PyFLOWGO: An open-source platform for simulation of channelized lava thermo-rheological properties
Magdalena Oryaëlle Chevrel, Jérémie Labroquère, Andrew Harris, Scott Rowland

To cite this version:

HAL Id: hal-01634842
https://hal-clermont-univ.archives-ouvertes.fr/hal-01634842
Submitted on 16 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
PyFLOWGO: an open-source platform for simulation of channelized lava thermo-rheological properties

Magdalena Oryaëlle Chevrel\textsuperscript{a}, Jérémie Labroquère\textsuperscript{b}, Andrew Harris\textsuperscript{a}, Scott Rowland\textsuperscript{c}

\textsuperscript{a}Université Clermont Auvergne, CNRS, IRD, OPGC, Laboratoire Magmas et Volcans, F-63000 Clermont-Ferrand, France
\textsuperscript{b}Private work. Formally working at THALES SERVICES SAS, 290 Allée du Lac, 31670 Labège, France
\textsuperscript{c}Department of Geology and Geophysics, University of Hawai‘i at Mānoa, Honolulu, Hawaii, USA

Abstract

Lava flow advance can be modeled through tracking the evolution of the thermo-rheological properties of a control volume of lava as it cools and crystallizes. An example of such a model was conceived by [Harris and Rowland (2001)] who developed a 1-D model, FLOWGO, in which the velocity of a control volume flowing down a channel depends on rheological properties computed following the thermal path estimated via a heat balance box model. We provide here an updated version of FLOWGO written in Python that is an open-source, modern and flexible language. Our software, named PyFLOWGO, allows selection of heat fluxes and rheological models of the user’s choice to simulate the thermo-rheological evolution of the lava control volume. We describe its architecture that offers more flexibility while reducing the risk of making error when changing models in comparison to the previous FLOWGO version. Three cases are tested using actual data from channel-fed lava flow systems and results are discussed in terms of model validation and convergence. PyFLOWGO is open-source and packaged in a Python library to be imported and reused in any Python program.

Keywords: FLOWGO, Python, lava flow, heat budget, rheology

1. Introduction

The thermo-rheological properties of lava flowing in a channel depend on the evolution of the moving volume, where viscosity and yield strength are increasing due to cooling and crystallization (e.g. Lipman and Banks\textsuperscript{[1987]} Crisp et al.\textsuperscript{[2002]})
Cashman et al. (1999). Harris and Rowland (2001) produced a 1-D model called FLOWGO in which velocity of a lava control volume flowing down a channel is computed via the Jeffreys (1925) equation as modified for a Bingham fluid by Moore (1987). In this approach velocity depends on the lava rheological properties computed according to the cooling and crystallization path of the control volume as estimated via a heat balance box model (Fig. 1). FLOWGO is thus a framework within which thermo-rheological models can be integrated to test fits between output parameters and natural data. By selecting appropriate models to place within this framework Harris and Rowland (2001) succeeded in simulating the down flow heat budget, cooling, crystallinity, viscosity, yield strength, velocity, channel width and maximum length of several lava flows including those of Mauna Loa 1984, Pu'u ʻŌʻō 1997 and Etna 1998. Rowland et al. (2004) later adapted the models contained within FLOWGO to run in a Martian environment allowing cooling-limited, channelized lava flows on Mars to be simulated and their emplacement properties to be inferred (Ramsey et al. 2016).

During the 14 years since the inception of this approach, the basic physical principles on which FLOWGO is based have not changed. But recently, Harris and Rowland (2015) and Harris et al. (2015) incorporated an alternative model to compute the melt phase viscosity that is based on lava composition, rather than on a given assumed viscosity as originally proposed. They also introduced a three phase rheological model to estimate the effect of crystals and bubbles on viscosity. To correctly simulate the evolution of thermo-rheological parameters down flow using FLOWGO the user thus is allowed a degree of flexibility so as to best-fit the natural cases, while changing thermo-rheological models and variables within plausible limits (e.g. Harris et al. 2007). Originally, Harris and Rowland (2001)
wrote FLOWGO in the programming language IDL (Interactive Data Language) but due to license price and other computing issues this code was set aside and an Excel version was written (officially published in Harris et al. 2015). This was freely shared when needed by other scientists. Although Excel is a convenient tool and is easily and widely used by geologists, it has limited applications, a poor flexibility for model evolution, and when many equations and input parameters are stacked in sequence, it becomes too easy to key in a hidden (or very-hard to find) error. Besides, it cannot be easily incorporated into other software. Lava modeling capabilities and computer processing power has improved over the past decade, and FLOWGO remains often cited, being recognized as the only thermo-rheological-based model. Some authors have therefore used it to reproduce natural flow evolution of past (e.g. Riker et al. 2009; Wantim et al. 2013) or ongoing eruptions (Harris et al. 2011; Wright et al. 2008), as well as producing hazards maps (Rowland et al. 2005) and applying FLOWGO as a reference to compare results of other models against (e.g. Cordonnier et al. 2015) or as input to develop new probabilistic models (e.g. QLA V AH, Mossoux et al. 2016).

The focus of the present work is thus to provide FLOWGO in a modern and flexible language. We chose Python because it provides useful libraries, is open-source, and its object-oriented approach allows for great flexibility. Python also has been widely adopted in scientific computing during the recent years and has been described as “the next wave in Earth Sciences Computing because it simply enables users to do more and better science” (Lin 2011). Furthermore, Python can be run on any operating system which guarantees portability. Here, we describe the architecture of our new open-source code, named PyFLOWGO, explaining the various models (heat flux, rheology, crystallization rate, crust temperature, crust
cover fraction) that can currently be chosen to set-up a lava flow simulation. So that the model can be trusted as an operational tool with known uncertainty we tested the output against previous iterations of the model. As validation, we followed three cases for which appropriate natural data are available and have been previously tested: Mauna Loa 1984 (Hawaii), Mauna Ulu 1974 (Kilauea, Hawaii) and Piton de la Fournaise 2010 (La Réunion), and results are discussed in terms of model convergence and error.

2. Model architecture in Python

PyFLOWGO is developed in Python v3 which is an object-oriented programming language. The code has been designed to allow the user to switch between any existing models and add new models as they become available, without modifying the architecture of the code. The software acts as a framework that provides interfaces to implement multiple models, and calls them in the correct sequence to build the lava flow differential equations and solve them using a numerical approach. The interfaces basically define the methods necessary for the solver to work and can be implemented with specific models depending on the desired simulation. The top level of the architecture is the integrator which solves the differential equations depending on heat fluxes and on input physical characteristics of the lava (described by the material lava class), terrain conditions and a given crystallization rate model (Fig. 2). The integrator solves the differential equations and updates accordingly the current lava state (temperature, crystallization, position, etc.) which is then used for the next integration step. This process is iterated until termination conditions are reached. The material lava class is composed of multiple models such as the melt viscosity model, the relative viscosity model, the yield
strength model and the vesicle fraction model. Each model is defined by the same interface that governs inputs and outputs delivered to and from the model. As an example, to compute the melt viscosity, all the models available to the user share a common interface called base melt model viscosity (Fig. 3). This interface makes sure that the model receives the state of the lava in order to deliver the viscosity value in the expected unit, that is Pa s. In the same way, all heat fluxes that compose the differential equation share the same interface called base flux (Fig. 4). In this case, the interface provides a unique method to compute and return the flux in W/m based on the state and channel dimensions as input parameters. With this architecture, new physical models or fluxes can then easily be added by implementing the given interface it depends from. Communication is carried out only between the interfaces, and models can be switched from one to another with no modification of the code structure, thus avoiding implementation errors and allowing a great flexibility.

