BALLISTA, a 3D ballistic trajectory simulator

User guide (Version 001)

Contents

1 Introduction

1.1 Ballistic projectiles in Volcanic Setting

Ballista is a numerical simulation program that models the trajectories and depositions of ballistic projectiles in volcanic settings.

Ballistic projectiles are volcanic pyroclasts which are transported in the air, decoupled from the gas phase at the early stage of transport, and follow independent parabolic trajectories. The diameters of these particles are generally larger than 64 mm and they are often called ballistic blocks or ballistic bombs.

Fig. 1-1: The 2015 strombolian eruption at the Central Crater of Aso Volcano in Japan. Photo by Yasuo Miyabuchi.

In strombolian eruption, pyroclasts released from the vent are rarely blown by wind and deposit around crater (e.g. Fig. 1). Therefore, these pyroclasts are classified as ballistic projectiles. Ballistic pyroclasts are also released in plinian and subplinian eruptions. Large pyroclasts separate from the eruptive column around jet region which is the lowest part of the volcanic plume in plinian and subplinian erutpions.

Ballista simulates trajectories and deposition positions of these pyroclasts in three dimensions. In order to talk about the physics and how this numerical model works, “pyroclasts” are expressed as “particles” after this.

This program accounts for multiple particles released at the same time and calculate their trajectory in three dimensions. Digital Elevation Model (DEM) is used for considering the topographic effect of deposition position.

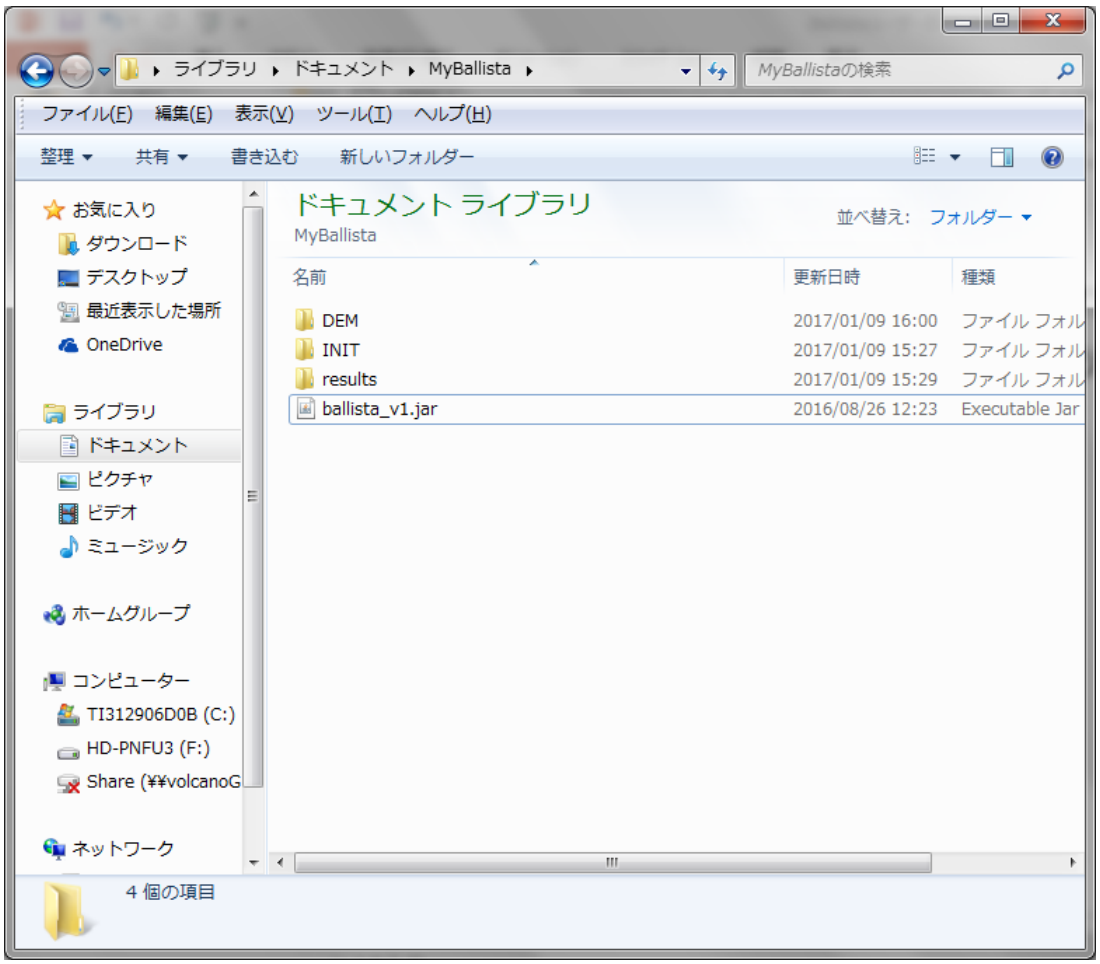
The system is developed based on Java technology and it is usable on any operating system. Furthermore, the graphical user interface (GUI) makes it easy to use without remembering complex commands.

