ALTERNATIVE TILINGS FOR THE FAST MULTIPOLE METHOD ON THE PLANE

YUANCHENG LUO AND RAMANI DURAISWAMI *

Abstract. The fast multipole method (FMM) performs fast approximate kernel summation to a specified tolerance ϵ by using a hierarchical division of the domain, which groups source and receiver points into regions that satisfy local separation and the well-separated pair decomposition properties. While square tilings and quadtrees are commonly used in 2D, we investigate alternative tilings and associated spatial data structures: regular hexagons (septree) and triangles (triangle-quadtree). We show that both structures satisfy separation properties for the FMM and prove their theoretical error bounds and computational costs. Empirical runtime and error analysis of our implementations are provided.

Key words. Fast Multipole Method, Septree, Triangle Quadtree, Self-replicating Tiling, Well-separated Pair Decomposition

AMS subject classifications. 41A58, 65D18, 68U05, 52C20

1. Introduction. In the original FMM works by Greengard and Rokhlin [7], multipole expansions are executed along hierarchical centers of hypercube arrangements that span an input domain. In the two dimensional case, the center of expansions can be mapped to a set of lattice points where their Vornonoi diagrams [1] represent cell boundaries. That is, cell boundaries represent the equidistant points between nearest lattice points. For a square lattice configuration on the Euclidean plane, this equates to a square tiling and a general hypercube arrangement in higher dimensions. Composing or decomposing these tilings naturally leads to a self-replicating hierarchical quadtree data structure [11] that preserves local separation and well-separateness pair decomposition (WSPD) properties in [4]. This is necessary for the FMM to load-balance across levels and achieve linear runtime.

While the square lattice has been used by the FMM and the closely related treecode algorithm [2] since their inceptions, we are not aware of other lattice configuration having been explored. In this paper, we investigate two alternative lattice groups that may be adapted into self-similar arrangements; Regular triangular and honeycomb lattice groups translate to regular hexagonal and triangle tilings. Although hexagonal tilings have been traditionally used in image processing [10] and triangular tilings for mesh navigation [9], they have not been applied to the FMM domain. We show how these arrangements form the bases for geometry preserving septree and triangle quadtree data structures that we adopt and implement for the hierarchical 2D FMM. Computational costs and error bounds induced by the data structures are derived to be comparable to that of quadtree.

The outline of this paper is as follows: Section 2 provides an algorithmic preface of a hierarchical FMM for two dimensional coulombic equations. Section 3 illustrates separation properties that emerge from the tilings. Section 4 details a set of auxiliary functions shared amongst all data structures. Sections 5 and 6 introduce respective septree and triangle quadtree data structures and their implementations. Section 7 derives level invariant separation ratios. Section 8 provides an analysis of error bounds and computational costs. Section 9 presents empirical results that validate the theoretical analysis. Section 10 concludes the paper and remarks on the generalizability of the data structures to higher dimensions.

*Perceptual Interfaces and Reality Laboratory, Department of Computer Science, University of Maryland, College Park, MD, USA, [yluo1,ramani]@umiacs.umd.edu

2. Fast Multipole Method. The basic goal of the FMM is to evaluate the effects of a set of potentials denoted by source points x_i with strengths u_i on a set of receiver or target points y_j. A potential is approximated upto an error bound via a series of hierarchical multipole expansions arranged over the spatial domain. In section 9, the algorithm is demonstrated on the 2D coulombic potential with kernel function

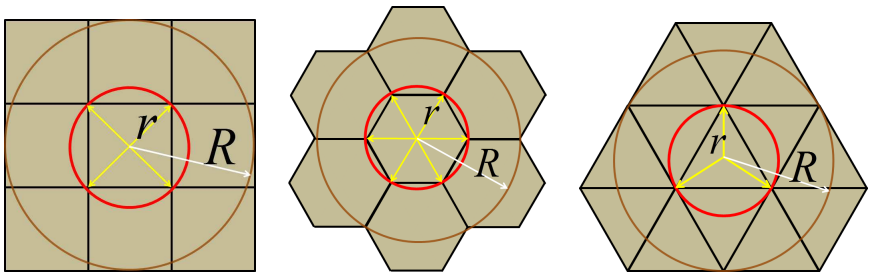
$$\Phi_{ji} = \log(y_j - x_i), \tag{2.1}$$

over the complex plane though the discussion applies to general kernels. The total evaluation at a target point j is

$$\Phi_j = \sum_{i=1}^{N} \Phi_{ji} u_i, \quad j = \{1, \ldots, m\}, \quad y, x \in \mathbb{C}. \tag{2.2}$$

An overview of the FMM algorithm is provided in [3]. For brevity, the expansion and translation operators in [12] and [6] for the coulombic kernel are omitted.

3. Separation Properties. A local separation property for multipole expansions and translations guarantees a minimum separation distance between points assigned to non-adjacent tiles. Geometrically, two circles that circumscribe and inscribe a tile and its adjacent neighbors in Figs. 3.1 bound a region. Formally, the separation ratio r/R of radii between minor and major circles induces an error in the series expansion and so affects the number of truncation terms for a lower error bound provided in section 8.



(a) Square (b) Hexagon (c) Triangle

Fig. 3.1: Minor and major radii r and major R for the local separation property affects error in multipole expansions and translations

The WSPD property defines a minimum separation distance between multipole to local (M2L) centers for translation. Formally, the distance between separated sets in Fig. 3.2 is expressed in terms of the ratio ρ/r and used to estimate the number of truncation terms.

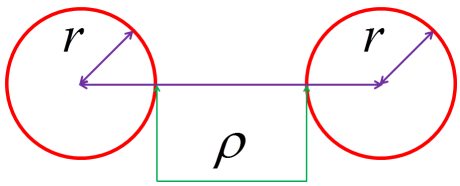
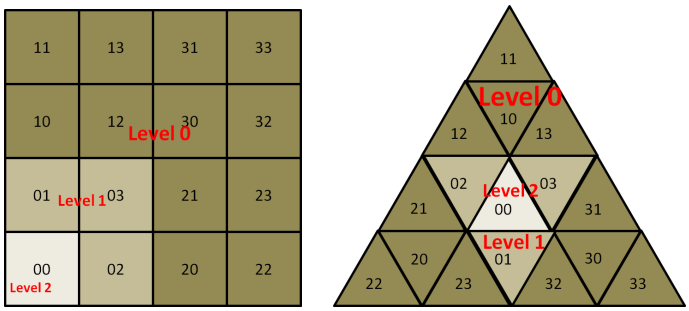


Fig. 3.2: WSPD distance ρ between lattice points affects error in M2L translations

4. Data Structures. To extend the local separation and WSPD properties beyond a basic tile, a concept of a cell is defined as a superset of tiles that shares a number of geometric and functional properties. For square and regular triangle tiles, cells are decomposable into smaller self-similar units. For square and regular hexagon tiles, cells are composable into larger aggregates. These hierarchical organizations give rise to indexable quadtree, septree, and triangle-quadtree data structures that when adapted for the FMM share the following attributes:

1. *Separation*: Cells satisfy local separation and WSPD properties across all levels.
2. *Indexing*: Cell indices satisfy some order relation in memory.
3. *Spatial addressing*: Cell centers are computable from addresses and query points can find its bounding cell.
4. *Hierarchical addressing*: Cell children, parent, and vertex neighbor indices are computable from addresses.