3. Modeling

3.1. Differential equation for heat budget and crystallization down flow

PyFLOWGO is built around the main differential equation established in FLOWGO which is based on the heat budget for a control volume of lava within a channel (e.g. Danes 1972, Park and Iversen 1984, Crisp and Baloga 1994, Keszthelyi 1995b, Keszthelyi and Self 1998). The thermal budget ($\Delta H$) represents the balance of the heat fluxes flowing in (gain) and out (loss) of a box model as illustrated in Figure 1. The change in heat content of a unit length per unit time ($\Delta H$ in J s$^{-1}$m$^{-1}$) is
therefore described by:

$$\Delta H = Q_{\text{rad}} + Q_{\text{conv}} + Q_{\text{rain}} + Q_{\text{cond}} - Q_{\text{cryst}} - Q_{\text{visc}}$$  \hspace{1cm} (1)$$

where $Q_{\text{rad}}$, $Q_{\text{conv}}$ and $Q_{\text{rain}}$ represent the heat loss from the surface due to radiation, forced atmospheric convection and rain vaporisation; $Q_{\text{cond}}$ is the heat loss by conduction through the flow base and levées; and $Q_{\text{cryst}}$ and $Q_{\text{visc}}$ are the heat gains due to crystallization and viscous dissipation. Note that heat loss due to entrainment of cold material from the crust into the hotter flow interior could also be involved (Crisp and Baloga 1994). This case is not treated here but all details may be found in Harris and Rowland (2001). Following Keszthelyi (1995b), heat gain from crystallization, $Q_{\text{cryst}}$ can be written as:

$$Q_{\text{cryst}} = \frac{\partial T}{\partial x} \rho_{\text{bulk}} L_{\text{cry}} E_r \frac{\partial \phi}{\partial T_{\text{cool}}}$$ \hspace{1cm} (2)$$

where $\partial T/\partial x$ is the cooling per unit length (in K/m), $\rho_{\text{bulk}}$ (kg/m$^3$) is lava bulk density, $L_{\text{cry}}$ (J/kg) is latent heat of crystallization, $E_r$ (m$^3$/s) is effusion rate, and $\partial \phi/\partial T_{\text{cool}}$ (K$^{-1}$) is the increase of crystal volume fraction per degree of cooling (crystallization down flow, $-\partial \phi/\partial T$). Combining and re-arranging Eq. 1 and Eq. 2, the cooling per unit length (as function of distance down flow) can be isolated and the following differential equation established (Harris and Rowland 2001; Harris and Rowland 2015; Harris et al. 2015):

$$\frac{\partial T}{\partial x} = -\frac{Q_{\text{rad}} + Q_{\text{conv}} + Q_{\text{cond}} - Q_{\text{rain}} + Q_{\text{visc}}}{E_r \rho_{\text{bulk}} L_{\text{cry}} \partial \phi/\partial T_{\text{cool}}}$$ \hspace{1cm} (3)$$
where the fluxes are expressed in W/m and detailed in the supplementary material (Appendix A) together with the thermal conditions of the lava (crust temperature and coverage). Each heat flux ($Q_i$) can be written as a function of the state ($Y$) and the position ($x$), such that: $Q_i(Y,x)$. This allows rewriting Eq.3 in the following form:

$$\frac{\partial T}{\partial x} = F \left(Y, x, \frac{\partial \phi}{\partial T_{cool}}\right)$$

with:

$$F \left(Y, x, \frac{\partial \phi}{\partial T_{cool}}\right) = \frac{1}{E_r \rho_{bulk} L_{cryst}} \frac{1}{\partial \phi / \partial T_{cool}} \sum_{fluxes} Q_i(Y,x)$$

The increase of down flow crystal fraction ($\partial \phi / \partial x$) is then computed via:

$$\frac{\partial \phi}{\partial x} = \frac{\partial \phi}{\partial T_{cool}} \frac{\partial T}{\partial x}$$

where the crystallization rate per degree of cooling, $\partial \phi / \partial T_{cool}$, can be calculated from one of the models presented in the supplementary material (Appendix A).

3.2. Conservation of volume

Based on mass conservation, if the effusion rate and channel depth are kept fixed down flow, then width can be computed at each down flow step from:

$$w = \frac{E_r}{d V_{mean}}$$

with $w$ and $d$ being the channel width and depth (in m), and $V_{mean}$ being the flow velocity (in m/s) which in turn is controlled by the underlying slope and the rheological properties of the lava. The effusion rate is computed at the vent from initial
flow geometry (width x depth), rheology and velocity.

3.3. Flow velocity

The velocity model originally used by FLOWGO is derived from the Jeffreys (1925) equation that was later adapted for a Bingham rheology by Moore (1987). Although this equation is based on the mean of the velocity gradient inside the channel, here it is used as to estimate a single value to best characterize the velocity of the control volume at each step. Note that this is not the rate of advance of the flow front and is expressed as:

\[ V_{\text{mean}} = \frac{\rho_{\text{bulk}} g d^2 \sin \theta}{n \eta_{\text{bulk}}} \left[ 1 - \frac{3}{2} \frac{\tau_0}{\tau_b} + \frac{1}{2} \left( \frac{\tau_0}{\tau_b} \right)^3 \right] \]  

(7)

where \( n \) is the channel shape factor, \( \theta \) is the underlying slope in radians, \( g \) (m/s²) is acceleration due to gravity, \( \rho_{\text{bulk}} \) (kg/m³) is the lava bulk density, \( \eta_{\text{bulk}} \) (Pa·s) is the bulk viscosity of the lava mixture (considering the melt phase and the effect of particles, see supplementary for details), and \( \tau_0 \) (Pa) and \( \tau_b \) (Pa) are respectively the lava yield strength and the basal shear stress. Channel shape factor can be obtained via \( 3(1 + d/w)^2 \) (Wilson and Parfitt 1993) and reduces to 3 when the channel is much wider than deeper. Solution of the velocity model now requires definition of \( \rho_{\text{bulk}}, \eta_{\text{bulk}}, \tau_0 \) and \( \tau_b \) where \( \eta_{\text{bulk}}, \tau_0 \) are functions of \( \partial \phi / \partial T_{\text{cool}} \) and \( \partial T / \partial x \) and hence dependent on the thermal box model. Details about the models to calculate these variables are given in Appendix A.
4. Equation solving and numerical methods

4.1. Numerical integration

PyFLOWGO uses the FLOWGO solution (Harris and Rowland 2001; Rowland et al. 2005; Harris and Rowland 2015) to solve the lava flow temperature and crystal fraction equations, i.e., Eq. 4 and Eq. 5, respectively. These two equations are discretized in the space domain using a linearization at the position $x_i$. By denoting $\delta L^2 = L_2 - L_1$ as the variation in variable $L$ from state 1 to state 2, and $L_i = L(x_i)$ as the variable value at position $x_i$, we obtain the following first order integration (neglecting higher order components):

$$\frac{\delta L^{i+1}}{\delta L^i} \approx F \left[ Y_i, x_i; \left( \frac{\partial \phi}{\partial T_{cool}} \right)_i \right]$$

(8)

which leads to the common Euler scheme for temperature:

$$T_{i+1} = T_i + (\delta L^i x) F \left[ Y_i, x_i; \left( \frac{\partial \phi}{\partial T_{cool}} \right)_i \right]$$

(9)

with $\delta L^i x$ being the step distance between $i$ and $i+1$. Then the second equation, for the crystal fraction, is solved using the same approach:

$$\phi_{i+1} = \phi_i + (\delta L^i x) \left( \frac{\partial \phi}{\partial T_{cool}} \right)_i F \left[ Y_i, x_i; \left( \frac{\partial \phi}{\partial T_{cool}} \right)_i \right]$$

(10)

The values of the state ($Y_{i+1}$) now allow computation of the rheology, and hence the velocity of the control volume, at position $x_{i+1}$, which is then used to estimate channel width by considering a constant effusion rate $E_r$ and a constant depth in the conservation of volume equation (Eq. 6).
The thermal conditions and the crystal fraction are thus integrated by propagating the initial state conditions at $x_0$ down-channel, and by updating velocity and control volume thermo-rheological properties at each step. In practice the step size is fixed such that $\forall i, \delta x_{i+1} = \Delta x$. Determination of this step is model and case dependent, and must be chosen to be small enough to remove any numerical error and provide enough accuracy in the variables of interest. A convergence study must therefore be performed for every new studied case.

4.2. Data interpolation

At multiple places in PyFLOWGO, interpolation is needed when data are provided as discrete values. In particular, the line-of-steepest-descent down which the control volume is moved is extracted from a Digital Elevation Model (DEM) where data are usually provided for every 1, 10, 50 or even every 100 meters (depending on the spatial resolution of the DEM). The same holds for the MELTS data used to estimate crystallization as a function of temperature (see crystallization rate model in supplementary material). In this case, data are usually given with a temperature step of 0.1, 1 or 10 °C. To be independent of data discretization, and thus able to read any type of data, the data has to be interpolated to be set at a common step value. In PyFLOWGO, we use a linear interpolation to reconstruct any missing data during the integration of the differential equations.