1.2 About this manual

We show general instructions of Ballista with images of GUI in ***Chapter 2*** to have an overview of this software. Then, the details come afterwards. Preparation of the simulation is divided into two works. One is to setup the computer environment such as installation of Java Runtime Environment and place this Ballista program in your computer (***Chapter 3***). Another is to create a DEM file and write the input parameter file (***Chapter 4***). Input parameter definition and explanations are in ***Chapter 5.*** Users should make an input parameter file of their own problem referring the examples in this chapter. Users obtain two files after the simulation; **resdepo.txt** and **restraje.txt**. The former is the file including parameters related to deposition position and conditions of each particles, and the letter is the file of the time dependent trajectory. Definitions of output values are explained in ***Chapter 7***.

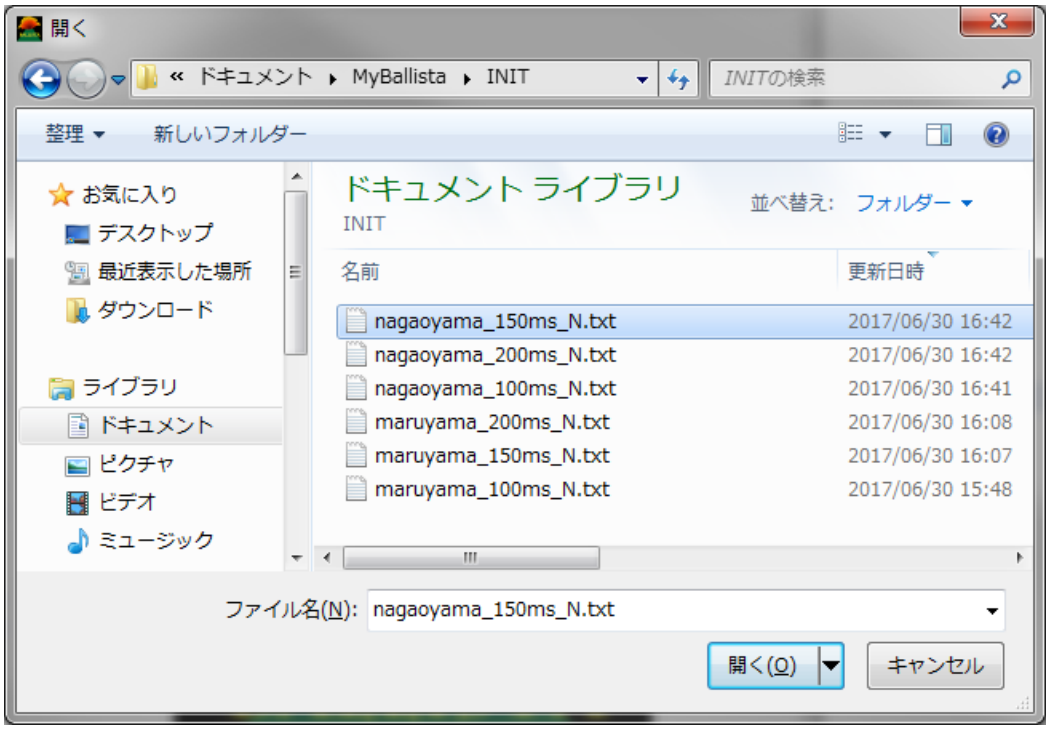
2 General Instructions/ Workflow

i. Install Java Runtime Environment and create directories as explained in section 3.
ii. Prepare an initial condition file, a Digital Elevation Model file and a center position file following the instructions in section 3 “Data Preparation”.
iii. To start Ballista, double click the file “Ballista_001.jar” (If you hide the extention, it is shown as “ballista_001”.) in MyBallista folder.

