The Fast Multipole Method on Shared Memory Architectures

Contents

1 Introduction

Particle simulation methods in computational physics require evaluating the pairwise interactions between N particles, where N usually determines the resolution of the used method and is sought to be as large as possible. Particle interactions are frequently described by a potential field $\phi(\mathbf{x})$ generated by and depending on parameters such as position and charge or mass associated with each particle. The potential has the property of being additive, i.e. the total potential of a system of N particles is given by the superposition

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} \phi_i(\mathbf{x}) \tag{1.1}$$

of the potentials $\phi_i(\mathbf{x})$, $i = 1, \ldots, N$ of the individual particles. Problems of the type (1.1) are known as summation problems and their direct solution (i.e. a single evaluation of the potential at a given point) requires $\mathcal{O}(N)$ operations. A classical example is the N-body problem of celestial mechanics, where $\phi(\mathbf{x})$ is the gravitational potential (energy), which is connected to the acceleration of a particle of mass m_i described by a trajectory $\mathbf{x}_i(t)$ by Newton's second law,

$$m_i \frac{\mathrm{d}^2 \mathbf{x}_i}{\mathrm{d}t^2} = -\nabla \phi(\mathbf{x}_i)\,. \tag{1.2}$$

Time integration methods used to determine the trajectories $\mathbf{x}_i(t)$, $i = 1, \ldots, N$ of all particles given initial positions and velocities require the evaluation of the right hand side of (1.2) for a large number of time steps. Since in every time step, forces have to be computed for all particles, $\mathcal{O}(N^2)$ operations are needed per time step, making large simulations prohibitively expensive.

The fast multipole method (FMM), pioneered by L. Greengard and V. Rokhlin during the late nineties is an algorithm that leverages properties of the electrostatic and gravitational potential together with a hierarchical divide and conquer strategy to reduce the asymptotic cost required for N evaluations of (1.1) to $\mathcal{O}(N)$ operations and thereby makes simulations possible, which would be infeasible to conduct using direct methods. Moreover, the FMM is parametrized by the accuracy ε to which potentials and forces are to be evaluated, with the time complexity being more precisely given by $\mathcal{O}\Big(N \log^{(d-1)}(1/\varepsilon)\Big)$ in d dimensions [1].

The mathematical apparatus underlying the FMM differs between two and three dimensions, while the algorithmic aspects differ for uniform and non uniform particle distributions.[1] The FMM was first introduced for two dimensions and uniform particle distributions [2] and has been shown to present a considerable improvement over the direct algorithm for arbitrary (i.e. non uniform) two dimensional problems as well. In three dimensions, the changed geometry and the $\mathcal{O}\Big(N \log^2(1/\varepsilon)\Big)$ scaling lead to a steep increase in cost, which presents a challenge except where the required accuracy is low [1]. However, several schemes for acceleration have been developed to combat these problems [1, 3].

In order to be practical, an algorithm for the solution of large summation problems has to be parallelized. While the direct algorithm is embarrassingly parallel, the FMM is also suited for parallelization, and by now many parallel implementations exist.

In this work, we give an overview of related work (sec. 2), the theoretical basis underlying the FMM in two and three dimensions (sec. 3), summarize the algorithmic aspects of treating both uniform and non uniform

[1] The algorithm used for uniform particle distributions is termed the balanced FMM, while the algorithm used for non uniform particle distributions is referred to as the adaptive FMM.

particle distributions (sec. 4) and present our own implementation for both two and three as well as uniform and non uniform problems and its parallelization for shared memory architectures (sec. 5) and then proceed to evaluate accuracy, single core performance and parallel efficiency in different problem settings (sec. 6).

2 Related Work

The FMM was first described in [2], where the case of a uniform distribution of particles in two dimensions is treated. The three dimensional uniform case was introduced in [4, 5] while a description of the adaptive two dimensional version can be found in [6]. There are as of yet no full textbook level treatments of the FMM as described in the above sources that we are aware of, but introductory treatments can be found in [1] and in the final chapter of [7].

A brief overview over particle methods competing with the FMM is given in [8]. Notably, if the assumption of a uniform distribution of sources is made, there exist methods based on Ewald summation and the FFT which provide performant alternatives. If the assumption of uniformity is dropped (which is unavoidable e.g. in gravitational N-body simulations), the Barnes-Hut algorithm [9], which scales as $\mathcal{O}(N \log N)$ but does not offer the error bounds guaranteed by the FMM, provides a particularly viable alternative that is widely used in astrophysics and has been found to be faster than the FMM in some circumstances [10].

Reports on sequential implementations of the FMM are given e.g. in [2] (2D, balanced), [6] (2D, adaptive), [11] (3D, balanced) and [3] (3D, adaptive)[2]. The implementations described in [2, 6, 11] cover a similar scope as our implementation and serve as reference and base of comparison for our evaluation.

One of the first parallel implementations described was due to Greengard and Gropp [12], targeted at shared memory architectures. Several other implementations have become available over the recent years, of some which we give a brief and necessarily incomplete account.
General parallelization approaches for the FMM on various different architectures (notably also GPUs) are discussed in [13]. Cruz et al. implemented a library called `PetFMM` [14], based on `PETSc`, which uses dynamic load balancing together with a decomposition scheme based on cutting the tree data structure described in sec. 4 at a given height. Their implementation works in the distributed setting and achieves a parallel efficiency of 85% for 64 processors.
Coulaud et al. describe in [15] a hybrid MPI-Thread implementation based on space filling curves[3], which are frequently used for workload decomposition of tree algorithms such as the FMM or the Barnes-Hut algorithm.

[2] This paper discusses one of the schemes for accelerating the 3D FMM, i.e. not the "naive" version found in [5] and is hence only of indirect relevance to this report.
[3] We refer to [16] for a detailed description of this approach

3 Mathematical Preliminaries

In this section, the central theoretical results underlying the FMM are explored. We take here a similar approach as in [1] and consider first the two dimensional case. This has the advantage that the involved mathematical complexity is considerably smaller, such that hardly any proofs have to be omitted. Furthermore, the 2D case is a good setting for developing a geometric intuition about the FMM which then generalizes straightforwardly to three dimensions.

The potential problems of gravitational physics and electrostatics are governed by Poisson's equation

$$\Delta\phi = f\,, \tag{3.1}$$

where Δ denotes the Laplace operator, ϕ denotes the electrostatic or gravitational potential and f is proportional to the charge (electrostatics) or mass (gravitation) density ρ.[4] Away from any sources, $\rho \equiv 0$ and (3.1) simplifies to Laplace's equation

$$\Delta\phi = 0\,. \tag{3.2}$$

Functions which satisfy (3.2) are called harmonic. Harmonic functions have many interesting properties, one of which is that they admit series expansions in terms of powers of a single complex variable (in two dimensions) or spherical harmonics (in three dimensions). This property is essential to the FMM and will be developed in the next two subsections, based on the expositions in [2, 4, 1].

3.1 The FMM in two dimensions

In two dimensions and with the appropriate choice of units[5], Poisson's equation reads

$$\Delta\phi = \pm 2\pi\rho\,, \tag{3.3}$$

where the positive sign applies to the gravitational and the negative sign to the electrostatic case. Assume $\rho(\mathbf{x})$ to be nonnegative on $\mathbb{R}^2$, then, due to the linearity of the Laplace operator, every solution to (3.3) for the electrostatic potential is related to the corresponding solution for the gravitational potential by a change of sign, s.t. any procedure for solving (3.3) for the electrostatic potential can be used to solve for the gravitational potential without difficulty. For this reason, we consider in the following only the electrostatic case.

The electrostatic potential of a point source of charge q located at $\mathbf{x}_0 = (x_0, y_0) \in \mathbb{R}^2$ is given at a point $\mathbf{x} = (x, y) \in \mathbb{R}^2$ by

$$\phi_{\mathbf{x}_0}(\mathbf{x}) = -q \ln \|\mathbf{x} - \mathbf{x}_0\|\,, \tag{3.4}$$

while the electrostatic field is obtained by taking the negative gradient of the potential,

$$\mathbf{E}_{\mathbf{x}_0}(\mathbf{x}) = -\nabla\phi(\mathbf{x}) = q\frac{\mathbf{x} - \mathbf{x}_0}{\|\mathbf{x} - \mathbf{x}_0\|^2}\,. \tag{3.5}$$

Since the charge density of a point charge q located at $\mathbf{x}_0$ is given by $q\delta^{(2)}(\mathbf{x} - \mathbf{x}_0)$, where δ denotes the Dirac delta function, it follows that $\rho \equiv 0$ on $\mathbb{R} \setminus \{\, \mathbf{x}_0 \,\}$ implying in turn that $\phi_{\mathbf{x}_0}$ is harmonic on $\mathbb{R} \setminus \{\, \mathbf{x}_0 \,\}$.

[4] We will use the term "source" to refer to both charges and masses in their respective settings.

[5] In SI units, the vacuum permittivity ε_0 and the gravitational constant G appear in the formulae for the Coulomb and the gravitational potential. In the numerical methods literature, units are usually chosen such as to remove these constants, a convention which we follow as well.

For every harmonic function u of two real variables, there exists a holomorphic function $w : \mathbb{C} \to \mathbb{C}$ s.t. $u = \mathrm{Re}(w)$, which is unique up to an additive constant. This close connection between harmonicity and holomorphy as well as the relationship between holomorphy and series expansion motivates using the language of complex analysis for simplification. For this, the vector $\mathbf{x} = (x, y) \in \mathbb{R}^2$ is identified with the complex number $z = x + iy \in \mathbb{C}$, and the electrostatic potential $\phi_{\mathbf{x}_0}$, seen to be proportional to the real part of the complex logarithm, is identified with the complex potential

$$\phi_{z_0}(z) = q \ln(z - z_0)\,, \tag{3.6}$$

to which it is related by $\phi_{\mathbf{x}_0}(\mathbf{x}) = \mathrm{Re}(-\phi_{z_0}(z))$. Both the theory and the practical implementation of the two dimensional FMM are based on this formulation in terms of complex analysis, and we will in the following refer to (3.6) and derived expressions as the potential.

In practical applications, the electrostatic field needs to be computed alongside the potential frequently, which necessitates establishing a connection between (3.5) and (3.6). This connection is provided by the following lemma.

Lemma 1. *If $u(x, y) = \mathrm{Re}(w(x + iy))$ describes the potential field at (x, y), then the corresponding force field is given by*

$$-\nabla u = -(u_x, u_y) = (-\mathrm{Re}(w'),\ \mathrm{Im}(w'))\,, \tag{3.7}$$

where $w' = \frac{\mathrm{d}w}{\mathrm{d}z}$ denotes the complex derivative of w.

Proof. If $w(z) = w(x + iy) = u(x, y) + iv(x, y)$ is complex differentiable, then its real and imaginary parts u, v satisfy the Cauchy-Riemann equations $u_x = v_y$ and $u_y = -v_x$. Since $w' = w_x = u_x + iv_x$, (3.7) follows. □

Lemma 2. *Let a point source of charge q be located at z_0, then for any z with $|z| > |z_0|$,*

$$\phi_{z_0}(z) = q \ln(z - z_0) = q\left(\ln(z) - \sum_{k=1}^{\infty} \frac{1}{k}\left(\frac{z_0}{z}\right)^k\right). \tag{3.8}$$

Proof. Note that $\ln(z - z_0) = \ln(z(1 - \frac{z_0}{z})) = \ln(z) + \ln(1 - \frac{z_0}{z})$. Since $|\frac{z_0}{z}| < 1$ per assumption, $\ln(1 - \frac{z_0}{z})$ can be expanded into a (convergent) Taylor series around $z_0 = 0$, which yields (3.8). □

Equation (3.8) represents an expansion of the potential ϕ_{z_0} around $z_0 = 0$, hence the Taylor polynomial obtained from truncating the series to p-th order will approximate ϕ_{z_0} well, as long as the source lies close to the origin - close in this context meaning much closer than the evaluation point z. For fixed z_0, (3.8) is a series in inverse powers of z, a consequence of which is that the relative accuracy obtained from the p-th order approximation increases with the distance between source and evaluation point. By itself, the expansion (3.8) of the potential due to a single point source presents no computational advantage over direct evaluation of the potential (3.4). However, a central advantage of the series representation is that it is easily summed, allowing for an accurate representation of the potential due to multiple charges in a compressed form. This is formalized in the following lemma.

Theorem 1 (Multipole expansion)**.** *Suppose that n charges of strengths q_i, $i = 1, \ldots, n$ are located at points z_i with $|z_i| < r$ for $\mathbb{R} \ni r > 0$, then for any $z \in \mathbb{C}$ with $|z| > r$, the potential $\phi(z)$ is given by*

$$\phi(z) = Q \ln(z) + \sum_{k=1}^{\infty} \frac{a_k}{z^k}\,,$$

where

$$Q = \sum_{i=1}^{n} q_i\,, \qquad a_k = -\frac{1}{k} \sum_{i=1}^{n} q_i z_i^k\,.$$

Furthermore, for any $p \geq 1$,

$$\left| \phi(z) - Q \ln(z) - \sum_{k=1}^{p} \frac{a_k}{z^k} \right| \leq \frac{A}{c-1} \left(\frac{1}{c} \right)^p\,, \tag{3.9}$$

where

$$c = \left| \frac{z}{r} \right| \qquad \textit{and} \qquad A = \sum_{i=1}^{n} |q_i|\,. \tag{3.10}$$

Proof. Equation (1) follows from the potential being additive and summation of (3.8). The error bound is obtained by inspection of the remainder:

$$\left| \phi(z) - Q \ln(z) - \sum_{k=1}^{p} \frac{a_k}{z^k} \right| = \left| \sum_{k=p+1}^{\infty} \frac{a_k}{z^k} \right| \leq \sum_{k=p+1}^{\infty} \frac{|a_k|}{|z|^k} \leq \sum_{k=p+1}^{\infty} \frac{A r^k}{k |z|^k} \leq \frac{A}{c^{p+1}} \sum_{k=0}^{\infty} \left(\frac{1}{c} \right)^k = \frac{A}{c-1} \left(\frac{1}{c} \right)^p.$$

□

The computational advantage of using the multipole expansion (1) for evaluating sums of the form (1.1) is illustrated by the following example. If m evaluation points fall outside of the disk enclosing all sources, instead of evaluating (1.1) m times at a total cost of $\mathcal{O}(nm)$, a series expansion can be formed by (1) and evaluated instead at a total cost of $\mathcal{O}(np + mp)$. It is possible to achieve a comparable asymptotic cost without requiring the source and evaluation points to be separated. For this, several further results are needed.

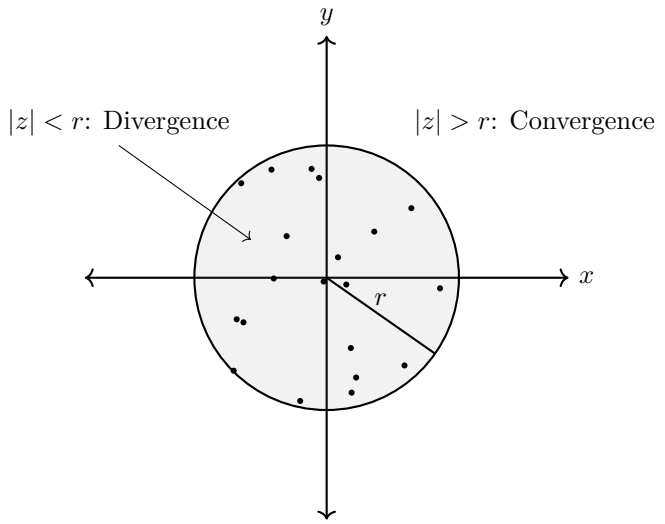

Figure 1a: Charges located in a disk of radius r around the origin in the complex plane. Outside of the disk, the multipole expansion (1) converges.

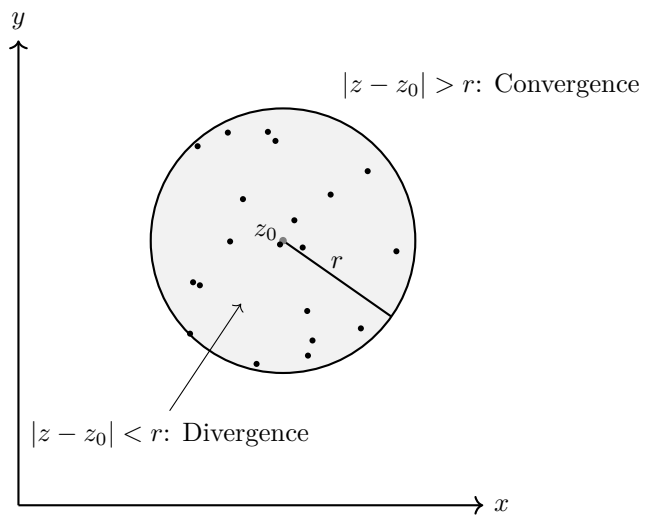

Figure 1b: Charges located in a disk of radius r around a point z_0 in the complex plane, regions of convergence and divergence of the multipole expansion around z_0.

Theorem 1 describes the series representation of the potential due to a set of charges located within the disk $D_r(0)$ of radius r centered at the origin (fig. 1a) and gives a rigorous bound on the absolute error made when this series is truncated to p-th order. In general however, there is no preferred origin, and the multipole expansion is computed w.r.t. the center of the smallest disk enclosing a given set of charges (fig. 1b). Given a multipole expansion of the form (1) w.r.t. an origin O, the corresponding expansion around a different origin $\overline{O}$ is found by the following argument.

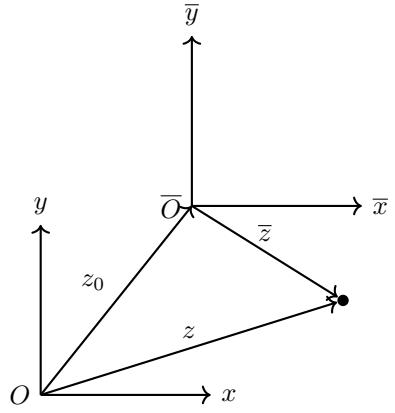

Figure 2: On the relation between the coordinates z, $\overline{z}$ of a point w.r.t. different origins.

