Università della Svizzera italiana

Efficient Computational Algorithms 2021

Final Project Report

Fast Multipole Methods

Abstract

The Fast Multipole Method is a numerical technique developed to decrease the time complexity of calculating long-ranged forces in N-body simulation problems. In contrast to typical N-body problem solvers, which require $\mathcal{O}(N^2)$ operations, the Fast Multipole Method runs in linear time, without severely compromising accuracy. There are several other applications where the Fast Multipole Method reduces the complexity to $\mathcal{O}(N)$, such as matrix-vector multiplication, LU-factorization or acceleration of the iterative solver in the method of moments. Within this report, we introduce the Fast Multipole Method for the gravitational N-body problem from a theoretical point of view and implement a 2D spiral galaxy simulation using the Fast Multipole Method with adaptive trees. The initial build of the tree requires a time of $\mathcal{O}(N \log N)$ while for the main Fast Multipole Method we achieve linear complexity $\mathcal{O}(N)$.

Student 1
Daniel Bauer

Student 2
Glejdis Shkëmbi

Student 3
Michael Zikeli

1 Introduction

The Fast Multipole Method (FMM) was first introduced by Greengard and Rokhlin in 1987 for the N-body simulation problem with pairwise interactions, such as elasticity, gravitation, electrostatics, or wave propagation [4, 8]. N-body problems typically have a complexity of $\mathcal{O}(N^2)$ [9]. The FMM solves this problem by introducing an approximation of the driving force term, achieving linear complexity [8]. Moreover, the FMM uses quad-trees or oct-trees to hierarchically split the computing domain and thus is frequently referred to as a "tree code" algorithm. The tree structure makes this algorithm well-suited for multi-core and parallel computing systems [8]. This report first sheds some light on the inner workings of the FMM by providing in depth details on the steps of the algorithm and analyzing its time complexity. Afterwards, the operators used by the algorithm are introduced and a rough analyis of the approximation error is given. Lastly, details of the implementation of the rotating galaxy simulation are shown, as well as the obtained results. After reading the report, the reader should comprehend the algorithm's underlying concepts and understand how the linear time complexity is achieved.
In this paper we consider N particles p_i placed at positions $\boldsymbol{x}_i \in \Omega \subset \mathbb{R}^2$, where Ω is a rectangular domain. For the ease of notation, we think of $\boldsymbol{x}_i$ as a point in the complex plane, where $\Re(\boldsymbol{x})$ is the real part of $\boldsymbol{x}$. Each of these particles induces a long-range potential $\Psi_i(\boldsymbol{x}) = m_i \log(\boldsymbol{x} - \boldsymbol{x}_i)$, where m_i is the mass of the source particle p_i. The aim is to obtain the net force $\boldsymbol{f}_j$ acting on the target particle p_j, which is the sum of all pairwise interactions [1]:

$$\boldsymbol{f}_j = \sum_{i=1, i\neq j}^{N} \boldsymbol{F}_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j, m_i m_j) = -\sum_{i=1, i\neq j}^{N} m_j \nabla \Re(\Psi_i(\boldsymbol{x}_j)) \tag{1}$$

2 The Algorithm

This section focuses on explaining the FMM algorithm by introducing the underlying data structure and giving a step by step introduction of its necessary operations. The general idea behind the algorithm is to reduce the number of pairwise interactions by introducing approximations representing entire "clusters" of interactions. By doing so, each of the N target particles p_j immediately interacts with only a few close particles, for all other "distant" interactions, the approximations are considered.

2.1 A Tree of Boxes, Neighbors and Interaction Lists

To achieve linear complexity, the FMM utilizes a tree based structure, in our 2D case a quad-tree. The root node of the tree corresponds to the domain Ω. From the root, the domain is recursively subdivided in four equalsized nodes until every node contains less than s particles. Nodes that do not need further refinement are called the leafs or leaf nodes of the tree [8].
Given two distinct nodes σ, τ, we define the following [8]:

The parent of τ is the node on the next coarser level that contains τ.

The children of τ compose the set $\mathcal{L}_\tau^{\text{child}}$ of nodes with τ as their parent.

The neighbors of τ compose the set $\mathcal{L}_\tau^{\text{nei}}$ of nodes on the same level that directly touch τ.

The interaction list of τ is the set $\mathcal{L}_\tau^{\text{int}}$ of all nodes σ such that σ and τ are on the same level, and do not touch each other, but their parents touch each other.

σ **is well-separated** from τ if they are on the same level and not neighbors.

2.2 The Fast Multipole Method

In the following, we elaborate on the main steps of the algorithm as described by Martinsson in [8], the mathematical details are given in Section 3.

