Parallel Simulation of Particle Transport in an Advection Field Applied to Volcanic Explosive Eruptions

Pierre Künzli^a, Kae Tsunematsu^{b,c}, Paul Albuquerque^d, Jean-Luc Falcone^a, Bastien Chopard^a,
Costanza Bonadonna^b

^aComputer Science Department, University of Geneva, Carouge, Switzerland

^bDepartment of Earth Sciences, University of Geneva, Geneva, Switzerland

^cMount Fuji Research Institute, Yamanashi, Japan

^dInstitute inIT, hepia, University of Applied Sciences and Arts Western Switzerland, Geneva, Switzerland

Abstract

Volcanic ash transport and dispersal models typically describe particle motion via a turbulent velocity field. Particles are advected inside this field from the moment they leave the vent of the volcano until they deposit on the ground. Several techniques exist to simulate particles in an advection field such as finite difference Eulerian, Lagrangian-puff or pure Lagrangian techniques. In this paper, we present a new flexible simulation tool called TETRAS (TEphra TRANsport Simulator) based on a hybrid Eulerian-Lagrangian model. This scheme offers the advantages of being numerically stable with no numerical diffusion and easily parallelizable. It also allows to output particle atmospheric concentration or ground mass load at any given time. The model is validated using the advection-diffusion analytical equation. We also obtained a good agreement with field observations of the tephra deposit associated with the 2450 BP Pululagua (Ecuador) and the 1996 Ruapehu (New Zealand) eruptions. As this kind of model can lead to computationally intensive simulations, a parallelization on a distributed memory architecture was developed. A related performance model, taking into account load imbalance, is proposed and its accuracy tested.

Keywords: Explosive volcanism, Volcanic plumes, Volcanic ash transport, Tephra, MPI, Parallelization

1. Introduction

Explosive volcanic eruptions are typically associated with the injection into the atmosphere of large amounts of particles (known as tephra) dispersed over long distances depending on size, shape and density. Accumulation at ground level of few millimeters to tens of centimeters of tephra can cause a wide range of hazards including collapse of roofs, damage to crops, killing of many grazing animals, contamination of water supplies, disruption of electricity and telecommunication networks and perturbation of ground transportation, while fine volcanic ash ($< 63[\mu\text{m}]$) significantly threatens aviation operations and human health ([1], [2], [3], [4], [5], [6]). Consequently, even though other volcanic phenomena are more likely to directly endanger human life (e.g. pyroclastic density currents and lahars), tephra dispersal and sedimentation can seriously impact entire economic sectors and disrupt critical infrastructure services, as demonstrated by the recent eruptions of Eyjafjallajökull (Iceland; 2010 [7]) and Cordon Caulle (Chile; 2011 [8]) volcanoes.

Email address: pierre.kuenzli@unige.ch (Pierre Künzli)

During the past decades, several Volcanic Ash Transport and Dispersal Models (VATDMs) have been developed with variable levels of complexity and different objectives mostly including: i) a better understanding of particle transport and sedimentation dynamics; ii) the compilation of real-time and long term hazard assessment associated with both ground load and atmospheric concentrations of ash; iii) and the determination of eruption source parameters through inversion strategies (see [9], [10], [11] for a review).

Most existing VATDMs are either Eulerian finite difference schemes such as FALL3D ([12], [13]) or Lagrangian particle-puff schemes such as VOL-CALPUFF ([14], [15]) or NAME III ([16]). Some are pure Lagrangian models such as PUFF ([17]).

Pure Lagrangian models dealing with point particles have the advantage of allowing the implementation of individual particle behaviour as well as particle interaction. Moreover, they remain unconditionally stable, unlike for example finite difference schemes, and allow to monitor ash concentration in the atmosphere as well as ground mass load at any given time. The drawback is that such representations require the simulation of large numbers of particles to yield statistically accurate results. This may thus produce very computationally intensive simulations. Although the increasing computational capacities of computers and supercomputers tend to make these simulations affordable, parallelization still remains essential to run them on large numbers of computing cores. However, pure Lagrangian models are difficult to implement on massively multicore systems when particle interactions are involved. This is why we propose an implementation based on a hybrid model.

In this paper, we present a new hybrid Eulerian-Lagrangian model for the dispersal and sedimentation of tephra with an emphasis on computational efficiency and model flexibility. Computational efficiency is particularly important for both real-time forecasting and long-term hazard assessments. In our approach, particles remain linked to a site and their exact positions are continuously tracked. This allows for an accurate treatment of diffusion while maintaining a known maximum spatial integration step. The associated data structure facilitates a parallel implementation as well as the addition of complex sedimentation processes, such as particle aggregation ([18], [19]). The implementation was designed with a modular approach to permit modification or addition of components to the physical model. It is then possible to use our simulation tool both for the study of the dynamics of particle dispersion and as a prediction tool for hazard assessment. As previously said, our model allows to easily monitor both atmospheric ash concentration and ground mass load. In this paper, we only focus on the latter.

2. Physical model description

During explosive volcanic eruptions, a hot mixture of particles and volcanic gases is typically ejected with an initial density several times larger than in the atmosphere, and rises due to momentum. As the ejected material entrains ambient air, the mixture density drastically decreases and the eruptive plume starts rising due to buoyancy. If the plume upward velocity is much larger than the horizontal wind velocity (strong plume, Figure 1a), the volcanic plume buoyantly rises up to the neutral buoyancy level (H_b) where it starts spreading laterally as a gravity current (umbrella cloud). In contrast, if the horizontal wind velocity dominates, the plume bends over above the basal jet before spreading laterally around the neutral buoyancy level (weak plume, Figure 1b) ([21], [22]). In all cases, cloud spreading results from the interplay between buoyancy and wind advection, with the contribution of buoyancy being proportional to plume height ([21], [23], [24]).

Figure 1: Representation of the main features of (a) a strong plume and (b) a weak plume (adjusted from [20]). H_b : Neutral buoyancy level; H_t : total plume height.

The physics of plume dynamics and rise is described in [25]. The model main parameters include the wind speed at the tropopause, the initial plume velocity U_0 , temperature θ_0 and radius r_0 , and the initial mass fraction of exsolved volatiles n_0 . The model then outputs the mass eruption rate and plume profile by giving the position, angle with horizontal, radius of the plume and speed at each point of the plume. The total erupted mass is derived directly from the mass eruption rate and the eruptive events durations. Depending on their size and density, particles are transported upward by the volcanic plume and, if sufficiently small, might be entrained within the umbrella cloud and sediment according to their terminal velocity. In particular, information on the Total Grain-Size Distribution (TGSD), namely the size distribution of particles injected into the atmosphere, combined with particle density is necessary to initialize the model.

Turbulent effects, which play an important role in this model, are represented through a diffusion process. Different diffusion coefficients are applied in the atmosphere (D_a) and in the plume (D_p), the latter coefficient being usually several times larger.