The separation properties for septree and triangle-quadtree are derived in section 7. The indexing property for the quadtree and triangle-quadtree follow Morton Z curves in Figs. 4.1a and 4.1b. The septree follows a spiral pattern in Fig. 5.1a. The spatial and hierarchical addressing are



(a) (b)

Fig. 4.1: Morton Z curve indexing for (a) quadtree and (b) triangle quadtree

accounted for by the following auxiliary functions:

CellCenter(n, l). Returns the point coordinates of the center of cell n on level l.

CellIndex(x, y, l). Returns a cell n on level l that contains point (x, y).

Children(n). Returns cell indices of children cells of cell n.

Parent(n). Returns a cell index of the parent of cell n.

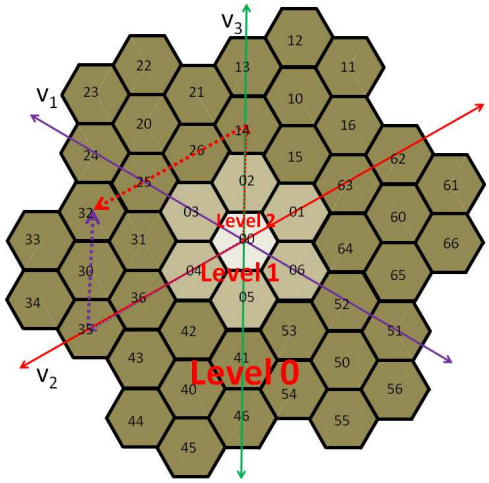
Neighbors(n, l). Returns cell indices on level l that share a vertex with cell n on level l.

NeighborsE4(n, l). Finds cell indices required for M2L translation via

$$Children[Parent(n) \cup Neighbors(Parent(n), l-1)] \cap Neighbors(n, l). \tag{4.1}$$

5. Septree. The septree data structure is an upward composable hexagonal tessellation of the Euclidean plane. While the basic hexagon units cannot be subdivided into self-similar components, they may be aggregated into larger hexagonal-like groups. A base 7 indexing scheme begins on level l_{max} with hexagon cell index 0_7 centered at the origin and adjacent to neighbors with indices $\{1, \ldots, 6\}$ tiled in a counter clock-wise direction. Aggregates of seven hexagonal cells on level l_{max} form a single cell on level $l_{max} - 1$ where they can be tiled and indexed in a similar fashion as seen

in Fig. 5.1a. Radial symmetry properties of the hexagon tessellation on level l_{max} give rise to the base 7 Generalized Balanced Ternary (GBT) [5] for cell indexing. This is investigated in section 5.1 and its properties are exploited for a number of the FMM functions.



(a)

0	**1**	**2**	**3**	**4**	**5**	**6**
1	63	15	2	0	6	64
2	15	14	26	3	0	1
3	2	26	25	31	4	0
4	0	3	31	36	42	5
5	6	0	4	42	41	53
6	64	1	0	5	53	52

(b)

Fig. 5.1: (a) Septree aggregates of hexagonal-like groups form a composable hierarchy where symmetries along the v_1, v_2, v_3 axis promote a base 7 GBT addressing scheme. (b) Table of all-pairwise GBT unit summations are analogous to vector summations to respective cells about the origin.

5.1. Generalized Balanced Ternary. The generalization of Knuth's balanced ternary notation hierarchically describes permutohedral regions of $N-$dimensional spaces. In 2D, GBT indexing and arithmetic apply to hexagonal-like cells where the data can is decomposed. One important property realized by GBT addition is its duality with vector addition. That is, index arithmetic under GBT have a spatial analog with vector arithmetic in a coordinate system illustrated in Fig. 5.1a.

For base 7 GBT unit addition, table 5.1b from Fig. 5.1a represents all pair-wise summations of unit vectors taken in each of the six neighboring cell directions. Subsequent summations of higher order cell indices are interpreted as a mapping between coordinates along axes v_1, v_2, v_3 and their base 7 cell addresses. For notation, denote index n_7 as the base 7 expansion of the usual base 10 index n and the GBT addition operator as $\circledast$, e.g. the sum of vectors to cell indices 14_7 and 35_7 along v_2 and v_3 axes is cell index $14_7 \circledast 35_7 = 32_7$.

Neighbors(n, l). The level independent neighbors of cell n are the pairwise GBT summations between cell index n_7 and the unit directions in Fig. 5.2a. To adjust for the level, cell indices greater than $7^l - 1$ are removed. The Neighbors(n, l) procedure is as follows:

1. Let $N_7 = n_7 \circledast \{1, \ldots, 6\}$
2. Return indices in N_{10} that are less than 7^l

CellCenter(n, l). The center of cell n on level l is geometrically found via a series of translations along zero-extended components of index n_7. For notation, let index $t = n_7$ and component t_1 be the left-most digit. The cell center is expressed as

$$(x, y) = \sum_{i=1}^{|t|} T(t, i), \tag{5.1}$$

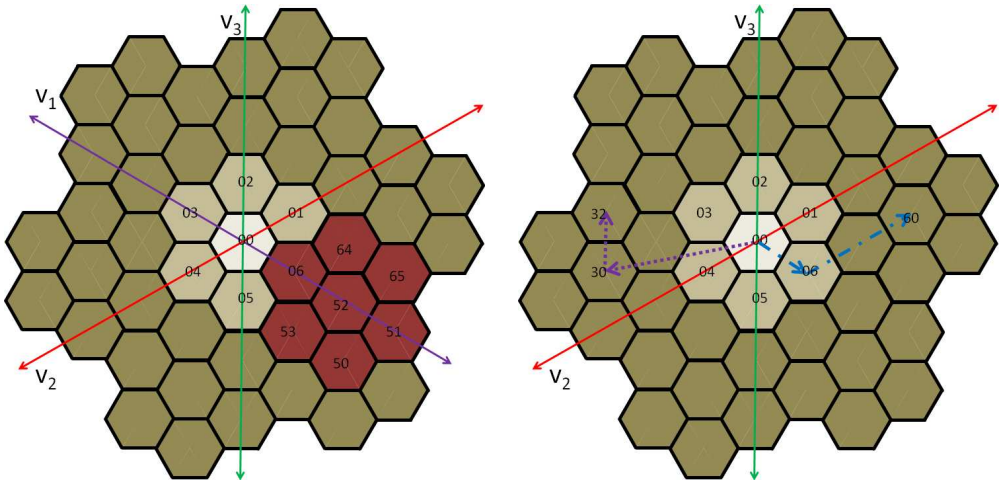


Fig. 5.2: (a) Neighbors of cell index 52_7 on level 2 are $\{65, 64, 6, 53, 50, 51\}_7$. (b) Cell center of index 32_7 found via translations along zero-extended components 30_2 and 2_7. Zero-extended index 60_7 found via $6_7 \circledast 1_7 \circledast 1_7$

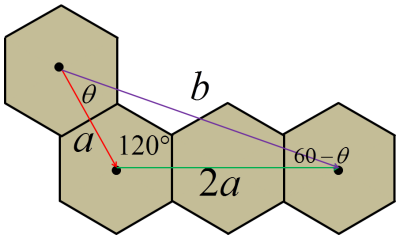
where $|t|$ is total number of components and function $T : \mathbb{Z}^2 \to \mathbb{R}^2$ extends the i^{th} component of t with $|t| - i$ zeros and returns the corresponding vector to the cell from the origin, e.g. the cell center of index 342_7 is the sum of vectors to the cell centers of indices $\{300, 40, 2\}_7$.