0. **Construct the tree and all interaction lists.**
   Before calculating any forces, we need to build the tree data structure and determine the interaction lists, as described above. This has complexity $\mathcal{O}(N \log N)$, however, it needs to be done only once during the initialization. Hence, for a sufficiently long simulation, this initial effort can be neglected.

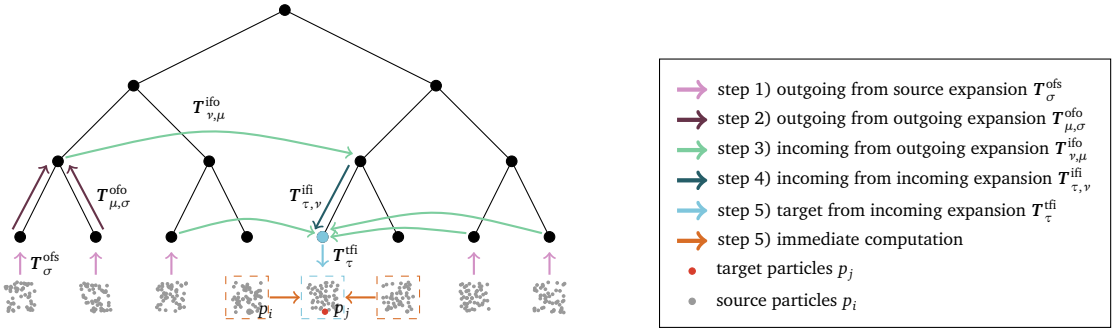


Figure 1. Flow of information through the tree (1D case)

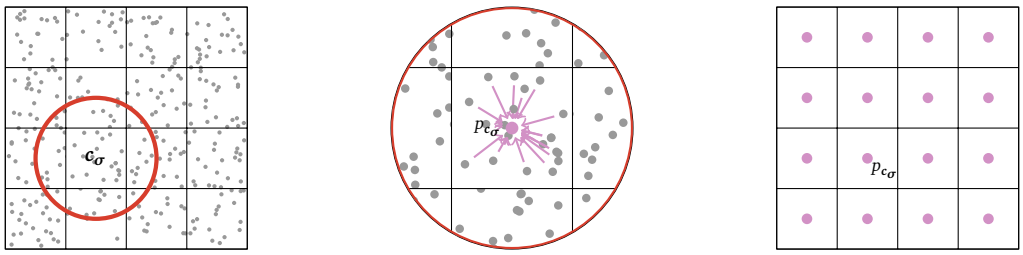


Figure 2. Step 1 of FMM

Figure 1 shows how the information of all source particles p_i propagates to the target particles p_j. The differently colored arrows in this figure represent the different steps of the algorithm which are described below.

1. **Outgoing from source expansion T^{ofs}_{σ}**
   Within each leaf node σ, the potentials of all source particles p_i are approximated at the center $\boldsymbol{c_\sigma}$ of the node. These approximations are then summed up to a "virtual pole" p_{c_σ} as represented in Figures 2a and 2b. This representation of the potentials is called outgoing expansion and obtained by applying the operator T^{ofs}_{σ} to the particles p_i. As will become clear in Section 3, the outgoing expansion of σ is valid only in nodes which are well-separated from σ. Since every particle contributes to exactly one outgoing expansion, the complexity of step 1 is $\mathcal{O}(N)$.

2. **Outgoing from outgoing expansion $T^{\text{ofo}}_{\mu,\sigma}$**
   After approximating p_{c_σ} for all leaf nodes σ, the information must be distributed throughout the tree. To this end, all child nodes σ pass their information to their parent node μ to determine its outgoing expansion. Just like before, the outgoing expansion is an approximation to the potential caused by the particles inside μ, which is valid only in well-separated nodes. To avoid considering every particle again, the outgoing expansion of μ is obtained by translating the expansions of its children to the center of μ via the operator $T^{\text{ofo}}_{\mu,\sigma}$. By summing the translated expansions, we obtain the ougoing expansion of μ at the new "virtual pole" p_{c_μ}, compare Figure 3. For the approximated potential to be present at each node, this step is recursively repeated up to the root node. Steps 1 and 2 are therefore known as the upward-pass. As the total number of nodes in the tree is of order N, step 2 has complexity $\mathcal{O}(N)$.