We divide space into three main zones (Figure 2), where the velocity fields are described as

- atmosphere : $\vec{u} = \vec{u}_s + \vec{u}_w + \vec{u}_{ra}$;
- volcanic column : $\vec{u} = \vec{u}_s + \vec{u}_w + \vec{u}_p + \vec{u}_{rp}$;
- umbrella cloud : $\vec{u} = \vec{u}_s + \vec{u}_w + \vec{u}_b + \vec{u}_{ra}$;

where

- $\vec{u}_s$ is the settling velocity of the particles ([21], [26]);
- $\vec{u}_w$ is the wind velocity, retrievable from analytical models ([22], [27]), weather reanalysis databases or weather forecasting models;
- $\vec{u}_p$ is the plume velocity ([25]);
- $\vec{u}_b$ is the spreading velocity of the umbrella cloud due to buoyancy ([21]);

Figure 2: Velocity fields considered in the model. Note that velocities are summed to produce the final advection field. For example in the umbrella cloud, $\vec{u}_b$ and $\vec{u}_w$ will be summed, so the umbrella cloud will be elongated in the downwind direction, as shown in Figure 1.

- $\vec{u}_{ra}$ and $\vec{u}_{rp}$ are random velocities for simulating diffusion in the atmosphere and in the volcanic column.

Once a set of parameters is selected, a simulation is run by injecting particles at the vent inside the domain and waiting for particles to either deposit on the ground or leave the domain.

3. Numerical model

Several techniques like lattice Boltzmann, probabilistic multi-particle cellular automata or Lagrangian methods, can be used to model point particle displacements in an advection field ([17], [28], [29]). Cellular automata are discrete Eulerian models where the simulated space is subdivided along a Cartesian grid. Each grid site stores the number of particles residing on this site for every particle class, together with the local advection field. A probabilistic scheme is then applied to transport particles from one site to another at each iteration. This kind of model has the advantage of facilitating parallelization. However, the probabilistic nature of particle motion creates an anisotropic numerical diffusion ([30]).

The lattice Boltzmann method is another technique for modeling an advection-diffusion process. However, its numerical stability and accuracy depend on simulation parameters ([31]). Cellular automata and lattice Boltzmann methods are compared in [30].

From the pure Lagrangian perspective, the characteristics of each particle are considered individually. This kind of model doesn't generate any numerical diffusion while particles are not constrained to stay at grid positions on the domain sites (Figure 3). This allows a precise treatment of diffusion, regardless of the spatial integration step. However, the drawback is that parallelization is more tedious when particles can interact.

For these reasons, we chose a hybrid representation defined as a Lagrangian on grid model. In this case, particles are moved according to a Lagrangian model, but remain attached to sites in space (this representation is comparable to the one used in the cell linked-lists algorithm [32]).

(a) Representation of the Lagrangian on grid model without the velocities.

(b) Differentiation between Eulerian and Lagrangian velocities.

Figure 3: Representation of the Lagrangian on grid model. Gray lines represent the grid, gray points are the center of the sites, black points are the particles and dotted lines represent association between particles and their respective sites.

This method combines the simplicity of pure Lagrangian and cellular automata models. As for a pure Lagrangian method, the Lagrangian on grid method doesn't generate any numerical diffusion. Moreover, this method remains unconditionally stable like cellular automata and Lagrangian methods but unlike lattice Boltzmann methods.

The Lagrangian on grid model represents space as a Cartesian grid. The grid nodes are the domain sites, each site storing the associated particles and applying the advection field to them. But in contrast to a cellular automata model, each particle stores a velocity as well as a location information (the displacement with respect to the site it is linked to).

Thus, the position of a particle at time t can be written $\vec{r}(t) = \vec{r}_s(t) + \vec{r}_p(t)$ with $\vec{r}_s$ the site position to which the particle is linked and $\vec{r}_p$ the particle displacement relative to this site. Similarly, the velocity applied to a particle can be written $\vec{u}(t) = \vec{u}_e(t) + \vec{u}_l(t)$ with $\vec{u}_e$ the Eulerian velocity (imposed by the site) and $\vec{u}_l$ the Lagrangian velocity (specific to the particle).

Then, when a particle is moved, we compute its new position after a time step Δt via $\vec{r}(t + \Delta t) = \vec{r}(t) + \Delta t \vec{u}(t)$. If the particle displacement is such that $\|\vec{r}(t + \Delta t) - \vec{r}_s(t)\| \leq \frac{\Delta x}{2}$ with Δx the distance between two sites, then $\vec{r}_s(t + \Delta t) = \vec{r}_s(t)$ (the particle remains linked to the same site). Otherwise the particle is linked to the closest site (Figure 3).

In the Lagrangian on grid model, no numerical diffusion is generated. In order to add some diffusion, we apply random velocities $\vec{u}_r$ to each particle ($\vec{u}_{ra}$ and $\vec{u}_{rp}$) (see supplementary materials for details).

4. Tephra transport simulator

We first implemented a generic C++ library to simulate particles in an advection field by applying the Lagrangian on grid method (PIAF, Particles In Advection Field). The TEphra TRANsport Simulator (TETRAS) was then built on top of this library.

4.1. Data structures

We implemented classes representing the model main components (particles, velocities, terrain, ...). Below are the most important data structures of the simulator.

140 **Particle** This class stores the characteristics of the particles (position, lagrangian velocity and
 141 its corresponding family) while **ParticleFamily** stores those of the particle families (density,
 142 size, diameter and mass).

143 **Velocity** Velocities are functions taking into account position, time and particle characteristics to
 144 compute the advection field.

145 **Domain** The simulation occurs inside a discretized domain, itself split into boxes of size Δx^3 in the
 146 3D case.

147 **Terrain** This class allows to compute where particles deposit, and to store them.

148 **Simulator** This class manages the other data structures and applies the numerical scheme described
 149 above.

150 More information is available in the supplementary materials.

151 4.2. Adaptive time step

152 In our application, we can decrease the number of iterations (and thus the computation time)
 153 by increasing Δt . Nevertheless, a movement for a particle is valid if the particle stays linked to the
 154 same site or moves to a direct neighbour. Consequently, the choice of Δt is limited by the choice
 155 of the grid spacing Δx and by the velocities $\vec{u}$.

156 We wish to choose the largest possible Δt for a given Δx . Since this value may vary over time,
 157 we propose to dynamically adapt Δt using the following simple heuristic :

- 158 • Every τ iterations, Δt is increased by a small quantity δt .
- 159 • If a particle makes an invalid movement, Δt is decreased by δt and the iteration cancelled
 160 and performed again.

161 In order to cancel an iteration, we keep the state of the system unmodified using temporary
 162 buffers until the iteration is complete. The behaviour of the simulator with adaptive Δt is summa-
 163 rized in algorithms 1 and 2.

