Fast multipole method and its extensions

Yue Sun

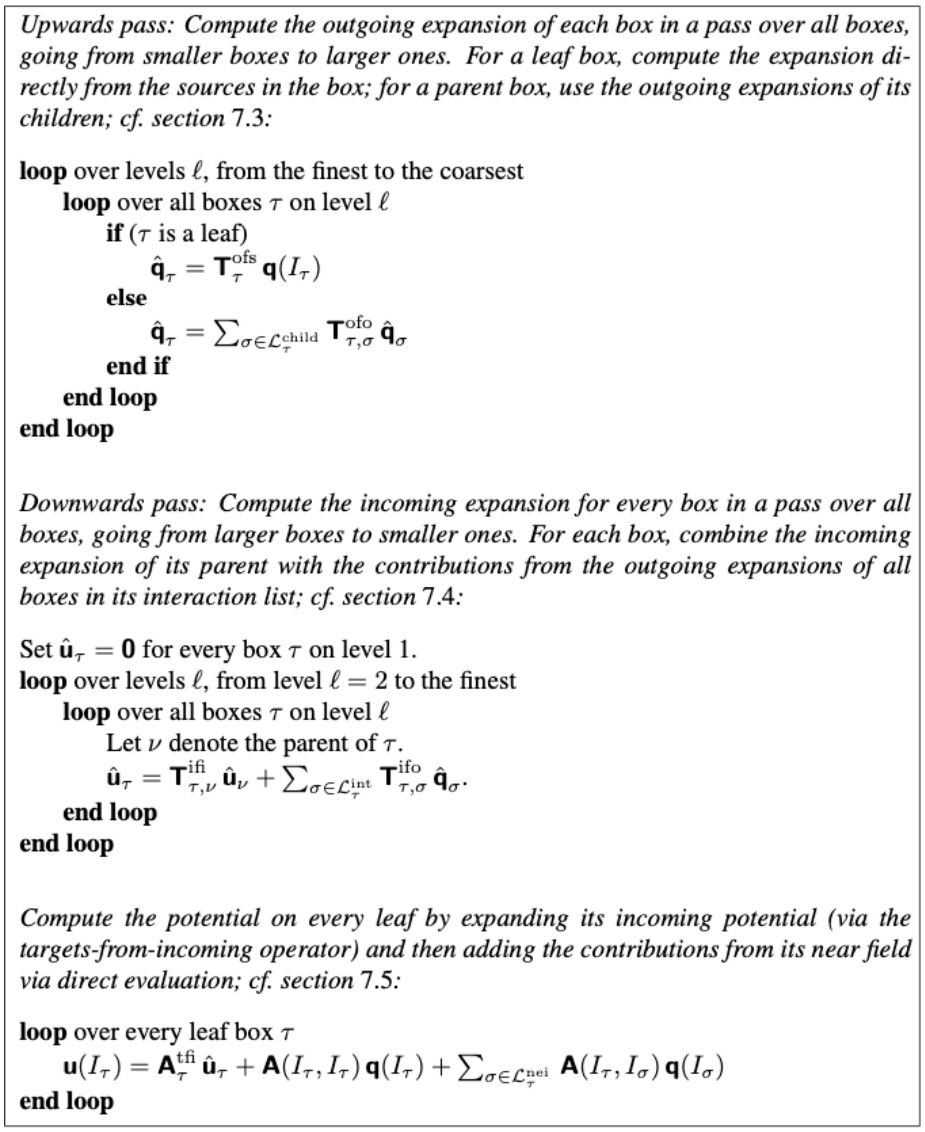
The fast multipole method is developed to evaluate the pairwise interactions between a given set of N electrically charged particles. The problem can be written as

$$u_i = \sum_{j=1}^{\infty} G(x_i, x_j)q_j, i = 1,2......N$$

Where $\{x_i\}_{i=1}^N$ is the point locations, $\{q_i\}_{i=1}^N$ is the corresponding sources, $\{u_i\}_{i=1}^N$is the potential values. The kernel function is represented as $G(x, y) = |log(x - y)|$. In my code implementation I use complex form $G(x, y) = Re(log(x - y))$.

1. Classic FMM Algorithm

In this section, we will first introduce the FMM algorithm.



Upwards pass: Compute the outgoing expansion of each box in a pass over all boxes, going from smaller boxes to larger ones. For a leaf box, compute the expansion directly from the sources in the box; for a parent box, use the outgoing expansions of its children; cf. section 7.3:

```
loop over levels $\ell$, from the finest to the coarsest
    loop over all boxes $\tau$ on level $\ell$
        if ($\tau$ is a leaf)
            $\hat{\mathbf{q}}_\tau = \mathbf{T}_\tau^{\mathrm{ofs}}\,\mathbf{q}(I_\tau)$
        else
            $\hat{\mathbf{q}}_\tau = \sum_{\sigma\in\mathcal{L}_\tau^{\mathrm{child}}} \mathbf{T}_{\tau,\sigma}^{\mathrm{ofo}}\,\hat{\mathbf{q}}_\sigma$
        end if
    end loop
end loop
```

Downwards pass: Compute the incoming expansion for every box in a pass over all boxes, going from larger boxes to smaller ones. For each box, combine the incoming expansion of its parent with the contributions from the outgoing expansions of all boxes in its interaction list; cf. section 7.4:

```
Set $\hat{\mathbf{u}}_\tau = \mathbf{0}$ for every box $\tau$ on level 1.
loop over levels $\ell$, from level $\ell = 2$ to the finest
    loop over all boxes $\tau$ on level $\ell$
        Let $\nu$ denote the parent of $\tau$.
        $\hat{\mathbf{u}}_\tau = \mathbf{T}_{\tau,\nu}^{\mathrm{ifi}}\,\hat{\mathbf{u}}_\nu + \sum_{\sigma\in\mathcal{L}_\tau^{\mathrm{int}}} \mathbf{T}_{\tau,\sigma}^{\mathrm{ifo}}\,\hat{\mathbf{q}}_\sigma$.
    end loop
end loop
```

