A PROVABLY FAST MULTIPOLE METHOD *

STEFAN HACHUL † AND MICHAEL JÜNGER ‡

Abstract. The evaluation of potential or force fields in systems of N particles whose interactions are Coulombic or gravitational is very important for several applications in natural science, applied mathematics, and computer science. A naive direct calculation of the interactions needs $\Theta(N^2)$ time per time step which is inappropriate for large systems. Therefore, fast hierarchical algorithms (called tree codes) are used that approximate the pairwise interactions in the systems. We present a new multipole-based tree code that runs in $O(N)$ time in the best case and in $O(N \log N)$ time in the worst case. This is an improvement in comparison with existing tree codes: Few of them run in $\Theta(N \log N)$ time. Others are $O(N)$ or $O(N \log N)$ in the best case but quadratic or even unbounded in the worst case. Our practical experiments indicate that the new multipole method is faster than several popular hierarchical N-body simulation algorithms for both uniform and highly non-uniform particle distributions.

Key words. N-body simulations, tree codes, fast multipole method, quadtree

1. Introduction. Astrophysics, electrical engineering, molecular dynamics, or quantum-mechanical simulations in chemistry are typical research areas that require fast *N-body simulation algorithms* (see e.g., [3, 24, 6, 27]). Additionally, in the last decade several of these methods have been successfully introduced into other research areas like applied mathematics [7, 11, 4], information visualization [28, 26, 19], or VLSI-design [10, 32]. In general, N-body simulation algorithms simulate the movements of N masses or charged particles in a specified time interval by iteratively calculating the force field or potential energy field in the system and updating the positions of the objects. Since a naive direct calculation of the interactions needs $\Theta(N^2)$ time per iteration, efficient local methods (see e.g. [22, 21]) and fast hierarchical methods, called *tree codes*, have been developed. We concentrate on the latter methods that are used to approximate the interactions if the desired accuracy is comparatively high.

Greengard [17] and Greengard and Rokhlin [18] invented the *fast multipole method* (`FMM`) that is based on evaluating multipole expansions using a hierarchical data structure called *quadtree* and runs in $O(N)$ best-case time. Additionally, Greengard [17] and Carrier et al. [9] invented an adaptive version of `FMM` for non-homogeneous particle distributions. It is straightforward to see that the worst-case running time of `FMM` is $O(N^2)$. Furthermore, Aluru [1] and Aluru et al. [2] have proven that the running times of the adaptive version of `FMM` and of the tree code of Barnes and Hut [5] depend on the particle distribution and cannot be bounded in N in the general case. This undesirable scaling is caused by the choice of the hierarchical data structures.

Several research results enhance the basic tree code `FMM` (see e.g., [25, 16]), the adaptive version of `FMM` (see e.g., [20, 12, 13, 14, 31]), or the Barnes-Hut method (see e.g., [29, 15]) in many ways. However, since all these methods are based on the quadtree or octtree data structure, the proofs of Aluru [1] and Aluru et al. [2] hold for these algorithms, too. Unlike this, there exist some provably sub-quadratic methods. In particular, Aluru et al. [2] developed an $\Theta(N \log N)$ multipole method that is distribution independent. Other $\Theta(N \log N)$ multipole methods that use a relaxed version of a k-d tree data structure (called *fair-split tree*) were presented by

*This work was partially supported by ADONET.

†Institut für Informatik, Universität zu Köln, Germany, `hachul@informatik.uni-koeln.de`

‡Institut für Informatik, Universität zu Köln, Germany, `mjuenger@informatik.uni-koeln.de`

Callahan and Kosaraju [8] and Xue and Lasher [30].

Based on previous work, we have developed a new provably fast multipole based tree code. In Section 2 we will present construction methods for the used hierarchical data structure that we call *reduced bucket quadtree*. The multipole framework will be presented in Section 3. Experimental comparisons with other popular standard tree codes will be made in Section 4. In the following, we will restrict on Coulombic interactions in two dimensions. But the presented ideas can be extended to gravitational systems or to three dimensions, too, e.g., by using the mathematical tools presented in [17].

2. Construction of the Reduced Bucket Quadtree. We first define the used hierarchical data structure, before we will present three different tree-construction methods in Sections 2.1, 2.2, and 2.3.

DEFINITION 2.1 (Quadtree, Reduced Bucket Quadtree). *Suppose, a set of N particles (or data points) $C = \{c_1, \dots, c_N\}$ are assigned distinct positions in a square D, and a* leaf capacity *$l \geq 1$ is fixed. Furthermore, suppose one recursively subdivides D into four squares of equal size (in the following denoted by* boxes*), until each box contains at most l particles. This process can be represented by an ordered rooted tree of maximum child degree four (with the root representing D) that is called* (bucket) quadtree. *The particles are stored in the leaves of the quadtree. A* degenerate path *$P = (v_1, \dots, v_p)$ in a quadtree is a path in which v_1 and v_p have at least 2 nonempty children, and $v_2, \dots, v_{p-1}$ each have exactly one nonempty child. A* reduced (bucket) quadtree *T can be obtained from a quadtree by shrinking degenerate paths $P = (v_1, \dots, v_p)$ to edges (v_1, v_p) (Figure 2.1 shows an example).*

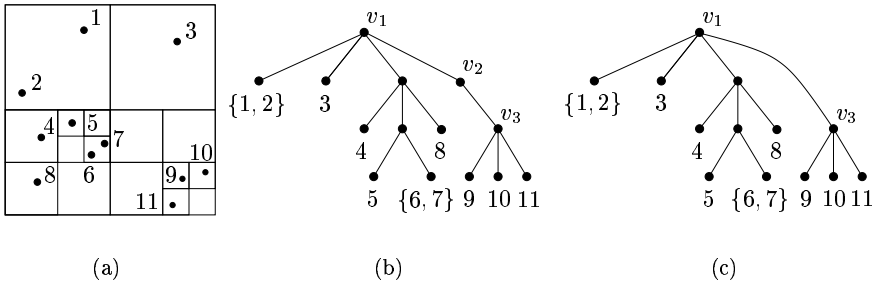

FIG. 2.1. *(a) A distribution of $N = 11$ particles in the plane. (b) The quadtree with leaf capacity $l = 2$ associated with (a). $P = (v_1, v_2, v_3)$ is a degenerate path in the quadtree. (c) The reduced quadtree with leaf capacity $l = 2$ associated with (a).*

One important property of a reduced quadtree is that — unlike the quadtree data structure — it contains only $O(N)$ nodes independently of the distribution of the particles. Aluru et al. [2] present an $\Theta(N \log N)$ method that constructs a reduced quadtree with leaf capacity $l = 1$ (which they call *modified tree*). It can be easily shown by a reduction from sorting that it is neither possible to construct a bucket quadtree nor a reduced bucket quadtree with constant fixed leaf capacity l for arbitrary distributions of the N particles in $o(N \log N)$ time.

2.1. Tree Construction Way A. In the following, let us suppose that N charged particles $C = \{c_1, \dots, c_N\}$ are distributed at distinct positions $p(C) = \{p_1, \dots, p_N\}$ in the plane and that a fixed constant leaf capacity $l \geq 1$ is given.

The tree construction method that we will present next uses a box-shrinking technique of Aluru et al. [2] and techniques that are motivated by ideas of Callahan and Kosaraju [8] in their construction method of the fair-split tree.

2.1.1. Part 1: Initialize the square D. First, we define a square D that covers the particles in C. This can be easily done in linear time.

2.1.2. Part 2: Create Sorted Coordinate Lists S_x and S_y. Now two lists S_x and S_y of length N are created. Each element of the lists contains a different particle of C and two other entries that are initialized as empty entries and will be used later. The list elements of S_x and S_y that contain the same particle are linked by a cross reference to enable efficient access operations. Then, the lists S_x and S_y are sorted according to increasing x-coordinates and y-coordinates of the positions of the contained particles, respectively. Table 2.1(top) shows an example.

S_x	2 $\emptyset,\emptyset$	8 $\emptyset,\emptyset$	4 $\emptyset,\emptyset$	5 $\emptyset,\emptyset$	1 $\emptyset,\emptyset$	6 $\emptyset,\emptyset$	7 $\emptyset,\emptyset$	11 $\emptyset,\emptyset$	3 $\emptyset,\emptyset$	9 $\emptyset,\emptyset$	10 $\emptyset,\emptyset$
S_y	11 $\emptyset,\emptyset$	8 $\emptyset,\emptyset$	9 $\emptyset,\emptyset$	10 $\emptyset,\emptyset$	6 $\emptyset,\emptyset$	7 $\emptyset,\emptyset$	4 $\emptyset,\emptyset$	5 $\emptyset,\emptyset$	2 $\emptyset,\emptyset$	3 $\emptyset,\emptyset$	1 $\emptyset,\emptyset$

$C_x(v_1)$	2	8	4	5	1	6	7	11	3	9	10
$C_y(v_1)$	11	8	9	10	6	7	4	5	2	3	1

TABLE 2.1

(top) The sorted lists S_x and S_y of the particle distribution of Figure 2.2(a) and (bottom) the corresponding lists $C_x(v_1)$ and $C_y(v_1)$ of root node v_1.

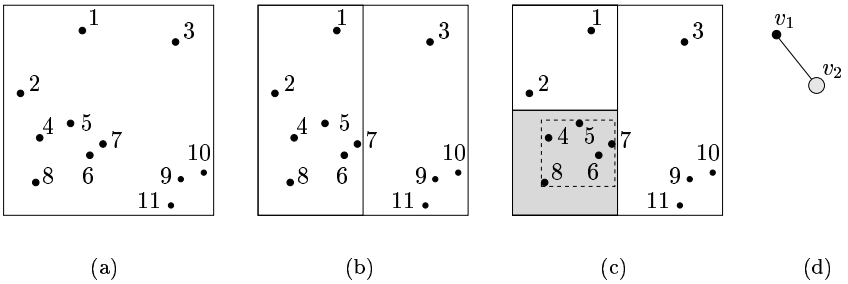

FIG. 2.2. *(a) A distribution of $N = 11$ particles in a square D that is associated with the root node v_1 of T. (b) Subdividing $D = \mathrm{box}(v_1)$ into two halves. The left half of $\mathrm{box}(v_1)$ contains more particles than the right half. (c) Subdividing the left half of D into two boxes. The left bottom box contains more particles than the left top box. The dashed rectangle R is the smallest rectangle that covers all particles contained in the grey sub-box of D. (d) Creating a new child v_2 of v_1 that corresponds to the left bottom sub-box of $\mathrm{box}(v_1)$.*

2.1.3. Part 3: Create the Root Node and Copies of S_x and S_y. The root node of the reduced bucket quadtree T is created next. Since this node is also the actual visited node, we denote it by v_{act}, and the corresponding square region is $box(v_{act}) := D$.

Furthermore, a copy of S_x and a copy of S_y are created. In contrast to S_x and S_y, they contain the particle entries only. These lists are assigned to v_{act} and are denoted by $C_x(v_{act})$ and $C_y(v_{act})$. To enable efficient access operations, for each

$i \in \{1, \ldots, N\}$ the i-th entry in S_x is linked with the i-th entry of $C_x(v_{act})$ and vice versa. Analog, the i-th entry in S_y is linked with the i-th entry of $C_y(v_{act})$ and vice versa. Additionally, cross references between elements of $C_x(v_{act})$ and $C_y(v_{act})$ that contain the same particle are created (Table 2.1(bottom)). If $N < l$, we assign all particles in C to v_1 and are done, since v_1 is already the reduced bucket quadtree. Otherwise, we proceed with the next parts of the algorithm.

2.1.4. Part 4: Alternating $C_x(v_{act})$ Traversal. $C_x(v_{act})$ is traversed alternating from the beginning and the end heading toward the middle. This is done in order to decide which particles of $C_x(v_{act})$ belong to the left half of $box(v_{act})$ and which particles belong to the right half of $box(v_{act})$.

