Methods for Solving Large Dense Systems

All in One Large and Dense Paper

Cody Diyn

Vin Isaia

Math 613, Applied Linear Algebra

Fall 2025

1 Introduction

In this paper, we will be introducing the concepts for solving large, dense matrix systems. We will be presenting the basics of the multipole methods as well as building up to understanding hierarchical matrices. Along with each of these, we will provide published papers deploying these methods to demonstrate their use. We will see a variety of fields using these methods, which shows how important they can be. These applications will be seen in topics such as automotive design, partial differential equations, political science, and quantum computing.

When dealing with elementary contexts of matrices, sizes of 20×20, 50×50, or even 100×100 seem to be large. However, sometimes we may need to solve systems with a square matrix that has 10^7 rows, e.g. see [1]. These matrices can become so large that storing the information in the memory of one's computer becomes a challenge on its own. This then motivates us to find methods for solving large matrix systems.

We can then define "dense" in a general sense as being a matrix with mostly non-zero entries. We can also define it a little more rigorously by saying it's any matrix that isn't sparse. This helps because as is mentioned on page 183 in the text and in [2], sparse matrices have only $O(n)$ non-zero entries. Then, with sparse matrices, one can omit the zeroes to save storage space. However, for dense matrices, this can't be done, so other measures will need to be taken.

2 Fast Multipole Method

We will begin by discussing section 1.1.1. In this section, the topic of interpolation is discussed. We first suppose that we have given data sites $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subseteq \mathbb{R}^d$ and observations $f_1, ..., f_n \in \mathbb{R}$. We also suppose there is some function f such that $f(\mathbf{x}_i) = f_i, 1 \leq i \leq n$. We then want a way to reconstruct this unknown f. To do so, we first choose basis functions $\psi_1, ..., \psi_n \in C(\mathbb{R})^d$. Then, we can approximate f as

$$s(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j \psi_j(\mathbf{x}), \ \mathbf{x} \in \mathbb{R}^d \tag{1}$$

where $a_j = f(\mathbf{x}_j)$. Then, the coefficients are given by

$$f_i = s(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j \psi_j(\mathbf{x}_i), \ 1 \leq i \leq n, \tag{2}$$

which can be represented in matrix form as

$$\begin{bmatrix} \psi_1(\mathbf{x}_1) & \psi_2(\mathbf{x}_1) & ... & \psi_n(\mathbf{x}_1) \\ \psi_1(\mathbf{x}_2) & \psi_2(\mathbf{x}_2) & ... & \psi_n(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(\mathbf{x}_n) & \psi_2(\mathbf{x}_n) & ... & \psi_n(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}. \tag{3}$$

We also have by section 1.1.3 that the matrix entries are of the form

$$a_{ij} = \int_{\Omega} \int_{\Omega} K(\mathbf{x}, \mathbf{y}) \chi_j(\mathbf{y}) \chi_i(\mathbf{x}) d\mathbf{y} d\mathbf{x}, \ 1 \leq i, j \leq n, \tag{4}$$

where each function $\chi_i, 1 \leq i \leq n$ form the basis of a given finite element space.As a result of these double integrals, it becomes very expensive to compute the entries of A. As a result, we want to compute the approximate matrix $\tilde{A}$.

We will then choose our basis functions such that $\psi_j = K(\mathbf{x}, \mathbf{x}_j)$ so the approximate is given by

$$s(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j K(\mathbf{x}, \mathbf{x}_j), \ \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, \tag{5}$$

where $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x} - \mathbf{y}) = \phi(||\mathbf{x} - \mathbf{y}||_2)$ is a positive-definite kernel. We then get a symmetric and positive definite interpolation matrix $A = (K(\mathbf{x}_i, \mathbf{x}_j))$. That is, each element of matrix A is obtained by evaluating the kernel at two data sites in Ω. As made clear in equation (3), we have a system of the form $A\alpha = \mathbf{f}$. By remark 3.6 in [3], we know that solving this system directly requires $O(n^3)$ time in $O(n^2)$ space. This is something that quickly becomes problematic when dealing with large systems.

As of now, we will be computing matrix-vector multiplications by evaluating sums n times, which gives a computational cost of $O(n)$. We want to significantly reduce this time. To begin doing so, we will first assume the above kernel can be written as

$$K(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{r} \phi_k(\mathbf{x})\psi_k(\mathbf{y}) \tag{6}$$

where ϕ_k, ψ_k are fixed functions with $1 \leq k \leq r$ and $r \in \mathbb{N}$. In this case, we can refer to K as a *degenerate kernel.* In this case, we can then plug in equation(6) into equation (5). This gives us

$$\begin{aligned}\sum_{j=1}^{n} \alpha_j K(\mathbf{x},\mathbf{x}_j) &= \sum_{j=1}^{n} \alpha_j \left[\sum_{k=1}^{r} \phi_k(\mathbf{x})\psi_k(\mathbf{x}_j)\right] \\ &= \sum_{k=1}^{r} \left[\sum_{j=1}^{n} \alpha_j \psi_k(\mathbf{x}_j)\right] \phi_k(\mathbf{x}).\end{aligned} \tag{7}$$

Then, we can define new coefficients as

$$\beta_k := \sum_{j=1}^{n} \alpha_j \psi_k(\mathbf{x}_j). \tag{8}$$

The sum in equation (5) then becomes

$$s(\mathbf{x}) = \sum_{k=1}^{r} \beta_k \phi_k(\mathbf{x}). \tag{9}$$

We now have to compute the value of β, which will cost $O(n)$ time, followed by evaluating the sum $s(\mathbf{x})$, which is now $O(r)$ time. This means the total cost will be $O(nr)$, which is significantly less than $O(n^2)$ given a small enough r.