The zero-extended i^{th} component is the GBT summation of three shorter zero-extended components. The base case in table 5.3a shows how successive GBT summations of a unit index j and twice $1 + (j \bmod 6)$ yield a zero-extended $j0$. For $i < |t|$, zero-extending the components ignores the preceding zeros, e.g. $10_7 \circledast 20_7 \circledast 20_7 = 100_7$. This is expressed by the relation

$$t_i 0^{|t|-i} = t_i 0^{|t|-i-1} \circledast [1 + (t_i \bmod 6)]0^{|t|-i-1} \circledast [1 + (t_i \bmod 6)]0^{|t|-i-1}. \tag{5.2}$$

$1 \circledast 2 \circledast 2 = 10$
$2 \circledast 3 \circledast 3 = 20$
$3 \circledast 4 \circledast 4 = 30$
$4 \circledast 5 \circledast 5 = 40$
$5 \circledast 6 \circledast 6 = 50$
$6 \circledast 1 \circledast 1 = 60$

(a)



(b)

Fig. 5.3: (a) GBT unit addition for zero-extension. (b) Geometric analog for a zero-extension via translation $\vec{b}$

A geometric interpretation of index $t_i 0^{|t|-i}$ in Fig. 5.3b reveals that the translation $\vec{b}$ is the sum of two vectors separated by $2\pi/3$ radians with a two-to-one magnitude ratio. Solving for the angle of separation and the magnitude of vector $\vec{b}$ in appendix eqs. D.1 and D.2 yield $\theta = \arctan \frac{\sqrt{3}}{2}$ and $b = a\sqrt{7}$. Hence, successive zero-extensions of a component t_i are rotated by θ radians with magnitude $a_{i+1} = a_i\sqrt{7}$ with a base magnitude $a_0 = r\sqrt{3}$.

The function $T(t,i)$ can now be expressed in polar coordinates

$$\begin{aligned} j &= (|t| - i) + (l_{max} - l), \\ \theta_i &= j \arctan\frac{\sqrt{3}}{2} + t_i\frac{\pi}{3} - \frac{\pi}{6}, \quad R_i = r\sqrt{3}\left(\sqrt{7}\right)^j, \end{aligned} \tag{5.3}$$

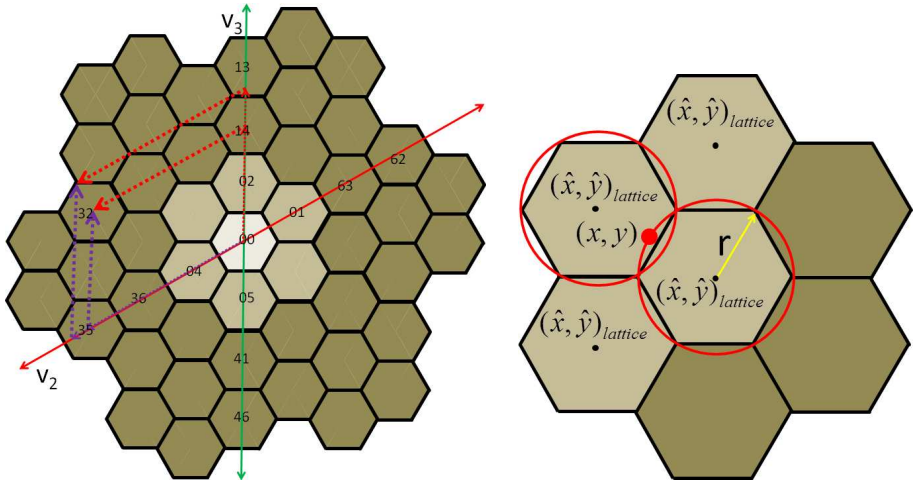
before $T(t,i) = (R_i\cos\theta_i, R_i\sin\theta_i)$ gives the translation in a Cartesian coordinate system. The CellCenter(n, l) procedure is as follows:

1. Let $(x,y) = (0,0)$, $i = 1$
2. Update $(x,y) = (x,y) + (R_i\cos\theta_i, R_i\sin\theta_i)$ from eq. 5.3
3. Increment i and repeat from step 2 until $i = l$
4. Return point (x,y)

CellIndex(x, y, l). An approximation of the cell that contains the query point (x,y) is found via a change of basis

$$(b,c) = \left(\frac{2x}{3r}, \frac{y\sqrt{3} - x}{3r}\right), \tag{5.4}$$

where coordinates (b,c) are defined along axes v_2, v_3. These associated cells along the axes are erroneous as points near the corners of true bounding hexagon may project incorrectly in Fig. 5.4a.



(a) (b)

Fig. 5.4: (a) Transforming query point $(x,y) \to (b,c)$ via change of basis may lead to incorrect cells along v_2, v_3 axes. (b) Hexagonal edges are Voronoi diagrams of equilateral triangular lattice points that construct cell centers

To address this issue, boundary conditions are checked by observing that hexagonal edges are the Voronoi diagrams of equilateral triangular lattice points. That is, hexagonal edges represent points equidistant between nearest lattice points. The query point (x,y) that falls within a hexagon would be the nearest neighbor of its center or lattice point. Furthermore, the query point may be bound between four neighbouring candidate lattice points in Fig. 5.4b. If the lattice points are defined w.r.t. axes v_2, v_3 as

$$(\hat{x},\hat{y})_{lattice} = \left(\frac{3r\hat{b}}{2}, \frac{r\sqrt{3}(2\hat{c} + \hat{b})}{2}\right), \tag{5.5}$$

then the candidate lattice points have coordinates

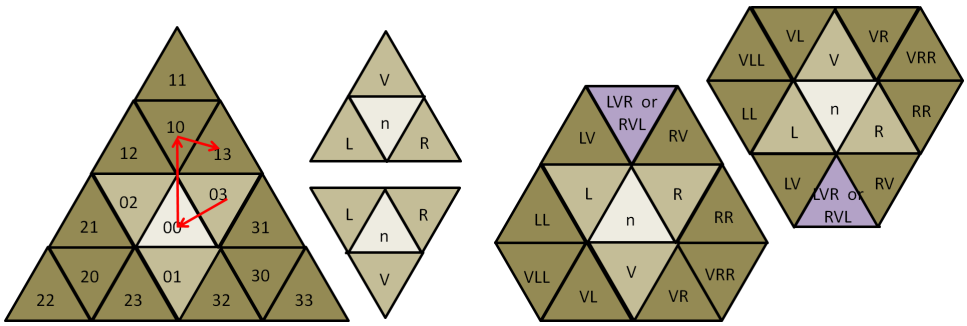
$$\{(\lfloor b \rfloor, \lfloor c \rfloor), (\lceil b \rceil, \lfloor c \rfloor), (\lfloor b \rfloor, \lceil c \rceil), (\lceil b \rceil, \lceil c \rceil)\}. \tag{5.6}$$

