



UNIVERSITÉ  
DE GENÈVE

FACULTY OF SCIENCES  
Department of Earth Sciences

---

# The Great Balls of Fire

Sebastien Biass<sup>1</sup>  
Jean-Luc Falcone<sup>2</sup>  
Costanza Bonadonna<sup>1</sup>

1. *Department of Earth Science, University of Geneva*
2. *Computer Science, University of Geneva*

## User manual

November 2015

---

# Contents

<b>1</b>	<b>Bomb simulator</b>	<b>1</b>
1.1	Requirements . . . . .	1
1.2	Getting started . . . . .	1
1.2.1	Binary package . . . . .	1
1.2.2	Building from source . . . . .	1
1.3	Running GBF . . . . .	1
1.4	Config file . . . . .	2
1.4.1	Syntax . . . . .	2
1.4.2	Parameters . . . . .	3
1.5	Output . . . . .	4
<b>2</b>	<b>Post-processing</b>	<b>4</b>
2.1	Running the post-processing . . . . .	4
2.2	Displaying results . . . . .	5
2.3	Exporting results . . . . .	6
2.4	Dependencies . . . . .	6
<b>3</b>	<b>FAQ</b>	<b>6</b>
3.1	How to cite ? . . . . .	6
3.2	What language/libraries where used to design GBF ? . . . . .	6
3.3	How to fix the (density, size, ejection speed, ejection angle) ? . . . . .	7
3.4	How to perform a sensitivity analysis ? . . . . .	7
3.5	How to run larger simulations ? . . . . .	7
3.6	Do you support distributed memory parallelism ? . . . . .	7
3.7	How to request for feature or report a bug ? . . . . .	7
3.8	How may I request for help ? . . . . .	7
3.9	Do you accept contribution ? . . . . .	7
3.10	What is the GBF license ? . . . . .	8

---

# 1 Bomb simulator

## 1.1 Requirements

The *Great Balls of Fire* model (**GBF**) is executed on the Java virtual machine (version 8). It is then compatible with any OS which supports Java, notably GNU/Linux, Apple MacOSX and MS Windows.

The computation is mainly CPU intensive, as each bomb trajectory is computed independently. Thus, only a limited amount of memory is required even for large simulations (1GB should be enough). Multicore CPU or multiprocessor machines will however greatly speed up the computation.

## 1.2 Getting started

As said above, the only external requirement is the Java Virtual Machine (JVM) version 8. A suitable version can be downloaded at Oracle website.

### 1.2.1 Binary package

**GBF** is provided as a "fat" *jar* containing all necessary libraries. It can be downloaded directly at the **GBF** download page. Just copy the `.jar` file in a working directory, without any further operations. Similarly, it can be "uninstalled" by deleting the `.jar` file.

### 1.2.2 Building from source

**GBF** can be built from the source code. You should install first the `sbt` script, instruction are available on the SBT documentation. A copy of the source code can be obtained on the GBF project page. If you use GIT, you can clone the repository using:

```
$ git clone https://github.com/unigeSPC/gbf
```

If you download the source manually, uncompress it into a fresh directory. Then input the following commands:

```
$ cd simulator/  
$ sbt  
> compile  
> oneJar
```

The resulting `gbf_2.11-yyy-one-jar.jar` file will be located in the `target/scala-2.11/` directory, where `yyy` is the GBF version. Note that the first time the project is built, all necessary libraries will be downloaded. This may take several minutes, but successive builds will be much faster.

## 1.3 Running GBF

**GBF** is a command line only software. To run it you need to first write a configuration file and the result will be stored in an output file. Both file syntax are described in the following sections. We recommend copying the example in `.` To run the **GBF** model, type:

```
java -jar <gbf.jar> <conf file> <numWorkers>
```

where `<gbf.jar>` is the Jar file downloaded or built as described in previous section; `<conf file>` is the configuration file and `<numWorkers>` is the number of threads used in the computation.

**Multithread execution** **GBF** is able to exploit the full power of a multicore/multiprocessor architecture. In order to reach best performances, the command line parameter `numWorkers` must equal the number of physical cores in your machine. Exceeding this number will result in a slow down. It is advised to disable hyper-threading if possible.

---

## 1.4 Config file

### 1.4.1 Syntax

**Types** The configuration types are:

- **section**: configuration sub-section. See below.
- **integer**: integer number (max: 2,147,483,647)
- **float**: floating point number (double precision)
- **string**: string of characters. Must be double quoted.

