PyBox: a Python tool for simulating the kinematics of Pyroclastic density currents with the box-model approach
Reference and User's Guide

Giovanni Biagioli, Andrea Bevilacqua,
Tomaso Esposti Ongaro, Mattia de' Michieli Vitturi

May 9, 2019

This report describes the physical principles, numerical method and implementation into a Python code of an integral (box) model to simulate the kinematics of *pyroclastic density currents* (PDCs) and the PDC invasion maps over a rugged topography.

The kinematic model analyzed here is based on the *box model* formulation described in many previous papers (Huppert and Simpson, 1980; Dade and Huppert, 1995a; Bonnecaze et al., 1995; Hallworth et al., 1998; Esposti Ongaro et al., 2016). It describes the propagation of a turbulent particle-laden gravity current, generated by the sudden release of a homogeneous fluid with suspended particles into a still atmosphere. Inertial effects are assumed to have a leading role with respect to viscous forces and particle-particle interactions. Particle sedimentation, which modifies the current inertia during propagation, is also taken into account.

The procedure to incorporate the effects of the topography on the PDC box model is based on the *energy conoid* approach proposed by Neri et al. (2015); Bevilacqua (2016); Bevilacqua et al. (2017).

In Section 1, we first briefly recall the main characteristics of PDCs, from a physical and geological point of view basing upon Roche et al. (2013); Dufek et al. (2015).

In Section 2, we derive the differential equations of the box model, for channelized (Cartesian) and axisymmetric geometries and for particle-laden gravity currents. The model is based on some simplifying assumptions, already discussed in previous works Bevilacqua (2016); Esposti Ongaro et al. (2016); Bevilacqua et al. (2017).

Section 3 deals with PDCs hazard assessment in active volcanic regions. In particular, the distribution of areas exposed to PDCs invasion is discussed and is the focus of the following subsections 4.5-4.6.

Finally, a new Python-3.x implementation of the box model is described and attached to these notes. Section 4 contains some discussion on the implementation of the `pybox.py` code.

1 Phenomenological aspects of PDCs

PDCs are horizontal, gravity-driven currents composed of a hot mixture of volcanic gas and solid particles, generated by explosive volcanic phenomena. They are typically generated by the gravitational collapse of lava domes or explosive eruption columns. PDCs display a coexistence of different flow regimes, ranging from a basal, dense *pyroclastic flow* (originating by the progressive sedimentation of granular particles) to an upper, dilute, turbulent *pyroclastic surge* (Roche et al., 2013; Esposti Ongaro et al., 2016).
The following Figure 1 schematically represents the above concepts.

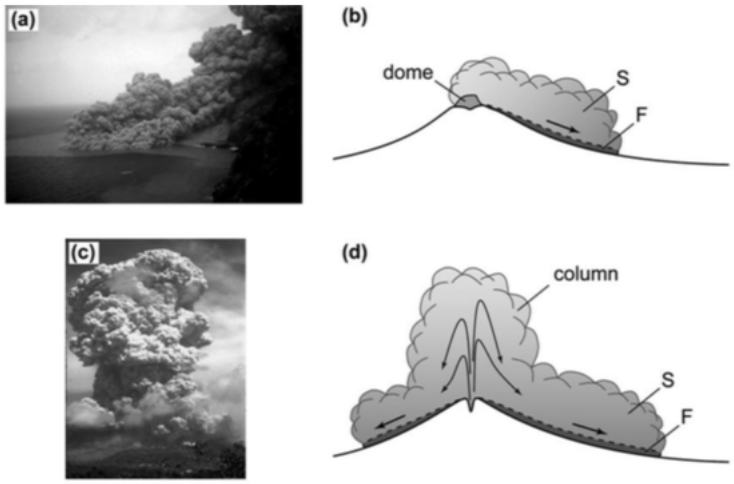


Figure 1: *Examples and schematic representations of PDCs different generation mechanisms: lava dome collapse (a, b) and volcanic plume collapse (c,d). Basal dense granular avalanche and upper pyroclastic surge are labeled F and S, respectively (from Roche et al. (2013)).*

The transition between the two regions with aforementioned regimes can be rough or progressive and can occur at different heights. Naturally, the concentrated flow at the current base is strongly influenced by topography (since it is commonly up to a few tens of meters thick), whereas the upper ash-laden cloud is not. Pyroclastic density currents can travel over distances up to tens of kilometres, at speed of up to $\sim 200\ m \cdot s^{-1}$ (Roche et al., 2013). However, dense pyroclastic flows can have relatively low speeds of $\sim 5 - 20\ m \cdot s^{-1}$, as discussed in Roche et al. (2016).

1.1 Density currents and PDC regimes

Gravity currents originate whenever a density difference between two fluids results in one laterally flowing into the other one. For PDCs, such a density difference arises

from the presence of suspended particles and hot gas in the flow, due to the turbulence generated by the flow. PDCs contain pyroclasts (deriving from magma fragmentation), whose size varies from the order of microns (ashes) to the order of centimeters (lapilli), up to meters (blocks).

Typically, density currents dynamics can be described in three distinct phases (Huppert and Simpson, 1980; Roche et al., 2013):

- the *slumping phase*, in which the gas-particle mixture collapses, intruding into the ambient fluid: herein, the flow dynamics is controlled by the release conditions and geometry;
- the *inertial phase*, during which the flow motion is governed by the balance of inertia and buoyancy forces;
- the *stopping phase*, wherein the buoyancy force of the intruding fluid is balanced by viscous forces and this balance leads to the stop of the current.

The box model presented in this report is suited for the simulation of the intertial phase of a particle-laden gravity current. In our model, based on numerical experiments by Esposti Ongaro et al. (2016) we assume that the basal flow has not a controlling role in the current kinematics.

2 The box model

The model consists of a set of ordinary differential equations, that provide the time evolution of the PDC front velocity, $u(t)$, together with the current height $h(t)$ and the solid particle volume fraction $\epsilon_i(t)$, $i = 1, \ldots, N$, N being the number of particle classes considered.
We assume the classical *dam break configuration*, in which a column of fluid instantaneously collapses and propagates, under gravity, in a surrounding atmosphere with uniform density ρ_{atm}. Several examples are available in literature and significant cases are provided by various Computational Fluid Dynamics toolboxes (e.g., *Breaking of a dam*, in CFD Direct *OpenFOAM v6 User Guide* (2018), available at `https://cfd.direct/openfoam/user-guide/v6-damBreak/`.) Other authors (Bonnecaze et al., 1995; Dade and Huppert, 1995b, 1996) have considered gravity currents produced by the constant-flux release of dense suspension from a plane source. Extension of the model presented here to such a configuration is straightforward and will be implemented in the next version of the model.

PDCs are driven by their density excess with respect to the surrounding air: the density of the current ρ_{c} is defined as the sum of the density of an interstitial gas, ρ_{g}, and the bulk densities of the pyroclasts carried by the flow, i.e.,

$$\rho_c = (1 - \epsilon_{\mathrm{tot}})\rho_{\mathrm{g}} + \sum_{i=1}^{N} \epsilon_i \rho_i^s, \quad \epsilon_{\mathrm{tot}} = \sum_{i=1}^{N} \epsilon_i, \tag{1}$$

where ρ_i^s, $i = 1, \ldots, N$, is the density of the $i-$th particle class. We assume $\rho_c > \rho_{atm}$ and $\rho_{atm} > \rho_g$, since at PDCs temperatures (typically $300 - 700$ °C) the density of the interstitial gas is lower than the atmospheric one (Dufek et al., 2015).
A proper way to express the density contrast between the current and the ambient fluid is given by the *reduced gravity*

$$g' \stackrel{\text{def}}{=} \frac{\rho_c - \rho_{atm}}{\rho_{atm}} g,$$

that can be rewritten as

$$g' = \frac{\left(1 - \sum_{i=1}^{N} \epsilon_i\right) \rho_g + \sum_{i=1}^{N} \epsilon_i \rho_i^s - \rho_{atm}}{\rho_{atm}} g = \frac{\rho_g - \rho_{atm}}{\rho_{atm}} g + \sum_{i=1}^{N} \epsilon_i \frac{\rho_i^s - \rho_g}{\rho_{atm}} g. \qquad (2)$$