Finding the nearest lattice point w.r.t. the query point (b, c) after converting back to Cartesian coordinates via eq. 5.4 yields the hexagon and its respective cell index $t = n_7$. The first l components or $t_{1:l}$ is the the cell index on level l. The CellIndex(x, y, l) procedure is as follows:

1. Compute axes coordinates (b, c) from eq. 5.4
2. Compute $i = 1 : 4$ candidates lattice axes coordinates $(\hat{b}, \hat{c})_i$ from eq. 5.6
3. Convert candidates lattice coordinates to Cartesian coordinates $(\hat{x}, \hat{y})_i$ with eq. 5.4
4. Let the nearest candidate lattice point w.r.t. query point (x, y) be $(\tilde{x}, \tilde{y})$ and equivalently $(\tilde{b}, \tilde{c})$
5. Find cell indices u_7, v_7 for coordinates $\tilde{b}$, $\tilde{c}$ along axes v_2, v_3
6. Return $u \circledast v$

6. Triangle Quadtree. A variation of the original quadtree data structure consists of a regular triangular tessellation of the Euclidean plane that is strictly decomposable. While each triangle may be subdivided into four similar units (center, vertical, left, right), a group of triangles is not upward composable in the sense that vertical orientations of descendant triangles depend on the orientation of their common ancestor triangle or root. That is, all children of a center cell will have inverted its orientation. For notation, denote the default upright orientation as $up_i = 1$ and the inverted orientation as $up_i = -1$ for a triangle on level i. Triangle quadtree indexing begins on level $i = 0$ with cell index 0 centered at the origin and enclosing the entire domain. Children cells on level $i+1$ with indices $\{0, \ldots, 3\}$ are associated with cell types {center, vertical, left, right} ordered w.r.t. the parent cell.

Neighbors(n, l). For regular quadtrees, neighbors of a cell index n need only to share an edge. For triangle quadtrees, denote the left, right, and vertical adjacent neighbors as $\{L, R, V\}$ in Fig. 6.1a.



(a) (b)

Fig. 6.1: (a) Neighbor finding from cell index 03 in the vertical direction. Cell 0 is nearest ancestor with vertical sibling cell 1 on level 1. Reflect the path leading to ancestor cell 0 gives cell index 13. (b) Neighbor finding from recursive calls to adjacent neighbors

Similar to [9], the adjacent neighbor of cell n in direction k is found with an ancestor-sibling-reflect method in the adjacentNeighbor(n, k) procedure:

1. Search for the closest common ancestor containing a sibling in direction k. For notation, let cell index $t = n_4$ and matrix D in table 6.1 map cell source types and search directions to destination cell types. From right to left components t_i, find the first valid entry $D_{t_i,k}$.

Source cell type	Left cell ID	Right cell ID	Vert cell ID
0	2*	3*	1*
1	3	2	0*
2	1	0*	2
3	0*	1	3

Table 6.1: Mapping from source cell types and search directions to destination cell types. * entries indicate valid siblings of a source cell

2. Move to sibling node. Replace component t_i with entry $D_{t_i,k}$.
3. Reflect the path taken to reach ancestor. For $j = \{(i+1), \ldots, l_{max}\}$, replace component t_j with entry $D_{t_j,k}$.
4. Return cell index t

To find all neighbors that share a vertex with cell n, make a set of recursive calls to adjacentNeighbor(n, k) in Fig. 6.1b. The Neighbors(n, l) procedure is as follows:

1. Let $\{L, R, V\}$ be adjacent neighbors of cell n in directions left, right, vertical
2. Let $\{VL, VR, LL, RR, LV, RV\}$ be adjacent neighbors of cells V, L, R in directions left, right, vertical
3. Let $\{VLL, VRR, LVR\}$ be adjacent neighbors of cells VL, VR, LV in directions left and right
4. Return cell indices $\{L, R, V, VL, VR, LL, RR, LV, RV, VLL, VRR, LVR\}$

CellCenter(n, l). The center of cell n on level l is found via a series of translations for each component of index n_4. Let index $t = n_4$, component t_1 be the left-most digit, and cell center

$$(x, y) = \sum_{i=1}^{|t|} F(t, i), \tag{6.1}$$

where function $F : \mathbb{Z}^2 \to \mathbb{R}^2$ returns a vector to cell index t_i that is adjusted for level and orientation, e.g. the cell center of index 302_4 is the sum of vectors to cell centers of indices $\{3, 0, 2\}_4$ on levels $\{1, 2, 3\}$ with orientations $\{1, -1, -1\}$.

To determine the orientation of index t on level l, observe that the orientation only flips for center cells in Fig. 6.2a. This is the zero-parity of cell index t (even is 0, odd is 1) adjusted for level l and the number of components $|t|$. The function

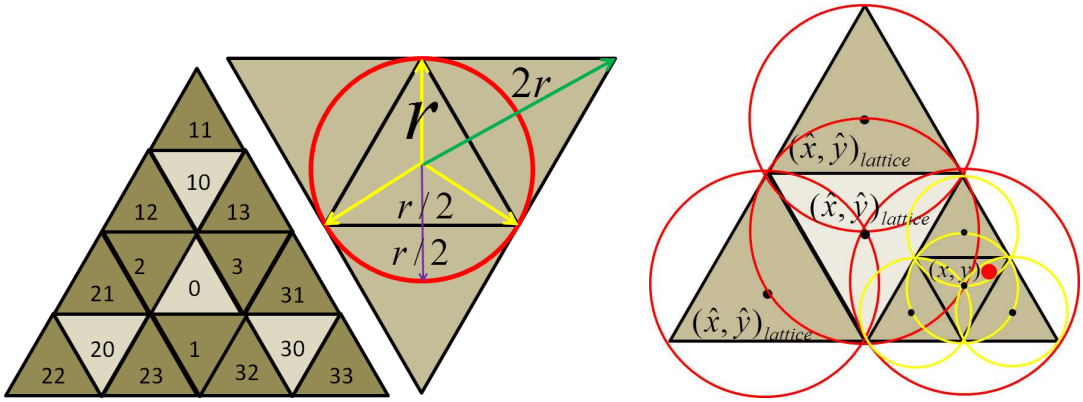
$$\text{isUp}(t, l) = 2[(\text{parity}(t) + l - |t| + 1) \bmod 2] - 1, \tag{6.2}$$

returns 1 for the up orientation, and -1 for the inverted orientation. To determine the orientation of component t_i, make a query to $\text{isUp}(t_{1:i}, l - |t| + i)$ where cell index $t_{1:i}$ is the level adjusted ancestor.