3. **Incoming from outgoing expansion $T^{\text{ifo}}_{\nu,\mu}$**
   After the upward-pass, every node μ in the tree contains approximations to the potentials caused by the particles located inside μ. To calculate the interactions between particles, this information needs to be spread to the other nodes. For this, each node gathers the expansions from the nodes in its interaction list. More precisely, by applying the operator $T^{\text{ifo}}_{\nu,\mu}$ to the outgoing expansions of nodes in $\mathcal{L}^{\text{int}}_{\nu}$ and summing the results, the incoming expansion of ν is obtained. In contrast to the outgoing expansion, the incoming expansion approximates the potentials caused by particles located outside of ν. As mentioned above, the ougoing expansions of the neighbors of ν are not valid inside ν, hence, only well-separated nodes are considered. Likewise, interactions with nodes beyond the interaction list can be ignored since they are handled on the coarser levels (compare

Figure 3. Step 2 of FMM. The yellow highlighted area is well-separated from μ.

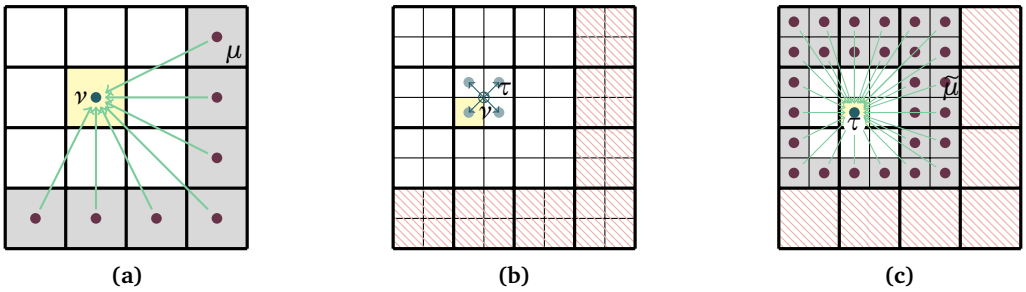


Figure 4. Steps 3 and 4 of FMM. The interaction lists of the nodes ν and τ are highlighted in gray.

Figures 4a and 4c). Step 3 also has a complexity of $\mathcal{O}(N)$ because a constant amount of work is performed for all $\mathcal{O}(N)$ nodes.

4. **Incoming from incoming expansion $T^{\text{ifi}}_{\tau,\nu}$**
   After step 3, an incoming expansion is present at each node ν for each level. This only considers the nodes contained in the interaction list $\mathcal{L}^{\text{int}}_{\nu}$ of ν. What is left, is to pass the potential information, gathered at each refinement level, down the tree up to the leaf nodes. Together with step 3 this is called the downward-pass. Complementary to $T^{\text{ofo}}_{\mu,\sigma}$ in step 2, the potential information of the parent node ν is passed down to its children $\mathcal{L}^{\text{child}}_{\nu}$ and approximated at their respective centers $\mathbf{c}_{\tau}$ (compare Figure 4b). Accordingly, step 4 consists of performing constant amount of work for each node of the tree, so its time complexity is $\mathcal{O}(N)$.

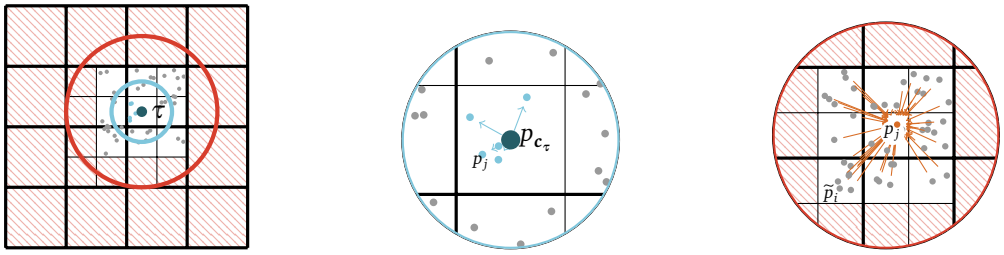
   Now it becomes clear why in step 3 we considered only nodes which are in the interaction list of ν. As can be seen from Figure 4c, at each refinement level there are some well-separated nodes, that were not well-separated on coarser levels, and therefore not considered yet. These missing approximations to the potentials from the nodes $\tilde{\mu}$ are added to the incoming expansion of τ during the downward-pass. Consequently, on the finer level only the "new" well-separated nodes $\tilde{\mu}$ which were not already considered on the coarser levels are elements of $\mathcal{L}^{\text{int}}_{\tau}$.

5. **Target from incoming expansion T^{tfi}_{τ}**

   In the last step, the resulting potential approximations are applied to the target particles p_j. For each leaf node τ, the incoming expansion is evaluated at the locations of the target particles p_j and the resulting forces are calculated. This operation is denoted by T^{tfi}_{τ} and represented in Figure 5b. The complexity of this step is $\mathcal{O}(N)$ as it involves constant work for each particle.