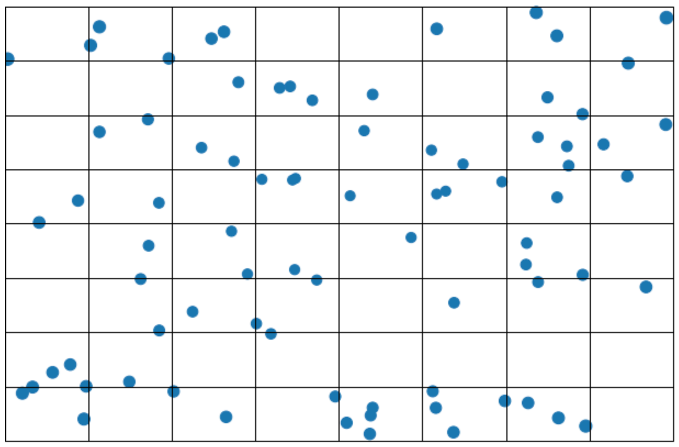
Compute the potential on every leaf by expanding its incoming potential (via the targets-from-incoming operator) and then adding the contributions from its near field via direct evaluation; cf. section 7.5:

```
loop over every leaf box $\tau$
    $\mathbf{u}(I_\tau) = \mathbf{A}_\tau^{\mathrm{tfi}}\,\hat{\mathbf{u}}_\tau + \mathbf{A}(I_\tau, I_\tau)\,\mathbf{q}(I_\tau) + \sum_{\sigma\in\mathcal{L}_\tau^{\mathrm{nei}}} \mathbf{A}(I_\tau, I_\sigma)\,\mathbf{q}(I_\sigma)$
end loop
```

1.1 Implementation of FMM

1.1.1 Tree of boxes

To execute this algorithm, First I need introduce the hierarchical tree of boxes. I fix the size of each box and label the levels using the integers l=1,2,3,….L. l=0 denotes the root and l=L dentist the leaf. The tree of boxes of level 3 can be visualized as below, it means we split the box equally into 64 smaller boxes no matter how many dots are there in each box.



1.1.2 Tanslation operators

Second I will define all the operator needed in this algorithm.

T^{ofs}:

$$\begin{cases} \hat{q}_0^{\tau} = \sum_{j\in I_\tau} q_j \\ \hat{q}_p^{\tau} = \sum_{j\in I_\tau} -\frac{1}{p}(x_j - c_\tau)^p q_j, \text{ p=1,2,...P-1} \end{cases}$$

Where I_τ denote a list of all the points x_j such that $x_j \in \Omega_\tau$, c_τ is the center of the box.

T^{ofo}:

$$\begin{cases} \binom{r}{s}(\boldsymbol{c}_\sigma - \boldsymbol{c}_\tau)^{r-s} & \textit{when } s \le r, \\ 0 & \textit{when } s > r. \end{cases}$$

T^{ifo}:

$$\begin{cases} \hat{v}_0^{\tau} = \hat{q}_0^{\sigma} \log(\boldsymbol{c}_\tau - \boldsymbol{c}_\sigma) + \sum_{p=1}^{\infty} \hat{q}_p^{\sigma} (-1)^p \frac{1}{(\boldsymbol{c}_\sigma - \boldsymbol{c}_\tau)^p}, \\ \hat{v}_r^{\tau} = -\hat{q}_0^{\sigma} \frac{1}{r(\boldsymbol{c}_\sigma - \boldsymbol{c}_\tau)^r} + \sum_{p=1}^{\infty} \hat{q}_p^{\sigma} (-1)^p \binom{r+p-1}{p-1} \frac{1}{(\boldsymbol{c}_\sigma - \boldsymbol{c}_\tau)^{r+p}} \end{cases}$$

T^{ifi}:

$$\begin{cases} \binom{p}{r}(\boldsymbol{c}_\sigma - \boldsymbol{c}_\tau)^{p-r} & \text{for } r \le p, \\ 0 & \text{for } r > p. \end{cases}$$

T^{tfi}:

$$(x_i - c_\tau)^{p-1}$$

1.1.3 Neighbors and interaction list

Third determine the algorithm to find the neighbors and interaction lists. Finding the neighbors is more complicated than finding the interaction list. I decompose the process of finding neighbors into finding the neighbor in eight directions, Right, Left, Up,Down, Up Right, Up Left, Down Right and Down Left. Here we only discuss the situation of finding the Right neighbor.
For the tree with each box of same size, the algorithm is described below:

```
GetNeighbor(node, direction=‘R’)
    If node.parent is root:
        return None
    If node is a right up child:
        return node.parent.left up child
    If node is a right down child:
        return node.parent.left down child
    parentNeighbor=GetNeighbor(node.parent, direction=‘R’)
    If parentNeighbor is root or parentNeighbor is leaf:
        return parentNeighbor
    If node is a right up child:
        return parentNeighbor.left up child
    If node is a right down child:
        return parentNeighbor.left down child
```

To find the interaction list of the node is all the boxes that are on the same level of the node and they don’t touch, but the parents of the node and the boxes do touch.

1.2 Running time analysis

The cost of processing the FMM algorithm can be split into three parts, the cost of building a tree, the cost of upward sweeping and the cost of downward sweeping.

The time complexity of building a tree is O(N), which is the cost of inserting all the points into each node.

The time complexity of upward sweeping is $T^{ofo}+T^{ofs} \sim PN + P^2N/b$

The time complexity of downward sweeping is $T^{tfi} + T^{ifo} + T^{ifi} + T^{near} \sim PN + P^2N/b + P^2N/b + Nb$

As we can see, by choosing b=P, we can get the running time of the algorithm is about PN.