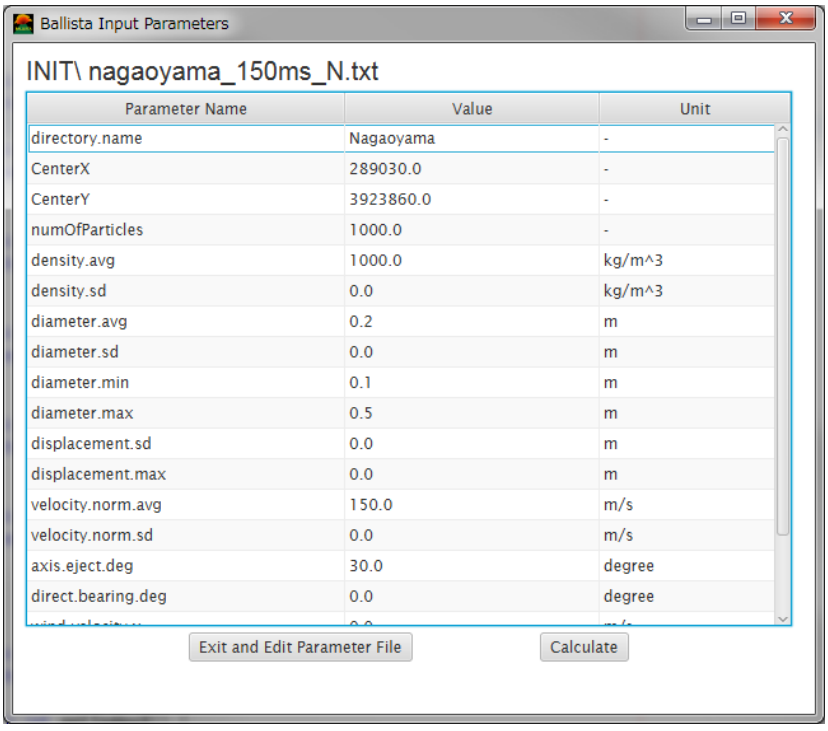


iv. Click the “Open File” button.

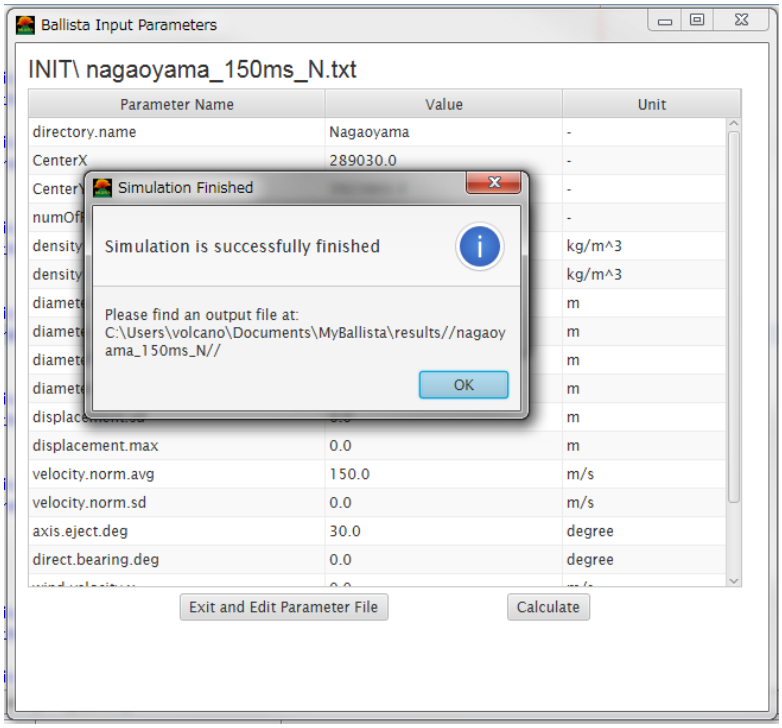
v. Choose the initial condition file designated extension ".txt".



