Complexity and Precision Analysis of the Barnes-Hut and Fast Multipole Hierarchical Methods for N-Body Simulation

Thomas Hartigan Physics Part II Computing Project

Project Name: Hierarchial Methods for N-body Simulation

Abstract

N-body simulations are vital for both cosmological and Coulomb energy optimisation problems, however, direct simulation of these systems incurs a computational cost $\mathcal{O}(N^2)$. Two algorithms, the fast multipole method and the Barnes-Hut method, by which the computational cost for these simulations can be reduced are implemented here. Few works have investigated the real-world computational complexity and precision achieved by these algorithms, so these performance parameters are investigated in this work, leading to the presentation of a new estimate of computational complexity for the Barnes-Hut method, when varying its precision parameter. Furthermore, figures demonstrating the relative performance of each algorithm subject to a range of initial conditions are reported.

1 Introduction

Calculation of the gravitational and electrostatic forces or potentials for a system of N particles typically requires conducting $\mathcal{O}(N^2)$ calculations - one for each particle interacting with each other particle. When conducting force calculations, this computation cost can be halved by exploiting Newton's Third Law [1]. Despite this, the computational cost remains large even for small systems. Hence, the abundant applications of N-body simulations to problems ranging from cosmology to Coulomb energy optimisation, particularly in the Hartree-Fock model [2, 3], have necessitated the development of more efficient, approximate solution algorithms. Two such algorithms, the Fast Multipole Method (FMM) [1] and the Barnes-Hut (BH) method [4] are investigated here. This is done by performing tests using Python [5] implementations of each algorithm, and comparing results to those obtained by direct calculation.

To fully assess the efficiency, accuracy and precision of these algorithms, a wide range of initial conditions are used. These include variation in number of particles interacting, precision parameters, and particle distributions. This enables comparison between the true and expected performances in each case. The fundamental theory and performance expectations for BH and FMM methods are presented in §2.2 and §2.3 respectively. Description of the testing regimes used along with evaluation of the developed Python

implementations then follow in §3, before results are presented and discussed in §4. Finally, conclusions are presented in §5.

2 Theory

Both FMM and BH methods decompose the simulation area into a tree structure of grids in 2D or boxes in 3D. In this work, a 2D approach will be taken for FMM implementation, and a 3D approach for BH implementation. It should be noted that in 2D, the electrostatic potential at $\boldsymbol{r}$, ϕ is given by

$$\phi(\boldsymbol{r}) = -\frac{1}{2\pi\epsilon_0}\int d^2\boldsymbol{r'}\sigma(r')\ln\left(\frac{|\boldsymbol{r}-\boldsymbol{r'}|}{a}\right), \tag{1}$$

where $\sigma(\boldsymbol{r})$ is the surface charge distribution and a is an arbitrary reference point [6]. In the following, a dimensionless approach is adopted, where the coefficient $(1/2\pi\epsilon_0)$ in (1), or any similar Green function form is taken to be unity. Hence, results either have relative units (RU) or no units (NU). In the case of a single particle at the origin in 2D, the form of the potential can be found using (2), where q is the property (such as charge or mass) generating the potential [6].

$$\phi(\boldsymbol{r}) = -q\ln(|\boldsymbol{r}|)\,. \tag{2}$$

In 3D, the potential at a point $\boldsymbol{r}$ due to a particle of property q at the origin is given by (3).

$$\phi(\boldsymbol{r}) = -\frac{q}{|\boldsymbol{r}|} \tag{3}$$

For a system of N particles, the total potential at a point can be found by summing the potential from each particle, as in (4), where ϕ_n is the potential due to a particle with distance $|\boldsymbol{r}_n - \boldsymbol{r}|$ from $\boldsymbol{r}$.

$$\phi_N(\boldsymbol{r}) = \sum_n \phi_n \tag{4}$$

The force on a particle i, F_i, with property q_i can then be found at any point using 5.

$$F_i = -q_i\frac{d\phi}{dr} \tag{5}$$

2.1 Direct Calculation

Using equation (4), calculation of the potential experienced by a single particle requires $N-1$ calculations, where N is the total number of particles in the system. Therefore, to calculate the potential at all N particle positions requires $\mathcal{O}(N^2)$ calculations. The precision with which a system may be simulated in this way is limited only by the computational round-off error.

2.2 The Barnes-Hut Method

The BH method exploits the $1/r^2$ decay of Coulomb forces in 3D. This is done by adopting a two-pass approach. First, the system is analysed starting with the whole volume, dividing it into a tree structure of boxes using Algorithm 1. Within this tree structure, each generated box has either zero or eight child boxes, which combined occupy the same volume as their parent box, as is depicted in Figure 1. A 2D version of this same approach is illustrated in Figure 2

The upwards pass then occurs, where each box with zero child boxes - hence containing either one or zero particles - is analysed. For each of these childless boxes, the box centre of mass (CoM) position and total mass is set equal to those of the particle contained within the box. If the box contains no particles, then the total mass is set to zero, essentially neglecting it in future calculations. For a fully optimised implementation, it could instead be recorded that this box is to be discounted in future calculations. Starting from the boxes which have been divided the most times, the CoM position and total mass of their respective parent boxes are then calculated. Similarly, the CoM position and total mass are then calculated level by level for all coarser levels, until reaching the box containing the whole system.

The force on each particle may then be calculated using Algorithm 2. Within Algorithm 2, the else if condition corresponds to a far away box, as determined by the precision parameter θ, which contains multiple particles. The else condition corresponds to a box which contains many particles, but is located nearby, again as determined by θ. Hence, the $1/r^2$ contribution is greater, so finer precision is required to analyse the contributions from that box.

Initialisation
| Begin with a box, b_1 the size of the simulation volume and a list of particles
for *each particle* **do**
| LoadParticle(particle, b_1)
end

Function `LoadParticle(`*particle, box*`):`
| Append particle to the list of particles within the box
| **if** *box contains more than 1 particle* **then**
| | Divide box into 8 equally sized child boxes
| | **for** *each particle in box* **do**
| | | Determine the child box within which this particle exists
| | | LoadParticle(particle, child box)
| | **end**
| **end**
| **return**

Algorithm 1: Barnes-Hut tree structure generation

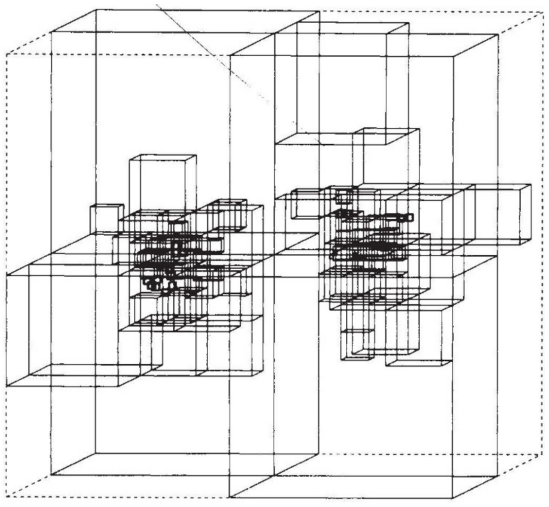

Figure 1: An example 3D tree structure generated using Algorithm 1 [4]

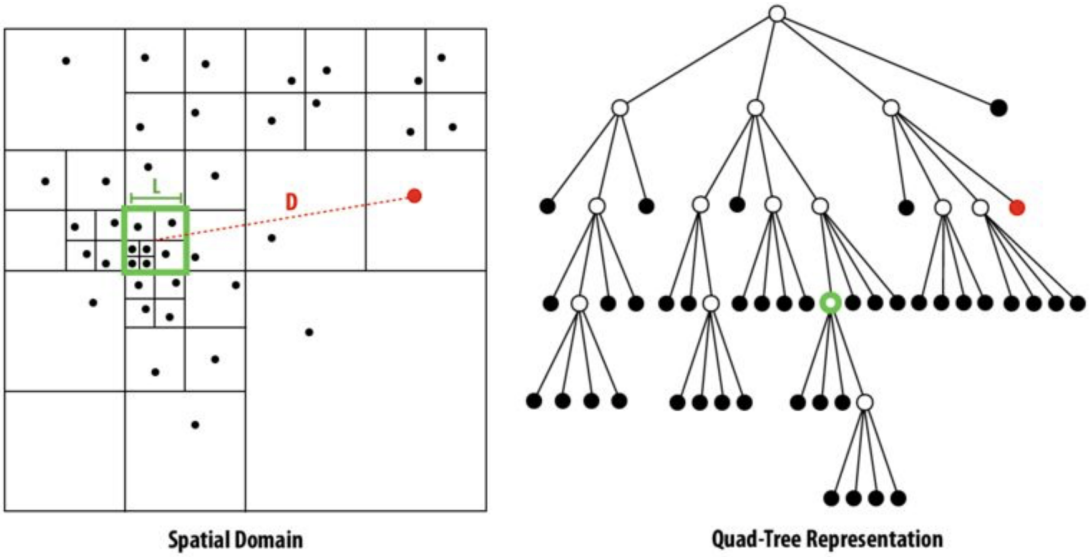

Figure 2: An example 2D tree structure generated using a 2D version of Algorithm 1 [7]

Initialisation
| Begin with a list of particles, a tree structure of boxes, and a value for the precision parameter θ
for *each particle* **do**
| CalculateForce(particle, coarsest level box, θ)
end

Function `CalculateForce`(*particle, box,* θ)**:**
| **if** *box contains a single particle, which is not the subject particle* **then**
| | Add the contribution of box to the particle force
| **else if** *box edge length / distance between box centre and particle* $< \theta$ **then**
| | Add a contribution to the total force on the subject particle using the box CoM position and total mass.
| **else**
| | **for** *each each child box corresponding to box* **do**
| | | CalculateForce(particle, child box, θ)
| | **end**
| **end**
| **return**

Algorithm 2: BH force calculation

The BH method has a theoretical complexity of

$$\mathcal{O}(N \log N) \tag{6}$$

[4], although this does not take consideration of the precision parameter, θ. Barnes and Hut also reported that the force error scales approximately as the -1.5 power of computation time. Testing of this claim, as well as determination of the effect of θ on computation time are undertaken in, §4.

The BH method is popular due to its ease of implementation, especially in three dimensions. However, as implied by the use of the term mass in this discussion, the algorithm performs poorly when considering both attractive and repulsive interactions simultaneously.