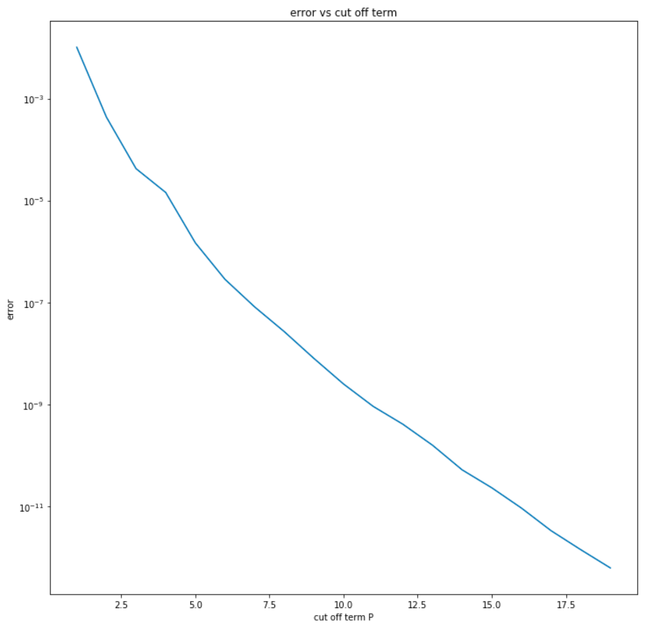
1.3 Error analysis

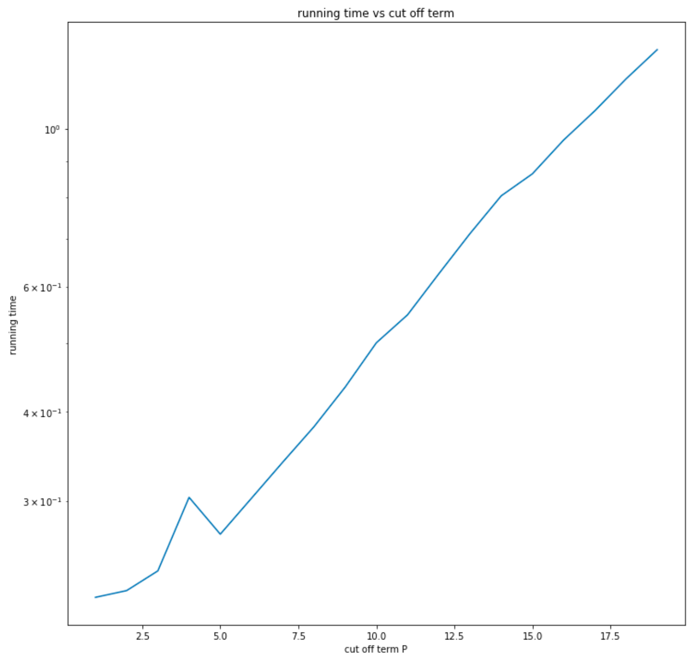
The error of the algorithm is η^P, where η is bounded by $\sqrt{2}/(4-\sqrt{3})$.

1.4 Numerical results

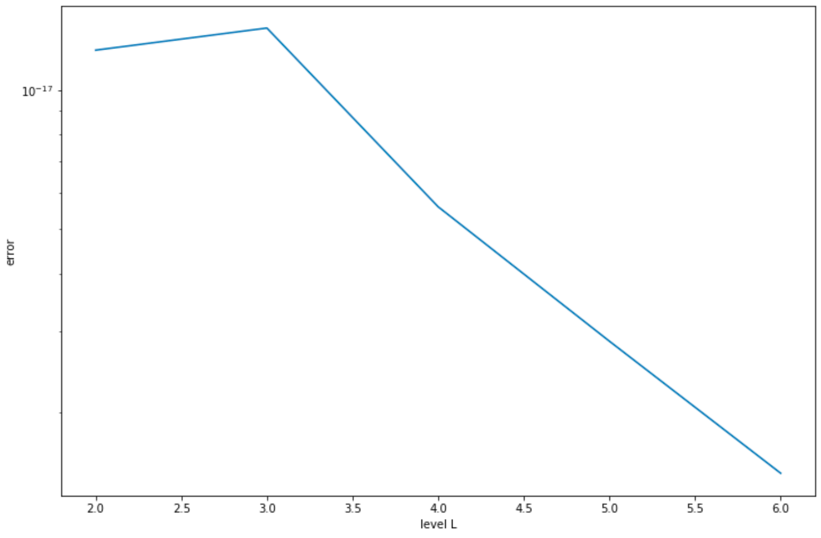
1.4.1 All the expansions of the operators are truncated after P terms, which causes in the inaccuracy of the final results. The plot shows as P gets larger, the error decreases exponentially but the running time increases exponentially. The error in all the tests are defined by:

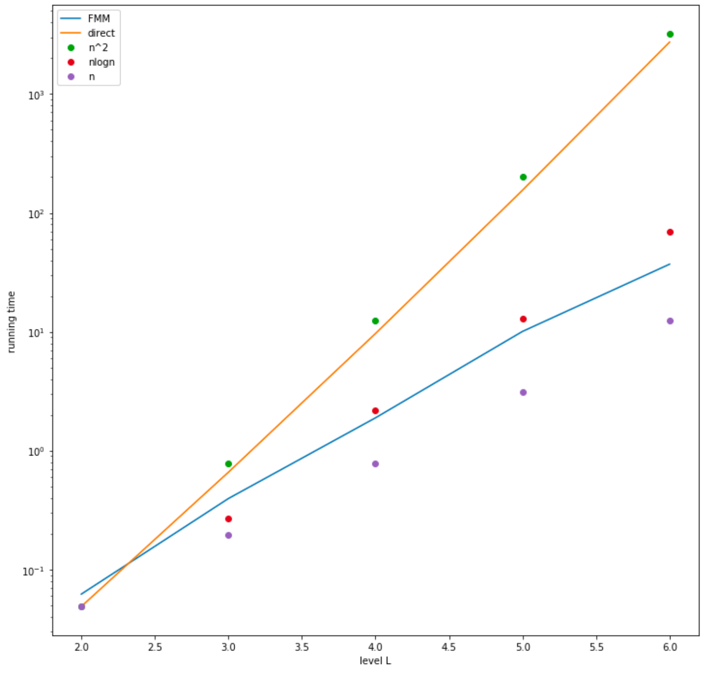
$$error = \frac{||u^{FMM} - u^{direct}||^2}{||u^{direct}||^2}$$