This gives us a nice way to solve the large, dense system given that A has a degenerate kernel, however, this is not something to rely on. Kernels are not usually degenerate, but rather they come in the form

$$K(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{r} \phi_k(\mathbf{x})\psi_k(\mathbf{y}) + R_r(\mathbf{x},\mathbf{y}). \tag{10}$$

Because there is usually some remainder R_r, we will define an approximate kernel similar to above as

$$K_r(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{r} \phi_k(\mathbf{x})\psi_k(\mathbf{y}), \tag{11}$$

which leads to an approximate sum of

$$\tilde{s}(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j K_r(\mathbf{x},\mathbf{x}_j). \tag{12}$$

We will now refer to the $\mathbf{x}_j$ in $K(\mathbf{x},\mathbf{x}_j)$ as a *source point* and the $\mathbf{x}$ as an *evaluation point.* We will also assume that $K(\mathbf{x},\mathbf{y})$ can be approximated as $K_r(\mathbf{x},\mathbf{x}_j)$ when $\mathbf{x},\mathbf{x}_j$ are well-separated and r is sufficiently large.

We then will assume that all sources are found in the same region, which will be centered at some point y_0. This region will then be referred to as a *panel.* Then, we have that $\mathbf{x}$ is sufficiently far from this panel. This is a result from the assumption that $\mathbf{x},\mathbf{x}_j$ are well-separated. This is illustrated below.

Within the source panel, we have an arbitrary number of dots. These represent the many different source points in our system. We also will suppose our goal is to evaluate function (5) at the point $\mathbf{x}$. Before, to calculate the value of $s(\mathbf{x})$, we would have a computational cost of $O(n^2)$. Now, with our approximate sum $\tilde{s}(\mathbf{x})$, we have a cost of $O(r)$. We can then see that this leads to the cost being linear instead of quadratic, and it's therefore cheaper to perform the approximation. We now will look at examples of this method in published papers.

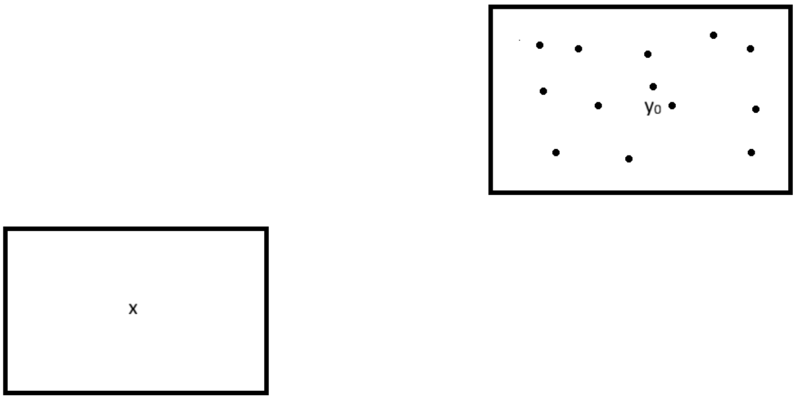


Figure 1: Evaluation (left) and source (right) panels

3 Examples of FMM

We will now look at practical examples of these multipoles methods. First, we will look at a paper studying how outside acoustics need to be accounted for when designing new car models. It walks through the traditional Boundary Element Method, then explains how this is not as efficient as a multipole method. Next, we will see an example where researchers develop an algorithm for spherical harmonics transforms using fast multipole methods. This not only shows the usefulness in learning about FMM, but also shows how far spread the applications can be.

3.1 Acoustic Load Predictions

Our first example will be from article [4]. As this article mentions, when developing car models, it is important to take into account acoustic pressure on the body of the vehicle. As would be expected, the ideal time to factor in pressure from the outside would be before any money has been spent on a prototype. They mention the difficulty of doing so as there are many factors that go into it that need to be accounted for. The usual way around this issue is to look at older models given the shape of the new car, but this method has its drawbacks as well. Specifically, if the new car has even a slightly different geometry than any of the previous ones, then it may still not be the most accurate way to solve the problem.