5. Description of the software package

PyFLOWGO is packaged as a module (or library) to be imported and reused in any Python program. The models and simulation configurations are chosen using a single configuration file (json format) that contains all the necessary numerical and
model parameters (Table A.1 in appendix). The user can therefore select which 
heat fluxes to consider and the associated models for lava thermal condition down 
flow including effective crust cover fraction, crust and uncrusted surface temper-
ature (see Table A.2 in appendix). The package also include models for crystal-
lization per degree of cooling, density, melt viscosity, relative viscosity, vesicle 
fraction and for yield strength and basal shear stress (see Appendix A for details).
As described above (section 2), any new model can be added by implementing the 
interface it depends from.

The line-of-steepest-descent down which the control volume is moved has to 
be previously extracted from a DEM and input as a x, y text file where x is distance 
down flow and y is the slope (in degrees) at that point. All computed variables (as 
a function of distance) can be stored for every step in an output file (CSV format).
Some tools are provided to plot graphs such as crystallization rate, velocity, evo-
lution of the bulk viscosity (interstitial melt + effect of particles), crystal fraction, 
yield strength, channel width as a function of the distance. Ground-truth data can 
also be added and plotted with the model results.

6. Model Verification

Three lava channels, Mauna Loa 1984 (ML84), Mauna Ulu 1974 (MU74) and 
Piton de la Fournaise 2010 (PdF2010) have been chosen as test cases from previ-
ously published FLOWGO papers. These flows were chosen because their input 
parameters are well constrained and because they cover a large range of lava flow 
characteristics: ML84 being a long channel of more than 25 km, MU74 being a 
mature channelized flow of about 8 km in length and PdF2010 being slower and 
cooler than the Hawaiian cases and of only 1-2 km in length. For these cases,
PyFLOWGO was run using the same input parameters as provided in Harris and Rowland (2015), Robert et al. (2014) and Harris et al. (2015), respectively for the three test cases (see table A.3 in appendix).

6.1. Convergence analysis

Numerical convergence is a necessary step to verify of the numerical solution. Indeed, the integration scheme propagates a numerical error which can then exceed the model error. The numerical error of an Euler integration scheme is controlled by the integration step size. Convergence analysis consists of reducing the step size and then tracking the value returned for the variable of interest. When the variable seems to stabilize within an acceptable range, then the step size is sufficiently small.

A convergence analysis is performed here for each case by reducing the iteration step size from 100 m down to 1 m. Figure 5 shows the maximum distance attained by the control volume (i.e. the point where mean velocity equals 0 m/s) and the lava core temperature near the end of the flow with respect to the step size. For the three cases the convergence for distance is reached at step size less than 10 m with an error of less than 50 m. For ML84 and MU74, the temperature convergence is reached at step sizes of less than 10 m, with an error within $\sim 2-3^\circ$C, but for PdF2010, temperature convergence seems to be reached only at smaller step sizes (Fig. 5). For these three test cases, we conclude that a 10 m step is a maximum step size that must be used to run PyFLOWGO in order to guarantee small numerical errors on simulated lava flow properties and dimensions. Initial convergence tests by Harris and Rowland (unpublished) and by us here indicate 10 m as being an optimum step size in terms of errors and run time. As mention in
the previous section a convergence study must be performed for every new studied case.

6.2. Results and validation against FLOWGO

Figure [6] plots lava core temperature, bulk viscosity and mean velocity computed with PyFLOWGO against the results from FLOWGO Excel spreadsheets prepared for the same test cases by Harris and Rowland (2015), Robert et al. (2014) and Harris et al. (2015), using identical input parameters (see table A.3 in appendix). For all the cases, PyFLOWGO reproduces the FLOWGO results perfectly. Note that the oscillations in mean velocity for ML84 and Pd2010 come from the small spatial resolution of the line of steepest descent. To obtain less noisy results, one could filter the DEM data. For Mauna Ulu 1974, we used the same slope path as the one provided by Robert et al. (2014), specifically one value every 200 m. The PyFLOWGO run at this 200 m step size thus reproduces very well the results of Robert et al. (2014), but a comparison with PyFLOWGO run at a step size of 10 m shows that convergence was not actually reached using the 200 m step size. Here one can see that using an appropriate step size is necessary to avoid large errors (in this case the distance reached differs by 1 km on a total distance of 6.5 km).

7. Conclusion

This paper describes PyFLOWGO, a software written in Python to run FLOWGO, a thermo-rheological framework for lava flowing in a channel as originally presented by Harris and Rowland (2001). PyFLOWGO is constructed in a similar
manner as FLOWGO to allow estimation of all parameters involved in the thermo-

rheological evolution of a control lava volume flowing down a channel. We present

here the architecture of the code, as well as the discretized formulation of the chan-
nelized lava flow problem and the various models that can be selected according
to the study case. This new code is written with the object-oriented programming
language Python v3, and offers more flexibility while reducing the risk of mak-
ing error when changing models in comparison to the previous FLOWGO version
which was written in Excel. The user can run PyFLOWGO using already imple-
mented models, or extend the code with new models by simply implementing the
base classes. Communication through interfaces allow tests of different models on
the same study case, without modifying the code architecture. This software, used
as a model testing platform, also allows the user to easily and quickly set up new
complex cases of lava flow simulation to test.

PyFLOWGO has been successfully validated against FLOWGO via three test
cases (Mauna Loa 1984, Mauna Ulu 1974 and Piton de la Fournaise 2010). For
each test case, a convergence study has been performed, which is an essential prac-
tice that must be conducted for each new study. PyFLOWGO has also been unit
tested and packaged in a Python library form to allow ease of installation. This
software is open-source, thus available at any location and institution.

Acknowledgements

The authors greatly acknowledge Simone Tarquini and an anonymous reviewer
for revisions that improve the clarity of the manuscript. M.O.C acknowledges the
Auvergne fellowship program for full support. This is ClerVolc publication number
270.
References


16


Harris, A., Favalli, M., Wright, R., Garbeil, H., 2011. Hazard assessment at Mount


Figure 1: Schematic view of the thermo-rheological model FLOWGO illustrating the heat box model of the control volume of lava advancing through a channel (modified from Harris and Rowland, 2001). The lava viscosity and yield strength are estimated within the control volume according to the lava state within the box (including thermal state: heat budget, temperature of core, base, surface, crust; and physical state: crustal coverage, crystallinity, vesicularity) in order to compute the velocity and corresponding channel width (for a fixed effusion rate) used for the next step. This model assumes a "cooling limited" lava flow behavior: the lava stops flowing because it has cooled to such an extent that its rheological behavior impede motion. $Q_{\text{rad}}$, $Q_{\text{conv}}$ and $Q_{\text{rain}}$ are heat losses into the atmosphere due to radiation, forced convection due to heating of the air above the lava surface and effect of rain, respectively. $Q_{\text{cond}}$ is the heat loss by conduction into the cooler base and levees. $Q_{\text{cryst}}$ and $Q_{\text{visc}}$ are the heat gain due to crystallization and viscous dissipation, respectively. Dimensions $d$ and $w$ are the channel depth and width, and $\theta$ is the underlying ground slope.
Figure 2: PyFLOWGO UML class diagram - top level. The interfaces (labeled "I" and with the prefix "base") provide parameters to main classes (labeled "C") that enable the Integrator to update the flow State at discrete positions along a slope.
Figure 3: Example of PyFLOWGO UML class diagram describing the interface, base melt viscosity model, for the various models to compute the melt viscosity (see Appendix A for details about the models). The user is free to chose the model of his choice (see Table A.2 in appendix for the available models at this date) or implement a new model.