Let z be the coordinates of an arbitrary point and z_0 the coordinates of $\overline{O}$, both w.r.t. O, as shown in fig. 2. It then follows that $z = \overline{z} + z_0 = \overline{z} - \overline{z}_0$, where $\overline{z}_0 = -z_0$. The potential w.r.t. $\overline{O}$ is given by $\overline{\phi}(\overline{z})$ and must satisfy $\overline{\phi}(\overline{z}) = \phi(z)$, since the potential at a point is independent of its representation in terms of coordinates. But then, we obtain

$$\overline{\phi}(\overline{z}) = \phi(z) = \phi(\overline{z} - \overline{z}_0) = Q \ln(\overline{z} - \overline{z}_0) + \sum_{k=1}^{\infty} \frac{a_k}{(\overline{z} - \overline{z}_0)^k}\,, \qquad a_k = -\sum_{i=1}^{n} \frac{q_i(\overline{z}_i - \overline{z}_0)^k}{k}\,.$$

Similarly, a multipole expansion centered at $\overline{O}$ is, w.r.t. O given by

$$\phi(z) = Q \ln(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z - z_0)^k}\,, \qquad a_k = -\sum_{i=1}^{n} \frac{q_i(z_i - z_0)^k}{k}\,. \tag{3.11}$$

We have seen previously that the ability to sum the series expansions of potentials due to different sources can be utilized to reduce the computational cost of evaluating the electrostatic potential. This ability is also central to the FMM. However, multipole expansions of the form (3.11) w.r.t. different origins can *not* be summed directly. The following theorem alleviates this problem by providing a formula for shifting the center of multipole expansion, i.e. determining the coefficients of an expansion of the form (3.8) given the coefficients of an expansion of the form (3.11).

Theorem 2 (Translation of a multipole expansion)**.** *Let*

$$\phi(z) = a_0 \ln(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z - z_0)^k} \tag{3.12}$$

be a multipole expansion of the potential due to a set of n charges of strengths q_i, $i = 1, \ldots, n$, located inside the disk $D_R(z_0)$ of radius R centered at z_0, then for z outside of the disk $D_{R+|z_0|}(0)$ of radius $R + |z_0|$ and center at the origin,

$$\phi(z) = a_0 \ln(z) + \sum_{l=1}^{\infty} \frac{b_l}{z^l}\,, \tag{3.13}$$

where

$$b_l = -\frac{a_0 z_0^l}{l} + \sum_{k=1}^{l} a_k z_0^{l-k} \binom{l-1}{k-1}\,. \tag{3.14}$$

Furthermore, for any $p \geq 1$,

$$\left| \phi(z) - a_0 \ln(z) - \sum_{l=1}^{p} \frac{b_l}{z^l} \right| \leq \frac{A}{1 - \left|\frac{|z_0| + R}{z}\right|} \left(\frac{|z_0| + R}{z} \right)^{p+1}\,, \tag{3.15}$$

with A defined in (3.10).

Proof. In (3.12), both $\ln(z - z_0)$ and $(z - z_0)^{-k}$ can be expanded into Taylor series around $z_0 = 0$,

$$\ln(z - z_0) = \ln(z) - \sum_{l=1}^{\infty} \frac{1}{l} \left(\frac{z_0}{z} \right)^l , \qquad (z - z_0)^{-k} = \sum_{l=k}^{\infty} \binom{l-1}{k-1} \frac{z_0^{l-k}}{z^l} , \tag{3.16}$$

which converge as long as $|z_0| < |z|$. Substituting these expressions into (3.12), we obtain the coefficients (3.14) by shifting summation indices, interchanging the order of summation in the double sum and factoring out the coefficients of z^{-l}. The error bound (3.15) is a direct consequence of (3.9) and the uniqueness of the Taylor series. □

The geometry of the situation described in the preceding theorem is depicted in fig. 3. Notably, by (3.14), computation of the coefficients b_l, $l = 1, \ldots, p$ requires only knowledge of a_k, $k = 1, \ldots, p$ and no further approximations beyond truncation are made, hence the given formula for the translation of a multipole expansion is exact.

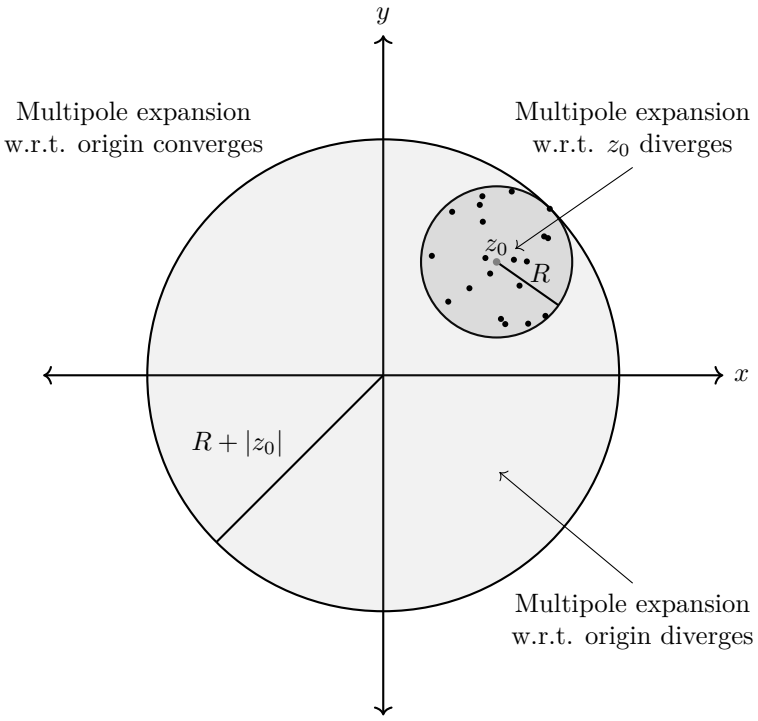

Figure 3: Charges located in a disk of radius R around a point z_0 in the complex plane. The multipole expansion around z_0 converges outside of $D_R(z_0)$, while the corresponding shifted expansion (w.r.t. the origin) converges outside of $D_{R+|z_0|}(0)$.

The theorems 1 and 2 allow forming multipole expansions of the potential due to a set of charges, and to shift such expansions to a different origin. The second lemma is very useful: it allows computing, given several multipole expansions of the potential due to different sets of charges and w.r.t. different origins, a single multipole expansion, convergent outside of the smallest disk containing the union of all the charges, by shifting all expansions to a common origin and summing the coefficients.

Theorem 1 by itself allows for the construction of an $\mathcal{O}(N \log N)$ algorithm for the evaluation of the potential due to N sources, while theorem 2 allows for improvements in the constant factor of the asymptotic runtime of this algorithm (cf. sec. 4). In order to reduce this even further, a new type of representation of the potential is needed, and introduced in the following lemma.

Lemma 3. *Let a point source of charge q be located at z_0, then for any z with $|z| < |z_0|$,*

$$\phi_{z_0}(z) = q \ln(z - z_0) = q\left(\ln(-z_0) - \sum_{l=1}^{\infty} \frac{1}{l}\left(\frac{z}{z_0}\right)^l\right). \tag{3.17}$$

Proof. In direct analogy to lemma 2, we have $\ln(z - z_0) = \ln\left(-z_0(1 - \frac{z}{z_0})\right) = \ln(-z_0) + \ln\left(1 - \frac{z}{z_0}\right)$. Taylor expansion around $z = 0$ leads to the series (3.17), which converges since $|z| < |z_0|$ per assumption. □

Theorem 3 (Local expansion)**.** *Suppose that n charges of strengths q_i, $i = 1, \ldots, n$ are located at points z_i with $|z_i| > r$ for some positive $r \in \mathbb{R}$, then for any $z \in \mathbb{C}$ with $|z| < r$, the potential $\phi(z)$ is given by*

$$\phi(z) = \sum_{k=0}^{\infty} b_l z^l\,, \tag{3.18}$$

where

$$b_0 = \sum_{i=1}^{n} q_i \ln(-z_i)\,, \qquad b_l = -\frac{1}{l}\sum_{i=1}^{n} \frac{q_i}{z_i^l} \quad \textit{for } l \geq 0\,. \tag{3.19}$$

Furthermore, for any $p \geq 1$,

$$\left|\phi(z) - \sum_{k=0}^{p} b_l z^l\right| \leq \frac{A}{c-1}\left(\frac{1}{c}\right)^p, \quad \textit{where} \qquad c = \left|\frac{r}{z}\right| \tag{3.20}$$

and A as defined in (3.10)*.*

Proof. In complete analogy to the proof of theorem 1. □

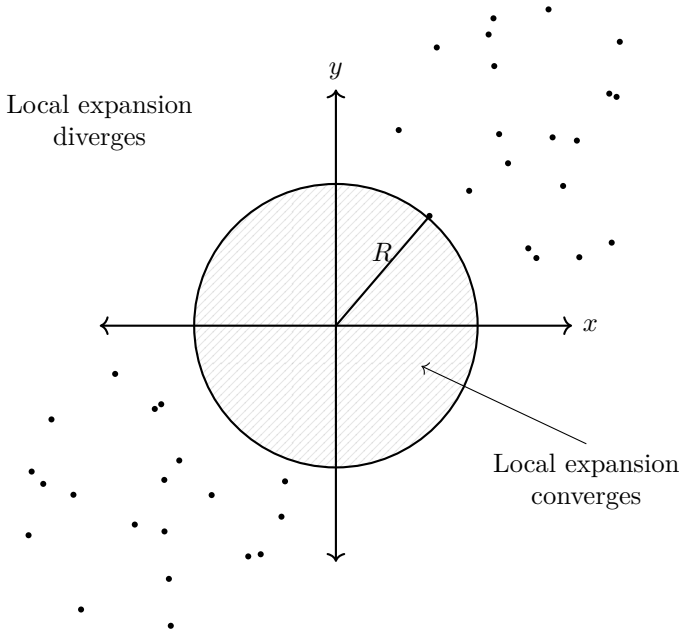

Figure 4: Regions of convergence and divergence of a local expansion of the potential due to a set of charges.

Note the symmetry between the theorems 1 and 3: While (3.8) represents an expansion of $\phi_{z_0}(z)$ around $z_0 = 0$, convergent as long as the evaluation point z falls outside of $D_{|z_0|}(0)$, (3.17) represents an expansion of the same potential around $z = 0$, convergent as long as z lies inside of $D_{|z_0|}(0)$. Correspondingly, the multipole expansion (1) of the potential due to a set of charges converges outside of the smallest disk centered at the origin and containing all the charges, while the local expansion (3.18) converges inside of the largest disk centered at the origin and containing none of the charges.

The asymptotic efficiency of the FMM derives from its ability to determine, for any evaluation point z of interest, a local expansion accounting for the potentials of all but a small, fixed number of sources close to z. Theorem 4 introduces a central tool needed for this task: the ability to convert a multipole into a local expansion.

Theorem 4 (Conversion of a multipole expansion into a local expansion)**.** *Suppose that n sources of strengths q_i, $i = 1, \ldots, n$ are located inside the disk $D_R(z_0)$ and that $|z_0| > (c+1)R$ where $c > 1$. The multipole expansion* (3.11) *of the potential due to the sources converges inside of $D_R(0)$ and is there described by a local expansion*

$$\phi(z) = \sum_{l=0}^{\infty} b_l z^l \,, \tag{3.21}$$

where

$$b_0 = a_0 \ln(-z_0) + \sum_{k=1}^{\infty} \frac{a_k}{z_0^k}(-1)^k \qquad \textit{and} \qquad b_l = -\frac{a_0}{l \cdot z_0^l} + \frac{1}{z_0^l} \sum_{k=1}^{\infty} \frac{a_k}{z_0^k} \binom{l+k-1}{k-1} (-1)^k \quad \textit{for } l \geq 1. \tag{3.22}$$

Furthermore, for any $p \geq \max(2, \frac{2c}{c-1})$,

$$\left| \phi(z) - \sum_{l=0}^{p} b_l z^l \right| < \frac{A(4e(p+c)(c+1) + c^2)}{c(c-1)} \left(\frac{1}{c}\right)^{p+1} , \tag{3.23}$$

with A as defined in (3.10) *and e denotes Euler's number.*

Proof. The coefficients are obtained similarly to theorem 2 by Taylor series expansion of $\ln(z - z_0)$ and $(z - z_0)^k$ around $z = 0$. For a proof of the error bound, we refer to [2, 4]. □

The geometry of the situation described in the preceding theorem is shown in fig. 5.

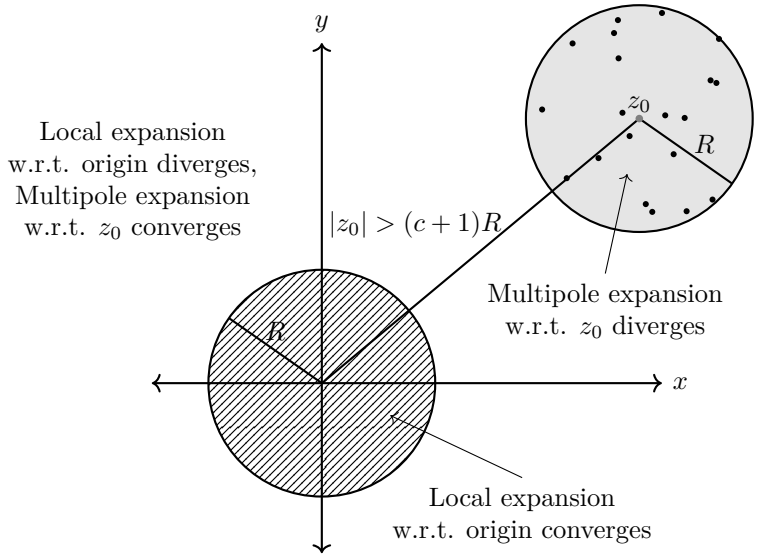

Figure 5: Regions of convergence and divergence of a multipole expansion and the corresponding shifted local expansion.

As in the case of the multipole expansion, a local expansion may be computed w.r.t. an arbitrary origin. By a similar argument as in the discussion preceding eq. (3.11), the local expansion around z_0 of the potential due to a set of sources outside of $D_R(z_0)$ is given by

$$\phi(z) = \sum_{k=0}^{\infty} b_k (z - z_0)^k \,, \tag{3.24}$$

where

$$b_0 = \sum_{i=1}^{n} q_i \ln(-(z_i - z_0)) \,, \qquad b_k = -\frac{1}{k} \sum_{i=1}^{n} \frac{q_i}{(z_i - z_0)^k} \quad \text{for } k \geq 0 \,. \tag{3.25}$$

Similar to the case with the multipole expansion, the FMM relies on the ability to combine local expansions that are w.r.t. different points. The following lemma (see also fig. 6) provides precisely this.

Theorem 5 (Translation of a local expansion)**.** *Let the local expansion of the potential due to a set of n charges of strengths q_i, $i = 1, \ldots, n$, located outside of the disk $D_R(z_0)$ be given by* (3.24)*. Then for $R > |z_0|$ and $z \in D_{R-|z_0|}(0)$,*

$$\phi(z) = \sum_{l=0}^{\infty} a_l z^l \,, \qquad \textit{where} \qquad a_l = \sum_{k=l}^{\infty} b_k \binom{k}{l} (-z_0)^{k-l} \,. \tag{3.26}$$

Furthermore, for any $p \geq 1$,

$$\left| \phi(z) - \sum_{l=0}^{p} a_l z^l \right| \leq \frac{A}{\frac{R-|z_0|}{|z|} - 1} \left(\frac{1}{\frac{R-|z_0|}{|z|}} \right)^{p+1} \,, \tag{3.27}$$

with A defined in (3.10)*.*

Proof. The coefficients (3.26) are obtained by application of the binomial theorem in (3.24),

$$\sum_{l=0}^{\infty} b_k (z - z_0)^k = \sum_{k=0}^{\infty} b_k \sum_{l=0}^{k} \binom{k}{l} z^l (-z_0)^{k-l} = \sum_{k=l}^{\infty} b_k \sum_{l=0}^{\infty} \binom{k}{l} z^l (-z_0)^{k-l} = \sum_{l=0}^{\infty} z^l \sum_{k=l}^{\infty} b_k \binom{k}{l} (-z_0)^{k-l} . \tag{3.28}$$

□

The translation formula (3.26) is exact. When translating a series truncated to p-th order, the summation for a_l contains at most p terms and the shifted local expansion enjoys the same error bound as the original expansion. An efficient algorithmic realization of (3.26) is possible via the Horner scheme [1].

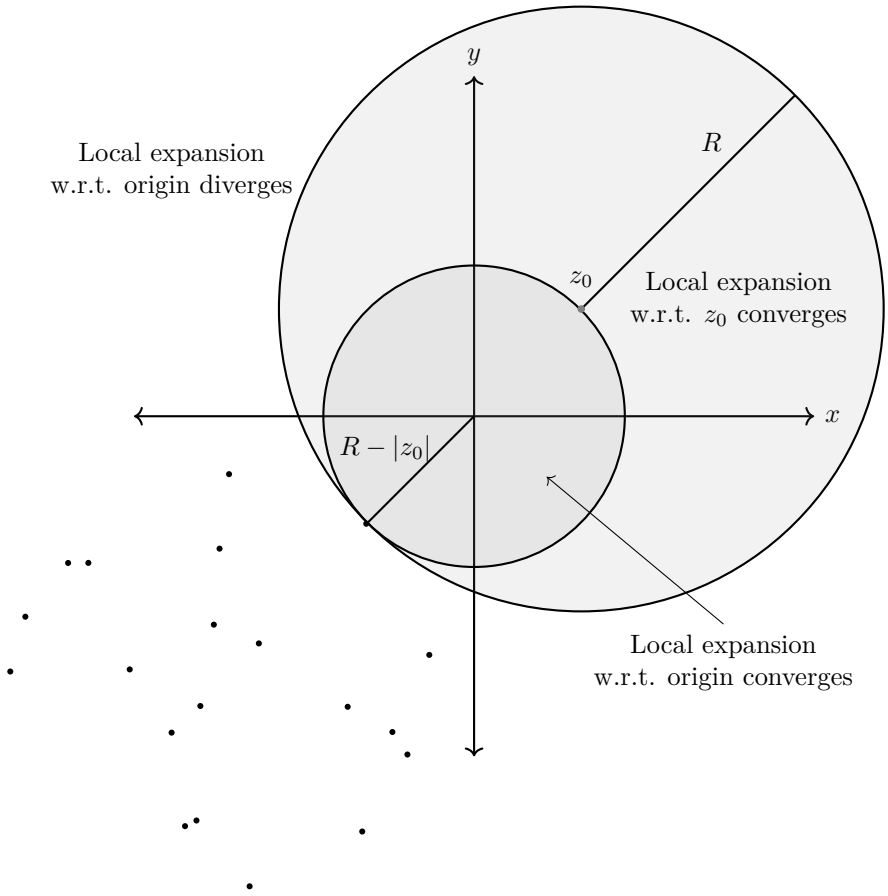

Figure 6: Charges located outside of a disk of radius R around a point z_0 in the complex plane. The local expansion around z_0 converges inside of $D_R(z_0)$, while the corresponding shifted expansion (w.r.t. the origin) converges inside of $D_{R-|z_0|}(0)$.