This motivated the authors to look at how the a multipole method would work by comparing it with a more traditional Boundary Element Method (BEM). The authors note that an issue with conventional methods is the global connectivity of the system. Because we are dealing with vibrations on the body of a car, there's likely some interference that makes it difficult to predict the acoustic load. This can be expressed by

$$\int_{\partial\Omega}\int_{\partial\Omega} f(\mathbf{x})G(\mathbf{x},\mathbf{y})g(\mathbf{y})dS(\mathbf{y})dS(\mathbf{x}),\ \mathbf{x},\mathbf{y}\in\partial\Omega, \tag{13}$$

where f, g are two functions defined over the boundary, and $G : \Omega \times \Omega \to \mathbb{R}$ is a kernel. We can then see that this equation matches up with equation (4) where

$$\begin{aligned} G(\mathbf{x},\mathbf{y}) &= K(\mathbf{x},\mathbf{y}) \\ f(\mathbf{x}) &= \chi_i(\mathbf{x}) \\ g(\mathbf{y}) &= \chi_j(\mathbf{y}). \end{aligned} \tag{14}$$

We are also taking these integrals over the boundary of Ω rather than just all of Ω itself. As a result, the author are then able to use a multipole method to work out the pressure on the body of the car. In the traditional BEM method, when a group of source points $\mathbf{x}_i$ are near a point $M1$ and are acting on some points $\mathbf{y}_i$ near another point $M2$, there is a lot of interference, which greatly increases the number of interactions and computations needed.

However, whenever they used the multipole method, they found some improvement. This led to information being centralized. Instead of there being direct interactions between source and evaluation points, the interactions were more indirect. That is, each source point $\mathbf{x}_i$ interacts with its nearby point $M1$. $M1$ in turn interacts with $M2$, which then interacts with the points nearby $\mathbf{y}_i$. This difference is illustrated below.

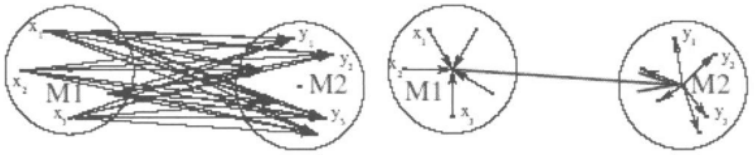


Figure 2: BEM (left) vs. FMM (right)

This means that the multipole method leads to less interactions between well-separated points on the body of the car, which means they lead to less calculations necessary. This makes it a significantly more efficient tool as opposed to the conventional methods.

3.2 Spherical Harmonics Transform

As a second example of the application of these methods, we will look at article [5]. This paper uses the above method to make a fast algorithm for spherical harmonics transform. The idea of harmonics comes from Laplace's equation:

$$\Delta f = \nabla \cdot \nabla f = \nabla^2 f = 0. \tag{15}$$

A function is said to be *harmonic* if it is a solution to equation (15), see [6]. Because $\Delta f = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ by definition, we are interested in solving the PDE $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} = 0$. Furthermore, as article [5] mentions, f is said to be a spherical harmonic if it is an eigenfunction of Δ on the surface of a sphere. A spherical harmonic transform takes some function g on the sphere and transforms it into

$$g(\lambda_j, \mu_k) = \sum_{m=0}^{N} \sum_{n=m}^{N} g_n^m Y_n^m(\lambda_j, \mu_k), \tag{16}$$

where Y_n^m denotes the spherical harmonics and g_n^m are the coefficients of each spherical harmonic. We also have that (λ_j, u_k) is the evaluation point given $1 \leq j \leq J, 1 \leq k \leq K, -1 \leq \mu_k \leq 1, 0 \leq \lambda_j < 2\pi$. The values of J, K can be approximated as $J \approx 3N, K \approx \frac{3N}{2}$. We then want to get the following spherical harmonic expansion

$$\begin{aligned} g^m(\mu_k) &= \frac{1}{J} \sum_{j=1}^{J} g(\lambda_j \mu_k) e^{im\lambda_j} \\ g_n^m &= \sum_{k=1}^{K} w_k g^m(\mu_k) P_n^m(\mu_k), \end{aligned} \tag{17}$$

where P_n^m is the associated Legendre function. In computing the transform desired, a direct approach was typical of the time. However, this would require $O(n^3)$ time. This led the authors to attempt to find a faster way to compute the transform. The authors modeled their fast algorithm for a spherical harmonics transform off of the fast multipole methods. They found through experimental results that the algorithm was not only stable, but also faster than direct computation whenever $N \geq 511$.

4 Hierarchical Matrices

Next, we will discuss hierarchical matrices. To do so, we need to first define a couple of terms. We will begin defining various partitionings, admissible functions, and block cluster trees.

4.1 Partitionings

To begin defining a hierarchical partitioning, we will first define a general partition.

Definition 1. *Say we have some index set $\mathcal{I} = \{1, ..., n\}$. Then, a partitioning $P(\mathcal{I})$ of the index set is a subdivision of $\mathcal{J}$ into $p \leq n$ disjoint index sets, labeled $J_1, ..., J_p$, with cardinality $|J_j| = n_j$ where $1 \leq j \leq p$.*

In other words, if $P(\mathcal{I}) = \{J_1, ..., J_p\}$ is a partitioning of the index set $\mathcal{J}$, then

$$\begin{aligned} & J_i \cap J_j = \emptyset, \ i \neq j \ \text{ and} \\ & \bigcup_{j=1}^{p} J_j = \mathcal{I}. \end{aligned} \tag{18}$$

We also have that $n = n_1 + ... + n_p$. Then, defining a hierarchical partitioning means we just need to build on the definition of a partitioning.