Figure 4: Example of PyFLOWGO UML class diagram describing the interface, base flux, for the various heat fluxes (see Appendix A for details about the fluxes). The user is free to chose which fluxes to consider (see Table A.2 in appendix for the available fluxes at this date) or implement a new flux.
Figure 5: Convergence of the PyFLOWGO runs at 1, 5, 10, 50, and 100 m or 200m step size. The maximum channel length (top) represent the distance reached at $v_{\text{mean}} = 0$ m/s. The convergence of the temperature (bottom) is shown for distances of 29.2, 6.6 and 3.7 km for the three lava flows, respectively. Note that the ML84 results here are for the "cold" lava as presented in Harris and Rowland (2015) and the PdF2010 is with the LiDAR-derived slope as presented in Harris et al. (2015).
Figure 6: Validation of PyFLOWGO (blue line) against the Excel version of FLOWGO (dashed red line). Here are shown examples of output data (velocity, core temperature and channel width) obtained for Mauna Loa 1984 (a, b, c) according to “cold” regime as given by Harris and Rowland (2015); for Mauna Ulu 1974 (d, e, f) according to a corrected data set from the published version of Robert et al. (2014); and for Piton de la Founaise 2010 (g, h, i) using the SRTM acquisition slope path from Harris et al. (2015). The input parameters are given in Table A.3 in appendix. Field data are also plotted for comparison. Note also that the line-of-steepest-descent for Mauna Ulu 1974 and for Piton de la Founaise 2001 is given only until 6000 m and 1000 m, respectively; the last section of the slope is therefore equal to the last slope value.
Appendix A. Models

Appendix A.1. Crystallization rate models

Crystallization rate per degree of cooling down flow \((\partial \phi / \partial T_{cool})\) can be calculated via different models. Here we provide four models that the user is free to chose to run in PyFLOWGO.

Appendix A.1.1. Basic model

The basic model proposed by Harris and Rowland (2001) takes into account the amount of crystallization during flow \((\phi_{grown})\). This is defined as the amount of crystallization occurring between the eruption temperature \((T_{erupt})\) and the temperature at which the lava cannot flow anymore \((T_{solid})\):

\[
\frac{\partial \phi}{\partial T_{cool}} = \frac{\phi_{grown}}{T_{erupt} - T_{solid}}
\] (A.1)

Appendix A.1.2. Bimodal model as function of distance

This bimodal model was proposed by Robert et al. (2014) and Harris and Rowland (2015) and allows the crystallization rate to be changed after a given distance \((x_{critic})\).

If \(x \leq x_{critic}\): \((\partial \phi / \partial T_{cool}) = C_1\)

If \(x > x_{critic}\): \((\partial \phi / \partial T_{cool}) = C_2\) \hfill (A.2)

where \(x_{critic}\) and the constants \(C_1\) and \(C_2\) are of the user’s choice.
Appendix A.1.3. Bimodal model as function of temperature

This bimodal model allows the crystallization rate to be changed after a given temperature \( T_{\text{critic}} \) is reached.

\[
\text{If } T_{\text{core}} \geq T_{\text{critic}}: \quad \frac{\partial \phi}{\partial T_{\text{cool}}} = C_1
\]

\[
\text{If } T_{\text{core}} < T_{\text{critic}}: \quad \frac{\partial \phi}{\partial T_{\text{cool}}} = C_2
\]

where \( T_{\text{critic}} \), and the constants \( C_1 \) and \( C_2 \) are of the user’s choice.

Appendix A.1.4. MELTS model

The MELTS model allows the crystallization rate per degree of cooling to be set from a MELTS-based look-up table as suggested by Harris and Rowland (2001), Harris and Rowland (2015) and Riker et al. (2009). The look-up table is a file containing the amount of crystals (fraction) as a function of temperature (in °C) that must be previously built using the MELTS software of Ghiorso and Sack (1995). A linear interpolation of these data is computed by PyFLOWGO and gives a function \( \phi_{\text{interp}} \) that represents the fraction of crystals grown as a function of temperature. The fraction of crystals grown per degree of cooling is then computed using the finite differences via the interpolated function:

\[
\frac{\partial \phi}{\partial T_{\text{cool}}} \approx -\frac{\phi_{\text{interp}}(T_{\text{core}} + \Delta T) - \phi_{\text{interp}}(T_{\text{core}} - \Delta T)}{2\Delta T}
\] (A.4)

with \( \Delta T \) being the temperature step that is chosen to be small enough (for example \( 10^{-6} \)). Note that this model considers that crystallization happens under equilibrium conditions, which is probably not the case during lava emplacement (e.g., Chevrel et al. 2013; Kolzenburg et al. 2016).
Appendix A.2. Heat flux models

Appendix A.2.1. Radiative heat flux

Heat loss due to radiation from the lava surface to the atmosphere is expressed as:

\[ Q_{\text{rad}} = \sigma \varepsilon T_{\text{eff}}^4 \times w \quad (A.5) \]

where \( \sigma \) (W/m\(^2\) K\(^4\)) is the Stefan–Boltzmann constant, \( \varepsilon \) is emissivity, \( w \) is the channel width and \( T_{\text{eff}} \) (K) is the effective surface temperature, which is calculated using a two-component model for the lava surface [Pieri and Baloga 1986, Crisp and Baloga 1990, Pieri et al. 1990]:

\[ T_{\text{eff}} = \left[ f_{\text{crust}}(T_{\text{crust}}^4 - T_{\text{atmo}}^4) + (1 - f_{\text{crust}})(T_{\text{hot}}^4 - T_{\text{atmo}}^4) \right]^{0.25} \quad (A.6) \]

where \( T_{\text{atmo}} \) is the temperature of the surrounding atmosphere, \( f_{\text{crust}} \) is the fraction of crusted lava, \( T_{\text{crust}} \) is the cool crust temperature, \( 1 - f_{\text{crust}} \) represents the fraction of exposed uncrusted hot lava and \( T_{\text{hot}} \) is the hot component temperature. The different models used to calculate \( f_{\text{crust}}, T_{\text{crust}} \) and \( T_{\text{hot}} \) are described in sections Appendix A.7, Appendix A.8 and Appendix A.9 respectively.

Appendix A.2.2. Forced convection heat flux

Heat loss due to forced atmospheric convection from the lava surface is calculated via (e.g. Keszthelyi et al. 2003):

\[ Q_{\text{conv}} = h_{\text{conv}}(T_{\text{conv}} - T_{\text{atmo}}) \times w \quad (A.7) \]
where $h_{conv}$ is the convective heat transfer (in W/m$^2$ K) and $T_{conv}$ (K) the characteristic surface temperature. The convective heat transfer depends on atmospheric conditions and can be defined as:

$$h_{conv} = U C_H \rho_{atmo} C_{p_{atmo}}$$  \hspace{1cm} (A.8)

where $U$ is wind speed (m/s), $C_H$ the wind friction factor as defined by Greeley and Iversen (1987), $\rho_{atmo}$ (kg/m$^3$) is atmospheric density and $C_{p_{atmo}}$ the heat capacity of the air (J/kg K) in contact with the lava surface. The characteristic surface temperature is calculated via:

$$T_{conv} = \left[ f_{crust} T_{crust}^{1.33} + (1 - f_{crust}) T_{hot}^{1.33} \right]^{0.75}$$  \hspace{1cm} (A.9)

Appendix A.2.3. Heat flux due to rain

The heat flux due to vaporization of rainwater falling onto the lava surface is expressed by:

$$Q_{rain} = \frac{\partial R}{\partial t} \rho_{H2O} L_{H2O} w$$  \hspace{1cm} (A.10)

where $\partial R/\partial t$ (m/s) is the rainfall rate and $\rho_{H2O}$ (kg/m$^3$) and $L_{H2O}$ (J/kg) are, respectively, the density and latent heat of vaporisation of water.

Appendix A.2.4. Conductive heat flux

The heat flux through the base and the levées of the flow occur via conduction and is expressed as (after Keszthelyi 1995a):

$$Q_{cond} = \kappa_{lava} \frac{T_{core} - T_{base}}{h_{base}} w$$  \hspace{1cm} (A.11)
where $\kappa_{lava}$ is the thermal conductivity of the lava (in W/mK), $T_{core}$ the lava core temperature (in K), $T_{base}$ (K) the temperature at the base of the basal layer and $h_{base}$ (m) the thickness of the basal layer that is defined between the underlaying surface and the thermal boundary when $T_{core}$ is reached. It is usually calculated via:

$$h_{base} = d H_b / 100$$  \hspace{1cm} (A.12)

where $H_b$ is the proportion occupied by the basal layer in respect to the entire flow thickness ($d$, in m).