2.3 The Fast Multipole Method

The fast multipole method exploits the convergence of multipole expansions within complex space to predict the potential field due to a collection of charges at a general well-separated position. An outline of the FMM algorithm in 2D follows below. However, complete descriptions of the FMM in 2D and 3D can be found in papers [1] and [8] respectively

Like the BH method, the FMM relies on an upward and downward pass approach. First, similarly to the BH method, and as depicted in Figure 2 the system is divided into a quad-tree grid structure. The approach implemented here results in $n \approx \log_4 N$ levels to this tree structure in all cases (level 0 represents the undivided system). However, the algorithm can be adapted to function with a dynamic tree structure like that described for the BH method [9]. Next, each particle is loaded into the tree level n box overlapping its position. This then allows construction of the multipole expansion for the potential field due to all particles within each level n box at relative position z about the centre of that box using equation (7),

$$\phi(z) = -Q \log(z) - \sum_{k=1}^{\infty} \frac{a_k}{z^k}\,, \tag{7}$$

where

$$Q = \sum_{i=1}^{p} q_i\,, \tag{8}$$

$$a_k = \sum_{i=1}^{p} \frac{-q_i z_i^k}{k}\,. \tag{9}$$

Here, q_i is the property (either charge or mass-like) of each particle i within the box, z_i is the complex position of particle i relative to the centre of that box, and $p \approx \log_2 \epsilon$ is the order to which all expansions will be calculated to achieve theoretical precision ϵ [1]. The

multipole expansions of the level n boxes corresponding to a level $n-1$ parent box can then be combined to form the multipole expansion of that level $n-1$ box. This is done by summing the results from shifting the centre of each child box multipole expansion using equation (10),

$$\phi(z) = -a_0 \log(z) - \sum_{l=1}^{\infty} \frac{b_l}{z^l}\,, \tag{10}$$

where

$$b_l = \left(\sum_{k=1}^{l} a_k z_0^{l-k} \binom{l-1}{k-1} \right) - \frac{a_0 z_0^l}{l}\,, \tag{11}$$

and $\binom{l-1}{k-1}$ are the binomial coefficients [1]. Here, z and z_0 are the complex positions for the centres of the boxes at level $n-1$ and level n respectively.

The next section of the algorithm is a downwards pass. At each box tree level, a local expansion for potential within each box, $\psi(z)$ is calculated, accounting for the interactions with boxes at the same tree level which are well separated from the box, and have not been accounted for in the calculation for the potential of the parent box potential. The list of boxes fulfilling this condition for box i is the interaction list of box i. For example, in Figure 3, the boxes marked x are within the interaction list of box i at tree level 3. The contribution from each box in the interaction list of box i is given by equation (12),

$$\phi(z) = -\sum_{l=0}^{\infty} c_l z^l\,, \tag{12}$$

where

$$c_0 = \sum_{k=1}^{\infty} \frac{a_k}{z_0^k}(-1)^k + a_0 \log(-z_0)\,, \tag{13}$$

and

$$c_l = \left(\frac{1}{z_0^l} \sum_{k=1}^{\infty} \frac{a_k}{z_0^k} \binom{l+k+1}{k-1} (-1)^k \right) - \frac{a_0}{l \cdot z_0^l} \text{ for } l \geq 1\,. \tag{14}$$

Here, z_0 is the complex relative position of the centre of box i relative to the centre of the box in the interaction list being accounted for. The total local expansion, $\psi(z)$ within the subject box is therefore given by the sum of the expansions for each box in the interaction list. Once the expansion for $\psi(z)$ has been found for a box at a particular level, the expansion is shifted to the centre of each of its child boxes. This shifted expansion is added to the local expansion, $\psi(z)$ about the child box centre, hence accounting for all interactions affecting the child box from beyond the neighbours of the child parent box. This shifted expansion is calculated using equation (15),

$$\sum_{k=0}^{n} a_k (z - z_0)^k = \sum_{l=0}^{n} \left(\sum_{k=l}^{n} a_k \binom{k}{l} (-z_0)^{k-l} \right) z^l\,, \tag{15}$$

where z is the complex position of the centre of the child box, and z_0 is the complex position of its parent box. This process of local potential calculation and propagation to the next tree level continues until a local expansion has been calculated for each box at level n, and encodes the potential due to all particles not in the immediate neighbour boxes of a particular box at level n. Finally, the potential for each particle within a

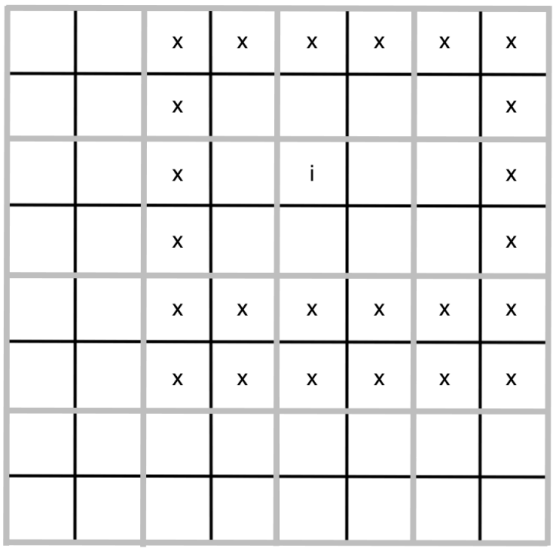

Figure 3: Members of the interaction list of box i at tree level 3 are marked x. Parent cell edges are marked in grey. Modified from [1].

box at level n can be calculated by evaluating (15) using the derived coefficients for the expansion within that box, taking its real part, and adding to the result the potential due to interactions with other particles within the box and its closest neighbour boxes using (2) [1].

Due to the analytic nature of this analysis, the force on a particle may also be calculated without introducing additional error by noting that "where $u(x, y) = \mathcal{R}(w(x, y))$ describes the potential field at (x, y), the force field is given by

$$\boldsymbol{\nabla} u = (u_x, u_y) = (\mathcal{R}(w'), -\mathcal{I}(w'))\,, \tag{16}$$

where w' is the derivative of w" [1]. A pictorial representation of the FMM algorithm is presented in Figure 4.

Greengard and Rokhlin [1] report that the expected precision of this algorithm is $\epsilon \approx 2^p$. They also report the expected computational cost as

$$\mathcal{O}(N[-2a\log_2(\epsilon) + 56b(\log_2(\epsilon))^2 + 4.5ck_n + d])\,, \tag{17}$$

where a, b, c and d are coefficients determined by the specific implementation, language and hardware, and k_n is the maximum number of particles per box at level n. Both of

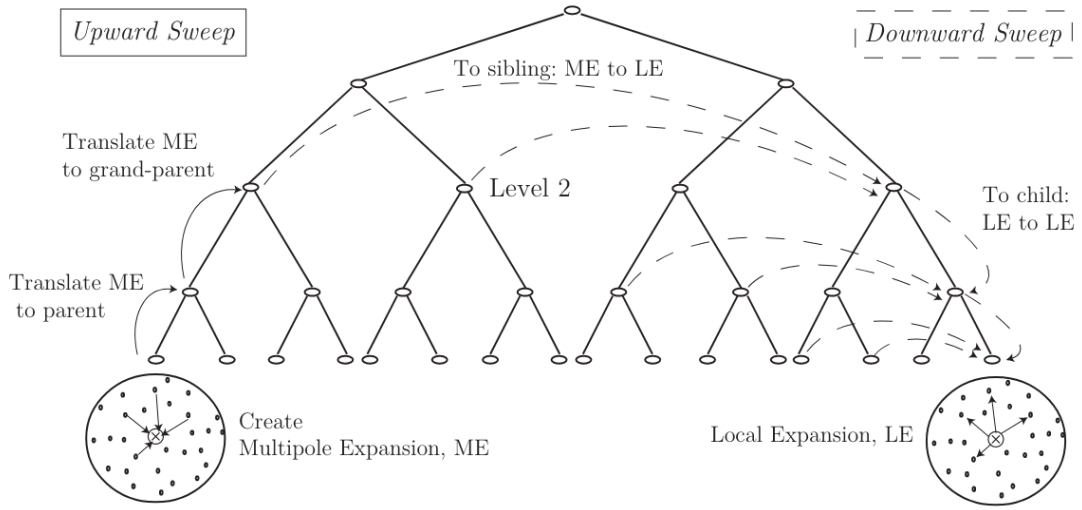

Figure 4: A schematic representation of the FMM algorithm in a system with $n = 4$ [10]

these metrics will be tested in §4.

3 Method

3.1 Experimental Process

To quantify the real-world performance of the BH and FMM algorithms, they, along with a direct calculation simulation, were implemented in Python [5], as described in §2. The additional modules used were numpy [11], matplotlib [12], scipy [13], and the python library modules [5] copy, typing, math and timeit.

Simulations were run using these implementations to enable production of figures quantifying how simulation time varies with the number of particles undergoing simulation, the precision parameter used (θ for BH and p for FMM), the minimum precision achieved and the average precision achieved. Furthermore, the variation in precision with the corresponding parameter and N, along with the extent to which measured simulation times match theoretical predictions (6) and (17) are also investigated. In each case, the values either $\theta = 10$, 2, 1.5, 1.25, 1, 0.9, 0.7, 0.5 and 0.3 or $p = 0$, 1, 4, 7, 10, 13, 16, 19, 22, 25, 28 and 31 are used with $N = 4$, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 and 8192. Four different initial conditions were used, given by two different particle distributions, either uniform throughout the simulation volume, or with the probability of occupation at a point along an axis, x, given by the beta distribution (18),

$$p(x) = \frac{x(1-x)}{\int_0^1 x(1-x)}\,, \tag{18}$$

Simulation Scenario	Particle Distribution	Particle Property Distribution
A	Uniform	Uniform $(0 \to q_{\max})$
B	Uniform	Uniform $(-q_{\max} \to q_{\max})$
C	Beta	Uniform $(0 \to q_{\max})$
D	Beta	Uniform $(-q_{\max} \to q_{\max})$

Table 1: The different simulation initial conditions used.

which is then scaled to the size of the system. For each particle distribution, simulations were undertaken with random, positive particle properties (up to a maximum value, $q_{\max}$), as well as with random particle properties ranging from $-q_{\max}$ to $+q_{\max}$. These test regimes are labelled A, B, C and D, as seen in Table 1.