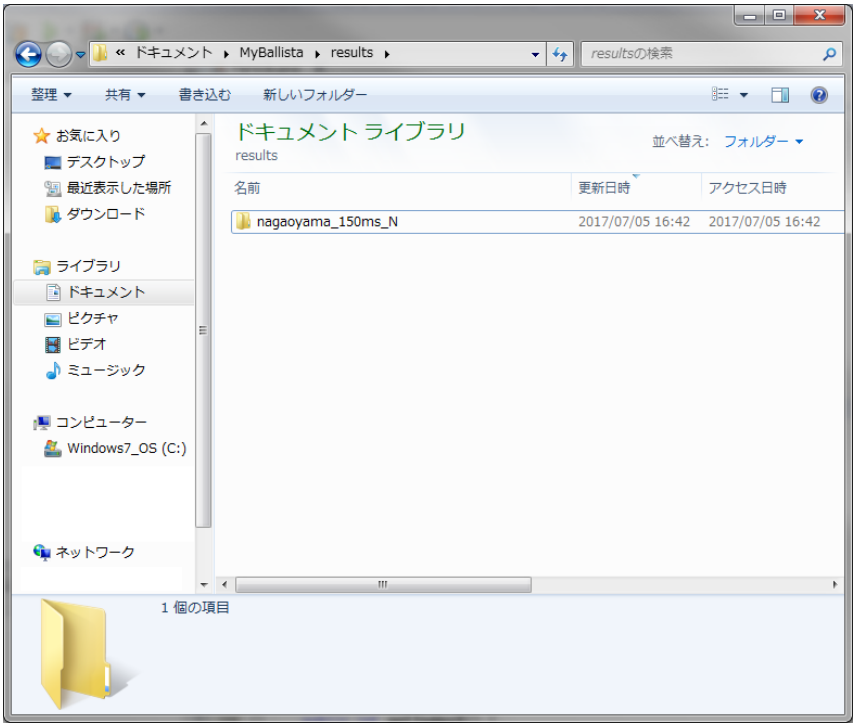
vi. You can see the parameters and their values on the list. Check if all the parameters are set correctl. If not, please close the **Ballista** by clicking the "Exit and Edit Parameter File" button and edit your initial condition file again. If parameters and values are correct, please click the "calculate" button to start the simulation.



vii. If the simulation finishes without any problems, it will show a pop-up saying “Simulation is successfully finished”. Check the path of the result file and close the simulator by clicking the “OK” button.



viii. You can find the result file in the “result” folder just under the “MyBallista” folder.



3 Installation and creating directories

3.1 Install Java Runtime Environment (JRE)

Ballista is a java based program and it functions on Windows, Linux and Mac.

3.2 Make the directory structures

Open the **"MyBallista"** folder from where you have saved it on your computer. It includes **ballista_001.jar** which is the executable file, and three directories "**DEM**", "**INIT**" and "**results**". If these directories do not exist, please create these folders and keep them together.

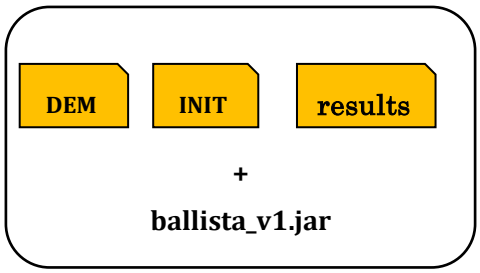


Fig. 3-1: File and directory structure.