That said, we make some additional simplifying hypotheses.
First of all, we assume that the mixture flow regime is incompressible and inviscid. The hypothesis of inviscid fluid is reasonable since we assume that the dynamics of the current is dominated by the balance between inertial and buoyancy forces, as previously stated: this allow us to neglect viscosity. Note that the assumption of incompressibility implies that the current volume $V(t)$ remains constant, i.e., $V(t) = V_0$ (see Figure 2). Neglecting compressibility effects is acceptable because velocity is much lower than the speed of sound (Esposti Ongaro et al., 2016).
Moreover, we assume that, within the current, the vertical mixing, due to turbulence, produces a vertically uniform distribution of particles. The particles are assumed to sediment out of the current at a rate proportional to their (constant) terminal (or settling) velocity w_i^s, $i = 1, \ldots, N$. Once deposited, they cannot be re-entrained by the flow. Finally, surface effects of the ambient fluid are neglected.

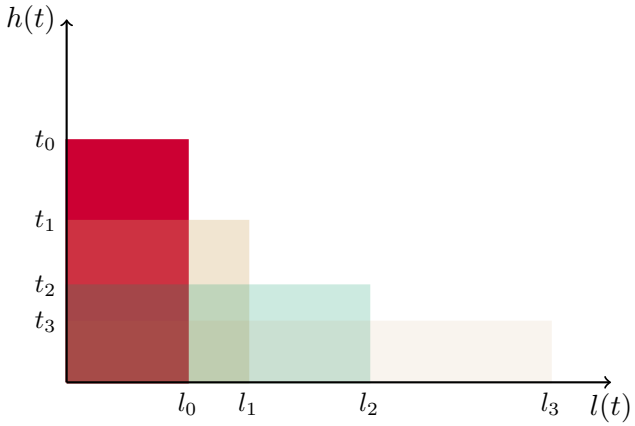


Figure 2: *Evolution of channelized currents through a series of equal-area rectangles, according to the model (hence the name "box model").*

Under these hypotheses, the box model for particle-laden gravity currents states that the velocity of the current front is related to the average depth of the current by the *von Kármán equation for density currents* (Benjamin, 1968; Huppert and Simpson, 1980; Dade and Huppert, 1995a; Bonnecaze et al., 1995; Hallworth et al., 1998; Bevilacqua, 2016)

$$u = \mathsf{Fr}\sqrt{g'h}, \tag{3}$$

where Fr is the *Froude number*, a dimensionless number expressing the ratio between inertial and buoyancy forces.

Nevertheless, since (3) is a single equation relating the unknown variables $u(t)$ and $h(t)$, we have to specify other equations to completely determine the current kinematics. So, we assume that particles can settle to the ground and this process changes the solid particle fractions $\epsilon_i(t)$, $i = 1, \dots, N$. In particular, the rate of the $i-$th particulate class volume loss is due to sedimentation over the current base surface B, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t}(\epsilon_i V_0) = -w_i^s \epsilon_i B, \quad i = 1, \dots, N,$$

which reduces to

$$\frac{\mathrm{d}\epsilon_i}{\mathrm{d}t} = -w_i^s \epsilon_i \frac{B}{V_0} = -\frac{w_i^s \epsilon_i}{h}, \quad i = 1, \dots, N.$$

Cartesian currents. Hence, the box model for channelized particle-laden gravity currents can be written as:

$$\frac{\mathrm{d}l}{\mathrm{d}t} = \mathsf{Fr}\sqrt{g'h} \tag{4}$$

$$lh = l_0 h_0 = V_0 \tag{5}$$

$$\frac{\mathrm{d}\epsilon_i}{\mathrm{d}t} = -\frac{w_i^s \epsilon_i}{h}, \quad i = 1, \dots N, \tag{6}$$

where l_0 and h_0 are, respectively, the length and the height of the current before being released.

Axisymmetric currents. When considering radially spreading currents, Eqns. (4) and (6) are unaltered (Huppert and Simpson, 1980) whereas the constant-volume condition is expressed by

$$\xi \pi l^2 h = \xi \pi l_0^2 h_0 = V_0,$$

where ξ is a *spreading factor*, $\xi = \dfrac{\theta}{2\pi}$, which allow us to consider currents spreading within a circular sector of central angle θ, measured in radians (for isotropic currents, $\xi = 1$). In such a way, we can describe the propagation of PDCs originating from the gravitational collapse of a portion of the eruption column (hence an asymmetric collapse).

The box model for axisymmetric currents thus reads:

$$\frac{\mathrm{d}l}{\mathrm{d}t} = \mathsf{Fr}\sqrt{g'h} \tag{7}$$

$$l^2 h = l_0^2 h_0 = \frac{V_0}{\xi\pi} \tag{8}$$

$$\frac{\mathrm{d}\epsilon_i}{\mathrm{d}t} = -\frac{w_i^s \epsilon_i}{h}, \quad i = 1, \dots N. \tag{9}$$

2.1 Analytical expression for the maximum current runout

In the case of monodisperse systems, we can determine analytical solutions for both cartesian and radially spreading currents, assuming the presence of either a buoyant interstitial gas or the neutrally buoyant interstitial ambient fluid (Neri et al., 2015; Bevilacqua, 2016).
In particular, an explicit expression for the maximum flow runout l_∞, i.e., the distance at which $\rho_\mathrm{c} = \rho_\mathrm{atm}$, is available in the above mentioned cases (Bonnecaze et al., 1995; Esposti Ongaro et al., 2016; Bevilacqua, 2019). Note that, when considering monodisperse systems, from (1) we have that

$$\frac{\rho_\mathrm{c} - \rho_\mathrm{atm}}{\rho_\mathrm{atm}} = \frac{\epsilon\rho^s + (1-\epsilon)\rho_\mathrm{g} - \rho_\mathrm{atm}}{\rho_\mathrm{atm}} = \epsilon\frac{\rho^s - \rho_\mathrm{g}}{\rho_\mathrm{atm}} + \frac{\rho_\mathrm{g} - \rho_\mathrm{atm}}{\rho_\mathrm{atm}} = (\epsilon - \epsilon_\mathrm{cr})\frac{\rho^s - \rho_\mathrm{g}}{\rho_\mathrm{atm}},$$

where $\epsilon_\mathrm{cr} \overset{\mathrm{def}}{=} \dfrac{\rho_\mathrm{atm} - \rho_\mathrm{g}}{\rho^s - \rho_\mathrm{g}}$. Note that, when the interstitial fluid is the ambient air, $\epsilon_\mathrm{cr} = 0$. Hence, the current flow stops when $\epsilon = \epsilon_\mathrm{cr}$.
The analytical expressions for the maximum runout are summarized below. We use the notation

$$g'_p \overset{\mathrm{def}}{=} \frac{\rho^s - \rho_\mathrm{atm}}{\rho_\mathrm{atm}} g \quad \text{and} \quad g''_p \overset{\mathrm{def}}{=} \frac{\rho^s - \rho_\mathrm{g}}{\rho_\mathrm{atm}} g.$$

Analytic expressions maximum flow runout

Cartesian current without interstitial buoyant gas

$$l_\infty = \left(\frac{5\mathsf{Fr}\left(\epsilon(0) g'_p V_0^3\right)^{\frac{1}{2}}}{w^s} + l_0^{\frac{5}{2}} \right)^{\frac{2}{5}}$$

Axisymmetric current without interstitial buoyant gas

$$l_\infty = \left(8\mathsf{Fr} \left(\epsilon(0) g'_p \left(\frac{V_0}{\xi\pi} \right)^3 \right)^{\frac{1}{2}} \frac{1}{w_s} + l_0^4 \right)^{\frac{1}{4}}$$