3.2 Preliminary Testing

The python implementations mentioned above were initially tested in simple environments, such as those in which only a single particle is present, and where a large number of particles are introduced and the simulation results compared with direct calculation results.

3.2.1 Barnes-Hut Method Testing

Results from single particle simulations using the BH method, as expected, agreed with analytic results to within the python float128 accuracy limit on a 2019 MacBook Pro, around 1×10^{-18}. An example result set is depicted in Figure 5. In the many particle

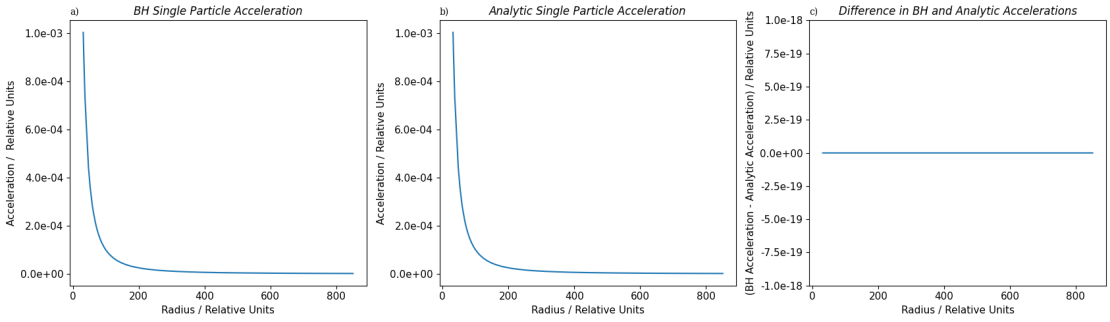

Figure 5: a) Analytic results for acceleration from a single particle in 3D space, b) BH simulation results for the same situation and c) The difference between these results.

test case, the BH algorithm also performed well, achieving log_{10}(Relative Errors) around -2.5, as can be seen in Figure 6. Throughout this analysis, relative error will be defined as

$$\text{Relative Error}(A - B) = \frac{A - B}{B}\,. \tag{19}$$

The many particle case was repeated for particles with a random distribution of both positive and negative properties, and as expected, the BH method was unable to accurately reproduce the direct calculation result, as seen in Figure 7.

To ensure BH precision increased with corresponding change in precision parameter θ, tests were run with $\theta = 10,\ 2,\ 1.5,\ 1.25,\ 1,\ 0.9,\ 0.7,\ 0.5,$ and 0.3. Each decrease in θ was seen to improve the minimum and average precision of the simulation, as is shown in Figure 8.

These results were all found to agree with the expected functioning of the BH algorithm, providing evidence that the results and conclusions presented in §4 are reliable.

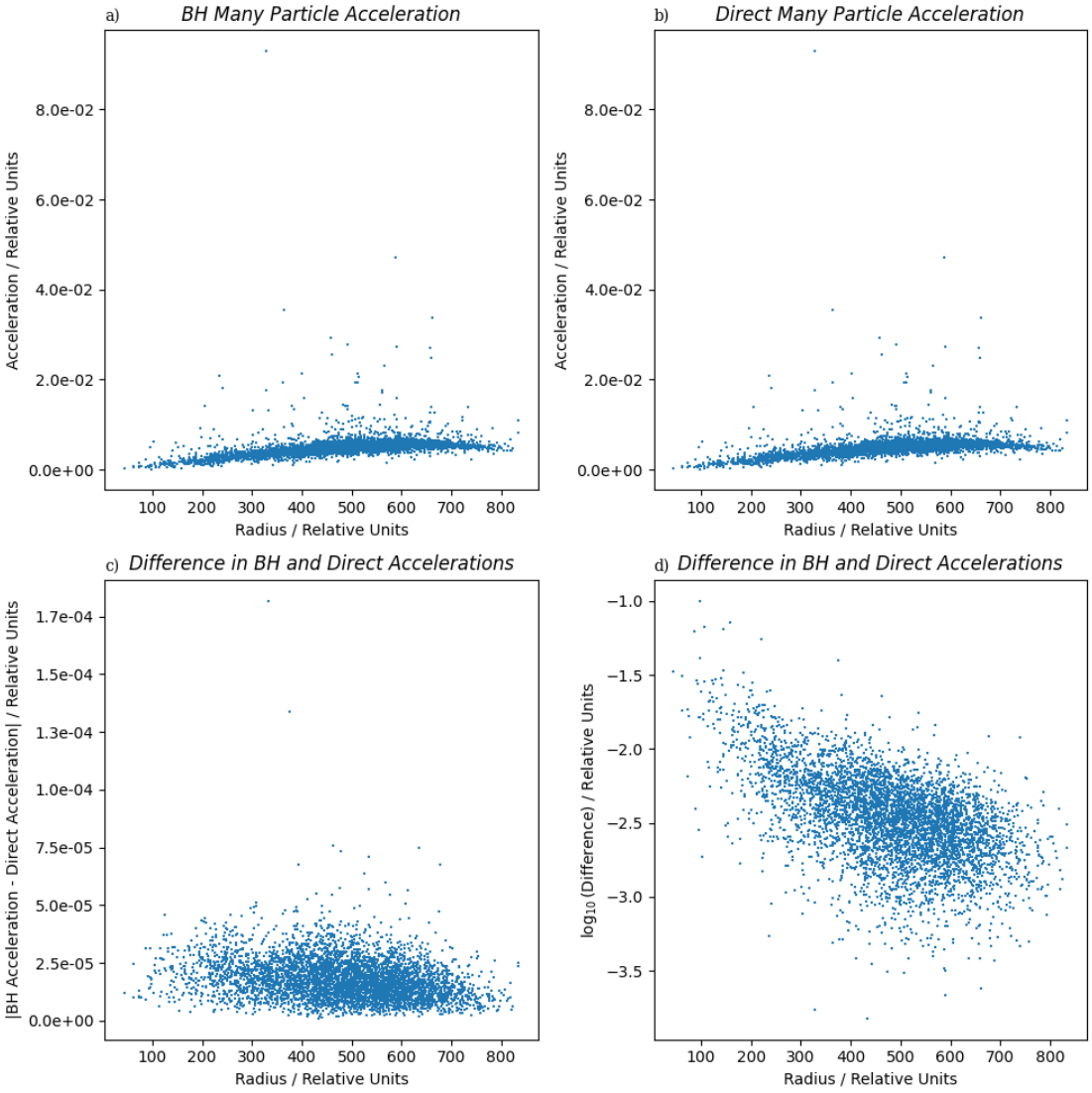

Figure 6: a) Analytic results for acceleration from a random spherical distribution of particles with positive property in 3D space, b) BH simulation results for the same situation c) The difference between these results, and d) The logarithm of this difference.

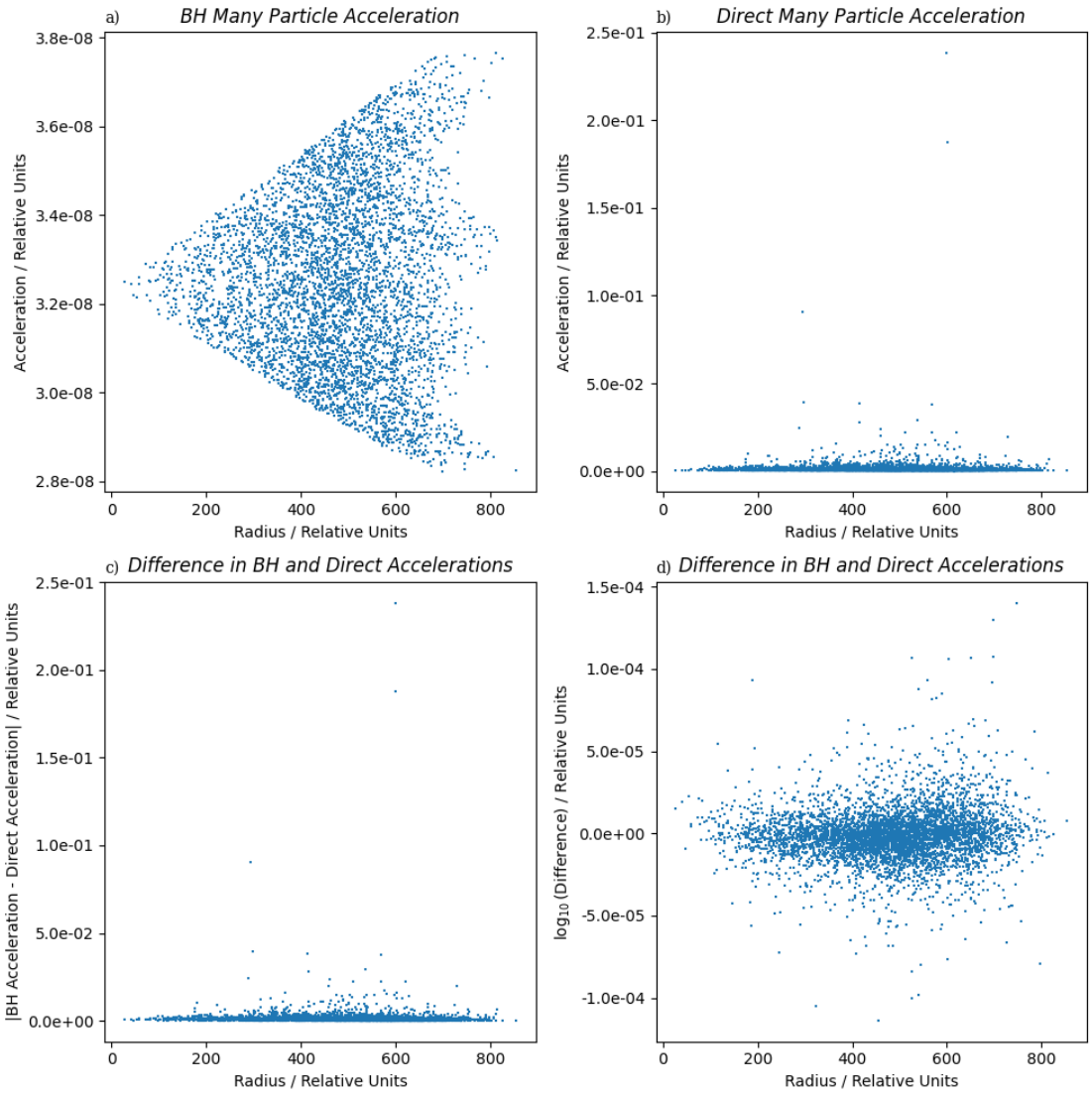

Figure 7: a) Analytic results for acceleration from a random spherical distribution of particles with both positive and negative properties in 3D space, b) BH simulation results for the same situation c) The difference between these results, and d) The logarithm of this difference.

