Coursework: Molecular Dynamics

Computing Methodologies III: Numerical Algorithms

Computing Methodologies III: Parallel Programming

Tobias Weinzierl

August 29, 2018

This assignment is to be completed and handed in via DUO. For deadlines, please consult DUO and the level handbook. This is a joint assessment for the two submodules Numerical Algorithms and Parallel Programming.

Instructions

- This assignment is to be completed using the assignment's source code provided via DUO. It simulates three molecules where one molecule is moving around the other molecules through gravity. The term molecule is used here synonymously with particles or space bodies.
- The only routine where you are allowed to make changes is `void updateBody()`.
- Notably, the code is not to use any other parameters and you are not allowed to make your submitted code output any additional information to the terminal (or a file). This does **not** mean that it does not make sense to add additional outputs for your own experiments (to extract additional helper information about particular molecules, e.g.).
- For each assignment step, you are expected to submit particular artefacts. It is either a piece of source code (do not submit multiple files, but keep the structure; ensure that the code continues to compile exactly the same way the given template does) and/or a PDF document. Ensure all page limitations are observed and submit PDFs only. Word files, e.g., are not accepted. Stick exactly to the submission format.

Step 1: Multiple free molecules

In the template code given to you, one molecule (the first one specified on the command line) is free, while all the other molecules are fixed in space. Alter the code such that all molecules move freely through space at the same time. Ensure that the global statistics (minimal distance between all particles and maximum velocity) are still computed correctly. Marks will be given for the correctness and also for the efficiency of the implementation (try to spot redundant calculations). The solution to this step is to be handed in as a single file called `solution-step1.c`.

This step is worth 25 marks for Numerical Algorithms.

Step 2: Lennard-Jones

For this step, we replace the gravitational interaction with that from the Lennard-Jones potential. The Lennard-Jones force computation reads is

$$f_{ij} = -4 \cdot \epsilon \cdot \left(-12 \cdot \left(\frac{\sigma}{dx_{ij}} \right)^{12} + 6 \cdot \left(\frac{\sigma}{dx_{ij}} \right)^{6} \right) \frac{1}{dx_{ij}}$$

between two molecules i and j with a distance of dx_{ij}. For all following experiments, we simulate Argon (Ar). For Argon, the molecules have the typical mass 39.948 (specified on command line), while the material constant $\sigma = 3.4 \cdot 10^{-10}$ and $\epsilon = 1.65 \cdot 10^{-21}$. Program this new potential. The solution to this step is to be handed in as a single file called `solution-step2.c`.

This step is worth 25 marks for Numerical Algorithms.

Step 3: Lennard-Jones experiments

For the third step, we study our code with the command line arguments `1e-6 1e-2 1e-3 0 0 0 0 0 39.948 0 0 0 0 0 0 39.948`. Study this setup with time step sizes $\Delta t \in \{10^{-15}, 10^{-16}, 10^{-17}, 10^{-18}\}$. Summarise your findings in a one page write-up and create a plot that displays the distance between the two molecules at $t = 10^{-2}$. Discuss what convergence order and what ODE properties you can observe from this plot. The solution to this step is to be handed in as a one-page PDF called `solution-step3.pdf`.

This step is worth 25 marks for Numerical Algorithms.

Step 4: Adaptive time stepping

You will observe from Step 3 that you simulations run extremely long even though there are only two particles involved. Implement an adaptive time stepping scheme, compare its computational efficiency (number of time steps required) for a given accuracy to a fixed time stepping scheme as used in Step 3, and, finally, test your setup for more than two particles. The solution to this step is to be handed in as a two-page PDF called `solution-step4.pdf` which discusses your findings and ideas. Furthermore, you have to submit your source code as file `solution-step4.c`.

Some hints:

- A time step size below 10^{-18} makes no sense.
- If the minimum distance between any two particles becomes smaller than 10^{-9}, the minimum time step size should be used. Plot the force formula to find out why 10^{-9} (or a similar threshold) are a good magic parameter.
- If the biggest velocity times the smallest distance means that two particles might jump "through each other", then the time step size should at least be halved (I usually divide by eight).
- If none of these criteria holds, I typically increase the time step size by around 1%.

This step is worth 25 marks for Numerical Algorithms.

Step 5: Parallelisation concept

For this step, you are allowed to "roll back" to the solution of Step 1, i.e. the gravity model where all particles are allowed to move freely. Inefficiencies in the solution to Step 1 will not have any knock on effect from hereon. Also any correctness flaws are not marked in Steps 5–7 unless they have been introduced through the parallelisation. If you are confident in your solution of Step 2, use this potential/force. Do not attempt to parallelise the adaptive time stepping.

Write a report which discusses which parts of the routine `updateBody()` are well-suited for parallelisation. Focus on OpenMP only. The solution to this step is to be handed in as a one-page PDF called `solution-step5.pdf`. Please phrase your statements in terms of well-known performance models, i.e. predict what efficiency you expect to obtain.

This step is worth 25 marks for Parallel Programming.

Step 6: Parallelisation implementation

Realise the parallelisation with OpenMP. The solution to this step is to be submitted as `solution-step6.c`.

This step is worth 50 marks for Parallel Programming.

Step 7: Experiments

Study the scalability of your code. For this, switch off all file output. Your findings are to be submitted as a one-page PDF report `solution-step7.pdf`. Use diagrams where appropriate and compare your results to your performance model/prediction from Step 5.

This step is worth 25 marks for Parallel Programming.