Let $M = \{c_1, \ldots, c_k\} \subseteq C$ be the set of particles in S_x that are contained in the half of $box(v_{act})$ that contains the fewest particles. Then, for each particle $c_i \in M$ a link to v_{act} and an index that identifies that c_i belongs to the left (index l) or right (index r) part of $box(v_{act})$ is added to the elements containing c_i in S_x and S_y. Afterwards, all elements of $C_x(v_{act})$ and $C_y(v_{act})$ that contain a particle $c_i \in M$ are deleted from $C_x(v_{act})$ and $C_y(v_{act})$, respectively (see Figure 2.2(b) and Table 2.2).

S_x	2 $\emptyset, \emptyset$	8 $\emptyset, \emptyset$	4 $\emptyset, \emptyset$	5 $\emptyset, \emptyset$	1 $\emptyset, \emptyset$	6 $\emptyset, \emptyset$	7 $\emptyset, \emptyset$	11 v_1, r	3 v_1, r	9 v_1, r	10 v_1, r
S_y	11 v_1, r	8 $\emptyset, \emptyset$	9 v_1, r	10 v_1, r	6 $\emptyset, \emptyset$	7 $\emptyset, \emptyset$	4 $\emptyset, \emptyset$	5 $\emptyset, \emptyset$	2 $\emptyset, \emptyset$	3 v_1, r	1 $\emptyset, \emptyset$

$C_x(v_1)$	2	8	4	5	1	6	7
$C_y(v_1)$	8	6	7	4	5	2	1

TABLE 2.2
The lists $S_x, S_y, C_x(v_1)$ and $C_y(v_1)$ after the alternating $C_x(v_1)$ traversal.

2.1.5. Part 5: Alternating $C_y(v_{act})$ Traversal. Let *large_half*(v_{act}) be the half of $box(v_{act})$ that contains the most particles. Then, $C_y(v_{act})$ is traversed alternating from the beginning and the end heading toward the middle in order to decide which particles of $C_y(v_{act})$ belong to the bottom half of *large_half*(v_{act}) and to the top half of *large_half*(v_{act}).

Let $M = \{c_1, \ldots, c_k\} \subseteq C$ denote the set of particles in S_y that are contained in the box of *large_half*(v_{act}) that covers the fewest particles. Then, for each particle $c_i \in M$ a link to v_{act} and an index that identifies that c_i belongs to the top half or bottom half of *large_half*(v_{act}) is added to the entries of c_i in S_x and S_y. Afterwards, all entries in $C_x(v_{act})$ and $C_y(v_{act})$ that contain a particle $c_i \in M$ are deleted from $C_x(v_{act})$ and $C_y(v_{act})$, respectively (see Figure 2.2(c) and Table 2.3).

S_x	2 v_1, lt	8 $\emptyset, \emptyset$	4 $\emptyset, \emptyset$	5 $\emptyset, \emptyset$	1 v_1, lt	6 $\emptyset, \emptyset$	7 $\emptyset, \emptyset$	11 v_1, r	3 v_1, r	9 v_1, r	10 v_1, r
S_y	11 v_1, r	8 $\emptyset, \emptyset$	9 v_1, r	10 v_1, r	6 $\emptyset, \emptyset$	7 $\emptyset, \emptyset$	4 $\emptyset, \emptyset$	5 $\emptyset, \emptyset$	2 v_1, lt	3 v_1, r	1 v_1, lt

$C_x(v_1)$	8	4	5	6	7
$C_y(v_1)$	8	6	7	4	5

TABLE 2.3
The lists S_x, S_y, $C_x(v_1)$ and $C_y(v_1)$ after the alternating $C_y(v_1)$ traversal.

2.1.6. Part 6: Create a Child Node. Now a child node v_{child} of v_{act} is created that corresponds to this sub-box of $box(v_{act})$. Furthermore, $C_x(v_{act})$ and $C_y(v_{act})$ are assigned to v_{child} and labeled $C_x(v_{child})$ and $C_y(v_{child})$, respectively (see Figure 2.2(d)). If the box $box(v_{child})$ contains at most l particles, we proceed with part 8 of this algorithm, otherwise we proceed with part 7.

2.1.7. Part 7: Use Box-Shrinking Technique. Let R be the smallest axis parallel rectangle that covers all particles that are contained in $box(v_{child})$. The box-shrinking technique of Aluru et al. [2] can be used to find the smallest sub-box B of $box(v_{child})$ in $O(1)$ time so that B covers R. Finally, we set $box(v_{child}) := B$. In our example, $box(v_2)$ is already the smallest sub-box of $box(v_2)$ which covers R (see Figure 2.2(c)).

2.1.8. Part 8: Create a Path. The previously described parts can be reused to create a path P in the reduced bucket quadtree T by setting $v_{act} := v_{child}$ and by repeating parts 4 to 7, until the box of the actual node contains less than l particles and, hence, is a leaf of T. Finally, the remaining particles of $C_x(v_{act})$ are assigned to the leaf v_{act} (see Figure 2.3 and Table 2.4).

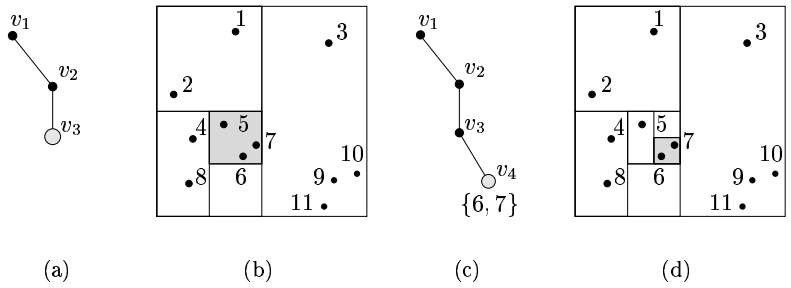

FIG. 2.3. *Constructing a path of the reduced bucket quadtree T. (a) A new node v_3 is added to the tree that corresponds to the grey box in (b). (c) A new node v_4 is added to the tree that corresponds to the grey box in (d). Since the box of v_4 contains only $l = 2$ particles the construction of the path stops, and the particles 6 and 7 are assigned to v_4.*

S_x	2	8	4	5	1	6	7	11	3	9	10
	v_1, lt	v_2, l	v_2, l	v_3, l	v_1, lt	$\emptyset, \emptyset$	$\emptyset, \emptyset$	v_1, r	v_1, r	v_1, r	v_1, r
S_y	11	8	9	10	6	7	4	5	2	3	1
	v_1, r	v_2, l	v_1, r	v_1, r	$\emptyset, \emptyset$	$\emptyset, \emptyset$	v_2, l	v_3, l	v_1, lt	v_1, r	v_1, lt

$C_x(v_3)$	6	7
$C_y(v_3)$	6	7

TABLE 2.4
The lists S_x, S_y, $C_x(v_3)$, and $C_y(v_3)$ after the alternating $C_x(v_3)$ and $C_y(v_3)$ traversal.

2.1.9. Part 9: Create All Children of the Nodes to Which Links in S_x Exist. Now for each node in the actual constructed tree T, to which links in S_x are stored, children are created:

First, the list S_x is traversed from left to right. If during this process one element $(c_j, v_i, region\_index)$ of S_x contains a link to a node v_i — and it is the first time that an element of S_x with a link to v_i has been visited — a new sorted list

$C_x(v_i, \textit{region\_index})$ is created that contains the particle c_j. If during this process one element $(c_j, v_i, \textit{region\_index})$ of S_x is visited that contains a link to a node v_i, and a corresponding list $C_x(v_i, \textit{region\_index})$ already exists, c_j is appended at the end of $C_x(v_i, \textit{region\_index})$. Afterwards, the list S_y is traversed from left to right and sorted lists $C_y(v_i, \textit{region\_index})$ are created in an analog way (see Table 2.5).

$C_x(v_1, lt)$	2	1
$C_y(v_1, lt)$	2	1

$C_x(v_2, l)$	8	4
$C_y(v_2, l)$	8	4

$C_x(v_3, l)$	5
$C_y(v_3, l)$	5

$C_x(v_1, r)$	11	3	9	10
$C_y(v_1, r)$	11	9	10	3

TABLE 2.5

The sorted lists of $C_x(v_i, \textit{region\_index})$ and $C_y(v_i, \textit{region\_index})$ that are obtained from the S_x and S_y lists of Table 2.4.

In order to enable fast access to the elements of these lists, cross references are built up. In particular, each element of S_x that contains a particle c_j is linked with the element of the corresponding list $C_x(v_i, \textit{region\_index})$ that contains the particle c_j and vice versa (analog for S_y and $C_y(v_i, \textit{region\_index})$). Then, for each element of a list $C_x(v_i, \textit{region\_index})$ that contains a particle c_j, a link to the element of the list $C_y(v_i, \textit{region\_index})$ that contains particle c_j is constructed and vice versa. For later use, the elements of S_x and S_y are reinitialized by replacing each element $(c_j, v_i, \textit{region\_index})$ by the element $(c_j, \emptyset, \emptyset)$.

Suppose that we have given a node v_i of the actual tree T for which the lists $C_x(v_i, \textit{region\_index})$ and $C_y(v_i, \textit{region\_index})$ exist, and *region_index* is either l or r. Then, we split each list $C_x(v_i, \textit{region\_index})$ with $\textit{region\_index} \in \{l, r\}$ into two sub-lists $C_x(v_i, lt)$ and $C_x(v_i, lb)$ (if $\textit{region\_index} = l$) and $C_x(v_i, rt)$ and $C_x(v_i, rb)$ (if $\textit{region\_index} = r$), respectively (see Table 2.6).

$C_x(v_1, lt)$	2	1
$C_y(v_1, lt)$	2	1

$C_x(v_2, lb)$	8
$C_y(v_2, lb)$	8

$C_x(v_2, lt)$	4
$C_x(v_2, lt)$	4

$C_x(v_3, lt)$	5
$C_x(v_3, lt)$	5

$C_x(v_1, rb)$	11	9	10
$C_y(v_1, rb)$	11	9	10

$C_x(v_1, rt)$	3
$C_x(v_1, rt)$	3

TABLE 2.6

The splat sorted lists of $C_x(v_i, \textit{region\_index})$ and $C_y(v_i, \textit{region\_index})$ that are obtained from the $C_x(v_i, \textit{region\_index})$ and $C_y(v_i, \textit{region\_index})$ lists of Table 2.5.

Now we create for each list $C_x(v_i, \textit{region\_index})$ a child w_i of v_i that corresponds to the sub-box of $\textit{box}(v_i)$ that is defined by index *region_index*. Furthermore, we assign w_i the lists $C_x(w_i) := C_x(v_i, \textit{region\_index})$ and $C_y(w_i) := C_y(v_i, \textit{region\_index})$ for later use. If the list $C_x(w_i)$ has length at most l, w_i is a leaf in the desired reduced bucket quadtree, and all particles that are contained in $C_x(w_i)$ are assigned to w_i. Otherwise, the box-shrinking technique of part 7 is used to obtain the smallest sub-box B of $\textit{box}(w_i)$, which covers the smallest rectangle R that contains the particles of $C_x(w_i)$. Finally, we set $\textit{box}(w_i) := B$ (see Figure 2.4).

2.1.10. Part 10: Recursion. If all lists $C_x(w_i)$ have length at most l, the resulting tree is already the desired reduced bucket quadtree with leaf capacity l. Otherwise, let $\{w_1, \ldots, w_k\}$ be the set of nodes for which the corresponding lists $C_x(w_i)$ have length larger than l. Then, parts 4 to 8 of the algorithm are applied

on each node w_i and its corresponding lists $C_x(w_i)$ and $C_y(w_i)$. As a results of this process, paths $\{P_1, \ldots, P_k\}$ that are rooted at $\{w_1, \ldots, w_k\}$ are generated. In order to create all children of the nodes to which links in the S_x lists are stored, part 9 is performed. Let $\{u_1, \ldots, u_m\}$ be the set of the newly created children for which the length of $C_x(u_i)$ is larger than l. Then, parts 4 to 8 of the algorithm are applied on each node u_i, followed by one call of part 9 and so forth. The recursion ends if for all newly created children the corresponding C_x and C_y lists have length at most l (see Figure 2.5).