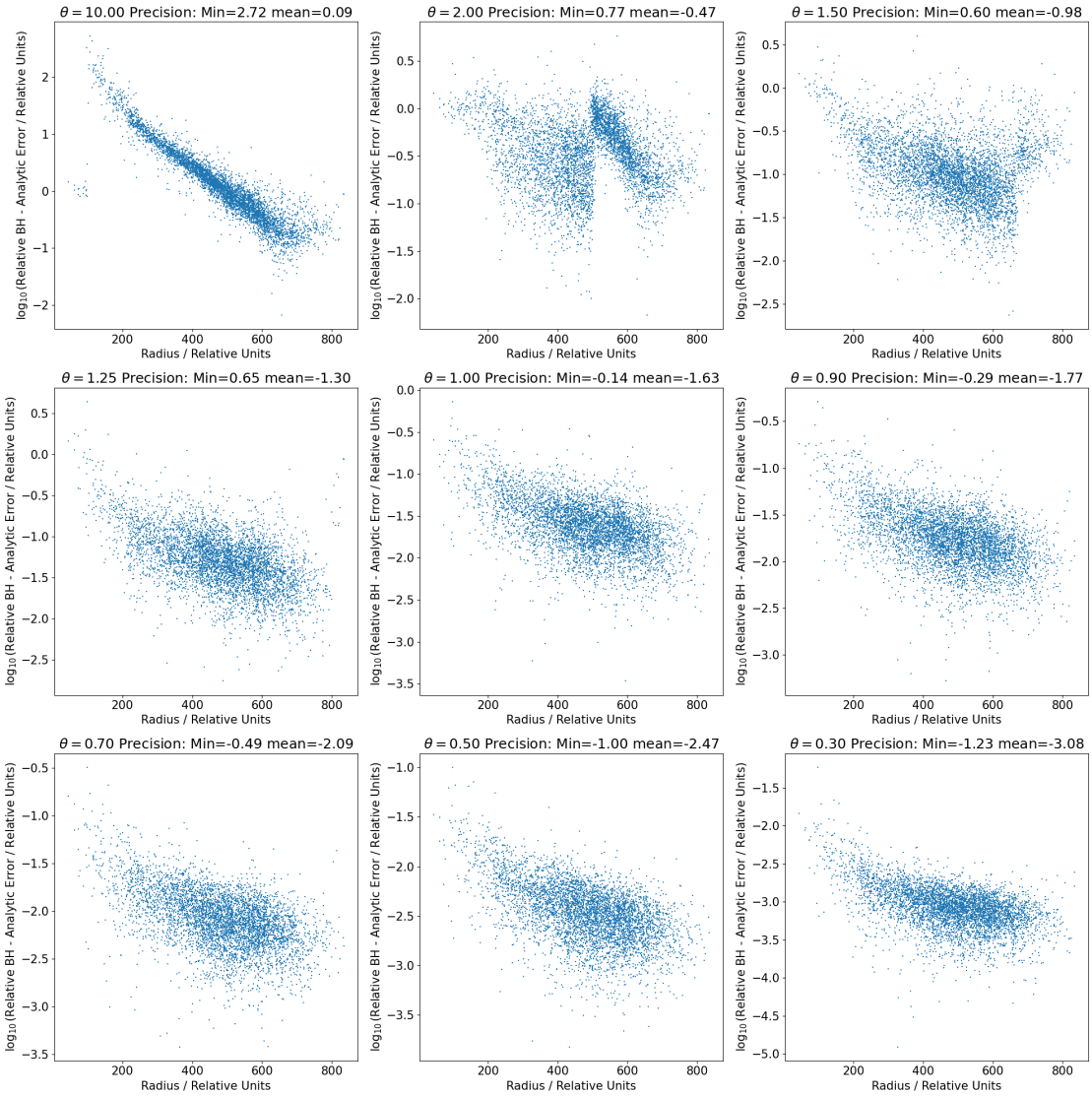

Figure 8: How to relative precision of the Barnes-Hut method changes with precision parameter θ for fixed $N = 5000$.

3.2.2 Fast Multipole Method Testing

To test the reliability of the produced FMM implementation, potentials for a single non-zero property particle were evaluated at three stages of the algorithm; the initial multipole expansion at the finest tree level, the shifted multipole expansion about the level 2 box containing the non-zero property particle, and the final FMM potential estimate for the whole system. Results from these tests for a range of p values, are reported in Figures 9, 10 and 11 respectively.

As expected, the potential differences between the multipole expansion at the finest level converges rapidly for most p values, and that at the coarsest level also converges quickly, but only reaches computational round-off error within the simulation box at $p = 16$, much slower than the fine level expansion which reaches the same level by $p = 7$. These results are both consistent with the radius of convergence expectations explained in [1]. Furthermore, the whole system estimates obtained in Figure 11 agree very well with those presented in Figure 12, from [10]. This excellent agreement between expected performance and simulation performance provide confidence in the reliability of these simulations for further experiments which follow in §4.

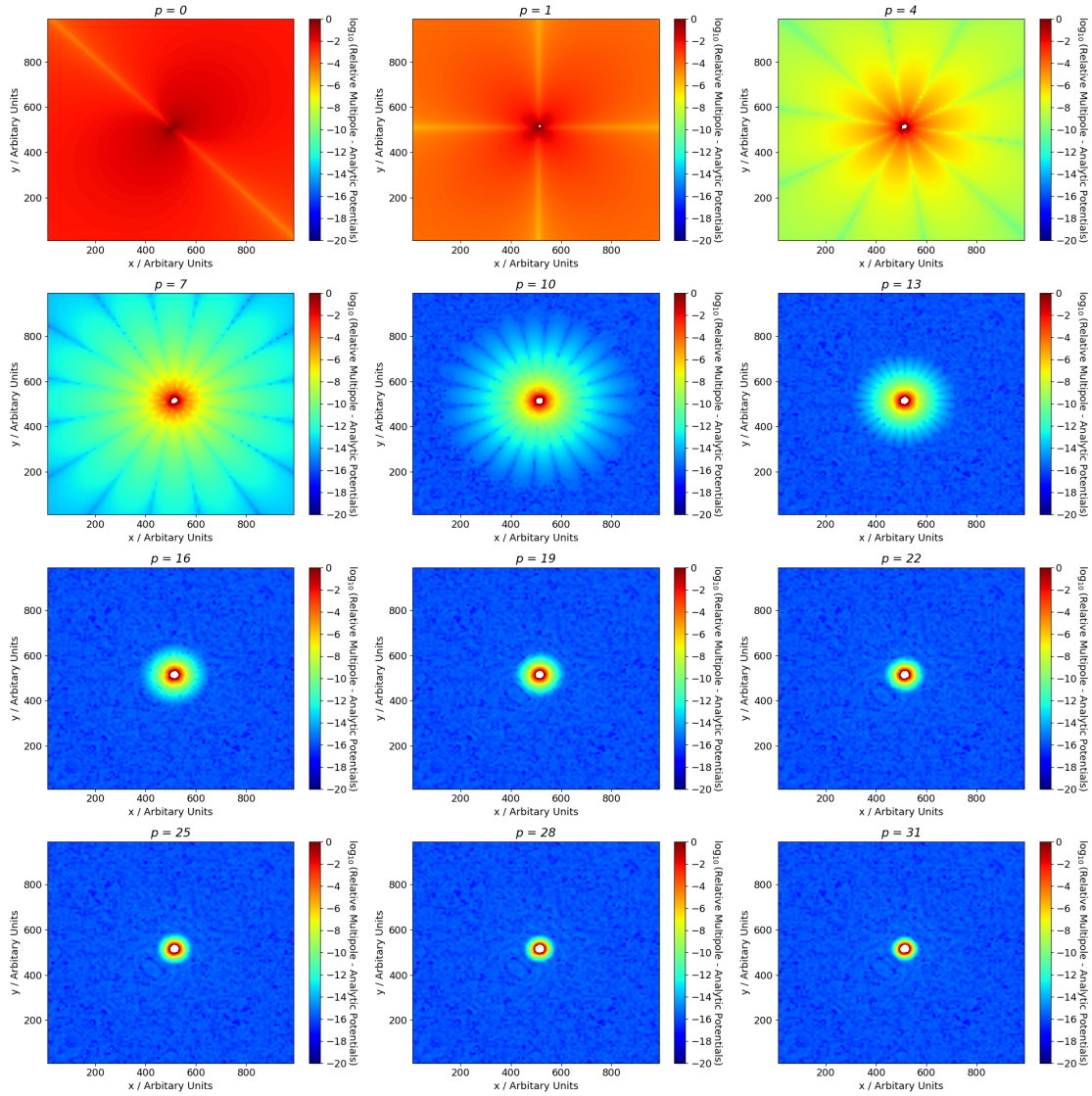

Figure 9: How the precision with which the fine level multipole expansion approximates the true potential for a single particle with non-zero property located at $x = 500$ and $y = 500$ for a range of p values.

