Fundamentals of Simulation Methods

Exercise Sheet 3 — Collisionless dynamics and the tree code

Daniel Rosenblüh, Janosh Riebesell · Lecturer: Volker Springel · November 13th, 2015

1 Collisionless system in equilibrium

The energy per unit mass of a particle in a stationary collisionless system can be written as

$$\epsilon_i = \frac{1}{2} v_i^2 + \Phi_i, \tag{1}$$

where v is the speed of the particle and Φ is the value of the gravitational potential at its current position. Prove that the total energy of the system is equal to

$$E = \frac{1}{3} M \langle \epsilon \rangle \tag{2}$$

where $\langle \epsilon \rangle$ is the mass-weighted average of ϵ for all particles, and M is the total mass of the particles.

The identity in eq. (2) follows from an application of the Virial theorem (VT) which relates the time-averaged total kinetic and potential energies, $\langle T \rangle$ and $\langle U \rangle$, of a stable many-particle system bound by potential forces:

$$2\langle T \rangle = n \langle U \rangle, \tag{3}$$

where n is determined by the force $\boldsymbol{F} = -\boldsymbol{\nabla} U(r)$ with a potential of the form $U(r) = \alpha r^n$ acting between any two particles of the system. Thus, for a gravitational potential, $n = -1$ and $2\langle T \rangle = -\langle U \rangle$. Using this to rewrite the total energy, we obtain the desired result:

$$\begin{aligned} E = \langle T \rangle + \langle U \rangle &= \frac{1}{3}\left(3\langle T \rangle + 3\langle U \rangle\right) \overset{\text{VT}}{=} \frac{1}{3}\left(\langle T \rangle + 2\langle U \rangle\right) \\ &= \frac{1}{3}\sum_{i=1}^{N} \frac{m_i}{2} v_i^2 - \frac{2}{3}\sum_{i<j}^{N} \frac{G m_i m_j}{|\boldsymbol{x}_i - \boldsymbol{x}_j|} = \frac{1}{3}\sum_{i=1}^{N} \frac{m_i}{2} v_i^2 - \frac{1}{3}\sum_{i \neq j}^{N} \frac{G m_i m_j}{|\boldsymbol{x}_i - \boldsymbol{x}_j|} \\ &= \frac{1}{3}\sum_{i=1}^{N} m_i \left(\underbrace{\frac{v_i^2}{2} - \sum_{j \neq i}^{N} \frac{G m_j}{|\boldsymbol{x}_i - \boldsymbol{x}_j|}}_{\Phi_i} \right) = \frac{1}{3}\sum_{i=1}^{N} m_i \epsilon_i = \frac{1}{3}\sum_{i=1}^{N} m_i \frac{\sum_{i=1}^{N} m_i \epsilon_i}{\sum_{i=1}^{N} m_i} = \frac{1}{3} M \langle \epsilon \rangle. \end{aligned} \tag{4}$$

2 Tree code

Tree algorithms can approximately calculate the forces in an N-body system by means of a hierarchical multipole expansion. This reduces the computational cost's scaling with N from $\mathcal{O}[N^2]$ to $\mathcal{O}[N \ln(N)]$. On the Moodle page of the lecture, a simple skeleton tree code (in C) can be downloaded (note that this implementation uses 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 $M_{\text{tot}} = 1$ placed randomly into a cubical box of unit side-length. For definiteness, 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

$$\varphi(\boldsymbol{r}) = -\frac{m}{\left(\boldsymbol{r}^2 + \epsilon^2\right)^{1/2}} \tag{5}$$

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 calculation time in comparison with direct summation, both as a function of N and of the opening angle parameter θ. 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. Also, please add a calculation of the exact forces by direct summation. When you're 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 measurement of the average force error after all forces have been calculated by the tree and direct summation. To this end, consider the relative force error

$$\eta_i = \frac{|\boldsymbol{a}_{i,\mathrm{tree}} - \boldsymbol{a}_{i,\mathrm{direct}}|}{|\boldsymbol{a}_{i,\mathrm{direct}}|}, \qquad i \in \{1, ..., N\}, \tag{6}$$

for each particle, and determine a simple arithmetic average $\langle\eta\rangle$ of the mean relative force error[1]. Also, 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.

(c) Make a little grid of calculations for $N \in \{5000, 10\,000, 20\,000, 40\,000\}$, and opening angles $\theta \in \{0.2, 0.4, 0.8\}$. In each case, measure the calculation time for the tree-based force calculation and for direct 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.

(d) Using the previous results, make a plot of the execution time of the force calculation with the tree as a function of N (for the $\theta = 0.4$ case). 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. Estimate the time needed for 10^{10} particles for both methods by extrapolating your timing results.