Cartesian current with interstitial buoyant gas

$$l_\infty = \left(\frac{5\mathsf{Fr}\left(\epsilon_\mathrm{cr} g''_p V_0^3\right)^{\frac{1}{2}}}{w^s} \left(\left(\frac{\epsilon(0)}{\epsilon_\mathrm{cr}} - 1 \right)^{\frac{1}{2}} - \arctan\left(\frac{\epsilon(0)}{\epsilon_\mathrm{cr}} - 1 \right)^{\frac{1}{2}} \right) + l_0^{\frac{5}{2}} \right)^{\frac{2}{5}}$$

Axisymmetric current with interstitial buoyant gas

$$l_\infty = \left(8\mathsf{Fr} \left(\epsilon_{\mathrm{cr}} g_p'' \left(\frac{V_0}{\xi \pi} \right)^3 \right)^{\frac{1}{2}} \frac{1}{w^s} \left(\left(\frac{\epsilon(0)}{\epsilon_{\mathrm{cr}}} - 1 \right)^{\frac{1}{2}} - \arctan \left(\frac{\epsilon(0)}{\epsilon_{\mathrm{cr}}} - 1 \right)^{\frac{1}{2}} \right) + l_0^4 \right)^{\frac{1}{4}}$$

The box model has been intensively tested against laboratory experiments and multi-dimensional Eulerian multiphase flow models, able to describe the dynamics of stratified PDCs, providing fairly accurate results (Roche et al., 2013; Neri et al., 2015; Esposti Ongaro et al., 2016).

3 PDCs invasion maps

A potential use for the box model is represented by the definition of proper hazard maps, the hazard being the possibility of being reached by a PDC.
Set a vent location, we calculate the maximum flow runout over flat topographies, computed by the box model and then we assess the capability of topographic reliefs to block the current.

In particular, the invasion areas can be obtained by using the so-called *energy-conoid model* (Orsucci, 2014), based on the assumption of non-linear, monotonic decay of flow kinetic energy with distance (Bevilacqua, 2016).
In more detail, we determine the maximum height $h_{\max}$ of an obstacle the flow can overcome. Then we compare the kinetic energy of the current front and the potential energy associated to the obstacle top, i.e.,

$$\frac{1}{2}\rho_{\mathrm{c}} \left(\frac{\mathrm{d}l}{\mathrm{d}t} \right)^2 = (\rho_{\mathrm{c}} - \rho_{\mathrm{atm}}) g h_{\max} \Rightarrow h_{\max} = \frac{1}{2} \frac{\rho_{\mathrm{c}}}{\rho_{\mathrm{c}} - \rho_{\mathrm{atm}}} \frac{1}{g} \left(\frac{\mathrm{d}l}{\mathrm{d}t} \right)^2 .$$

In the case of monodisperse mixtures without hot interstitial fluid, we can analytically solve the equation on the left, for both cartesian and cylindrical currents (Neri et al., 2015; Bevilacqua, 2016).
For simplicity, we are neglecting returning waves. Moreover, a sea surface, if present, is considered as a flat topography with no influence on a PDC traveling over (Bevilacqua et al., 2017).

4 Implementation of the PYroclastic BOX Model in the Python code `pybox.py`

The Python routine described in the sequel has been developed at Istituto Nazionale di Geofisica e Vulcanologia (INGV) - Sezione di Pisa . Its most important characteristics are summarized below.

4.1 Input parameters

The code requires the following parameters:

- `dt`, that sets the time step at which output of data is stored;[1]
- `l0`, `h0` and `theta0`, i.e., the initial current front position, depth and temperature. Note that, if `theta0` is equal to the ambient temperature of 300 K, the interstitial fluid density is assumed to be the ambient density, while values above it allow us to take into account the presence of a buoyant interstitial gas (see Section 2);
- `eps0`, `rhos` and `ds`, which are volume fraction, density and dimension for each grain size. Pyroclasts are assumed to be perfectly spherical, hence their representative size `ds` will be their diameter;
- `Fr`, `g` and `rhoa`, that are Froude number, gravitational acceleration value and air density (at standard conditions of pressure and temperature);
- `alpha`, the *packing fraction*, i.e., the volume fraction of solid particles in the deposit;
- `flag_coords`, by which we can select the geometry (channelized *vs.* axisymmetric currents): in particular, we can solve the box model equations in cartesian (Eqns. (4)-(6)) or cylindrical (Eqns. (7)-(9)) coordinates, simply by setting `flag_coords` to values greater or lower or equal than zero, respectively;
- `flag_DEM`, by which we can choose whether to read a *Digital Elevation Model* (DEM, see subsection 4.2) or not. Indeed, as previously said, it is possible to analyze the behaviour of PDCs both over flat topographies and over topographies with reliefs and obstacles. We allow either of these cases, depending on the value of `flag_DEM`: if `flag_DEM` is set to `False`, we consider the flow runout in absence of topographies, while, if `flag_DEM` is `True`, we account for the presence of a complex topography. In this last case, we have to set other parameters, that are:
  - `vertical_scale_factor`, a scaling factor by which all the elevations registered in the DEM are multiplied. It must be consistent with the other data units;
  - `zone_number` and `zone_letter`, that identify the UTM (*Universal Transverse of Mercator*) grid zone the volcanic area is in;
  - `xv, yv`, the UTM coordinates of the volcanic vent location;
  - `rad_res`, i.e., a resolution for the generation of a grid along each radial direction, centered on the vent;
  - `anglemin`, `anglemax`, minimum and maximum angles defining a radial sector in which to study the collapse of an eruption column;
  - `differential_topography`, that can be set to `True` or `False`, depending on whether we want to consider the current dynamics with respect to elevations above sea level or differential topographies, or not.

[1] Indeed, the solver for the numerical solution of the differential system discretizes the equations with an adaptive time step technique (see subsection 4.3).

- `r_tol` and `a_tol`, relative and absolute tolerances for the control of the local error estimates when using *Runge-Kutta-Fehlberg method* to solve the differential problem (see subsection 4.3).

4.2 Operations on DEMs

Reading a DEM

As already said, if `flag_DEM` is set to `True`, we are provided with a DEM (that is, a digital representation of the elevation of a given territory) of a volcanic area.
Before proceeding, we want to spend a few words about DEM files. In particular, we have a DEM in `.asc` format: it is a text file with the following entries in the header.

```
ncols         1900
nrows         1502
xllcorner     418015.70
yllcorner     4514318.30
cellsize      10.00
NODATA_value  0
```

Looking at the rest of the file, it as a matrix of "pixels" (below, also "cells"), each "pixel" recording an elevation value above sea level. The first two entries of the `.asc` file header represent the number of columns and rows of this matrix, whereas `xllcorner` and `yllcorner` indicate the UTM coordinates of the lowermost-leftmost point[2] of the topographic domain. Besides, `cellsize` specifies the dimension (in meters) of each "pixel", while `NODATA_value`, in our case, establishes the elevation value for the cells at the sea level.

In the Python code, we define a subroutine `read_dem` for extracting information from a DEM in `.asc` format.
Then we define two vectors `xdem` and `ydem`, containing the coordinates of each "pixel" center, and a matrix `zdem` with cell elevations. Note that we read the elevation data from the upper-left component, so, since the UTM coordinates of each cell are based on the lower-left element ones, we need to flip the entries in each column in the up-down direction, with Python `numpy` command `flipud`.

Writing a DEM

The subroutine `write_dem` allows us to a create a new file and write to it the above mentioned entries of a DEM in `.asc` format.

[2]Not pixel!

Dividing a DEM into sectors

As we will explain later,[3] hazard quantification can be improved by dividing the volcanic area, centered on the vent, in N_s azimuthal sectors of central angle $\left(\frac{360}{N_s}\right)^\circ$.