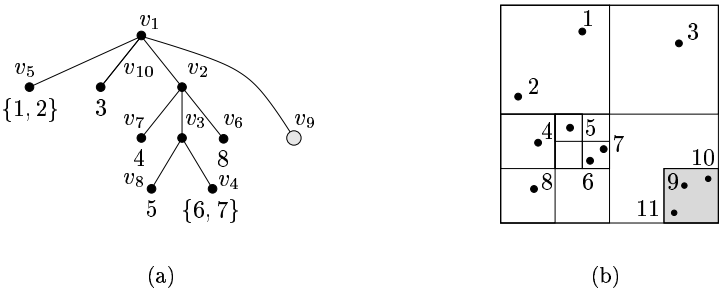

FIG. 2.4. *(a) The tree T after part 9 of the construction algorithm. (b) The corresponding sub-boxes of the square D. The node v_9 in (a) corresponds to the grey shaded region in (b).*

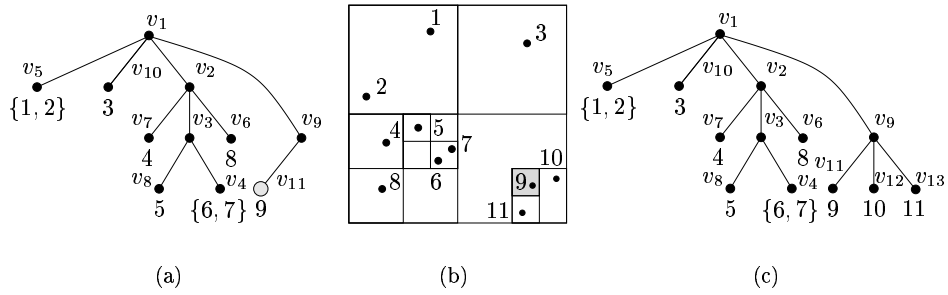

FIG. 2.5. *Recursion: (a) Creating a new child v_{11} of v_9. (b) The grey shaded region corresponds to v_{11}. (c) After creating the children v_{12} and v_{13} of v_9 the tree-construction algorithm ends.*

LEMMA 2.2 (Reduction of the Number of Particles in Leaves). *Suppose, the previously described tree construction algorithm is used to construct a reduced bucket quadtree with leaf capacity l, and the box of the actual visited node v_{act} contains $|C_x(v_{act})|$ particles. Furthermore, suppose, the parts 4 to 9 of the algorithm have to be performed on v_{act} and that the leaves of the newly created subtree that is rooted at v_{act} are denoted by $\{v_1, \ldots, v_k\}$. Then, for each leaf $v_i \in \{v_1, \ldots, v_k\}$ the corresponding box box(v_i) contains at most $\max\{l, \lfloor |C_x(v_{\text{act}})|/2 \rfloor\}$ particles.*

Proof. It is clear that after performing parts 4 to 8 of the algorithm, a path P has been created that contains one leaf v_i, and the corresponding region *box*(v_i) contains at most l particles. In part 9 the other children of the nodes of P are created. For $v_i \in \{v_1, \ldots, v_k\}$ the $C_x(v_i)$ and $C_y(v_i)$ lists have length at most $max(l, \lfloor |C_x(v_{act})|/2 \rfloor)$ since we assign non-empty links only to those elements of S_x and S_y, which contain

particles that are covered by the half of two sub-regions that contains the fewest particles. The length of each such list $C_x(v_i)$ corresponds to the number of particles that are covered by the sub-box of the leaf v_i, which completes the proof. □

THEOREM 2.3 (Tree Construction Way A). *Suppose, $C = \{c_1, \ldots, c_N\}$ is a set of particles that are placed at distinct positions $p(C) = \{p_1, \ldots, p_N\}$ and $l \geq 1$ is an integer constant. Then, the previously described algorithm constructs a reduced bucked quadtree with leaf capacity l in $\Theta(N \log N)$ time.*

Proof. Obviously, parts 1 and 3 need $O(N)$ time, while part 2 needs $\Theta(N \log N)$ time. We concentrate on the running time of the parts 4 to 8 now: Let v_{act} be the actual visited node. Then, the alternating $C_x(v_{act})$ traversal and alternating $C_y(v_{act})$ traversals (parts 4 and 5) need time that is proportional to the number of particles in the part of $box(v_{act})$ or the part of $large\_half(v_{act})$, respectively that contain the fewest particles. Since in S_x and S_y all elements are marked that contain these particles, the running time is proportional to the number of elements in S_x for which non-empty links to nodes of T have been created, too. Part 6 and 7 need $O(1)$ time. Therefore, the running time of the iterative path construction (part 8) is bounded above by the initial length of $C_x(v_{act})$. Since v_{act} is the root of T, the initial length of $C_x(v_{act})$ has been N. In part 9 the remaining children of the nodes in T to which links in S_x exist are created in $O(N)$ time.

Let $L^1 = \{v_1, \ldots, v_k\}$ be the set of all leaves that have been created in part 9 and that contain more than l particles. If $L^1 = \emptyset$ we are done. Otherwise, we analyze the running time of one recursion of part 10: Let v_i be a node in L^1 with associated lists $C_x(v_i)$ and $C_y(v_i)$. Since $\sum_{v_i \in L^1} initial\_length(C_x(v_i)) \leq N$, the execution of parts 4 to 8 for all nodes $v_i \in L^1$ and, hence, one recursion of part 10 needs $O(N)$ time.

Hence, the total running time of the algorithm is $\Theta(N \log N + N \cdot |recursion\_levels|)$, where *recursion_levels* is the number of recursions. It follows from Lemma 2.2 that the number of recursion levels is bounded above by $O(\log N)$ which completes the proof. □

2.2. Tree Construction Way B. We will explain another new tree construction method that is conceptually simpler and works as follows: After initializing the square D, we build up a complete truncated quadtree T^1 with depth $\max\{1, \lfloor \log_4 N / l \rfloor\}$. Then, all particles are assigned to the leaves of T^1. Afterwards, the tree is traversed bottom up and thereby for each internal tree node the number of particles that are contained in its associated box are calculated. Figure 2.6(a) shows an example.

In the next step, we thin out T^1. Therefore, we traverse the quadtree T^1 top down and thereby delete all nodes that do not contain particles and shrink degenerate path to edges. If (during this process) we visit an internal node v that is the root of a subtree which contains at most l particles, this subtree is deleted, and all the particles that were stored in the deleted subtree are assigned to v (see Figure 2.6(b)). If none of the leaves of T^1 contains more than l particles, the procedure ends. Otherwise, we repeat the previous steps recursively for all leaves v_i that contain $N_i > l$ particles (see Figure 2.6(c)-(d)).

THEOREM 2.4 (Tree Construction Way B). *Suppose that $C = \{c_1, \ldots, c_N\}$ is a set of particles that are placed at distinct positions $p(C) = \{p_1, \ldots, p_N\}$ and that $l \geq 1$ is an integer constant. Suppose that the maximum and minimum of the Euclidean distances between any two particles in C are denoted by d_{max} and d_{min}, respectively. Then, the described algorithm constructs a reduced bucked quadtree with leaf capacity l in $O(N \cdot \log(d_{max}/d_{min}))$ time in the worst case. Its best-case running time is $O(N)$.*

Proof. Since the complete quadtree T^1 contains $O(N)$ nodes, and its structure is

predefined, it can be build up and thinned out in $O(N)$ time. If T^1 is not the reduced quadtree, we build up subtrees $T^2(v_1), \ldots, T^2(v_k)$ for all leaves $v_1, \ldots, v_k$ of T^1 that contain more than l particles. This needs $O(N)$ time in total, since the sum of the tree nodes contained in all $T^2(v_i)$ is at most $O(N)$. Then, we possibly have to build up subtrees rooted at the leaves of the T^2-trees and so forth. Since for each $j \geq 1$ the sum of the tree nodes of all T^j is bounded above by $O(N)$, the total running time is $O(|\text{recursion\_levels}| \cdot N)$. The number of recursion levels is bounded above by $\log(d_{max}/d_{min})$ which completes the proof. □

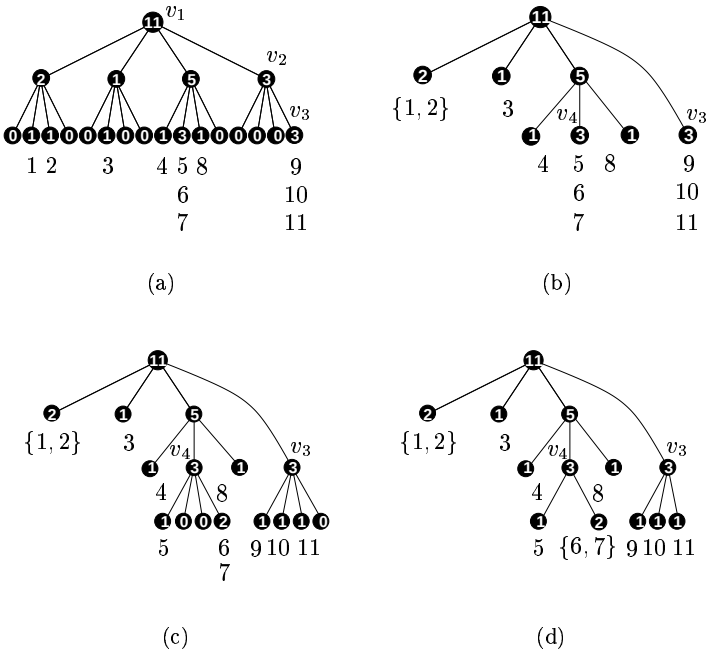

FIG. 2.6. *Building up the reduced quadtree T with leaf capacity $l = 2$ and $N = 11$ particles for the distribution of Figure 2.2(a). (a) First step: Building up the complete quadtree T^1. (b) Second step: Thinning out T^1. (c) Recursion: Building up the complete quadtrees $T^2(v_3)$ and $T^2(v_4)$. (d) The reduced quadtree is constructed after thinning out $T^2(v_3)$ and $T^2(v_4)$.*

2.3. Tree Construction Way C. A hybrid version that is based on the previously described tree-construction ways is straightforward: First, a maximum number of $O(\log N)$ recursion levels is fixed. Then, tree-construction way B is started and the tree is recursively build up. Tree-construction way B is stopped if either the tree has been built up completely or the maximum number of recursion levels has been reached. In the latter case, tree construction way A is called for each leaf of the partially constructed tree that contains more than the constant number l of particles.

COROLLARY 2.5 (Tree Construction Way C). *Suppose that $C = \{c_1, \ldots, c_N\}$ is a set of particles that are placed at distinct positions $p(C) = \{p_1, \ldots, p_N\}$ and that $l \geq 1$ is an integer constant. Then, the previously described hybrid tree-construction method constructs a reduced bucked quadtree with leaf capacity l in $O(N)$ best-case and $O(N \log N)$ worst-case running time.*

Proof. The corollary follows directly from Theorem 2.3 and Theorem 2.4. □

3. The Multipole Framework. In Sections 3.1 and 3.2 we will introduce the needed mathematical tolls and some terminology, respectively. The multipole framework will be described in Section 3.3.

3.1. Tools From Complex Analysis. The following theorems have been proven by Greengard and Rokhlin [18] and Greengard [17] under the assumption that either z_0 or z_1 is the origin. We use them in the more general setting. The proofs are analog to the ones presented in [18, 17] and omitted here for brevity.

Note that each point $p = (x, y) \in \mathbb{R}^2$ is identified with a point $z = x + iy \in \mathbb{C}$.