Definition 2. *Let $\mathcal{I}$ be the index set defined above. Then, a hierarchical partitioning $P_L(\mathcal{I})$ of $\mathcal{I}$ of depth L is a family $P_\ell(\mathcal{I}) = \{J_1^{(\ell)}, ..., J_{p\ell}^{(\ell)}, 0 \leq \ell \leq L$. This partitioning then has the following properties:*

1. The partitioning on the level $\ell = 0$ is given as $P_0(\mathcal{I}) = \mathcal{I}$.

2. For $1 \leq \ell \leq L$, we have
• Every subdivision $J^{(\ell-1)} \in P_{\ell-1}(\mathcal{I})$ is the disjoint union of at least two sets $J^{(\ell)} \in P_\ell(\mathcal{I})$ or it is disjoint with each set $J^{(\ell)}$.
• Each subdivision $J^{(\ell)}$ is contained in exactly one set $J^{(\ell-1)}$.
3. There is at least one non-empty set $J^{(L)}$ of level L.

We now have a definition for hierarchical partitioning, but we still need the following definition before moving on.

Definition 3. *For two index sets $\mathcal{I} \times \mathcal{J}$, the tensor product partitioning is given by the partitions $P_1(\mathcal{I}), P_2(\mathcal{J})$. We then give a definition for this partitioning as*

$$\begin{aligned} P(\mathcal{I} \times \mathcal{J}) &:= P_1(\mathcal{I}) \times P_2(\mathcal{J}) \\ &= \{I \times J | I \in P_1(\mathcal{I}), J \in P_2(\mathcal{J})\}. \end{aligned} \tag{19}$$

4.2 Functions

Definition 4. *We will say $P_K(\mathcal{I}), P_L(\mathcal{J})$ are both hierarchical partitionings of the index sets $\mathcal{I}, \mathcal{J}$. Then, an admissible function for the hierarchical partitioning is a function*

$$adm : P(\mathcal{I} \times \mathcal{J}) \to \{true, false\}. \tag{20}$$

We then say that $I \times J$ is admissible if $adm(I \times J) = true$. We also say that $I \in P_K(\mathcal{I})$ is admissible with respect to $J \in P_L(\mathcal{J})$ if $adm(I \times J) = true$.

4.3 Trees

Definition 5. *We will suppose we have $V = \{v_1, v_2, ...\}$ as a set of nodes and $B = \{b_1, b_2, ...\}$ as a set of leaves. Then, every element $b_i \in B$ is a tree and b_i is the tree's root. For b_i to be a root implies there is no node above it.*

As an example, we can look at

Then, we have that $V = \{2, 7, 5, 6, 9\}$ as the nodes. We also have $B = \{2, 10, 5, 11, 4\}$, which are leaves. We then say that leaves are nodes with no *children node*, meaning no nodes come after it. We can also see then that the 2 at the very top is the root of the whole tree. This is because there is no *parent node*, or node above it.

Definition 6. *Assume we have multiple trees $T_1, ..., T_m$. We will say these trees have pairwise disjoint sets of nodes and leaves. We will also say that $v \in V$ is a new node for each tree. Then, we can make this new node v be the root of a new tree $(v, T_1, ..., T_m)$. That is, v will connect to the roots of the other m trees we have. This means the root of the node is $\rho(v) = m$, and we call T_i the ith sub-tree.*

From here, we build the definition of a *cluster tree.*

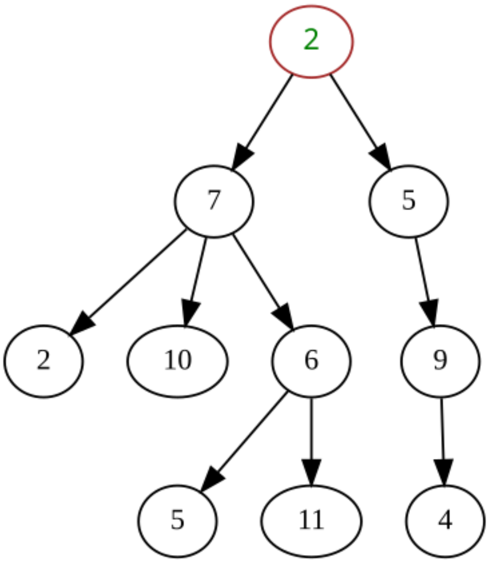
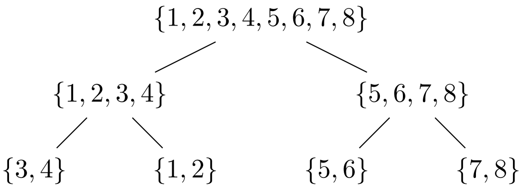


Figure 3: Example of a Tree

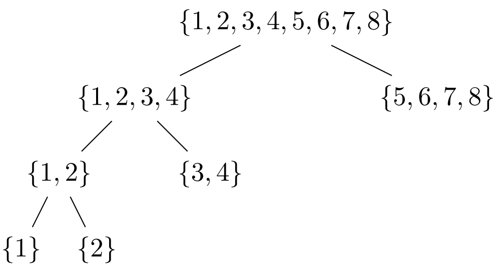
Definition 7. *Let $\mathcal{T} = \{1, ..., n\}$. Then, we say T is a cluster tree of $\mathcal{T}$ if both the nodes and leaves of T contain subsets of $\mathcal{T}$ as information. We then want the following to be true:*

- *The root of the tree contains $\mathcal{T}$ as information*
- *Each node has at least two children*
- *Let $C(t)$ denote children for node t. Then, the information stored at t is the disjoint union of the infomation stored at t's children $s \in C(t)$.*

We now will provide an example of a cluster tree below.