Algorithm 1 Main algorithm of the simulation, we suppose that data are shared between the main
 algorithm and the function `tryMove()`

```

while particles in domain do
  if iteration mod  $\tau == 0$  then
     $\Delta t += \delta t$ 
  end if
  while not tryMove() do
     $\Delta t -= \delta t$ 
  end while
  deposit particles of terrain buffer on the terrain
  put all particles of repository buffer into repository
  swap domain and domain buffer
end while

```

Algorithm 2 Algorithm of the function `tryMove()`

```
for all site in domain do
  compute velocity at the site
  if site is in contact with ground then
    for all particle in site do
      put particle in terrain buffer
    end for
  else
    for all particle in site do
      compute velocity of the particle
      compute new position of the particle
      if new position is valid then
        if particle is outside the domain then
          put the particle in repository buffer
        else
          put the particle in domain buffer
        end if
      else
        return false
      end if
    end for
  end if
end for
return true
```

4.3. Parallelization

For the implementation to scale well with the amount of data, our library has been parallelized using the MPI (Message Passing Interface) library. In our case, we chose to split the domain into blocks of sites, which are distributed across multiple processors. We implemented two solutions : a 1D domain splitting and a 3D block cyclic domain splitting (Figure 4). The first consists in slicing the domain along a given axis, with each processor taking care of one domain slice. The latter distribution allows to deal better with an inhomogeneous distribution of particles in the domain which would otherwise lead to load imbalance between the processors and thus performance loss. Indeed, since the domain is split across multiple processors and there is a unique injection point (in the case of the volcano model), at the beginning of a simulation, particles tend to remain linked to sites managed by a single processor.

Figure 4: Illustration of (a) unidimensional domain splitting and (b) two dimensional block cyclic domain splitting with four processors, each colour representing a different processor. The implemented three dimensional block cyclic domain splitting is an extension to three dimension of the two dimensional block cyclic distribution presented here. Figures are inspired from [33].

We want to assign each processor a certain number of small domain blocks, distributed over the entire domain. We thus hope to obtain a better load balance as each processor will have to manage blocks located at different places in the domain. The technique described here is comparable to the 2D block cyclic matrix distribution described in [33].

We first arrange the processors along a virtual 3D grid and thus assign each processor an index (p_x, p_y, p_z) on the grid. Given the number of processors np , we factor np writing $np_x \cdot np_y \cdot np_z = np$ (it may occur that only a 1D or 2D structure is constructed). To a processor of rank p ($0 \leq p \leq np - 1$), we assign a grid index (p_x, p_y, p_z) with $p_x = p \bmod np_x$, $p_y = \frac{p \bmod (np_x \cdot np_y)}{np_x}$ and $p_z = \frac{p}{np_x \cdot np_y}$.

We now discuss the 3D domain block splitting and the assignment of blocks to the processors. Let s_x, s_y, s_z and h_x, h_y, h_z be the dimensions respectively of the domain and of a block. The number of blocks nb_x, nb_y, nb_z in each dimension is given by:

$$(nb_x, nb_y, nb_z) = \left(\left\lceil \frac{s_x}{h_x} \right\rceil, \left\lceil \frac{s_y}{h_y} \right\rceil, \left\lceil \frac{s_z}{h_z} \right\rceil \right) \quad (1)$$

187 Note that blocks on the boundary of the domain may be smaller than $h_x \cdot h_y \cdot h_z$.

188 We next decide how to allocate the blocks to the processors. The domain blocks are assigned
 189 cyclically to the processors along each dimension. More precisely, a site with global index (r_x, r_y, r_z)
 190 will belong to a block with index $b = \left(\frac{r_x}{h_x}, \frac{r_y}{h_y}, \frac{r_z}{h_z}\right)$. It also has a local index inside the block as in
 191 the case of the 1D splitting. This block will be managed by the processor with index

$$(p_x, p_y, p_z) = (b_x \bmod np_x, b_y \bmod np_y, b_z \bmod np_z) \quad (2)$$

192 A processor with index (p_x, p_y, p_z) takes care along the x -axis of $\lfloor nb_x/np_x \rfloor$ blocks if $p_x < nb_x$
 193 $\bmod np_x$, and $\lceil nb_x/np_x \rceil$ blocks otherwise. Similar expressions hold along the y - and z -axes.

194 5. Validation of the physical model with field observations

195 We ran our simulations on two systems : the *Baobab* cluster of the University of Geneva and
 196 the *Piz Daint* supercomputer of the Swiss National Supercomputing Center.

197 We compared the model output with field data for two eruptions characterized by very different
 198 plume dynamics and interaction with the surrounding wind field. The first is the 2450BP eruption of
 199 Pululagua volcano (Ecuador)[34], which produced a 25[km] high strong plume that rose in absence
 200 of wind (plume heights are here all indicated as above the vent). The second is the 17 June 1996
 201 eruption of Ruapehu volcano (New Zealand)[20], which produced a 7[km] high weak plume in a
 202 wind field with an average speed of 24[$\frac{m}{s}$] at the tropopause. The associated field observations used
 203 for comparing computed and observed mass per unit area (MPA) include:

- 204 • the geographical position (UTM WGS84 for Pululagua or NZGD1949 for Ruapehu);
- 205 • the total deposited mass per unit area MPA [$\frac{kg}{m^2}$];
- 206 • the proportion of MPA for each particle family.

207 Some crucial input parameters are reported in the literature or can be derived from external
 208 models (e.g. TGSD, erupted mass, wind profile and atmospheric conditions), while others (mostly
 209 plume and diffusion parameters) are more difficult to define. In particular, the TGSD is reported
 210 by [35] for Ruapehu (i.e. Md_ϕ and sorting of -0.8 ϕ and 2.4, respectively) and by [34] for Pululagua
 211 (i.e. Md_ϕ and sorting of 0.2 ϕ and 1.9, respectively), while [34] suggests a total erupted mass of
 212 $4.5 \times 10^{11}[kg]$ for Pululagua. Given that the Ruapehu eruption was observed and consisted of two
 213 sustained events of 4h30 and 2h respectively, the total mass was directly derived from the mass
 214 eruption rate combined with the total eruption time (i.e. $10^{10}[kg]$). This result is in agreement
 215 with the total erupted mass of $5 \times 10^9[kg]$ derived by [20] based on field data. We used the NOAA
 216 NCEP/NCAR reanalysis database for the wind and atmospheric data, which provides four daily
 217 profiles averaged on a $2.5^\circ \times 2.5^\circ$ grid (<http://www.esrl.noaa.gov/psd/>, [36]). For this particular
 218 test case, we used wind data at the vent location, and applied it to the whole domain. It is worth
 219 noting that, as shown by [20], the spreading of the Ruapehu cloud was strongly controlled by wind
 220 advection and, therefore, gravitational spreading was neglected. In contrast, the spreading of the
 221 Pululagua cloud was controlled by buoyancy, given that the eruption occurred in absence or weak
 222 atmospheric wind.