   Until now, only the well-separated nodes on all refinement-levels were considered. However, as can be seen from Figure 5a, there are also not well-separated source particles p_i that influence the target particles p_j, i.e. particles in the neighbors $\mathcal{L}^{\text{nei}}_{\tau}$ of the current node τ, as well as the node τ itself. These source particles are too close to the targets p_j so that their potentials can not be approximated without introducing overly large errors. Therefore, they are handled in a direct manner using Equation (1) as illustrated in Figure 5c. Since the total number of direct source particles p_i is bounded by the constant $9s$ for each target particle p_j, the complexity of the direct computation is $\mathcal{O}(N)$ as well.

As discussed, all steps of the FMM have linear time complexity so the runtime of the FMM is in $\mathcal{O}(N)$.



(a) (b) (c)

Figure 5. Step 5 of FMM

3 Operators

The previous section introduced the FMM without going into details about the operators which are used to determine and apply the expansions. This section supplies the missing details.

3.1 Outgoing from Sources

In step 1 of the FMM, we seek to find a fixed size representation of the potential caused by all the particles at positions $\boldsymbol{x}_i$ in a source box σ centered around $\boldsymbol{c}_\sigma$. We can find this representation by summing the potentials of all particles in σ and rearranging [9, 8]:

$$\Psi(\boldsymbol{x}) = \sum_i \Psi_i(\boldsymbol{x}) = \sum_i m_i \log(\boldsymbol{x} - \boldsymbol{x}_i) \tag{2}$$

$$= \sum_i m_i \log((\boldsymbol{x} - \boldsymbol{c}_\sigma) - (\boldsymbol{x}_i - \boldsymbol{c}_\sigma)) = \sum_i m_i \log\left((\boldsymbol{x} - \boldsymbol{c}_\sigma) \cdot \left(1 - \frac{\boldsymbol{x}_i - \boldsymbol{c}_\sigma}{\boldsymbol{x} - \boldsymbol{c}_\sigma}\right)\right) \tag{3}$$

$$= \sum_i m_i \left(\log(\boldsymbol{x} - \boldsymbol{c}_\sigma) + \log\left(1 - \frac{\boldsymbol{x}_i - \boldsymbol{c}_\sigma}{\boldsymbol{x} - \boldsymbol{c}_\sigma}\right)\right) = \sum_i m_i \left(\log(\boldsymbol{x} - \boldsymbol{c}_\sigma) - \sum_{k=1}^{\infty} \frac{1}{k} \frac{(\boldsymbol{x}_i - \boldsymbol{c}_\sigma)^k}{(\boldsymbol{x} - \boldsymbol{c}_\sigma)^k}\right) \tag{4}$$

$$= \hat{\boldsymbol{q}}_0^\sigma \log(\boldsymbol{x} - \boldsymbol{c}_\sigma) + \sum_{k=1}^{\infty} \frac{\hat{\boldsymbol{q}}_k^\sigma}{(\boldsymbol{x} - \boldsymbol{c}_\sigma)^k} \tag{5}$$

Where in Equation (4) we do a Taylor expansion around $\frac{\boldsymbol{x}_i - \boldsymbol{c}_\sigma}{\boldsymbol{x} - \boldsymbol{c}_\sigma} = 0$.
Now, if we truncate the series in Equation (5) to $P-1$ terms, we obtain the outgoing expansion, i.e. P complex valued coefficients [8]:

$$\begin{aligned} \hat{\boldsymbol{q}}_0^\sigma &= \sum_i m_i \\ \hat{\boldsymbol{q}}_k^\sigma &= -\sum_i \frac{m_i}{k} (\boldsymbol{x}_i - \boldsymbol{c}_\sigma)^k, \qquad k = 1 \ldots P-1 \end{aligned} \tag{6}$$

The operator that determines the outgoing expansion from the source particles is denoted by $\boldsymbol{T}_\sigma^{\text{ofs}}$ [8]. Note that the series in Equation (5) only converges if $|\boldsymbol{x}_i - \boldsymbol{c}_\sigma| < |\boldsymbol{x} - \boldsymbol{c}_\sigma|$, which is guaranteed if the boxes of $\boldsymbol{x}_i$ and $\boldsymbol{x}$ are well-seperated but not if they are neighboring.