- **DEM**: This directory includes Digital Elevation Model (DEM) files for including topography and a file for the vent location. Creating a new directory within the **DEM** directory that is named after that volcano e.g. **FujiDEM**. This way, DEM files from different volcanoes can be held in separate directories.
- **INIT:** This directory holds the input parameter files. These input parameter files should have an extension ".txt". Without this file extension, the programme will not recognize it as an input parameter file.
- **results:** This directory is empty before we run Ballista. The Ballista program will create output files automatically under this directory with the same name as the input parameter file used. Thus, you can recognise the results files of particular input parameters. In this directory, the system saves a deposition data file "**resdepo.txt**" and a trajectory data file "**restraje.txt**".
- ***Ballista_001.jar:*** This is the executable file of **Ballista**. It starts when you double click this file.

4 Data preparation

4.1 Initial Condition file

The initial condition file should be stored in the "INIT" folder just under the "MyBallista" folder. It should be a text file and named as "*example*.txt" (You can put any name instead of "*example*"). The name *example* will be the name of the folder which stores simulation result files.

e.g. Initial condition file name: *Fuji_150ms_N.txt*

→ Result folder name: *Fuji _150ms_N*

In the initial condition file, "a #" mark at the beginning of the line indicates the comment line. The value in this line will be ignored, so the parameter explanation is entered on this line.

The parameter values are set after the key of the parameter and a colon " : " mark.

e.g.

```
velocity.norm.avg: 150
```

In this example, the key of the parameter is "velocity.norm.avg" and the value is "150".

An explanation of each parameter is in section 5 "Input Parameters".

4.2 Digital Elevation Model (DEM)

Topography around the vent is given by a Digital Elevation Model (DEM) file in the ESRI ASCII raster format. Here, we introduce how to prepare a DEM file using gdal and QGIS system.

DEM preparation

DEM files are often provided in a Geotiff format so it is necessary to convert them into the ESRI ASCII raster file (extension: .asc).

There is a way to convert Geotiff files with QGIS (http://www.qgis.org/) or gdal (http://www.gdal.org). QGIS is free GIS software with GUI, which can be used on Mac/Windows/Linux. Gdal is a command based GIS software working on Linux.

i. With gdal command:

Type the following command to convert your Geotiff file in the Linux terminal on which the gdal software is installed.

% gdal_translate -of AAIGrid *INPUT*.tif *OUTPUT*.asc

INPUT.tif : a Geotiff file name to input

OUTPUT.asc: a name of the output file in ASCII format.

ii. With QGIS:

a) Launch QGIS.

b) Open your Geotiff file by drag-and-drop or from the menu → Layer → Add Layers → Add Raster Layer

c) Launch conversion tool from menu → Raster → Conversion → Type Conversion

Fig. 4-1: Display of QGIS for converting the Geotiff file to ASCII file.

d) Select the input layer (Geotiff) and output file name. The output file name should be set by clicking the "Select" button. Choose the "Arc/Info ASCII Grid (*.asc * ASC)" as a file type.

