mantle Melting in Space and Time". *Geochemistry, Geophysics, Geosystems* 23(4), e2022GC010421. DOI: 10.1029/2022GC010421.

Ball, P., N. White, J. Maclennan, and S. Stephenson (2021). "Global influence of mantle temperature and plate thickness on intraplate volcanism". *Nature Communications* 12(1), pages 1–13. DOI: 10.1038/s41467-021-22323-9.

Brown, E. L. and C. E. Lesher (2014). "North Atlantic magmatism controlled by temperature, mantle composition and buoyancy". *Nature Geoscience* 7(11), pages 820–824. DOI: 10.1038/ngeo2264.

– (2016). "REEBOX PRO: A forward model simulating melting of thermally and lithologically variable upwelling mantle". *Geochemistry, Geophysics, Geosystems* 17(10), pages 3929–3968. DOI: 10.1002/2016GC006579.

Gibson, S. and D. Geist (2010). "Geochemical and geophysical estimates of lithospheric thickness variation beneath Galápagos". *Earth and Planetary Science Letters* 300(3-4), pages 275–286. DOI: 10.1016/j.epsl.2010.10.002.

Gleeson, M., C. Soderman, S. Matthews, S. Cottaar, and S. Gibson (2021). "Geochemical Constraints on the Structure of the Earth's Deep Mantle and the Origin of the LLSVPs". *Geochemistry, Geophysics, Geosystems* 22(9), e2021GC009932. DOI: 10.1029/2021GC009932.

Harðardóttir, S., S. Matthews, S. A. Halldórsson, and M. G. Jackson (2022). "Spatial distribution and geochemical characterization of Icelandic mantle end-members: Implications for plume geometry and melting processes". *Chemical Ge-*

ology, page 120930. DOI: 10.1016/j.chemgeo.2022.120930.

Ito, G. and J. J. Mahoney (2005). "Flow and melting of a heterogeneous mantle: 1. Method and importance to the geochemistry of ocean island and mid-ocean ridge basalts". *Earth and Planetary Science Letters* 230(1-2), pages 29–46. DOI: 10.1016/j.epsl.2004.10.035.

Jennings, E. S., T. J. Holland, O. Shorttle, J. Maclennan, and S. A. Gibson (2016). "The composition of melts from a heterogeneous mantle and the origin of ferropicrite: application of a thermodynamic model". *Journal of Petrology* 57(11-12), pages 2289–2310. DOI: 10.1093/petrology/egw065.

Katz, R. F., M. Spiegelman, and C. H. Langmuir (2003). "A new parameterization of hydrous mantle melting". *Geochemistry, Geophysics, Geosystems* 4(9). DOI: 10.1029/2002GC000433.

Krein, S. B., M. D. Behn, and T. L. Grove (2020). "Origins of major element, trace element, and isotope garnet signatures in mid-ocean ridge basalts". *Journal of Geophysical Research: Solid Earth* 125(12), e2020JB019612. DOI: 10.1029/2020JB019612.

Lambart, S., M. B. Baker, and E. M. Stolper (2016). "The role of pyroxenite in basalt genesis: Melt-PX, a melting parameterization for mantle pyroxenites between 0.9 and 5 GPa". *Journal of Geophysical Research: Solid Earth* 121(8), pages 5708–5735. DOI: 10.1002/2015JB012762.

Maclennan, J., D. McKenzie, and K. Gronvöld (2001). "Plume-driven upwelling under central Iceland". *Earth and Planetary Science Letters* 194(1-2), pages 67–82. DOI: 10.1016/S0012-821X(01)00553-2.

Matthews, S., O. Shorttle, and J. Maclennan (2016). "The temperature of the Icelandic mantle from olivine-spinel aluminum exchange thermometry". *Geochemistry, Geophysics, Geosystems* 17(11), pages 4725–4752. DOI: 10.1002/2016GC006497.

Matthews, S., K. Wong, O. Shorttle, M. Edmonds, and J. Maclennan (2021). "Do olivine crystallization temperatures faithfully record mantle temperature variability?" *Geochemistry, Geophysics, Geosystems* 22(4), e2020GC009157. DOI: 10.1029/2020GC009157.

McKenzie, D. and M. Bickle (1988). "The volume and composition of melt generated by extension of the lithosphere". *Journal of Petrology* 29(3), pages 625–679. DOI: 10.1093/petrology/29.3.625.

McKenzie, D. and R. K. O'Nions (1991). "Partial melt distributions from inversion of rare earth element concentrations". *Journal of Petrology* 32(5), pages 1021–1091. DOI: 10.1093/petrology/32.5.1021.

Pertermann, M. and M. M. Hirschmann (2003). "Partial melting experiments on a MORB-like pyroxenite between 2 and 3 GPa: Constraints on the presence of pyroxenite in basalt source regions from solidus location and melting rate". *Journal of Geophysical Research: Solid Earth* 108(B2). DOI: 10.1029/2000JB000118.

Phipps Morgan, J. (2001). "Thermodynamics of pressure release melting of a veined plum pudding mantle". *Geochemistry, Geophysics, Geosystems* 2(4). DOI: 10.1029/2000GC000049.

Plank, T. and C. H. Langmuir (1992). "Effects of the melting regime on the composition of the oceanic crust". *Journal of Geophysical Research: Solid Earth* 97(B13), pages 19749–19770. DOI: 10.1029/92JB01769.

Putirka, K. D. (2008). "Thermometers and barometers for volcanic systems". *Reviews in Mineralogy and Geochemistry* 69(1), pages 61–120. DOI: 10.2138/rmg.2008.69.3.

Shorttle, O., J. Maclennan, and S. Lambart (2014). "Quantifying lithological variability in the mantle". *Earth and Planetary Science Letters* 395, pages 24–40. DOI: 10.1016/j.epsl.2014.03.040.

Smith, P. M. and P. D. Asimow (2005). "Adiabat_1ph: A new public front-end to the MELTS, pMELTS, and pHMELTS models". *Geochemistry, Geophysics, Geosystems* 6(2). DOI: 10.1029/2004GC000816.

Turcotte, D. L. and G. Schubert (2002). *Geodynamics*. Cambridge University Press.

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.

White, R. S., D. McKenzie, and R. K. O'Nions (1992). "Oceanic crustal thickness from seismic measurements and rare earth element inversions". *Journal of Geophysical Research: Solid Earth* 97(B13), pages 19683–19715. DOI: 10.1029/92JB01749.