3.2 Outgoing from Outgoing

In Section 2 we efficiently calculate the outgoing expansion of a parent box μ from the expansions of its children using the outgoing-from-outgoing translation operator $\boldsymbol{T}_{\mu,\sigma}^{\text{ofo}}$. This operator translates the expansions which are centered around the center of the children $\boldsymbol{c}_\sigma$ to the center of the parent $\boldsymbol{c}_\mu$. It can be derived by rewriting the following two terms in Equation (5) [1, 8]:

$$\log(\boldsymbol{x} - \boldsymbol{c}_\sigma) = \log((\boldsymbol{x} - \boldsymbol{c}_\mu) - (\boldsymbol{c}_\sigma - \boldsymbol{c}_\mu)) = \log(\boldsymbol{x} - \boldsymbol{c}_\mu) - \sum_{k=1}^{\infty} \frac{1}{k} \frac{(\boldsymbol{c}_\sigma - \boldsymbol{c}_\mu)^k}{(\boldsymbol{x} - \boldsymbol{c}_\mu)^k} \tag{7}$$

$$(\boldsymbol{x} - \boldsymbol{c}_\sigma)^{-k} = ((\boldsymbol{x} - \boldsymbol{c}_\mu) - (\boldsymbol{c}_\sigma - \boldsymbol{c}_\mu))^{-k} = \sum_{l=k}^{\infty} \binom{l-1}{k-1} \frac{(\boldsymbol{c}_\sigma - \boldsymbol{c}_\mu)^{l-k}}{(\boldsymbol{x} - \boldsymbol{c}_\mu)^l} \tag{8}$$

Truncating the series and rearranging the terms results in the translated outgoing expansion [1, 8]

$$\hat{\boldsymbol{q}}_0^{\mu}\log(\boldsymbol{x}-\boldsymbol{c}_{\mu})+\sum_{k=1}^{P-1}\frac{\hat{\boldsymbol{q}}_k^{\mu}}{(\boldsymbol{x}-\boldsymbol{c}_{\mu})^k} \tag{9}$$

with

$$\begin{aligned}\hat{\boldsymbol{q}}_0^{\mu} &= \hat{\boldsymbol{q}}_0^{\sigma}\\ \hat{\boldsymbol{q}}_k^{\mu} &= -\hat{\boldsymbol{q}}_0^{\sigma}\frac{1}{k}\left(\boldsymbol{c}_{\sigma}-\boldsymbol{c}_{\mu}\right)^k+\sum_{j=1}^{k}\hat{\boldsymbol{q}}_j^{\sigma}\binom{k-1}{j-1}\left(\boldsymbol{c}_{\sigma}-\boldsymbol{c}_{\mu}\right)^{k-j}, \qquad k=1\ldots P-1\end{aligned} \tag{10}$$

3.3 Incoming from Outgoing

The outgoing expansion is a compact, approximative representation of the potential caused by particles located inside a disk which is valid everywhere outside of this disk. Additionally, we need a representation of the potential caused by particles located outside of a disk which is valid inside this disk. This representation is called incoming expansion and can be calculated from an outgoing expansion via the incoming-from-outgoing operator $\boldsymbol{T}^{\text{ifo}}_{\nu,\mu}$ by introducing the following substitutions to Equation (9) [1, 8]:

$$\log(\boldsymbol{x}-\boldsymbol{c}_{\mu})=\log((\boldsymbol{x}-\boldsymbol{c}_{\nu})-(\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu}))=\log(\boldsymbol{c}_{\nu}-\boldsymbol{c}_{\mu})-\sum_{k=1}^{\infty}\frac{1}{k}\frac{(\boldsymbol{x}-\boldsymbol{c}_{\nu})^k}{(\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu})^k} \tag{11}$$

$$(\boldsymbol{x}-\boldsymbol{c}_{\mu})^{-k}=((\boldsymbol{x}-\boldsymbol{c}_{\nu})-(\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu}))^{-k}=\frac{1}{(\boldsymbol{c}_{\nu}-\boldsymbol{c}_{\mu})^k}\sum_{l=0}^{\infty}\binom{l+k-1}{k-1}\frac{(\boldsymbol{x}-\boldsymbol{c}_{\nu})^l}{(\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu})^l} \tag{12}$$

Note that with these substitutions, $\boldsymbol{x}$ appears in the numerator of the fraction, before it was in the denominator. This means that the series converges if $|\boldsymbol{x}-\boldsymbol{c}_{\nu}|<|\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu}|$, i.e. if $\boldsymbol{x}$ is closer to the center of the incoming expansion $\boldsymbol{c}_{\nu}$ than is the center of the outgoing expansion $\boldsymbol{c}_{\mu}$. Accordingly, we obtain the incoming expansion of the form [1, 8]

$$\sum_{k=0}^{P-1}\hat{\boldsymbol{u}}_k^{\nu}(\boldsymbol{x}-\boldsymbol{c}_{\nu})^k \tag{13}$$