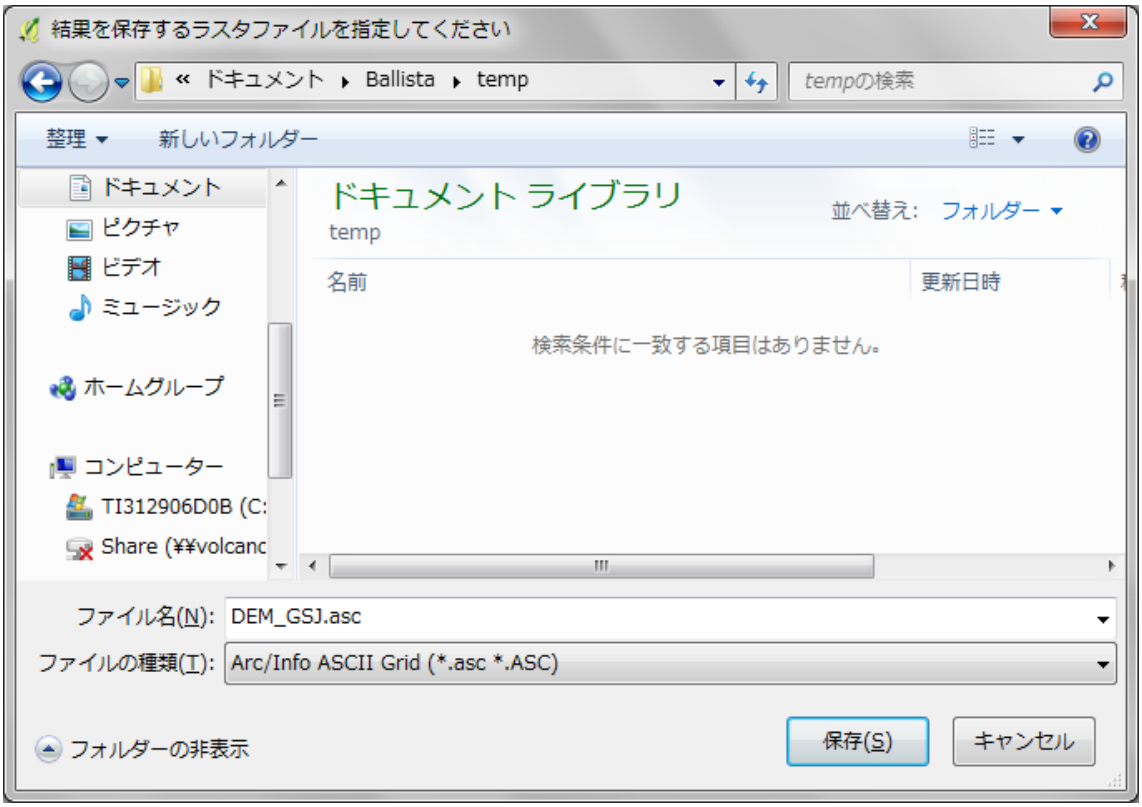


Fig. 4-2: Save the ASCII file into your folder

e) Click the "OK" button to start the conversion.

Please read a more detailed explanation on the QGIS tutorial website.

5 Input Parameters

Input parameter values can be input in several data types:

1) ***Int:*** The int data type represents an unsigned 32-bit integer.
2) ***Double:*** The double data type is a double-precision 64-bit IEEE 754 floating point. For decimal values, this data type is generally the default choice.
3) ***String:*** Literals of types, String may contain any Unicode (UTF-16) characters. String accepts multiple characters.

Please refer the Java website for further explanation (http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html).

5.1 Topography and Map

Parameters in Input file	Description
#Topography file directory directory.name	This directory includes a Digital Elevation Model (DEM) file for including topography and a file showing vent location. DEM files should be provided in ESRI ASCII Raster format. The DEM data should have an extension ".asc" and vent location files should have an extension ".txt" **Value Type**: String **Default value:** OntakeDEM **Unit**: -

5.2 Center position of the vent

Parameters in Input file	Description
# Center Position of the vent CenterX CenterY	"CenterX" is the X coordinate of the center position, and "CenterY" is the Y coordinate of the center position. These coordinate should be the same as the DEM coordinate. **Value Type**: Double **Default value:** CenterX: 0 CenterY: 0 **Unit**: It should be the same as DEM units. e.g.) If your DEM is based on the UTM coordinate system, the unit is meter. If the DEM is based on the WGS (latitude/longitude), the unit is degree

5.3 Particle Attributes

Parameters in Input file	Description
# Number of particles per burst numOfParticle.avg	Users can select any number of particles to emit from the vent. **Value Type**: double **Default value:** numOfParticle.avg: 100.0 **Unit**: -

Parameters in Input file	Description
# Particle density density.avg density.std	Particle density is set as a Gaussian distribution with average (density.avg) and standard deviation (density.std) values. If you would like to set the single value, please input only "density.avg" and put "density.std" as 0. **Value Type**: double **Default value:** density.avg: 2300 density.sd: 0.0 **Unit**: kg/m^3

Parameters in Input file	Description
# Particle diameter diameter.avg diameter.sd diameter.min diameter.max	Particle diameter is set as a Gaussian distribution with an average (diameter.avg) and a standard deviation (diameter.std). We can include a limit on particle diameter with minimum value (diameter.min) and maximum value (diameter.max). To set the single value, please input only "diameter.avg" and put "diameter.std" as 0. If the standard deviation is 0, then the simulator automatically ignores minimum and maximum values. **Value Type**: double **Default value:** diameter.avg: 0.5 diameter.sd: 0.1 diameter.min: 0.0 diameter.max: 1.0 **Unit**: metre