### Appendix A.2.5. Viscous heating

Viscous heating in the lava channel is expressed here, for a channel that is wider than it is deep ($w > d$) following Costa and Macedonio (2003):

$$Q_{visc} = \eta_{bulk} (V_{mean} / d)^2 w$$  \hspace{1cm} (A.13)

where $\eta_{bulk}$ (Pa·s) is the bulk viscosity of the molten lava as calculated in section Appendix A.4 and $V_{mean}$ is the mean velocity of the lava as calculated in section 3.3.

### Appendix A.3. Density model

PyFLOWGO provides one model to calculate the bulk density:

$$\rho_{bulk} = \phi_b \rho_{DRE}$$  \hspace{1cm} (A.14)

where $\rho_{DRE}$ is the density of the dense rock equivalent and $\phi_b$ is the volume fraction of bubbles in the lava obtained via the methods described in section Appendix A.5.
Appendix A.4. Viscosity models

Magma and lava are complex systems composed of a polydispersed particle mixture of crystals and bubbles of various shapes and sizes in a liquid phase (the silicate melt). The viscosity of this mixture may be defined as:

\[ \eta_{\text{bulk}} = \eta_{\text{melt}} \eta_r \]  

(A.15)

where the viscosity of the interstitial melt, \( \eta_{\text{melt}} \) (Pa s) is Newtonian and depends on temperature and composition, and the relative viscosity, \( \eta_r \) (dimensionless) is obtained by the ratio \( \eta_{\text{bulk}} / \eta_{\text{melt}} \), and depends on the volumetric abundance and aspect ratio of the particles (bubbles and crystals) in the mixture as well as on the strain rate of the flow. PyFLOWGO offers the possibility of calculating the bulk viscosity of the lava using one of four melt viscosity models which can be combined with one of five relative viscosity models.

Appendix A.4.1. Melt viscosity models, \( \eta_{\text{melt}} \)

Dragoni and basic model

The Dragoni model calculates the viscosity of the melt at the lava temperature \( T_{\text{core}} \) using the relation proposed by Dragoni (1989):

\[ \eta_{\text{melt}} = \eta_0 \exp^{0.04(T_0-T_{\text{core}})} \]  

(A.16)

where \( \eta_0 \) (Pa s) is the viscosity of the lava at the liquidus temperature \( T_0 \).

The basic model, as proposed in the original FLOWGO version, is adapted from Dragoni (1989) where instead of liquidus viscosity and temperature, it is the eruption viscosity \( \eta_{\text{erupt}} \) and temperature \( T_{\text{erupt}} \) that are used in Eq. A.16.
**Shaw model**

This model calculates the melt viscosity according to the Arrhenian relationship proposed by Shaw (1972) and reformulated here as:

\[
\log(\eta_{\text{melt}}) = \left[ s \frac{10^4}{T_{\text{core}}} - 1.5s - 6.4 \right]/2.303 - 1 \quad (A.17)
\]

where \(s\) is the characteristic slope of the \(\eta_{\text{melt}}\) versus \(T_{\text{core}}\) relationship that needs to be computed from the melt chemical composition using Shaw (1972).

**VFT model**

This model is based on the Vogel-Fulcher-Tammann equation (Vogel 1921, Fulcher 1925, Tamman and Hesse 1926) and takes into account the non-Arrhenian behavior of the melt viscosity (Dingwell 1996) and allows \(\eta_{\text{melt}}\) to be calculated via:

\[
\log(\eta_{\text{melt}}) = A + \frac{B}{C - T_{\text{core}}} \quad (A.18)
\]

where \(A\) (Pa s), \(B\) (J/mol) and \(C\) (K) are fitting parameters that depend on chemical composition. These fitting parameters need to have been previously determined either from viscosity measurements at high and low temperature or from the melt chemical composition using for example the model proposed by Giordano et al. (2008).

**Appendix A.4.2. Relative viscosity models**

The first four relative viscosity models given here take into account the affect of crystals whereas the fifth model considers those of both crystals and bubbles. More complex formulations may take into account bimodal particle size distribution and
shape (e.g. Castruccio et al. 2010; Cimarelli et al. 2011; Moitra and Gonnermann 2015) or bubble content as a function of their ability to deform (Llewelin and Manga 2005; Pal 2003) but they are not presented here.

Einstein-Roscoe model

This model calculates the effect of crystals on viscosity according to the Einstein-Roscoe relationship, as first introduced by Shaw (1965) and as used in the original FLOWGO version:

$$\eta_r = (1 - R\phi)^{-2.5}$$  \hspace{1cm} (A.19)

where here $R = 1.51$ for spherical solid particles, as suggested by Pinkerton and Stevenson (1992), this equation is therefore only applicable for spherical particles and for a volume fraction maximum of 0.66 (i.e., $1/R$).

Krieger-Dougherty model

This model calculates the effect of crystals on viscosity according to the Krieger-Dougherty relationship (Krieger 1972, Krieger and Dougherty 1959, Pabst 2004):

$$\eta_r = (1 - \phi/\phi_m)^{-b\phi_m}$$  \hspace{1cm} (A.20)

where $b$ is the Einstein coefficient (also termed intrinsic viscosity) and $\phi_m$ is the crystal maximum packing, both being fitting parameters that depend on particle shape. In theory, for spherical particles (aspect ratio of 1) this relationship reduces to Eq. A.19. For elongated particles of aspect ratio of approx. 9, Mueller et al. (2010) give $b = 6.07$ and $\phi_m = 0.343$. See for more examples Mueller et al. 2010, Cimarelli et al. 2011 and Mader et al. 2013.
Maron-Pierce model

This model calculates the effect of crystal cargo on relative viscosity according to Maron and Pierce (1956):

\[ \eta_r = (1 - \phi / \phi_m)^{-2} \] (A.21)

where \( \phi_m \) is a fitting parameter that depends on particle shape (e.g. Mueller et al. 2010 and Mader et al. 2013). For example, Mueller et al. (2010) use \( \phi_m = 0.633 \) for spherical particles and \( \phi_m = 0.339 \) for elongated particles with an aspect ratio of approx. 9.

Costa model

The Costa model allows the effect of crystal fraction in an intermediate range of crystallinity (30 to 80 vol % crystals) to be calculated by taking into account applied deformation (strain rate) following Costa et al. (2009):

\[ \eta_r = \frac{1 + \left( \frac{\phi - \phi_*}{\phi_*} \right)}{(1 - F) \delta \phi} \] (A.22)

in which:

\[ F = (1 - \xi) \text{erf} \left[ \frac{\sqrt{\pi} \phi}{2(1 - \xi) \phi_*} \left( 1 + \left( \frac{\phi}{\phi_*} \right)^{\gamma} \right) \right] \]

here, \( \phi_* \) is the critical solid fraction that is present at the onset of the exponential increase in \( \eta_r \) with \( \phi \); \( \gamma \) is the slope of the relation between \( \eta_r \) and \( \phi \) as the crystal fraction approaches \( \phi_* \), and \( \delta \) is the slope of the relations for values of \( \phi \) greater than \( \phi_* \). \( \xi, \gamma \) and \( \delta \) are all empirical parameters that depend on particle shape and applied strain rate. Two default models are offered in PyFLOWGO: \textit{costa1} that
is only applicable for spherical particles (aspect ratio of 1) and costa2 that is for elongated particles (aspect ratio of 9). Both models can be used for strain rate set either at 1s\(^{-1}\) or 10\(^{-4}\)s\(^{-1}\) and use the values given in Cimarelli et al. (2011). For example, at 1s\(^{-1}\), costa1 produces: \(\phi^* = 0.67\), \(\xi = 0.01\), \(\gamma =1.6\) and \(\delta = 11.4\), while costa2 produces: \(\phi^* = 0.28\), \(\xi =0.001\), \(\gamma =8.55\) and \(\delta = 4.45\). More examples can be found in Costa et al. (2009), Cimarelli et al. (2011) and Chevrel et al. (2013).

**Phan-Thien and Pham model**

PyFLOWGO offers one model that allows the treatment as a three-phase mixture comprising a suspension of rigid spherical particles (\(\phi\)) and bubbles (\(\phi_b\)) following Phan-Thien and Pham (1997). This model is applicable only for \(\phi + \phi_b < 1\).