Note that triangle centers for components $t_i = 0$ are equivalent to its ancestor's center as no translations are necessary. The function $F(t, i)$ can now be expressed in polar coordinates

$$\begin{aligned} &j = l - |t| + i, \quad 1 \le t_i \le 3, \\ &\theta_i = \text{isUp}(t_{1:i}, j) \left(\frac{\pi}{2} + \frac{2(t_i - 1)\pi}{3} \right), \quad R_i = 2^{l_{max} - j} r, \end{aligned} \tag{6.3}$$

where radius R is scaled by a factor of two per level in Fig. 6.2a. The CellCenter(n, l) procedure is as follows:



(a) (b)

Fig. 6.2: (a) Triangle orientation determined by cell index zero-parity. Translation magnitudes to adjacent neighbors scale by a factor of two per level. (b) Triangle edges are Voronoi diagrams of hexagonal lattice points at cell centers. A cell index is found via searching nearest lattice points within a current triangle.

1. Let current center coordinates $(x, y) = (0, 0)$, $i = 1$
2. Update center $(x, y) = (x, y) + (R_i \cos\theta_i, R_i \sin\theta_i)$ from eq. 6.3
3. Increment i and repeat from step 2 until $i = l$
4. Return (x, y)

CellIndex(x, y, l). A cell index $t = n_7$ contains point (x, y) on level l if all ancestor cell indices $t_{1:l}$ also contain it. Since triangles descendants are contained within ancestor triangles, a search proceeds from level 0 to l s.t. successive descendant triangles contain point (x, y).

Observe that triangle edges form Voronoi diagrams of regular hexagonal lattice points. That is, the triangle edges represent points equidistant between nearest lattice points. Hence, a query point (x, y) that falls within a triangle is also the nearest neighbor of the triangle's center or lattice point in Fig. 6.2b.

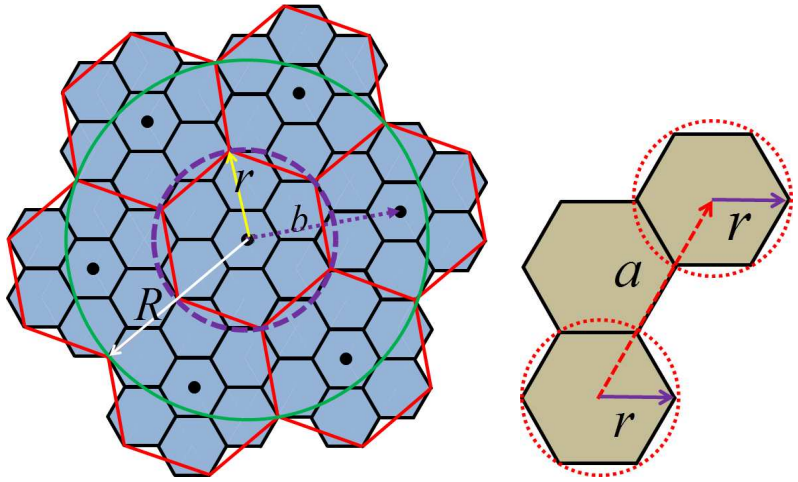
To track the progress of triangle centers, denote point (u, v) as the current center. Note that triangle centers for components $t_i = 0$ are equivalent to its ancestor's center. That is, if a center triangle is found to be the nearest neighbor, then the current center remains unchanged while the vertical orientation *up* flips. In polar coordinates, denote the three triangle lattice points about the origin as

$$\theta_i = up_i \left(\frac{\pi}{2} + \frac{2(t_i - 1)\pi}{3} \right), \quad R_i = 2^{l_{max} - i} r, \quad 1 \leq t_i \leq 3, \tag{6.4}$$

where similar to the CellCenter function, the radius R is scaled by a factor of two per level. The CellIndex(x, y, l) procedure is as follows:

1. Set to origin the current center $(u, v) = (0, 0)$, let $i = 1$
2. Assign component t_i the cell type with the nearest lattice to query point (x, y) on level i
3. Update current center $(u, v) = (u, v) + (R_i \cos\theta_i, R_i \sin\theta_i)$ from eq. 6.4
4. If $t_i = 0$, then flip orientation $up_i = -up_{i-1}$
5. Increment i and repeat from step 2 until $i = l$
6. Return t

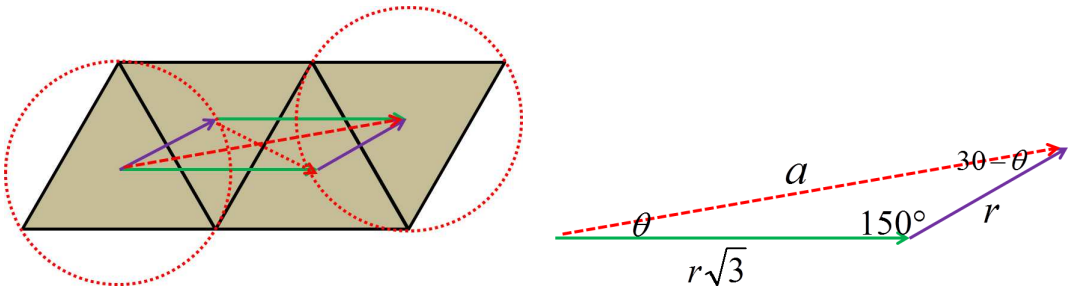
7. Separation Ratios. To show that septree local separation and WSPD ratios are level invariant, consider the base case in Fig. 3.1b where ratio $r/R = 1/2$. For successive levels, lattice points induce Voronoi diagrams that form rotated and scaled hexagonal boundaries that intersect points along minor and major radii r and R in Fig. 7.1a. The rotation and scaling are proportional to eq. 6.3 adjusted for level i. The WSPD distance $\rho = r$ in Fig. 7.1b and local separation ratio are preserved as the induced hexagonal boundaries are self-similar to the base case.



(a) (b)

Fig. 7.1: (a) Local separation ratio r/R drawn from level 1 lattice points and their induced hexagonal Voronoi diagrams. Either radius r or R is computable from magnitude b obtained from eq. 6.3. (b) WSPD distance $\rho = 3r - a = r$ generalized over level induced hexagonal boundaries

To show that the triangle-quadtree local separation ratio is level invariant, observe that ratio $r/R = 1/2$ for the same level in Fig. 6.2a. On successive levels, a major radius R is equivalent to the minor radius r from the preceding level and can be computed from eq. 6.4. The WSPD ratio $\rho/r = \sqrt{7} - 2$ in Fig. 7.2 is derived in appendix E. Similar to the quadtree, the level invariance stems from self-similar tiling arrangements with the base case in Fig. 3.1c.