**Sections** Configuration parameters are organized into hierarchical sections. Section can be defined either using braces or by prefixing parameters with section names (separated using a dot). For instance, both examples below are equivalent:

```
#Example 1
terrain {
    demFile = "dem/dem_10m.txt"
    vent {
        E = 496682.0
        N = 4250641.0
        altitude = 400
    }
}
```

```
#Example 2
terrain.demFile = "dem/dem_10m.txt"
terrain.vent.E = 496682.0
terrain.vent.N = 4250641.0
terrain.vent.altitude = 400
```

**Include** It is possible to add an **include** directive at the start of a config file referring to another config file. For instance:

```
#In file basic.conf
wind {
    speed = 10
    direction = 12.5
}

#In file highWind.conf
include "basic.conf"
wind.speed = 90
```

In this example, whenever **highWind.conf** is used as an input file, the **basic.conf** parameters will be used except when redefined in **highWind.conf**. Both files are thus equivalent to:

```
wind {
    speed = 90
    direction = 12.5
}
```

**System properties** To simplify parameter scan such as sensitivity analysis, parameters can also be redefined as command line options. Relevant parameters must be prefixed with **gbf**. For instance, using bash shell:

```
java -Dgbf.wind.speed=40 -jar gbf_0.0.1.jar basic.conf 4
```

the wind speed will be set to 40 m/s. Command line definitions always take precedence on configuration files.

---

### 1.4.2 Parameters

Here are the parameters used in the configuration file. Types between parenthesis are defined above:

- **rng**: random generator (section)
  - **seed**: the random generator seed (integer). Using the same seed, with the same configuration option, will produce exactly the same output. Useful for reproducibility
- **terrain**: terrain description (section)
  - **demFile**: DEM terrain file (string) in a ArcMap Ascii Raster format
  - **vent**: vent location (section). All following options are in meters
    - \* **E**: vent easting (float)
    - \* **N**: vent northing (float)
    - \* **altitude**: vent altitude (float)
- **source**: source parameters (section)
  - **densAvg**: density average in kg/m<sup>3</sup> (float)
  - **densStd**: density standard deviation in kg/m<sup>3</sup> (float)
  - **phiAvg**: size average in  $\Phi$  units (float)
  - **phiStd**: size standard deviation in  $\Phi$  units (float)
  - **velocityAvg**: velocity average in m/s (float)
  - **velocityStd**: velocity standard deviation in m/s (float)
  - **inclinationAvg**: inclination average in degree (float). A perfectly vertical inclination is equal to 0
  - **inclinationStd**: inclination standard deviation in degrees (float)
- **wind**: wind parameters (section)
  - **speed**: wind speed in m/s (float)
  - **direction**: wind direction in degrees (float). A direction of 0 corresponds to a wind coming from North
- **drag**: parameters used in drag computation (section)
  - **timeStep**: time step used for trajectory integration in seconds (float)
  - **pressure0**: pressure at sea level in Pa (float)
  - **temp0**: temperature at sea level in Kelvins (float)
  - **thermalLapse**: thermal lapse in Kelvins/meter (float)
  - **reducedDragRadius**: radius of the disc area centred on the vent, where drag is reduced, in meters (float)
- **experiment**: experiment parameters
  - **size**: number of bombs in the experiment (integer)
  - **outputFile**: file name of the output file (string). If the path of the output file points to a different directory, the location should exist before running the simulation.

---

## 1.5 Output

The output is written to the file defined in the `experiment.outputFile` parameter. Every line of these file represents a different bomb impact. Characteristics of the impact are separated by a single space. Columns are:

- 1: Easting in meters.
- 2: Northing in meters.
- 3: Bomb mass in kg
- 4: Bomb diameter in meters.
- 5: Kinetic energy at impact in kJ
- 6: Incidence angle at impact in degrees. A perfectly vertical impact has an incidence angle of 0.
- 7: Bomb ejection angle in degrees. A perfectly vertical ejection has an angle of 0.
- 8: Bomb flight time in seconds.

## 2 Post-processing

The post-processing of the **GBF** output is achieved using the `process_GBF.m` *Matlab* script. Additionally, results can be visualized using the `display_GBG.m` script. As thoroughly described in the companion paper, two approaches are used to transform the results from the **GBF** model (i.e. discrete impacts) into probabilities of impacts to exceed hazardous values of kinetic energies.

The first approach, referred to as **pixel-based** approach, considers VBP impacts on an equally-spaced grid for each pixel of area  $A_{i,j}$  in order to quantify the probability of occurrence a VBP of a given energy threshold ( $E_T$ ) in a given pixel:

$$P(A_{i,j}, E_T) = \frac{\sum VBP_{A_{i,j}, E_T}}{n_{VBP}}, \quad (1)$$

where  $n_{VBP}$  is the total number of simulated VBPs. This approach is also used to compute the energy occurring in a pixel for a given probability of occurrence. Over the total number of VBPs impacting a given pixel, an empirical survivor function of the kinetic energy is created, from which energy is expressed as a function of probability. Two caveats should be considered when using this approach. First, since the  $n^{th}$  percentile returns the lowest  $n\%$  of the population, there is a  $100 - n\%$  probability that the energy will exceed the energy given by the  $n^{th}$  percentile. Second, such energies are calculated over individual pixels and do not convey any notion of probability of occurrence compared to the total number of simulated VBPs. Energies should thus be associated with their respective probabilities of occurrence (i.e. number of VBPs impacting each pixel normalized over the total number of simulated VBPs).