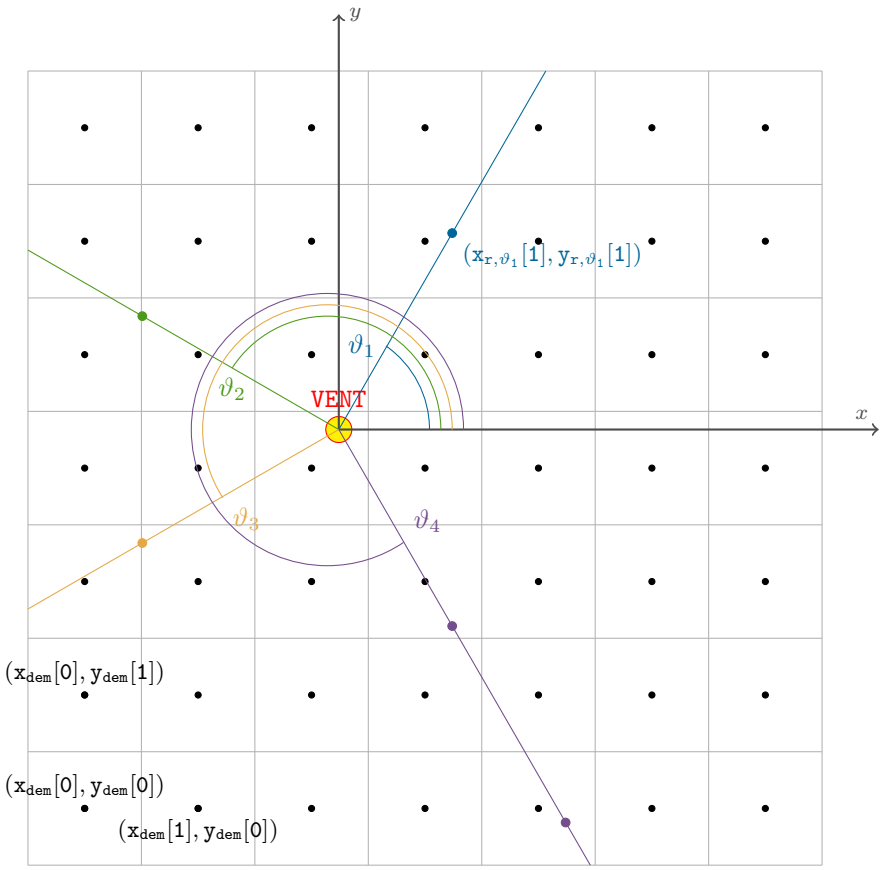


Figure 3: *Radial discretization of the space.*

In particular, we can divide each of these N_s radial directions in a certain number of subintervals of fixed width s_r (that is, the input parameter `rad_res`).
Assigned an angle ϑ, starting from the vent, we construct two vectors, $\mathbf{x}_{r,\vartheta}$ and $\mathbf{y}_{r,\vartheta}$, the first elements being the vent coordinates and the other ones being given by

$$\begin{aligned} x_{r,\vartheta}^{(i)} &= x_{r,\vartheta}^{(0)} + i \cdot s_r \cos(\vartheta) \\ y_{r,\vartheta}^{(i)} &= y_{r,\vartheta}^{(0)} + i \cdot s_r \sin(\vartheta), \quad i = 1, \ldots, n_{r_{\max}}, \end{aligned}$$

where $n_{r_{\max}}$ is the maximum number of grid points in this new system (see Figure 3). Naturally, it may happen that the nodes such created exceed the limits of the DEM or

[3] See subsection 4.5.

are more than the maximum flow runout away from the vent, in which cases the grid construction is stopped.
Then, for each point of the new grid, we determine the "pixel" of the initial DEM it is in. In this way, we can find the elevation $z_{r,\vartheta}^{(i)}$ of the $i-$th node, $i = 0, \ldots, n_{r_{\max}}$.

The procedure just designed is implemented in the Python subroutine `dem_section`, in which $\mathbf{x}_{r,\vartheta}$, $\mathbf{y}_{r,\vartheta}$, $\mathbf{z}_{r,\vartheta}$, $n_{r_{\max}}$ and s_r are respectively labeled `x`, `y`, `z`, `nmax` and `d`. As we can see from the code, when looking for the "pixel" containing a certain node, we distinguish between four cases, according to the quadrant the angle ϑ terminates in.

4.3 Numerical integration

The set of equations (4)-(6) (or (7)-(9)) is numerically integrated by using *Runge-Kutta-Fehlberg method* (below, also RKF45).

Explicit $s-$stages Runge-Kutta methods

Let us consider the problem of numerically solving the first order system of Ordinary Differential Equations (ODEs)

$$\begin{cases} \dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in (t_0, t_{\max}] \\ \mathbf{y}(t_0) = \mathbf{y}_0, \end{cases} \tag{10}$$

where $\mathbf{y}(t) \in \mathbb{R}^m$, $t \in [t_0, t_{\max}]$, and $\mathbf{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$.[4]

Recall that, in the most general form, all the *explicit s-stages Runge-Kutta methods* for the numerical solution of the initial value problem (below, IVP) (10) can be written as (Dormand and Prince, 1980; Atkinson, 1989; Lambert, 1992; Gautschi, 1997)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + (\Delta t)_n \, \mathbf{F}(t_n, \mathbf{y}_n, (\Delta t)_n ; \mathbf{f}), \quad n = 0, \ldots, N_{\max} - 1, \tag{11}$$

where, once constructed a discrete mesh $\{t_i\}_{i=0,\ldots,N_{\max}}$ in the interval $[t_0, t_{\max}]$, $\mathbf{y}_n$ represents an approximation to the values $\mathbf{y}(t_n)$ of the solution at the grid points, i.e., $\mathbf{y}_n \approx \mathbf{y}(t_n)$. Furthermore, $(\Delta t)_n$ is the current time step, that is, $t_{n+1} = t_n + (\Delta t)_n$, and $\mathbf{F}$ is a function defined as follows:

$$\mathbf{F}(t_n, \mathbf{y}_n, (\Delta t)_n ; \mathbf{f}) = \sum_{i=1}^{s} \gamma_i \mathbf{V}_i, \; \mathbf{V}_i = \begin{cases} \mathbf{f}(t_n, \mathbf{y}_n), \; i = 1, \\ \mathbf{f}\left(t_n + \alpha_i \, (\Delta t)_n , \mathbf{y}_n + (\Delta t)_n \sum_{j=1}^{i-1} \beta_{ij} \mathbf{V}_j \right), \; i > 1. \end{cases}$$

Coefficients $\{\beta_{ij}\}$, $\{\alpha_i\}$ and $\{\gamma_i\}$ completely characterize a Runge-Kutta method and are arranged in the so-called *Butcher tableau*

$$\begin{array}{c|c} \boldsymbol{\alpha} & \beta \\ \hline & \boldsymbol{\gamma}^t \end{array}$$

[4]Here we suppose that all the hypotheses for the local existence and uniqueness of the solution hold.

We impose the conditions (Gautschi, 1997)

$$\sum_{i=1}^{s} \gamma_i = 1 \quad \text{and} \quad \alpha_i = \sum_{j=1}^{i-1} \beta_{ij}, i = 2, \ldots, s.$$

We also recall that an explicit s-stages Runge-Kutta method cannot have order of accuracy greater than s.[5] Moreover, we remark that a Runge-Kutta method of order s for a scalar IVP may have order less than s when applied to solve a system of ODEs (Lambert, 1992).

Herein, we do not deal with the stability analysis of explicit Runge-Kutta schemes. It is worth mentioning that explicit A-stable Runge-Kutta methods do not exist, although the stability regions become larger, as the order increases (Lambert, 1992).