Then, for $\mathcal{T} = \{1, 2, 3, 4, 5, 6, 7, 8\}$, we have that this is a cluster tree with a balanced partitioning. We can then check each part of definition (7) and see that the above is indeed an example of a cluster tree. We refer to the above cluster tree as having a balanced partitioning due to the fact that each of its leaves are on the same level ℓ. We may, however, have an unbalanced partitioning, but still have a cluster tree. The figure below is an example of this.



We can once again verify that this is a cluster tree by definition (7). We now have defined what a cluster tree is, as well as a hierarchical partitioning and an admissible function. With these definitions, we can then define a *block cluster tree.*

Definition 8. *Let $T_{\mathcal{I}}, T_{\mathcal{T}}$ be two cluster trees for the index sets $\mathcal{I}, \mathcal{J}$. We will further say that these index sets correspond to the hierarchical partitionings $P_K(\mathcal{I}), P_L(\mathcal{T})$. Then, we will let $adm : P(\mathcal{I} \times \mathcal{J}) \to \{true, false\}$ be an admissible function. Then, we define the block cluster tree $T_{\mathcal{I}\times\mathcal{J}}$ recursively as follows.*

1. *The root of $T_{\mathcal{I}\times\mathcal{J}}$ is $\mathcal{I} \times \mathcal{J}$.*
2. *The recursion starts with $I \times J$ for $I = \mathcal{I}, J = \mathcal{J}$.*
   a. *If either I or J is a leaf in its cluster tree, then $I \times J$ is also a leaf.*
   b. *If $I \times J$ is admissible, then $I \times J$ is a leaf.*
   c. *Otherwise, $I \times J$ is a node and the children $C(I \times J)$ of $I \times J$ are defined as*

$$\{I' \times J' | I' \in C(I), J' \in C(J)\}. \tag{21}$$

   d. *If $I \times J$ is a node, the procedure is repeated for all its children.*

4.4 Hierarchical Matrices

Now we have all of the pieces needed to define a *hierarchical matrix.*

Definition 9. *Let $T_{\mathcal{I}\times\mathcal{J}}$ be some block cluster tree for the index sets $\mathcal{I}, \mathcal{J}$. Let the cardinality of these sets be $|\mathcal{I}| = m, |\mathcal{J}| = n$. Further, let $T_{\mathcal{I}\times\mathcal{J}}$ be based on the cluster trees $T_{\mathcal{I}}, T_{\mathcal{J}}$ with an admissible function adm. Then, a hierarchical matrix, or $\mathcal{H} - matrix$, of rank r with respect to $T_{\mathcal{I}\times\mathcal{J}}$ is a matrix $A \in \mathbb{R}^{m\times n}$ which satisfies*

$$rank(A_{I,J}) \leq r \tag{22}$$

for all admissible leaves $I \times J$. We say that $A_{I,J}$ denotes the sub-matrix of A such that its rows are given by I and its columns are given by J. We denote the set of all $\mathcal{H}$-matrices by $\mathcal{H}(T_{\mathcal{I}\times\mathcal{J}}, r)$.

Then, we can see that block cluster trees are foundational in defining an $\mathcal{H}-$matrix. This relation means we need to store just the information of the leaves for the $\mathcal{H}-$matrix. There are two possibilities for these leaves; either they are admissible or they are not admissible. In the former case, we can store the leaf's information in standard matrix form. For the latter, we can use the following corollary from the text.

Corollary 1. *Every matrix $A \in \mathbb{R}^{m\times n}$ of rank r can be written in the form $A = BC^T$ with $B \in \mathbb{R}^{m\times r}, C \in \mathbb{R}^{n\times r}$.*

Then, if the leaf happens to be admissible, the above corollary tells us the matrix block can be represented for the leaf can be represented in the form of BC^T with $B \in \mathbb{R}^{\mathcal{I}\times r}$ and $C \in \mathbb{R}^{\mathcal{J}\times r}$. What we get from this talk of the admissibility of the leaves is how much the admissible function *adm* plays a role when it comes to estimating the space needed to store an $\mathcal{H}-$matrix as well as its supporting data structure. As a result, we cannot provide a general estimate for estimating the space needed to store $\mathcal{H}-$matrices, but rather, we need to focus on specific cases. To demonstrate this, we will look at a specific example.