3.2 The FMM in three dimensions

In three dimensions, Poisson's equation for the electrostatic potential reads

$$\Delta\phi = -4\pi\rho . \tag{3.29}$$

The electrostatic potential of a point source of charge q located at $\mathbf{x}_0 = (x_0, y_0, z_0) \in \mathbb{R}^3$ is given at a point $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ by[6]

$$\phi_{\mathbf{x}_0}(\mathbf{x}) = \frac{q}{\widetilde{r}}\,, \tag{3.30}$$

where $\widetilde{r} = \|\mathbf{x} - \mathbf{x}_0\|$, while the electrostatic field is given by

$$\mathbf{E}_{\mathbf{x}_0}(\mathbf{x}) = -\nabla\phi(\mathbf{x}) = \frac{q}{\widetilde{r}^3}(\mathbf{x} - \mathbf{x}_0)\,. \tag{3.31}$$

As in two dimensions, the electrostatic potential is harmonic away from sources. Several results are required to derive series expansions of the potential (3.30) with properties analogous to the two-dimensional case, and are stated here mostly without proof.

The three-dimensional FMM makes heavy use of spherical coordinates, which are central for simplifying problems with radial symmetry. Let the spherical coordinates of $\mathbf{x}$ be given by (r, θ, φ) and similarly those of $\mathbf{x}_0$ by (ρ, α, β), and let γ denote the angle subtended between them.[7] Then $\widetilde{r}^{-1}$ can be expanded as follows (for a proof, see [17])

$$\frac{1}{\widetilde{r}} = \frac{1}{\sqrt{r^2 + \rho^2 - 2r\rho\cos\gamma}} = \sum_{n=0}^{\infty} \frac{r_<^n}{r_>^{n+1}} P_n(\cos\gamma)\,, \tag{3.32}$$

where $r_< = \min(r, \rho)$, $r_> = \max(r, \rho)$ and P_n denotes the Legendre Polynomial of order n. While (3.32) does represent a series expansion of the potential in powers of distances, $P_n(\cos\gamma)$ depends on both source and evaluation point, such that a series representation based exclusively on (3.32) would require recomputing coefficients at every evaluation point, which is completely infeasible. The following theorem, the proof of which can be found in [17], provides a solution to this problem.[8]

Theorem 6 (Addition Theorem for Spherical Harmonics)**.** *Let* $\mathbf{x}$ *and* $\mathbf{x}_0$ *be coordinate vectors with spherical coordinates* (r, θ, φ) *and* (ρ, α, β) *which subtend an angle* γ *between them, then it holds that*

$$P_n(\cos\gamma) = \sum_{m=-n}^{n} Y_n^{-m}(\alpha, \beta) \cdot Y_n^m(\theta, \varphi)\,, \tag{3.33}$$

where the terms Y_n^m *are known as spherical harmonics and defined by*

$$Y_n^m = \sqrt{\frac{(n - |m|!)}{(n + |m|)!}} \cdot P_n^{|m|}(\cos\theta) e^{im\varphi}\,, \tag{3.34}$$

and the functions P_n^m *are the associated Legendre functions.*

The addition theorem for spherical harmonics, together with (3.32) leads to the following theorem:

Theorem 7 (Multipole expansion)**.** *Consider* k *charges of strengths* q_i *located at points* $\mathbf{x}_i = (\rho_i, \alpha_i, \beta_i)$, $i = 1, \ldots, k$ *with* $\rho_i < a$*, then for any* $\mathbf{x} = (r, \theta, \varphi)$ *with* $r > a$*, the potential* $\phi(\mathbf{x})$ *is given by*

$$\phi(\mathbf{x}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} \cdot Y_n^m(\theta, \varphi)\,, \tag{3.35}$$

[6] Similarly, the gravitational potential of a point mass is given by $\phi_{\mathbf{x}_0}(\mathbf{x}) = -\frac{m}{\widetilde{r}}$.

[7] We will from now on often write statements such as $\mathbf{x} = (r, \theta, \varphi)$ to denote the components of the vector $\mathbf{x}$ in spherical coordinates.

[8] The convention chosen for the definition of the spherical harmonics in the FMM literature differs from that usually used in physics, e.g. in [17], in normalization and behaviour under the interchange $m \leftrightarrow -m$, which has to be taken into account during implementation.

where

$$M_n^m = \sum_{i=1}^{k} q_i \cdot \rho_i^n \cdot Y_n^{-m}(\alpha_i, \beta_i)\,. \tag{3.36}$$

Furthermore, for any $p \geq 1$,

$$\left| \phi(\mathbf{x}) - \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} \cdot Y_n^m(\theta, \varphi) \right| \leq \frac{A}{r-a} \left(\frac{a}{r} \right)^{p+1}\,, \tag{3.37}$$

where

$$A = \sum_{i=1}^{k} |q_i|\,. \tag{3.38}$$

For the proof of the error bound, we refer to [4]. As in the two-dimensional case, a multipole expansion w.r.t. a given origin $\mathbf{x}_0$ may be translated into a multipole expansion w.r.t. the origin.

Theorem 8 (Translation of a multipole expansion)**.** *Suppose that* l *charges of strengths* q_i, $i = 1, \ldots, l$ *are located inside the ball* $B_a(\mathbf{x}_0)$ *of radius* a *centered at* $\mathbf{x}_0 = (\rho, \alpha, \beta)$*, and that for* $\mathbf{x} = (r, \theta, \varphi)$ *outside of* $B_a(\mathbf{x}_0)$*, the potential due to these charges is given by the multipole expansion*

$$\phi(\mathbf{x}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{O_n^m}{r'^{n+1}} \cdot Y_n^m(\theta', \varphi')\,, \tag{3.39}$$

where $\mathbf{x} - \mathbf{x}_0 = (r', \theta', \varphi')$*, then for any point* $\mathbf{x}$ *outside of the ball* $B_{a+\rho}(\mathbf{0})$ *of radius* $a + \rho$ *and center at the origin,*

$$\phi(\mathbf{x}) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{r^{j+1}} \cdot Y_j^k(\theta, \varphi)\,, \tag{3.40}$$

where

$$M_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} \frac{O_{j-n}^{k-m} \cdot i^{|k|-|m|-|k-m|} \cdot A_n^m \cdot A_{j-n}^{k-m} \cdot \rho^n \cdot Y_n^{-m}(\alpha, \beta)}{A_j^k}\,, \tag{3.41}$$

with A_n^m *defined by*

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}\,. \tag{3.42}$$

Furthermore, for any $p \geq 1$,

$$\left| \phi(\mathbf{x}) - \sum_{j=0}^{p} \sum_{k=-j}^{j} \frac{M_j^k}{r^{j+1}} \cdot Y_j^k(\theta, \varphi) \right| \leq \frac{A}{r-(a+\rho)} \left(\frac{a+\rho}{r} \right)^{p+1}\,, \tag{3.43}$$

As a second consequence of theorem 6, we obtain the following theorem.

Theorem 9 (Local expansion)**.** *Consider* k *charges of strengths* q_i *located at points* $\mathbf{x}_i = (\rho_i, \alpha_i, \beta_i)$, $i = 1, \ldots, k$ *with* $\rho_i > a$*, then for any* $\mathbf{x} = (r, \theta, \varphi)$ *with* $r < a$*, the potential* $\phi(\mathbf{x})$ *is given by*

$$\phi(\mathbf{x}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} M_n^m \cdot Y_n^m(\theta, \varphi) \cdot r^n\,, \tag{3.44}$$

where

$$M_n^m = \sum_{i=1}^{k} \frac{q_i}{\rho_i^{n+1}} \cdot Y_n^{-m}(\alpha_i, \beta_i)\,. \tag{3.45}$$

The following theorem establishes the possibility of conversion of a multipole into a local expansion under suitable conditions.

Theorem 10 (Conversion of a multipole expansion into a local expansion)**.** *Suppose that l charges of strengths q_i are located inside the ball $B_a(\mathbf{x}_0)$ of radius a centered at $\mathbf{x}_0 = (\rho, \alpha, \beta)$, and that $\rho > (c+1)a$ with $c > 1$. Then the multipole expansion* (3.39) *around $\mathbf{x}_0$ converges inside the ball $B_a(\mathbf{0})$ of radius a centered at the origin. Inside $B_a(\mathbf{0})$, the potential due to the charges is described by a local expansion:*

$$\phi(\mathbf{x}) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta, \varphi) \cdot r^j \,, \tag{3.46}$$

where

$$L_j^k = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{O_n^m \cdot i^{|k-m|-|k|-|m|} \cdot A_n^m \cdot A_j^k \cdot Y_{j+n}^{m-k}(\alpha, \beta)}{(-1)^n A_{j+n}^{m-k} \cdot \rho^{j+n+1}} \,, \tag{3.47}$$

with A_r^s defined by (3.42)*. Furthermore, for any $p \geq 1$,*

$$\left| \phi(\mathbf{x}) - \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta, \varphi) \cdot r^j \right| \leq \frac{A}{a(c-1)} \left(\frac{1}{c} \right)^{p+1} \,, \tag{3.48}$$

with A as defined in (3.38)*.*

As with the multipole expansion, a local expansion w.r.t. a specific origin may be translated to a local expansion w.r.t. a different origin (within the former's sphere of convergence). The corresponding translation operator is provided by the following theorem.

Theorem 11 (Translation of a local expansion)**.** *Let $\mathbf{x}_0 = (\rho, \alpha, \beta)$ be the origin of a local expansion*

$$\phi(\mathbf{x}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} O_n^m \cdot Y_n^m(\theta', \varphi') \cdot r'^n \,, \tag{3.49}$$

where $\mathbf{x} = (r, \theta, \varphi)$ and $\mathbf{x} - \mathbf{x}_0 = (r', \theta', \varphi')$. Then

$$\phi(\mathbf{x}) = \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k \cdot Y_j^k(\theta, \varphi) \cdot r^j \,, \tag{3.50}$$

where

$$L_j^k = \sum_{n=j}^{p} \sum_{n=-n}^{n} \frac{O_n^m \cdot i^{|m|-|m-k|-|k|} \cdot A_{n-j}^{m-k} \cdot A_j^k \cdot Y_{n-j}^{m-k}(\alpha, \beta) \cdot \rho^{n-j}}{(-1)^{n+j} \cdot A_n^m} \,, \tag{3.51}$$

with A_r^s defined by (3.42)*.*

In three, as in two dimensions, it is possible to obtain an approximation to the force by taking the gradient of the series expansion of the potential. Consider a scalar function $\psi(r, \theta, \varphi)$, then $\nabla\psi$ is, in spherical coordinates, given by [17]

$$\nabla\psi = \frac{\partial\psi}{\partial r}\hat{\boldsymbol{r}} + \frac{1}{r}\frac{\partial\psi}{\partial\theta}\hat{\boldsymbol{\theta}} + \frac{1}{r\sin\theta}\frac{\partial\psi}{\partial\varphi}\hat{\boldsymbol{\varphi}} \,, \tag{3.52}$$

where $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\varphi}})$ denotes the spherical coordinate basis, i.e.

$$\begin{aligned}
\hat{\boldsymbol{r}} &= \sin\theta\cos\varphi\hat{\boldsymbol{x}} + \sin\theta\sin\varphi\hat{\boldsymbol{y}} + \cos\theta\hat{\boldsymbol{z}}\,, \\
\hat{\boldsymbol{\theta}} &= \cos\theta\cos\varphi\hat{\boldsymbol{x}} + \cos\theta\sin\varphi\hat{\boldsymbol{y}} + \cos\theta\hat{\boldsymbol{z}}\,, \\
\hat{\boldsymbol{\varphi}} &= -\sin\varphi\hat{\boldsymbol{x}} + \cos\varphi\hat{\boldsymbol{y}}
\end{aligned} \tag{3.53}$$

and $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ are the Cartesian unit vectors. From the forms (3.35) and (3.44) of the multipole and local expansions and the definition (3.34) of the spherical harmonics, it can be seen that the only partial derivative in (3.52) whose evaluation poses a challenge is that w.r.t. the polar angle θ, i.e. $\frac{\partial}{\partial\theta}P_n^m(\cos\theta)$.

It can be shown that the associated Legendre polynomials satisfy several recurrence relations [18], which allow expressing $\frac{\partial}{\partial x}P_n^m(x)$ e.g. through $P_n^m(x)$ and $P_n^{m-1}(x)$, which yields truly "analytical" derivatives and eliminates the need for numerical differentiation.

4 Algorithm Description

The central idea underlying the FMM is that the electrostatic and gravitational potentials are smooth far away from sources and should hence be representable in a compressed form, as long as compromises in accuracy are made.[9] The multipole expansion of the potential due to a set of sources provides precisely such a representation, whose accuracy depends on the ratio between the radius r of the disk or ball containing the sources and the distance $|z|$ of the evaluation point from the center of that disks (cf. eq. (3.9)). In particular, to obtain a fixed accuracy, particles far from the evaluation region may be grouped into larger disks and hence processed with less computational effort. The FMM is essentially a scheme for leveraging this fact in order to extract the desired features of the potential with the least computational effort possible by grouping sources and forming multipole expansions at various scales of spatial resolution, and in that shares characteristics with other multi-resolution algorithms, such as the multi-grid method.

In the following two subsections, the data structure and algorithm tying the previously developed theory together to form the fast multipole method are discussed. The next section introduces the balanced FMM, originally introduced in [2], which relies on the assumption of a uniform spatial distribution of sources. This assumption of uniformity is a strong one and the balanced FMM faces several competitors (cf. sec. 2) in this context, but, it is conceptually simpler than the adaptive version described thereafter, and well suited for an introduction. Once the required mathematical machinery is in place, the algorithmic details are in large parts agnostic of the dimension of the problem, and so the description in the following sections applies equally to the two and the three dimensional FMM. We will for this reason sometimes be a bit informal in this description, e.g. by talking about disks instead of disks and balls and referencing primarily the theorems from sec. 3.1, but every component of the two dimensional FMM has a direct counterpart in two dimensions, hence this is admissible.

4.1 The Balanced Algorithm

This section follows the presentation in [1] and begins by introducing an $\mathcal{O}(N\log N)$ algorithm for solving the summation problem (1.1) which relies only on theorem 1 and serves as a useful stepping stone for the full FMM.

[9] And naturally, such compromises are intrinsic to floating point arithmetic. Other methods, such as the Barnes-Hut algorithm are based on the same idea.

4.1.1 The $\mathcal{O}(N \log N)$ algorithm

In order to realize the idea of accounting for groups of particles by multipole expansions (and collecting more distant particles into larger groups in order to save computational effort), the computational domain, i.e. the smallest box containing all sources[10], is partitioned into a regular grid of boxes with multiple refinement levels. More specifically, refinement level 0 consists of the entire computational domain, and refinement level $l+1$ is obtained by subdivision of each box b at level l into 2^d equal parts, considered b's children. An example in two dimensions is shown in fig. 7. In terms of data structures, this decomposition is realized by a perfect k-ary tree[11], where $k = 2^d$, i.e. a quadtree in two and an octree in three dimensions.[12]

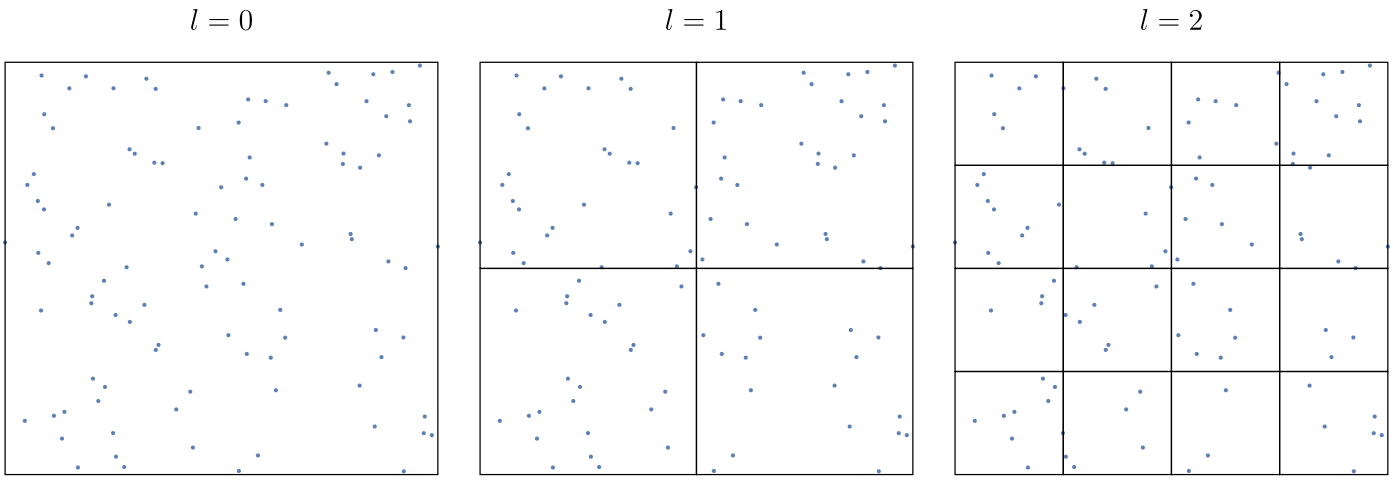

Figure 7: The computational domain and two levels of refinement

The boxes at the various refinement levels represent groups of particles whose potential will represented by multipole expansions, and we will frequently refer to the multipole expansion of the potential due to all particles in a box b as b's multipole expansion. The tree data structure is used to keep track of which particles fall into which box, and, crucially, of the proximity relations between boxes that enable the multipole approximation to be made. To proceed, several definitions [1] are required:

Definition 1. *Two boxes are said to be* ***near neighbours****, if they are at the same refinement level and share a boundary point, and a box is a near neighbour of itself. With each box b is associated a* ***near neighbour list****, containing all near neighbours of b.*

Let the boxes b and b' with centers z_0, z_0' and lengths L, L' be near neighbours, then the multipole expansion of the potential due to all the charges in b converges outside of $D_{\sqrt{d}L/2}(z_0)$. As this disk with intersects b', the multipole approximation is *not* valid for near neighbours.