Embedded Runge-Kutta methods

Since Runge-Kutta methods are one-step, they are particularly suitable for changing step size, as long as we are provided with an acceptable estimator of the local truncation error (12), introduced in the single integration step.
This aim can be achieved by taking two $s-$stages Runge-Kutta methods of orders p and q, $q > p$ (usually $q = p + 1$), and applying them simultaneously to the same problem. Moreover, in order to cut down the computational effort, we impose that the two schemes share the same function evaluations (i.e., have same coefficients α_i, β_{ij}).
So, at each step, we can express the two approximations for the solution as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + (\Delta t)_n \sum_{i=1}^{s} \gamma_i \mathbf{V}_i,$$

$$\bar{\mathbf{y}}_{n+1} = \mathbf{y}_n + (\Delta t)_n \sum_{i=1}^{s} \bar{\gamma}_i \mathbf{V}_i,$$

and the Butcher tableau can be summarized as

$\boldsymbol{\alpha}$	β
	$\boldsymbol{\gamma}^t$
	$\bar{\boldsymbol{\gamma}}^t$

[5] A numerical scheme for solving IVPs is said to be *$p-$th order accurate* if, once defined the *local truncation error* $\boldsymbol{\tau}_{n+1}$ of (11) as the residual

$$\boldsymbol{\tau}_{n+1} = \mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) - (\Delta t)_n \, \mathbf{F}(t_n, \mathbf{y}(t_n), (\Delta t)_n ; \mathbf{f}), \quad n = 0, \ldots, N_{\max} - 1, \tag{12}$$

p is the largest integer such that $\boldsymbol{\tau}_{n+1} = \mathcal{O}\left((\Delta t)_n^{p+1}\right)$ (Dormand and Prince, 1980; Lambert, 1992; Brugnano and Trigiante, 1998).
Otherwise said, $\boldsymbol{\tau}_{n+1}$ represents the error introduced in the single integration step, in the local hypothesis $\mathbf{y}_n = \mathbf{y}(t_n)$.

By definition (see note 9), we have that

$$\boldsymbol{\tau}_{n+1} = \mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1} = \mathcal{O}\left((\Delta t)_n^{p+1}\right)$$
$$\bar{\boldsymbol{\tau}}_{n+1} = \mathbf{y}(t_{n+1}) - \bar{\mathbf{y}}_{n+1} = \mathcal{O}\left((\Delta t)_n^{q+1}\right),$$

hence, by subtracting the two above relationships, we get

$$\boldsymbol{\tau}_{n+1} \approx \boldsymbol{\omega}_{n+1} \stackrel{\text{def}}{=} \bar{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1} = (\Delta t)_n \sum_{i=1}^{s} (\bar{\gamma}_i - \gamma_i)\mathbf{V}_i \stackrel{\text{def}}{=} (\Delta t)_n \sum_{i=1}^{s} c_i \mathbf{V}_i.$$

Step control. Now, the idea is the following: if the two numerical solutions are in close agreement, the approximation $\mathbf{y}_{n+1}$ is accepted, whereas, if their difference exceeds a specified accuracy, the step size is reduced. In particular, we impose that (Atkinson, 1989)

$$\|\boldsymbol{\tau}_{n+1}\|_\infty \leq \mathsf{tol}_\mathsf{a} + \|\mathbf{y}_n\|_\infty \mathsf{tol}_\mathsf{r} \stackrel{\text{def}}{=} \varepsilon_{n+1}.$$

In practice, starting from an initial guess $(\Delta t)_0$, we compute $\mathbf{y}_1$, $\bar{\mathbf{y}}_1$ and hence the estimator $\boldsymbol{\omega}_1$ for $\boldsymbol{\tau}_1$. Suppose the aforementioned test has not passed, so we have to reduce $(\Delta t)_0$ ($(\Delta t)_n$, for the generic step). How this can be done?
Assume that the local truncation error can be written as

$$\boldsymbol{\tau}_{n+1} = \mathbf{T}(t_n, \mathbf{y}(t_n))\,(\Delta t)_n^{p+1} + \mathcal{O}\left((\Delta t)_n^{p+2}\right)$$

where the function $\mathbf{T}(t, \mathbf{y}(t))$ is called the *principal error function*.
Thus, considering a smaller step size, i.e., $(\Delta t)_n \to (\Delta t)_n^{\text{new}} = \zeta\,(\Delta t)_n$, $\zeta < 1$, we have

$$\boldsymbol{\tau}_{n+1}^{\text{new}} \approx \mathbf{T}(t_n, \mathbf{y}(t_n))\zeta^{p+1}\,(\Delta t)_n^{p+1} = \zeta^{p+1}\boldsymbol{\omega}_{n+1}^{\text{old}}$$

and we force

$$\|\zeta^{p+1}\boldsymbol{\omega}_{n+1}^{\text{old}}\|_\infty \leq \varepsilon_{n+1} \Rightarrow \zeta \leq \left(\frac{\varepsilon_{n+1}}{\|\boldsymbol{\omega}_{n+1}^{\text{old}}\|_\infty}\right)^{\frac{1}{p+1}} < 1.$$

Set a *safety factor* $\mu \in (0, 1)$,[6] we estimate $(\Delta t)_n^{\text{new}}$ as

$$(\Delta t)_n^{\text{new}} := (\Delta t)_n = \mu \left(\frac{\varepsilon_{n+1}}{\|\boldsymbol{\omega}_{n+1}^{\text{old}}\|_\infty}\right)^{\frac{1}{p+1}} (\Delta t)_n$$

and accept the solution $\mathbf{y}_{n+1}^{\text{new}}$ (obtained with this new time step) as $\mathbf{y}_{n+1}$. It is recommended to control whether or not the requested accuracy has actually been reached, anyway.

[6] We put it since our estimation of the local truncation error is not exact.

Once admitted the time step $(\Delta t)_n$, since $\|\boldsymbol{\omega}_{n+1}\|_\infty < \varepsilon_{n+1}$, we can try to estimate $(\Delta t)_{n+1}$ as

$$(\Delta t)_{n+1} = \underbrace{\left(\frac{\varepsilon_{n+1}}{\|\boldsymbol{\omega}_{n+1}\|_\infty}\right)^{\frac{1}{p+1}}}_{>1} (\Delta t)_n \,,$$

i.e., the step size is increased, and we repeat, for the current time, the tests above described for the previous one.

RKF45

As already said, we perform our numerical integrations with RKF45, that embeds Runge-Kutta formulas of order 4 and 5: it controls the local truncation errors assuming accuracy of the fourth-order method and taking the steps with the fifth-order accurate scheme.
RKF45 Butcher tableau, according to one of the *Dormand-Prince formulas*,[7] is shown below (Dormand and Prince, 1980):[8]

$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$-\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

[7] Dormand-Prince formulas are a class of Runge-Kutta schemes of order 4 and 5, first derived in Dormand and Prince (1980). Herein, we make use of one of these formulas.

[8] This scheme is quite similar to the RKF45 method implemented in Python function `scipy.integrate.solve_IVP`. Proper modifications with respect to the Butcher tableau here represented are introduced, in accordance with Shampine (1986).

Following the notation from this subsection, we have to apply RKF45 to numerically solve Eqns. (4)-(6) or (7)-(9), with

$$\mathbf{f}(t,\mathbf{y}) = \begin{pmatrix} \mathsf{Fr}\sqrt{g'(y_2,\ldots,y_{N+1})\dfrac{V_0}{y_1}} \\ -\dfrac{w_{s,1}y_2y_1}{V_0} \\ \vdots \\ -\dfrac{w_{s,N}y_{N+1}y_1}{V_0} \end{pmatrix} \quad \text{or} \quad \mathbf{f}(t,\mathbf{y}) = \begin{pmatrix} \mathsf{Fr}\sqrt{g'(y_2,\ldots,y_{N+1})\dfrac{V_0}{\xi\pi}}\dfrac{1}{y_1} \\ -\dfrac{\xi\pi w_{s,1}y_2y_1^2}{V_0} \\ \vdots \\ -\dfrac{\xi\pi w_{s,N}y_{N+1}y_1^2}{V_0} \end{pmatrix},$$

depending on whether `flag_coords` is greater than zero or not. In the code, we define proper subroutines for computing these terms and call them `fun_cartesian` and `fun_cylindrical`, respectively.