with

$$\begin{aligned}\hat{\boldsymbol{u}}_0^{\nu} &= \hat{\boldsymbol{q}}_0^{\mu}\log(\boldsymbol{c}_{\nu}-\boldsymbol{c}_{\mu})+\sum_{j=1}^{\infty}\hat{\boldsymbol{q}}_j^{\mu}(-1)^j\frac{1}{(\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu})^j}\\ \hat{\boldsymbol{u}}_k^{\nu} &= -\hat{\boldsymbol{q}}_0^{\mu}\frac{1}{k(\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu})^k}+\sum_{j=1}^{\infty}\hat{\boldsymbol{q}}_j^{\mu}(-1)^j\binom{k+j-1}{j-1}\frac{1}{(\boldsymbol{c}_{\mu}-\boldsymbol{c}_{\nu})^{k+j}}, \qquad k=1\ldots P-1\end{aligned} \tag{14}$$

3.4 Incoming from Incoming

Similar to the outgoing expansion, the incoming expansion can be translated to a new center $\boldsymbol{c}_{\tau}$. The new coefficients result from the incoming-from-incoming translation operator $\boldsymbol{T}^{\text{ifi}}_{\tau,\nu}$ as follows [8]:

$$\hat{\boldsymbol{u}}_k^{\tau}=\sum_{j=k}^{\infty}\hat{\boldsymbol{u}}_j^{\nu}\binom{j}{k}(\boldsymbol{c}_{\tau}-\boldsymbol{c}_{\nu})^{j-k}, \qquad k=1\ldots P-1 \tag{15}$$

3.5 Targets from Incoming

The incoming expansion of a node τ represents a potential which exerts forces on the particles in τ. In the last step of the FMM, these forces are determined by means of the targets-from-incoming operator $\boldsymbol{T}^{\text{tfi}}_{\tau}$. For this, Equation (1) is used, where the gradient of the potential is determined from Equation (13) as follows [1]:

$$\nabla\Re(\Psi(\boldsymbol{x}))=\left(\Re\left(\frac{\partial\Psi}{\partial\boldsymbol{x}}\right),-\Im\left(\frac{\partial\Psi}{\partial\boldsymbol{x}}\right)\right)^T \qquad \text{where} \qquad \left(\frac{\partial\Psi}{\partial\boldsymbol{x}}\right)=\sum_{k=1}^{P-1}\hat{\boldsymbol{u}}_k^{\tau}k(\boldsymbol{x}-\boldsymbol{c}_{\tau})^{k-1} \tag{16}$$