[10] If the potential is to be evaluated at arbitrary points, then the computational domain encompasses all source and evaluation points.

[11] By this we understand a tree where within each level, every node has either 0 or k children and all leaves are at the same depth.

[12] We use the terms box and node to refer to the regions which originate from the subdivision process described above as well as nodes of the tree data structure used to describe this subdivision (the former emphasizes the geometrical aspect while the latter is more appropriate when talking about data structures).

Definition 2. *Two boxes are said to be* ***well separated*** *if they are at the same refinement level and are not near neighbours.*

Consider two boxes b, b' as before, but now assume they are well separated. The minimum separation between any point $z \in b'$ and the center z_0 of b is $s_{\min} = \frac{3}{2}L$, s.t. the constant c in the error bound (3.9) is bounded below by

$$c_{\min} = \frac{s_{\min}}{\sqrt{d}L/2} = \frac{3}{\sqrt{d}} = \begin{cases} \frac{3}{\sqrt{2}} \approx 2.12 & \text{in 2D,} \\ \sqrt{3} \approx 1.73 & \text{in 3D.} \end{cases} \tag{4.1}$$

Thus, for well separated boxes, the multipole approximation *is* valid, and the truncation error decays roughly as $1/2^p$ (in 2D) or 0.57^p (in 3D) in the order p of the truncated series. Specifically, a given accuracy (absolute error) ε can be achieved by choosing

$$p \approx \begin{cases} \log_2(\varepsilon^{-1}) & \text{in 2D,} \\ \log_{1.73}(\varepsilon^{-1}) & \text{in 3D.} \end{cases} \tag{4.2}$$

Definition 3. *With each box b is associated an* ***interaction list****, consisting of the children of the near neighbours of b's parent which are well separated from b.*

The definition of the interaction list is motivated by the following line of thought. Let z be an evaluation point that falls into a box b at level l, and let a be b's parent. Any box at level $l-1$ that is well separated from a is also well separated from b - but if the potential is to be accounted for via multipole expansions, it makes sense to evaluate multipole expansions at the level of the least possible refinement (since then the savings in work are the greatest), hence we can assume that all boxes well separated from a have already been processed at levels $l-1$ and higher.

The only boxes not processed at level $l-1$ are the near neighbours of a. Since a is an internal node, the near neighbours of a (including a) are refined into boxes at level l, where they are either well separated from b (and hence part of b's interaction list) or near neighbours of b. Thus the set of children of a's near neighbours splits w.r.t. b into two sets: once the interaction list of b, where the multipole approximation *is* valid, and the list of b's near neighbours, which must either be processed at a finer level (where the same split occurs for b's children) or processed directly (if b is a leaf).

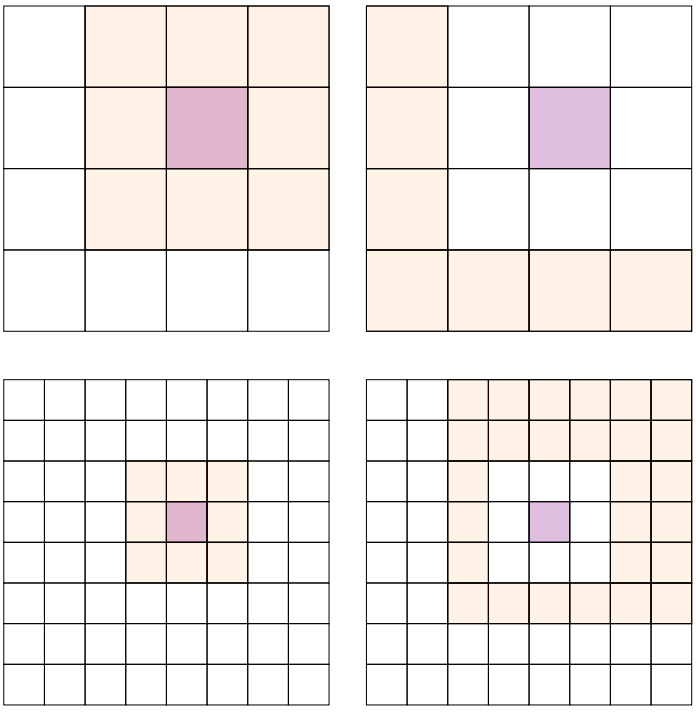

Figure 8: Near neighbours (left) and interaction lists (right) for a box at refinement levels two (top) and three (bottom)

These considerations lead to the following algorithm:

1. Given a level of accuracy ε, chose p according to (4.2).

2. Refine the computational domain into a tree of boxes as described previously, and halt the refinement after $\approx \log N$ levels.

3. At every level, for every box, form a p-th order multipole expansion of the potential due to all sources within that box.

4. For evaluating the potential at a given point z, start with $b = root$ (at level 0) and let $\phi = 0$. For the current box b, evaluate the potential in b's interaction list, add the results to ϕ, set b to the child of the current box containing z and repeat the procedure until the current box is a leaf. Then, evaluate the potentials due to the sources contained in the current node's near neighbours directly.

A perfect k-ary tree of height $\mathcal{O}(\log N)$ contains $\mathcal{O}(N)$ nodes. Distributing all the sources to the leaves takes $\mathcal{O}(N)$ operations (since determining the leaf a source falls into requires one multiplication for every dimension). In step 3, every particle contributes to one multipole expansion per level, hence $\mathcal{O}(N \log N)$ operations are needed for the construction of all multipole expansions. Assume the potential is to be evaluated at the locations of the sources: For every source, at every level, the multipole expansions of the boxes in exactly one interaction list have to be evaluated. Since the size of the interaction list is bounded by 27 in 2D and 189 in 3D, the work done per internal level and source is constant. At the highest level of refinement, the potentials due to the sources in one box's near neighbour list have to be processed directly - but since the height of the tree is $\approx \log N$, the number of charges contained in a leaf and hence the work per source done in this step is constant, s.t. the total amount of work done in the evaluation phase and by the complete algorithm is $\mathcal{O}(N \log N)$.

4.1.2 The FMM

The $\mathcal{O}(N \log N)$ algorithm of the previous section can be improved to an $\mathcal{O}(N)$ scheme, the full FMM, via the following observations:

1. The construction phase takes $\mathcal{O}(N \log N)$ operations, since every particle enters into a multipole expansion at every level. But theorem 2 provides a mean to obtain the multipole expansion of a parent box b by shifting the multipole expansions of its children to b's center and summing the resulting expansions. This step reduces the cost of constructing all multipole expansions to $\mathcal{O}(N)$, since construction and shifting of a multipole expansion takes a constant amount of work, $\mathcal{O}(N)$ expansions have to be created at the leaf level and $\mathcal{O}(N)$ shifts have to be performed (2^d for every internal node of the tree).

2. The other contribution to the $\mathcal{O}(N \log N)$ runtime of the previous algorithm was the evaluation phase, in which every evaluation of the total potential required $\mathcal{O}(\log N)$ multipole expansions to be evaluated. This problem is surprisingly similar to that associated with the direct solution of (1.1), and the solution is, surprisingly enough, analogous: It consists of combining the representations of the potentials due to multiple groups of sources, described by multiple multipole expansions, into a single local expansion, and the mechanism of this combination is given by theorem 4. The idea, then, is to not evaluate any multipole expansions directly, but rather to convert, at every box b, the multipole expansions of the boxes in b's interaction list into local expansions around b's center.

3. By itself, the conversion of multiple multipole expansions into a local expansion yields no asymptotic improvements in computational cost.[13] However, as in the case of the multipole expansion, local expan-

[13] Evaluating the local expansions directly would merely decrease the number of series to be evaluated per box from 2^d to 1.

sions w.r.t. different centers can be combined via theorem 5, allowing to delay the evaluation of the local expansion of a box b at level l by shifting that expansion to the centers of b's children and adding the resulting expansion to the local expansions obtained by the children themselves by conversion of multipole expansions. The effect of this is that for any given evaluation point z only one local expansion is ever evaluated, that being the local expansion of the leaf ℓ containing z. This local expansion describes the potential of all sources except for those in boxes in ℓ's near neighbour list. These remaining boxes are, as in the $\mathcal{O}(N \log N)$ algorithm, processed directly, with the computational cost for this step remaining unchanged at $\mathcal{O}(1)$.

These considerations lead to the following algorithm, which besides an initialization phase consists of two passes: An upward pass, in which multipole expansions due to sources are first formed at the leaf level and then shifted upwards in the hierarchy, and a downward pass, in which the multipole expansions are converted to local expansions and combined with local expansions shifted downwards from parents to children.

1. Initialization Phase

    1. Choose a level of refinement, $n \approx \log_{2^d}(N/s)$, s.t. there are on average s particles per leaf, and a level of accuracy ε, and set p according to (4.2).

    2. Refine the computational domain into n levels.

2. Upward Pass

    1. At the highest refinement level n, for every leaf ℓ, form multipole expansions of the potential due to all sources contained in ℓ around ℓ's center by theorem 1.

    2. For refinement levels $n-1$ to 0, for every box b in a level, form a multipole expansion due to all sources contained in b w.r.t. the center c of b by shifting the multipole expansions of b's children to c via theorem 2 and adding them.

3. Downward Pass

    1. For refinement levels 2 to n, for every box b, convert the multipole expansions of each box in b's interaction list into a local expansion around b's center and add the resulting expansions.

    2. For refinement levels 2 to $n-1$, for every box b, shift b's local expansion to the centers of b's children, and add the result to their local expansions.

Once the downward pass is completed, a local expansion has been created for every leaf that describes the potential due to all sources outside the leaf's near neighbours. The number of remaining sources is bounded by $9s$ (as a given box has at most 9 near neighbours including itself) and these sources are processed directly.

4.2 The Adaptive Algorithm

If the assumption of a uniform spatial distribution of sources is dropped, the balanced algorithm of the previous section performs rather poorly, as the direct computation of near neighbour interactions remains at quadratic

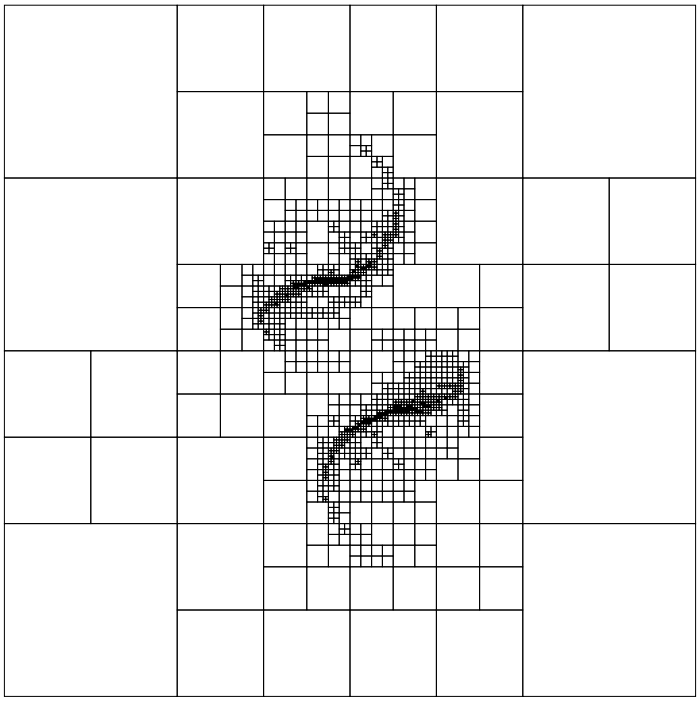

Figure 9: Leaves of an adaptive quadtree of height 10 for a data set of 36000 masses (obtained as a snapshot of a gravitational simulation).

complexity such that the asymptotic cost of the direct evaluation phase may approach $\mathcal{O}(N^2)$ in the worst case. Similarly, refining the domain into more than $\mathcal{O}(\log N)$ levels increases the total number of boxes and hence the total computational cost above $\mathcal{O}(N)$.

These problems can be overcome by refining the computational domain adaptively, i.e. having a higher level of refinement in regions of high particle density. This is again realized by a quadtree (2D) or octree (3D) data structure, now of the adaptive variety, which is constructed as follows:

1. Fix a threshold $s > 0$, which specifies the maximal number of sources that a leaf is allowed to contain.

2. Beginning at level 0, split every node containing more than s sources into four equal boxes (its children) and stop once all boxes contain no more than s sources.

Figure 9 shows an example of a tree constructed by the procedure outlined above for a highly non-uniform data set. For such distributions, the ideas underlying the balanced algorithm described in the previous subsection remain essentially unchanged. However, some modifications are necessary in order to account for the interactions between boxes at different refinement levels. To proceed, the following definitions, taken from [6], are required.

Definition 4. *With each box b at level l are associated five lists of boxes, determined by their positions w.r.t. b.*

1. *The list U_b is non-empty only for childless (leaf) boxes: It contains b and all childless boxes that share a boundary point with b.*

2. *The list V_b is similar to the interaction list of def. 3.*[14] *It contains all children of b's parent's near neighbours, that are well separated from b.*

3. *The list W_b is, as U_b, non-empty only for childless (leaf) boxes: It consists of all descendants of b's near neighbours, who are not adjacent to b themselves, but whose parents are adjacent to b.*

4. *The list X_b is formed by all boxes c s.t. $b \in W_c$.*

5. *The list Y_b is formed by all boxes that are well separated from b's parent.*

As an example, consider the box b in fig. 10.

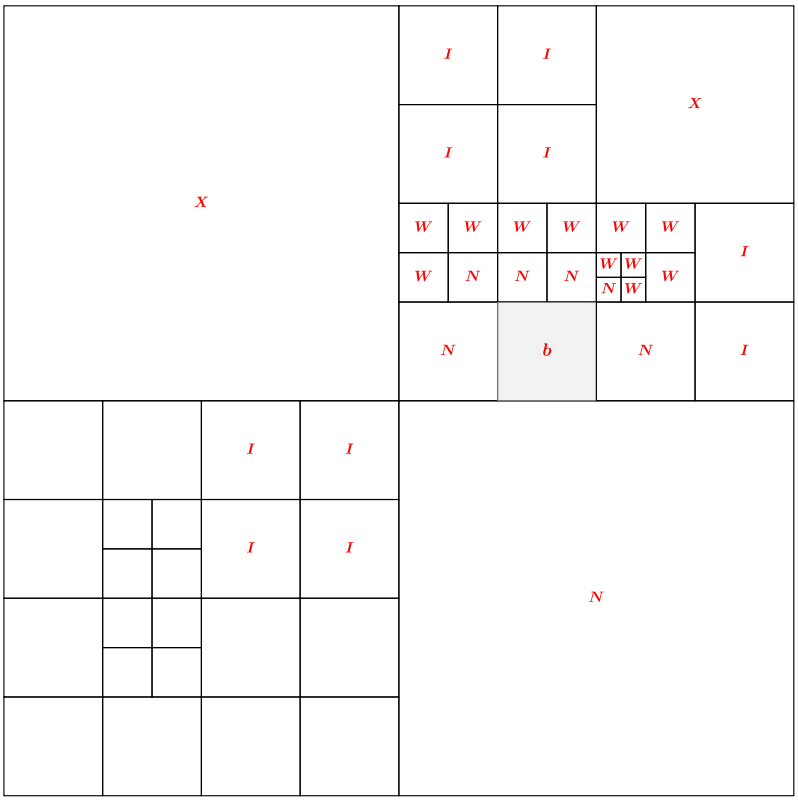

Figure 10: Leaves of a simple adaptive tree, a box b and its relations with other boxes as per def. 4.

1. Interactions between sources in b and those in $b' \in U_b$ have to be computed directly, since neither multipole nor local expansions provide valid approximations.

2. The potential due to sources contained in $\overline{b} \in V_b$ can be accounted for everywhere in b by converting $\overline{b}$'s multipole expansion into a local expansion centered at b.

3. Every box $\overline{b} \in W_b$ is smaller than b and separated from b's center by at least $\frac{3}{2} l_{\overline{b}}$, where $l_{\overline{b}}$ denotes the

[14] The crucial difference is that parts of the region occupied by nodes in the interaction list of the balanced algorithm may now be occupied by leaves from a coarser level of refinement, cf. fig. 10.

length of the side of $\overline{b}$, s.t. for sources in b, evaluating $\overline{b}$'s multipole expansion truncated to p-th order with p given by (4.2) guarantees an error smaller than ε.[15]

4. Every box $\overline{b} \in X_b$ is childless and larger than b. While $\overline{b}$'s multipole expansion does not converge in b, thm. 3 allows forming a local expansion of the potential due to the sources in $\overline{b}$ around b's center which converges at a rate consistent with (4.2).

5. The interactions between sources in b and sources in boxes in Y_b are not evaluated explicitly during the processing of b, but rather during the processing of b's ancestors, at coarser levels.

Based on the above considerations, the following algorithm can be formulated[16] which closely resembles the balanced algorithm of the previous subsection.

1. Initialization Phase

    1. Choose a threshold s of allowed particles per leaf, a level of accuracy ε and set p according to (4.2).

    2. Refine the computational domain until no box contains more than s particles as described in sec. 4.2, and let n denote the highest refinement level.

2. Upward Pass

    1. For every leaf ℓ (at every refinement level), form multipole expansions of the potential due to all sources contained in ℓ around ℓ's center by theorem 1.

    2. For refinement levels $n-1$ to 0, for every box b in a level, form a multipole expansion due to all sources contained in b w.r.t. the center c of b by shifting the multipole expansions of b's children to c via theorem 2 and adding them.

3. Downward Pass

    1. For refinement levels 2 to n, for every box b, convert the multipole expansions of each box in V_b into a local expansion around b's center and add the resulting expansions using theorem 4.

    2. For refinement levels 2 to n, for every box b, form local expansions of the potential due to all charges in boxes in X_b around the center of b using theorem 3.

    3. For refinement levels 2 to $n-1$, for every box b, shift b's local expansion to the centers of b's children using theorem 5, and add the result to their local expansions.

