Geo1000 Assignment 4 Report

Group 7: Gong Sicong (5711932)

Link to the Code:https://github.com/GongSicong/nbody

Number of the Words: 607 (Counted by Overleaf)

1 The Difference between Python and C++ Program

In C++, we use class to represent the bodies and their states. In the class, we use a string to represent the name, use user-define classes (Vector3D) to represent the location and velocity, and use a double to represent the mass.

In Python, we use a dictionary to represent the bodies and their states. In the dictionary, we use the key to represent the name, use lists (containing 3 elements) to represent the location and velocity and use a double to represent the mass.

In C++, the class maintains the names while in Python, we lost the names when transforming the dictionary to the tuple. In C++, the iterations are outside the advance function while in Python, the iterations are inside the advance function. These differences lead to differences in execution flows.

2 Reflection of solving the task

We start with the C++ program:

(a) Include headers such as fstream (for writing file) and time.h (for timing) packages; **(b)** Construct write function, as the state and BODIES_COUNT are global variables, we only need two arguments, the int n (iteration times) and bool output (deciding whether to write file or not); **(c)** Put the advance function into the write function; **(d)** Modify the main function, initialize the testing cases (n for the iteration times), and start testing the time cost.

As we mentioned in the previous section, the execution flows of Python program is different from those of C++, to simplify the question, we alter the advance function in Python:

(a) To make it the same in C++, we move the iteration out of the advance function; **(b)** Construct the write function, and put the iteration inside it, for every iteration, we call the advance function to move the state to the next one; **(c)** As we lost the names during the tuple transformation, we need to construct a list containing the names of the planets for the writing function. **(d)** Modify the main function, which would be similar to C++. This time, we import time package to measure the running time.

As for other tasks:

(a) Once I finish one source code (nbody.cpp, nbody.py and ploy_chart.py, I push them to GitHub); **(b)** I got stuck because the execution flows of the 2 programs are different. After several errors and searching them on the internet. I found the point, then, I modified the one in Python to be similar in C++. **(c)** The result is acceptable.

3 The Table and Chart for Timing

To get a more accurate result, We run the programs five times and get the averaged times as the results. And the results are shown in Table 1.

Table 1: The Statistics of Timing (Unit: ms)

Language	**Mode**	**Iteration Times**			
		5000	**500000**	**5000000**	**50000000**
Python	-	172	6859	69519	670747
C++	**Debug**	195	515	3654	31155
C++	**Realese**	204	228	549	3876

We could see that with the increasing iteration times, the time cost is increasing as well.

As we can see in Figure 1 (In order to better visualize, we need to log the time consumption), for small iteration times, the time costs in different languages are similar, but for big iteration times, the time cost shows that python is the longest, C++ release mode is the shortest, and C++ debug mode is in the middle. There may be an order of magnitude difference between the time cost of Python and C++.

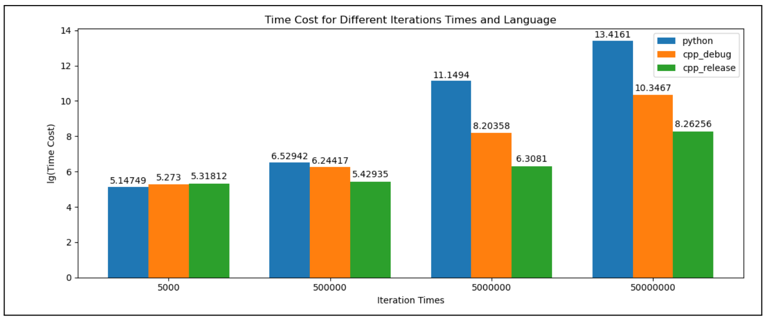

Figure 1: Time Cost

4 Visualization of Result by QGIS

We could use Layer → Add Layer → Add Delimited Text Layer to visualize the result. As we also output the "step" attribute, we could colourize the points by it.

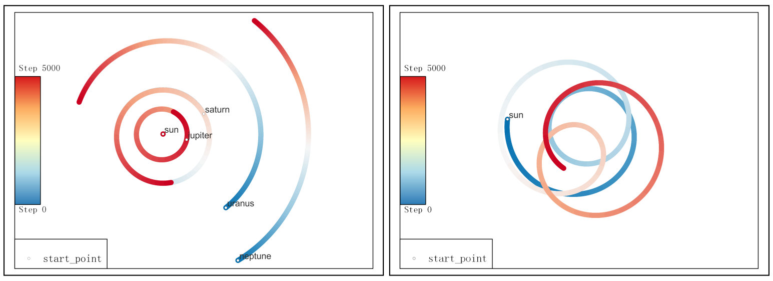

Figure 2: Visualization of Result

The Visualization of the Result is shown in Figure 2. We could see that macroscopically (left), the planets revolve around the sun (relatively stationary) in a near-circular motion, and microscopically (right), the sun itself is also moving.