The second approach, referred to as **zone-based** approach, considers VBP impacts in a zone of interest  $Z$ , which can be defined either as a concentric “donuts” or as radial sectors around the vent. Probabilities of VBPs to exceed given energy thresholds  $E_T$  can then be expressed as normalized either on the total number of VBPs simulated or on the number of VBPs that fell in a given zone  $Z$ . In the first case,  $P(Z, E_T)$  answers the question “what is the probability of a VBP to exceed a given energy threshold  $E_T$  in a zone  $Z$ ?”. In the second case,  $P(E_T|Z)$  answers the question “knowing that a VBP impacts the zone  $Z$ , what is its probability to exceed an energy threshold  $E_T$ ?”.

### 2.1 Running the post-processing

To run the post-processing, open *Matlab* and navigate to the folder where the `process_GBF.m` is located. Start the script by typing:

```
>> process_GBF
```

You are asked to fetch the `.dat` output file of the **GBF** model, after which a window appears requesting you do fill the following parameters:

- 
- Name:** Your run name, which will be used to create an output folder and name output files. In the case a folder with a similar name already exists, a warning will appear
  - Grid resolution:** The grid size used for the pixel-based approach;
  - Energy thresholds:** Energy thresholds (J) used for computing probability maps. Separate different entries by comas;
  - Probability thresholds:** Probability thresholds (%) used for computing probabilistic energy maps. Separate different entries by comas;
  - Distance interval:** Interval (m) used to compute concentric probabilities around the vent. Separate different entries by comas;
  - Radial interval:** Interval (degrees) used to compute radial probabilities around the vent. Separate different entries by comas;
  - Vent easting:** Easting coordinate (UTM) of the vent. Should be expressed as the zone specified in **UTM zone**;
  - Vent northing:** Northing coordinate (UTM) of the vent. Should be expressed as the zone specified in **UTM zone**;
  - UTM zone:** UTM zone of the erupting vent.

The post-processing reads all VBP impacts modelled with the **GBF** model and computes the various probabilities described above. An output folder is created, which contains the following files, where:

- name.mat:** Main project file, where **name** is the run name previously defined;
- en\_PRB%.txt:** Energy for a probability of occurrence of PRB (Pixel-based approach; ESRI ASCII Raster format);
- nb\_part.txt:** Number of VBPs in each pixel (Pixel-based approach; ESRI ASCII Raster format);
- prob\_pixel\_EN.txt:** Probability (%) of occurrence of VBPs exceeding an energy threshold EN (Pixel-based approach; ESRI ASCII Raster format);
- prob\_dist\_all\_EN.txt:** Probability  $P(Z, E_T)$  (%) at a given distance from the vent (Zone-based approach; ESRI ASCII Raster format);
- prob\_dist\_zone\_EN.txt:** Probability  $P(E_T|Z)$  (%) at a given distance from the vent (Zone-based approach; ESRI ASCII Raster format);
- prob\_radial\_all\_EN.txt:** Probability  $P(Z, E_T)$  (%) at a given radial sector around the vent (Zone-based approach; ESRI ASCII Raster format);
- prob\_radial\_zone\_EN.txt:** Probability  $P(E_T|Z)$  (%) at a given radial sector around the vent (Zone-based approach; ESRI ASCII Raster format).

## 2.2 Displaying results

Results of the post-processing can be easily plotted using the dedicated GUI proposed in the **display\_GBF.m**. Open *Matlab* and navigate to the folder where the **display\_GBF.m** is located. Start the script by typing:

```
>> display_GBF
```

You are first prompted to select the **.mat** file created by the post-processing function. Once selected, a new figure opens, allowing to create the following plots:

- **General**
  - **Matrix:** Plots all output parameters (Section 1.5) against each other;
  - **Scatter:** 3D scatter plot of VBPs based on easting, northing and landing altitude. An extra panel opens allowing to plot a fourth parameter as a color gradient;
  - **Energy vs distance:** Plots i) the median energy and ii) the number of VBPs with distance from the vent;

- **Histograms**: Plots the results of the zone-based approach as histograms;
- **Number of VBPs**: Plots the map of number of VBPs obtained with the pixel-based approach;

- **Probability maps**

- **Pixel**: Plots probability maps obtained with the pixel-based approach;
- **Distance  $P(E_T|Z)$** : Plots the map of  $P(E_T|Z)$  obtained with the zone-based approach on concentric donuts;
- **Distance  $P(Z, E_T)$** : Plots the map of  $P(Z, E_T)$  obtained with the zone-based approach on concentric donuts;
- **Radial  $P(E_T|Z)$** : Plots the map of  $P(E_T|Z)$  obtained with the zone-based approach on radial sectors;
- **Radial  $P(Z, E_T)$** : Plots the map of  $P(Z, E_T)$  obtained with the zone-based approach on radial sectors;

- **Energy maps** Plot energy maps for a given probability of occurrence obtained with the pixel-based approach.