Upon completion of the downward pass, a local expansion has been computed for every leaf ℓ, which accounts for the potential of all sources outside of boxes in $U_\ell \cup W_\ell$. To see this, note that for any box b, the region that would, in the balanced algorithm, make up b's interaction list, now contains either nodes at the same level as b (making up V_b) or leaves from a coarser level of refinement (making up X_b), the influence of both of which is

[15] Converting and combining the multipole expansions of boxes $\overline{b} \in W_b$ into a single local expansion seems attractive at first, since it saves on the number of series to be evaluated, but thm. 4 does not guarantee the required radius of convergence in this case.

[16] The formulation given here is based on [6], however arranged differently. Furthermore, in [6], thm. 3 is omitted entirely and instead thm. 4 is used to a similar effect.

accounted for in steps 3.1 and 3.2 and, if b is a parent node, transmitted down to ℓ in step 3.3 of the algorithm above. The region that consists of b's near neighbours in the balanced algorithm is processed recursively by b's children if b is a parent box.

If b is a leaf, the set of boxes contained in this region splits into two subsets, those being the set of leaves adjacent to b (making up U_b) and the descendants of b's near neighbours which are not adjacent to b (making up W_b). Thus, for evaluating the potential at a given point in any leaf, it remains to account for the influence of sources in boxes in $U_b \cup W_b$, which has been discussed in the remarks surrounding definition 4. We arrive at the following procedure for evaluating the potential at any point $\mathbf{x}$ in the computational domain:

1. Determine the leaf ℓ that $\mathbf{x}$ falls into and let $\phi = 0$.

2. Evaluate ℓ's local expansion at $\mathbf{x}$ and add the result to ϕ.

3. For every box $b \in W_\ell$, evaluate b's multipole expansion and add the result to ϕ.

4. For every box $b \in U_\ell$, compute the potential due to all sources in b at $\mathbf{x}$ directly and add the results to ϕ.

In particular, if potential or force are to be computed at every source location, it is beneficial to process sources by leaves to avoid repeated tree traversals.

The time complexity of the adaptive algorithm is argued to be $\mathcal{O}(N)$ in [6], where the argument is made that the fixed numerical precision available on modern computer architectures limits the height of the adaptive tree to a constant. This argument has, however, been called into doubt [19] with some authors suggesting that the complexity of the adaptive FMM is $\mathcal{O}(N \log N)$ rather than $\mathcal{O}(N)$.

4.3 The Parallel Algorithm

The parallelization of the FMM is aided by two observations that can be made of the algorithms described in the preceding sections.

1. Once the tree has been constructed (and all information such as near neighbour, interactions lists, etc. is available), both the upward and the downward pass proceed level by level.

   In the upward pass, the only results required to process a parent box b at level l are the multipole expansions of b's children at level $l - 1$, so boxes can be processed in parallel within levels, while levels have to be processed sequentially.

   In the downward pass, the results required by a box b at level $l \geq 2$ are 1. multipole expansions of those boxes in b's interaction list, which are already available at this stage, and 2. the local expansion of b's parent, which must have been computed while processing level $l - 1$ in the downward pass. If the multipole-to-local and the local-to-local operations are processed during a single pass, the implication of this is again that levels are processed sequentially during the downward pass, while all boxes within a level may be processed in parallel.

2. The potential and force evaluation phase requires only the evaluation of local (and multipole, in the adaptive case) expansions and direct evaluations of the potential, all of which are read-only operations,

and hence any number of evaluations of potential and force may be done in parallel once all expansions have been computed.

5 Implementation

In this section, we give a quick, hands-on introduction (sec. 5.1), as well as brief overviews over the structural and design decisions made in our implementation (sec. 5.2) and the process of parallelization (sec. 5.3).

The programming language used was `C++ 17`, with the only library dependency (besides the `C++ STL` and the `C++ filesystem` library) being the `GNU Scientific Library (GSL)`, which is used for the computation of factorials, binomial coefficients and, in particular, spherical harmonics. For ease of use, an interface to `Wolfram Mathematica 12` also exists, which makes available a subset of the features of the `C++` implementation in a high level language.

5.1 Examples & Running the Code

Our implementation is header-only. The relevant header files are `balanced_fmm_tree.hpp` for the balanced, and `adaptive_fmm_tree.hpp` for the adaptive algorithm, i.e. either

```
#include "/path/to/lib/balanced_fmm_tree.hpp"
```

or

```
#include "/path/to/lib/adaptive_fmm_tree.hpp"
```

suffice for access to all relevant implementation components, which are part of the namespace `fmm`. The code can be built with the command

```
gcc-8 -Wall -Wno-unknown-pragmas -std=c++17 -pedantic  \
    -O3 -fopenmp helloFMM.cpp -o helloFMM -lstdc++fs -lgsl -lgslcblas
```

We provide a class `Vector_<d>` for d-dimensional vectors, which can be accessed like arrays and constructed with an `std::array<double,d>`, as well as a class `PointSource_<d>` for point sources in d dimensions, constructed by a tuple of a vector and a source strength, as shown below:

```
using namespace fmm;

Vector_<3> v{{1,2,3}};  // double {{}} constructs std::array intermittently
PointSource_<3> ps{v, 4};
```

The FMM algorithm is executed by constructing an instance of `BalancedFmmTree` or `AdaptiveFmmTree`. The constructor has two template arguments, the dimension as well as an optional boolean `field_type`, signalling whether to compute gravitational (if `field_type` is `true`, default) or Coulombic (if `field_type` is `false`) potentials and forces. Its main arguments are

- `sources`, a `std::vector<PointSource_d>`,
- `sources_per_leaf`, `int` specifying the maximal number of sources per leaf,
- `eps`, a double specifying the desired accuracy,
- `force_smoothing_eps` (optional, default 0), a regularization parameter used in N-body simulations.[17]

Once all these parameters are defined, a tree can be built e.g. by

```
AdaptiveFmmTree<d, field_type> fmm_tree(sources, sources_per_leaf, eps,
    force_smoothing_eps);
```

This constructor builds a tree with the desired structure, determines neighbourhood information and executes the up- and downward passes of the FMM. To evaluate potential or force field at a point given as a `Vector_<d>` object `eval_point`, the following calls are required:

```
double potential = fmm_tree.evaluatePotential(eval_point);
Vector_<d> force = fmm_tree.evaluateForcefield(eval_point);
```

The potential energies of and forces on all sources can be acquired in a similar fashion:

```
vector<double> potential_energies = fmm_tree.evaluateParticlePotentialEnergies();
std::vector<Vector_<d>> forces = fmm_tree.evaluateParticleForces();
```

5.2 Structure & Design

This section gives a rough overview over the ideas and components we consider central to our implementation. It is necessarily brief and we refer the interested reader to the source code and its comments for details. The guiding principle that we attempted to follow in our implementation was maximizing code reuse while providing a simple interface. As discussed in the previous sections, the FMM may be used in multiple configurations: for gravitational and Coulombic interactions, in two and three dimensions and for uniform and non-uniform distributions of sources. The following observations motivated the structure of our implementation:

- The difference between the Coulomb potential and the Newtonian gravitational potential is, in both two and three dimensions, one of sign only (cf. (3.3) and footnote 6), if it is neglected that one is proportional to the charge and the other to the mass. Implementation wise, it is therefore sensible to represent both point charges and point masses as tuples of a vector and a source strength, compute an electrostatic potential due to these point sources and then, once the electrostatic potential is known wherever desired, switch signs if the gravitational potential is to be returned.

[17] Regularization is used to deal with detrimental effects which the singularity of inverse distance force laws (cf. eqs. (3.5), (3.31)) at $\mathbf{x} = \mathbf{x}_0$ has on numerical simulations, which involves [20] "smoothing" the force by introducing a constant factor ε_s as such:

$$\mathbf{E}_{\mathbf{x}_0}(\mathbf{x}) = \begin{cases} \frac{q}{r^2+\varepsilon_\mathrm{s}}(\mathbf{x}-\mathbf{x}_0)\,, & \text{in 2D,} \\ \frac{q}{r^3+\varepsilon_\mathrm{s}}(\mathbf{x}-\mathbf{x}_0)\,, & \text{in 3D} \end{cases} \tag{5.1}$$

- The difference between the two and three dimensional algorithms is in detail and not structural. By this we mean that, while obvious differences, e.g. in geometry (number & positions boxes in the near neighbour and interactions lists etc.) or in expansion order (cf. eq. (4.2)) exist, these differences appear in fixed locations and can be accounted for at compile time. The template mechanism offered by `C++` is ideally suited for this purpose.

- The difference between the balanced and the adaptive algorithm is, however, structural: From tree construction, upward and downward pass and potential evaluation to the information which needs to be stored in nodes and leaves, significant differences between the algorithms exist, strongly suggesting to separate these algorithms.

Based on these observations, we opted for organizing the relevant components into several classes, related by inheritance and shown schematically in fig. 11.

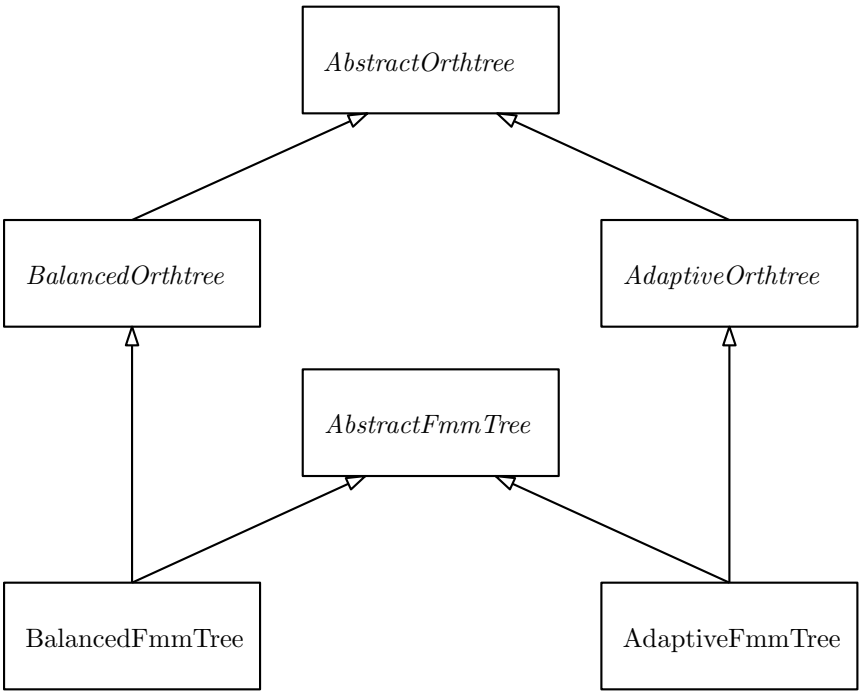

Figure 11: Schematic diagram of the main classes and their inheritance relations. An arrow $B \longrightarrow\!\triangleright\, A$ indicates that class B inherits from ("is-a") class A, italicization denotes abstract classes.

- The class `AbstractOrthtree`[18] defines the tree structure (pointer based) and `BaseNode` class (nested, see fig. 12), captures essential geometric information (height, number of children of a node) allows to query the directions in which the centers of a given box's children lie.

- The classes `BalancedOrthtree` and `AdaptiveOrthtree` provide functions which are required for the construction of balanced and adaptive trees, e.g. for determining which leaf of a balanced tree a given point falls into.

- The class `AbstractFmmTree` contains members and methods shared by both the balanced and adaptive versions of the FMM, e.g. computing the extent of the computational domain.

[18]The terminology is based on *orthants*, which generalize quadrants and octants. In the literature, the term hyperoctree is used alternatively.

- The classes `BalancedFmmTree` and `AdaptiveFmmTree` are considerable larger than the others and contain Node and Leaf classes (nested) specific to the respective algorithms as well as routines that correspond to the tree building (upward & downward pass) and evaluation algorithms described in sec. 4.

Thee tree structure is, as mentioned, pointer based. Figure 12 shows a class diagram detailing the inheritance relationships between the different types of nodes as well as their most important members for the adaptive case. In particular,

- The `BaseNode` class stores geometric information such as center, extent, parent and children associated with a node and provides methods for checking adjacency of boxes as well as membership (of a point, in the region covered by the node).

- The `FmmNode` class stores neighbour information, in particular also the lists V and X described in sec. 4.2, as well as the multipole and local expansions associated with every node.

- The `FmmLeaf` class stores additionally the list W as well as the sources contained in a leaf.

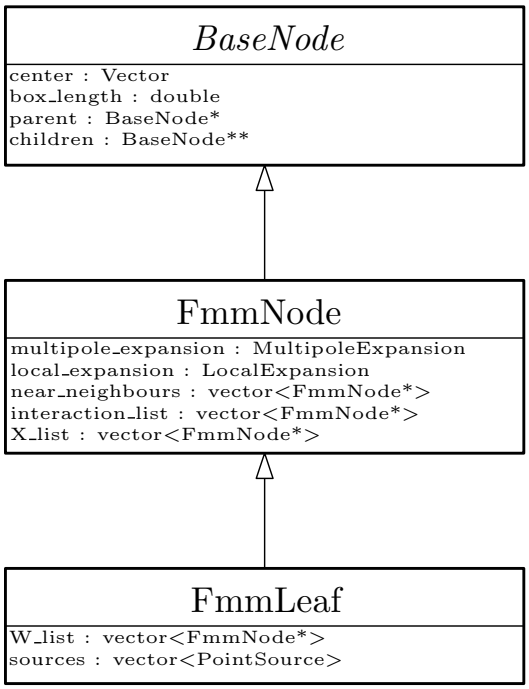

Figure 12: Schematic diagram of the structure of nodes and leaves and their inheritance relations for the adaptive algorithm. An arrow $B \longrightarrow\!\!\triangleright\ A$ indicates that class B inherits from ("is-a") class A, italicization denotes abstract classes and methods. All classes shown in this diagram are nested classes associated with a tree class from fig. 11.

Figure 13 shows a class diagram of the components of the code which realize the multipole and local expansions.

- The abstract class `SeriesExpansion` bundles information and code common to both multipole and local expansions, such as expansion center, order and coefficients, provides a method for adding series expansions w.r.t. the same center and defines an interface for evaluating potential and force due to a series expansion.

- The classes `MultipoleExpansion` and `LocalExpansion` realize the expansions introduced in section 3. They contain the implementations of the various shift and conversion operations, constructors (from sources, from

other expansions) and implementations of the methods for evaluation of both potential and force due to the charges represented by the expansion.

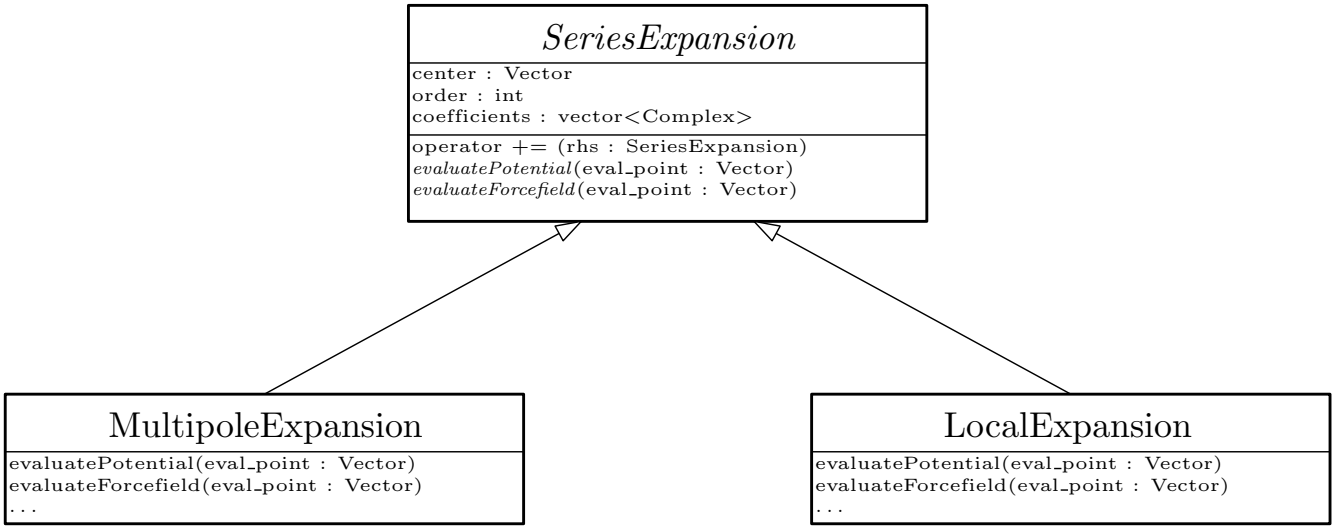

Figure 13: Class diagram of the series expansion interface. An arrow $B \longrightarrow A$ indicates that class B inherits from ("is-a") class A, italicization denotes abstract classes and methods.

5.3 Parallelism

The sequential implementation was parallelized using `OpenMP` version 4.5. Once the available parallelism is identified (cf. 4.3), exploiting it is possible in a relatively straightforward manner using `OpenMP`'s loop constructs, provided nodes and leaves are stored in a way which facilitates looping over them level by level.

In the balanced case, this is realized by allocating a contiguous region of memory for the entire tree with the nodes of subsequent levels being stored in consecutive slices of this region, while with the adaptive algorithm, this is realized by splitting nodes level by level and storing nodes level-wise in dynamic length arrays.

6 Experimental Evaluation

In this section, we present the results of evaluating our implementation w.r.t. accuracy (sec. 6.1), as well as sequential (sec. 6.2) and parallel performance (sec. 6.3).

Since on one hand the adaptive algorithm is essentially equivalent to the balanced algorithm when the particle distribution is sufficiently uniform, and on the other hand, the balanced algorithm is in large parts equivalent to the direct algorithm when the data set is sufficiently non uniform, it suffices to consider (a) the adaptive algorithm with a non uniform data set and (b) the balanced algorithm with a uniform data set. We found the case (b) to be in general more challenging for our implementations in the sense that runtimes were slightly higher and accuracies slightly lower than with case (a). In the sequential setting, these differences were not observed to be substantial, so we restrict ourselves to presenting accuracies and sequential runtimes for the hardest case, i.e. (b), in order to avoid repetition. When evaluating parallel efficiencies, these differences are considered as well.

Wherever we refer to the evaluation of forces or potentials by the FMM or the direct algorithm in the sections that follow, the computation of all pairwise interactions is meant, i.e. the computation of the potential or forces at all source locations.