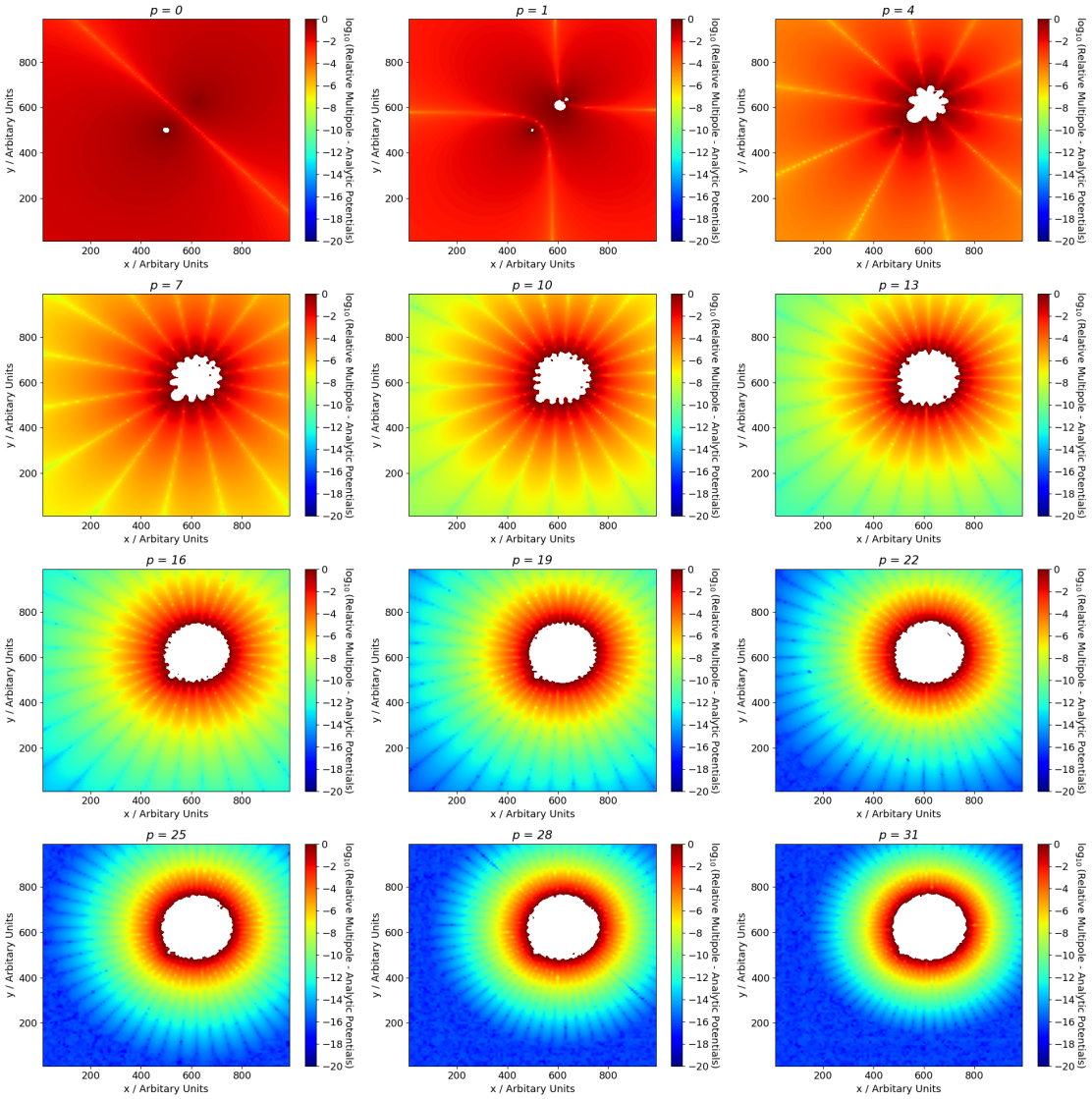

Figure 10: How the precision with which the coarse (tree level 2) multipole expansion approximates the true potential for a single particle with non-zero property located at $x = 500$ and $y = 500$ for a range of p values.

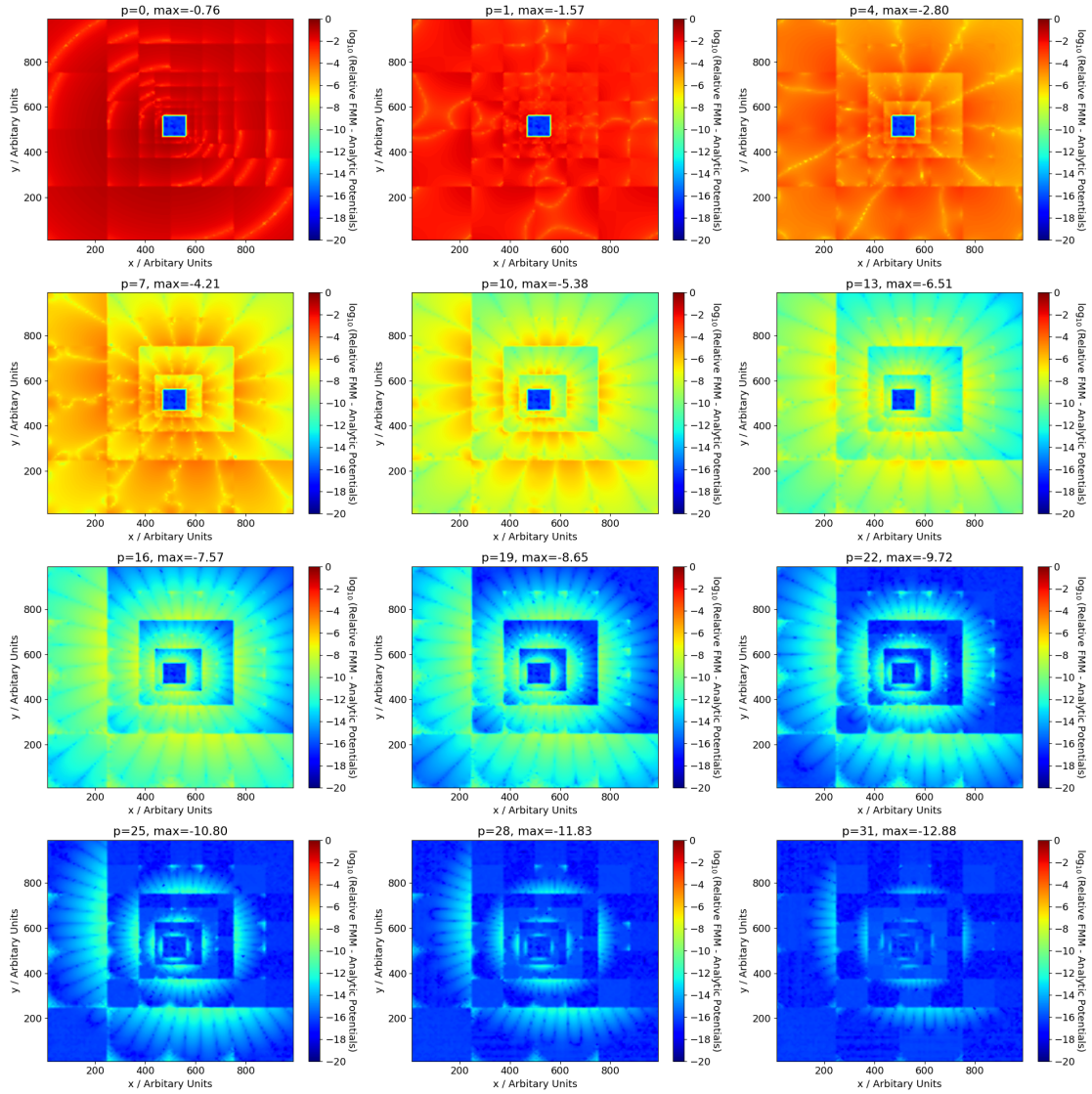

Figure 11: How the precision with which the fast multipole method evaluates the potential for a single particle with non-zero property located at $x = 500$, $y = 500$ change for a range of p values, with $n = 5$.

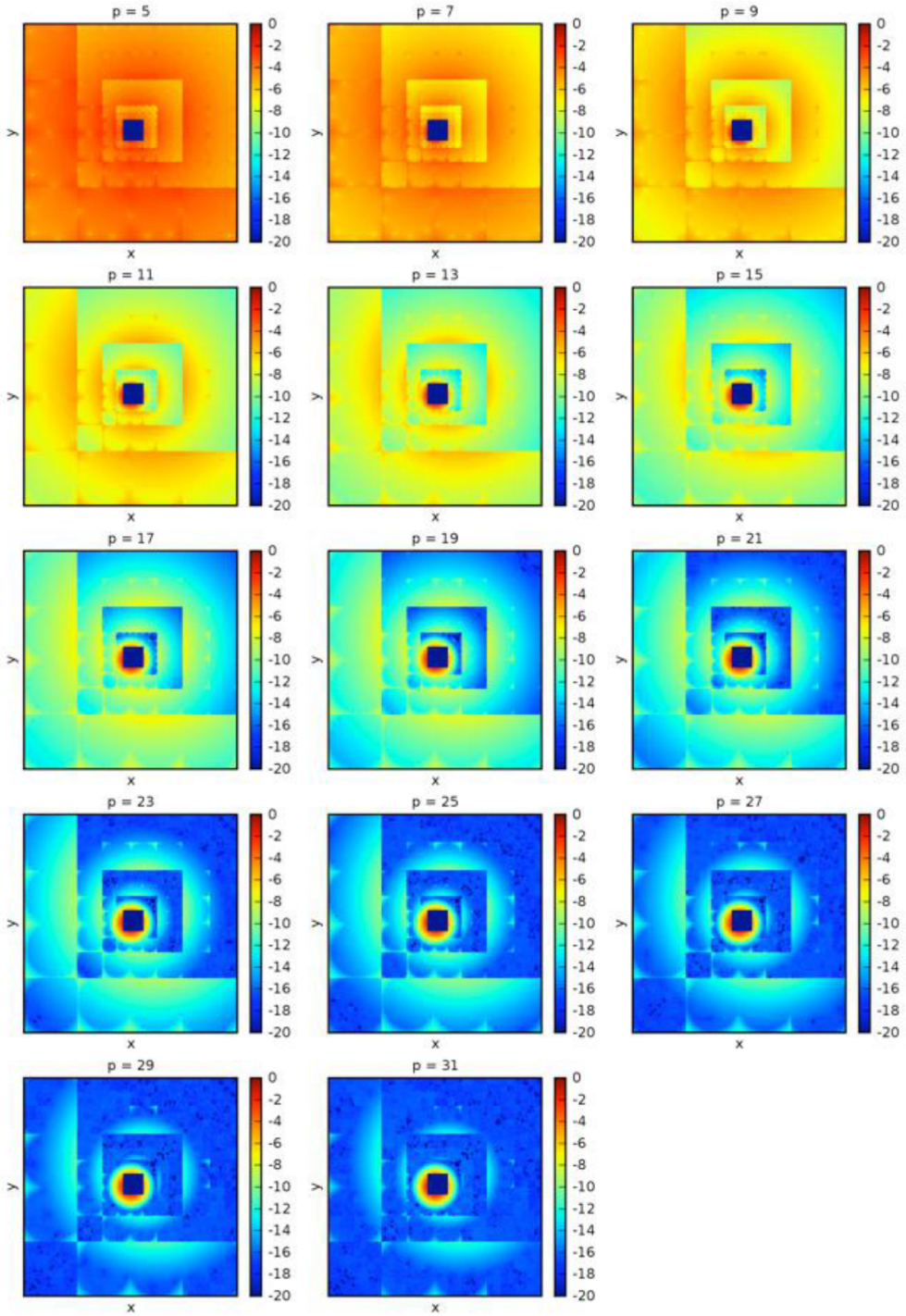

Figure 12: Literature expectation for how the precision with which the fast multipole method evaluates the potential for a single particle with non-zero property, located in the centre of the simulation box for a range of p values and with $n = 5$, from [10].