5.4 Ejection Location

Parameters in Input file	Description
#Displacement of the launch point from the vent centre displacement.avg displacement.sd displacement.max	Assuming the vent is a circle, the displacement is a distance from the centre of the vent where the particles exit. Particles therefore do not all exit from a single point but from a wider area (see Fig. 1). **Value Type**: double **Default value:** displacement.avg: 0.0 displacement.sd: 15.0 displacement.max: 10.0 **Unit**: metre

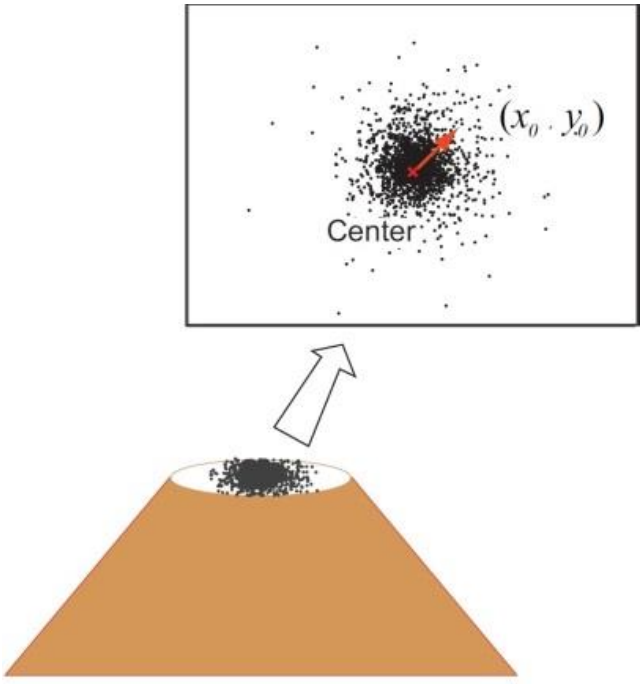


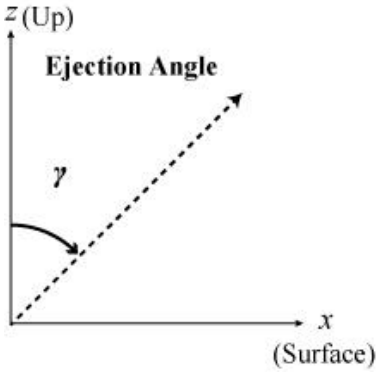
Fig. 5-1: Displacement of ejection points from the vent centre.

5.5 Ejection Conditions

Parameters in Input file	Description
#Particle ejection velocity velocity.norm.avg velocity.norm.sd	Particle ejection velocity $v = \sqrt{v_x^2 + v_y^2 + v_z^2}$. The value is picked randomly from a Gaussian distribution using an average (velocity.norm.avg) and standard deviation (velocity.norm.sd). **Value Type**: double **Default value:** velocity.norm.avg: 100 velocity.norm.sd: 0.0 **Unit**: m/s

Parameters in Input file	Description
#Ejection angle axis.eject.deg	Ejection angle is the angle between the axis of ejection and the vertical. If the ejection angle is 0 then the eruption is vertical. **Value Type**: double **Default value:** axis.eject.deg: 20.0 **Unit**: degree

Parameters in Input file	Description
#Direction bearing direct.bearing.deg	Direction bearing is the bearing from north reflecting the ejection direction from the vent (e.g. an eruption directed to the NE would have a bearing of 45°). **Value Type**: double **Default value:** direct.bearing.deg: 20 **Unit**: degree



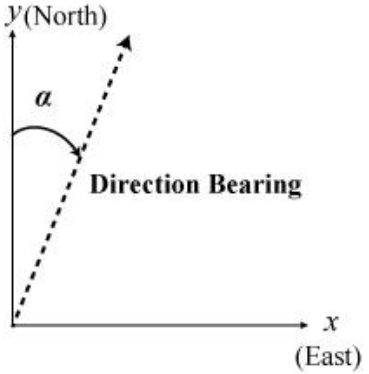


Fig. 5-2 (a) Ejection Angle *(b) Direction bearing*

Parameters in Input file	Description
#Wind Velocity wind.x wind.y	Wind velocity is a constant wind velocity during the simulated ballistic transport. This wind is applied everywhere in the simulation sites and the value cannot be changed. **Value type**: double **Default value:** wind.x: 0.0 wind.y: 0.0 **Unit**: m/s