Python function `scipy.integrate.solve_IVP`. We solve the box model equations with the already existing function `scipy.integrate.solve_IVP`, available in Python-3.x,[9] which makes use of a variant of the Dormand-Prince formulas (Dormand and Prince, 1980), discussed in Shampine (1986).
For the numerical solution of the generic problem (10), the basic syntax is `sol = scipy.integrate.solve_IVP(fun, tspan, y0, method, t_eval, events, rtol, atol)`, where `fun` is the r.h.s. of the system, `tspan` and `y0` are respectively the interval of integration[10] and the initial condition, `method` indicates the integration scheme (`'RK45'`,[11] in our case), while `t_eval` specifies the times at which to store the computed solution[12] and the `event` entry allows to stop the integration process if a specific condition occurs. In our case, the integration process is blocked when the reduced gravity g' falls below zero. Finally, tolerances $\mathsf{tol_a}$ and $\mathsf{tol_r}$ (above referred to as `atol` and `rtol`) are set to `a_tol` and `r_tol` values, specified as input by the user.

[9] Please refer to *SciPy v1.2.1 Reference Guide* (2019), available at `https://docs.scipy.org/doc/scipy/reference/index.html`.

[10] The maximum run time $t_{\max}$ is estimated as

$$t_{\max} = -\frac{h_0}{\min\limits_{i=1,\ldots,N} w_i^s}\log\left(\frac{\rho_{\text{atm}}-\rho_{\text{g}}}{\sum_{i=1}^N \epsilon_{0,i}(\rho_i^s-\rho_{\text{g}})}\right) \quad \text{or} \quad t_{\max} = -\frac{h_0}{\min\limits_{i=1,\ldots,N} w_i^s}\log\left(\frac{\sum_{i=1}^N \epsilon_{0,i}\rho_{\text{atm}}}{\sum_{i=1}^N \epsilon_{0,i}\rho_i^s}\right),$$

depending on whether a buoyant interstitial gas is present or not, as follows by equating ρ_{c} and ρ_{atm} (since, when g' falls below zero, the flow stops). In order to obtain the estimates above, in the equations (6), (9) we have replaced the ratios $\dfrac{w_i^s}{h(t)}$ with the (lower) quantities $\dfrac{\min\limits_{i=1,\ldots,N} w_i^s}{h_0}$, leading to an overestimation of the effective $t_{\max}$.

[11] `'RK45'`, that is, RKF45, is employed by default.

[12] From the discussion on the step control mechanism, we note that the vector of time output may not have any "regularity". For example, we could desire the approximated solution on a more refined grid or on uniform meshes. This can be achieved by setting up a proper `t_eval` entry. In these cases, more dense (or, in general, different) output can be produced by interpolation (Shampine, 1986).

4.4 Subroutines for computing physical quantities

Computing density of a polydisperse mixture

As we can see in the Python code, we enable the possibility of considering polydisperse mixtures.
In particular, in the subroutine `polydisperse_density`, we compute the density of the polydisperse current, according to (1).

Computing reduced gravity of a polydisperse mixture

Similarly, referring to (2), we determine the reduced gravity of the polydisperse mixture by means of the `polydisperse_rg` subroutine. In the code, we respectively call `rg` and `phi[i]` the quantities

$$\frac{\rho_{\mathrm{g}} - \rho_{\mathrm{atm}}}{\rho_{\mathrm{atm}}} \quad \text{and} \quad \frac{\rho_i^s - \rho_{\mathrm{g}}}{\rho_{\mathrm{atm}}}.$$

Estimating terminal velocities

We determine each particle class terminal velocity with the `settling` subroutine.
In particular, once denoted the grain-size class diameters by $d_{s,i}$, $i = 1, \ldots, N$, the asymptotic, stationary settling velocities are calculated by means of the *Newton's impact formula* (Dellino et al., 2005; Dioguardi et al., 2018)

$$w_i^s = \sqrt{\frac{4 d_{s,i} \rho_i^s}{3 C_{\mathrm{D}_i} \rho_{\mathrm{g}}} g}, \quad i = 1, \ldots, N, \tag{13}$$

where the *gas-particle drag coefficient* C_{D_i} is defined, as a function of the *relative gas-particle Reynolds number* Re_{r_i}, $\mathsf{Re}_{r_i} = \dfrac{\rho_{\mathrm{g}} d_{s,i} w_i^s}{\mu_g}$, μ_g being the interstitial fluid dynamic viscosity,[13] by the following expression

$$C_{\mathrm{D}_i} = \begin{cases} \dfrac{24}{\mathsf{Re}_{r_i}} \left(1 + 0.15 \cdot \mathsf{Re}_{r_i}^{0.687}\right) & \text{if } \mathsf{Re}_{r_i} < 1000, \\ 1 & \text{otherwise.} \end{cases}$$

We have therefore used the *Schiller-Naumann correlation* (Crowe et al., 2011), which accurately describes the drag force acting on a sphere up to $\mathsf{Re}_{r_i} \simeq 1000$, whereas, for $\mathsf{Re}_{r_i} \geq 1000$, we have set $C_{\mathrm{D}_i} = 1$, according to Woods and Bursik (1991).

[13] Dynamic viscosity as a function of temperature is computed according to *Sutherland's law* for ideal gases (Sutherland, 1893), which states that

$$\mu(\vartheta) = \frac{C_1 \vartheta^{\frac{3}{2}}}{\vartheta + C_2},$$

ϑ being the absolute temperature of the gas, $C_1 = 1.458 \cdot 10^{-6}\ kg \cdot \left(m \cdot s \cdot K^{\frac{1}{2}}\right)^{-1}$, $C_2 = 110.4\ K$.

As we can see, the computation of settling velocities requires an iterative procedure: in fact, the Newton's impact formula (13) must be solved together with the relationship for the Reynolds number and the correlation between C_{D_i} and Re_{r_i}.
Hence, starting from an initial guess $w_i^{s(0)}$,[14] we determine the corresponding Reynolds number, which is further used to find settling velocity by (13); a new Reynolds number is then calculated and the process is continued until the solution converges[15] or iterations exceed a maximum number, be it `kmax`.

Computing deposits

For each particle class, we can compute the amount of mass loss by sedimentation, per unit area, per time step, as

$$m_i(t) = w_i^s \rho_i^s \epsilon_i(t) \Delta t, \quad i = 1, \ldots, N,$$

Δt being the time step.
Thus, for a given time $\bar{t}$ and the corresponding current front position $l(\bar{t})$, we determine the total mass deposited by the PDC in that point as the sum of $m_i(t)$, for all i, for all time $t \in [\bar{t}, t_{\max}]$, where $t_{\max}$ is the final run time, as usual.[16] Similarly, by keeping i, we can compute the deposited total mass of the $i-$th particle class, hence the deposited mass fraction of that class, at distance $l(\bar{t})$.
At last, from the previous quantities, we deduce the thickness profile of the $i-$th class deposited layer as the ratio of the deposited total mass of that class to the $i-$th class solid density (multiplied by the packing fraction, above referred to as `alpha`, see 4.1).
In the code, the vector $\mathbf{m}$ is called `deposit`, whereas the other quantities above introduced are named `totalmass`, `mass`, `fract` and `thick`, respectively.

4.5 Writing invasion maps

Before building invasion maps, we express $h_{\max}$ in terms of l: starting from the initial front position l_0, we thus determine the value of $h_{\max}$ corresponding to all the possible integer current lengths up to the maximum flow runout.
For this purpose, in the code we construct a proper vector named `hmax_of_l`, in which the evolution of $h_{\max}$ as a (discrete) function of l is recorded.