3.6 Error Analysis

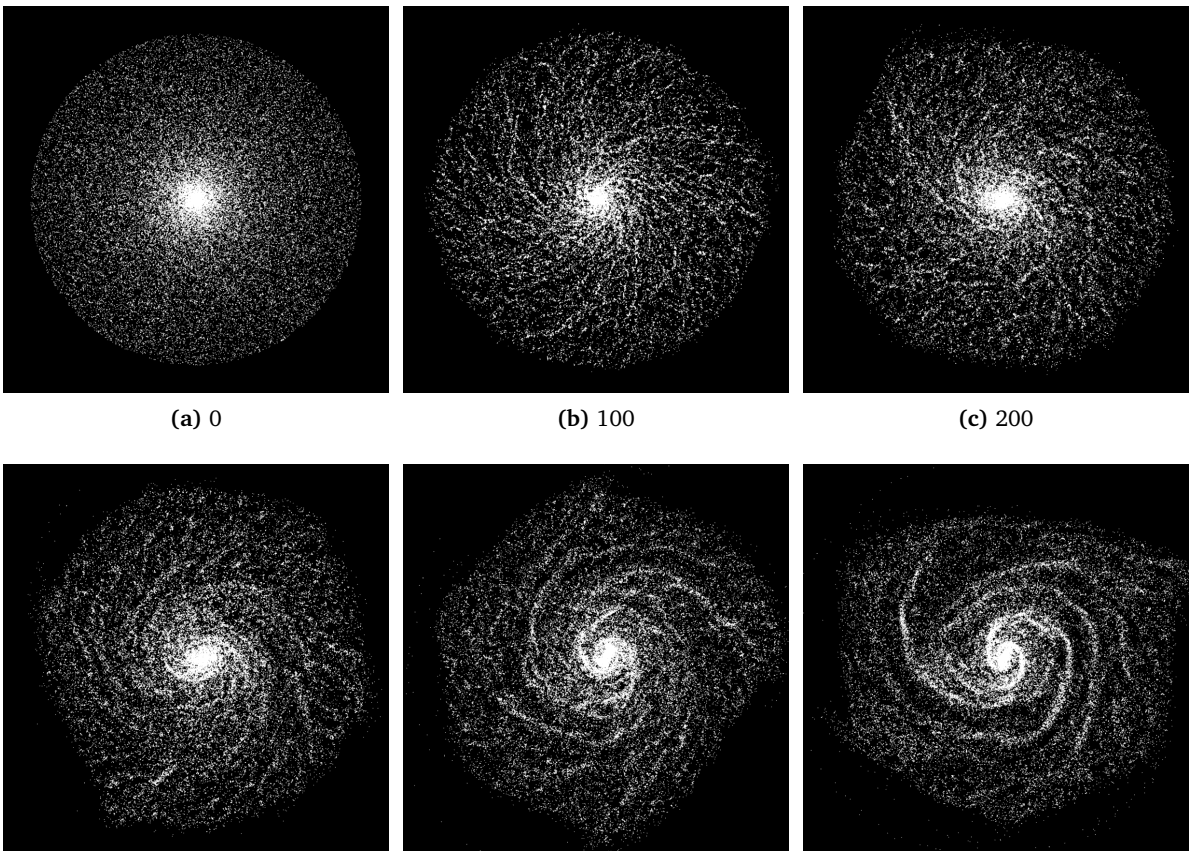
Due to the fact that all expansions are reduced to P terms, the potentials estimated by the FMM are not precise. In most cases, the global error is similar to the worst-case local truncation error [2]. In other words, it scales as α^P where $\alpha = \frac{\sqrt{2}}{4-\sqrt{2}} \approx 0.5469$. Therefore, in order to achieve a given tolerance ϵ, it should hold that $P \approx \frac{\log(\epsilon)}{\log(\alpha)}$ [8]. Moreover, if we assume that each leaf holds $\mathcal{O}(P)$ sources, the asymptotic complexity of the 2D FMM is $\mathcal{O}(PN)$. As a result, the overall complexity scales as $\mathcal{O}(\log\left(\frac{1}{\epsilon}\right)N)$ as $\epsilon \to 0$ and $N \to \infty$ [8].

4 Implementation

In this section we introduce a few points that we deem helpful for implementing the FMM.
First, the particles might be distributed very unevenly inside the computational domain, e.g. in the case of colliding galaxies. In these cases building the tree up to a fixed level, ignoring the particle distribution, results in a tree where either the majority of the particles is located in a small subset of the leaf nodes or the majority of leaf nodes is empty. Both cases lead to non-linear complexity of the force calculation. Therefore, it is important to subdivide boxes exactly if they contain more than a fixed number of particles. The resulting tree consists of leaf boxes of varying sizes and is commonly called an adaptive tree [1].
Since adaptive trees contain leaf boxes of different sizes, neighboring leaf boxes might reside on different levels. Stated differently, neighbors of leaf boxes might be inner nodes. Remember that forces between neighboring leaf boxes must be calculated in a direct fashion. In the case of adaptive trees, this is also true for forces acting between particles in the large leaf box and particles in all leaf boxes of the neighboring inner node.
Building the tree has complexity $\mathcal{O}(N \log N)$. To ensure that this cost is negligible compared to the force calculations, it is important to build the tree only once at the start of the simulation. However, in each time step every particle changes its position, and in consequence might move to a different leaf box. Therefore, the tree has to be repaired. This means moving particles to the correct node in the tree and minimizing the tree afterwards, i.e. deleting empty leaf boxes and merging leaf boxes which contain only a small number of particles [6].
Lastly, obtaining the interaction lists can be quite painful [9]. Our approach is to store links to the direct neighbors of each node in the tree data structure. The corresponding code is easy to separate from the main algorithm and easy to test. Thanks to the neighbor links it is straightforward to loop over all nodes in the interaction list of a child of a particular node, i.e. the interaction list of any node except the root node. As the interaction list of the root node is empty, we can safely ignore it.