6.1 Accuracy

In this section, we investigate the accuracy of our implementation. This is done mainly via two routes: (i) For a fixed desired accuracy ε, we compare the results of the FMM with those obtained by the direct algorithm in double precision as the input size N varies, and (ii) for fixed N, we compare the results obtained via the FMM again with those obtained directly in double precision, but this time as ε varies. While (i) provides some insight into whether the error shrinks, stagnates or grows uncomfortably as N grows, (ii) is informative about how much accuracy can be expected in practice for different values of ε.

For evaluating accuracy, data sets of the forms shown in fig. 14 were used with varying numbers of particles.

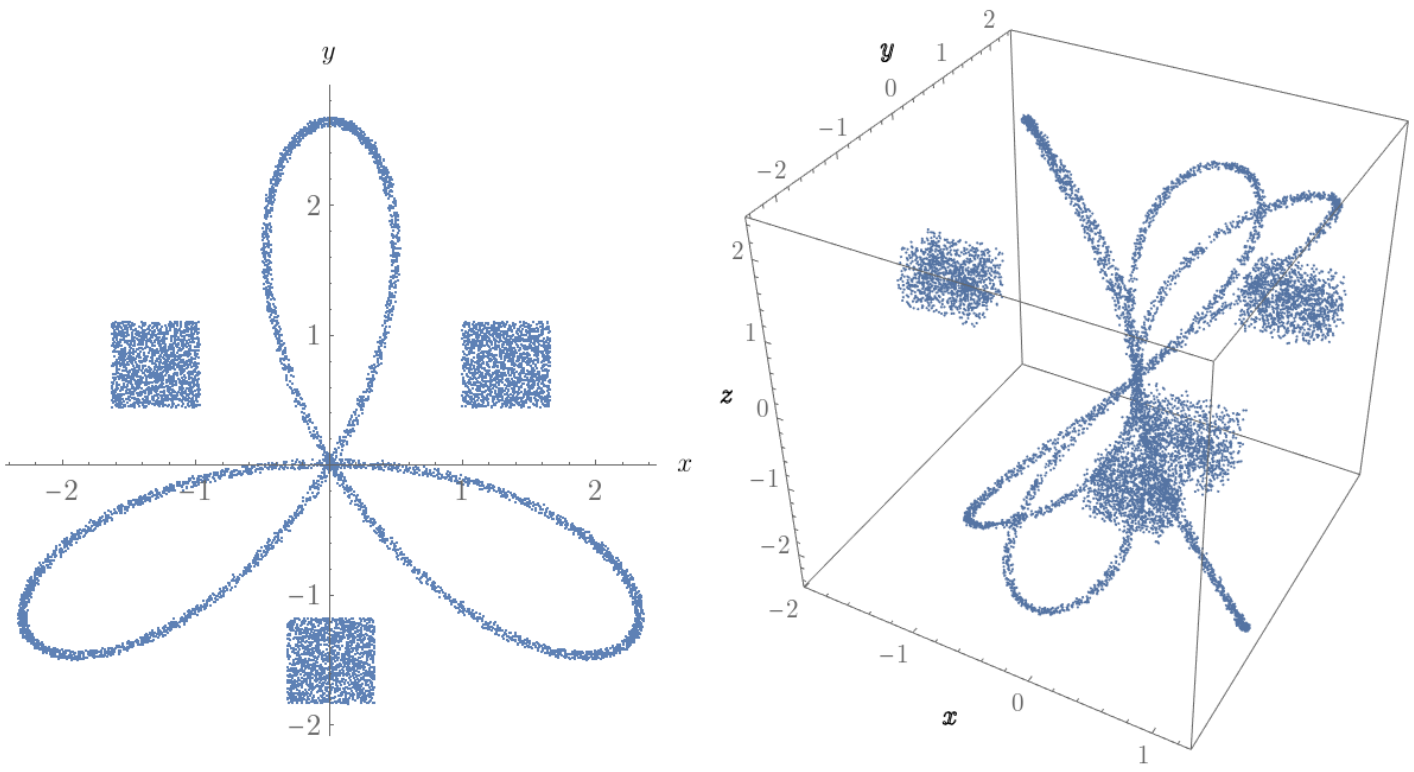

Figure 14: Uniform (2D) and non-uniform (2D, 3D) particle distributions of the type used for evaluating sequential performance and accuracy, $N = 5000$.

We denote with E_P and E_F potential and force error, which can be measured in several ways. The absolute total error $E_{\text{abs,tot}}$ is quantified by the 2-norm of the deviation between the direct (double precision) solution vector q and the approximate solution vector $\widetilde{q}$,

$$E_{\text{abs,tot}} = \|q - \widetilde{q}\|_2 = \sqrt{\sum_i (q_i - \widetilde{q}_i)^2}\,, \tag{6.1}$$

while the relative total error $E_{\text{rel,tot}}$ is defined by

$$E_{\text{rel,tot}} = \frac{\|q - \widetilde{q}\|_2}{\|q\|_2} = \frac{\sqrt{\sum_i (q_i - \widetilde{q}_i)^2}}{\sqrt{\sum_i q_i^2}}\,. \tag{6.2}$$

For the potential, the sum goes over the indices $i = 1, \ldots, N$, while for the forces, $i = 1, \ldots, dN$, i.e. the sum goes over all force components at all source locations. The maximum component errors are defined as

$$E_{\text{abs, max}} = \|q - \widetilde{q}\|_\infty = \max_i |q_i - \widetilde{q}_i|\,, \tag{6.3}$$

$$E_{\text{rel, max}} = \max_i \frac{|q_i - \widetilde{q}_i|}{|q_i|}\,, \tag{6.4}$$

where again $i = 1, \ldots, N$ for potentials and $i = 1, \ldots, dN$ for forces. Furthermore, the letters N, s and ε are used to denote the input parameters of the FMM, i.e. particle number, max. number of sources per leaf and desired accuracy, respectively.

All measurements displayed in this section were taken on an Intel Core i5-4300 quad core with 1.9 GHz.

Accuracy in Two Dimensions

Figure 15 shows force and potential errors obtained from the 2D adaptive FMM and for a non uniform data set for a fixed desired accuracy ε as N increases, while fig. 16 shows, again, force and potential errors, but this time for fixed N as ε is varied.

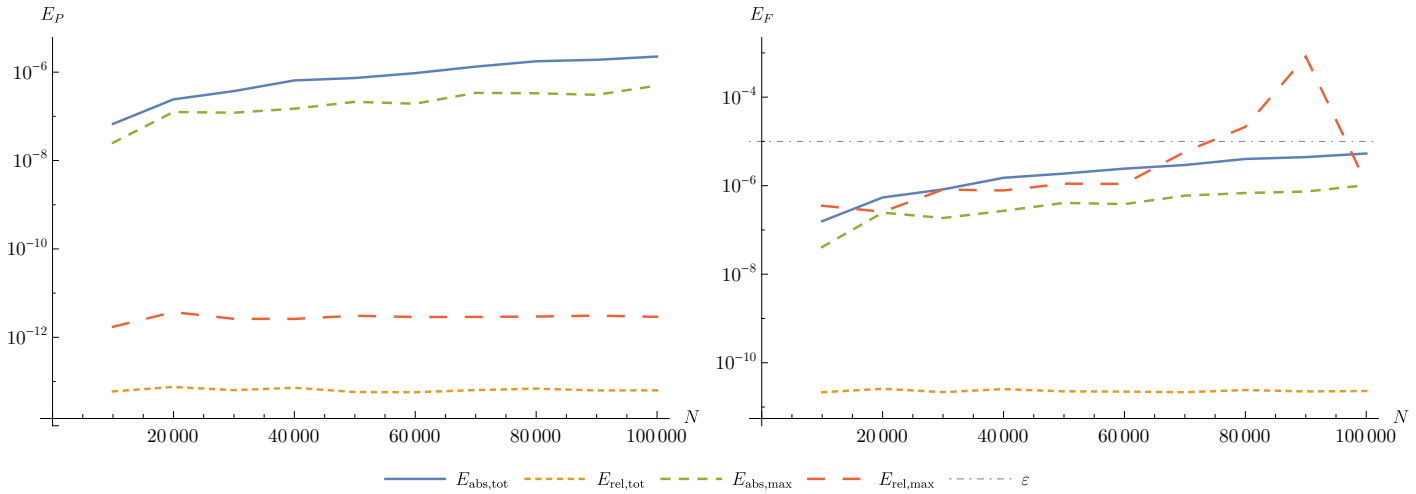

Figure 15: Potential and force errors vs N in two dimensions with $\varepsilon = 10^{-5}$, $s = 200$ and for the non-uniform dataset.

It can be seen that for the data set and the problem sizes considered, the absolute potential errors fall below the desired accuracy ε, while the total relative error is seen to be close to machine precision.[19]

The situation changes for the forces, where again the absolute errors satisfy the error bound and the average relative error is uniformly small (though larger than for the potentials), but where now the maximum relative error grows higher than ε for some values of N.

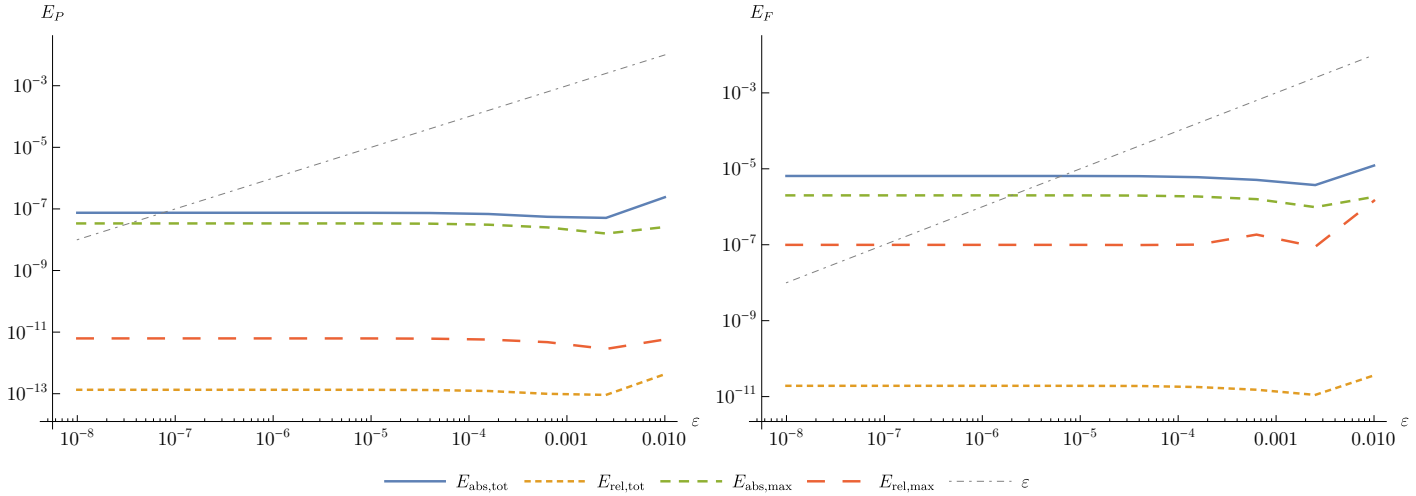

Figure 16: Potential and force errors vs ε in two dimensions with $N = 10^5$, $s = 200$ and for the non-uniform dataset.

[19] Some sources, e.g. [2] find much higher relative errors, but this is likely an effect of the absolute magnitude of the solutions rather than genuine differences in accuracy.

For the error as a function of the input parameter ε, we see an interesting picture in two dimensions. For both potential and force errors, the error initially shrinks with ε, but then stagnates. In particular the absolute total error does not shrink below approximately 10^{-7} for potentials and 10^{-5} for forces. This of course means that e.g. for values smaller than $\varepsilon = 10^{-7}$, the absolute error made by the FMM is not within the desired bounds, and this effect is more pronounced for the force, where the cut-off point is at $\varepsilon \approx 10^{-5}$.

Accuracy in Three Dimensions

Figures 17 and 18 show again force and potential errors vs N and ε, respectively, but this time for the three dimensional adaptive FMM and again with a non-uniform data set.

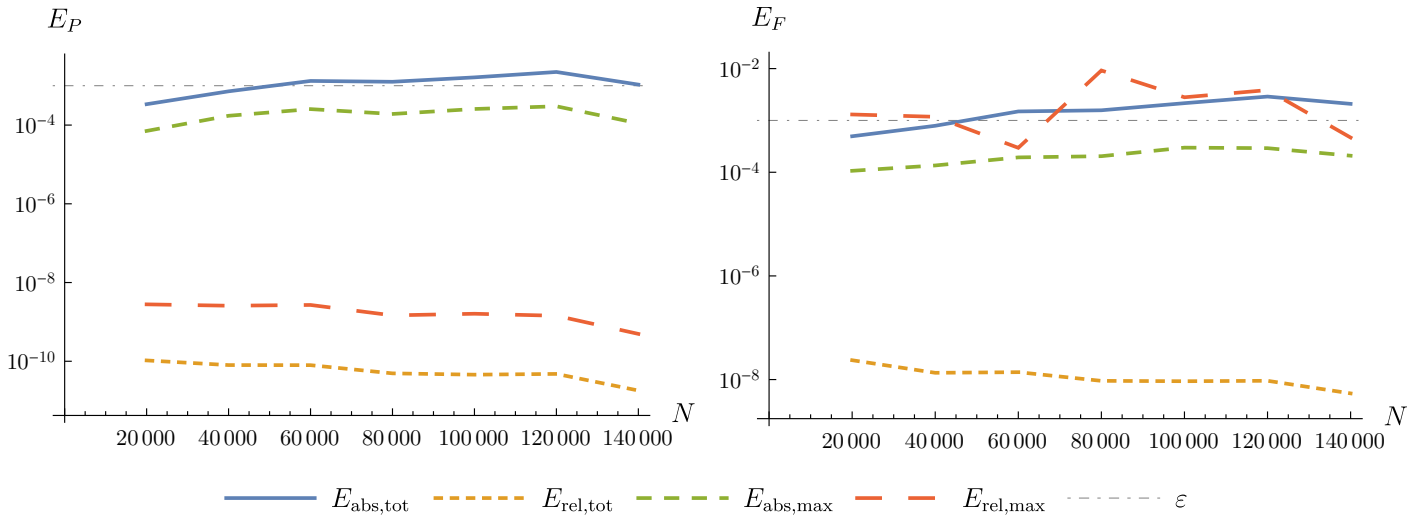

Figure 17: Potential and force errors vs N in two dimensions with $\varepsilon = 10^{-3}$, $s = 2000$ and for the non-uniform dataset.

For the error as a function of the input size N, a similar picture emerges as in the two dimensional case, although now the absolute total error is found to be slightly above $\varepsilon = 10^{-3}$. This does not imply a violation of the error bounds of the FMM however, since the bound is satisfied componentwise for both forces and potentials. As in two dimensions, the force errors are generally larger, and we see here again that the maximum componentwise relative error can become large relative to ε even though the total relative error is very small.

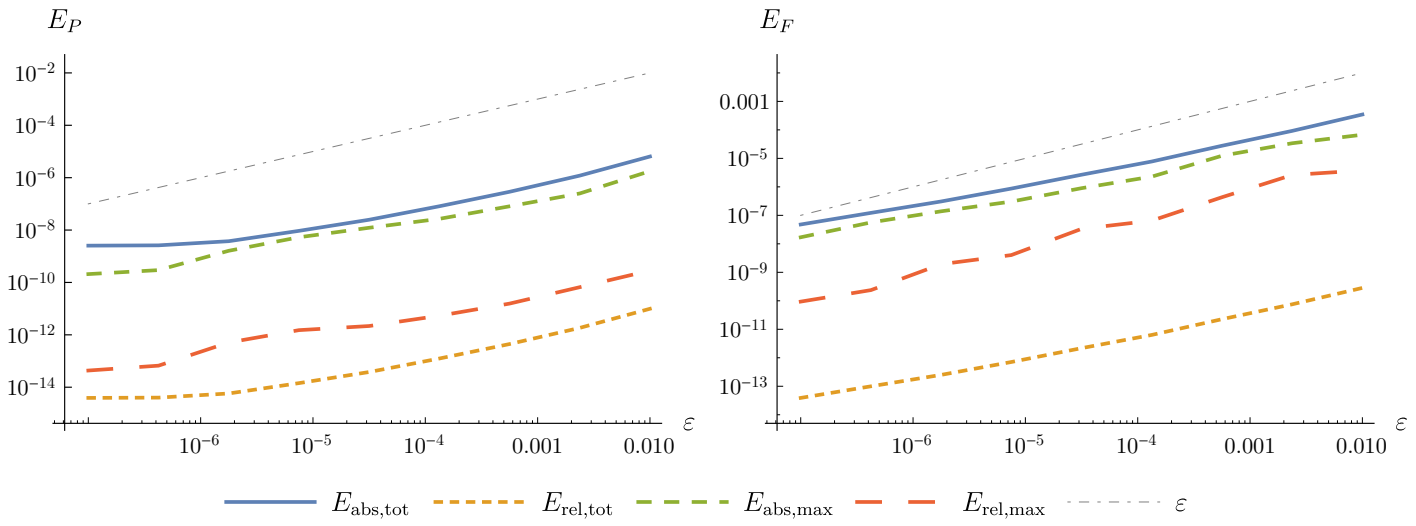

Figure 18: Potential and force errors vs ε in three dimensions with $N = 10^5$, $s = 200$ and for the non-uniform dataset.

For the error as a function of ε, we find a somewhat different pattern than in two dimensions: Now, there is a more direct correspondence between the desired accuracy ε and the error made by the FMM. Nevertheless, also in three dimensions, stagnation can eventually be observed at least in the case of the potential, with the final total absolute error observed being slightly above 10^{-9} for potentials and 10^{-7} for forces - roughly two orders of magnitude less than in the two dimensional case.

Concluding, we note that our implementation of the FMM delivers adequate accuracy until $\varepsilon \approx 10^{-7}$ after which a violation of the expected error bounds seems likely based on the available empirical data. There are several possible reasons for this, e.g. floating point errors occurring in the special functions required for the FMM but it is equally possible that some component of our implementation is the cause of these errors, or that they are intrinsic to the FMM.

