# “Tracking a biological weapon fallout”: code development

The initial problem states that a biological weapon has been placed on top of a building and this will spread bacteria into a certain area. There is a set of rules that dictate the way each particle will be spread depending on the height of where the weapon was placed and wind conditions, both based on probabilities.

The aim of the code is to:

1. Plot and locate the building where the weapon was placed
2. Simulate the spread of particles
3. Plot their final location as a density map
4. Save the output map as a text file

## Data

The only input data used for this code was a raster file named **bomb.txt** in the code folder. This file is a grid of numbers from 0 to 255 representing the location of the building, identified with the number 255.

In addition to the input file, three libraries were used to develop the code:

1. **Random**. Used to generate random (decimal or float) numbers between 0 and 1.
2. **Matplotlib**. This library was used for the graphical representations of the raster file.
3. **Csv.** The csv library was used to read (open) the input file and write the outputs from the model.

## Building the code

Before building the code itself it was important to identify the elements provided. In the first place, the raster provided would be used as the **environment** where our agents will interact. This environment comprised a grid of numbers, all of them with the value of 0 but one where the building (weapon) was located, represented with a value of 255.

Subsequently, we needed to build the agents that will interact with the environment. In this case there is no specification for the bacteria particles to interact with each other. Therefore, each particle will represent an agent with an independent trajectory set by the probabilities given in the problem specification.

The direction of their trajectory depends on the following probabilities:

* A 5% chance for the particle to head west.
* A 10 % chance for the particle to head either north or south.
* A 75% chance for the particle to head east.

The falling rate will depend on the height were the weapon is located. At the beginning we are told that the weapon is placed on top of a 75m height building. The falling rate changes depending on whether the bacteria start falling above that height.

If the height value is higher than 75:

* A 20% chance that the particle will raise by one metre per second due to air turbulence.
* 10% chance that it will stay at the same level.
* A 70% chance that it will fall (1m/s).

If the height is equal or less than 75:

* The particle will fall one metre per second.

For this assignment, one iteration of the model is equal to one second.

These sets of rules were then used to build the agent framework. This can be found at **agent\_particle.py.** The agent framework consists on the creation of the agent itself containing three attributes: a coordinate x, coordinate y and the particle’s height. These three elements will be obtained further from specific variables in the model. The aim is to create a set of particles all with a specific location and height before the algorithm is run. Taking into account the set of rules previously given, we create two methods for our Particle class: direction and fall.

The **direction** method will create a local variable that generates a random decimal number between 0 and 1 and the direction will be given depending on the value of such number.

The **fall** method will depend on the current height of the particle. If this is bigger than 75, then a random number will be generated and its height will depend on the probabilities given before. If its current height is lower, it will fall at a constant rate. Note that the height of a certain particle is updated with each iteration.

The model is divided in sections where 1 corresponds to preliminary processes including file inputs, 2 the algorithm and model itself and 3 to the output section. The first section of the code (1.1) is used to read the environment, i.e. the raster file. Subsequently, 1.2 reads every line in the environment in order to find the values that are different from 0, i.e. the location of the building. Once the latter is located, it prints a message detailing the and coordinates. These coordinates are stored in two different variables, one for x and one for y. These will be used later to set the starting point of the bacteria.

Section 2.1 is where the parameters of the model are specified. As can be seen, the starting height of the bacteria is a variable that allows the user to change it accordingly. This was coded in this way since the problem itself doesn’t specify a height other than the building (75m) and the fall rate includes a different behaviour for a height higher than this. This also allows the user to try different heights and explore the possibilities of the bacteria spread. The height is set by default at 76m. The next parameter is the number of particles wanted in the model. The problem specifies a quantity of 5,000 particles, however, this can be changed if required.

The model starts with the creation of a list of agents (2.2). For the creation of each, the code takes the initial height specified in the parameters section and the x and y variables that store the location of the building. In section 2.3, the simulation of the bacteria takes place. Each particle will be randomly moved (depending on the pre-set probabilities) both in direction and fall until its height reaches 0, at this point the code will move on to the next particle. The simulation stops when all the particles have reached the ground.

Section 3.1 of the code simply creates a list of the final location of each of the particles by extracting the x and y coordinates information stored in each agent plus an identification number. This information is placed in a list and section 3.2 creates a text file where these can be accessed.

The density plot is created in section 3.3 by simply increasing the original environment value by 5 where a certain particle has fallen. Therefore, the more particles that fall in a certain location, the greater the value it will contain. When this is plotted we obtain an image similar to the one shown in Figure 1. As can be seen, the highest concentrated spots are represented with a yellow colour. Finally, section 3.4 writes a text file where the output density map is saved.

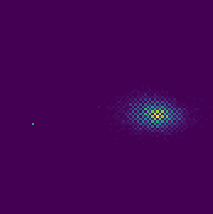
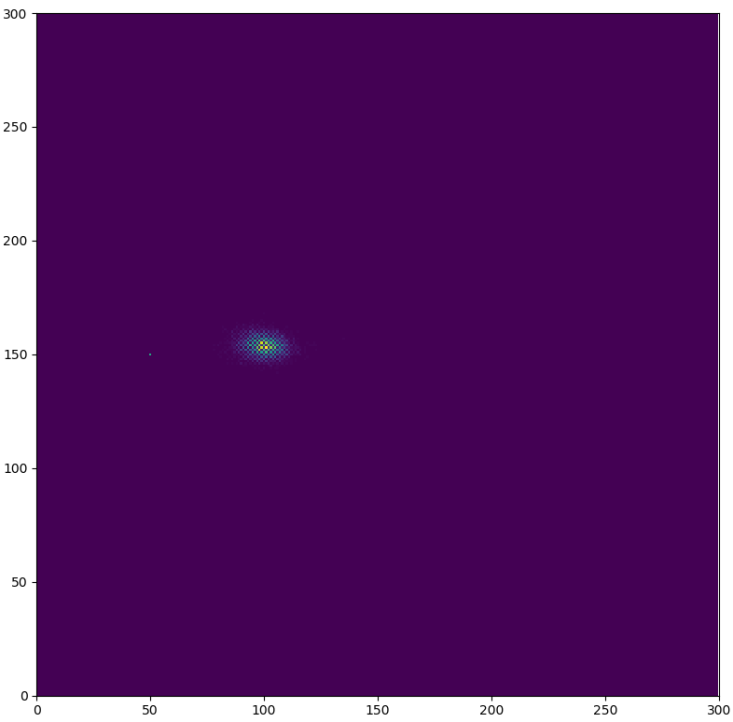