5 Results

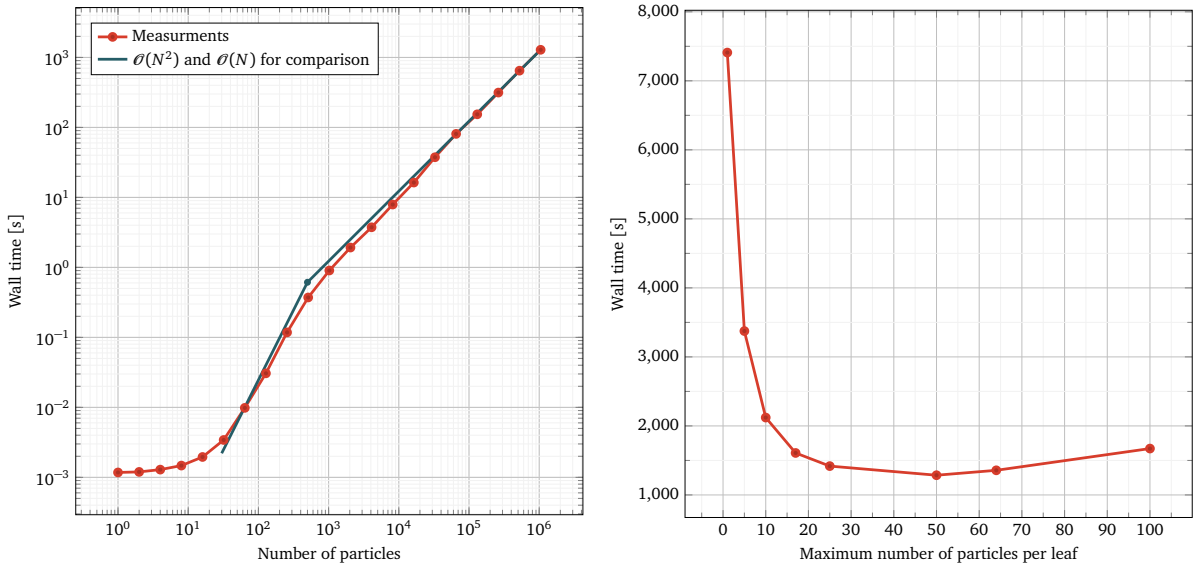
We test our implementation by simulating a rotating galaxy. To this end, we place $N = 40\,000$ particles at random positions inside a disk with radius 1.25 in the center of a square domain of size 50×50. The positions are sampled uniformly in polar coordinates which results in a higher particle density in the center of the galaxy. All particles have the mass $m = 1/N$ and an initial velocity in tangential direction with magnitude $0.9\sqrt{r}$, where r is the distance of the particle to the center of the disk. This results in a rotating disk of particles such that gravitational and centripetal forces roughly balance each other. Furthermore, we use the velocity-Verlet algorithm [7] with time steps of size $\mathrm{dt} = 0.01$, expansions with $P = 5$ and store up to $s = 64$ particles per leaf. Figure 6 shows the distribution of the stars after different number of time steps. We observe that patterns in the particle distribution evolve which eventually form the typical spiral arms.
In Figure 7a you can see the scaling behavior of the FMM for our galaxy example with maximal 50 particles per leaf node. For easier interpretation we also show how quadratic and linear scaling would look like. For small number of particles ($N < 500$) the algorithm scales approximately quadratically. This is to be expected since all particles could fit into ten leaves which means that almost all interactions are calculated directly. On the other hand, we see that the complexity approaches $\mathcal{O}(N)$ for larger N.
Furthermore, we investigate the effect that the maximum number s of particles per leaf has on the runtime. Figure 7b collects benchmark results for $N = 2^{20} \approx 10^6$. We observe the shortest runtimes with values of s between 25 and 64. The optimum among the tested values is $s = 50$, which reduces the runtime by more than a factor of five compared to $s = 1$.



(a) 0 **(b)** 100 **(c)** 200

(d) 300 **(e)** 400 **(f)** 500

Figure 6. Results of the galaxy simulation at different time steps.



(a) Scaling behavior with maximal 50 particles per leaf

(b) Runtime for varying number of particles per leaf with $N = 2^{20}$

Figure 7. Measured runtime with varying number of particles and maximum particles per leaf node

References

[1] R. Beatson and L. Greengard. *A short course on fast multipole methods*, pages 1–37. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997.

[2] E. Darve. The fast multipole method i: Error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis*, 38, 06 2000.

[3] J. Dongarra and F. Sullivan. Guest editors introduction to the top 10 algorithms. *Computing in Science & Engineering*, 2:22–23, 02 2000.

[4] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[5] L. Greengard and V. Rokhlin. A new version of the Fast Multipole Method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997.

[6] M. Griebel, S. Knapek, and G. Zumbusch. *Numerical simulation in molecular dynamics, vol. 5 of Texts in Computational Science and Engineering*. Springer, Berlin, 2007.

[7] R. D. Groot and P. B. Warren. Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. 1997.

[8] P.-G. Martinsson. *Fast Multipole Methods*, pages 498–508. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[9] N. Siccha. The fast multipole method, 2016.

[10] R. Tanno, K. Arulkumaran, D. Alexander, A. Criminisi, and A. Nori. Adaptive neural trees. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6166–6175. PMLR, 09-15 Jun 2019.