UPPSALA
UNIVERSITET

HIGH PERFORMANCE PROGRAMMING

Report - Assignement 3
- The Gravitational N-body Problem -

Group 5
Nils Gumaelius
Vistoria Sedig
Simona Stoyanoska
*
februari 2020

Contents

1 Introduction

In this problem we address the N-body problem, i.e. we are numerically solving for a system of particles in a gravitational field. The goal is to make our code as fast as possible.

2 Theory

The physical model for this problem is described by Newtons model. The general form of the force of attraction between to particles i and j is:

$$\mathbf{f}_{i,j} = \frac{Gm_i m_j}{\mathbf{r}_{i,j}^3}\hat{r}_{i,j},$$

$\mathbf{r}$ denoting he distance between them in direction $\hat{r}$ and G is the universal gravitational constant.

To implement the law of attraction on one particle only, we are using:

$$\mathbf{F}_i = -Gm_i \sum_{i\neq j, j=0}^{N-1} \frac{m_j}{\mathbf{r}_{i,j}^2}\hat{r}_{i,j}$$

In the next step we may use a symplectic Euler ($\mathcal{O}(n^2)$) for the time integration, and thus are enabled to update particle's position and velocity:

$$\begin{aligned} \mathbf{a}_i^n &= \frac{\mathbf{F}_i^n}{m_i} \\ \mathbf{u}_i^{n+1} &= \mathbf{u}_i^n + \Delta t \mathbf{a}_i^n \\ \mathbf{x}_i^{n+1} &= \mathbf{x}_i^n + \Delta t \mathbf{u}_i^{n+1} \end{aligned}$$

We thus simulate the movement of N particles in order to analyse the dynamics of the galaxy in time. To simplify the problem, we work with 2D.

3 Optimization methods

3.1 Plummer Spheres

The theoretical description of the gravitational force cannot be implemented as such nummerically, due to the finite space in the computers. In order to avoid this we bound the original exrpession above as follows:

$$\mathbf{F}_i = -Gm_i \sum_{i\neq j, j=0}^{N-1} \frac{m_j}{(\mathbf{r}_{i,j}^2 + \epsilon)}\hat{r}_{i,j}$$

Explicitly we just avoid dividing by zero, by adding a small enough ϵ.

3.2 Physical analysis

Note that each particle experiences the same force but in opposite direction. We used this to avoid double computations. But testing later on shows that this doesn't make a huge difference in time efficiency. But still it is a preferable to avoid unnecessary computations.

3.3 Optimization of the program

We used the following techniques to optimize the code:

- For our optimized version we used allocated memory for our data structures instead of static.

- In the inner loop we changed its max so that the loop only runs until it is as large as the value from the outer loop. This had a large impact on the cost.

- All the constants were changed to const int instead of int and we made a register for a few variables.

The largest difference came from adding complier flags. We did some research and tried a lot of different flag combinations and came to that the following gave the best result:
$-Ofast - march = native - ffinite - math - only - fno - signed - zeros$ (*)

4 Time analysis

To test the performance, we measured the time for many iterations while changing the number of particles in the system, as seen in table 1. The results of both the user and the real time shows that the computational time approximately depends N quadraticly, as expected this can be seen in figure 1.

In order to see this pattern, we need to run the simulation for N large enough, otherwise the computational time will be mostly concentrated in memory access.

Table 1: Meauremnets for testing the system with fixed number of iterations(1000), while changing N, the number of particles. The test was done with optimized code and flags(*) seen in chapter 4.

N	Real time [s]	User time [s]	System time[s]
1000	0.007	0.006	0.001
2000	0.015	0.010	0.004
3000	0.019	0.018	0.000
4000	0.027	0.02	0.00
5000	0.032	0.020	0.005
6000	0.031	0.027	0.004
7000	0.039	0.034	0.004
8000	0.045	0.040	0.004
9000	0.080	0.075	0.004
10000	0.118	0.097	0.004

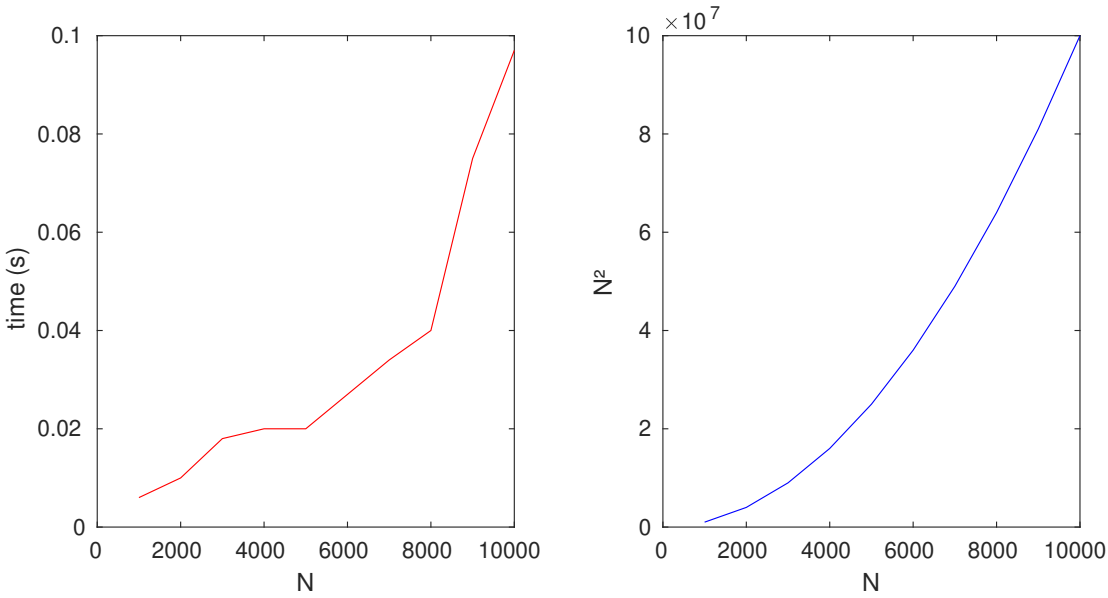

Figure 1: Time plot for fixed number of iterations(200), while changing N. Time complexity tends to be $\mathcal{O}(n^2)$, as expected. The test was done with optimized code and flags(*) seen in chapter 4.

4.1 Optimization efficiency

Comparing the unoptimized code without flags and N = 1000 and 200 timesteps gives the result:
$Unoptimized$: 29.48s
$Optimized$: 13.95
Comparing the optimized version with flags(*) and the same constants gives the result: one.
$Unoptimized$: 2.661s
$Optimized$: 0.005s
We can see that the optimized version was much faster both with and without flags.