Two additional options can be used to customize plots. Firstly, some plots, such as the above-defined *Matrix* and *Scatter* plots can become very heavy if large sets of VBPs were simulated. In this case, the *Subset* option allows to plot only a given percentage of the total dataset. Secondly, some parameters of interest can span different orders of magnitude. In this case, it can be useful to use the *Log10* option to plot parameters using a logarithmic scale.

## 2.3 Exporting results

Output files created with the post-processing function are formatted as ESRI ASCII Rasters and can be easily imported in most GIS platforms. For more details on this format, see the ESRI Documentation. To save figures obtained with the display GUI, it is strongly recommended to use the *export\_fig* function available on the Matlab File Exchange website (contribution 23629).

## 2.4 Dependencies

The post-processing and display functions use two additional functions available on the Matlab File Exchange website. All credits go to their respective authors!

- **utm2deg** by Rafael Palacios (contribution 10914)
- **plot\_google\_map** by Zohar Bar-Yehuda (contribution 27627)

# 3 FAQ

## 3.1 How to cite ?

The GBF software was first described in:

Sébastien Biass, Jean-Luc Falcone, Costanza Bonadonna, Federico Di Traglia, Marco Pistolesi, Mauro Rosi, Pierino Lestuzzi, *Great Balls of Fire: A probabilistic approach to quantify the hazard related to ballistics — A case study at La Fossa volcano, Vulcano Island, Italy*, Journal of Volcanology and Geothermal Research, Volume 325, 1 October 2016, Pages 1-14, <http://dx.doi.org/10.1016/j.jvolgeores.2016.06.006>

## 3.2 What language/libraries where used to design GBF ?

**GBF** was written in the Scala 2.11 language. It uses Akka to parallelize trajectory computation.



---

### 3.3 How to fix the (density, size, ejection speed, ejection angle) ?

Bomb and ejection parameters are defined by providing a average and a standard deviation. Values will be drawn at random for each bomb. A parameter can be made constant by setting the standard deviation to 0.

### 3.4 How to perform a sensitivity analysis ?

You can override any parameter in the command line. You can then write a script to launch several execution. For instance if you want to try several wind speed values using bash, you can use the following snippet:

```
wind="0 10 25 50 100"
for w in wind; do
  out="impacts_${w}.dat"
  java -Dgbf.wind.speed=$w -Dgbf.experiment.outputFile=$out \
    -jar gbf.jar default.conf 4
done
```

Pay attention to use a different output files (provided here by the `-Dgbf.experiment.outputFile` option).

### 3.5 How to run larger simulations ?

Since the number of bombs launched in a simulation is limited by the maximum size of an integer, you can run bigger simulation by launching several execution with a different seed and a different output file. You can then pool the result by concatenating the output files. On unix (Linux or MacOSX):

```
$ cat output1.dat output2.dat output3.dat > pooled_output.dat
```

### 3.6 Do you support distributed memory parallelism ?

No. Only shared memory is currently supported. However you can launch several jobs on several nodes (using a different seed each time). The results can be pooled as described in previous answer.

### 3.7 How to request for feature or report a bug ?

You can use our issue tracker. However, because our time is unfortunately limited we will mainly focus on correcting bug and we may be reluctant to add new features.

### 3.8 How may I request for help ?

Well, this guide should be enough. If a section is not clear, feel free to report an issue as described in the previous answer. We will try to improve its content whenever possible.

We do not plan to provide individual help, except in the context of a scientific collaboration. Feel free to contact the paper authors if you are interested in such collaboration.

### 3.9 Do you accept contribution ?

Of course. If you plan to collaborate, feel free to fork the project and send us a pull request with your modifications.

---

## 3.10 What is the GBF license ?

**GBF** is free and open source software. It is licensed with the GPL 3 license. Refer to the provided `LICENSE.txt` file for the exact details. In layman terms you can:

- Freely download and use the software and its source code.
- Redistribute the software, under the condition you keep the same license and you redistribute the source code.
- Modify the software or integrate it in another project. However if you plan to redistribute the result, you should license it under a compatible license.

However not covered by GPL, we ask you to cite the original paper if you publish data generated with **GBF**.