Lemma 1. *Let $\mathcal{I} = \{1, ..., 2^L\}$ be recursively subdivided by bisection into equally sized subsets, such that $P_\ell(\mathcal{I})$, $0 \leq \ell \leq L$, consists of 2^ℓ subsets of size $2^{\ell-1}$. Further, let $T_{\mathcal{I}\times\mathcal{I}}$ be the associated block cluster tree with the admissible function*

$$adm(I \times J) = \begin{cases} true & if\ dist(I, J) \geq 2^{L-\ell} = |I| = |J|, \\ false & else, \end{cases} \tag{23}$$

where $I, J \in P_\ell(I)$. Also, let $A \in \mathcal{H}(T_{I\times I}, r)$ be a hierarchical matrix. Then, the number of blocks required to represent A will be bounded by $O(n)$ and the memory to store A is of size $O(rn \cdot log(n))$.

By this example, we can see that both the number of blocks and the memory required are less than $O(n^2)$. This shows that this method is more efficient than individually storing each element.

In solving matrix systems, however, the bigger concern is performing matrix-vector products. To do so, we again become interested in storing the leaves, which means the structure of the block cluster tree itself plays a role in the computational cost for each product. This tells us that once again, we can't develop a general formula for the computational cost of some matrix-vector product. We need to rely on specific examples. We again demonstrate the effectiveness of $\mathcal{H}-$matrices by reusing the previous example and calculating its computational cost.

Lemma 2. *Assume we have $I, adm, T_{I\times I}$, and A as defined in lemma (1). Then, the computational cost for computing a matrix vector product $\boldsymbol{y} := A\boldsymbol{x}, \boldsymbol{x} \in \mathbb{R}^n, n = 2^L$, is of size $O(rn \cdot log(n))$.*

Similar to before, we find that using these hierarchical matrices to store the information of A, the computational cost becomes less than $O(n^2)$. This begins to show its usefulness in practical cases of large dense systems. With all we've discussed of this method so far, a couple of questions may come up.

First, one might wonder how the block cluster tree or, equivalently, the hierarchical partitioning of the index set are decided. However, the index sets can be decided by geometric information based on the underlying problem. This means if enough is known about the context of the problem, we can relatively easily find the index sets, which then gives us a hierarchical partitioning. However, in the event that nothing is known, other partitioning strategies can be utilized.

Second, and maybe the biggest concern, not every matrix can be represented as an $\mathcal{H}-$matrix. As we saw above, an $\mathcal{H}-$matrix is a very specific structure that assumes we have a block cluster tree with certain index sets as well as some admissible function. Then, it makes this requirement about the rank of sub-matrices in relation to the rank of A for each admissible leaf. With how specific this definition is, matrices usually can't be represented as an $\mathcal{H}-$matrix. This implies that we must then try to approximate the blocks that are admissible by rank-r matrices.

We then will go through general ways of approximating a given matrix A by a rank-r matrix. We will do so by incorporating the Euclidean and Frobenius norms, with the former being a better approximation error and the latter being more easily accessible. As before, we will let $A \in \mathbb{R}^{m \times n}$ denote our given matrix block.

Theorem 1. *Let $A \in \mathbb{R}^{m \times n}$ be given such that $A = U\Sigma V^T$, where $U = (\boldsymbol{u}_1, ..., \boldsymbol{u}_m) \in \mathbb{R}^{m \times m}, V = (\boldsymbol{v}_1, ..., \boldsymbol{v}_n) \in \mathbb{R}^{n \times n}$ are both orthogonal matrices and $\Sigma = diag(\sigma_1, ..., \sigma_n) \in \mathbb{R}^{m \times n}$ with $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$. Then, the matrix*

$$A_r := \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T \tag{24}$$

gives the best approximation to A from all rank-r matrices in the Euclidean and Frobenius norms. This means it satisfies the following:

$$\begin{aligned} ||A - A_r||_2 &= min\{||A - B||_2 | B \in \mathbb{R}^{m \times n} \ with \ rank(B) \leq r\}, \\ ||A - A_r||_F &= min\{||A - B||_F | B \in \mathbb{R}^{m \times n} \ with \ rank(B) \leq r\}. \end{aligned} \tag{25}$$

Furthermore, the norm of the errors $A - A_r$ can be expressed as

$$\begin{aligned} ||A - A_r||_2 &= \sigma_{r+1} \\ ||A - A_r||_F &= \left(\sum_{k=r+1}^{n} \sigma_k^2 \right)^{\frac{1}{2}} \end{aligned} \tag{26}$$

using the singular values of A.

From here, as an example, if we had a given matrix $A \in \mathbb{R}^{n \times n}$ that was symmetric, we could choose $U = V$. Then, the singular values $\sigma_i \in \Sigma$ would become the eigenvalues of A. We can see it is much easier to obtain a symmetric matrix than it is to obtain an $\mathcal{H}-$matrix. From this, we get the norm of the approximations to be

$$\begin{aligned} ||A - A_r||_2 &= \lambda_{r+1} \\ ||A - A_r||_F &= \left(\sum_{k=r+1}^{n} \lambda_k^2 \right)^{\frac{1}{2}}. \end{aligned} \tag{27}$$

Using this method above does indeed give the best possible rank-r approximation to a given matrix. However, as the matrix blocks get larger and larger, the computational cost get increasingly more expensive. We will now discuss a method that is computationally more feasible than the one above using singular value decomposition. First, we use the following definition.