223 Finally, the strategy adopted to derive the plume and diffusion parameters was to stochastically
 224 sample appropriate variation ranges and identify the values associated with the best fit of observed
 225 MPA. Parameter exploration for each eruption was made with : $U_0 \in [20, 300]$, $r_0 \in [10, 100]$,

Eruption	2450BP Pululagua	1996 Ruapehu
U_0	$300[\frac{m}{s}]$	$40[\frac{m}{s}]$
r_0	$50[m]$	$50[m]$
θ_0	1200	1273
D_a	$300[\frac{m^2}{s}]$	$1000[\frac{m^2}{s}]$
D_p	$3000[\frac{m^2}{s}]$	$4500[\frac{m^2}{s}]$

Table 1: Results of parameter exploration. These values are used for the presented simulation outputs.

226 $D_a \in [100, 10000]$, $D_p \in [1000, 30000]$. For Ruapehu, $\theta_0 = 1273$ comes from [37]. For Pululagua,
 227 θ_0 was sampled in $[1000, 1200]$ and n_0 set to 0.03. Table 1 shows the best-fit parameters for the
 228 simulations. Plume heights above vent based on these best-fit parameters are $25[km]$ and $7[km]$
 229 for Pululagua and Ruapehu, respectively, which are in good agreement with observations ([34],
 230 [35]). Cloud dispersal was initialized considering plume height above vent (i.e. $\sim 3[km]$ for both
 231 Pululagua and Ruapehu volcanoes).

232 A quantitative comparison with observed MPA for both eruptions is shown in Figure 5 and
 233 Figure 6. In particular, the computed isomass map for the 1996 Ruapehu eruption well describes
 234 the strong wind advection (Figure 5). The associated plume being strongly bent-over by the wind,
 235 most lapilli and coarse ash particles fell in proximal area ([20]). Very proximal sedimentation
 236 couldn't be reproduced because of the lack of topography in our model. Topography would indeed
 237 strongly affect particle deposition up to $20[km]$ from vent.

Figure 5: Simulated contour map of MPA (coloured lines) and observed MPA (coloured circles) for the 2450BP Pululagua deposit (a) and the 1996 Ruapehu deposit (b).

238 A dedicated sensitivity analysis, following [38], was performed to optimize the number of par-

Figure 6: Comparison of computed and observed MPA values for the Pululagua 2450BP (a) and 1996 Ruapehu (b) eruptions. In the case of a perfect match between computed and observed values, dots would be located on the blue line. Dashed lines represents over or under estimations of 5 and 1/5 times the observed values.

titles used in the simulations. In fact, the total number of particles corresponding to the erupted mass and the particle characteristics given in this section would be of the order of magnitude of 10^{22} to 10^{23} particles for Ruapehu and Pululagua simulations. Current computers cannot deal with such a number of particles injected in the simulator.

We therefore resorted to a statistical sampling approach. We injected a much smaller number of particles, and then scaled the output to get the correct number. Running our simulations with 10^6 particles per family proved to be statistically significant. This corresponds to a total of $18 \cdot 10^6$ particles with 18 particle families for 2450BP Pululagua simulations and $23 \cdot 10^6$ particles for 1996 Ruapehu simulations. For a complete description of the procedure, see the supplementary materials.

Note that prior to running our volcano simulations, we validated the advection-diffusion process (also in supplementary materials).

6. Performance models

The main goal of a performance model is to characterize the execution time of a simulation in terms of its parameters. This execution time will of course depend on the physical model implemented. A parallel performance model will provide insight into the efficiency of the parallel implementation with respect to the computational resources involved. We have carried out a performance study with both simulation cases described before. Performance measurements for Pululagua were made with square domains with the vent located in the middle, and for Ruapehu with an elongated domain in the north direction with the vent located near the south-west corner of the domain, while the wind was blowing towards north-east.

6.1. Choice of the adaptive time step

It is important to carefully choose δt to get an optimal performance improvement. We observe the evolution of execution time as a function of δt . Knowing that the initial value of $\Delta t \approx 0.5$ at

Figure 7: Execution time and number of iterations for a sequential simulation of the 2450BP Pululagua eruption. In this simulation, 18×10^6 particles are injected into a domain of size $60[km] \times 60[km] \times 30[km]$. Parameter values: $\Delta t = 0.5[s]$ initially, $\Delta x = 500[m]$, $\delta t \in [0.001 : 3.0]$.

the beginning of the simulation, we empirically choose δt values $\in [0.001 : 3.0]$ (Figure 7).

We see that execution time falls steeply for small values of δt , then becomes stable and finally begins to grow again slowly as δt increases in $[0.5 : 3.0]$. The latter phenomenon occurs when more and more adjustments are rejected.

Now a crucial question is to determine if the dynamical adjustment of Δt involves a significant overhead. Figure 7 shows the evolution of the number of iterations until the end of the simulation as a function of δt as well as the corresponding simulation time. The curves are very similar.

This supports the claim that the use of an adaptive Δt scheme decreases the number of iterations without adding a significant amount of work, when the value of δt is properly chosen.

6.2. Computational work

The total work performed in the simulation is the sum of the work W_i done at each iteration i , plus the initial and final work ($W^{(init)}$ and $W^{(final)}$). With $n^{(it)}$ the total number of iterations, we thus have:

$$W^{(tot)} = W^{(init)} + W^{(final)} + \sum_{i=1}^{n^{(it)}} W_i \quad (3)$$

The work W_i consists of several parts which are :

- the work required for the computation of Eulerian dynamic velocities $n^{(s)} \cdot W^{(ed)}$ with $W^{(ed)}$ the work to compute one site and ns the number of sites in the domain;
- the work to compute the Lagrangian velocities and to move the particles $n_i^{(p)} \cdot W^{(lm)}$ with $W^{(lm)}$ the work to compute and move one particle, and $n_i^{(p)}$ the number of particles in motion at iteration i ;

281 • the work involved for the particles that leave the domain, which can be neglected as this work
 282 occurs only once for each particle.

283 Thus, we can write $W_i = n^{(s)}W^{(ed)} + n_i^{(p)}W^{(lm)}$

284 Equation (3) can be rewritten (by taking into account that $W^{(init)} + W^{(final)} \ll \sum_{i=1}^{n^{(it)}} W_i$) :

$$\begin{aligned}
 W^{(tot)} &= \sum_{i=1}^{n^{(it)}} (n^{(s)}W^{(ed)} + n_i^{(p)}W^{(lm)}) \\
 &= n^{(it)}n^{(s)}W^{(ed)} + \left(\sum_{i=1}^{n^{(it)}} n_i^{(p)}\right)W^{(lm)} \\
 &= n^{(it)}(n^{(s)}W^{(ed)} + n^{(p)}W^{(lm)})
 \end{aligned} \tag{4}$$

285 where $n^{(p)} = \frac{1}{n^{(it)}} \sum_{i=1}^{n^{(it)}} n_i^{(p)}$ is the average number of particles in motion.