THEOREM 3.1 (Multipole Expansion Theorem). *Suppose, m charged particles $\{c_1, \ldots, c_m\}$ with charges $\{q_1, \ldots, q_m\}$ are located at points $\{p_1, \ldots, p_m\}$ inside a circle of radius r with center z_0. Then, for any $z \in \mathbb{C}$ with $|z - z_0| > r$ the potential energy $\mathcal{E}(z)$ at point z induced by the m charged particles is given by*

$$\mathcal{E}(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z - z_0)^k} \text{ , where}$$

$$a_0 = \sum_{i=1}^{m} q_i \quad \text{and} \quad a_k = \sum_{i=1}^{m} \frac{-q_i (p_i - z_0)^k}{k} .$$

LEMMA 3.2 (Translation of Multipole Expansions). *Suppose, $\mathcal{E}(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z-z_0)^k}$ is a multipole expansion of the potential energy due to a set of charged particles that are located inside a circle of radius r and center z_0. Then, for any z outside a circle of radius $r + |z_0 - z_1|$ and center z_1, the potential energy induced by these particles is given by*

$$\mathcal{E}(z) = a_0 \log(z - z_1) + \sum_{l=1}^{\infty} \frac{b_l}{(z - z_1)^l} \text{ , where}$$

$$b_l = \frac{-a_0 (z_0 - z_1)^l}{l} + \sum_{k=1}^{l} a_k (z_0 - z_1)^{l-k} \binom{l-1}{k-1} .$$

LEMMA 3.3 (Conversion of Multipole Expansions to Local Expansions). *Suppose, C_0 is a set of charged particles that are located inside a circle of radius r and center z_0, the corresponding multipole expansion is given by $\mathcal{E}(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z-z_0)^k}$, and that z_1 is a point with $|z_1 - z_0| > 2r$. Then, inside a circle of radius r and center z_1 the potential energy due to these particles is given by the power series*

$$\mathcal{E}(z) = \sum_{l=0}^{\infty} b_l \cdot (z - z_1)^l \text{ , where}$$

$$b_0 = a_0 \log(z_1 - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z_1 - z_0)^k} \quad \text{and}$$

$$b_l = \frac{(-1)^{l+1} a_0}{(z_1 - z_0)^l \cdot l} + \left(\frac{1}{z_0 - z_1}\right)^l \sum_{k=1}^{\infty} \binom{l+k-1}{k-1} \frac{a_k}{(z_1 - z_0)^k} \text{ , for } l \geq 1 .$$

LEMMA 3.4 (Translation of Local Expansions). *For any complex z_0, z_1, z, and $\{a_0, \ldots, a_p\}$*

$$\sum_{k=0}^{p} a_k (z - z_0)^k = \sum_{l=0}^{p} \left(\sum_{k=l}^{p} a_k \binom{k}{l} (z_1 - z_0)^{k-l} \right) (z - z_1)^l .$$

The finite Laurent series $M^p(z)$ that is obtained by calculating only the coefficients 0 to p of a multipole expansion is called *p-term multipole expansion.* Analog, the finite power series $L^p(z)$ that is obtained by calculating only the coefficients 0 to p of a local expansion is called is called *p-term local expansion.* Note that if $\mathcal{E}(z)$ describes the potential energy field in a system of charged particles, and z is contained in a region of analyticity, then, the forces that act on a particle of unit charge at position z are given by $(Re(\mathcal{E}'(z)), -Im\ (\mathcal{E}'(z)))$.

REMARK 3.5. *Clearly the following statements (assuming all operations are applied on well-defined sets) hold:*

- *(a) The coefficients of a p-term multipole expansion due to m charged particles can be obtained in $O(pm)$ time.*
- *(b) The coefficients of a shifted p-term multipole expansion can be obtained in $O(p^2)$ time.*
- *(c) The coefficients of a p-term local expansion can be obtained from the coefficients of a p-term multipole expansion in $O(p^2)$ time.*
- *(d) The coefficients of a shifted p-term local expansion can be obtained in $O(p^2)$ time.*
- *(e) The derivative of a p-term multipole expansion and the derivative of a p-term local expansion can be obtained in $O(p)$ time.*
- *(f) An approximation of the force that acts on a particle of unit charge at position z — which is induced by the potential energy field that is described by a p-term multipole expansion or a p-term local expansion — can be obtained in $O(p)$ time.*

3.2. Some Terminology. In Section 2 we have associated each node v of the reduced bucket quadtree with its box $box(v)$. In the following, we will associate two additional boxes with a node v that are defined next.

DEFINITION 3.6 (Small Cell, Large Cell). *Suppose, T is a reduced bucket quadtree for a given set $C = \{c_1, \ldots, c_N\}$ of particles that are distributed in a square D, and v is a node of T. The* small cell *or* small box *of v (shorter $Sm(v)$) is the smallest sub-box of D that covers all particles that are associated with v. If v is a leaf that contains only one particle, then, $Sm(v)$ is a point. If v is the root of T, the* large cell *or* large box *of v (shorter $Lg(v)$) is equal to $Sm(v)$. Otherwise, let parent(v) be the parent of v in T. Then, $Lg(v)$ is the largest sub-box of $Sm(parent(v))$ with size smaller than $Sm(parent(v))$ that covers $Sm(v)$.*

REMARK 3.7. *It follows from the definition of $Sm(v)$, $Lg(v)$, and the reduced bucket quadtree T that the small cell $Sm(v)$ of an interior node v is exactly $box(v)$. For a leaf v of T, $Sm(v)$ is covered by $box(v)$. Furthermore, $Lg(v) \setminus Sm(v)$ covers no particles of C, and the size of $Lg(v)$ is $\frac{1}{4}$ of the size of $Sm(parent(v))$ if v is not the root of T (see Figure 3.1).*

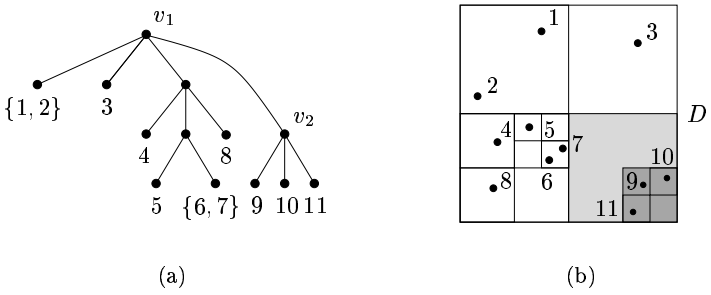

FIG. 3.1. *(a) A reduced bucket quadtree that corresponds to the particle distribution in the square D shown in (b). The small cell $Sm(v_2)$ corresponds to the small dark-gray box. The small cell of* $\text{parent}(v_2) = v_1$ *corresponds to the square D. The large cell $Lg(v_2)$ corresponds to the big light-gray box that covers the small dark-gray box in (b).*

DEFINITION 3.8 (Neighbor, Well-Separated, Ill-Separated). *Two boxes B_1 and B_2 are called* neighbors *if the boundaries of B_1 and B_2 touch, but B_1 and B_2 do not overlap. Two boxes B_1 and B_2 of same size are* well-separated *if they are no neighbors. Otherwise, B_1 and B_2 are* ill-separated. *Suppose, nodes u and v are nodes of a reduced bucket quadtree T and $Sm(u) \geq Sm(v)$. Then, u and v are* well-separated *if and only if $Sm(u)$ and the box that covers $Sm(v)$ and that has the same size as $Sm(u)$ are well-separated. Otherwise u and v are* ill-separated *(Figure 3.2 shows an example).*

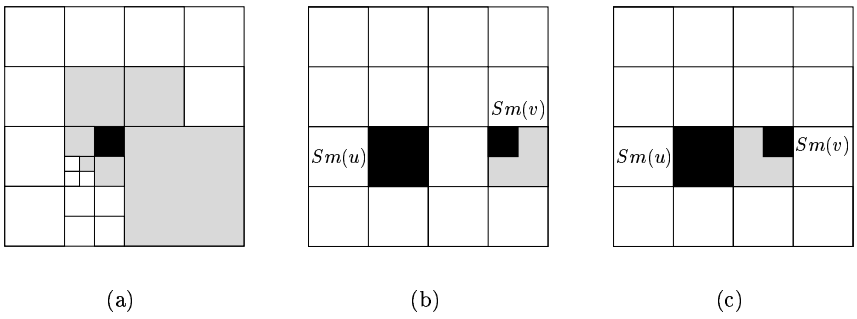

FIG. 3.2. *(a) Unlike the white boxes, the gray boxes are neighbors of the black box. (b-c) The black boxes $Sm(u)$ and $Sm(v)$ are small cells of two nodes u and v of a reduced bucked quadtree. (b) Since the gray box is no neighbor of $Sm(u)$, u and v are well-separated. (c) Since the gray box is a neighbor of $Sm(u)$, u and v are ill-separated.*

Like in [17, 2] the terminology of well-separateness is used to define an interaction set that is used do generate p-term local expansions from suitable p-term multipole expansions.

DEFINITION 3.9 (Interaction Set, Minimal Ill-Separated Set). *Suppose, nodes u and v are nodes of a reduced (bucket) quadtree T. The* interaction set $I(v)$ *of a node v is the set of all nodes u that are ill-separated from the parent of v, well-separated from v, and the parent of u is ill-separated from v. Thus,* $I(v) = \{u \mid$ `Well_Separated`(v,u), `Ill_Separated` $(parent(v),u)$, `Ill_Separated`$(parent(u),v)\}$.

The minimal ill-separated set $R(v)$ *of a node* v *is the set of all nodes that are ill-separated from* v *and have the small cell smaller or equal and the large cell larger or equal than the small cell of* v. *Hence,* $R(v) = \{u \mid \texttt{Ill\_Separated}(v,u), Sm(u) \leq Sm(v)) \leq Lg(u)\}$.

Aluru et al. [2] associate the lists $I(v)$ and $R(v)$ with each node v of their hierarchical data structure. Since our data structure is more general, we have to defining some additional sets which are assigned to each node of the tree.

DEFINITION 3.10 (The Sets $D_1(v)$, $D_2(v)$, $D_3(v)$, and $K(v)$). *For each node* v *of the reduced bucket quadtree* $T = (V, E)$, $D_1(v)$ *is the set of all leaves* $w \in V$ *so that* $Sm(v) < Sm(w)$, *and* $Sm(v)$ *and* $Sm(w)$ *are neighbors.* $D_2(v)$ *is the set of all leaves* w *of* T *so that* $Sm(v) < Sm(w)$, $Sm(v)$ *and* $Sm(w)$ *are no neighbors,* v *and* w *are ill-separated, and* w *is not contained in the set* $D_2(u)$ *of an ancestor* u *of* v. *For each leaf* $v \in V$ *we additionally define the sets* $D_3(v)$ *and* $K(v)$, *where* $D_3(v)$ *is the set of all leaves* $w \in V$ *so that* $Sm(v) \geq Sm(w)$, *and* $Sm(v)$ *and* $Sm(w)$ *are neighbors. For a leaf* $v \in V$ *the set* $K(v)$ *contains all nodes* w *so that* $Sm(v)$ *and* $Sm(w)$ *are no neighbors. Additionally, it is required for each* $w \in K(v)$ *that either* $w \in R(v)$ *or an ancestor of* w *is contained in* $R(v)$, *and* $Sm(parent(w))$ *and* $Sm(v)$ *are neighbors.*

3.3. Formal Description of the Multipole Framework. The pseudocode of the multipole framework is shown in Function `Multipole_Framework` and Function `Calculate_Local_Expansions_and_Node_Sets`, and its parts are explained next.

Part 1: (Trivial Case and Initializations) If the reduced bucket quadtree $T = (V, E)$ consists of only one node, the forces that act on each particle c_i due to all other particles in C are calculated directly. Otherwise, for each node v of T the sets $R(v), I(v), D_1(v), D_2(v)$ (and for the leaves additionally $D_3(v)$ and $K(v)$) are initialized as empty sets.

Part 2: (Bottom-Up Traversal of the Tree) Now, for each leaf v of T the coefficients $\{a_0, \ldots, a_p\}$ of the p-term multipole expansion $M^p(v)$ that reflects an approximation of the potential energy field due to the particles that are contained in $Sm(v)$ are calculated. This is done by using Theorem 3.1 and choosing the center of $Sm(v)$ as the variable z_0 in Theorem 3.1.