4 Results and Discussion

The tests described in §3.1 were undertaken, and the results of each are presented in Figures 13, 14, 15, and 16. These figures correspond to the initial conditions A, B, C and D, respectively, as defined in Table 1. In each case, sub-figures a) and b) show how FMM and BH simulation times vary with N and precision parameter; c) and d) show how simulation times varies with N and mean simulation result error achieved relative to a direct calculation; e) and f) show the same as c) and d), but for the maximum simulation result error relative to direct calculation; g) and h) show how well the simulation times fit theoretical expectations; finally, i) and j) show how the minimum precision achieved by each simulation varies with N and the precision parameter. These figures, along with the computational cost analysis below, can be used to predict the performance of these algorithms in a range of conditions.

4.1 Barnes-Hut Results

In the BH case, it was found that a fit to (6) was poor due to variation with the θ parameter used. Hence, to identify the full form for how computational cost varies, including θ in BH simulations, further analysis was performed. By plotting how simulation time varies for fixed N across a range of θ values, it was found that an optimised fit took the form

$$\text{simulation cost} = a \times (1/\theta)^b \,, \tag{20}$$

as depicted in Figure 17. The ways in which fit parameters a and b then vary with particle number was then also investigated, as seen in Figure 18. This variation is itself fitted by the form

$$y = c \times x + d \,. \tag{21}$$

Hence, the simulation cost for a BH implementation was found to take the form

$$\mathcal{O}(a \times N \log(N) + b \times N \times \theta^{c \times log_2(N)}) \,. \tag{22}$$

This is the form used to obtain subplot h) in each of the primary result Figures 13, 14, 15, and 16. It should also be noted that the triangular sections of results in BH subplots d) and f) are a consequence of linear interpolation, and hence the results within these regions of the plots are linear estimates, which may be unrealistic.

By comparing the primary result figures, it is clear that the BH algorithm generally produces results with relative precision $\approx 10^{-2}$ with a computation time $\lesssim$ that predicted by (22). Therefore, convergence to higher precision levels is slow. It is also again seen that convergence fails completely in simulations B and D, where the particles that undergo simulation have both positive and negative property values. Performance was otherwise found to be similar across the whole simulation set, regardless of initial conditions.

4.2 Fast Multipole Method Results

The FMM simulation performance fit was constructed using optimised parameters for (17). It was found that this form predicted performance to around one order of relative magnitude, as seen in subplot g) in each primary result figure, and can hence be used to estimate the computational cost of further simulations - which approximately takes the form $\mathcal{O}(N)$ for a given p value.

FMM simulations, were able to quickly reproduce the direct simulation results for the system to a level limited by the computational round-off precision, demonstrating the great power of this approach. Due to the implementation difficulty, FMM is often not used, but this work highlights how major its advantages can be over BH simulation for a given computation time.

The FMM algorithm also demonstrated good performance for both the uniform and beta distribution initial condition cases, although relative errors in the beta case were found to be around two orders of magnitude higher than those in the uniform case. This highlights that the FMM performs best in homogeneous environments. Another interesting observation from subplot a) in each primary result figure is that simulation times are lower where $\log_2(N)$ is even - whether this is an effect of the algorithm or the implementation requires further research.

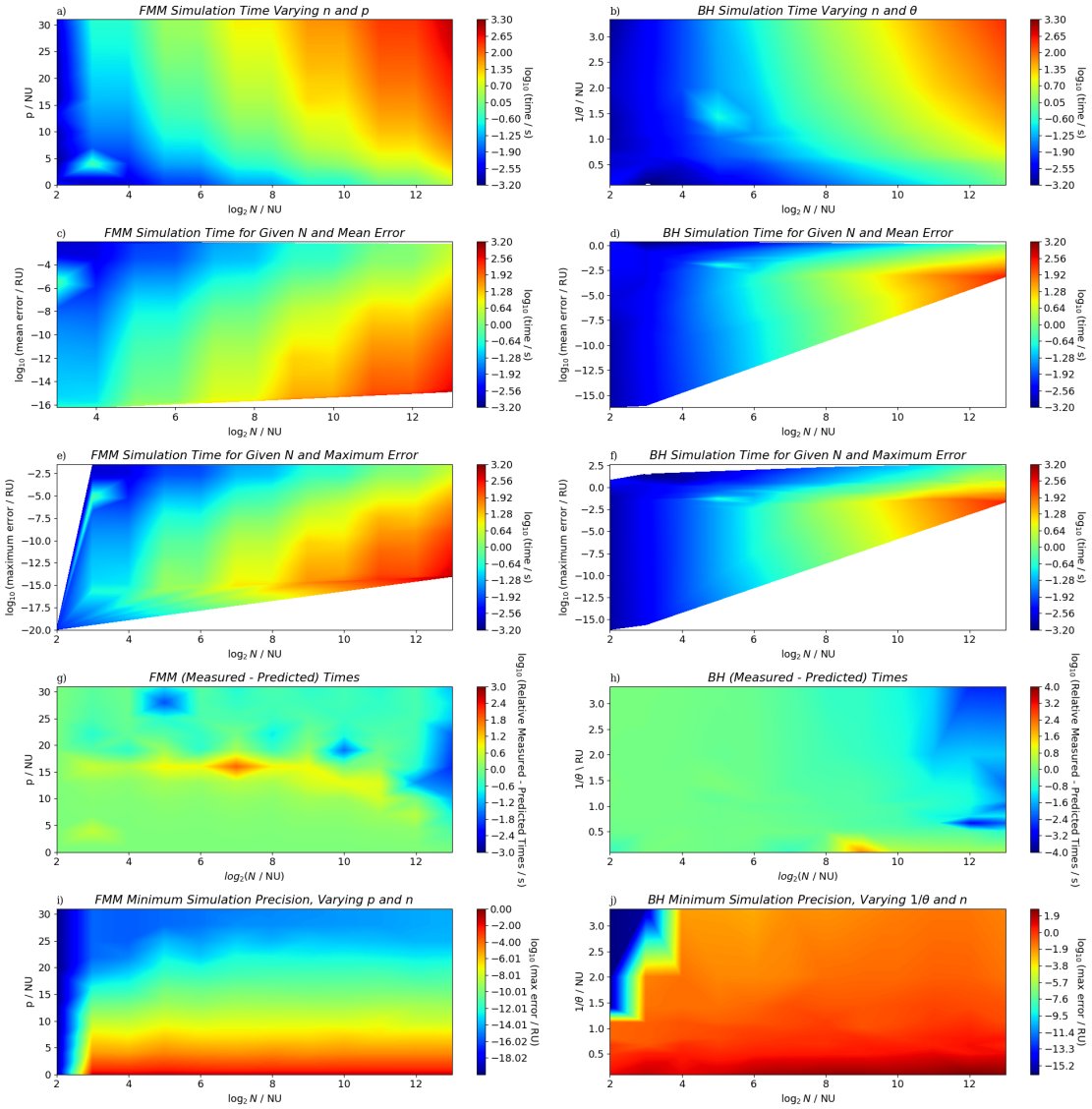

Figure 13: Results for simulation A. The fit coefficients for i) and j) are $a = 1.1 \times 10^{-4}$, $b = 1.9 \times 10^{-5}$, $c = 8.1 \times 10^{-5}$, and $d = -2.9 \times 10^{-2}$ in the FMM case and $a = -9.1 \times 10^{-6}$, $b = 2.1 \times 10^{-3}$, and $c = -1.7 \times 10^{-1}$ in the BH case.