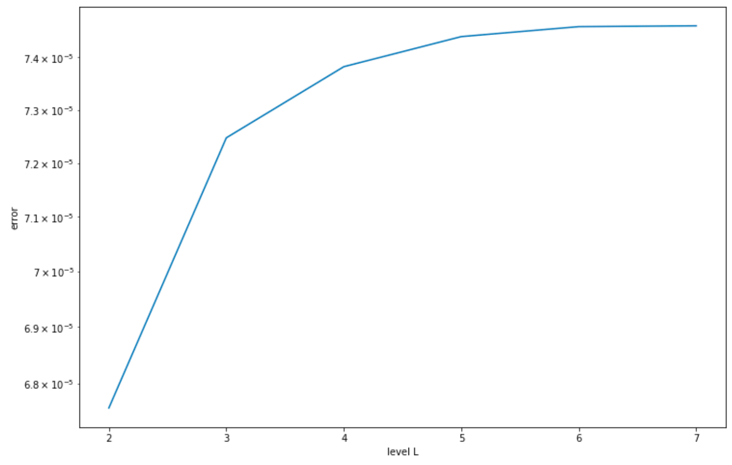


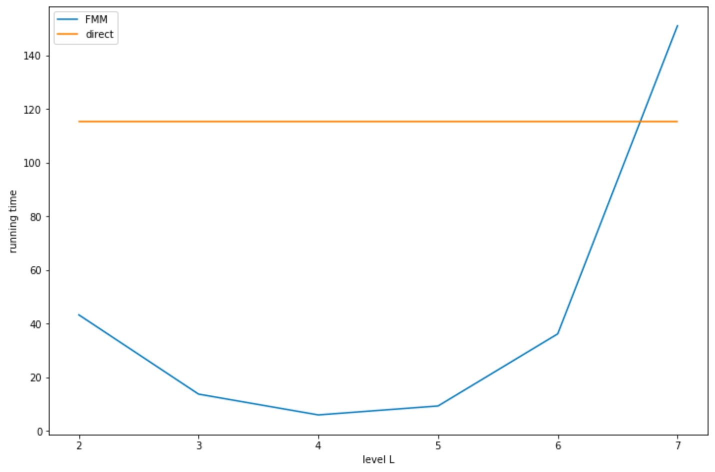
1.4.2. I fix the number of dots of each box to 10, and increase the level size L, the number dots in total is $10 * 2^{2L}$, as we can see, the running time of the FMM is between O(N) and O(NlogN), compared to the brute force algorithm which is O(N*N), which is much faster.





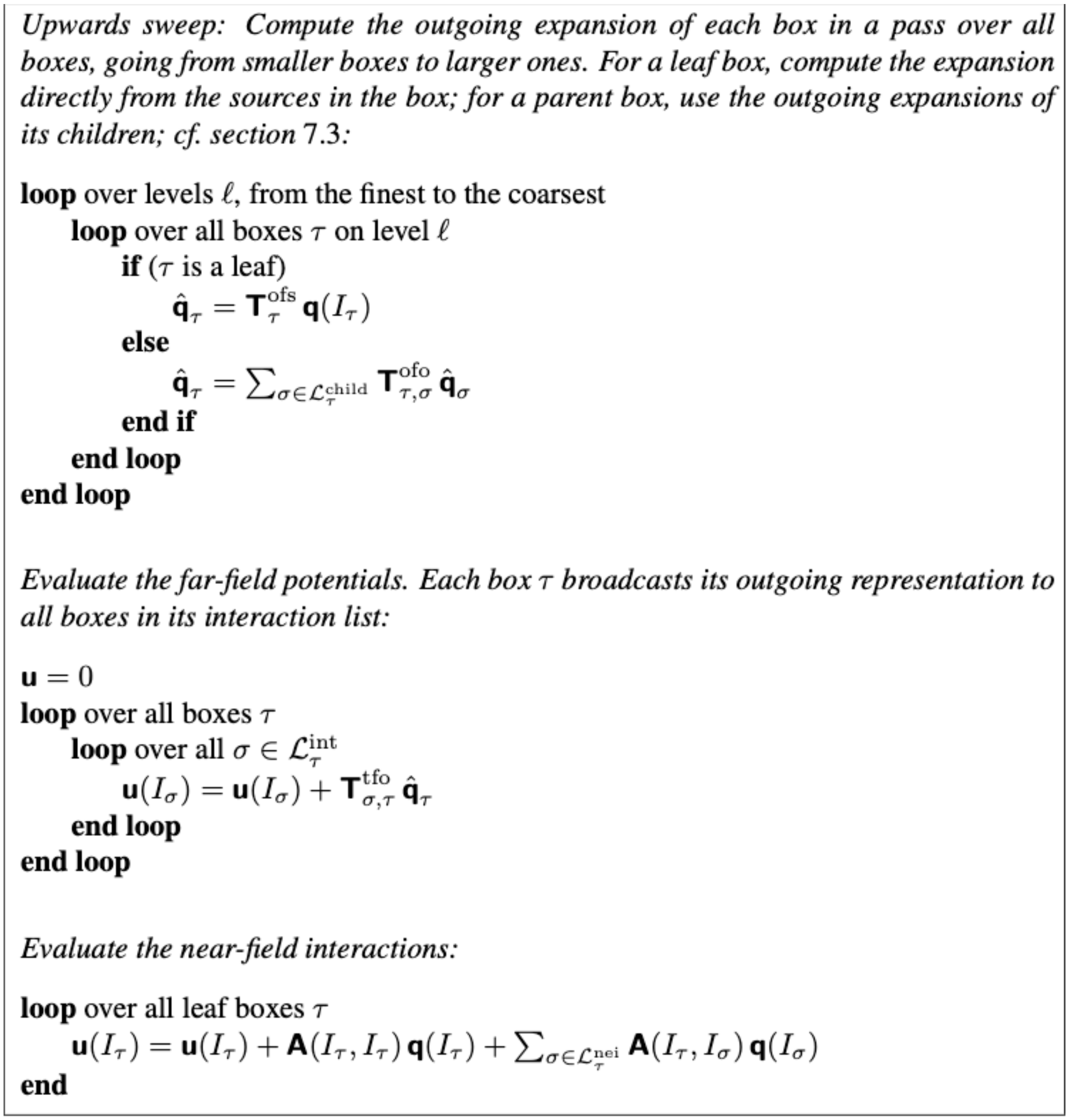
1.4.3 I fix the total number of dots N=8000 and increase the level size L. Then the average dots per box is $\frac{N}{2^{2L}}$. The plot shows that when the error increases as the level increases. When the level is small, direct method is faster, when the level is large, FMM is faster.