286 6.3. Sequential performances

287 We can directly deduce the sequential complexity from Equation (4).

288 Given the total work $W^{(tot)}$ (in clock cycles) performed in a simulation and $Freq[Hz]$ the
 289 frequency of the processor, the total computation time for a sequential execution is :

$$T^{(tot)} = \frac{W^{(tot)}}{Freq} \tag{5}$$

$$T^{(tot)} = n^{(it)}(n^{(s)}T^{(ed)} + n^{(p)}T^{(lm)}) \tag{6}$$

290 with $n^{(s)}$ the number of domain sites, $n^{(p)}$ the average number of particles in motion, $n^{(it)}$ the
 291 number of iterations, and the times $T^{(ed)} = W^{(ed)}/Freq$ and $T^{(lm)} = W^{(lm)}/Freq$.

292 We verified on some simulations that the execution time $T^{(tot)}$ indeed depends linearly on $n^{(p)}$
 293 and $n^{(s)}$. Note that the average number of particles in motion in a simulation is itself proportional
 294 to the total number of particles injected in the domain.

295 We first observed the effect of the number of particles on execution time. We fixed Δx , Δt
 296 and $n^{(it)}$ as well as the domain size (we didn't use the adaptive time step in this case) and varied
 297 the number of particles injected in the domain in order to confirm the linear dependence of the
 298 execution time on the number of particles (data not shown).

299 Next, we examined the effect of the number of domain sites on the execution time. Here, the
 300 number of injected particles, the domain size and the number of iterations were kept constant, while
 301 the number of sites was varied. Here again, we observed a linear dependence between the number
 302 of sites and execution time (data not shown).

303 Finally, we express the execution time $T^{(tot)}$ in terms of the time step Δt and the lattice spacing
 304 Δx . If we assume the physical time and Δt are constant in the simulation, then the number of
 305 iterations is directly proportional to $\frac{1}{\Delta t}$. Moreover, the number of sites $n^{(s)}$ can be expressed in
 306 terms of volume V of the domain and Δx as $n^{(s)} = V/(\Delta x)^3$.

307 By taking this into account and rewriting Equation (6), we get :

$$T^{(tot)} = C \frac{1}{\Delta t} \left(\left(\frac{1}{\Delta x} \right)^3 \cdot VT^{(ed)} + n^{(p)}T^{(lm)} \right) \tag{7}$$

308 for some constant C .

309 With the adaptive Δt scheme, the value of Δt is no longer a constant. Recall that the goal
 310 of this optimization is to allow the fastest particles in the domain to perform the largest possible
 311 movement. Their motion is however limited by Δx . This hints towards the number of iterations
 312 $n^{(it)}$ not being anymore proportional to $\frac{1}{\Delta t}$, but to $\frac{1}{\Delta x}$. Therefore, we will assume that the number
 313 of iterations $n^{(it)}$ in Equation (6) is no longer proportional to $\frac{1}{\Delta t}$, but to $\frac{1}{\Delta x}$. By taking that into
 314 account, we get the following expression :

$$T^{(tot)} = \hat{C} \frac{1}{\Delta x} \left(\left(\frac{1}{\Delta x} \right)^3 \cdot VT^{(ed)} + n^{(p)} T^{(lm)} \right) \quad (8)$$

315 for some constant $\hat{C}$.

Figure 8: Sequential execution time of the simulation of the 2450BP Pululagua eruption as a function of Δx with a domain of size $60[km] \times 60[km] \times 30[km]$ and $1 \cdot 10^5$ particles, using the adaptive Δt scheme. Theoretical curve with $\hat{C} \cdot T^{(ed)} \approx 1.411 \cdot 10^{-1}$ and $\hat{C} \cdot T^{(lm)} \approx 6.237$.

316 To verify the model, we performed several simulations to record the execution time as a function
 317 of Δx with fixed physical simulation time and fixed number of particles injected into the domain.
 318 The adaptive Δt scheme is used. The results are shown on Figure 8. By adjusting the constants
 319 using the `curve_fit` function of `scipy`¹, we can make the theoretical curve fit the simulation data
 320 well. It seems therefore that the proposed performance model is satisfactory.

321 6.4. Parallel performances

322 In an ideal context, the execution time of a parallel implementation should equal the sequential
 323 time divided by the number of processors used. Unfortunately, the performances and scalability of
 324 a parallel application are limited mainly by three factors: network communications, work overhead
 325 and load imbalance. Their impact must be taken into account in a parallel performance model.

¹Non-linear least squares method optimized with the Levenberg-Marquardt algorithm.

326 6.4.1. Communication time

327 For simplicity, we consider that all processors can communicate at the same time. In the block
 328 cyclic splitting of the domain, each domain block has 26 neighbours, implying that each processor
 329 has virtually 26 neighbours. At each iteration, each processor sends and receives from its neighbours
 330 a particle packet. Obviously, we don't know how many particles will travel from a domain block
 331 to another, but we can assume that for a given physical model, the average number of particles
 332 $n^{(p)}$ travelling between domain blocks at each iteration, is proportional to the number of particles
 333 injected into the domain.

334 So each processor must send data to its 26 neighbours, first sending the data size, before trans-
 335 mitting the data itself. This implies that each communication pays 52 times the latency. The rest
 336 of the communication time is on average directly proportional to the number of particles in the
 337 domain. However, the latency contribution can be neglected if $n^{(p)}$ is large enough, which is the
 338 case in our simulations. We will thus approximate the communication time by :

$$T^{(com)} = C \cdot n^{(p)} \quad (9)$$

339 for some constant C .

340 6.4.2. Work overhead

341 Each particle that travels from a domain block to another is placed into a particle packet before
 342 being transmitted. Upon reception, the particles received must be inserted at the correct location
 343 in the recipient's local domain. We thus must add a term for this work, here again proportional to
 344 $n^{(p)}$:

$$T^{(over)} = C \cdot n^{(p)} \quad (10)$$

345 for some constant C .

346 6.4.3. Load imbalance

347 Load imbalance is actually the factor that impacts the most performances in this application.
 348 In order to characterize load imbalance, let's define two notions: real work and apparent work.

349 Real work is simply the sum of the work achieved by each processor during the execution of the
 350 algorithm, in other words the sum of all clock cycles consumed by the CPUs during the computation.
 351 This can be written :

$$W^{(r)} = \sum_{i=1}^{n^{(it)}} \sum_{j=1}^{n^{(proc)}} W_{ij} \quad (11)$$

352 where W_{ij} is the work performed by processor $j \in \{1, \dots, n^{(proc)}\}$ at iteration i . Note that the
 353 total work W_i at iteration i is given by : $W_i = \sum_{j=1}^{n^{(proc)}} W_{ij}$.