The p-term multipole expansions of the interior nodes are calculated by traversing the tree bottom up: Suppose, v is an interior node of T with small cell $Sm(v)$, the center of $Sm(v)$ is z_1, and the p-term multipole expansions $M^p(w)$ of each of v's children w have been calculated. Then, the coefficients of the p-term multipole expansion $M^p(v)$ are obtained by, first, shifting the center of each $M^p(w)$ to z_1 (using Lemma 3.2) and, then, adding the coefficients of this shifted p-term multipole expansion to the corresponding coefficients of $M^p(v)$.

Part 3: (Top-Down Traversal of the Tree) In the top-down traversal of the tree T, Function `Calculate_Local_Expansions_and_Node_Sets` is called for each child of the root node.

Part 3.1: (Find $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$) Here, the sets $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$ are constructed starting with a set $E(v)$ that is assigned $R(parent(v)) \cup D_1(parent(v))$. Note that if $parent(v)$ is the root of T, then, $R(parent(v)) = v$ and $D_1(parent(v)) = \emptyset$. Now, iteratively a node u is taken from $R(parent(v)) \cup D_1(parent(v))$, and it is checked if u already belongs to one of the previous mentioned sets or if one has to explore the subtrees that are rooted at u recursively in order to assign its children to these sets.

In particular, if u and v are well-separated, then, u belongs to $I(v)$. [This can be seen as follows: If u is a node of $R(parent(v)) \cup D_1(parent(v))$, then since u and

Function `Multipole_Framework`(C, T, p)

input : a set $C = \{c_1, \ldots, c_N\}$ of charged particles of unit charge that are placed at distinct positions $p(C) = \{p_1, \ldots, p_N\}$, a reduced bucket quadtree $T = (V, E)$ with fixed constant leaf capacity l that is associated with C, and a fixed constant precision parameter p

output: a function $F\colon C \longrightarrow \mathbb{R}^2$ so that $F(c_i)$ is an approximation of the forces that act on c_i due to all other particles in C

begin
- **begin** {`Part 1: Trivial Case and Initializations`}
  - **if** T contains only one node **then**
    - **foreach** $c_i \in C$ **do**
      - $F(c_i) \leftarrow$ `Naive_Direct_Force_Calculation`;
    - `Exit`;
  - **foreach** $v \in V$ **do** $R(v) \leftarrow I(v) \leftarrow D_1(v) \leftarrow D_2(v) \leftarrow \emptyset$;
  - **foreach** leaf $v \in V$ **do** $D_3(v) \leftarrow K(v) \leftarrow \emptyset$;
- **end**
- **begin** {`Part 2: Bottom-Up Traversal`}
  - **foreach** leaf $v \in V$ **do**
    - calculate the coefficients of $M^p(v)$ due to all particles contained in $Sm(v)$;
  - **foreach** interior node $v \in V$ for which coefficients of $M^p(w)$ of all children w of v have been calculate **do**
    - calculate the coefficients of $M^p(v)$ due to all particles contained in $Sm(v)$ by adding coefficients of shifted $M^p(w)$ of all children w of v;
- **end**
- **begin** {`Part 3: Top-Down Traversal`}
  - **foreach** child v of the root of T **do**
    - `Calculate_Local_Expansions_and_Node_Sets`(T, C, p, v);
- **end**
- **begin** {`Part 4: Obtain the Forces`}
  - **foreach** leaf $v \in V$ **do**
    - **foreach** particle $c_i \in Sm(v)$ **do**
      - let $C(v)$ be the set of charged particles that are contained in $Sm(v)$;
      - calculate $F_{local}(c_i)$ using the coefficients of $L^p(v)$;
      - calculate $F_{direct}(c_i)$ using $D_1(v) \cup D_3(v) \cup C(v)$;
      - calculate $F_{multipole}(c_i)$ using $K(v)$;
      - $F(c_i) \leftarrow F_{local}(c_i) + F_{direct}(c_i) + F_{multipole}(c_i)$;
- **end**

end

Function `Calculate_Local_Expansions_and_Node_Sets`(T, C, p, v)

input : C, T, p are defined like in Function `Multipole_Framework`, a node v of T

output: the coefficients $\{b_0, \ldots, b_p\}$ of the p-term local expansions of v, the sets $R(v)$, $I(v)$, $D_1(v)$, $D_2(v)$, and additionally $D_3(v)$ and $K(v)$ if v is a leaf

begin

- **begin** {`Part 3.1: Find` $R(v), I(v), D_1(v)$, `and` $D_2(v)$}
  - **if** *parent*(v) is the root of T **then** $E(v) \leftarrow$ *parent*(v);
  - **else** $E(v) \leftarrow R($*parent*$(v)) \cup D_1($*parent*$(v))$;
  - **while** $E \neq \emptyset$ **do**
    - pick some $u \in E(v)$, and set $E(v) \leftarrow E(v) \setminus \{u\}$;
    - **if** `Well_Separated`(u, v) **then** $I(v) \leftarrow I(v) \cup \{u\}$;
    - **else if** $Sm(v) \geq Sm(u)$ **then** $R(v) \leftarrow R(v) \cup \{u\}$;
    - **else if** v is no leaf of T **then** $E(v) \leftarrow E(v) \cup$ *children*(u);
    - **else if** `Neighbors`$(Sm(u), Sm(v))$ **then** $D_1(v) \leftarrow D_1(v) \cup \{u\}$;
    - **else** $D_2(v) \leftarrow D_2(v) \cup \{u\}$;
- **end**
- **begin** {`Part 3.2: Calculate Coefficients of` $L^p(v)$}
  - **foreach** $u \in I(v)$ **do**
    - convert $M^p(u)$ to $L^p(u)$, and add coefficients of $L^p(u)$ to coefficients of $L^p(v)$;
  - **foreach** $u \in D_2(v)$ **do**
    - **foreach** $c_i \in Sm(u)$ **do**
      - calculate $M^p(c_i)$, convert $M^p(c_i)$ to $L^p(c_i)$, and add coefficients of $L^p(c_i)$ to coefficients of $L^p(v)$;
  - **if** coefficients of $L^p($*parent*$(v))$ have been calculated **then**
    - add coefficients of shifted $L^p($*parent*$(v))$ to $L^p(v)$;
- **end**
- **begin** {`Part 3.3: Find` $D_3(v)$ `and` $K(v)$ `for Leaves`}
  - **if** v is a leaf of T **then**
    - set $E(v) \leftarrow R(v)$;
    - **while** $E(v) \neq \emptyset$ **do**
      - pick some $u \in E(v)$, and set $E(v) \leftarrow E(v) \setminus \{u\}$;
      - **if** not `Neighbors`$(Sm(u), Sm(v))$ **then** $K(v) \leftarrow K(v) \cup \{u\}$;
      - **if** `Neighbors`$(Sm(u), Sm(v))$ and u is leaf **then** $D_3(v) \leftarrow D_3(v) \cup \{u\}$;
      - **else** $E(v) \leftarrow E(v) \cup$ *children*(u);
- **end**
- **begin** {`Part 3.4: Recursion`}
  - **if** v is no leaf of T **then foreach** child w of v **do**
    - `Calculate_Local_Expansions_and_Node_Sets`(T, C, p, w);
- **end**

end

v are well-separated $u \in R(parent(v))$. Hence, $parent(v)$ and u are ill-separated by definition of $R(parent(v))$. $Lg(u) \geq Sm(parent(v))$ by definition of $R(parent(v))$ and, hence, $parent(u)$ and v are ill-separated, too. If in the complementary case, u is a proper descendant of a node in $R(parent(v)) \cup D_1(parent(v))$, u is a proper descendant of a node $R(parent(v))$ since the nodes in $D_1(parent(v))$ are leaves. Hence, $parent(v)$ and u are ill-separated. Furthermore, $parent(u)$ and v are ill-separated, since otherwise $parent(u) \in I(v)$ by the dynamic construction of $I(v)$.]

If u and v are ill-separated and additionally $Sm(v) \geq Sm(u)$, then, u belongs to $R(v)$. [To see this, we have to prove that $Lg(u) \geq Sm(v)$: Suppose, $u \in R(parent(v)) \cup D_1(parent(v))$, then, it follows from definition of $D_1(parent(v))$ that $u \in R(parent(v))$. Therefore, $Lg(u) \geq Sm(parent(v))$ and, hence, $Lg(u) \geq Sm(v)$. In the complementary case, u is a proper descendant of a node in $R(parent(v))$ by definition of $D_1(parent(v))$. In this case $Lg(u) \geq Sm(v)$ holds since $Sm(parent(u)) > Sm(v)$ by the dynamic construction of $E(v)$.]

If u and v are ill-separated, $Sm(v) < Sm(u)$, u is a leaf, and $Sm(u)$ and $Sm(v)$ are neighbors, then, u is contained in $D_1(v)$.

If u and v are ill-separated, $Sm(v) < Sm(u)$, u is a leaf, and $Sm(u)$ and $Sm(v)$ are no neighbors, then, u is contained in $D_2(v)$, since $R(parent(v)) \cup D_1(parent(v))$ and $D_2(parent(v))$ are disjoint and u is an element of $R(parent(v)) \cup D_1(parent(v))$ or a descendant of an element in $R(parent(v)) \cup D_1(parent(v))$.

In the remaining case $Sm(v) < Sm(u)$ and u is an interior node of T that is ill-separated from v. Hence, one has to check whether its children belong to one of the sets $R(v)$, $I(v)$, $D_1(v)$, or $D_2(v)$.

Note that by construction of part 3.1 the particles that are covered by the small cells of all nodes in $R(parent(v)) \cup D_1(parent(v))$ are exactly the particles that are covered by the small cells of all nodes in $R(v) \cup I(v) \cup D_1(v) \cup D_2(v)$. Furthermore, each such particle is covered by the small cell of exactly one node in $R(v) \cup I(v) \cup D_1(v) \cup D_2(v)$.

Part 3.2: (Calculate Coefficients of $L^p(v)$) For each u in the interaction set $I(v)$ the coefficients of p-term multipole expansions $M^p(u)$ are converted to coefficients of p-term local expansions $L^p(u)$ and added to the corresponding coefficients of $L^p(v)$ using Lemma 3.3 and choosing $z_0 := center(Sm(u))$ and $z_1 := center(Sm(v))$.

Some difficulties arise for the nodes in $D_2(v)$: Since each $u \in D_2(v)$ is a leaf that is ill-separated from v with $Sm(v) < Sm(u)$, one cannot apply Lemma 3.3 on u. However, we found another way to calculate the local expansions due to the particles that are contained in u: Let $\{c_1, \ldots, c_k\}$ be the set of charged particles that are contained in $Sm(u)$ at positions $\{p_1, \ldots, p_k\}$. First, we calculate the coefficients of the p-term multipole expansion $M^p(c_i)$ for each single particle c_i using Theorem 3.1 and choosing $z_0 := p_i$. Since we can interpret each point p_i as a dimensionless box B_i, the largest cell that covers B_i and that has the same size as $Sm(v)$ is no neighbor of $Sm(v)$ due to the definition of $D_2(v)$. Hence, $Sm(v)$ and B_i are well-separated, and Lemma 3.3 can be used to convert $M^p(c_i)$ to a p-term local expansion $L^p(c_i)$ choosing $z_0 := p_i$ and $z_1 := center(Sm(v))$. Finally, for each $c_i \in \{c_1, \ldots, c_k\}$ the coefficients of $L^p(c_i)$ are added to $L^p(v)$.

Following standard practice, the coefficients of the shifted p-term local expansion L^p $(parent(v))$ of the parent of v are added to the corresponding coefficients of $L^p(v)$. Thus, an approximation of the potential energy field of the region that is reflected by $L^p(parent(v))$ is inherited to v using Lemma 3.4 and choosing the center of $Sm(v)$ as z_1. As a result of part 3.2, the coefficients of $L^p(v)$ reflect an approximation of the

potential energy field due to all particles contained in the small cells of the nodes in $I(v)$, $D_2(v)$, and in the $I(u)$ and $D_2(u)$ sets of all ancestors of v. Furthermore, the small cells of the nodes in $R(v) \cup D_1(v)$ are exactly the regions that have not been considered yet in the calculation of the potential energy in $Sm(v)$ due to all particles in the region $D \setminus Sm(v)$.

