UNIVERSITÉ
DE GENÈVE

FACULTY OF SCIENCES
Department of Earth Sciences

The Great Balls of Fire

Sebastien Biass[1]
Jean-Luc Falcone[2]
Costanza Bonadonna[3]

1. *Earth Observatory, NTU, Singapore*
2. *Computer Science, University of Geneva, Switzerland*
3. *Department of Earth Science, University of Geneva, Switzerland*

User manual

April 2018

Contents

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 muliprocessor 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 dowloaded 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 ???????
```

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

```
$ sbt
> compile
> oneJar
```

The resulting `.jar` file will be located in the `target/scala-2.11/` directory. 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 ouptut 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) *Note:* The *source* can be conceptually depicted as a cone defined by an angle from a horizontal plane centered on the vent (`tilt`), a geographic azimuth from the North (`azimuth`) and a spread around the vector defined by `tilt` and `azimuth` (`spread`).
  - densAvg: density average in kg/m^3 (float)
  - densStd: density standard deviation in kg/m^3 (float)
  - phiAvg: size average in Φ units (float)
  - phiStd: size standard deviation in Φ units (float)
  - velocityAvg: velocity average in m/s (float)
  - velocityStd: velocity standard deviation in m/s (float)
  - azimuth: azimuth of the center of the ejection cone in degrees from North (float) (e.g. West = 270)
  - tilt: inclination average in degree from a horizontal plane centered on the vent (float). A perfectly vertical inclination is equal to 0
  - spread: spread of the ejection cone in degrees from the center of the cone (float). Used as one standard deviation to sample a deviation from the center of the cone from a Normal distribution (e.g. a spread of 20 degrees means that ~68% of VBPs will be ejected within $\pm$40 degrees from the center of the vector defined by `tilt` and `azimuth`)
- **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 ??? (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 model output is extensively described on the following blog post: https://e5k.github.io/codes/2017/10/09/ballistic-post-processing/

2.1 Exporting results

Output files created with the post–processing function are formated 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).

2.2 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 ?

Biass, S., Falcone, J.-L., Bonadonna, C., Di Traglia, F., Pistolesi, M., Rosi, M., Lestuzzi, P., 2016. Great Balls of Fire: A probabilistic approach to quantify the hazard related to ballistics — A case study at La Fossa volcano, Vulcano Island, Italy. J. Volcanol. Geotherm. Res. 325, 1–14. doi: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 followin 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 concatening the output files. On unix (Linux or MacOSX):

```
$ cat output1.dat output2.dat output3.dat > pooled_ouput.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 unfotunately 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**.