Unfortunately, the error tables found in the primary sources make it difficult to draw definite conclusions. E.g. in [6], errors are only given for orders $p = 17$ and $p = 20$, which ($p \approx \log_2(1/\varepsilon)$) correspond to $\varepsilon < 10^{-7}$. In three dimensions, neither timings nor errors are given in the sources where the theoretical basis and algorithm were first described (e.g. in [4]) and those error results given in [11] are for $p \leq 16$.

Spatial distribution of the Error

As a closing note regarding accuracy, it is of interest to consider not just the magnitudes of the errors made by the FMM, but also their spatial distribution. Figure 19 shows a plot of the contours of the relative errors made by the 2D (adaptive) FMM when the potential is evaluated on a uniform grid of points, when sources are both uniformly and non-uniformly distributed.

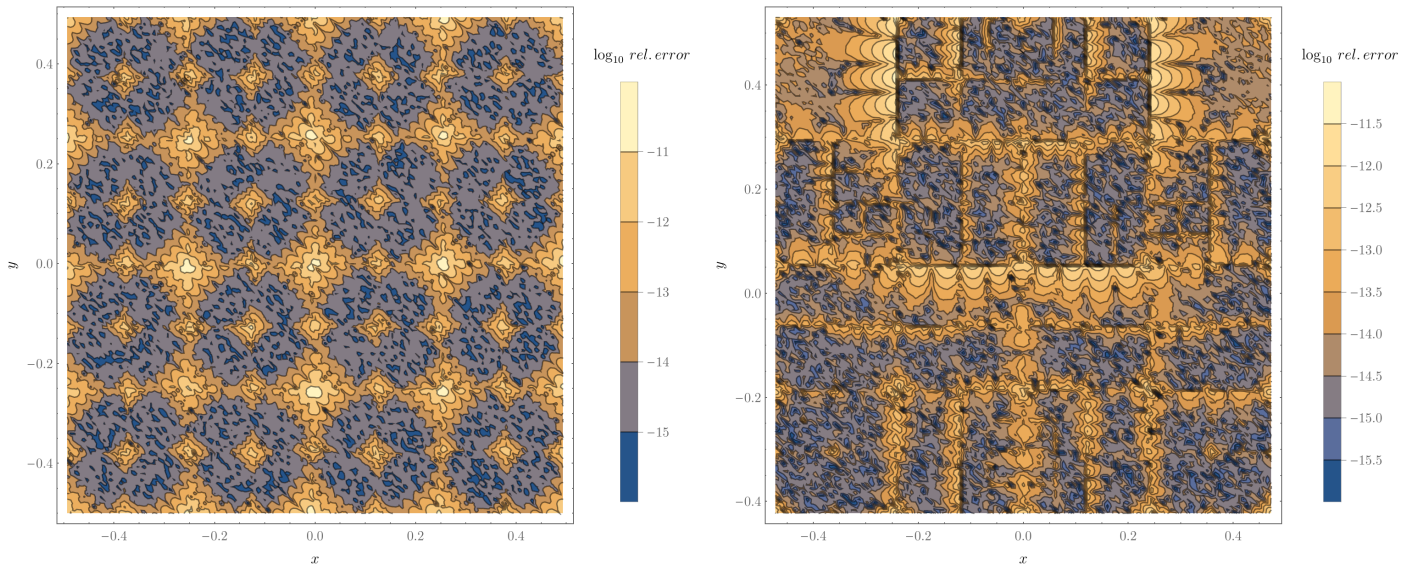

Figure 19: Relative potential errors obtained in sampling the potential on a uniform grid. Uniform (left) and non-uniform (right) particle distributions of the shapes shown in 14, adaptive algorithm, $N = 10000$ particles, $\varepsilon \approx 10^{-3}$ and max. 200 sources per leaf.

It can be seen that regions of high error coincide with box boundaries, with higher errors corresponding to larger boxes and the highest errors being found in box corners. A possible explanation for this is found in the convergence properties of the local expansion (cf. (3.20)), for which the worst error is incurred the farthest from a box's center.

6.2 Runtime comparison with the direct algorithm

In this section, the runtimes of our implementations of the FMM and the direct algorithm are compared for the adaptive algorithm and with a data set of non-uniformly distributed sources of the form shown in figure 14. Regarding terminology, we denote with t_P and t_F the times needed for the evaluation of potentials and forces, and with $S_P = t_{P,\text{dir}}/t_{F,\text{FMM}}$ and $S_F = t_{F,\text{dir}}/t_{F,\text{FMM}}$ the (sequential) speedup.

In two dimensions

Figure 20 shows runtimes and speedups of computing forces and potentials for various N via the FMM (including the cost of building the tree) as compared to the direct algorithm in two dimensions.

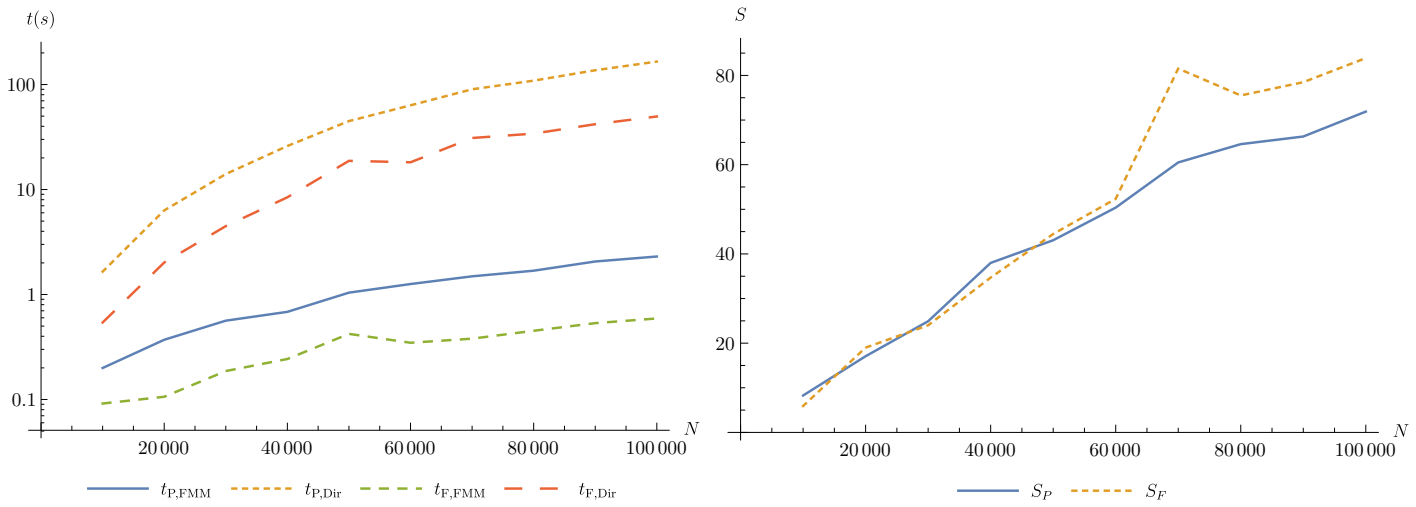

Figure 20: Runtimes of the FMM and the direct algorithm (left), speedups of the FMM over the direct algorithm (right) in 2D.

It can be seen that in two dimensions, our implementation provides a significant improvement over the direct algorithm, with $S_P \approx S_F \approx 20$ for $N = 2 \cdot 10^4$ particles and $S_P \approx 70$, $S_F \approx 80$ for potential and force evaluations with $N = 10^5$ particles. Since the absolute runtimes involved at such N are modest, these speedups enable particle simulations of considerably larger sizes than possible with the direct algorithm to be tackled even on workstations.

In three dimensions

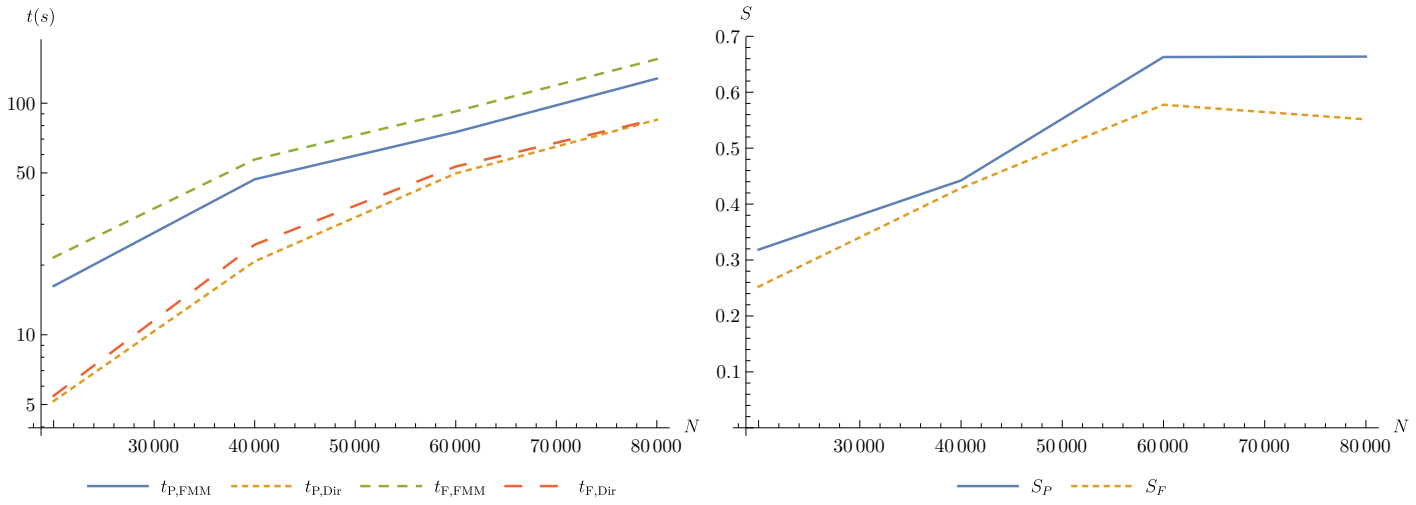

Figure 21: Runtimes of the FMM and the direct algorithm (left), speedups of the FMM over the direct algorithm (right) in 3D.

In three dimensions, we observed the FMM being unable to outperform the direct algorithm for many combinations of the parameters ε, s and N, in particular also those shown here. While it is possible to achieve a speedup over the direct algorithm with our implementation, the cutoff point at which this is reached lies beyond $N = 10^5$. This roughly matches the observations made in [11].

6.3 Parallel Efficiency

In this section, we investigate the parallel efficiencies of our implementation of the FMM and compare with a parallel implementation of the direct algorithm. All results discussed in this section were obtained on a server with 4 sockets of 16 Intel Xeon E7-8867 cores each.

Regarding the terminology used, we denote with p the number of processors, with $S = T_1/T_p$ the speed-up on p processors, where T_1 and T_p are the execution times on one and p processors, respectively, and with $E = S/p = T_1/(pT_p)$ the parallel efficiency. For the efficiency assessment, we used uniform and non-uniform data sets of the form shown in fig. 22.

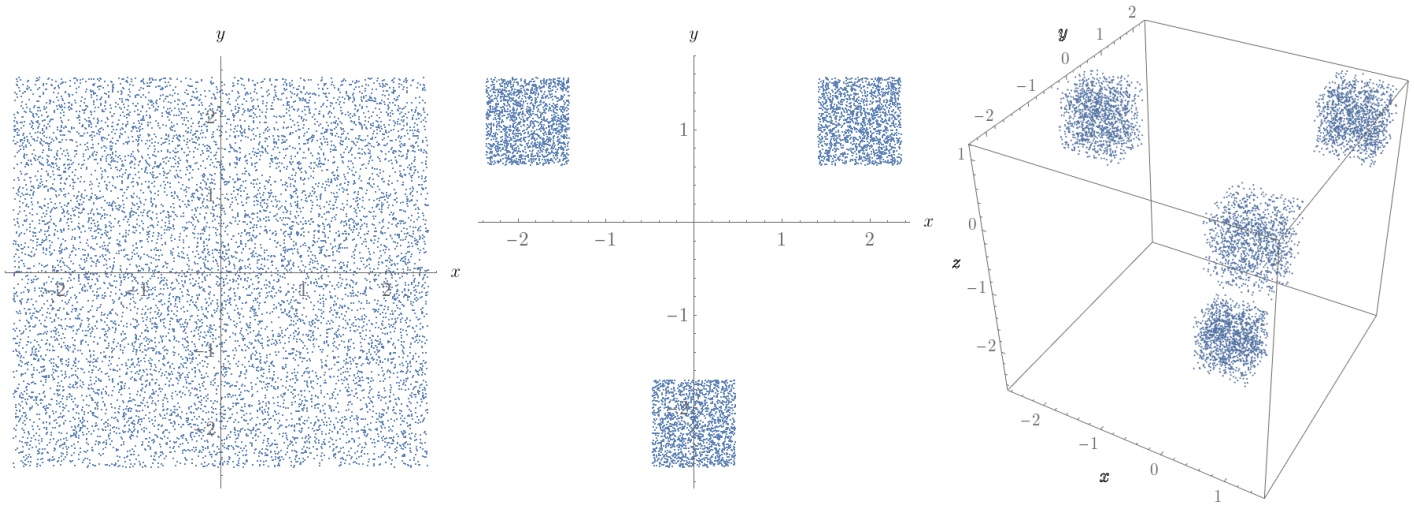

Figure 22: Uniform (2D) and non-uniform (2D, 3D) particle distributions of the type used for evaluating parallel efficiency, $N = 7500$.

In two dimensions

Figures 23 to 25 show runtimes and parallel efficiencies of the components of our implementation for the balanced and direct algorithms with a uniform data set as well as for the adaptive algorithm with both uniform and non uniform data sets. More specifically, for the FMM, the efficiencies of the tree building phase, the potential evaluation phase and the force evaluation phase as well as the efficiencies of the their combination are shown, while for the direct algorithm, efficiencies are shown for potential and force evaluation as well as for their combination.

Input size and accuracy were $N = 10^5$ and $\varepsilon = 10^{-3}$ for all tests, while for the maximal number of sources per leaf, a relatively large value of 1000 was used. The qualitative scaling behaviour of the direct algorithm does not differ between the plots, but is included for comparison.

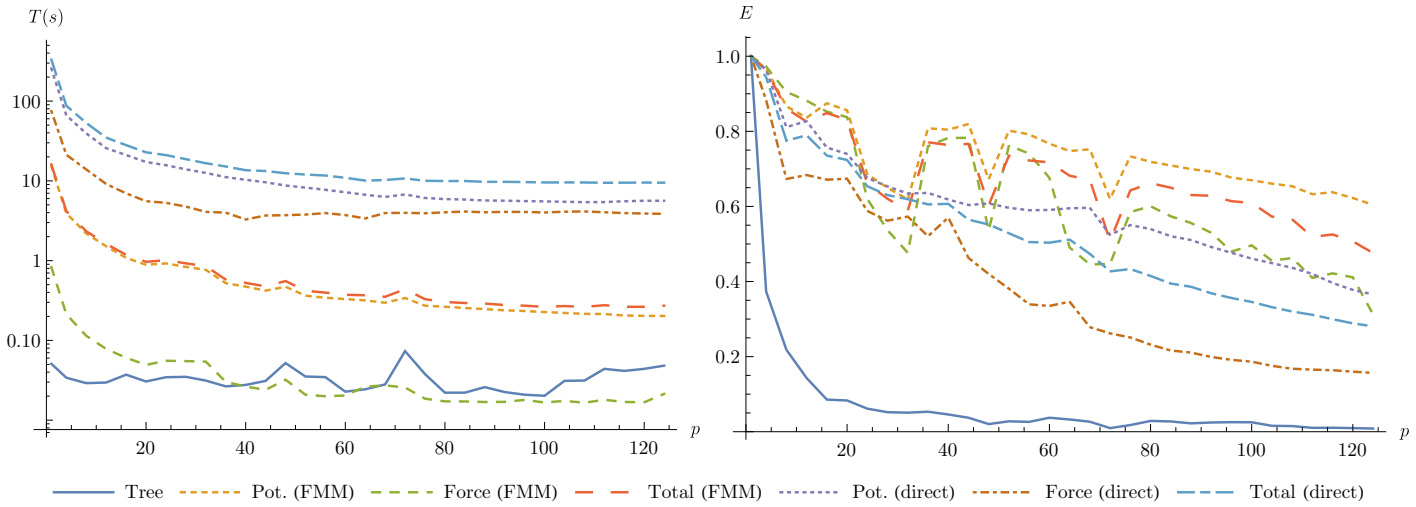

Figure 23: Runtimes and parallel efficiencies in two dimensions, uniform data set & balanced tree.

For the balanced FMM, a total efficiency of $\approx 70\%$ is obtained for $p = 64$, while for $p = 124$, the total efficiency drops to $\approx 45\%$. It can be seen that the component for which the efficiency is - by far - the worst is the tree building phase, for which no measurable speedup is achieved for all tested values of p. This is due to two circumstances: For one, it is an effect of granularity: The tree building phase accounts in the two dimensional case (and for this choice of parameters) for a small fraction of the total cost, which is dominated by the potential evaluation in two dimensions.[20] However, the same argument can be made for the force evaluations via the FMM, which make for an equally small part of the total cost and scale considerably better. The second contribution is from the increased parallel overhead and sequential fraction associated with the tree building phase (cf. sec. 4.3).

It is also curious to note the comparably worse scaling of the direct algorithm, which seems very counter-intuitive initially. A likely explanation for this effect is that our implementation of the direct algorithm exploits the (anti)symmetry of the potential (force) kernel to save on half of the computations - theoretically - since this optimization requires the use of atomics to be applicable in the parallel setting, and it is this fact that we suspect of causing degradation of parallel performance seen in 26. Finally, in the direct setting the potential evaluations fare somewhat better due to increased granularity and less overhead (since more atomic operations are needed for vector operations).

[20] This is due to the logarithm involved in the potential, which turns out to be much more expensive than the square root operation in practice.