354 Processors must synchronize with each other at each iteration. Therefore, at each iteration, all
 355 processors must wait for the processor with the heaviest load. It thus appears as if all processors
 356 seems had the same load as the processor with the heaviest load at each iteration. We now define
 357 the apparent work as :

$$W^{(a)} = n^{(proc)} \cdot \sum_{i=1}^{n^{(it)}} \max_{j=1, \dots, n^{(proc)}} W_{ij} \quad (12)$$

358 Then a natural indicator of load balance is the ratio :

$$Eq = \frac{W^{(r)}}{W^{(a)}} \quad (13)$$

359 Load imbalance is then simply defined as the inverse of load balance.

360 In order to illustrate these definitions, let's consider a simulation with 1D domain splitting over
 361 16 processors. The graph of real work $W^{(r)}$ is shown on Figure 9a. We observe a very high work
 362 load at the beginning of the execution. The load imbalance appears clearly during the first 10000
 363 iterations. This execution shows a global load balance of $Eq \approx 0.45$, implying a load imbalance of
 364 $1/Eq \approx 2.2$.

365 Figure 9b shows the same execution but with the block cyclic domain splitting. This time, the
 366 load is better balanced between processors, and the value of the balance indicator is $Eq \approx 0.72$,
 367 representing an imbalance of $1/Eq \approx 1.4$. This is a big improvement over to the previous simulation
 368 and shows that the 3D block cyclic domain distribution helps to reduce load imbalance.

Figure 9: Real work achieved by 16 CPUs in the 3000 first iterations of the 2450BP Pululagua simulation, each colour representing the work of one processor.

369 6.4.4. Execution time and speedup

370 We saw that the block cyclic domain splitting allows to achieve better load balance. We thus
 371 examine the performance and speedup obtained with this implementation. Based on the discussions
 372 of the previous section, we express the parallel execution time as :

$$T^{(par)} = \frac{1}{Eq} \left(\frac{T^{(seq)}}{n^{(proc)}} + T^{(over)} \right) + T^{(com)} \quad (14)$$

373 with $Eq \in]0, 1]$ the load balance, $T^{(seq)}$ the sequential execution time, $n^{(proc)}$ the number of proces-
 374 sors used, $T^{(over)}$ the work overload due to parallelization and $T^{(com)}$ the communication time. We

375 know that $T^{(over)}$ and $T^{(com)}$ are proportional to $n^{(p)}$. Unfortunately, we don't know the evolution
 376 of load balance as a function of $n^{(proc)}$. We know that load balance has a maximum value of 1 and
 377 it is likely that it decreases as $n^{(proc)}$ increases. Empirically, we state for testing that :

$$Eq = (n^{(proc)})^{\frac{-1}{C}}$$
(15)

378 for some constant C , could be an acceptable approximation.

379 Finally the parallel execution time takes the following form :

$$T^{(par)} = (n^{(proc)})^{\frac{1}{C_1}} \left(\frac{T^{(seq)}}{n^{(proc)}} + C_2 \cdot n^{(p)} \right) + C_3 \cdot n^{(p)}$$
(16)

380 with parameters C_1 , C_2 and C_3 which allow to tune the load balance, the work overhead and the
 381 communication time.

382 Three simulation cases were tested in order to observe the behaviour of the parallelized imple-
 383 mentation and the accurateness of the performance model. The first two are simulations of the
 384 2450BP Pululagua eruption, with a small domain of $60[km] \times 60[km] \times 30[km]$ then with a larger
 385 domain of $200[km] \times 200[km] \times 30[km]$. The last is a simulation of the 1996 eruption of Ruapehu
 386 with a domain of size $650[km] \times 1050[km] \times 30[km]$. Each simulation case was run with $\Delta x = 500m$
 387 and 10^6 particles per family.

388 Figure 10 shows the speedups obtained. We see that the expression in Equation (16) is a good
 389 approximation for the parallel performances. Note that the average number of particles in motion
 390 depends on the domain size. This is due to the fact that in a smaller domain more particles will
 391 exit the domain during a given physical simulation.

392 Figure 11 shows a comparison between theoretical speedup for the different simulation cases. We
 393 see that the larger the domain, the better the performances with Pululagua simulations. Alternately,
 394 we see that Ruapehu parallel simulations are less efficient. Indeed, while particles spread in every
 395 direction in the case of Pululagua, they mainly move along the same direction in the case of Ruapehu,
 396 which makes it a more difficult case to parallelize, particles not being well spread across the domain.
 397 Nonetheless, performance remains good thanks to the 3D domain block cyclic distribution.

398 7. Conclusion and future work

399 We presented a new VATDM, which is based on a hybrid Eulerian-Lagrangian numerical scheme
 400 (Lagrangian on grid model) and offers multiple advantages. First, simulations remain uncondition-
 401 ally stable; second, it does not generate numerical diffusion; third, space which is represented as a
 402 Cartesian grid, can be easily subdivided, thus facilitating a parallel implementation. Finally, our
 403 numerical model has a modular structure easing future implementations (e.g. particle aggregation)
 404 and allows for both the monitoring of atmospheric ash concentration and ground mass load at any
 405 given time during a simulation.

406 Our implementation of the model consists of two main parts: a library dedicated to the im-
 407 plementation of the numerical model and its parallelization (PIAF) and the TEphra TRANsport
 408 Simulator (TETRAS), which contains the physical model. It also integrates a particular opti-
 409 mization feature with the dynamic adaptation of the time step, which was shown to increase the
 410 efficiency of the simulator. Note that both the model and the simulation tool are flexible in the
 411 sense that the design is made to include new components for additional physical submodels.

(a) Domain size $60[km] \times 60[km] \times 30[km]$. Theoretical curve with $C_1 \approx 1.745 \cdot 10^1$, $C_2 \approx 6.463 \cdot 10^{-4}$ and $C_3 \approx 1.259 \cdot 10^{-10}$. $R^2 = 0.99$. Computation time with one processor ≈ 10 hours.

(b) Domain size $200[km] \times 200[km] \times 30[km]$. Theoretical curve with $C_1 \approx 1.095 \cdot 10^1$, $C_2 \approx 4.546 \cdot 10^{-4}$ and $C_3 \approx 3.570 \cdot 10^{-11}$. $R^2 = 0.995$. Computation time with one processor ≈ 48 hours.

(c) Domain size $650[km] \times 1050[km] \times 30[km]$. Theoretical curve with $C_1 \approx 1.013 \cdot 10^1$, $C_2 \approx 3.484 \cdot 10^{-4}$ and $C_3 \approx 1.284 \cdot 10^{-11}$. $R^2 = 0.996$. Computation time with one processor ≈ 31 hours.

Figure 10: Speedup for two simulation cases of 2450BP Pululagua eruption with 18×10^6 particles (a,b) and one simulation case of 1996 Ruapehu eruption with 23×10^6 particles (c), $\Delta x = 500[m]$.