2. Comparison to Barnes-Hut algorithms:

The difference of FMM algorithm and Barnes-Hutt algorithm is in the downward sweeping part. Barnes-Hut algorithm has a time complexity of O(NlogN), which is worse than FMM O(N), but it has the advantage of shorter expansion to attain a give accuracy.



Upwards sweep: Compute the outgoing expansion of each box in a pass over all boxes, going from smaller boxes to larger ones. For a leaf box, compute the expansion directly from the sources in the box; for a parent box, use the outgoing expansions of its children; cf. section 7.3:

```
loop over levels $\ell$, from the finest to the coarsest
    loop over all boxes $\tau$ on level $\ell$
        if ($\tau$ is a leaf)
            $\hat{\mathbf{q}}_\tau = \mathbf{T}^{\rm ofs}_\tau \, \mathbf{q}(I_\tau)$
        else
            $\hat{\mathbf{q}}_\tau = \sum_{\sigma \in \mathcal{L}^{\rm child}_\tau} \mathbf{T}^{\rm ofo}_{\tau,\sigma} \, \hat{\mathbf{q}}_\sigma$
        end if
    end loop
end loop
```

Evaluate the far-field potentials. Each box τ broadcasts its outgoing representation to all boxes in its interaction list:

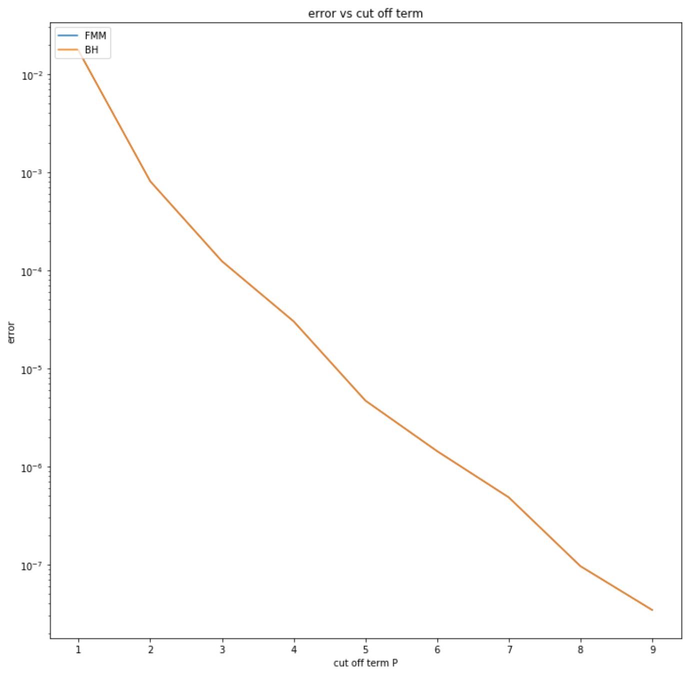
```
$\mathbf{u} = 0$
loop over all boxes $\tau$
    loop over all $\sigma \in \mathcal{L}^{\rm int}_\tau$
        $\mathbf{u}(I_\sigma) = \mathbf{u}(I_\sigma) + \mathbf{T}^{\rm tfo}_{\sigma,\tau} \, \hat{\mathbf{q}}_\tau$
    end loop
end loop
```

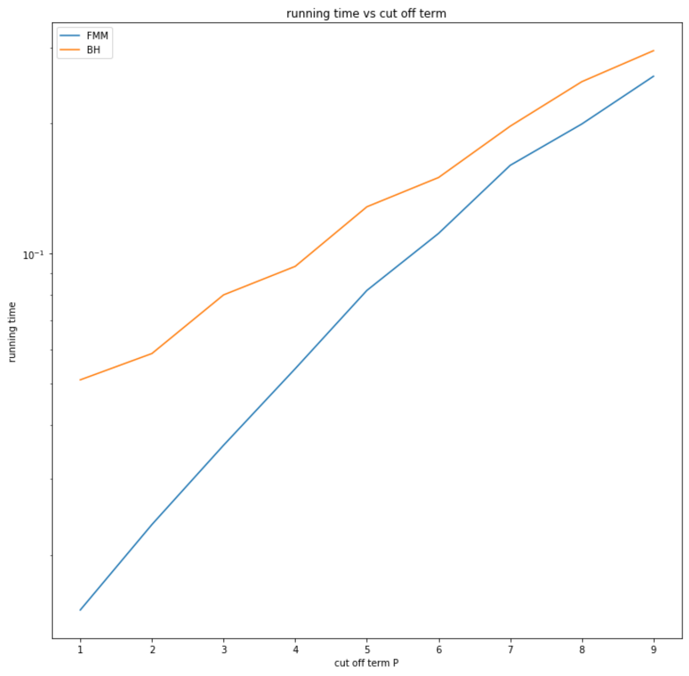
Evaluate the near-field interactions:

```
loop over all leaf boxes $\tau$
    $\mathbf{u}(I_\tau) = \mathbf{u}(I_\tau) + \mathbf{A}(I_\tau, I_\tau)\,\mathbf{q}(I_\tau) + \sum_{\sigma \in \mathcal{L}^{\rm nei}_\tau} \mathbf{A}(I_\tau, I_\sigma)\,\mathbf{q}(I_\sigma)$
end
```