(a) (b)

Fig. 7.2: (a) Triangle tilings for nearest M2L translation. (b) Geometric analysis for WSPD distance

8. Cost Analysis. To estimate the costs of a hierarchical FMM, a uniform source and target point distribution over the data structure's geometric domain is assumed [8]. The total computational costs w.r.t. the quadtree, septree and triangle quadtree are derived in appendices A, B and

Struct	$P4$	$P2$	r/R	ρ/r	S_{opt}	**Opt Costs**
Q-tree	27	9	$\frac{\sqrt{2}}{3}$	$2(\sqrt{2}-1)$	$p\left(\frac{116N}{27M}\right)^{.5}$	$(M+N+32.64(MN)^{.5})p$
S-tree	42	7	$\frac{1}{2}$	1	$p\left(\frac{308N}{42M}\right)^{.5}$	$(M+N+35.2(MN)^{.5})p$
TQ-tree	39	13	$\frac{1}{2}$	$\sqrt{7}-2$	$p\left(\frac{164N}{39M}\right)^{.5}$	$(M+N+46.65(MN)^{.5})p$

Table 8.1: FMM properties for quadtree, septree and triangle quadtree where p is the number of truncation terms for each expansion, P4 is number of M2L translations per cell, P2 number of cells in neighborhood, ρ is the multipole to local separation distance and S_{opt} the optimal number of points per cell

C. The overall costs depend on a density quantity S_{opt} of source and target points per cell. This density quantity is a function of the number of M2L translations $P4$ and the number of neighbor cells $P2$ which is optimized by equating the number of M2L translations to the number of direct evaluations per cell. Substituting density S_{opt} back into the FMM total costs yields the optimal FMM costs listed in table 8.1.

The total FMM costs also depend on the number of truncation terms p which are chosen a priori to guarantee a minimum error bound. This quantity depends on the kernel expansion and translation formulations as specified in [12]. Maximum absolute error bounds [6] are written in terms of local separation ratio r/R and WSPD ratio ρ/r between M2L translations. For a multipole expansion and translation, the absolute error is bounded by

$$|\epsilon_p| \leq \frac{\sum |u_i|_{i=1}^{k}}{R-r}\left(\frac{r}{R}\right)^{p+1}. \tag{8.1}$$

For a M2L translation, the absolute error is bounded by

$$|\epsilon_p| \leq \frac{\sum |u_i|_{i=1}^{k}}{\rho}\left(\frac{1}{1+\rho/r}\right)^{p+1}. \tag{8.2}$$

Last, the local to local translations are exact.

For a fixed number of truncation terms, the total optimized cost for septree is slightly greater than that of quadtree as the lower neighbor count does not fully compensate for the greater number of M2L translations per cell. The triangle quadtree is less cost efficient in both categories and so yields a larger leading coefficient term. For multipole expansions, the larger local separation ratios r/R in both septree and triangle quadtree suggest wider error bounds than that of quadtree. For M2L translations, the smaller WSPD ratio ρ/r for septree suggests a tighter error bound than that of both quadtree and triangle-septree. This last property may have considerations in setting the maximum l_{max} of the data structure.

9. Experiments. To validate the theoretical FMM costs and error bounds from section 8, a set of test cases suitable for uniform conditions for each data structure is generated. That is, the input domain consist of S_{opt} uniformly random source and target points fitted to each cell. A regular polygon point picking method in Fig. 9.1 handles the various tile geometries. For a regular cell, the polygon is subdivided into disjoint isosceles triangles per edge where halves of triangles are rearranged to form rectangles. Uniformly random points (x, y) are picked from this rectangle and a third uniformly random variable z assigns the point to one of the isosceles triangles. To modify the total number of source and target points, the total number of occupied cells is reduced.

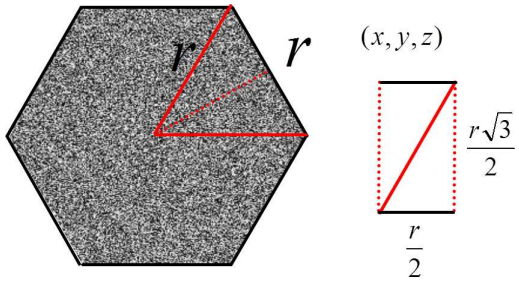


Fig. 9.1: Regular polygon point picking via parameterization into uniform variables (x, y, z) where a point (x, y) is uniformly sampled from a rectangle while uniformly random variable z chooses the triangle

For experiments, both runtime and error estimates of the FMM are considered while fixing either the number of source and target points (N, M), or the number of truncation terms p. Runtime comparisons w.r.t. the direct evaluation method are shown when appropriate. The implementation is written and tested on Matlab 2010b and Intel i7-2630QM hardware.

The runtime results for a variable number of source and target points in Fig. 9.2 match the theoretical estimates. All three hierarchical FMM data structures obtain linear asymptotic runtimes. The basic quadtree data structure outperforms the septree by a small margin for both max levels l_{max}. The triangle quadtree performs the slowest out of the three. The direct method obtains a quadratic runtime and where an estimated cross-over point between the number of source and target points is between $10^{3.5}$ to $10^{3.75}$. The error results in Fig. 9.3 indicate a linear increase in error w.r.t. the number of source and target points N and M. The quadtree obtains the least error presumably due to the smallest local separation ratio. One explanation for the greater maximum error in the septree compared to the triangle quadtree is the larger number of corner points in the tiling where source-target point distances are minimized.

The runtime results for a variable number of truncation terms p in Fig. 9.4 reveal a cross-over point between septree and triangle-quadtree. This may be due to the greater number of M2L translations in the septree and the inefficiencies in the implementation where the translation matrices are computed on the fly. The error results in Fig. 9.5 exhibit an exponential error loss for increasing truncation terms.

10. Conclusions. In this paper, we have shown that regular hexagonal and triangle tilings of the Euclidean plane generate septree and triangle-quadtree data structures that satisfy local separation and WSPD properties for the FMM. We derived their implementations and respective geometric properties with regards to FMM error bounds and costs. The empirical results validated the theoretical claims and have shown to be comparable to the original quadtree. While the quadtree remained ideal for a uniformly distributed data set, both the septree and triangle-quadtree may have applications for non-uniform domains, especially when the input is clustered about respective lattice points.

As a final remark, the septree's base hexagon unit is an order-3 permutohedron. Permutohedrons are $(d-1)$ dimensional polytopes embedded in d dimensional space that can tessellate the domain and is indexable via a similar GBT addressing scheme. Future work may investigate constructions of a permutohedron-tree and its separation properties in higher dimension.

