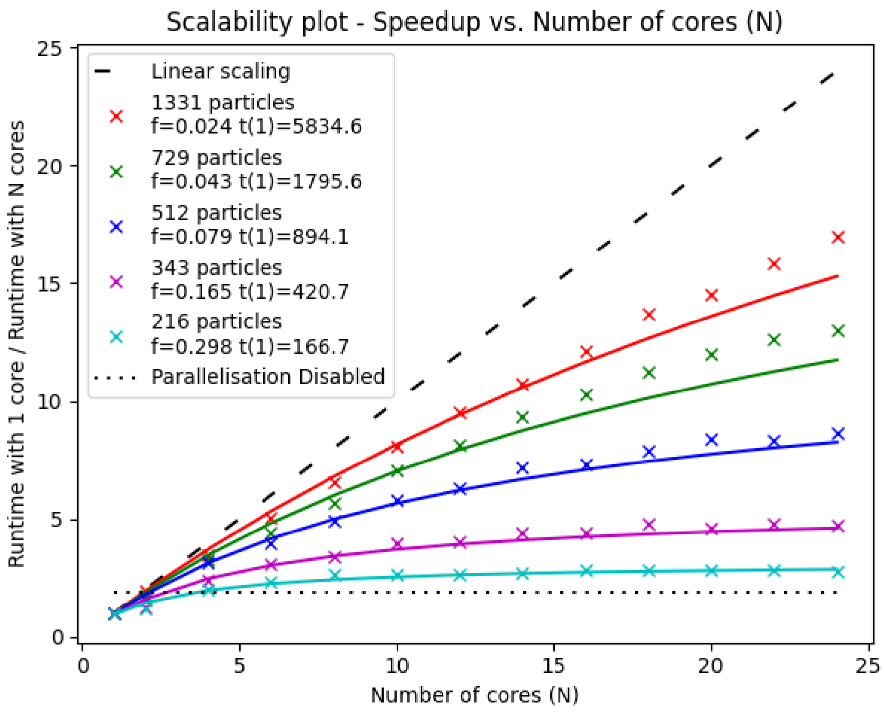
Parallel Scientific Computing I Report – Colliding Suns, an N-body solver – Max Woolterton

Thanks is due to Durham University for the use of their HPC facilities, Hamilton, on which all tests were performed. More specifically, all runs were executed on a 24 core Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz. All code was compiled using Intel's C++ compiler (19.0.5.281) using the following compile options: "icpc -fopenmp -O3 -xhost --std=c++0x". All parameters were fitted using Scipy's curve_fit. All raw data is represented by X's and the order of occurrence top to bottom is the same on graph and legend, for clarity in black and white print outs. The fitted curves are Amdahl's Law and Ch^{p} respectively.

Strong Scaling Study:

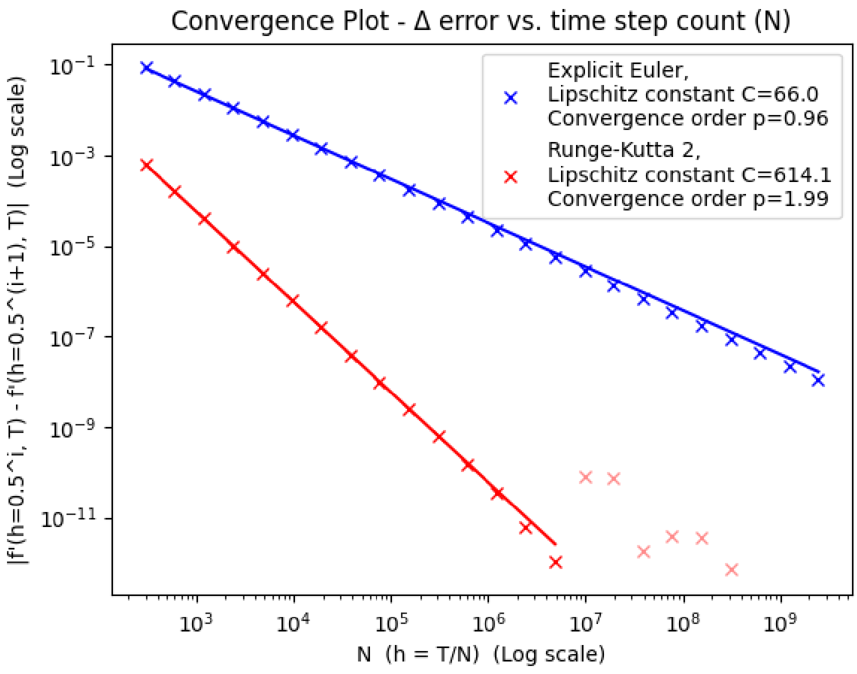
While studying the scalability of my code, I tested runtime for 1, 2, 4, … 22, 24 cores. Each run lasted for an extended period and was repeated 4 times, with the results being averaged, so as to minimise measurement noise (which proved not to be significant). IO was disabled during all runs. For each of the varying numbers of particle counts chosen, 5 randomised setups were tested, each with a fixed duration of 400,000 time steps (so that the scalability w.r.t the number of particles can also be observed using t(1)). The setup time (which ranged from 0.02-0.05s depending on the problem size) was measured and taken into account.



Amdahl's' Law ("AL") is fitted to the data and plotted, with the t(1) and f values shown in the legend, respectively for the runtime on 1 core and the percentage of serial code (which can be seen to decrease rapidly from 30-2%). For lower particle counts AL is followed almost perfectly with strong scaling stagnation being more prevalent as the problem is split into too small chunks, thus the parallelisation overhead and the coordination costs start to dominate and cause the speedup to plateau. However, at higher particle counts the code's parallel efficiency decreases at a far slower rate and we start to see the code scale even better than AL. To provide a fair idea of the true cost of parallelisation, the speedup from disabling all parallelisation was also measured and found to vary between 1.88 and 1.95, decreasing with particle count. Its average is shown as a dotted line for reference.

Convergence Order Study:

For my convergence order study, I decided to track the velocity of a particle at a fixed time shortly before collision in a symmetrical 2 particle setup. The particles began travelling in parallel before curving towards each other and colliding. The quantity was tracked each time the time step (h) was halved with the difference between successive values subsequently being plotted as an estimate for the true "error". Various similar set ups were also plotted, and their convergence was found to be consistent. I chose the particles' masses to be high (50) and tracked a time close to collision so that the rate of change of the 2 derivatives would be high, leading to an error prone setup.



The errors plotted are a resultant accumulation of the round off error due to floating point arithmetic and the truncation error (of the Taylor series) resulting from the time stepping scheme. At higher time steps (lower N) the truncation error, the quantity we are interested in, dominates. However, as we approach the level of machine precision, at lower h, the round off error begins to dominate. This is what we can see happening on the right-hand side of the graph for RK2 (faded entries) and is why these values are excluded during the convergence order calculations. I calculated the convergence orders of explicit Euler and Runge-Kutta 2 as 0.96 and 1.99 respectively. I believe these values are slightly lower than 1 and 2 due to the contributions throughout from the round off error.