With a view to define invasion maps, we take each "pixel" of the original DEM and build a new DEM, in which "pixels" do not register elevation data but binary values, 1 or 0, depending on whether the cell has been invaded or not by the current.

[14] In the code, the initial guess is provided by the *Stokes free-fall velocity formula*, i.e.,

$$w_i^s = \frac{d_{s,i} \rho_i^s}{18 \mu_g} g,$$

which has remarkable experimental confirmations within the regime $\mathsf{Re}_{r_i} < 1$.

[15] We mean that the difference between two consecutive iterates is within a `tol` tolerance.

[16] This choice is due to the fact that, before $t = \bar{t}$, the current has not yet arrived to the position $l(\bar{t})$.

Energy-conoid model: method 1

A first, simple way to construct invasion maps is described below.
We set a vent location, then we compute the distance `dist` between the vent and each point (`xdem[i], ydem[j]`) of the original DEM. So, we deduce the value of $h_{\max}$ corresponding to the nearest integer to `dist` and compare it with the elevation of the cell; we also compare `dist` with the maximum flow runout. If `dist` exceeds the maximum runout or the "pixel" elevation is greater than $h_{\max}$, that cell is assumed not to be invaded.

Energy-conoid model: method 2

Nevertheless, the approach above described may lead to unphysical invasion regions in the lee of topographical barriers (see Fig. 4(a)).
For this reason, in order to improve hazard assessment, in the case of radially spreading currents, we can divide the 360-azimuth volcanic area, centered on the vent, in a certain number of thin circular sectors, be it N_s (typically, $N_s = 360$, see subsection 4.2), and compute the energy-conoid solution along all the N_s sectors.

Step 1. By using the function `dem_section`, defined in subsection 4.2, for each angle ϑ we determine the elevation of the grid points along the direction specified by ϑ. Then, we check if their elevation is greater than the value of $h_{\max}$ relative to the closest integer to the distance from the vent. As soon as this condition is satisfied, the flow is stopped in that direction and the maximum traveled distance is stored.

Step 2. Once done this, we compute the distance `dist` between the vent and each point (`xdem[i], ydem[j]`) of the original DEM. Then, we determine which direction (starting from the vent) the point is in, hence the angle corresponding to this direction.

Step 3. Since we consider currents spreading within a circular sector of central angle θ, we take every DEM point lying in this sector and, if its distance from the vent is lower than the maximum runout in the corresponding direction (as determined above), the cell containing the point is assumed to be invaded.
It is worth noting that, for both the approaches, the model does not allow a partial block of the current: a cell cannot be partially invaded.

The difference between these two approaches is shown in Figure 4: under the same initial conditions, the invasion maps obtained via energy-conoid method 1 (a) and energy-conoid method 2 (b) are extremely dissimilar.

Currents spreading on slopes

When investigating the current flow on complex topographies, we have to take into account that the flow may start from positive elevation or encounter upward slopes after

downward slopes.[17] In this case, enabled by setting `differential_topography` input parameter to `True`, it is reasonable to compare $h_{\max}$ at a given distance from the vent not with the corresponding topographic elevation but with the difference in level experienced by the current between the previous and the present sampled positions.
Figure 5 compares a PDC propagation in the Mt. Vesuvius volcanic region (Italy), with `differential_topography = True` (a) and `differential_topography = False` (b). Note that in the first case the invaded area is greater: this is due to the fact that, in proximity to the vent, the $h_{\max}$ required to completely stop the current is higher than in the second case.

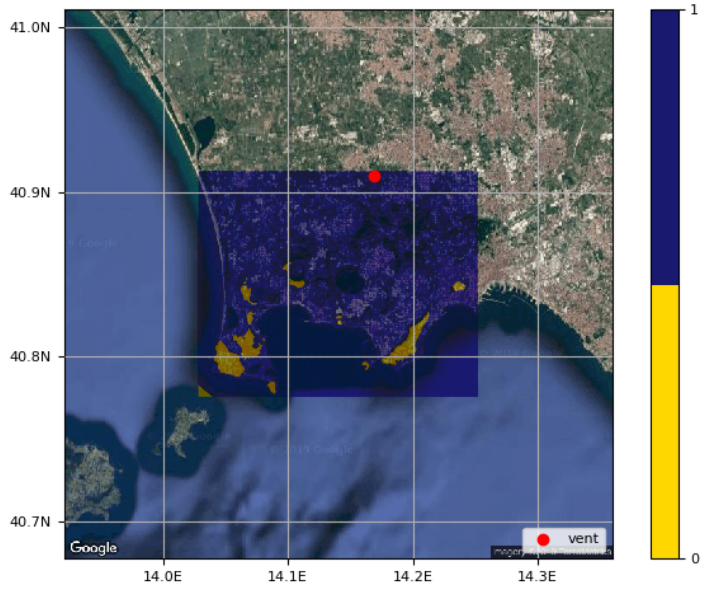
4.6 Visualizing invasion maps

Finally, we included a plotting tool in the code.[18]
In particular, after having found longitude and latitude coordinates of the DEM limits and the vent via Python module `utm`, we download the map of the corresponding zone from the Google Maps Static API. The Maps Static API service returns the map as an image we can display. Then, using `salem` module,[19] we can reconstruct on the image such obtained the grid of the original DEM and hence add and plot our georeferenced data about the invasion.
Results of this process are shown in Figures 4 and 5, where each pixel of the original DEM (easily distinguishable from the rest of the region) is colored blue or yellow, depending on whether it is invaded or not. Moreover, the vent is marked as a red spot on the map.

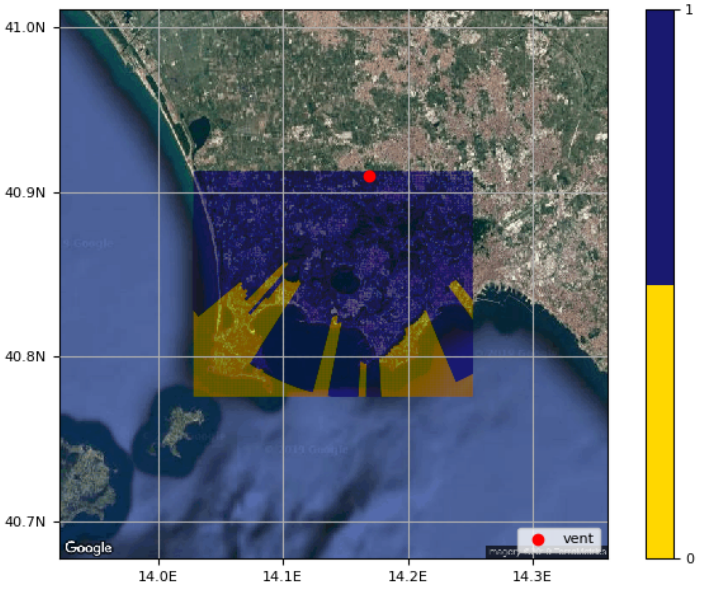
[17] In calderas, where the mean topographic slope is about zero, we can make use of the procedure just described.

[18] Although both the previous approaches were present in the original version of the code, the first one has been removed, since we actually visualize invasion maps defined only by energy-conoid method 2.

[19] See `https://salem.readthedocs.io/en/v0.2.3/`.