Definition 10. *Let $A \in \mathbb{R}^{m \times n}$ be given. Then, the adaptive cross approximation (ACA) of A is recursively defined as follows. We set $A_0 := 0$ and $R_0 = A$. Then, for k=1,2,...*

- *choose $i_k \in \{1, ..., m\}, j_k \in \{1, ..., n\}$ such that $p_k := (R_{k-1})_{i_k, j_k} \neq 0$,*

- *define*

$$\begin{aligned}\boldsymbol{u}_k &:= \frac{1}{p_k} R_{k-1}\boldsymbol{e}_{j_k} \\ \boldsymbol{v}_k^T &:= \boldsymbol{e}_{i_k}^T R_{k-1} \\ A_k &:= A_{k-1} + \boldsymbol{u}_k \boldsymbol{v}_k^T \\ R_k &:= R_{k-1} - \boldsymbol{u}_k \boldsymbol{v}_k^T .\end{aligned} \tag{28}$$

After r steps of ACA, we have matrices A_r, R_r. These satisfy the equation $A_r + R_r = A$ and

$$A_r = \sum_{k=1}^{r} \boldsymbol{u}_k \boldsymbol{v}_k^T =: U_r V_r^T . \tag{29}$$

Now that we have the definition of this approximation, there are some things to consider before implementing the method in a practical application. First, the method should stop if the error $||R_r||$ become less than some tolerance $\varepsilon > 0$. We have a choice for the norm to be either the Frobenius or Euclidean norm. We had mentioned previously that the Frobenius norm is usually easier to compute in our circumstances. Despite this, the complexity to compute the error is not linear. This means in practical applications, one would normally look at the difference of two errors $R_k - R_{k-1}$ and find their norm as

$$||R_k - R_{k-1}||_2 = ||\mathbf{u}_k \mathbf{v}_{k^T}||_2 = ||\mathbf{u}_k||_2 ||\mathbf{v}_k||_2, \tag{30}$$

which can be computed in linear complexity.

The next thing to consider is the indices i_k, j_k. If no outside information is known, we can choose i_k randomly or choose it as the index such that

$$|\mathbf{e}_{i_k}^T \mathbf{u}_{k-1}| = \underset{1\leq i\leq m}{max} |e_i^T \mathbf{u}_{k-1}. \tag{31}$$

Ideally, however, we would know more about the origin of the given matrix. This would allow us to choose a better i_k. After making this choice, the next index j_k is usually picked such that

$$|(R_{k-1})_{i_k,j_k}| = |\mathbf{e}_{j_k}^T \mathbf{v}_k| = \underset{1\leq j\leq n}{max} |\mathbf{e}_{j_k}^T \mathbf{v}_k| = \underset{1\leq j\leq n}{max} |(R_{k-1})_{i_k,j})|. \tag{32}$$

Lastly, it's important to understand we don't need to build the matrices A_k, R_k completely. The kth step only needs to have the j_kth column and i_kth row of R_{k-1} to compute R_k. This means instead of computing two separate matrices r times, which would be very expensive in space and time, we can simplify the work drastically.

In this section, we have built up the definition of a hierarchical matrix. We defined all the necessary parts up to it and showed the use of these $\mathcal{H}-$matrices. We then discussed how uncommon it is for a matrix to be able to be represented as an $\mathcal{H}-$matrix. As a result, we saw a way to approximate the matrix using a singular value decomposition and adaptive cross approximation. We will now examine examples of these methods.

5 Examples of Hierarchical Matrix Methods

In this section, we will look at practical examples of a singular value decomposition as well as for an adaptive cross approximation.

5.1 Singular Value Decomposition

In [7], the authors discuss the singular value decomposition method we described above. They mention its usefulness for three separate observations. These are the political positions of representatives, the growth rate of crystals in igneous rock, and entanglement in quantum computing. We will then summarize each of these examples the authors give.

5.1.1 Voting Habits of Representatives

First, they look at the voting similarities among politicians. They then defined an $m \times n$ matrix A such that the rows represented all m legislators and the columns represented all n bills they voted on. The element A_{ij} represents how legislator i voted on bill j. If they voted "yea" on the bill, the element is $+1$, "nay" means the element is -1, and a lack of a vote corresponds to 0. The authors also note that the sign of their votes has no connection to the liberal or conservative nature of the politicians' voting habits. We know that this then produces a relatively large matrix, but not one too large the the method is far too expensive. Similarly, we know that the matrix will be dense as representatives usually put in some vote one way or another on each bill. This tells us that singular value decomposition can be utilized.