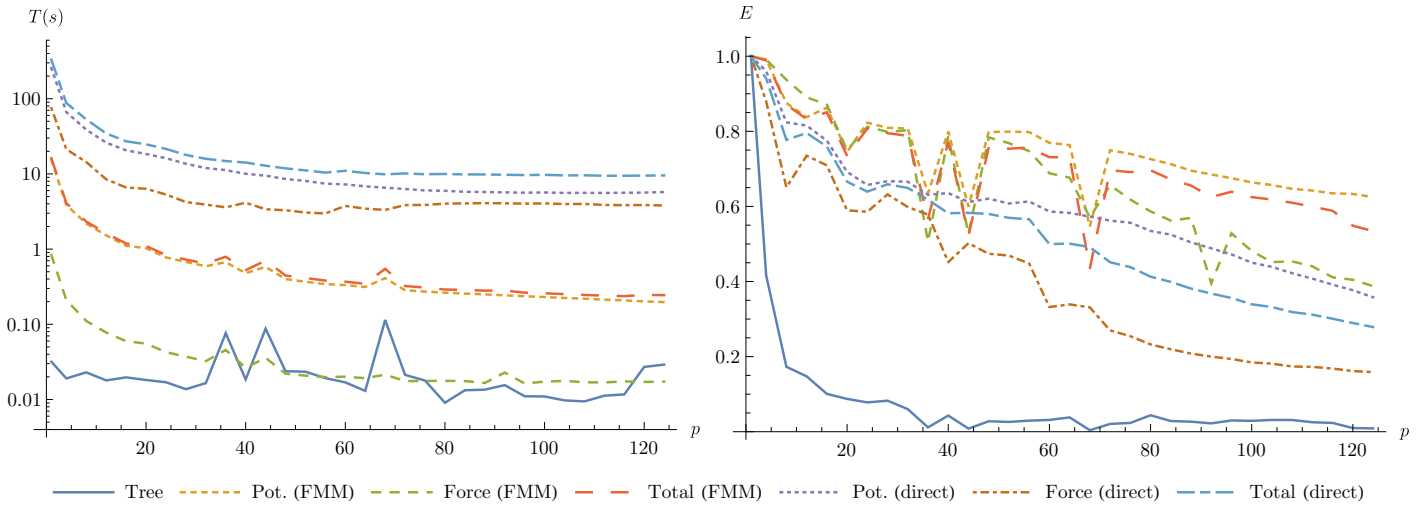

Figure 24: Runtimes and parallel efficiencies in two dimensions, non uniform data set & adaptive tree.

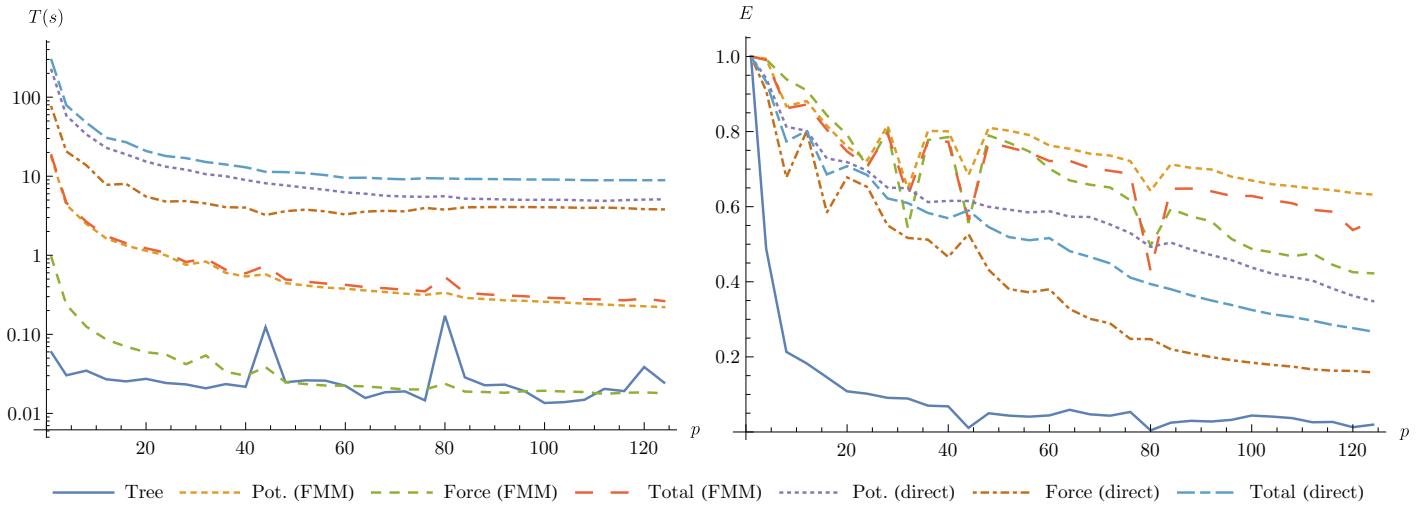

Figure 25: Runtimes and parallel efficiencies in two dimensions, non uniform data set & adaptive tree.

The parallel efficiencies observed with the adaptive algorithm are slightly better than those seen with the balanced algorithm for both uniform and non uniform data sets: While the efficiency at $p = 64$ is similar at $E \approx 70\%$, for $p = 124$ we have $E \approx 55\%$, i.e. approximately ten percent higher than for the balanced algorithm.

In three dimensions

Figures 26 to 28 show parallel efficiencies for similar configurations of the balanced and adaptive algorithm (with uniform and non uniform data sets), but now for three dimensions.

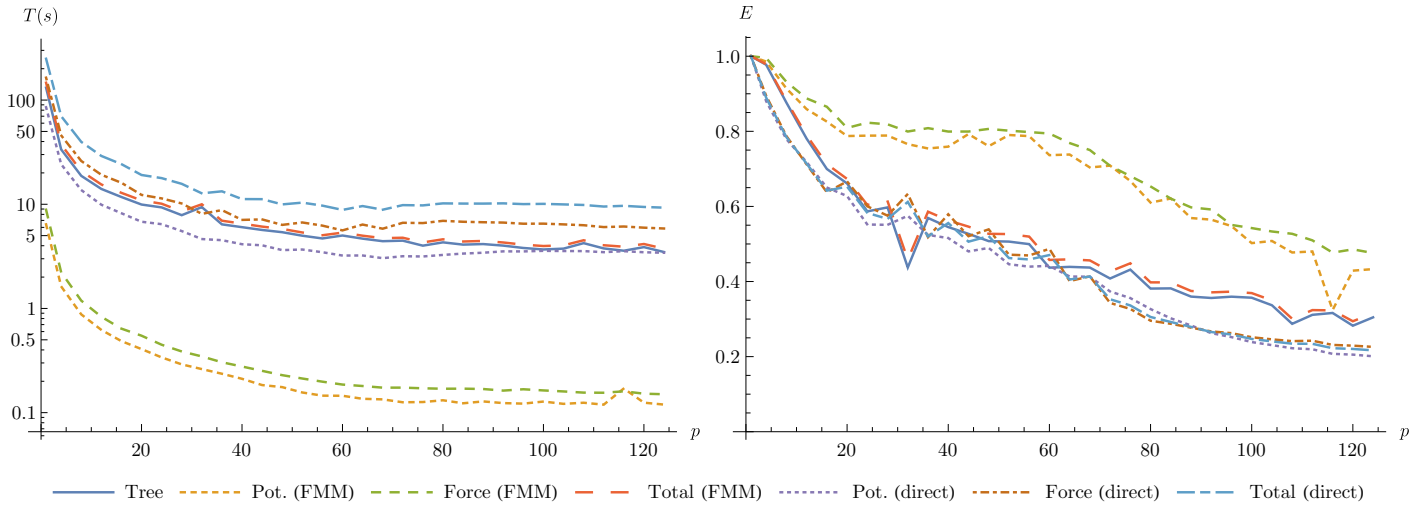

Figure 26: Parallel efficiencies in three dimensions, uniform data set & balanced tree. $N = 10^5$ particles, $\varepsilon = 10^{-3}$ and max. 1000 sources per leaf.

The most notable change seen in the three dimensional case is that the tree building phase dominates the total runtime, with potential and force evaluations taking approximately an order of magnitude less time. Fortunately, in this setting, the tree building phase scales better, even though still considerably worse than the force and potential evaluation phases alone. With the balanced algorithm and uniformly distributed sources, for $p = 64$, an efficiency of $\approx 45\%$ is reached while for $p = 128$, this has dropped to $\approx 30\%$.

For the direct algorithm, the situation is reversed w.r.t. the two dimensional case, since the force computation now involves more operations

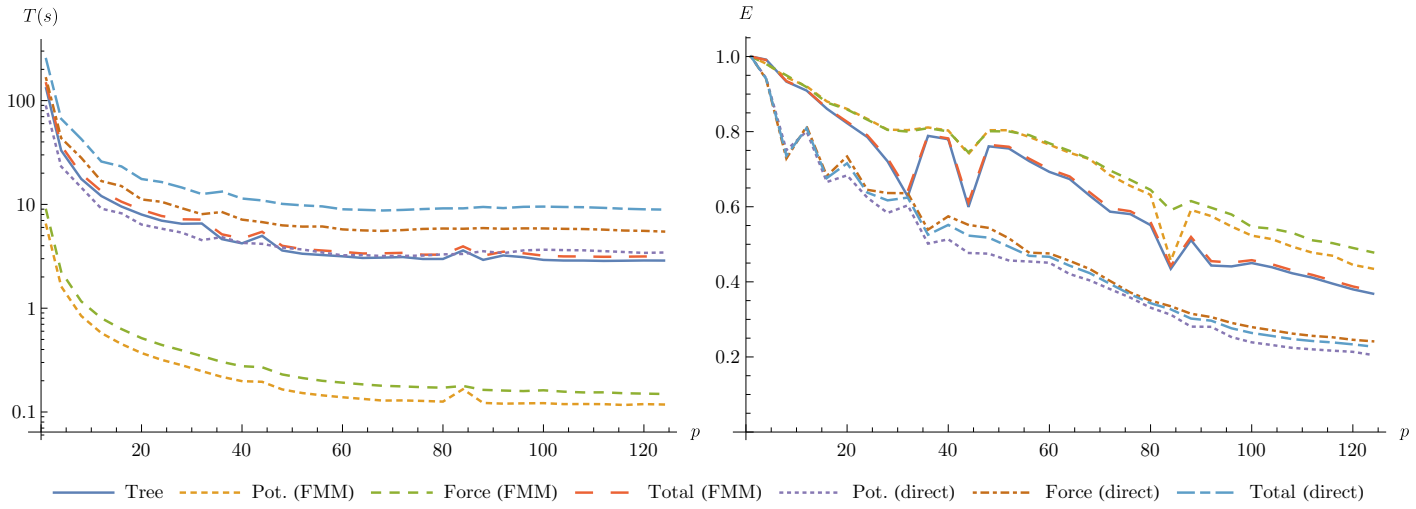

Figure 27: Parallel efficiencies in three dimensions, uniform data set & adaptive tree. $N = 10^5$ particles, $\varepsilon = 10^{-3}$ and max. 1000 sources per leaf.

The adaptive algorithm yields a similar picture as the balanced algorithm when sources are uniformly distributed, which is in line with expectations. Notably, however, it reaches better efficiencies, e.g. for $p = 64$, E

is slightly above 65%, while for $p = 124$, $E \approx 40\%$.

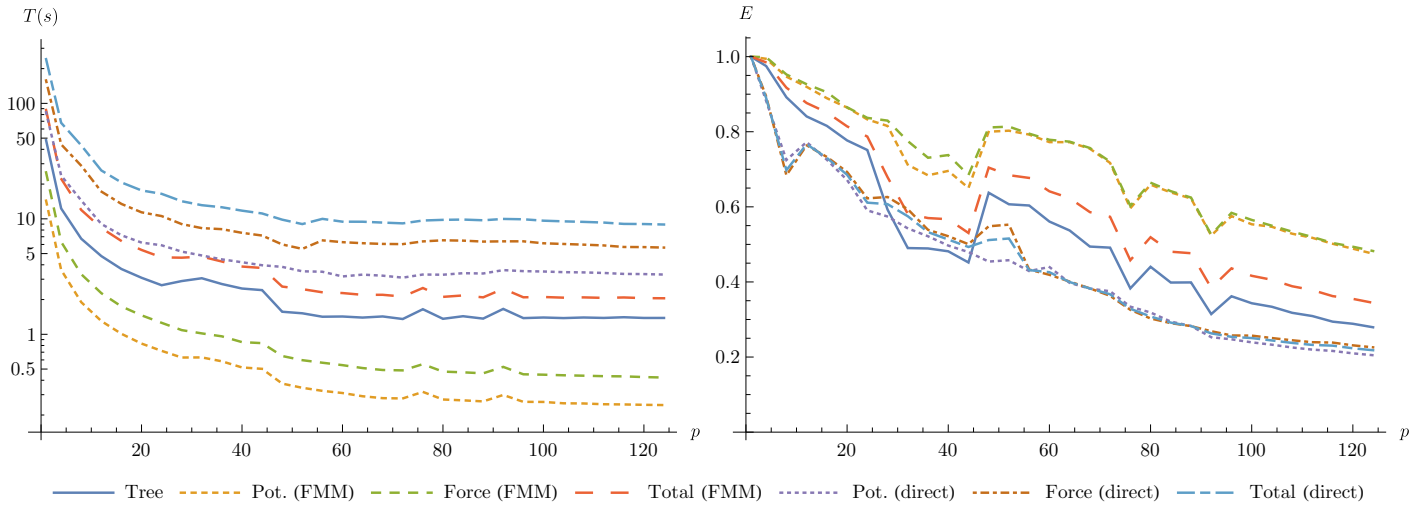

Figure 28: Parallel efficiencies in three dimensions, non-uniform data set & adaptive tree. $N = 10^5$ particles, $\varepsilon = 10^{-3}$ and max. 1000 sources per leaf.

For the non uniform data set, the difference in runtime between the tree building and the evaluation phases is less pronounced. The overall efficiency of the adaptive FMM is only slightly lower than in uniform case, with $E \approx 60\%$ for $p = 64$ and $E \approx 35\%$ for $p = 124$.

7 Conclusion and Further Work

We have described both the mathematical and the algorithmic aspects of the fast multipole method in the variants introduced in [2, 4, 5] and [6] and presented the architecture and performance of our implementation.

Notably, we have found that the two dimensional FMM performs very well when compared with the direct algorithm in the sequential setting, which aligns with the observations presented in [2] and [6]. On the other hand, we found the picture to be more varied in the three dimensional case, where the cutoff point at which our implementation provides a runtime advantage over the direct algorithm is past $N = 10^5$.

Regarding the accuracy of our implementation, we have shown that the analytical error bounds are well satisfied as long as $\varepsilon \gtrapprox 10^{-7}$, after which violations can be expected. The parallel efficiency of our implementation is adequate for small numbers of processors, but certainly improvable for larger systems.

Several avenues for further work exist. To improve the sequential performance of the three dimensional algorithm, implementing the acceleration schemes discussed e.g. in [1, 3] is a viable option. The phenomenon of the reached accuracy stalling beyond certain values of ε certainly deserves deeper investigation aimed at finding and - if possible - eliminating its causes. At last, it would be worthwhile to determine the origin of the insufficient parallel efficiency, e.g. by thorough profiling. It would also be interesting to investigate workload decomposition schemes different from those intrinsic to `OpenMP`'s loop directives, e.g. task based formulations.

References

[1] Rick Beatson and Leslie Greengard. "A short course on fast multipole methods". In: *Wavelets, multilevel methods and elliptic PDEs* 1 (1997), pp. 1–37.

[2] Leslie Greengard and Vladimir Rokhlin. "A fast algorithm for particle simulations". In: *Journal of computational physics* 73.2 (1987), pp. 325–348.

[3] Hongwei Cheng, Leslie Greengard, and Vladimir Rokhlin. "A fast adaptive multipole algorithm in three dimensions". In: *Journal of computational physics* 155.2 (1999), pp. 468–498.

[4] Leslie Greengard. *The rapid evaluation of potential fields in particle systems*. MIT press, 1988.

[5] L Grengard and Vladimir Rokhlin. "The rapid evaluation of potential fields in three dimensions". In: *Vortex Methods*. Springer, 1988, pp. 121–141.

[6] J Carrier, Leslie Greengard, and Vladimir Rokhlin. "A fast adaptive multipole algorithm for particle simulations". In: *SIAM journal on scientific and statistical computing* 9.4 (1988), pp. 669–686.

[7] Susanne Pfalzner and Paul Gibbon. *Many-body tree methods in physics*. Cambridge University Press, 2005.

[8] Jakub Kurzak and Bernard M Pettitt. "Fast multipole methods for particle dynamics". In: *Molecular simulation* 32.10-11 (2006), pp. 775–790.

[9] Josh Barnes and Piet Hut. "A hierarchical O (N log N) force-calculation algorithm". In: *nature* 324.6096 (1986), p. 446.

[10] R Capuzzo-Dolcetta and P Miocchi. "A comparison between the fast multipole algorithm and the tree-code to evaluate gravitational forces in 3-D". In: *Journal of Computational Physics* 143.1 (1998), pp. 29–48.

[11] Kevin E Schmidt and Michael A Lee. "Implementing the fast multipole method in three dimensions". In: *Journal of Statistical Physics* 63.5-6 (1991), pp. 1223–1235.

[12] Leslie Greengard and William D Gropp. "A parallel version of the fast multipole method". In: *Computers & Mathematics with Applications* 20.7 (1990), pp. 63–71.

[13] Eric Darve, Cris Cecka, and Toru Takahashi. "The fast multipole method on parallel clusters, multicore processors, and graphics processing units". In: *Comptes Rendus Mecanique* 339.2-3 (2011), pp. 185–193.

[14] Felipe A Cruz, Matthew G Knepley, and Lorena A Barba. "PetFMM—A dynamically load-balancing parallel fast multipole library". In: *International journal for numerical methods in engineering* 85.4 (2011), pp. 403–428.

[15] Olivier Coulaud, Pierre Fortin, and Jean Roman. "Hybrid MPI-thread parallelization of the fast multipole method". In: *Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07)*. IEEE. 2007, pp. 52–52.

[16] Michael S Warren and John K Salmon. "A parallel hashed oct-tree n-body algorithm". In: *Supercomputing'93: Proceedings of the 1993 ACM/IEEE conference on Supercomputing*. IEEE. 1993, pp. 12–21.

[17] John David Jackson. *Classical electrodynamics*. 1999.

[18] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Vol. 55. Courier Corporation, 1965.

[19] Srinivas Aluru. "Greengard's N-body algorithm is not order N". In: *SIAM Journal on Scientific Computing* 17.3 (1996), pp. 773–776.

[20] E Athanassoula et al. "Optimal softening for force calculations in collisionless N-body simulations". In: *Monthly Notices of the Royal Astronomical Society* 314.3 (2000), pp. 475–488.