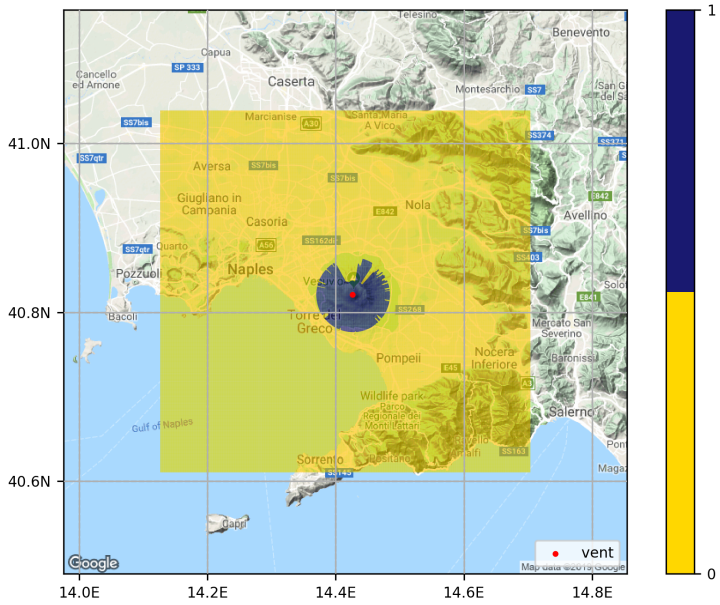


(a)

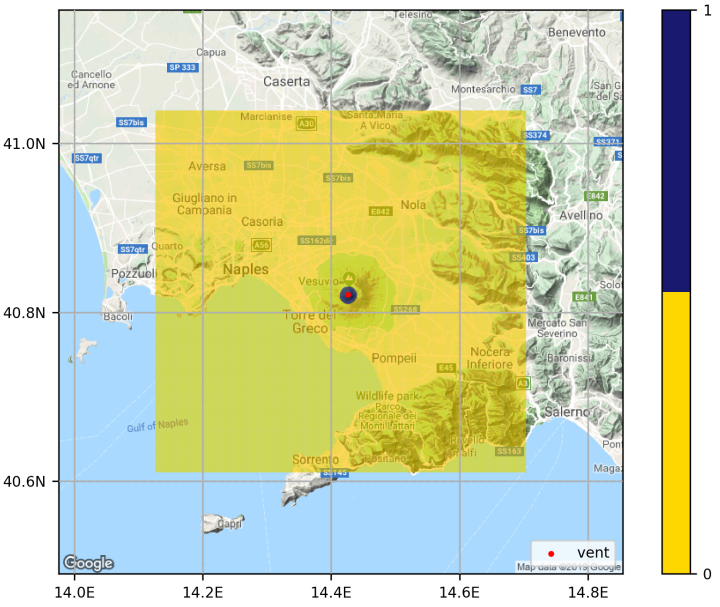


(b)

Figure 4: *Comparison between the two approaches, outlined in subsection 4.5, to obtain invasion maps, with regard to the numerical simulation of a PDC propagation at Campi Flegrei Caldera, Italy. In the second case, some topographical reliefs are able to completely shield the rear zones against the current.*



(a)



(b)

Figure 5: *Comparison of the invasion maps achieved with regard to the numerical simulation of a PDC propagation at Mt. Vesuvius area (Italy), by using the two boolean values allowable for* `differential_topography`*. Note the differences near the vent.*

References

Atkinson, K. E. (1989). *An Introduction to Numerical Analysis*. John Wiley and Sons, 2nd edition.

Benjamin, T. B. (1968). Gravity currents and related phenomena. *Journal of Fluid Mechanics*, 31(2):209–248.

Bevilacqua, A. (2016). Doubly stochastic models for volcanic hazard assessment at Campi Flegrei caldera. 21:227.

Bevilacqua, A. (2019). Notes on the analytic solution of box model equations for gravity-driven particle currents with constant volume. Technical report.

Bevilacqua, A., Neri, A., Bisson, M., Esposti Ongaro, T., Flandoli, F., Isaia, R., Rosi, M., and Vitale, S. (2017). The Effects of Vent Location, Event Scale, and Time Forecasts on Pyroclastic Density Current Hazard Maps at Campi Flegrei Caldera (Italy). *Frontiers in Earth Science*, 5:1–16.

Bonnecaze, R. T., Hallworth, M. A., Huppert, H. E., and Lister, J. R. (1995). Axisymmetric particle-driven gravity currents. *Journal of Fluid Mechanics*, 294:93–121.

Brugnano, L. and Trigiante, D. (1998). *Solving Differential Problems by Multistep Initial and Boundary Value Methods*. Gordon and Breach Science Publishers.

Crowe, C. T., Schwarzkopf, J. D., Sommerfeld, M., and Tsuji, Y. (2011). *Multiphase Flows With Droplets and Particles*. CRC Press, 2nd edition.

Dade, W. B. and Huppert, H. E. (1995a). A box model for non-entraining, suspension-driven gravity surges on horizontal surfaces. *Sedimentology*, 42(3):453–470.

Dade, W. B. and Huppert, H. E. (1995b). Runout and fine-sediment deposits of axisymmetric turbidity currents. *Journal of Geophysical Research*, 100:18597–18609.

Dade, W. B. and Huppert, H. E. (1996). Emplacement of the Taupo ignimbrite by a dilute turbulent flow. *Nature*, 381(6582):509–512.

Dellino, P., Mele, D., Bonasia, R., Braia, G., La Volpe, L., and Sulpizio, R. (2005). The analysis of the influence of pumice shape on its terminal velocity. *Geophysical Research Letters*, 32(21).

Dioguardi, F., Mele, D., and Dellino, P. (2018). A New One-Equation Model of Fluid Drag for Irregularly Shaped Particles Valid Over a Wide Range of Reynolds Number. *Journal of Geophysical Research: Solid Earth*, 123(1):144–156.

Dormand, J. R. and Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

Dufek, J., Esposti Ongaro, T., and Roche, O. (2015). Chapter 35 - Pyroclastic Density Currents: Processes and Models. In Sigurdsson, H., editor, *The Encyclopedia of Volcanoes (Second Edition)*, pages 617–629. Academic Press, 2nd edition.

Esposti Ongaro, T., Orsucci, S., and Cornolti, F. (2016). A fast, calibrated model for pyroclastic density currents kinematics and hazard. *Journal of Volcanology and Geothermal Research*, 327:257–272.

Gautschi, W. (1997). *Numerical Analysis: An Introduction*. Birkhäuser.

Hallworth, M. A., Hogg, A. J., and Huppert, H. E. (1998). Effects of external flow on compositional and particle gravity currents. *Journal of Fluid Mechanics*, 359:109–142.

Huppert, H. E. and Simpson, J. E. (1980). The slumping of gravity currents. *Journal of Fluid Mechanics*, 99(4):785–799.

Lambert, J. (1992). *Numerical methods for Ordinary Differential Systems*. John Wiley and Sons.

Neri, A., Bevilacqua, A., Esposti Ongaro, T., Isaia, R., Aspinall, W. P., Bisson, M., Flandoli, F., Baxter, P. J., Bertagnini, A., Iannuzzi, E., Orsucci, S., Pistolesi, M., Rosi, M., and Vitale, S. (2015). Quantifying volcanic hazard at Campi Flegrei caldera (Italy) with uncertainty assessment: 2. Pyroclastic density current invasion maps. *Journal of Geophysical Research: Solid Earth*, 120(4):2330–2349. 2014JB011776.

Orsucci, S. (2014). *Multiphase flow modeling and numerical simulation of pyroclastic density currents*. PhD thesis, Università di Pisa.

Roche, O., Buesch, D. C., and Valentine, G. A. (2016). Slow-moving and far-travelled dense pyroclastic flows during the Peach Spring super-eruption. *Nature Communications*, 7.

Roche, O., Phillips, J. C., and Kelfoun, K. (2013). Pyroclastic density currents. In Fagents, S. A., Gregg, T. K. P., and Lopes, R. M. C., editors, *Modeling Volcanic Processes: The Physics and Mathematics of Volcanism*, pages 203–229. Cambridge University Press.

Shampine, L. F. (1986). Some Practical Runge-Kutta Formulas. *Mathematics of Computation*, 46(173):135–150.

Sutherland, W. (1893). LII. The viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(223):507–531.

Woods, A. W. and Bursik, M. I. (1991). Particle fallout, thermal disequilibrium and volcanic plumes. *Bulletin of Volcanology*, 53(7):559–570.