412 The correctness of the advection and diffusion processes described in the model was validated (we
 413 recall that turbulence was modelled using a diffusion process). We also obtained a good agreement
 414 between computed and observed MPA for the Pululagua and Ruapehu deposits, even though the

Figure 11: Theoretical speedup comparison between a small and a large domain for Pululagua eruption (respectively $60[km] \times 60[km] \times 30[km]$ and $200[km] \times 200[km] \times 30[km]$) and a domain of size $650[km] \times 1050[km] \times 30[km]$ for Ruapehu.

implementation of topography in the model would largely improve the proximal sedimentation, especially for strongly bent-over plumes occurring at high altitude volcanoes, such as that of the 1996 Ruapehu eruption. We also determined the adequate number of particles to use in the simulations as a function of correlation thresholds. For our case studies, we obtained a mean correlation of 0.99 with a little more than 10^6 particles.

Finally, we established a performance model for the sequential implementation, which takes into account the adaptive time step scheme. We then derived a parallel performance model which also encompasses load imbalance. These models were validated by recording execution times over many simulations.

Future work will require an improvement of the physical model, including the implementation of ground topography, particle aggregation and coupling with weather prediction models. These implementations would allow for the description of proximal sedimentation at high volcanoes, of fine ash that typically falls as particle clusters and of far-range ash dispersal that can be affected by complex wind advection with distance from vent. The ultimate goal is to propose a full multiphysics tephra dispersal model, together with an implementation using a multiscale modelling framework on a high performance computing infrastructure able to simulate the tephra dispersal at multiple scales.

Acknowledgements. This work was supported by the Swiss National Science Foundation project No 200020-125024 and the PASC project “Optimal deployment of multiscale applications on a HPC infrastructure”². We also wish to thank Dr. Wim Degruyter for his help in the use of the plume model.

²<http://www.pasc-ch.org/projects/projects/multiscale-applications/>

References

- [1] C. J. Horwell, P. J. Baxter, The respiratory health hazards of volcanic ash: a review for volcanic risk mitigation, *Bulletin of Volcanology* 69 (1) (2006) 1–24. doi:10.1007/s00445-006-0052-y.
- [2] M. Guffanti, G. C. Mayberry, T. J. Casadevall, R. Wunderman, Volcanic hazards to airports, *Natural Hazards* 51 (2) (2008) 287–302. doi:10.1007/s11069-008-9254-2.
- [3] T. J. Casadevall, Volcanic ash and aviation safety; proceedings of the First international symposium on Volcanic ash and aviation safety, USGS Numbered Series 2047 (1994). URL <http://pubs.er.usgs.gov/publication/b2047>
- [4] J. B. Wardman, T. M. Wilson, P. S. Bodger, J. W. Cole, C. Stewart, Potential impacts from tephra fall to electric power systems: a review and mitigation strategies, *Bulletin of Volcanology* 74 (10) (2012) 2221–2241. doi:10.1007/s00445-012-0664-3.
- [5] T. Wilson, C. Stewart, J. Cole, D. Johnston, S. Cronin, Vulnerability of farm water supply systems to volcanic ash fall, *Environmental Earth Sciences* 61 (4) (2009) 675–688. doi:10.1007/s12665-009-0380-2.
- [6] T. M. Wilson, C. Stewart, V. Sword-Daniels, G. S. Leonard, D. M. Johnston, J. W. Cole, J. Wardman, G. Wilson, S. T. Barnard, Volcanic ash impacts on critical infrastructure, *Physics and Chemistry of the Earth, Parts A/B/C* 4546 (2012) 5–23. doi:10.1016/j.pce.2011.06.006.
- [7] M. T. Gudmundsson, R. Pedersen, K. Vogfjrd, B. Thorbjarnardttir, S. Jakobsdttir, M. J. Roberts, Eruptions of Eyjafjallajkull Volcano, Iceland, *Eos, Transactions American Geophysical Union* 91 (21) (2010) 190–191. doi:10.1029/2010E0210002.
- [8] T. Wilson, C. Stewart, H. Bickerton, P. Baxter, V. Outes, G. Villarosa, E. Rovere, Impacts of the June 2011 Puyehue Cordon-Caulle volcanic complex eruption on urban infrastructure, *GNS Science*, 2012.
- [9] A. Folch, A review of tephra transport and dispersal models: Evolution, current status, and future perspectives, *Journal of Volcanology and Geothermal Research* 235236 (2012) 96–115. doi:10.1016/j.jvolgeores.2012.05.020.
- [10] C. Bonadonna, A. Costa, Modeling tephra sedimentation from volcanic plumes, in: *Modeling Volcanic Processes*, Cambridge University Press, 2013. doi:10.1017/CB09781139021562.009.
- [11] C. Bonadonna, A. Folch, S. Loughlin, H. Puempel, Future developments in modelling and monitoring of volcanic ash clouds: outcomes from the first IAVCEI-WMO workshop on Ash Dispersal Forecast and Civil Aviation, *Bulletin of Volcanology* 74 (1) (2011) 1–10. doi:10.1007/s00445-011-0508-6.
- [12] A. Costa, G. Macedonio, A. Folch, A three-dimensional Eulerian model for transport and deposition of volcanic ashes, *Earth and Planetary Science Letters* 241 (34) (2006) 634–647. doi:10.1016/j.epsl.2005.11.019.