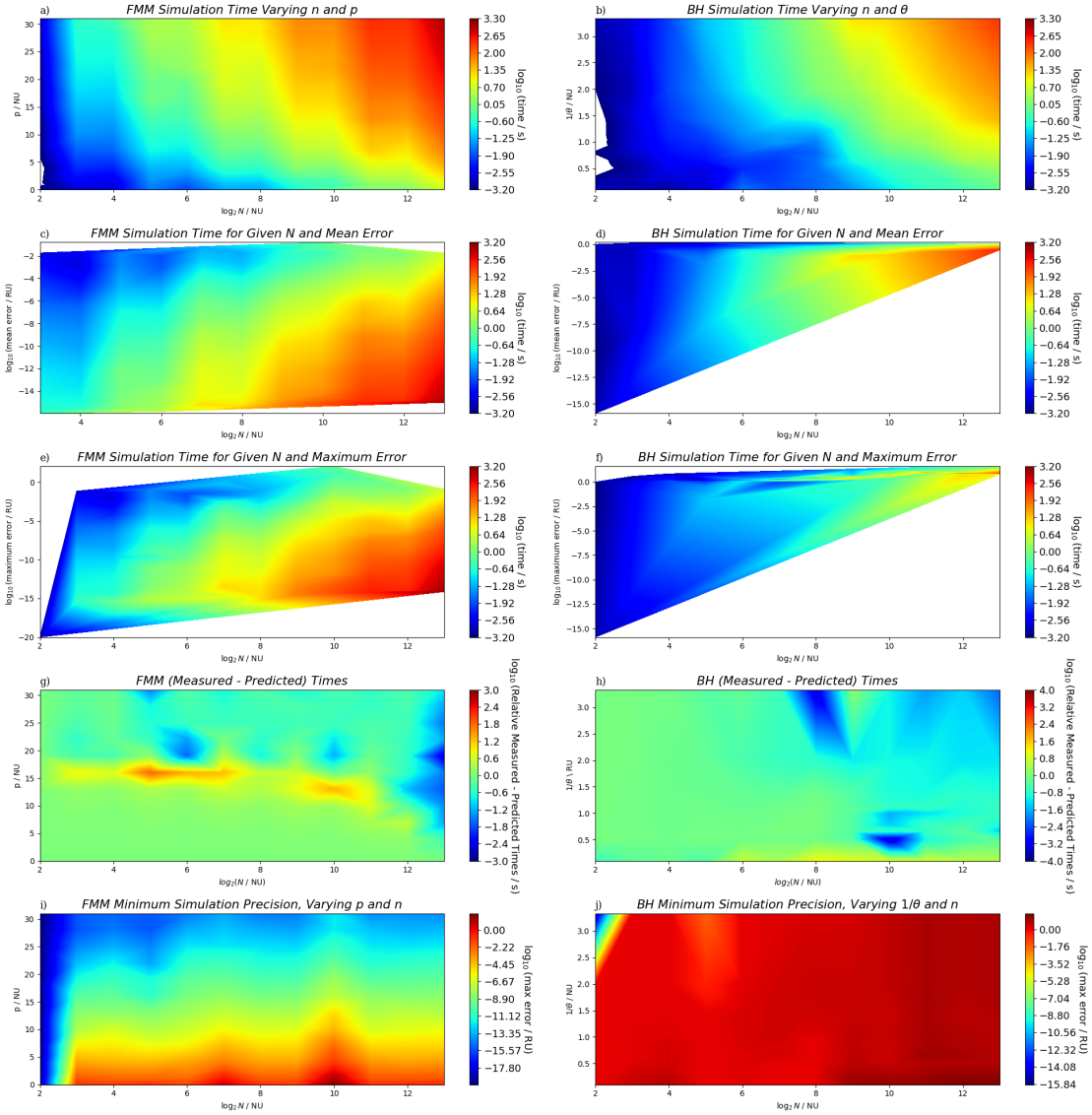

Figure 14: Results for simulation B. The fit coefficients for i) and j) are $a = 1.9 \times 10^{-4}$, $b = 2.3 \times 10^{-6}$, $c = 8.5 \times 10^{-5}$, and $d = -3.0 \times 10^{-2}$ in the FMM case and $a = -4.6 \times 10^{-6}$, $b = 1.2 \times 10^{-3}$, and $c = -2 \times 10^{-1}$ in the BH case.

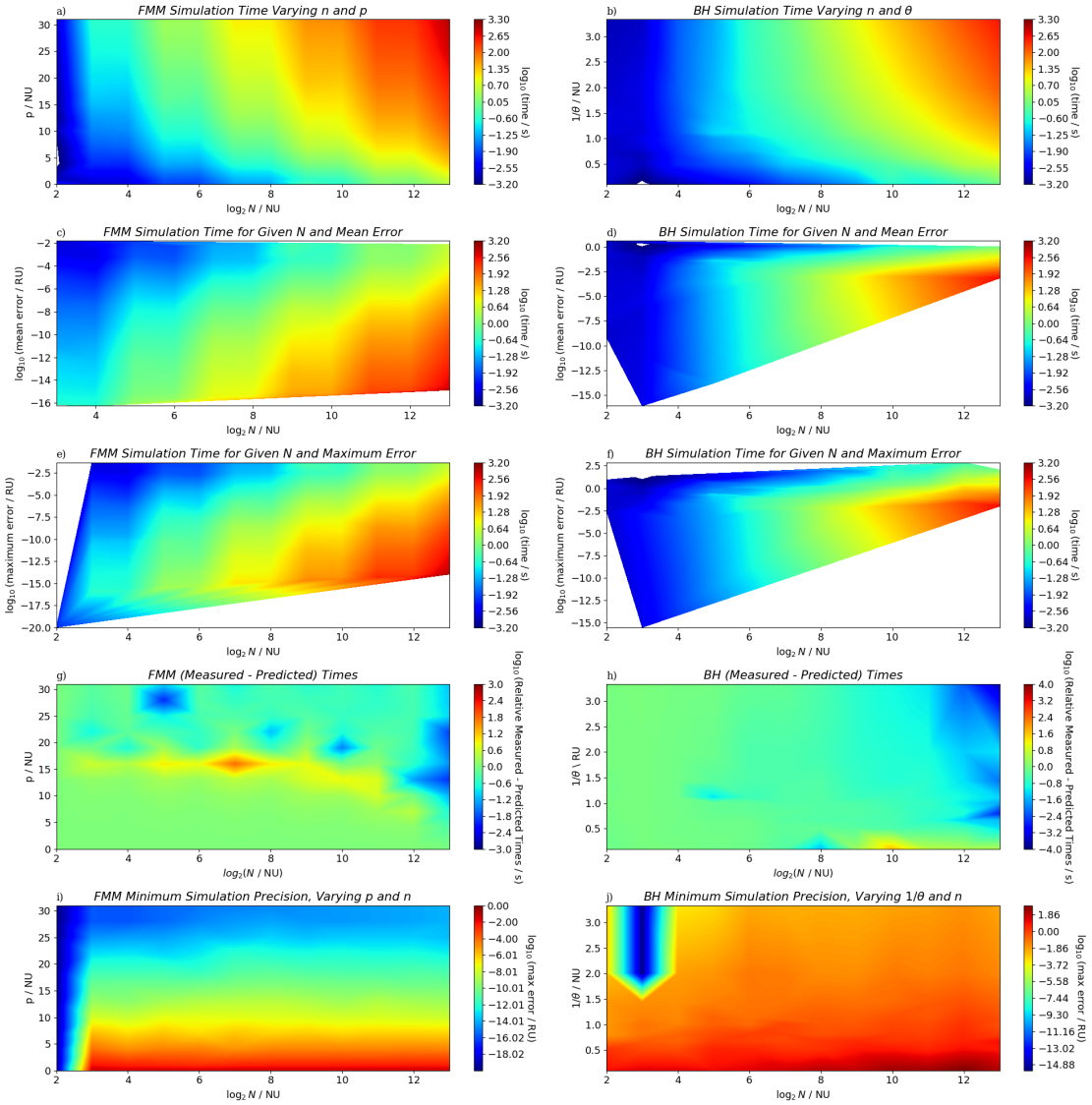

Figure 15: Results for simulation C. The fit coefficients for i) and j) are $a = 9.9 \times 10^{-5}$, $b = 2.0 \times 10^{-6}$, $c = 8.5 \times 10^{-5}$, and $d = -3.1 \times 10^{-2}$ in the FMM case and $a = -9.1 \times 10^{-6}$, $b = 3.0 \times 10^{-3}$, and $c = -1.7 \times 10^{-1}$ in the BH case.

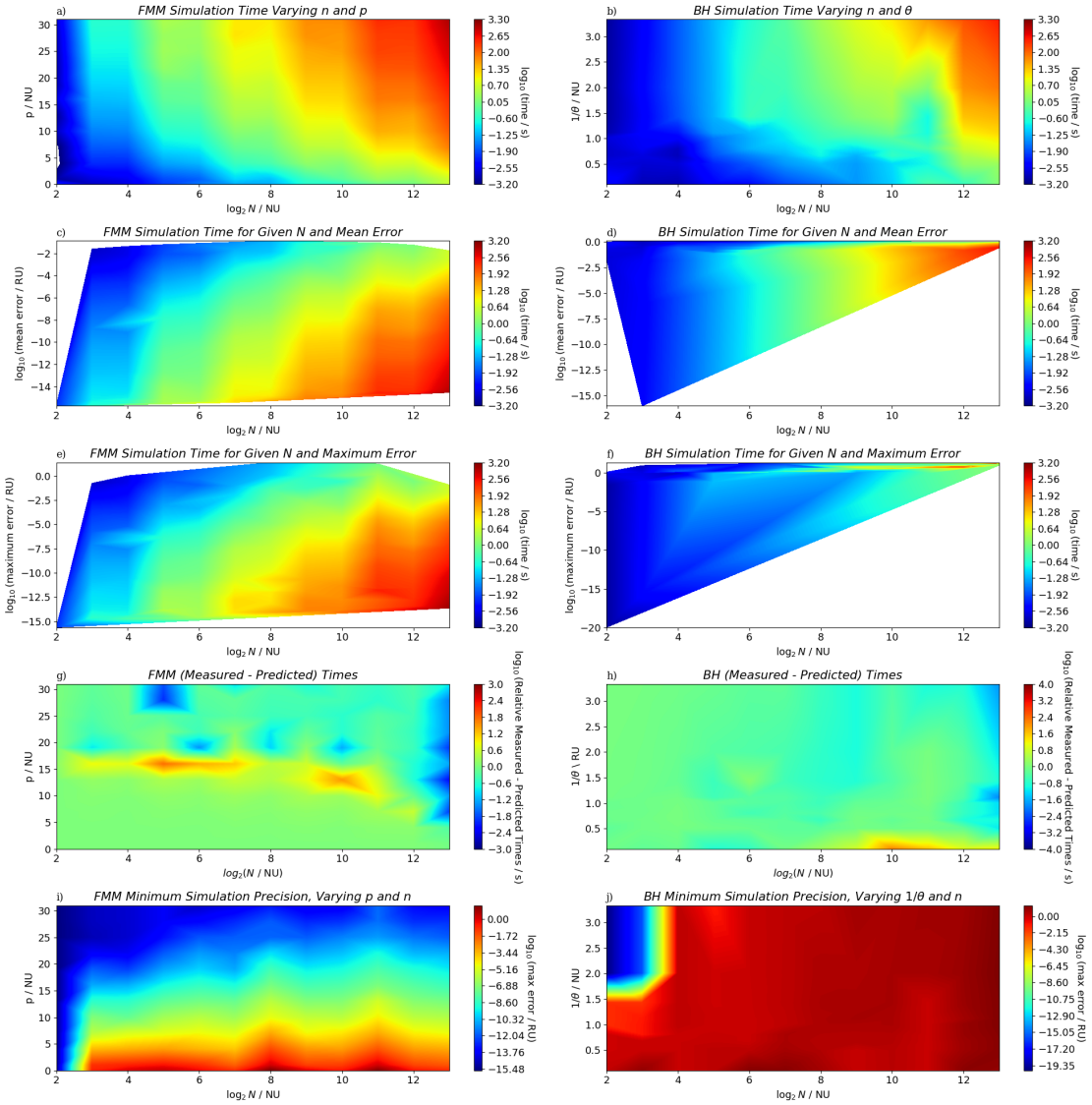

Figure 16: Results for simulation D. The fit coefficients for i) and j) are $a = 1.5 \times 10^{-4}$, $b = 2.2 \times 10^{-6}$, $c = 8.4 \times 10^{-5}$, and $d = -3.0 \times 10^{-2}$ in the FMM case and $a = -1.8 \times 10^{-6}$, $b = 1.6 \times 10^{-3}$, and $c = -2.1 \times 10^{-1}$ in the BH case.