One of the three following cases can be applied:

- **Case ptp1**, crystals are smaller than bubbles:

\[
\eta_r = \left(1 - \frac{\phi}{1 - \phi_b}\right)^{-5/2} (1 - \phi_b)^{-1}
\]

(A.23)

- **Case ptp2**, crystals and bubbles are the same size:

\[
\eta_r = (1 - \phi - \phi_b)^{5\phi_b - 2\phi_b^2 \phi_b^2}
\]

(A.24)

- **Case ptp3**, crystals are larger than bubbles:

\[
\eta_r = \left(1 - \frac{\phi_b}{1 - \phi}\right)^{-1} (1 - \phi)^{-5/2}
\]

(A.25)

**Appendix A.5. Vesicle fraction models, \(\phi_b\)**

Two vesicle state models are available. The first is a simple model whereby the vesicle fraction is held constant down flow and is equal to the initial (at vent) value.
The second is a model as proposed by Harris and Rowland (2015), which allows the vesicle fraction to be changed after a given distance, \( x_{\text{critic}} \), and is intended to take into effect down flow degassing. In the PyFLOWGO framework these are the constant and bimodal model, respectively. In the bimodal model:

\[
\begin{align*}
\text{If } x \leq x_{\text{critic}}: & \quad \phi_b = \phi_{b1} \\
\text{If } x > x_{\text{critic}}: & \quad \phi_b = \phi_{b2}
\end{align*}
\] (A.26)

where \( x_{\text{critic}} \), \( \phi_{b1} \) and \( \phi_{b2} \) are the proximal and distal vesicularities, respectively and can be set using down flow assessments of lava density (e.g. Robert et al. 2014).

Appendix A.6. Yield strength and shear stress model

Velocity depends also on the yield strength of lava and on the basal shear stress (Eq. 7). PyFLOWGO provides one basal shear stress (\( \tau_b \)) model (Hulme 1974):

\[
\tau_b = dg \rho_{\text{bulk}} \sin(\theta)
\] (A.27)

where \( \tau_b \) is in Pa.

Lava yield strength can instead be calculated as a function of temperature and crystallinity following Dragoni (1989), and Pinkerton and Stevenson (1992) as proposed in the original version of FLOWGO. PyFLOWGO uses this approach of allowing \( \tau_0 \) to be calculated as function of lava temperature (\( T_{\text{core}} \)) using the liquidus temperature (\( T_0 \)) and the lava crystal content (\( \phi \)) as proposed by Ryerson et al. (1988) in:

\[
\tau_0 = 0.01 \left[ \exp^{0.08(T_0 - T_{\text{core}})} - 1 \right] + 6500\phi^{2.85}
\] (A.28)
Following [Harris and Rowland (2001)], PyFLOWGO also allows a model that considers the eruption temperature ($T_{erupt}$) instead of $T_0$ in Eq. A.27. In the PyFLOWGO framework these are the *dragoni* and *basic* model, respectively.

**Appendix A.7. Effective crust cover fraction model**

The upper surface of the lava is partially covered by a cooler crust. The fraction of this crusted lava is termed as the effective crust cover fraction, $f_{crust}$, and varies between zero (crust free, no insulation: rare in nature) and one (complete crust coverage, well-insulated; but not equivalent to a lava tube). Effective crust cover fraction directly affects the effective surface temperature (Eq. A.6) and the characteristic surface temperature (Eq. A.9) which, in turn, influences the heat fluxes due to radiation and forced convection. PyFLOWGO offers two models to calculate $f_{crust}$.

The *basic* model, as proposed in the original version of FLOWGO, allows $f_{crust}$ to vary down flow as function of velocity:

$$f_{crust} = f_{init} \exp^{\alpha V_{mean}} \quad \text{(A.29)}$$

where $f_{init}$ is the initial (at vent) crust fraction and $\alpha$ is a coefficient that varies crust cover as function of $V_{mean}$: crust cover increases as flow velocity decreases. Based on examination of aerial photographs of active channels flowing at known velocities, [Harris and Rowland (2001)] derived $f_{init}$ of 0.9 and $\alpha$ of -0.16, for poorly insulated flow, and $f_{init}$ of 1.0 and $\alpha$ of -0.00756 for more heavily crusted flow. Alternatively, $f_{crust}$ can be held constant down flow, equals $f_{init}$ at all down flow location when $\alpha = 0$.

The second model, named *bimodal* in PyFLOWGO, allows the dependence of
effective crust cover fraction with velocity to be changed after a given distance, $x_{\text{critic}}$, as proposed by Harris and Rowland (2015):

$$\begin{align*}
\text{If } x & \leq x_{\text{critic}}: f_{\text{crust}} = f_{\text{init}} \exp^{\alpha_1 V_{\text{mean}}} \\
\text{If } x > x_{\text{critic}}: f_{\text{crust}} = f_{\text{init}} \exp^{\alpha_2 V_{\text{mean}}}
\end{align*}$$

(A.30)

where $\alpha_1$ and $\alpha_2$ are the crust cover growth coefficients for proximal and distal channel reaches, respectively and are determined by field observation; for which we need more measurements [i.e. for the relationship between $V_{\text{mean}}$ and $f_{\text{crust}}$ (Harris and Rowland 2015)].

Appendix A.8. Crust temperature models

PyFLOWGO provides three models to calculate the temperature of the crust, $T_{\text{crust}}$. The constant model allows the at-vent initial crust temperature to be held constant down flow. The hon model, as suggested in the original FLOWGO version, allows calculation of $T_{\text{crust}}$ (in °C) following Hon et al. (1994):

$$T_{\text{crust}} = -140 \log \left( \frac{\text{time}}{3600} \right) + 303 + 273.15$$

(A.31)

where $\text{time}$ is in seconds and is calculated via:

$$\text{time} = \frac{\partial x}{V_{\text{mean}}}$$

(A.32)

in which $\partial x$ is one down flow distance increment. This equation implies implicitly that the initial crust temperature is 1070°C and as it is an empirical relationship determined from Hawaiian pahoehoe lava, but - given that it is based on the Stefan cooling problem (Harris et al. 2005) - it can be adapted to any basaltic surface.
cooling due to radiation.

The third model, as suggested by [Harris and Rowland (2015)], allows $T_{\text{crust}}$ to vary as function of time according to [Hon et al. (1994)] from the vent until a given distance, $x_{\text{critic}}$, and then to be held constant:

$$\begin{align*}
\text{If } x \leq x_{\text{critic}}: & \quad T_{\text{crust}} = -140 \log \left( \frac{\text{time}}{3600} \right) + 303 + 273.15 \\
\text{If } x > x_{\text{critic}}: & \quad T_{\text{crust}} = T_{\text{init}}
\end{align*}$$

(A.33)

In the PyFLOWGO framework, this is the honbimodal model.

**Appendix A.9. Uncrusted surface temperature model**

The temperature of the uncrusted lava surface, $T_{\text{hot}}$, will be lower than of the flow core (e.g. Calvari et al. 1994, Flynn and Mouginis-Mark 1994, Harris et al. 1998). PyFLOWGO accounts for this difference via:

$$T_{\text{hot}} = T_{\text{core}} - \text{buffer}$$

(A.34)

where $\text{buffer}$ is the temperature difference between the maximum surface temperature and the core temperature, and is set by the user. Based on field measurements using thermocouples and radiometers, [Harris and Rowland (2001)] give $\text{buffer} = 140$. The buffer value may also be lower; [Bailey et al. (2006)] described an active channel on Etna with a maximum surface temperature of 1042 °C and a core temperature of 1065 °C.
Table A.1: Description of the input parameters contained in the json file.