Figure 1. Density map (output).

## Conclusion

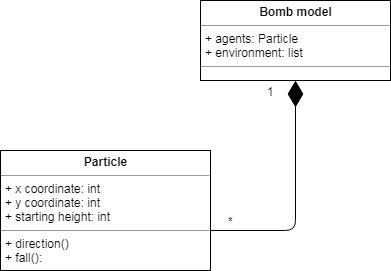
The code solves the “biological weapon” problem and allows the user to edit the height and the number of particles while it still performs properly. However, a sensitivity analysis on the processing time or an evaluation of the algorithm efficiency was not carried out since it falls out of the scope of this report. Under other circumstances this type of analysis is important in order to assess the performance of the model and could potentially lead to improvements in the code.

Although the 1.2 section can display a message pointing out the location of each of the particles once the model has been run, writing new lines of code for this specific purpose and saving them into a text file was considered to be a better and useful option. This provides a neater and clear outcome from the model.

In addition, the model was tested with an environment containing a different building location and the model performed as good as expected. Nevertheless, it is important to note that this code will not work for simultaneous bacteria sources although it would successfully locate all the buildings.

Finally, the appendices section of this report includes the UML class diagram, the UML activity diagram (describing both the classes’ characteristics and the code’s flow respectively) and the code used for the model.

### Appendix A: UML class diagram



### Appendix B: UML flow diagram

### Appendix C: Code

**agent\_particle.py (agent framework)**

import random

class Particle:

def \_\_init\_\_(self, x, y, start\_height):

self.x = x

self.y = y

self.height = start\_height

def direction(self) :

rndm = random.random()

#print(rndm)

if rndm <= 0.75:

self.x = (self.x + 1) % 300 # change the boundary rule

elif 0.75 < rndm <= 0.85:

self.x = (self.x - 1) % 300

elif 0.85 < rndm <= 0.95:

self.y = (self.y + 1) % 300

elif 0.95 < rndm <= 1:

self.y = (self.y - 1) % 300

def fall(self):

if self.height > 75:

rndm = random.random()

#print(rndm)

if rndm <= 0.20:

self.height += 1

elif 0.20 < rndm <= 0.30:

self.height = self.height

elif 0.30 < rndm <= 1:

self.height -= 1

else:

self.height -= 1

**Bomb\_model\_v1.py**

import matplotlib.pyplot

import csv

import agent\_particle

# 1.1 first we read the raster containing the map with the building where the weapon is placed.

f = open('bomb.txt', newline='')

reader = csv.reader(f, quoting=csv.QUOTE\_NONNUMERIC)

environment = []

for row in reader:

rowlist = []

for value in row:

rowlist.append(value)

environment.append(rowlist)

f.close()

# 1.2 This bit of code helps us to locate the building and returns its position as a meesage.

row\_counter = 0

for row in environment:

for value in row:

if value != 0:

x = row.index(value)

y = row\_counter

print("Weapon located at coordinates","x =", x, "y =", y)

row\_counter += 1

# 2.1 This section is used to specify the parameters of the model.

start\_height = 76

num\_of\_particles = 5000

# 2.2 Here we create each of the bacteria particles (agents)

particles = []

for i in range(num\_of\_particles):

particles.append(agent\_particle.Particle(x, y, start\_height))

# 2.3 Spreading the particles. The algorithm will move each particle, one by one,

# until it reaches the ground.

for j in range(len(particles)):

while particles[j].height != 0:

particles[j].direction()

particles[j].fall()

# 3.1 Creating a list with the coordinates of every particle (no. particle, x, y)

particle\_location = []

for p in range(num\_of\_particles):

no\_part = []

no\_part.append(p + 1)

no\_part.append(particles[p].x)

no\_part.append(particles[p].y)

particle\_location.append(no\_part)

# 3.2 Creating a text file with the coordinates of evety particle

with open('bacteria\_coordinates.txt', 'w', newline='') as f:

csvwriter = csv.writer(f, delimiter=',', quoting=csv.QUOTE\_NONNUMERIC)

csvwriter.writerow(["Particle","x","y"])

for row in particle\_location:

csvwriter.writerow(row)

# 3.3 Making the density map.

#Each particle will add a value of 5 into the x,y coordinates where it falls. Thus, density will be higher in certain coordinates

# if particles fall in the same spot.

for z in range(len(particles)):

environment[particles[z].y][particles[z].x] += 5

matplotlib.pyplot.ylim(0, 300)

matplotlib.pyplot.xlim(0, 300)

matplotlib.pyplot.imshow(environment)

# 3.4 Writing a text file for the density map.

with open('bacteria.density.txt', 'w', newline='') as f:

csvwriter = csv.writer(f, delimiter=',', quoting=csv.QUOTE\_NONNUMERIC)

for row in environment:

csvwriter.writerow(row)