5.6 Particle Drag Coefficient

Parameters in Input file	Description
#Drag Coefficient dragCoefficient.cnst	Drag Coefficient is a constant for defining the particle drag when it travels in the air. For the detail description and equation, please refer Tsunematsu et al (2016). **Value type**: double **Default value:** dragCoefficient.cnst: 0.8 **Unit**: -

6 Output files, parameters and definitions

Ballista saves two files "resdepo.txt" and "restraje.txt" in the **results** directory. They are in a text delimited with tabs format. These files can be opened by Excel or text editors. The **resdepo** file holds all the particle deposition data which can be turned into a spatial distribution map. The **restraje** file holds all the particle trajectory files and can be imported into Matlab/Octave to create trajectory figures.

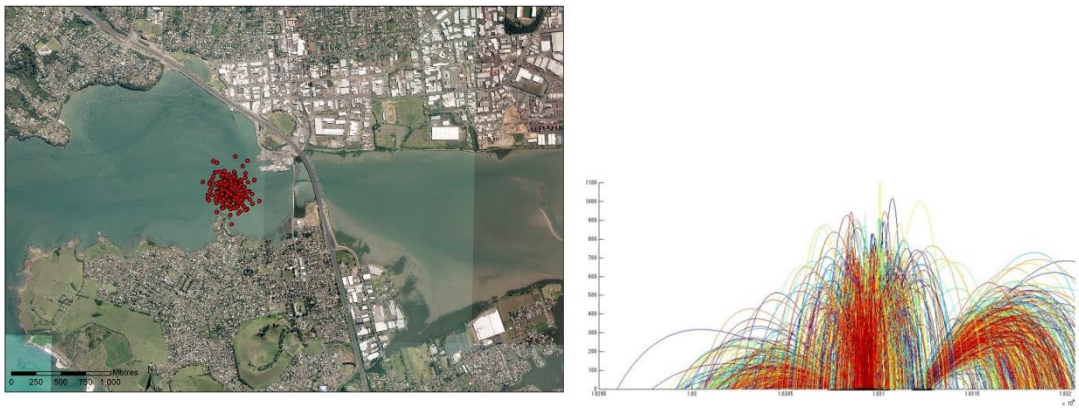


Figure 6-1. Left: Map created using resdepo file. Right: Image created using restraje file.

6.1 Deposition data

Included in the **resdepo** file:

Column Number	Parameter Name (Header description)	Explanation
1	x	Coordinate of location where the particle deposit in ***meter***. It uses the same coordinate system as DEM.
2	y	
3	z	
4	Distance from the vent	Horizontal Distance calculated by Distance = $\sqrt{x^2 + y^2}$ in ***meter***.
5	Particle mass	Mass is calculated by Mass = $\frac{Di^3\pi}{6} \times \rho_p$ in ***kg***.
6	Particle velocity when it impacts the surface	This is the norm of velocity calculated by $\|V\| = \sqrt{V_x^2 + V_y^2 + V_z^2}$ in ***m/s***.
7	Particle density	Density of the particle. If it is not randomly generated, it is as the same as the input value in $\boldsymbol{kg/m^3}$.
8	Particle diameter	Diameter of the particle. If it is constant,

		it is as the same as the input value in ***meter***.
9	Particle id	Particle id is automatically generated by **Ballista** in order to have a reference with trajectory data.

6.2 Trajectory data

Included in the **restraje** file:

Column Number	Parameter Name (Header description)	Explanation
1	Time	Time after the event starts in ***second***.
2	Particle id	Particle id is automatically generated by **Ballista** in order to have a reference with deposition data.
3	x	Coordinate of location when the particle the particle travels in the air or deposit in ***meter***.
4	y	
5	z	