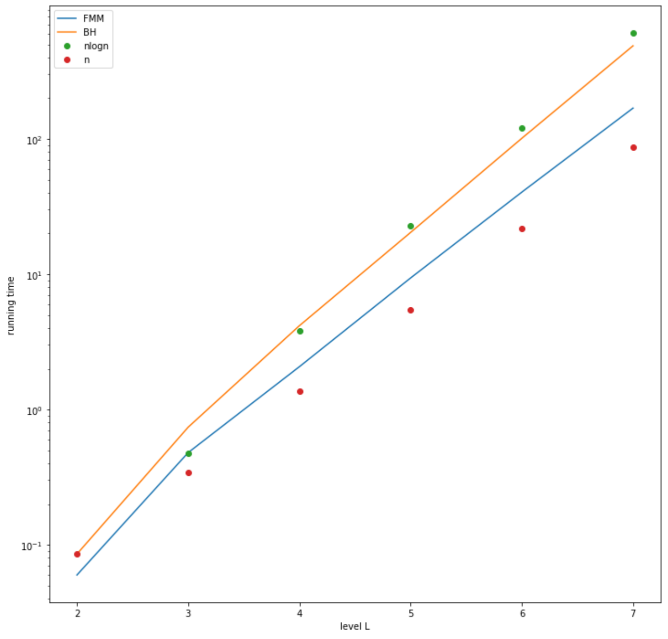
2.1 Numerical results

If I fix the number of points=1000 and level =3 , only change the cut off terms, the error on both algorithms are very small. While the running time, FMM is faster than Barnes-Hut.





If I fix term P=10 and number of dots in a box=10, the time complexity of FMM algorithm is between O(N) and O(NlogN) while the time complexity of Bernes-Hutt is approximating O(NlogN).

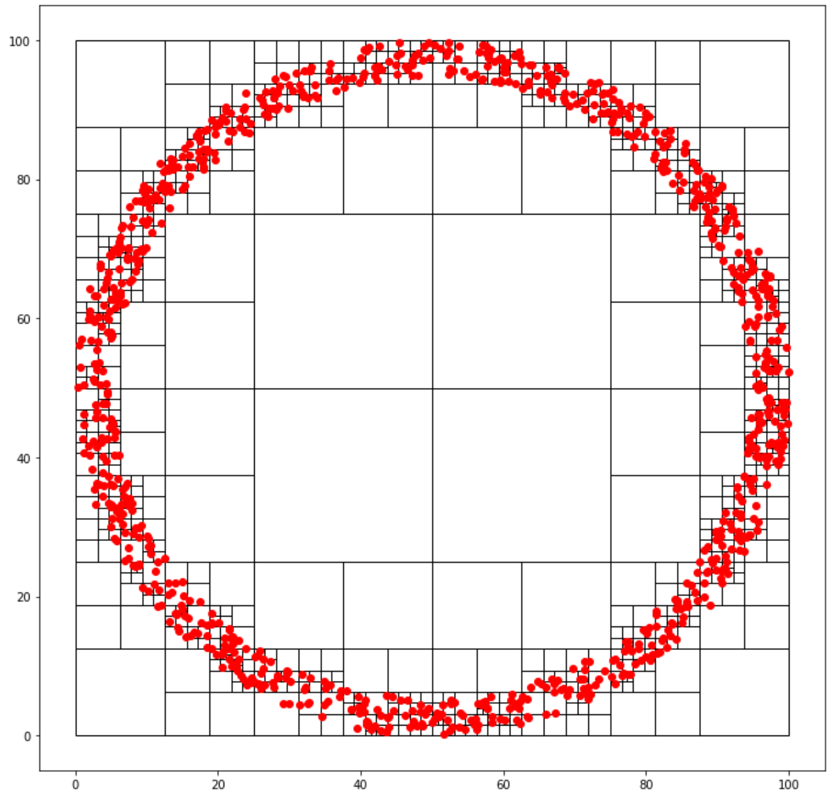


3. Extensions to nonuniform trees

Consider the situation I have a dataset where all the data are not uniformly distributed in the square.