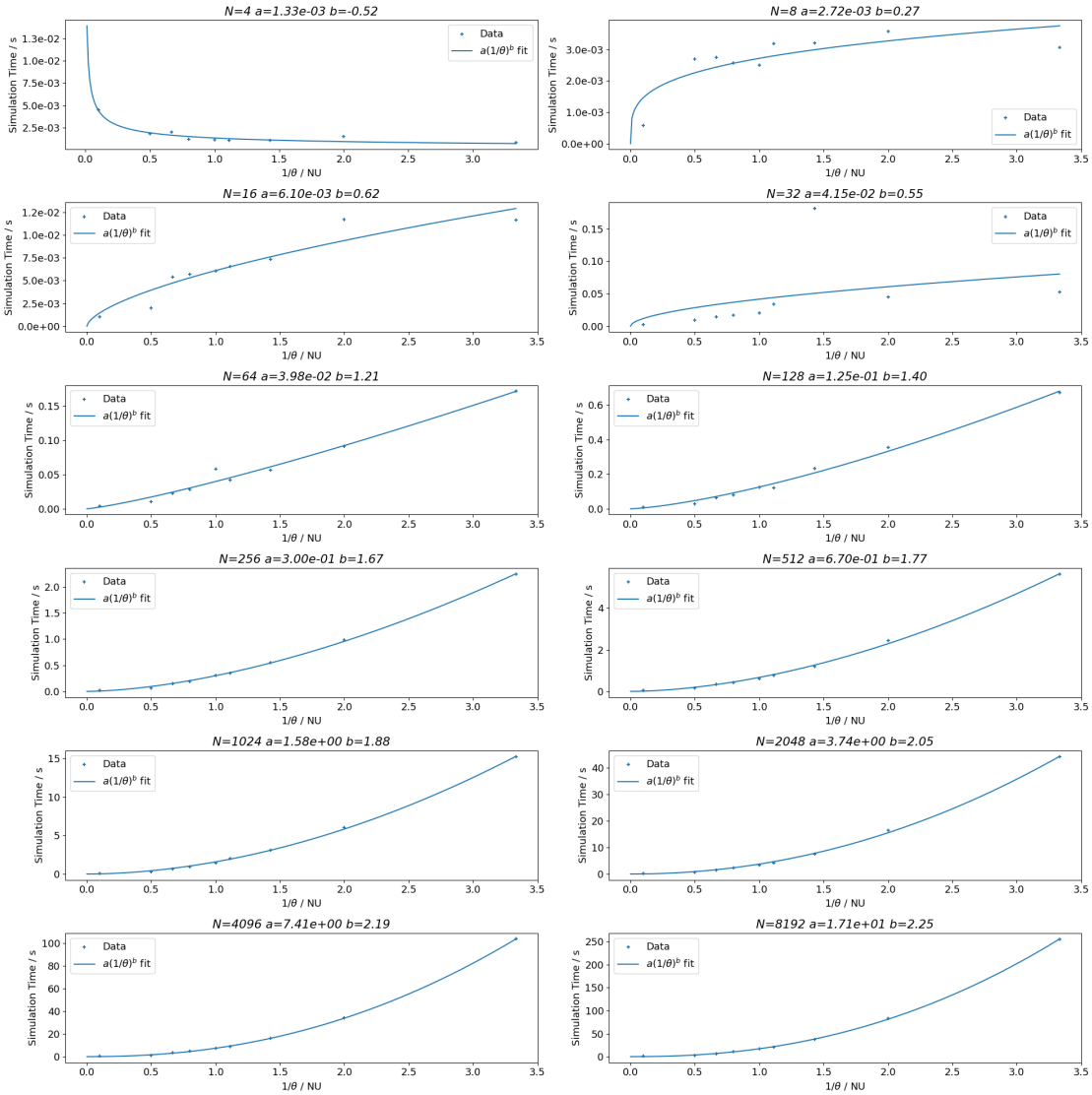

Figure 17: The way in which the computational cost of the BH algorithm varies with θ, and fitted according to (20).

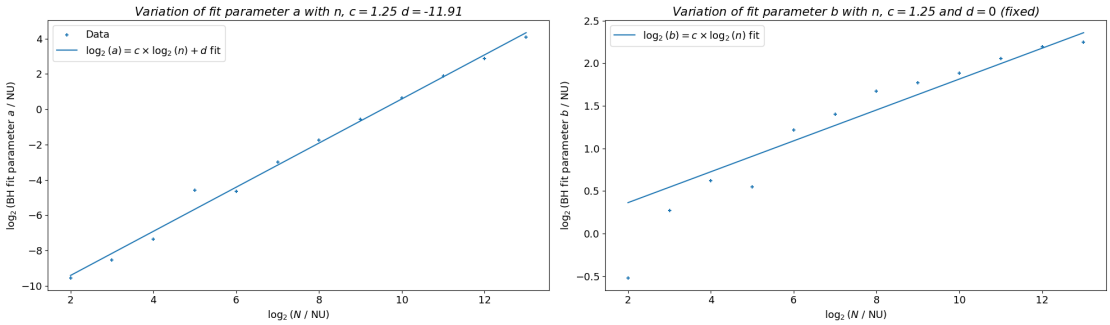

Figure 18: The way in which the fit parameters in Figure 17 vary with N

5 Conclusions

The theory behind the FMM and BH algorithms has been presented, along with results from simulations covering a wide range of initial conditions, via variation of the precision parameters involved in both algorithms, the number of particles to simulate, and the initial distribution of those particles. It has been seen that for a given simulation time, the FMM algorithm produces results with much higher precision, although it must be noted that the FMM implementation was only conducted in 2 dimensions. Results were also presented to enable prediction of computational cost for other particle counts and precision parameters, and it was found that the computational cost of the BH algorithm was found as (22).

References

[1] L Greengard and V Rokhlin. "A fast algorithm for particle simulations". In: *Journal of Computational Physics* 73.2 (1st Dec. 1987), pp. 325–348. ISSN: 0021-9991. DOI: 10.1016/0021-9991(87)90140-9.

[2] Konstantin N Kudin and Gustavo E Scuseria. "A fast multipole algorithm for the efficient treatment of the Coulomb problem in electronic structure calculations of periodic systems with Gaussian orbitals". In: *Chemical Physics Letters* 289.5 (19th June 1998), pp. 611–616. ISSN: 0009-2614. DOI: 10.1016/S0009-2614(98)00468-0.

[3] Elias Rudberg and Paweł Sałek. "Efficient implementation of the fast multipole method". In: *The Journal of Chemical Physics* 125.8 (23rd Aug. 2006), p. 084106. ISSN: 0021-9606. DOI: 10.1063/1.2244565.

[4] Josh Barnes and Piet Hut. "A hierarchical O(N log N) force-calculation algorithm". In: *Nature* 324.6096 (Dec. 1986). Publisher: Nature Publishing Group, pp. 446–449. ISSN: 1476-4687. DOI: 10.1038/324446a0.

[5] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[6] D. Boito et al. "On Maxwell's electrodynamics in two spatial dimensions". In: *Revista Brasileira de Ensino de Física* 42 (22nd June 2020). Publisher: Sociedade Brasileira de Física, e20190323. ISSN: 1806-1117, 1806-9126. DOI: 10.1590/1806-9126-RBEF-2019-0323.

[7] *The Barnes-Hut Algorithm : 15-418 Spring 2013*. URL: http://15418.courses.cs.cmu.edu/spring2013/article/18 (Accessed 5/5/2024).

[8] Leslie Greengard and Vladimir Rokhlin. "A new version of the Fast Multipole Method for the Laplace equation in three dimensions". In: *Acta Numerica* 6 (Jan. 1997), pp. 229–269. ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S0962492900002725.

[9] J. Carrier, L. Greengard and V. Rokhlin. "A Fast Adaptive Multipole Algorithm for Particle Simulations". In: *SIAM Journal on Scientific and Statistical Computing* 9.4 (July 1988), pp. 669–686. ISSN: 0196-5204, 2168-3417. DOI: 10.1137/0909044.

[10] Felipe A Cruz and L A Barba. "Characterization of the accuracy of the Fast Multipole Method in particle simulations". In: (2008).

[11] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[12] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[13] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

A Code Structure and Functionality

Main Files - Files which should be executed to produce all figures and results

- Barnes-Hut_p.ipynb - Runs a simulation using the Barnes-Hut algorithm to assess its correct functioning.
- Barnes-Hut_pm.ipynb - Runs a simulation using the Barnes-Hut algorithm to confirm its breakdown when using particles with both positive and negative property values.
- FMM.ipynb - Runs a simulation using the FMM algorithm to confirm its correct functioning.
- complexity_uniform.ipynb - Runs simulations to generate results for the complexity and precision achievable using the FMM and BH algorithms for a uniform distribution of particles all with positive particle properties.
- complexity_uniform_mixed_properties.ipynb - Runs simulations to generate results for the complexity and precision achievable using the FMM and BH algorithms for a uniform distribution of particles with a range of positive and negative particle properties.
- complexity_beta.ipynb - Runs simulations to generate results for the complexity and precision achievable using the FMM and BH algorithms for a beta distribution of particles all with positive particle properties.
- complexity_beta_mixed_properties.ipynb - Runs simulations to generate results for the complexity and precision achievable using the FMM and BH algorithms with a range of positive and negative particle properties.
- Barnes-Hut_p.py - A .py version of Barnes-Hut_p.ipynb
- Barnes-Hut_pm.py - A .py version of Barnes-Hut_pm.ipynb
- FMM.py - A .py version of FMM.ipynb
- complexity_uniform.py - A .py version of complexity_uniform.ipynb
- complexity_uniform_mixed_properties.py - A .py version of complexity_uniform_mixed_properties.ipynb
- complexity_beta.py - A .py version of complexity_beta.ipynb
- complexity_beta_mixed_properties.py - A .py version of complexity_beta_mixed_properties.ipynb

Module Files

- Analyic_Classes.py - Provides classes to simulate a single particle analytically

- BH_Classes.py - Provides classes which perform a simulation using the BH algorithm
- Direct_Classes.py - Provides classes to perform direct particle force and potential simulations
- FMM_Classes.py - Provides classes to perform FMM simulations
- Particle.py - Provides the particle class, storing information about each particle
- PointTesting.py - Provides classes to test FMM multipole expansions about specific points
- Utility.py - Provides utility plotting functions in 2D and 3D.