- 473 [13] A. Folch, A. Costa, G. Macedonio, FALL3D: A computational model for transport and depo-
474 sition of volcanic ash, *Computers & Geosciences* 35 (6) (2009) 1334–1342. doi:10.1016/j.
475 cageo.2008.08.008.
- 476 [14] S. Barsotti, A. Neri, J. S. Scire, The VOL-CALPUFF model for atmospheric ash dispersal: 1.
477 Approach and physical formulation, *Journal of Geophysical Research: Solid Earth* 113 (B3)
478 (2008) B03208. doi:10.1029/2006JB004623.
- 479 [15] S. Barsotti, A. Neri, The VOL-CALPUFF model for atmospheric ash dispersal: 2. Application
480 to the weak Mount Etna plume of July 2001, *Journal of Geophysical Research: Solid Earth*
481 113 (B3) (2008) B03209. doi:10.1029/2006JB004624.
- 482 [16] A. Jones, D. Thomson, M. Hort, B. Devenish, The U.K. Met Office’s Next-Generation At-
483 mospheric Dispersion Model, NAME III, in: C. Borrego, A.-L. Norman (Eds.), *Air Pollu-
484 tion Modeling and Its Application XVII*, Springer US, 2007, pp. 580–589. doi:10.1007/
485 978-0-387-68854-1_62.
- 486 [17] C. Searcy, K. Dean, W. Stringer, PUFF: A high-resolution volcanic ash tracking model, *Journal*
487 *of Volcanology and Geothermal Research* 80 (12) (1998) 1–16. doi:10.1016/S0377-0273(97)
488 00037-1.
- 489 [18] R. J. Brown, C. Bonadonna, A. J. Durant, A review of volcanic ash aggregation, *Physics and*
490 *Chemistry of the Earth, Parts A/B/C* 4546 (2012) 65–78. doi:10.1016/j.pce.2011.11.001.
- 491 [19] A. Costa, A. Folch, G. Macedonio, A model for wet aggregation of ash particles in volcanic
492 plumes and clouds: 1. Theoretical formulation, *Journal of Geophysical Research: Solid Earth*
493 115 (B9) (2010) B09201. doi:10.1029/2009JB007175.
- 494 [20] C. Bonadonna, J. C. Phillips, B. F. Houghton, Modeling tephra sedimentation from a Ruapehu
495 weak plume eruption, *Journal of Geophysical Research: Solid Earth* 110 (B8) (2005) B08209.
496 doi:10.1029/2004JB003515.
- 497 [21] C. Bonadonna, J. C. Phillips, Sedimentation from strong volcanic plumes, *Journal of Geophys-
498 ical Research: Solid Earth* 108 (B7) (2003) 2340. doi:10.1029/2002JB002034.
- 499 [22] S. Carey, R. S. J. Sparks, Quantitative models of the fallout and dispersal of tephra from
500 volcanic eruption columns, *Bulletin of Volcanology* 48 (2-3) (1986) 109–125. doi:10.1007/
501 BF01046546.
- 502 [23] C. Bonadonna, M. Pistolesi, R. Cioni, W. Degruyter, M. Elissondo, V. Baumann, Dynamics
503 of wind-affected volcanic plumes: The example of the 2011 Cordn Caulle eruption, Chile,
504 *Journal of Geophysical Research: Solid Earth* 120 (4) (2015) 2014JB011478. doi:10.1002/
505 2014JB011478.
- 506 [24] A. Costa, A. Folch, G. Macedonio, Density-driven transport in the umbrella region of volcanic
507 clouds: Implications for tephra dispersion models, *Geophysical Research Letters* 40 (18) (2013)
508 4823–4827. doi:10.1002/grl.50942.
- 509 [25] W. Degruyter, C. Bonadonna, Improving on mass flow rate estimates of volcanic eruptions,
510 *Geophysical Research Letters* 39 (16) (2012) L16308. doi:10.1029/2012GL052566.

- [26] D. Kunii, O. Levenspiel, Fluidization Engineering, Butterworth-Heinemann, 1991.
- [27] R. E. Holasek, S. Self, GOES weather satellite observations and measurements of the May 18, 1980, Mount St. Helens eruption, *Journal of Geophysical Research: Solid Earth* 100 (B5) (1995) 8469–8487. doi:10.1029/94JB03137.
- [28] A. Masselot, B. Chopard, A lattice Boltzmann model for particle transport and deposition, *EPL (Europhysics Letters)* 42 (3) (1998) 259. doi:10.1209/epl/i1998-00239-3.
- [29] K. Tsunematsu, J.-L. Falcone, C. Bonadonna, B. Chopard, Applying a Cellular Automata Method for the Study of Transport and Deposition of Volcanic Particles, in: H. Umeo, S. Morishita, K. Nishinari, T. Komatsuzaki, S. Bandini (Eds.), *Cellular Automata*, no. 5191 in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2008, pp. 393–400. doi:10.1007/978-3-540-79992-4_51.
- [30] K. Tsunematsu, B. Chopard, J.-L. Falcone, C. Bonadonna, Comparison of Two Advection-Diffusion Methods for Tephra Transport in Volcanic Eruptions, *Communications in Computational Physics* doi:10.4208/cicp.311009.191110s.
- [31] S. Suga, Numerical schemes obtained from lattice boltzmann equations for advection diffusion equations, *International Journal of Modern Physics C* 17 (11) (2006) 1563–1577. doi:10.1142/S0129183106010030.
- [32] T. Pöschel, T. Schwager, *Computational Granular Dynamics*, Springer, 2005. doi:10.1007/3-540-27720-X.
- [33] W. Petersen, P. Arbenz, *Introduction to Parallel Computing: A practical guide with examples in C*, Oxford Texts in Applied and Engineering Mathematics 9, 2004.
- [34] A. C. M. Volentik, C. Bonadonna, C. B. Connor, L. J. Connor, M. Rosi, Modeling tephra dispersal in absence of wind: Insights from the climactic phase of the 2450 BP Plinian eruption of Pululagua volcano (Ecuador), *Journal of Volcanology and Geothermal Research* 193 (12) (2010) 117–136. doi:10.1016/j.jvolgeores.2010.03.011.
- [35] C. Bonadonna, B. F. Houghton, Total grain-size distribution and volume of tephra-fall deposits, *Bulletin of Volcanology* 67 (5) (2005) 441–456. doi:10.1007/s00445-004-0386-2.
- [36] E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, A. Leetmaa, R. Reynolds, M. Chelliah, W. Ebisuzaki, W. Higgins, J. Janowiak, K. C. Mo, C. Ropelewski, J. Wang, R. Jenne, D. Joseph, The NCEP/NCAR 40-Year Reanalysis Project, *Bulletin of the American Meteorological Society* 77 (3) (1996) 437–471. doi:10.1175/1520-0477(1996)077<0437:TNYRP>2.0.CO;2.
- [37] W. Degruyter, C. Bonadonna, Impact of wind on the condition for column collapse of volcanic plumes, *Earth and Planetary Science Letters* 377378 (2013) 218–226. doi:10.1016/j.epsl.2013.06.041.
- [38] S. Scollo, M. Prestifilippo, M. Coltelli, R. A. Peterson, G. Spata, A statistical approach to evaluate the tephra deposit and ash concentration from PUFF model forecasts, *Journal of Volcanology and Geothermal Research* 200 (34) (2011) 129–142. doi:10.1016/j.jvolgeores.2010.12.004.

- 550 [39] B. Chopard, M. Droz, Cellular Automata Modeling of Physical Systems, Cambridge University
551 Press, Cambridge, 1998. doi:10.1017/CB09780511549755.
- 552 [40] K. Tsunematsu, New numerical solutions for the description of volcanic particle dispersal,
553 Ph.D. thesis, University of Geneva (2012).
554 URL <https://archive-ouverte.unige.ch/unige:26675>
- 555 [41] A. W. Woods, The fluid dynamics and thermodynamics of eruption columns, Bulletin of Vol-
556 canology 50 (3) (1988) 169–193. doi:10.1007/BF01079681.
- 557 [42] L. Wilson, G. P. L. Walker, Explosive volcanic eruptions - VI. Ejecta dispersal in plinian
558 eruptions: the control of eruption conditions and atmospheric properties, Geophysical Journal
559 International 89 (2) (1987) 657–679. doi:10.1111/j.1365-246X.1987.tb05186.x.