Assignement 3
Mary Chrishani Jayaweera
Magnus Sahlin

1 Introduction

The aim of this report is to build and optimize a program that can simulate the forces between masses and thereby simulate the movement of stars and planets. We also want to evaluate the complexity of the program.

2 Implementation

A struct was made to keep track of position, velocity, force, mass and brightness of each star and a vector struct to keep track of the x and y of the vector valued variables in the star-struct. The program reads from the command line and initializes the stars using the initStar function with values read from a .gal file with information on all the objects. The stars are initialized using a for-loop and are kept track of using an array of star-pointers. The forces between stars are calculated in an inner loop and the outer loop keep tracks of all of the stars and separate for-loop later updates the positions. All of these updates are in the outermost loop updating the time step which is made using Euler forward.

At first the updates were made using function calls to do vector arithmetics and to do the position updates. Later the code was optimized so that there were less function calls to make the code faster. All forces from all stars was computed for each star separately at first. This was later changed, since the forces acting between two stars are the same but opposite. Now when one force is calculated for a star, the corresponding opposite force is also added to the other star. This results in half as many force calculations.

3 Performance

Performance was evaluated by measuring the time of a set number of time steps but with different amount of stars in the simulation.

In table 1 we can see that as N increases, the time increases squared. This is according to the specification in the assignment. For small N the pattern is not apparent, since most of the computing time is not directed towards the actual simulation, but rather memory accesses and other initialization things. For the first four measurements, the number of particles are doubled for each test, so the time should be quadrupled. The same is true for the last four.

N [#]	Real [s]	User [s]	System [s]
100	0.092	0.083	0.009
200	0.213	0.209	0.004
400	0.319	0.316	0.004
800	2.284	2.280	0.004
1000	3.747	3.739	0.008
2000	14.004	13.957	0.044
4000	56.885	56.878	0.004
8000	229.6830	229.484	0.177

Table 1: Time for 200 iterations with different N

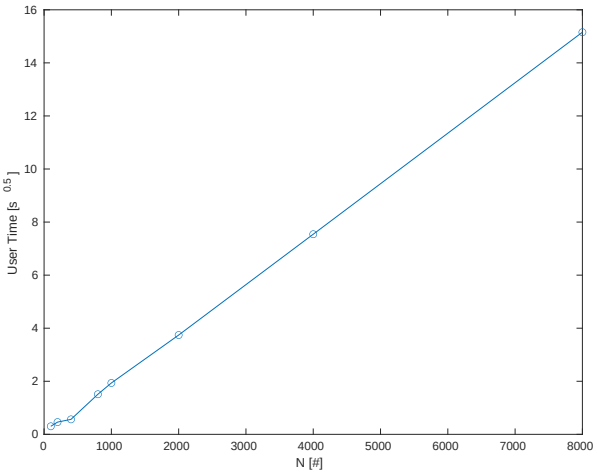

Figure 1: Time for 200 iterations with different N

In figure 1 we can clearly see that when the time is plotted squared rooted, we see a linear relation. This means that the model is of $\mathcal{O}(n^2)$ complexity.

3.1 Compiler Optimization Flags

We found that the Ofast flag is the best for our purpose. Running the same simulation with 3000 stars for 100 iterations without graphics takes 3.771 seconds with the O2 flag, 3.539 second with the O3 flag, and 3.092 seconds with the Ofast flag. For correctness and reproducibility all tests were made on the same computer with processor i5-6200U and the same compiler GCC version 7.4.