(a) We verify that our tree method force calculation is approximately correct by comparing the acting forces to those obtained via direct summation in a random sample fashion, i.e. we look at just one, but *any* one particle per calculation. A typical example of such a random sample for an opening angle of $\theta = 0.8$ looks like

$$\begin{aligned} a_{\mathrm{tree},x} &= -2.80, \qquad a_{\mathrm{tree},y} = 1.60, \qquad a_{\mathrm{tree},z} = -6.16, \\ a_{\mathrm{dir},x} &= -2.78, \qquad a_{\mathrm{dir},y} = 1.57, \quad a_{\mathrm{dir},z} = -6.18, \end{aligned} \tag{7}$$

which we take to be sufficiently accurate.

(b) See `tree.c`.

[1] 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.

(c) Calculation times both for the tree method (t_{tree}) and for a direct summation of forces (t_{dir}), relative errors (η), and average number of particle-node interactions per particle in the tree calculation (n_{int}) for a range of parameters are shown in Table 1.

	N	5000	10 000	20 000	40 000
$\theta = 0.2$	t_{dir}	3.87 s	15.06 s	62.41 s	239.42 s
	t_{tree}	1.75 s	4.91 s	12.52 s	32.62 s
	η	0.02 %	0.02 %	0.02 %	0.02 %
	n_{int}	1736.48	2414.91	3128.12	3975.71
$\theta = 0.4$	t_{dir}	3.85 s	15.10 s	61.80 s	246.06 s
	t_{tree}	0.51 s	1.19 s	3.00 s	6.93 s
	η	0.20 %	0.16 %	0.14 %	0.12 %
	n_{int}	507.94	623.32	738.95	867.69
$\theta = 0.8$	t_{dir}	3.93 s	15.10 s	60.13 s	252.77 s
	t_{tree}	0.11 s	0.24 s	0.54 s	1.37 s
	η	1.29 %	1.13 %	1.00 %	0.85 %
	n_{int}	111.63	129.23	144.16	162.31

Table 1: Computational results

(d) Figure 1 shows a log-log plot of the execution time as a function of the particle number N for the force calculations of both the tree method and direct summation. It depicts the case $\theta = 0.4$. Slope and intercept values for the linear regressions of $t_{\text{tree}}(N)$ and $t_{\text{dir}}(N)$ are given in Table 2. Using these values to project an estimate on the execution time for

	$t_{\text{tree}}(N)$	$t_{\text{dir}}(N)$
slope m	1.26	2.00
intercept b	−11.41	−15.72

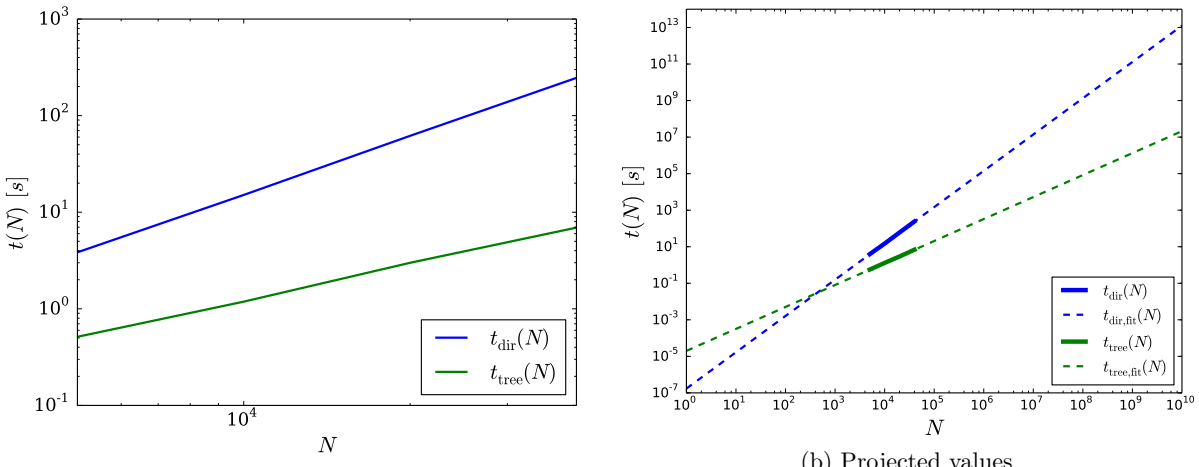
Table 2: Linear regression values

a calculation involving 10^{10} particles according to

$$t_{\text{exec}}[\text{s}] = e^{m \cdot \ln(N) + b}, \tag{8}$$

yields

$$\begin{aligned} t_{\text{tree}}(N = 10^{10}) &\approx 4.31 \times 10^7\,\text{s} \approx 500\,\text{d}, \\ t_{\text{dir}}(N = 10^{10}) &\approx 1.60 \times 10^{13}\,\text{s} \approx 506\,245\,\text{yr}. \end{aligned} \tag{9}$$



(a) Calculated values

(b) Projected values

Figure 1: Log-log plot of the execution time as a function of the particle number N

A graphical representation of these estimates is given in Figure 1 (b). Interestingly, this plot also shows at which value of N the overhead introduced by the tree method actually becomes a

time penalty: $N \lesssim 340$. However, extrapolations from our four data points into such low ranges of N need to be taken with a grain of salt. Actual runs showed that the tree method was still slightly faster even below $N = 100$.