<table>
<thead>
<tr>
<th>Input parameters in json file</th>
<th>Symbol</th>
<th>Definition</th>
<th>Unit</th>
<th>Constant for Earth</th>
</tr>
</thead>
<tbody>
<tr>
<td>lava_name</td>
<td></td>
<td>name of the lava flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slope_file</td>
<td></td>
<td>file containing distance (m) and slope (°)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>step_size</td>
<td></td>
<td>step size for lava advance down flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>terrain_conditions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>w</td>
<td>channel’s width</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>depth</td>
<td>d</td>
<td>channel’s depth</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>gravity</td>
<td>g</td>
<td>gravity of the planet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_channel_length</td>
<td>Lmax</td>
<td>maximum flow length*</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td><strong>eruption_conditions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eruption_temperature</td>
<td>T_{erupt}</td>
<td>temperature of the eruption</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>viscosity_eruption</td>
<td>\eta_{erupt}</td>
<td>viscosity of the lava at T_{erupt}, only for “basic” melt viscosity model</td>
<td>Pa s</td>
<td></td>
</tr>
<tr>
<td>position</td>
<td>x</td>
<td>distance from the vent at which the iteration starts</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>critical_distance</td>
<td>x_{critic}</td>
<td>distance when the bimodal models change, only when “bimodal” models</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>t</td>
<td>time at which the iteration starts, only for “hon” and “honbimodal” crust temperature model</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>crystal_fraction</td>
<td>\phi</td>
<td>initial crystal fraction</td>
<td>n.a.</td>
<td></td>
</tr>
<tr>
<td>density_dre</td>
<td>\rho_{DRE}</td>
<td>dense rock equivalent density</td>
<td>kg/m³</td>
<td></td>
</tr>
<tr>
<td>vesicle_fraction</td>
<td>\phi_v</td>
<td>initial fraction of vesicle at the vent</td>
<td>n.a.</td>
<td></td>
</tr>
<tr>
<td>liquidus_temperature</td>
<td>L_0</td>
<td>temperature of the liquidus, only for “dragoni” melt viscosity model</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td><strong>radiation_parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stefan-boltzmann_sigma</td>
<td>\sigma</td>
<td>stefan-boltzmann constant</td>
<td>W/m²K</td>
<td>5.669E-8</td>
</tr>
<tr>
<td>emissivity</td>
<td>\epsilon</td>
<td>emissivity</td>
<td>n.a.</td>
<td>0.98</td>
</tr>
<tr>
<td><strong>conduction_parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>basal_temperature</td>
<td>I_{base}</td>
<td>temperature at the base of the flow</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>core_base_distance</td>
<td>H_0</td>
<td>percentage of base layer over flow depth</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td><strong>rain_parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rainfall_rate</td>
<td>\partial R/\partial t</td>
<td>rainfall rate</td>
<td>m/s</td>
<td></td>
</tr>
<tr>
<td>density_water</td>
<td>\rho_{w,o}</td>
<td>density of the water</td>
<td>kg/m³</td>
<td>958</td>
</tr>
<tr>
<td>latent_Heat_vaporization</td>
<td>L_{v,w,o}</td>
<td>latent heat of vaporisation of the water</td>
<td>J/kg</td>
<td>2800000</td>
</tr>
<tr>
<td><strong>convection_parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wind_speed</td>
<td>U</td>
<td>wind speed</td>
<td>m/s</td>
<td></td>
</tr>
<tr>
<td>air_temperature</td>
<td>T_{atmo}</td>
<td>temperature of the air</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>air_density</td>
<td>\rho_{atmo}</td>
<td>density of the air</td>
<td>kg/m³</td>
<td>0.4411</td>
</tr>
<tr>
<td>air_specific_heat_capacity</td>
<td>C_{P_{atmo}}</td>
<td>heat capacity of the air</td>
<td>J/kg K</td>
<td>1099</td>
</tr>
<tr>
<td><strong>thermal_parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>buffer</td>
<td>buffer</td>
<td>difference between T_{core} and T_{hot}</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>crust_cover_fraction</td>
<td>f_{crust}</td>
<td>initial crust cover fraction</td>
<td>n.a.</td>
<td></td>
</tr>
<tr>
<td>alpha</td>
<td>\alpha</td>
<td>coefficient for velocity dependence of the crust cover</td>
<td>n.a.</td>
<td></td>
</tr>
<tr>
<td>initial_crust_temperature</td>
<td>T_{init}</td>
<td>chilled crust temperature</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td><strong>melt viscosity_parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shaw_slope</td>
<td>s</td>
<td>coefficient calculated from melt chemical composition only for “shaw” melt viscosity model</td>
<td>Pa.s</td>
<td></td>
</tr>
<tr>
<td>a_vft</td>
<td>A</td>
<td>coefficient calculated from melt chemical composition, only for “vft” melt viscosity model</td>
<td>Pa/mol</td>
<td></td>
</tr>
<tr>
<td>b_vft</td>
<td>B</td>
<td>coefficient calculated from melt chemical composition, only for “vft” melt viscosity model</td>
<td>Pa/mol</td>
<td></td>
</tr>
<tr>
<td>crystals_parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>crystals_grown_during_cooling</td>
<td>\phi_{grown}</td>
<td>fraction of crystal grown during emplacement, only for “basic” crystallization rate model</td>
<td>n.a.</td>
<td></td>
</tr>
<tr>
<td>solid_temperature</td>
<td>T_{solid}</td>
<td>temperature at which the lava cannot flow, only for “basic” crystallization rate model</td>
<td>K</td>
<td>crystals/C</td>
</tr>
<tr>
<td>crystallization_rate_1</td>
<td>C_1</td>
<td>crystallization rate, only for “bimodal” crystallization rate model</td>
<td>K</td>
<td>crystals/C</td>
</tr>
<tr>
<td>crystallization_rate_2</td>
<td>C_2</td>
<td>crystallization rate, only for “bimodal” crystallization rate model</td>
<td>K</td>
<td>crystals/C</td>
</tr>
<tr>
<td>latent_heat_of_crystallization</td>
<td>L</td>
<td>latent heat of crystallization</td>
<td>J/kg</td>
<td>350000</td>
</tr>
<tr>
<td><strong>relative_viscosity_parameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_packing</td>
<td>\phi_{b}</td>
<td>maximum fraction of crystals, only for “kd” and “mp” relative viscosity models</td>
<td>n.a.</td>
<td></td>
</tr>
<tr>
<td>einstein_coeff</td>
<td>\phi_{e}</td>
<td>Einstein coefficient or intrinsic viscosity, only for “kd” relative viscosity models</td>
<td>Pa.s</td>
<td>0.0001 or 1</td>
</tr>
<tr>
<td>strain_rate</td>
<td>b</td>
<td>maximum fraction of crystals, only for “costa1” and “costa2” relative viscosity models</td>
<td>Pa.s</td>
<td></td>
</tr>
</tbody>
</table>

*used in case the limiting conditions (\upsilon_{mean} = 0 or \phi = \phi_{max} or T_{core} = T_{solid}) are not reached.

** C_{U} = (U/U')^2 where U’ is the fraction of wind speed according to Keszthelyi and Denlinger, 1996.
Table A.2: Models’ choice

<table>
<thead>
<tr>
<th>Model's name</th>
<th>Symbol</th>
<th>What to write in the json file</th>
</tr>
</thead>
<tbody>
<tr>
<td>heat_budget_models</td>
<td></td>
<td></td>
</tr>
<tr>
<td>radiation</td>
<td>$Q_{rad}$</td>
<td>“yes” / “no”</td>
</tr>
<tr>
<td>conduction</td>
<td>$Q_{cond}$</td>
<td>“yes” / “no”</td>
</tr>
<tr>
<td>convection</td>
<td>$Q_{conv}$</td>
<td>“yes” / “no”</td>
</tr>
<tr>
<td>rain</td>
<td>$Q_{rain}$</td>
<td>“yes” / “no”</td>
</tr>
<tr>
<td>viscous_heating</td>
<td>$Q_{vis}$</td>
<td>“yes” / “no”</td>
</tr>
<tr>
<td>crystallization_rate_model</td>
<td>$\partial \vartheta / \partial T$</td>
<td>“basic” / “bimodal” / “bimodal_f_temp” / “melts”</td>
</tr>
<tr>
<td>melt_viscosity_model</td>
<td>$\eta_{melt}$</td>
<td>“basic” / “dragoni” / “shaw” / “vft”</td>
</tr>
<tr>
<td>relative_viscosity_model</td>
<td>$\eta$</td>
<td>“er” / “mp” / “kd” / “costa1” / “costa2” / “ptp1” / “ptp2” / “ptp3”</td>
</tr>
<tr>
<td>yield_strength_model</td>
<td>$T_0$</td>
<td>“basic” / “dragoni”</td>
</tr>
<tr>
<td>crust_temperature_model</td>
<td>$T_{crust}$</td>
<td>“basic” / “hon” / “bimodal”</td>
</tr>
<tr>
<td>effective_cover_crust_model</td>
<td>$f_{crust}$</td>
<td>“basic” / “bimodal”</td>
</tr>
<tr>
<td>vesicle_fraction_model</td>
<td>$\phi_b$</td>
<td>“constant” / “bimodal”</td>
</tr>
</tbody>
</table>