Part 3.3: (Find $D_3(v)$ and $K(v)$ for Leaves) Suppose, v is a leaf of T and u is a node in $R(v)$. By definition of $R(v)$ we know that u and v are ill-separated and that $Sm(v) \geq Sm(u)$. Three cases can arise: If u and v are no neighbors, u is an element of $K(v)$. If u and v are neighbors and u is a leaf, u is an element of $D_3(v)$. Otherwise, u is a neighbor of v but an interior node of T. Thus, we can recursively assign the children of u to either $K(v)$ or $D_3(v)$, since all children of u are ill-separated from v, their small cell is smaller than $Sm(v)$, and $Sm(u)$ is a neighbor of $Sm(v)$.

Note that the particles that are covered by the small cells of the nodes in $R(v)$ are exactly the particles that are covered by the small cells of the nodes in $K(v) \cup D_3(v)$. Furthermore, each such particle is covered by the small cell of exactly one node in $K(v) \cup D_3(v)$.

Part 3.4: (Recursion) Suppose, v is an interior node of T. The small cells of the nodes in $R(v) \cup D_1(v)$ are exactly the regions that have not been considered yet in the calculation of the potential energy in $Sm(v)$ due to all particles that are contained in the region $D \setminus Sm(v)$. Hence, we can inherit the sets $R(v)$ and $D_1(v)$ to each child w of v and call Function `Calculate_Local_Expansions_and_Node_Sets` for w.

As a result of parts 1 to 3 we have given the coefficients of the p-term local expansions $L^p(v)$ for each leaf v of T, and $L^p(v)$ reflects an approximation of the potential energy field induced by the particles that are not covered by the small cells of all nodes contained in $D_1(v) \cup D_3(v) \cup K(v) \cup Sm(v)$.

Part 4: (Obtain the Forces) Suppose, v is a leaf of T so that $Sm(v)$ contains the particles $\{c_1, \ldots, c_k\}$, which are placed at positions $\{p_1, \ldots, p_k\}$. We can obtain an approximation of the forces that act on each $c_i \in \{c_1, \ldots, c_k\}$ due to all particles in the system $C = \{c_1, \ldots, c_N\}$ as follows:

Let L' be the derivative of $L^p(v)$. Remark 3.5(f) can be used to to obtain the forces that are induced by L' and act on c_i by setting $F_{local}(c_i) = (Re(L'(p_i)), -Im(L'(p_i)))$.

The forces that act on c_i due to all particles that are contained in $Sm(v) \cup \{Sm(u) \mid u \in D_1(v) \cup D_3(v)\}$ are calculated directly by a naive exact force calculation, since v and all nodes $u \in D_1(v) \cup D_3(v)$ are leaves. The forces are denoted by $F_{direct}(c_i)$.

We only have to concentrate on the particles that are covered by the small cells of the nodes in $K(v)$. Let u be a node in $K(v)$. Since u is ill-separated from v and $Sm(v) > Sm(u)$, we cannot apply Lemma 3.3. However, we can use a similar trick like the trick that we have invented in part 3.2: Let c_i be a particle that is placed at position $p_i \in Sm(v)$. Since we can interpret p_i as a dimensionless box B_i, the largest cell that contains c_i and that has the same size as $Sm(u)$ are no neighbors due to the definition of $K(v)$. Hence, $Sm(u)$ and B_i are well-separated, and Remark 3.5(f) can be used in order to obtain the forces that act on c_i due to all particles contained in $Sm(u)$. In particular, let M' be the derivative of $M^p(u)$. Then, the force on c_i that is induced by $M^p(u)$ is given by $F_{multipole}(c_i) = (Re(M'(p_i)), -Im(M'(p_i)))$.

Therefore, the approximation of the force that acts on a particle c_i due to all particles in $C = \{c_1, \ldots, c_N\}$ is given by the sum of $F_{direct}(c_i)$, $F_{local}(c_i)$, and $F_{multipole}(c_i)$.

3.4. The Running Time of the Multipole Framework. We need the following lemma.

LEMMA 3.11 (Sizes of the Sets $R(v)$, $I(v)$, $D_1(v)$, $D_2(v)$, $D_3(v)$, $K(v)$). *Suppose, $T = (V, E)$ is a reduced bucket quadtree with constant leaf capacity l that is associated with a set $C = \{c_1, \ldots, c_N\}$ of N charged particles that are placed at distinct positions $\{p_1, \ldots, p_N\}$, and leaves(T) is the set of the leaves of T. Then,*

$$\sum_{v \in V} |R(v)| = O(N), \tag{3.1}$$

$$\sum_{v \in V} |I(v)| = O(N), \tag{3.2}$$

$$\sum_{v \in V} |D_1(v)| = O(N), \tag{3.3}$$

$$\sum_{v \in V} |D_2(v)| = O(N), \tag{3.4}$$

$$\sum_{v \in \text{leaves}(T)} |D_3(v)| = O(N), \text{ and} \tag{3.5}$$

$$\sum_{v \in \text{leaves}(T)} |K(v)| = O(N). \tag{3.6}$$

Proof. We prove Equation (3.1) first. Let v be an arbitrary node of T. Each node $u \in R(v)$ is ill-separated from v and $Sm(v) \geq Sm(u)$. Hence, $Sm(u)$ is either covered by $Sm(v)$ or covered by a neighbor box B of $Sm(v)$ that has the same size as $Sm(v)$. In the first case, we know from the definition of $R(v)$ that $Sm(v) \leq Lg(u)$. By definition of $Lg(u)$ it follows that $u = v$. In the second case, by using that $Sm(v) \leq Lg(u)$ and Remark 3.7, we know that $Lg(u) \setminus Sm(u)$ contains no particles. Hence, $B \setminus Sm(u)$ contains no particles, too. Since $Sm(v)$ has exactly 8 neighbor boxes B of equal size, we get that $\sum_{v \in V} |R(v)| \leq (1 + 8) \cdot |V| = O(|V|) = O(N)$.

We now prove Equation (3.2). Suppose, w is a node in $I(v)$, then, either w is in $R(parent(v))$ or an ancestor u of w is in $R(parent(v))$ or w is in $D_1(parent(v))$ or an ancestor u of w is in $D_1(parent(v))$. The last two options can be excluded by the fact that all elements of $D_1(parent(v))$ are neighboring leaves of $Sm(parent(v))$ with a small cell that is larger than $Sm(parent(v))$ and, hence, are ill-separated from v and have no descendants. Therefore,

$$\begin{aligned} \textstyle\sum_{v \in V} |\{w \mid w \in I(v)\}| \quad &= \quad \textstyle\sum_{v \in V} |\{w \mid w \in I(v), w \in R(parent(v))\}| + \\ &\quad\; \textstyle\sum_{v \in V} |\{w \mid w \in I(v), w \notin R(parent(v))\}|. \end{aligned}$$

By Equation (3.1) we know that $\sum_{v \in V} |\{w \mid w \in I(v), w \in R(parent(v)\}| = O(N)$. We can rewrite the second term of this equation as follows:

$$\sum_{v \in V} |\{w \mid w \in I(v), w \notin R(parent(v))\}| = \sum_{w \in V} |\{v \mid w \in I(v), w \notin R(parent(v))\}|$$

Hence, in order to show that Equation (3.2) holds, it is sufficient to show that $S(w) := |\{v \mid w \in I(v), w \notin R(parent(v))\}| = O(1)$. Let w be arbitrarily fixed. Since $w \notin R(parent(v))$ there exists a node $u \in R(parent(v))$ that is an ancestor of w. Thus, $parent(v)$ and $parent(w)$ are ill-separated, and $Sm(parent(v)) \geq Sm(parent(w))$. It can be shown that there exists a box B that is a neighbor of box $Sm(parent(w))$, B has the same size as $Sm(parent(w))$, and B is a sub-box of $Sm(parent(v))$ that contains $Sm(v)$. [To see this: $Sm(v)$ cannot be larger than $Sm(parent(w))$, since — by definition of $I(v)$ — v and $parent(w)$ are ill-separated but v and w are well-separated. Suppose, $Sm(v)$ is not covered by a box B that is a neighbor of $Sm(parent(w))$ and that has the same size as $Sm(parent(w))$, then, v and $parent(w)$ are well-separated, which contradicts the fact that $w \in I(v)$ by definition of $I(v)$. Finally, $Sm(parent(v))$ covers B, since $Sm(parent(v)) \geq Sm(parent(w))$.] For each such box B there exist at most 4 sub-boxes like $Sm(v)$ since $Sm(parent(v))$ covers B. Since for each w, the number of neighbor boxes of equal size is bounded above by 8, $|S(w)|$ is bounded above by $4 \cdot 8 = 32$, which completes the proof of Equation (3.2).

The proof of Equation (3.3) is easy. Let v be an arbitrary node of T. Since there exist at most 8 boxes that are neighbors of $Sm(v)$ and that have the same size as $Sm(v)$, there exist less than 8 leaves w that are neighbors of v with $Sm(w) > Sm(v)$. Hence, $\sum_{v \in V} |D_1(v)| < 8|V| = O(N)$.

Next, we prove Equation (3.4). Let us suppose that the leaf capacity l is 1 and that v is a node of T. Then, the small cells of all leaves have size zero. By definition of $D_1(v)$ and $D_2(v)$ it follows that $D_1(v) = D_2(v) = \emptyset$. Thus, by Equations (3.1) and (3.2) for $l = 1$ the equality $\sum_{v \in V} |R(v)| + |I(v)| + |D_1(v)| + |D_2(v)| = O(N)$ holds. Furthermore, the elements of $R(parent(v)) \cup D_1(parent(v))$ or the descendants of the elements of $R(parent(v)) \cup D_1(parent(v))$ are partitioned into the disjoint subsets $R(v)$ and $I(v)$ (since $D_1(v) = D_2(v) = \emptyset$). Now, let us suppose that $l > 1$. Then, the nodes contained in $D_2(v)$ are either elements of the set $R(parent(v)) \cup D_1(parent(v))$ or are descendants of the elements of $R(parent(v)) \cup D_1(parent(v))$. In the first case, the number of elements of $D_2(v)$ that are contained in $R(parent(v)) \cup D_1(parent(v))$ is bounded above by $O(N)$ using Equations (3.1) and (3.3). In the second case, we use that by Equations (3.1), (3.2), and (3.3) $\sum_{v \in V} |R(v)| + |I(v)| + |D_1(v)| = O(N)$. Hence, it is sufficient to show that $\sum_{v \in V} |R(v)| + |I(v)| + |D_1(v)| + |D_2(v)| = O(N)$. This can be shown as follows: Suppose, leaf u is a descendant of a node in $R(parent(v)) \cup D_1(parent(v))$, contains k particles with $2 \leq k \leq l$, and is neither assigned to $I(v)$ nor to $R(v)$. Then, u is assigned to either $D_1(v)$ or $D_2(v)$ since it has no further descendants in T. Hence, the cardinality of the actual set $R(v) \cup I(v) \cup D_1(v) \cup D_2(v)$ is increased by one. In contrast to this, in the case that $l = 1$ the subtree rooted at u would have been explored further. This exploration would result in adding at least two nodes to $R(v) \cup I(v)$. Therefore, in the case that $l > 1$ the expression $\sum_{v \in V} |R(v)| + |I(v)| + |D_1(v)| + |D_2(v)|$ is bounded above by $O(N)$, too.

