ID2090 Assignment 3 Report

Vaibhav Mahapatra (ME19B197)

July 2021

Contents

1 Introduction **1**

2 Fitting N spheres in smallest square possible **1**

2.1 My Approach 1

2.2 Other approaches 2

2.3 Links to code and outputs 2

3 Modelling a Vaccine Container **2**

3.1 My Approach 2

3.1.1 Data generation 2

3.1.2 Data validation 3

3.2 Links to code and outputs 3

4 Conclusion **3**

1 Introduction

The purpose of this document is to serve as a report explaining my coded solutions to the problems of ID2090 Assignment 3. I have primarily used Python as the language to answer both the questions. This is also my first detailed Report, so any constructive criticism from your side will be much appreciated.

Repository for my code: `https://github.com/vai-bhav-m/ID2090-Assn3`

2 Fitting N spheres in smallest square possible

2.1 My Approach

My approach to solving this problem is primarily a brute-force method. It essentially searches for viable spots to place a circle in a Grid Search Manner and places them accordingly. It can broadly be explained as follows:

1. Initialize N random radii from a normal distribution and sort them in descending order.

2. Estimate the first guess of the required square's dimensions using the below formula

$$s = \sqrt{\sum_{i=1}^{n} \pi r_i^2}$$

3. The square is placed with coordinates of its corners as (0,0), (0,s), (s,0), (s,s)
4. Access a single radius at a time and place each circle in the following manner:
   (a) The circle is initially placed with its center at (r,r) where its radius is denoted by r
   (b) If the circle overlaps with an already present circle, move it to the right by the precision parameter, i.e., the new location of the circle's center is (r+prec,r)
   (c) Repeat the above step of moving the circle right till it can be placed without any overlaps or till it encounters the right boundary of the square.
   (d) If it encounters the right boundary of the square repeat steps (a) to (c) with the initial starting location as (r,r+prec)
   (e) Repeat steps (a) to (d) till the circle can be placed. If it encounters the top boundary of the square, we can conclude that the square is too small to fit all circles. In such a case, increment the side length of the square by the precision parameter, $s_{new} = s_{old} + prec$ and repeat the process of placing each circle at a time from scratch.
5. Save each plot in the format "sq_circle_plot...png"
6. Using imagej to create a video with all generated pics named "

2.2 Other approaches

There were multiple other mathematical approaches that are more efficient than my approach. However, I was unable to completely comprehend the approaches and was hence unable to implement it in my code. Most solutions seemed to either have an optimizing approach or a greedy approach. A few such approaches that I found are:

- Approximate Packing Circles in a Rectangular Container: Valid Inequalities and Nesting
- Greedy Algorithms for Packing Unequal Circles into a Rectangular Container

2.3 Links to code and outputs

- Python code (q1_circles_square.py): `https://github.com/vai-bhav-m/ID2090-Assn3/blob/main/q1_circles_square.py`
- Video animation of fitting process (q1_animation.avi): To view the animation, download it from `https://github.com/vai-bhav-m/ID2090-Assn3/blob/main/q1_animation.avi`

3 Modelling a Vaccine Container

3.1 My Approach

3.1.1 Data generation

1. Initializing a Temperature rise rate with value in range [0,1)
2. Refilling the container 10 times, treating each refill as a new experiment of its own

3. Sampling replenishing periods randomly from a range which induces chances of reaching critical temperature (200K) in at least one of the trials
4. Storing this thermocouple data along with the time in a Pandas DataFrame
5. Creating a 'Total Time' column and plotting the termocouple data against it. This plot is then saved as a figure named 'q2_temp_time_plot.png'
6. The dataframe is exported and saved as a csv file named 'q2_temp_data.csv'

3.1.2 Data validation

1. Reads the csv file 'q2_temp_data.csv' and stores the data in a dataframe
2. Iterate over each row of the dataframe to check if the temperature crosses the 200K threshold
3. If the threshold is crossed, a FAILURE is reported along with the timestamp when it occurred
4. In case of any FAILURE, the container is flagged as unfit for use

3.2 Links to code and outputs

- Data Generation python code (q2_data.py): `https://github.com/vai-bhav-m/ID2090-Assn3/blob/main/q2_data.py`
- Data Validation python code (q2_test_data.py): `https://github.com/vai-bhav-m/ID2090-Assn3/blob/main/q2_test_data.py`
- CSV file of generated data (q2_temp_data.csv): `https://github.com/vai-bhav-m/ID2090-Assn3/blob/main/q2_temp_data.csv`
- Thermocouple Readings plot (q2_temp_time_plot.png): `https://github.com/vai-bhav-m/ID2090-Assn3/blob/main/q2_temp_time_plot.png`

4 Conclusion

I thoroughly enjoyed solving this assignment. By far, this assignment has kept me the busiest, from hunting down research articles on these topics, to googling syntax to assist my code. Even though my approach to solving the above problems may be primitive or naive, I can definitely guarantee you that I learned a lot from the research articles I picked up along the way.