3.1 tree construction

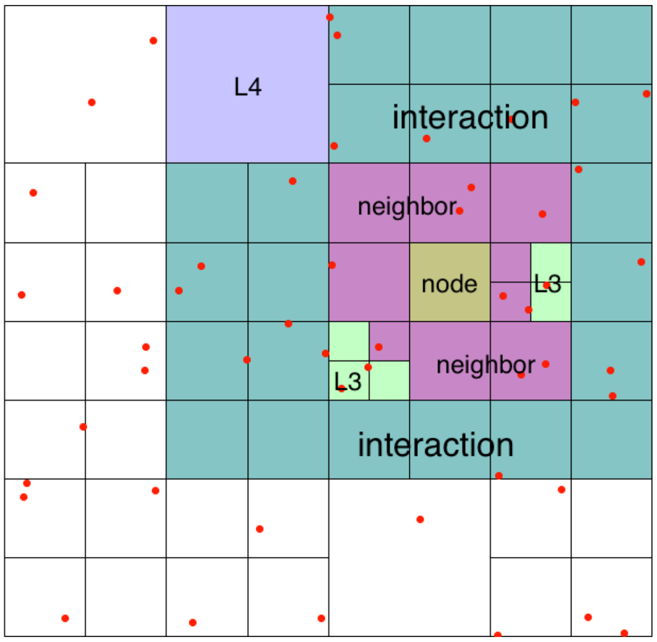
It is more efficient to build an adaptive quad tree and keep the number of dots in each box under the threshold. The picture below is an adaptive tree with the threshold is 3, which means if the number of dots is larger than 3, I will split the box into four equal smaller boxes.



To use FMM in this nonuniform tree, I need expand the correlated lists of each node. The definition is below:

$\mathcal{L}_\tau^{\mathrm{child}}$	The children of τ.
$\mathcal{L}_\tau^{\mathrm{nei}}$	For a parent box τ, $\mathcal{L}_\tau^{\mathrm{nei}}$ is empty. For a leaf box τ, $\mathcal{L}_\tau^{\mathrm{nei}}$ is a list of the leaf boxes that directly border τ.
$\mathcal{L}_\tau^{\mathrm{int}}$	A box $\sigma \in \mathcal{L}_\tau^{\mathrm{int}}$ if and only if σ and τ are on the same level, σ and τ are well separated, and the parents of σ and τ are not well separated.
$\mathcal{L}_\tau^{(3)}$	For a parent box τ, $\mathcal{L}_\tau^{(3)}$ is empty. For a leaf box τ, a box $\sigma \in \mathcal{L}_\tau^{(3)}$ if and only if σ lives on a finer level than τ, τ is well separated from σ, and τ is not well separated from the parent of σ.
$\mathcal{L}_\tau^{(4)}$	The dual of $\mathcal{L}_\tau^{(3)}$. In other words, $\sigma \in \mathcal{L}_\tau^{(4)}$ if and only if $\tau \in \mathcal{L}_\sigma^{(3)}$.

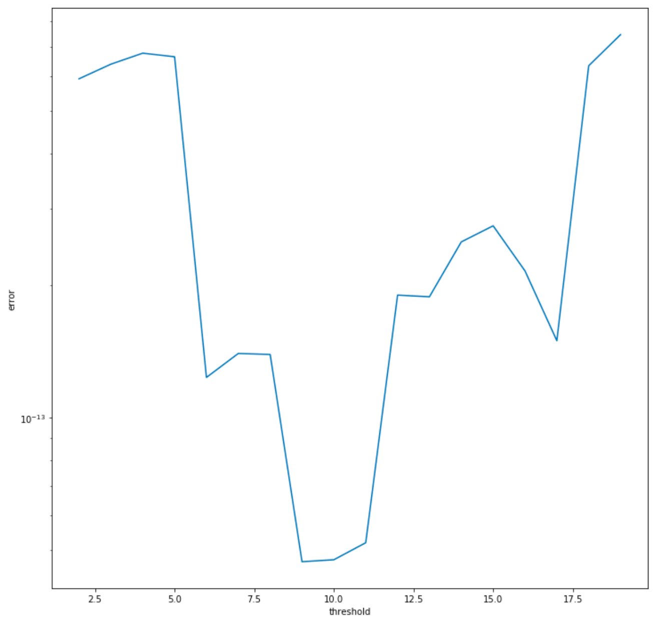
The relationship of different lists is visualized in the following plot.

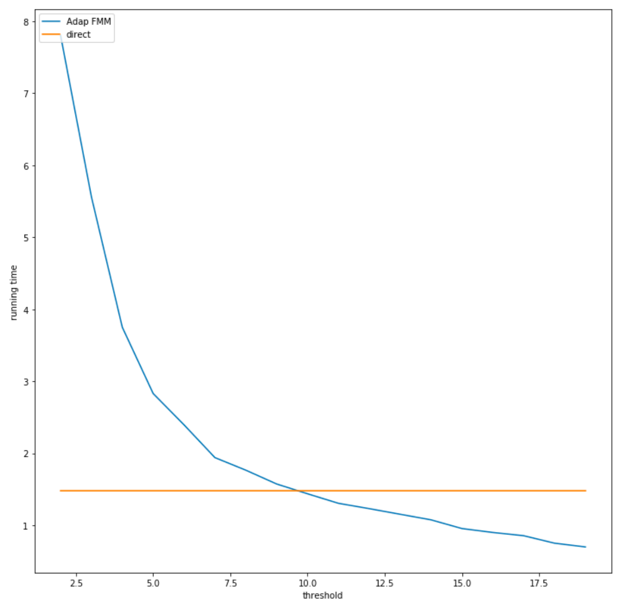


I also make some changes to the algorithm. For the upward pass, the algorithm stays the same as the classic FMM. For the downward pass, for each box the contribution from its parent's far-field is computed by translating the local expansion of parent to the center of the node. Then the multipole expansion of all boxes in the interaction list are accumulated into local expansions around the center of the node. If the node is a leaf, then I also add the interactions of its L3 list and L4 list. The final summation step, the potential of each dot in the leaf box is calculated by add the incoming expansion with the contributions from its neighbors.