In order to prove Equation (3.5), we can use that $\sum_{v \in leaves(T)} |\{w \mid Sm(v) \geq Sm(w), leaf(w)\}| = \sum_{w \in leaves(T)} |\{v \mid Sm(v) \geq Sm(w), leaf(w)\}|$. It is sufficient to show that for each leaf w the set $S'(w) := |\{v \mid Sm(v) \geq Sm(w), leaf\ (w)\}|$ is bounded by a constant. Since for each such w at most 8 boxes exist that are neighbors of $Sm(w)$ and have the same size as $Sm(w)$, we get that $|S'(w)| \leq 8$.

Finally, we prove Equation (3.6). By definition of $K(v)$, we know that for each leaf v of the tree $K(v) = K_1(v) \cup K_2(v)$ so that $K_1(v)$ is the set of all nodes w with $Sm(v)$ and $Sm(w)$ are no neighbors and $w \in R(v)$. $K_2(v)$ is the set of all nodes w so that $Sm(v)$ and $Sm(w)$ are no neighbors, $Sm(parent(w))$ and $Sm(v)$ are neighbors,

and an ancestor of w is contained in $R(v)$. Hence, we get that $\sum_{v\in leaves(T)} |\{w \mid w \in K(v)\}| = \sum_{v\in leaves(T)} |\{w \mid w \in K_1(v)\}| + \sum_{v\in leaves(T)} |\{w \mid w \in K_2(v)\}|$.

The first term is bounded by $O(N)$ using Equation (3.1). To estimate the second term, we use that

$$\sum_{v\in leaves(T)} |\{w \mid w \in K_2(v)\}| = \sum_{w\in leaves(T)} |\{v \mid w \in K_2(v)\}|.$$

Let $S''(w) := |\{v \mid w \in K_2(v)\}|$. Then, it is sufficient to show that $|S''(w)|$ is bounded by a constant for an arbitrary node w of T. Suppose, there exists a leaf v of T so that $w \in K_2(v)$. Then, $Sm(parent(w))$ and $Sm(v)$ are neighbors. Since $parent(w)$ or an ancestor of $parent(w)$ is contained in $R(v)$, it is clear that $Sm(v) \geq Sm(parent(w))$. Since at most 8 such leaves v of size larger or equal than $Sm(parent(w))$ are neighbors of $Sm(parent(w))$, we get that $|S''(w)| \leq 8$, which completes the proof. □

THEOREM 3.12 (Multipole Framework). *Suppose, $C = \{c_1, \ldots, c_N\}$ is a set of charged particles of unit charge that are placed at distinct positions $p(C) = \{p_1, \ldots, p_N\}$, $T = (V, E)$ is a reduced bucket quadtree with fixed constant leaf capacity l that is associated with C, and p is a fixed constant precision parameter. Then, Function* `Multipole_Framework` *approximates the force that acts on each particle c_i due to all other particles in C in $O(N)$ time.*

Proof. If T contains only one node, $|C| = N \leq l$. Since l is a constant, the exact naive force calculation in part 1 needs constant time. Otherwise, the initialization of the sets in part 1 needs $O(|V|) = O(N)$ time.

In part 2 the coefficients of the p-term multipole expansions $M^p(v)$ are calculated for all leaves v, using Theorem 3.1. Let $m(v)$ denote the number of particles that are contained in $Sm(v)$ for each leaf v of T. By Remark 3.5(a) this needs $\sum_{v\in leaves(v)} O(p \cdot m(v)) = O(p \cdot N) = O(N)$ time. The coefficients of the p-term multipole expansions of the interior nodes are obtained using Lemma 3.2. By Remark 3.5(b) and the fact that $|V| = O(N)$ the total running time of this step is $O(p^2 \cdot |V|) = O(N)$.

In part 3.1 the sets $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$ are found for a fixed node v. This is done by exploring a collection of $|R(parent(v)) \cup D_1(parent(v))|$ rooted subtrees, where each node in $R(parent(v)) \cup D_1(parent(v))$ is a root. The nodes in $R(parent(v)) \cup D_1(parent(v))$ are either assigned to one of the sets $R(v)$, $I(v)$, $D_1(v)$, and $D_2(v)$ directly, or their children are examined later. Since T is a reduced bucket quadtree, each interior node has at least two children. Hence, the total number of nodes that are visited in the exploration of all subtrees of the nodes in $R(parent(v)) \cup D_1(parent(v))$ is proportional to $|R(v)| + |I(v)| + |D_1(v)| + |D_2(v)|$. It follows from Lemma 3.11 that applying parts 3.1 to all nodes v of T needs $O(N)$ time.

In part 3.2 for each node $u \in I(v)$ converting the coefficients of the p-term multipole expansion $M^p(u)$ to the p-term local expansion $L^p(u)$ and adding these coefficients to the corresponding coefficients of p-term local expansion $L^p(v)$ of v can be done in $O(p^2)$ time using Lemma 3.3 (see Remark 3.5(c)). For each $u \in D_2(v)$ there exists at most l particles c_i that are contained in $Sm(u)$. For each such particle the work needed to calculate $M^p(c_i)$, to convert it to $L^p(c_i)$, and to add its coefficients to the corresponding coefficients of $L^p(v)$ is $O(p^2)$ using Theorem 3.1 and Lemma 3.3 (see Remarks 3.5(a) and (c)). Adding the coefficients of the shifted p-term local expansions $L^p(parent(v))$ of the parent of a fixed node v to $L^p(v)$ can be done in $O(p^2)$ time using Lemma 3.4 (see Remark 3.5(d)). Since we know from Lemma 3.11 that $\sum_{v\in V} I(v) \cup D_2(v) = O(N)$, $O(l \cdot p^2 \cdot N) = O(N)$ time is needed to apply part 3.2 on all nodes $v \in V$.

In part 3.3 the sets $D_3(v)$ and $K(v)$ are constructed by exploring a set of $|R(v)|$ rooted subtrees with roots in $R(v)$. The total number of nodes that are visited in the exploration of all subtrees that are rooted at the nodes in $R(v)$ is proportional to $|D_3(v)| + |K(v)|$. Using Lemma 3.11, applying part 3.3 on all leaves v of T needs $O(N)$ time in total.

In part 4 the forces $F_{direct}(c_i)$, $F_{local}(c_i)$, and $F_{multipole}(c_i)$ are calculated for each particle c_i that is contained in a leaf v of T. The calculation of $F_{local}(c_i)$ needs $O(p)$ time by Remark 3.5(f). Calculating $F_{direct}(c_i)$ can be done in $O(l \cdot (|D_1(v)| + |D_3(v)| + 1))$ time. Calculating $F_{multipole}(c_i)$ can be done in $O(p \cdot |K(v)|)$ time by Remark 3.5(f). Adding $F_{direct}(c_i)$, $F_{local}(c_i)$, and $F_{multipole}(c_i)$ to obtain $F(c_i)$ needs $O(1)$ time. Hence, it follows from Lemma 3.11 that the running time of part 4 is bounded by $\sum_{v \in leaves(T)} \sum_{c_i \in Sm(v)} O(p + l \cdot (|D_1(v)| + |D_3(v)| + 1) + p \cdot |K(v)| + 1) = O(N)$. □

4. Experimental Comparisons.

4.1. The Experimental Framework. In this section we will experimentally compare several tree codes. In particular, we implemented the tree code of Barnes and Hut [5] (`BaHu`), `FMM` by Greengard [17] and Greengard and Rokhlin [18], the method of Aluru et al. [2] (`Al`), and variants of the new multipole method that are based on tree-construction ways A and B ($\texttt{NMM}_\texttt{A}$ and $\texttt{NMM}_\texttt{B}$). Additionally, we implemented the naive exact force-calculation algorithm (`Ex`) that is used as a benchmark. Note that the results for the hybrid tree-construction method ($\texttt{NMM}_\texttt{C}$) are identical to that of $\texttt{NMM}_\texttt{B}$ by a suitable choice of the maximum number of recursion levels.

The algorithms have been implemented in C++ using LEDA-libraries [23]. All experiments were performed on a 2.8 GHz Intel Pentium 4 PC with one gigabyte of memory. We tested these algorithms on three different classes of distributions of N particles. For each distribution we let N range from 4000 up to 128000.

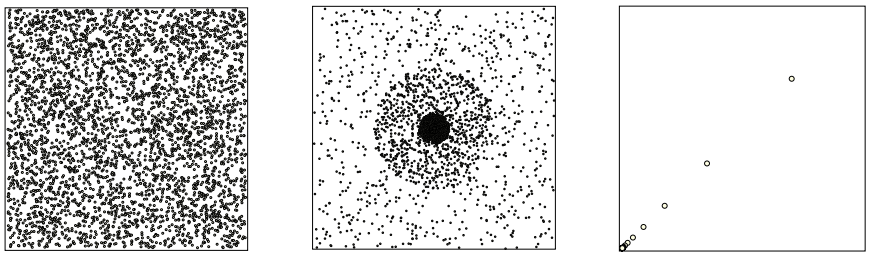

FIG. 4.1. *(left) A uniform and (middle) a non-uniform distribution of* 4000 *particles. (right) A quasi-converging distribution of* 125 *particles.*

We distributed the particles randomly with uniform probability within the square $[0,1] \times [0,1]$ and, following standard practice, call these distributions *uniform* distributions.

The second class of distributions are *non-uniform* distributions. They were created by distributing 20% of the particles randomly with uniform probability within the box $[0,1] \times [0,1]$. Another 20% of the particles were randomly distributed within a disc of radius $\frac{1}{4}$ with center $(\frac{1}{2}, \frac{1}{2})$. The rest of the particles was distributed analogously within discs of radii $\frac{1}{16}$, $\frac{1}{64}$ and $\frac{1}{256}$ with center $(\frac{1}{2}, \frac{1}{2})$.

Finally, we constructed *quasi-converging* distributions by distributing the N particles on the line connecting $(0,0)$ and $P := (10^{25}, 10^{25})$. In particular, the first particle was placed at position $\frac{3}{4}P$. The i-th particle was placed at position $p_i := \frac{3}{4}\frac{P}{2^{i-1}}$, for each i with x-coordinate of p_i greater than $Q := 10^{-25}$. The other particles were placed uniformly on the line connecting $(0,0)$ and Q. Figure 4.1 illustrates the three classes of distributions.

Suppose, $\mathcal{A}$ is a force-approximation method, $F_{\mathcal{A}}(i)$ denotes the approximation of the force that acts on particle i due to all other particles, and $F_{Ex}(i)$ denotes the exact force that acts on particle i due to all other particles. The *error* of the approximation generated by $\mathcal{A}$ is:

$$\text{Error}(\mathcal{A}) := \sqrt{\frac{\sum_{i=1}^{N} \|F_{Ex}(i) - F_{\mathcal{A}}(i)\|^2}{\sum_{i=1}^{N} \|F_{Ex}(i)\|^2}}$$

We say that an approximation that is generated by $\mathcal{A}$ is of *low*, *medium*, and *high* accuracy if $\text{Error}(\mathcal{A}) < 10^{-2}$, $\text{Error}(\mathcal{A}) < 10^{-3}$, and $\text{Error}(\mathcal{A}) < 10^{-4}$), respectively.

4.2. The Numerical Results. For each tree code and class of distributions, we determined the parameters that guarantee low, medium, and high accuracy for all sizes of N. We selected the set of parameters that resulted in the fastest running times for the fixed accuracies. These parameters are listed in Table 4.1. The bucket capacity l in $\texttt{NMM}_\texttt{A}$ and $\texttt{NMM}_\texttt{B}$ is 25.

Class of Distributions	Accuracy	Parameters of Algorithm:				
		BaHu	FMM	Al	$\texttt{NMM}_\texttt{A}$	$\texttt{NMM}_\texttt{B}$
uniform	low	$\alpha = 0.7$	$p = 3$	$p = 3$	$p = 3$	$p = 3$
	medium	$\alpha = 0.2$	$p = 4$	$p = 4$	$p = 4$	$p = 4$
	high	$\alpha = 0.1$	$p = 6$	$p = 6$	$p = 6$	$p = 6$
non-uniform	low	$\alpha = 0.7$	$p = 3$	$p = 3$	$p = 3$	$p = 3$
	medium	$\alpha = 0.2$	$p = 4$	$p = 5$	$p = 5$	$p = 5$
	high	$\alpha = 0.1$	$p = 6$	$p = 7$	$p = 7$	$p = 7$
quasi-converging	low	$\alpha = 0.3$	$p = 1$	$p = 4$	$p = 4$	$p = 4$
	medium	$\alpha = 0.1$	$p = 1$	$p = 6$	$p = 6$	$p = 6$
	high	$\alpha = 0.03$	$p = 1$	$p = 8$	$p = 8$	$p = 8$

TABLE 4.1
Parameters of tree codes that guarantee a desired accuracy.