Acknowledgements. We would like to thank Dr. Nail Gumerov for his lectures in the fast mutlipole methods course at the University of Maryland, College Park and partial support from

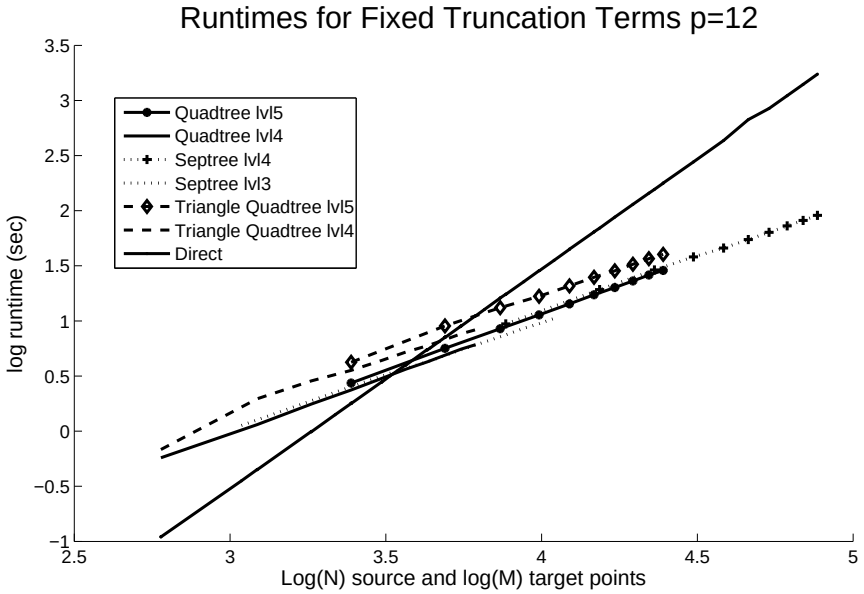


Fig. 9.2: Runtime (seconds) for FMM data structures with variable number of source N and target M points and fixed truncation terms $p = 12$

the ONR Office of Naval Research under the MURI grant N00014-08-10638.

REFERENCES

[1] F. Aurenhammer, *A survey of a fundamental geometric data structure*, ACM Computing Surveys, 23 (1991), pp. 345–405.
[2] J. Barnes and P. Hut, *A hierarchical O(N log N) force-calculation algorithm*, Nature, 324 (1986), pp. 446–449.
[3] R. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods and Elliptic PDEs, M. Ainsworth, J. Levesley, W. Light, and M. Marletta, eds., Oxford University Press, 1997, pp. 1–37.
[4] P. Callahan and S. Kosaraju, *A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields*, ACM, 42 (1995), pp. 67–90.
[5] L. Gibson and D. Lucas, *Spatial data processing using generalized balanced ternary*, 1982, pp. 566–571.
[6] L. Greengard, *The rapid evaluation of potential fields in particle systems*, MIT Press, Cambridge, Massachusettes, 1998.
[7] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, Journal of Computational Physics, 73 (1987), pp. 325–348.
[8] N.A. Gumerov, R. Duraiswami, and E.A. Borovikov, *Data structures, optimal choice of parameters, and complexity results for generalized multilevel fast multipole methods in d dimensions*, Tech. Report CS-TR-4458, University of Maryland Institute for Advanced Computer Studies, 2003.
[9] M. Lee and H. Samet, *Navigating through triangle meshes implemented as linear quadtrees*, ACM Transactions on Graphics, 19 (2000), pp. 79–121.
[10] L. Middleton and J. Sivaswamy, *Hexagonal Image Processing: A Practical Approach*, Springer, 2005.
[11] H. Samet, *The quadtree and related hierarchical data structures*, ACM Computing Surveys, 16 (1984), pp. 187–260.
[12] Y. Wang, *The fast multipole method for 2D coulombic problems: Analysis, implementation and visualization*, master's thesis, University of Maryland Institute for Advanced Computer Studies, 2005.

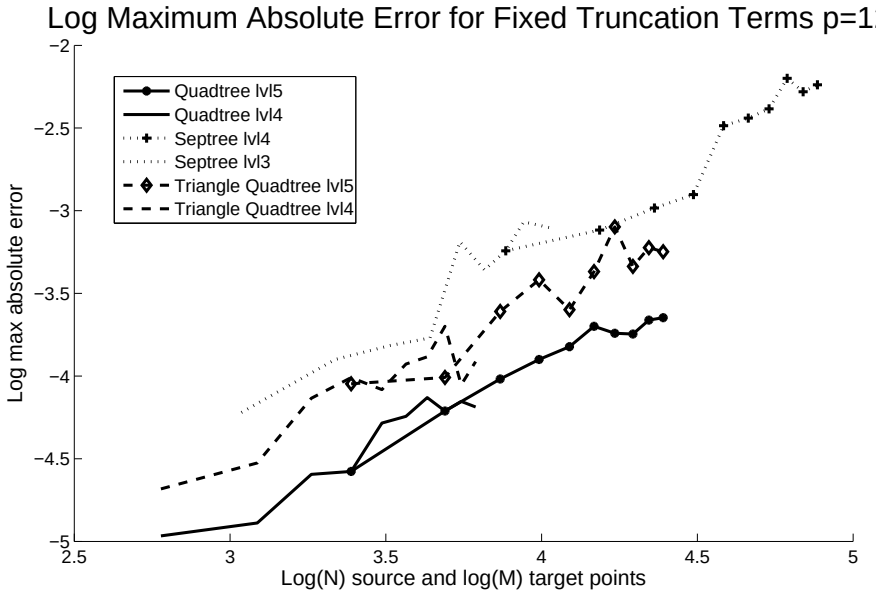


Fig. 9.3: Maximum absolute errors for FMM data structures with variable number of source N and target M points and fixed truncation terms $p = 12$

Appendix A. FMM Quadtree Costs. FMM evaluation costs before and after substituting optimal point density per cell for quadtree data structure in eqs. A.1 and A.2.

$$N = 2^{L_* d}, \quad S = 2^{L_s d}, \quad L = L_* - L_s, \quad K = 2^{dL},$$
$$P_4(d) = 3^d(2^d - 1), \quad P_2(d) = 3^d.$$

$$\begin{aligned} \text{cost(FMM)} &= (M+N)P + \left(K\frac{2^d}{2^d-1}(P_4(d)+2) - \frac{2^{3d+1} - P_4(d)2^{2d}}{2^d-1} \right) P^2 \\ &\quad + M(P_2(d)s + P). \end{aligned} \tag{A.1}$$

$$\begin{aligned} \text{opt cost} &= (M+N)P + \left(\sqrt{\frac{2^d}{2^d-1}} + \sqrt{\frac{2^d-1}{2^d}} \right) (MN(3^d(2^d-1)+2)3^d)^{.5} P \\ &\approx \left(M + N + 32.64(MN)^{.5}\right) P, \quad d = 2. \end{aligned} \tag{A.2}$$

Appendix B. FMM Septree Costs. FMM evaluation costs before and after substituting optimal point density per cell for septree data structure in eqs. B.1 and B.2.