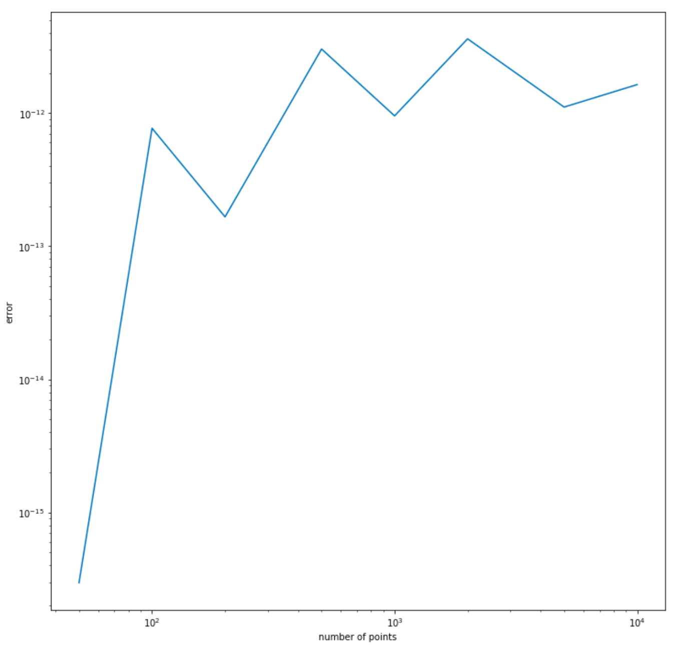
3.2 Numerical results:

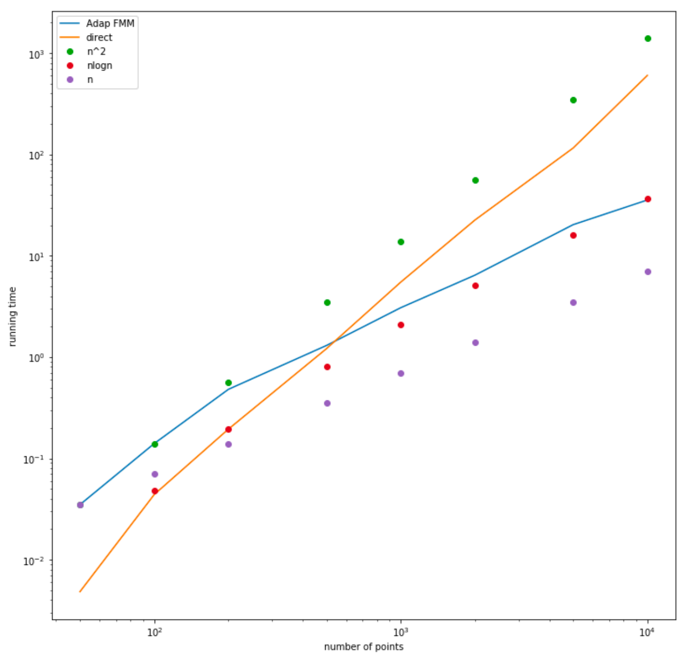
3.2.1 I generate a random dataset with the shape of a circle and keep the number of total dots N=1000. And I find that as the threshold of the maximum number of dots in each box decreases, error doesn't change a lot, but the running time decreases exponentially.



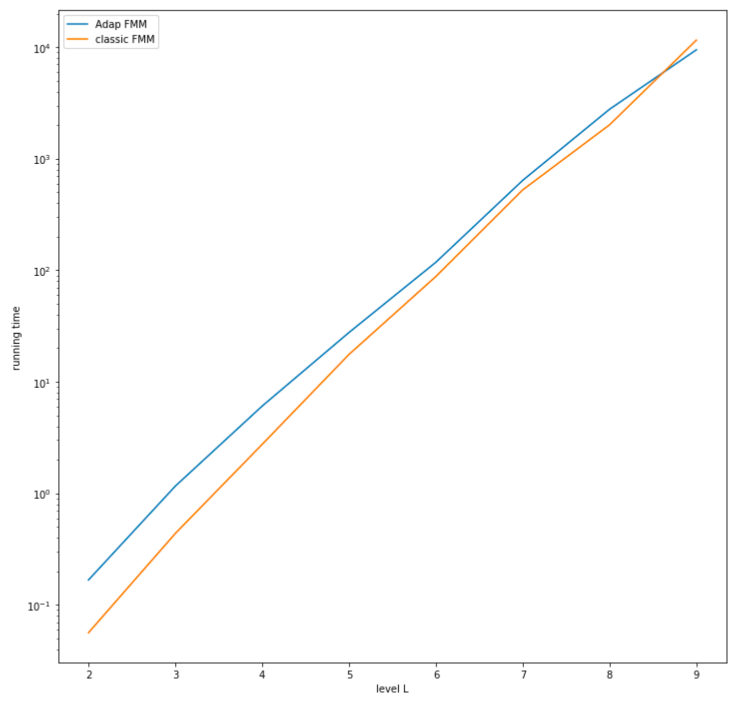


3.2.2 If I fix the threshold as 10, and increases the number of dots in total, the error doesn't change much, and the adaptive FMM is slower in the small dataset but faster in large datasets.





3.2.3 Last I compare the performance of classic FMM and adaptive FMM on non uniform datasets. The number of dots in total $N = 2^{2L} * 10$, I keep the threshold of the number of dots in each box in adaptive FMM as 10, and change the level of the tree L. The comparison of the running time of the two algorithms are below, it shows adaptive FMM is a bit slower than classic FMM in small datasets but in larger dataset, adaptive FMM is faster.



3.3. Further ideas

As seen from the comparison of classic FMM and adaptive FMM on nonuniform dataset, the adaptive FMM is not more efficient than the classic FMM. Some reasons I guess is 1) The computation cost in calculating neighbors, interaction list, L3 list and L4 list is much higher in adaptive FMM. 2) I didn't use tuning technique in adaptive FMM, by tuning out the blank boxes, it should be faster.

4. Reference

[1]Greengard, L., & V. Rokhlin. "A Fast Algorithm for Particle Simulations." Journal of Computational Physics
[2]Luis Barroso-Luque Github: https://github.com/lbluque/fmm
[3]Martinsson, Fast Direct Solvers for Elliptic PDEs, 2019 Chapter 6,7,8