Note that choosing $p = 1$ in the method FMM for quasi-converging distributions is sufficient to guarantee all desired accuracies, since there exists a leaf in each complete quadtree that contains nearly all particles. Hence, the interactions between these nodes are calculated exactly, and the choice of p is marginal. The parameters of $\texttt{NMM}_\texttt{A}$ and $\texttt{NMM}_\texttt{B}$ are identical, since both methods differ in the tree construction procedure only. For brevity, we only present the running times for low and high desired accuracies. For each size of N, each class of distribution, and each algorithm, the reported times are the average times of 100 tests.

For uniform distributions (see Table 4.2) and low and high accuracy all tree codes are much faster than the exact method. In comparison with BaHu the multipole methods scale much better if high accuracy is desired. The expected running time of FMM and $\texttt{NMM}_\texttt{B}$ is linear for these distributions. Therefore, it is not surprising that they are very fast. $\texttt{NMM}_\texttt{B}$ is the fastest method for both accuracies. Al is roughly a factor 3 to 5 slower than $\texttt{NMM}_\texttt{B}$. $\texttt{NMM}_\texttt{A}$ is roughly a factor 2 slower than $\texttt{NMM}_\texttt{B}$.

Particles	Accuracy	Tree Codes					Exact
		BaHu	FMM	Al	NMM_A	NMM_B	Method
4000	Low	0.18	0.09	0.34	0.15	0.08	2.03
8000		0.42	0.34	0.70	0.39	0.21	8.59
16000		0.95	0.40	1.46	0.83	0.40	35.52
32000		2.07	1.42	2.93	1.90	0.93	142.58
64000		4.57	1.73	6.01	3.96	1.79	572.90
128000		10.16	5.86	12.28	9.00	4.07	2288.73
4000	High	3.23	0.20	0.60	0.18	0.11	2.03
8000		8.49	0.78	1.24	0.49	0.29	8.59
16000		21.30	0.85	2.57	0.96	0.50	35.52
32000		51.22	3.25	5.22	2.36	1.29	142.58
64000		120.42	3.59	10.60	4.64	2.26	572.90
128000		282.20	13.33	21.50	10.91	5.54	2288.73

TABLE 4.2

Running times in seconds for the calculation of the forces and uniform distributions.

As expected, for non-uniform distributions (see Table 4.3) the running times of FMM grow significantly in comparison with the running times of FMM for uniform distributions. The method BaHu is up to a factor 2 slower in comparison with the CPU times that are needed by BaHu for uniform distributions. Only the running times of Al, NMM_A, and NMM_B keep nearly unchanged. Again, for both desired accuracies NMM_B is the fastest method.

Particles	Accuracy	Tree Codes					Exact
		BaHu	FMM	Al	NMM_A	NMM_B	Method
4000	Low	0.26	0.53	0.33	0.17	0.12	2.06
8000		0.58	1.95	0.68	0.36	0.22	8.57
16000		1.29	10.33	1.41	0.81	0.46	35.54
32000		2.72	25.07	2.81	1.65	0.91	142.37
64000		5.97	109.34	5.78	3.67	1.92	572.12
128000		12.71	230.60	11.87	7.92	3.80	2283.39
4000	High	5.31	0.64	0.70	0.20	0.14	2.06
8000		15.34	2.46	1.44	0.48	0.32	8.57
16000		38.97	11.01	2.98	0.97	0.61	35.54
32000		91.76	28.44	6.07	2.26	1.40	142.37
64000		212.94	120.67	12.30	4.53	2.58	572.12
128000		468.64	250.17	25.04	10.45	5.88	2283.39

TABLE 4.3

Running times in seconds for the calculation of the forces and non-uniform distributions.

For quasi-converging distributions (see Table 4.4) the running times of FMM are nearly identical with that of Ex. NMM_A and NMM_B are the fastest methods for both accuracies and need nearly the same amounts of running time. The method Al is a factor 2 to 3 slower than NMM_A and NMM_B.

We can summarize that the multipole methods Al, NMM_A, NMM_B (and, hence, NMM_C) are best suited for approximating Coulombic interactions in the plane if high accuracy is desired and the particle distributions are not uniform. Since the constant factors of

Particles	Accuracy	Tree Codes					Exact
		BaHu	FMM	Al	$\texttt{NMM}_\texttt{A}$	$\texttt{NMM}_\texttt{B}$	Method
4000	Low	0.87	2.06	0.26	0.08	0.15	2.03
8000		1.79	8.83	0.53	0.18	0.27	8.53
16000		3.67	35.92	1.04	0.39	0.55	35.83
32000		7.34	144.59	2.10	0.77	0.97	142.14
64000		14.99	577.18	4.11	1.67	1.95	573.30
128000		30.92	2315.93	8.44	3.61	3.67	2280.16
4000	High	1.67	2.06	0.39	0.10	0.16	2.03
8000		3.65	8.83	0.80	0.21	0.29	8.53
16000		7.89	35.92	1.60	0.45	0.60	35.83
32000		16.11	144.59	3.18	0.94	1.06	142.14
64000		35.62	577.18	6.32	2.03	2.13	573.30
128000		79.57	2315.93	12.83	4.34	4.00	2280.16

TABLE 4.4
Running times in seconds for the calculation of the forces and quasi-converging distributions.

the $\Theta(N \log N)$ methods `Al` and $\texttt{NMM}_\texttt{A}$ are quite large, $\texttt{NMM}_\texttt{B}$ (and $\texttt{NMM}_\texttt{C}$) outperform the other tree codes for all desired accuracies and uniform and non-uniform distributions. For quasi-converging distributions $\texttt{NMM}_\texttt{B}$ (and $\texttt{NMM}_\texttt{C}$) are nearly as fast as $\texttt{NMM}_\texttt{A}$ which, in this case, is the fastest method.

5. Summary and Future Work. We have presented a new multipole-based tree code that runs in $O(N)$ best-case and in $O(N \log N)$ worst-case running time. Our practical experiments for Coulombic systems in two dimensions indicate that the new multipole methods is faster than several popular tree codes for both uniform and highly non-uniform particle distributions. It is straightforward to extend it to three dimensions or gravitational systems. The algorithm has already been applied successfully to the visualization of large and complex networks [19]. Additionally, we plan to adopt these techniques to solve large-scale placement problems in VLSI design.

REFERENCES

[1] S. Aluru. Greengard's N-body algorithm is not order N. *SIAM Journal on Scientific Computing*, 17(3):773–776, 1996.

[2] S. Aluru, J. Gustafson, G. M. Prabhu, and F. E. Sevilgen. Distribution-Independent Hierarchical Algorithms for the N-body Problem. *Journal of Supercomputing*, 12:303–323, 1998.

[3] J. S. Bagla. Cosmological N-body simulation: Techniques, scope and status. *Current Science*, 88(7):1088–1100, 2005.

[4] L. Banjai and L. N. Trefethen. A multipole method for Schwarz-Christoffel mapping of polygons with thousands of sides. *SIAM Journal on Scientific Computing*, 25(3):1042–1065, 2003.

[5] J. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(4):446–449, 1986.

[6] J. A. Board, Z. S. Hakura, W. S. Elliot, D. C. Gray, W. J. Blanke, and J. F. Leathrum. Scalable implementations of multipole-accelerated algorithms for molecular dynamics. Technical Report 94-002, Duke University, 1994.

[7] O. Bokanowski and M. Lemou. Fast multipole method for multivariable integrals. *SIAM Journal on Numerical Analysis*, 42(5):2098–2117, 2005.

[8] P. B. Callahan and S. R. Kosaraju. A Decomposition of Multidimensional Point Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields. *Journal of the Association for Computing Machinery*, 42(1):67–90, 1995.

[9] J. Carrier, L. Greengard, and V. Rokhlin. A Fast Adaptive Multipole Algorithm for Particle Simulations. *SIAM Journal on Scientific and Statistical Computing*, 9(4):669–686, 1988.

[10] T. F. Chan, J. Cong., T. Kong, and J. R. Shinnerl. Multilevel optimization for large-scale circuit placement. In *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on computer-aided design*, pages 171–176. IEEE Press, 2000.

[11] S. Chandrasekaran and M. Gu. A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices. *Numerische Mathematik*, 96(4):723–731, 2004.

[12] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155(2):468–498, 1999.

[13] W. Dehnen. A hierarchical O(N) force calculation algorithm. *Journal on Computational Physics*, 179(1):27–42, 2002.

[14] Z. Gimbutas and V. Rokhlin. A generalized fast multipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing*, 24(3):796–817, 2003.

[15] A. Y. Grama, V. Sarin, and A. Sameh. Improving Error Bounds For Multipole-Based Treecodes. *SIAM Journal on Scientific Computing*, 21(5):1790–1803, 1998.

[16] L. Greengard and V. Rokhlin. A New Version of the Fast Multipole Method for Lapalce Equations in Three Dimensions. *Acta Numerica*, 6:229–269, 1997.

[17] L. F. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. ACM distinguished dissertations. The MIT Press, Cambridge, Massachusetts, 1988.

[18] L. F. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[19] S. Hachul and M. Jünger. Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm (Extended Abstract). In J. Pach, editor, *Graph Drawing 2004*, volume 3383 of *Lecture Notes in Computer Science*, pages 285–295. Springer-Verlag, 2005.

[20] T. Hrycak and V. Rokhlin. An improved fast multipole algorithm for potential fields. *SIAM Journal on Scientific Computing*, 19(6):1804–1826, 1998.

[21] A. C. Maggs. Dynamics of a local algorithm for simulating coulomb interactions. *Journal of Chemical Physics*, 117:1975, 2002.

[22] A. C. Maggs and V Rossetto. Local Simulation Algorithms for Coulomb Interaction. *Physical Review Letters*, 88:196402, 2002.

[23] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[24] K. Nabors and J. White. Multipole-accelerated Capacitance Extraction Algorithms for 3-D Structures with Multipole Dielectrics. *IEEE Transactions on Circuits and Systems I-Fundamental Theory and Applications*, 39(11):946–954, 1992.

[25] H. G. Petersen, D. Soelvason, and J. W. Perram. The very fast multipole method. *The Journal of Chemical Physics*, 101(10):8870–8876, 1994.

[26] A. Quigley and P. Eades. FADE: Graph Drawing, Clustering, and Visual Abstraction. In J. Marks, editor, *Graph Drawing 2000*, volume 1984 of *Lecture Notes in Computer Science*, pages 197–210. Springer-Verlag, 2001.

[27] M. C. Strain, G. E. Scuseria, and M. J. Frisch. Achieving linear scaling for the electronic quantum coulomb problem. *Science*, 271(5245):51–53, 1996.

[28] D. Tunkelang. *A Numerical Optimization Approach to General Graph Drawing*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1999. CMU-CS-98-189.

[29] M. S. Warren and J. K. Salmon. Astrophysical N-body Simulations Using Hierarchical Tree Data Structures. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, pages 570–576, 1992.

[30] G. L. Xue and M. R. Lasher. Fast evaluation of potential and force field in particle systems using a fair-split tree spatial structure. *Optimization Methods & Software*, 14(4):245–265, 2001.

[31] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, 2004.

[32] Y. H. Yuan and P. Banerjee. A parallel implementation of a fast multipole-based 3-D capacitance extraction program on distributed memory multicomputers. *Journal of Parallel and Distributed Computing*, 61(12):1751–1774, 2001.