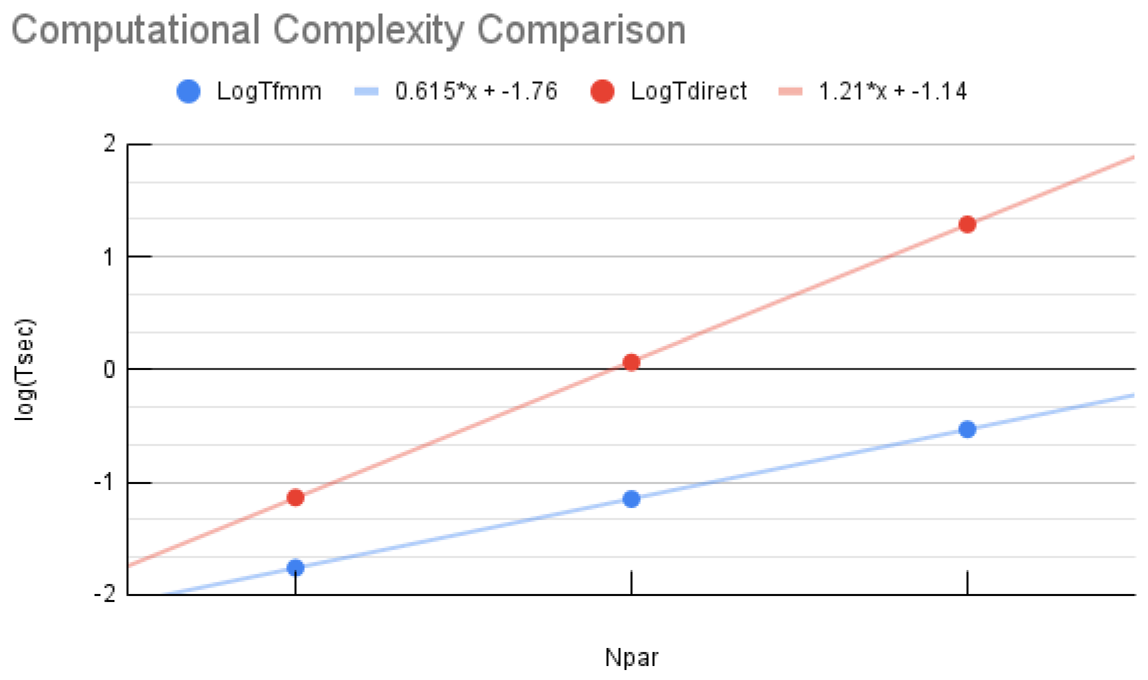
Computational Complexity



In the graph above, you can see that the LogTFmm grows much more slowly than the LogTDirect. The reason is down to how the programs calculate the particle simulation. With the TDirect method, we are calculating each particle separately. For N bodies, this means we need to check each body to determine the calculation for what we should be doing. This means, Tdirect alone has a runtime of $O(N^2)$. Compare that to FMM, which actually separates the particle simulation down to smaller areas. In essence, the program calculates based on dividing the world into, say, K regions. These regions can have anywhere from 1 to N particles. However, FMM only visits each region once, stores the particle data, and then references it for future calculations. Rather than the Direct method, which must recalculate at each point, FMM gets roughly an average performance $O(N)$ as it only needs to calculate a particle in most cases once. At small scales, these are hardly big gains, but they make a huge impact if you need to simulate potentially thousands of particles in a scene.

Gflop Calculations

```
===== Max potential difference = 5.974920e-06 =====
===== Total FMM vs. direct energies & error = -2.572756e+07 -2.572756e+07 -4.756415e-09 =====
===== FMM & direct CPU times = 7.344300e-02 1.057334e+00 =====
===== FMM & direct floating-point operation counts = 1.564078e+08 5.119712e+09 =====
===== FMM & direct Gflop/s = 2.129649e+00 4.842095e+00 =====
```

System Info:
Intel(R) Core(TM) i9-14900HX (2.20-5.80 GHz)
NVIDIA RTX 4070
64-bit operating system, x64-based processor