44
Table A.3: Input parameters of the json files used for the test cases

<table>
<thead>
<tr>
<th>Parameter</th>
<th>ML34-HK2015_cold</th>
<th>ML74_Robertetal2014</th>
<th>Pdf-2010_srtm</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;lava_name&quot;</td>
<td>&quot;ML34-slope_file&quot;</td>
<td>&quot;DEM_maulaulu74.txt&quot;</td>
<td>&quot;DEM_pdf2010_srtm.txt&quot;</td>
</tr>
<tr>
<td>&quot;step_size&quot;</td>
<td>10.0</td>
<td>200.0</td>
<td>10.0</td>
</tr>
<tr>
<td>&quot;models&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;crystallization_rate_model&quot;</td>
<td>&quot;basic&quot;</td>
<td>&quot;bimodal&quot;</td>
<td>&quot;basic&quot;</td>
</tr>
<tr>
<td>&quot;melt_viscosity_model&quot;</td>
<td>&quot;shaw&quot;</td>
<td>&quot;vft&quot;</td>
<td>&quot;vft&quot;</td>
</tr>
<tr>
<td>&quot;relative_viscosity_model&quot;</td>
<td>&quot;er&quot;</td>
<td>&quot;pdf2&quot;</td>
<td>&quot;er&quot;</td>
</tr>
<tr>
<td>&quot;yield_strength_model&quot;</td>
<td>&quot;dragoni&quot;</td>
<td>&quot;vft&quot;</td>
<td>&quot;constant&quot;</td>
</tr>
<tr>
<td>&quot;crust_temperature_model&quot;</td>
<td>&quot;hon&quot;</td>
<td>&quot;constant&quot;</td>
<td>&quot;constant&quot;</td>
</tr>
<tr>
<td>&quot;effective_cover_crust_model&quot;</td>
<td>&quot;basic&quot;</td>
<td>&quot;basic&quot;</td>
<td>&quot;basic&quot;</td>
</tr>
<tr>
<td>&quot;vesicle_fraction_model&quot;</td>
<td>&quot;constant&quot;</td>
<td>&quot;constant&quot;</td>
<td>&quot;constant&quot;</td>
</tr>
<tr>
<td>&quot;heat_budget_models&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;radiation&quot;</td>
<td>&quot;yes&quot;</td>
<td>&quot;yes&quot;</td>
<td>&quot;yes&quot;</td>
</tr>
<tr>
<td>&quot;conduction&quot;</td>
<td>&quot;yes&quot;</td>
<td>&quot;yes&quot;</td>
<td>&quot;yes&quot;</td>
</tr>
<tr>
<td>&quot;convection&quot;</td>
<td>&quot;yes&quot;</td>
<td>&quot;yes&quot;</td>
<td>&quot;yes&quot;</td>
</tr>
<tr>
<td>&quot;viscous_heating&quot;</td>
<td>&quot;no&quot;</td>
<td>&quot;no&quot;</td>
<td>&quot;no&quot;</td>
</tr>
<tr>
<td>&quot;terrain_conditions&quot;</td>
<td>see note</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;width&quot;</td>
<td>21</td>
<td>2.475</td>
<td>4.5</td>
</tr>
<tr>
<td>&quot;depth&quot;</td>
<td>3</td>
<td>2.475</td>
<td>1.4</td>
</tr>
<tr>
<td>&quot;gravity&quot;</td>
<td>9.8</td>
<td>9.8</td>
<td>9.8</td>
</tr>
<tr>
<td>&quot;max_channel_length&quot;</td>
<td>50000</td>
<td>10000</td>
<td>4000</td>
</tr>
<tr>
<td>&quot;eruption_condition&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;eruption_temperature&quot;</td>
<td>1410.15</td>
<td>1438.15</td>
<td>1387.15</td>
</tr>
<tr>
<td>&quot;lava_state&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;critical_distance&quot;</td>
<td>0</td>
<td>4190</td>
<td>0</td>
</tr>
<tr>
<td>&quot;position&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&quot;time&quot;</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>&quot;crystal_fraction&quot;</td>
<td>0.25</td>
<td>0.1097</td>
<td>0.104</td>
</tr>
<tr>
<td>&quot;density_die&quot;</td>
<td>2724</td>
<td>2900</td>
<td>2970</td>
</tr>
<tr>
<td>&quot;vesicle_fraction&quot;</td>
<td>0.140</td>
<td>0.4866</td>
<td>0.64</td>
</tr>
<tr>
<td>&quot;liquidus_temperature&quot;</td>
<td>1473.15</td>
<td>1473.15</td>
<td>1473.15</td>
</tr>
<tr>
<td>&quot;radiation_parameters&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;stefan-boltzmann_sigma&quot;</td>
<td>5.67e-8</td>
<td>5.67e-8</td>
<td>5.67e-8</td>
</tr>
<tr>
<td>&quot;emissivity_epsilon&quot;</td>
<td>0.95</td>
<td>0.98</td>
<td>0.95</td>
</tr>
<tr>
<td>&quot;conduction_parameters&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;basal_temperature&quot;</td>
<td>773.15</td>
<td>1273.15</td>
<td>773.15</td>
</tr>
<tr>
<td>&quot;core_base_distance&quot;</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>&quot;convection_parameters&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;wind_speed&quot;</td>
<td>5.12259623</td>
<td>5.12259623</td>
<td>5</td>
</tr>
<tr>
<td>&quot;ch_air&quot;</td>
<td>0.0036</td>
<td>0.0036</td>
<td>0.0036</td>
</tr>
<tr>
<td>&quot;air_temperature&quot;</td>
<td>283.15</td>
<td>293.15</td>
<td>293.15</td>
</tr>
<tr>
<td>&quot;air_density&quot;</td>
<td>0.4412</td>
<td>0.4412</td>
<td>0.4412</td>
</tr>
<tr>
<td>&quot;air_specific_heat_capacity&quot;</td>
<td>1099</td>
<td>1099</td>
<td>1099</td>
</tr>
<tr>
<td>&quot;thermal_parameters&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;buffer&quot;</td>
<td>140</td>
<td>0</td>
<td>140</td>
</tr>
<tr>
<td>&quot;crust_cover_fraction&quot;</td>
<td>0.9023</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>&quot;alpha&quot;</td>
<td>-0.1601</td>
<td>0</td>
<td>-0.0076</td>
</tr>
<tr>
<td>&quot;crust_temperature&quot;</td>
<td>698.15</td>
<td>1273.15</td>
<td>773.15</td>
</tr>
<tr>
<td>&quot;melt_viscosity_parameters&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;shaw_slope&quot;</td>
<td>2.36</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&quot;a_vft&quot;</td>
<td>0</td>
<td>-4.7</td>
<td>-4.52</td>
</tr>
<tr>
<td>&quot;b_vft&quot;</td>
<td>0</td>
<td>5429.7</td>
<td>5558</td>
</tr>
<tr>
<td>&quot;c_vft&quot;</td>
<td>0</td>
<td>595.5</td>
<td>582.9</td>
</tr>
<tr>
<td>&quot;crystals_parameters&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;crystals_grown_during_cooling&quot;</td>
<td>0.45</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>&quot;solid_temperature&quot;</td>
<td>1243.15</td>
<td>1268.15</td>
<td>1237.15</td>
</tr>
<tr>
<td>&quot;latent_heat_of_crystallization&quot;</td>
<td>350000</td>
<td>350000</td>
<td>350000</td>
</tr>
<tr>
<td>&quot;crystallization_rate_1&quot;</td>
<td>-</td>
<td>0.003</td>
<td>-</td>
</tr>
<tr>
<td>&quot;crystallization_rate_2&quot;</td>
<td>-</td>
<td>0.025</td>
<td>-</td>
</tr>
</tbody>
</table>
Highlights:

- PyFLOWGO is an updated version of FLOWGO (Harris & Rowland 2001) written in Python
- PyFLOWGO simulates the thermo-rheological properties of lava flowing down a channel
- Its architecture is based on communication of interfaces allowing great flexibility
- Users can switch between models or add new ones without modifying the architecture
- PyFLOWGO has been unit tested and packaged in a open access Python library