$$N = 7^{L_*}, \quad S = 7^{L_s}, \quad L = L_* - L_s, \quad K = 7^L, \quad P_4 = 42, \quad P_2 = 7.$$

$$\text{cost(FMM)} = (M+N)P + \left(K\frac{7}{6}(P_4+2) - \frac{7^4 - P_4 7^2}{6} \right) P^2 + M(P_2 s + P). \tag{B.1}$$

$$\begin{aligned} \text{opt cost} &= (M+N)P + \left(\sqrt{\frac{7}{6}} + \sqrt{\frac{6}{7}} \right) (308MN)^{.5} P \\ &\approx \left(M + N + 35.2(MN)^{.5}\right) P. \end{aligned} \tag{B.2}$$

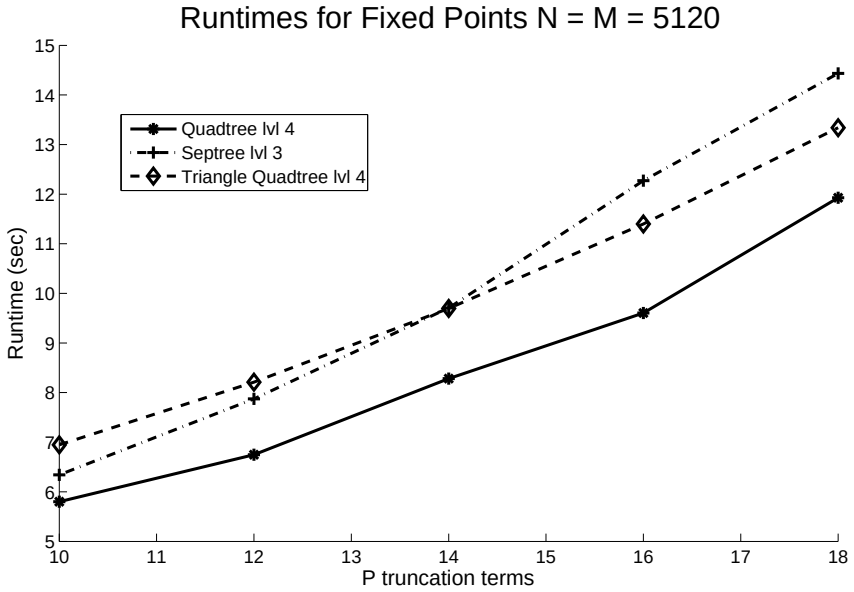


Fig. 9.4: Runtime (seconds) for FMM data structures with variable number of truncation terms p and fixed number source and target points $N = M = 5120$

Appendix C. FMM Triangle Quadtree Costs. FMM evaluation costs before and after substituting optimal point density per cell for triangle quadtree data structure in eqs. C.1 and C.2.

$$N = 4^{L_*}, \quad S = 4^{L_s}, \quad L = L_* - L_s, \quad K = 4^L, \quad P_4 = 39, \quad P_2 = 13.$$

$$\text{cost(FMM)} = (M+N)P + \left(K\frac{4}{3}(P_4+2) - \frac{4^4 - P_4 7^2}{3}\right)P^2 + M(P_2 s + P). \tag{C.1}$$

$$\begin{aligned} \text{opt cost} &= (M+N)P + \left(\sqrt{\frac{4}{3}} + \sqrt{\frac{3}{4}}\right)(533MN)^{.5}P \\ &\approx \left(M + N + 46.65(MN)^{.5}\right)P. \end{aligned} \tag{C.2}$$

Appendix D. GBT Translations. Septree GBT translations for zero-extending indices. The angular separation θ in Fig. 5.3b is

$$\frac{\sin\theta}{2a} = \frac{\sin 120}{b} = \frac{\sin(60-\theta)}{a} \Rightarrow \theta = \arctan\frac{\sqrt{3}}{2}. \tag{D.1}$$

The magnitude of translation vector $\vec{b}$ in Fig. 5.3b is

$$b \sin\arctan\frac{\sqrt{3}}{2} = b\frac{\frac{\sqrt{3}}{2}}{\sqrt{1+\left(\frac{\sqrt{3}}{2}\right)^2}} = a\sqrt{3} \Rightarrow b = a\sqrt{7}, \quad a_0 = r\sqrt{3}. \tag{D.2}$$

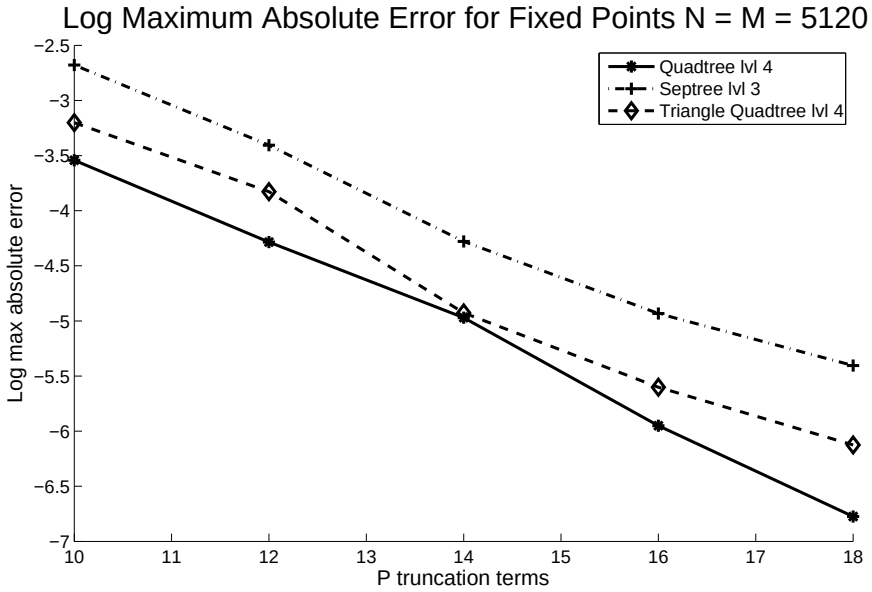


Fig. 9.5: Maximum absolute errors for FMM data structures with variable number of truncation terms p and fixed number source and target points $N = M = 5120$

Appendix E. Triangle-quadtree WSPD. The min M2L translation distance ρ for triangle quadtree in Fig. 7.2 is

$$\begin{aligned} \frac{\sin\theta}{r} = \frac{\sin 150}{a} = \frac{30-\theta}{r\sqrt{3}} \Rightarrow \theta = \arctan\frac{\sqrt{3}}{9}, \\ \frac{r}{2} = a \sin\arctan\frac{\sqrt{3}}{9} = a\frac{\frac{\sqrt{3}}{9}}{\sqrt{1+\left(\frac{\sqrt{3}}{9}\right)^2}} = a\frac{1}{2\sqrt{7}}, \\ a = r\sqrt{7}, \quad \rho = r(\sqrt{7}-2). \end{aligned} \tag{E.1}$$