MVCOMP1, Ex.5

To be submitted by November 22, 2024, 6PM

Tree code *(24 points)*

Tree algorithms can approximately calculate the forces in an N-body system by means of a hierarchical multipole expansion. This reduces the computational cost to something of order $\mathcal{O}(N \ln N)$. On the Moodle page of the lecture, a simple skeleton tree code in Python can be downloaded (tree.py)[1]. This implementation is a toy code using only monopole order and is not optimized for speed or memory consumption. You can use this template for this exercise, either directly or translated to another language, or if you wish, you may also write your own tree code from scratch.

Consider N particles of total mass $M_{\rm tot} = 1$ placed randomly into a cubical box of unit side-length. We assume all particles have the same mass, and we adopt $G = 1$. Assume that the gravitational potential of individual particles is Plummer-softened with a gravitational softening length $\epsilon = 0.001$, i.e. we adopt the potential

$$\phi(\boldsymbol{r}) = -G\frac{m}{(\boldsymbol{r}^2 + \epsilon^2)^{1/2}} \tag{1}$$

for a single particle of mass m.

Calculate the gravitational forces for all N particles with an oct-tree and determine the typical force accuracy and calculational time in comparison with direct summation, both as a function of N and of the opening angle parameter $\theta^\star$. To this end, carry out the following steps:

(a) The provided code template is incomplete, so start by filling in the missing pieces. Places where you have to complete the code are marked with the comment 'TO BE FILLED IN'. In particular, you need to add the computation of the multipole moments of tree nodes from their subnodes, as well as the partial force calculation from a tree node that is used in the tree walk (*only at the monopole approximation!*). Also, please add a calculation of the *exact* forces by direct summation. When you are done, verify that the force approximation delivered by the tree code is roughly correct by adding suitable output to the code.

(b) To allow a more quantitative analysis, add an automatic calculation of the relative acceleration error for each particle

$$\eta = \frac{|\boldsymbol{a}_{\rm tree} - \boldsymbol{a}_{\rm direct}|}{|\boldsymbol{a}_{\rm direct}|}, \tag{2}$$

where $\boldsymbol{a}_{\rm tree}$ is the acceleration calculated with the multipole expansion and $\boldsymbol{a}_{\rm direct}$ is the acceleration estimated with direct summation. Determine a simple arithmetic average $\langle\eta\rangle$ of the mean relative acceleration error[2].

(c) Add diagnostic code that tells you the average number of particle-node interactions per particle, i.e. how many nodes the multipole expansion on average involves.

(d) Make a little grid of calculations, adopting $N = 500$, 1000, 2000 and 4000. In each case, measure the calculation time for the tree-based force calculation and for direction summation, as well as $\langle\eta\rangle$ and the mean number of terms the tree code used per particle. Report the results in a table.

[1] **Exercise and code credit: Dr. Philipp Girichidis.**

[2] Note that a more careful analysis of the errors should really consider the full distribution of the errors. It could well be that there are occasionally very large, catastrophic errors that go unnoticed in a simple average such as $\langle\eta\rangle$. To diagnose this, one needs to look at the error distribution function.

(e) Using the previous results, make a plot of the execution time of the force calculation with the tree as a function of N. Use logarithmic axes and fit a regression line (in the log) to the 4 data points you obtained. Also include the results for direct summation.

(f) In the example core, the opening angle is $\theta^\star = 0.6$. For a fixed number of particles ($N = 1000$), change the opening angle to 0.2, 0.4, and 0.8. What is the difference in terms of force error and execution time? Please, discuss this in your submission.