Then, the authors take the singular value decomposition of A to find which representatives voted similar on bills. As is mentioned in Theorem (1), the singular value decomposition of A is given by

$$A_r = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T. \tag{33}$$

Then, we can truncate matrix A to be

$$A_k = \sum_{j=1}^{k} \sigma_j \mathbf{u}_j \mathbf{v}_j^T \approx A, \tag{34}$$

such that $k \leq r$. As the authors mention in their paper, a representative's voting habits only needs two coordinates. These are based on partisan and bipartisan issues. The partisan coordinate is a measure of how often the representative votes in favor of their party, while the bipartisan coordinate is a measure of how often they vote with the majority of their Congress.

As a result of needing only two coordinates, we can approximate A with A_2. This ends up being a *2-mode truncation* of A. They then use this information to construct a plot where the x-coordinate represents the representatives' affinity for voting with their fellow party members and the y-coordinate represents their bipartisan votes. As should be expect, Democrats were mostly separated completely from Republicans in their voting habits. However, there were indeed some instances of Democrats being more conservative than some Republicans.

On top of this initial graph, the authors also created a graph to show the predictability of the representatives' votes. They did so by examining matrix A_2 again and analyzing the signs of the matrix elements. What they found was that the moderate voters, or members of the Independent party, were usually considerably more unpredictable in their votes than the rest of Congress. On the opposite side of things, if a representative had a higher magnitude for their partisan coordinate, it tended to translate to a higher predictability. This means if a member of Congress was more likely to vote with their party, than their bipartisan votes were more predictable.

5.1.2 Crystal Growth Rates

Next, the authors focus on crystal growth rates in igneous rocks. They do so by focusing on 3D *crystal size distributions* (CSD), which help give important information in regards to the crystallization rates. As of the publication of the paper, determining the grain sizes becomes difficult as the grain shapes become irregular. As a result, singular value decomposition was proposed to measure the CSD for the crystal shapes prism, plate, and cuboid. We find that the CSDs in these rocks are nearly linear, which means we only need the following laws.

- *Nucleation Rate Laws:* $N(t) = e^{\alpha t}$, where N is the number of new nuclei formed at each step t and α is the nucleation constant.

- *Crystal Growth Rate Law:* $G = \frac{\Delta L}{\Delta t}$, where $\frac{\Delta L}{\Delta t}$ is the rate of change of grain diameter per time step.

Next, the authors illustrate approximating the size and shape of a grain with two ellipsoids. The first ellipsoid encloses the crystal's grain and the second inscribes the grain. They then are focused on comparing the results of the enclosing ellipsoid, the inscribed ellipsoid, and the mean of the two. To determine all three of these measures, singular value decomposition is employed.

After using this decomposition, we can calculate the ith radius of the ellipsoid, which is D-dimensional, to be $r_i = \frac{1}{\sqrt{\sigma_i}}$. We say σ_i is the ith singular value in A and that $i \in \{1, ..., D\}$. As the paper mentions, there was great difficulty in determining the size and shapes of these grains, especially because of their irregularity. At the time, it was suggested that at least one of the diameters or radii of the inscribed ellipsoid could be used to determine the size of the grain. If that were the case, then using the singular value decomposition as described above could be used to solve this tricky problem. However, as of the publication of this article, the problem remains unsolved.

5.1.3 Quantum Information

Last, the authors delve into *quantum information* applications of the method. This refers to information held in the state of a quantum system. Whenever dealing with topics that are "quantum", the term *entanglement* usually comes up, which is a type of correlation that is much stronger than the standard meaning of correlation. Quantum computation relies heavily on the inseparability of certain states, which is measured using entanglement.

Before building up the matrix the authors use, we need to understand what is meant by *distinguishable particles*. As mentioned in [8], particles are said to be *indistinguishable* if all of their fundamental properties are exactly the same. That is, if we have two particles and we find that their mass, spin, charge, and any other intrinsic property is the same between the two, then the particles are indistinguishable. As an example, any two electrons, protons, or atoms will be indistinguishable from one another. This basically means there is no way to tell one particle apart from the other.

Therefore, particles are *distinguishable* if there is some difference in their properties. We then suppose we are observing two distinguishable particles A and B. We then can use what is known as Dirac's bra-ket notation to write a joint pure-state wave function as $|\Psi\rangle$. As mentioned in [9], $|\Psi\rangle$ is a unit column vector. Likewise, we also have the Hermitian conjugate of this, which is the row vector $\langle\Psi|$.

Now, we will assume we have some $m \times n$ probability matrix C such that $tr(CC^{\dagger 1}) = tr(C^{\dagger}C) = 1$. Then, if we expand $|\Psi\rangle$, the prefactor for each term consists of each component C_{ij} of C. We then can apply singular value decomposition to C as well as transforming to a singular particle basis. This means we can then diagonalize $|\Psi\rangle$, which is said to be entangled if more than one singular value is nonzero. From here, we can determine if two particles are entangled or not by calculating their entropy.

5.2 Adaptive Cross Approximation

In this section we will explore two examples of the ACA method being used practically. The first example will deal with the voltage capacity of a transformer. We will see how ACA can be used to effectively model a transformer while also being more efficient than the standard method. Next, we will see how ACA can be used to benefit medical professionals who perform transcranial magnetic stimulation on patients.

5.2.1 Transformer Capacitance

Transformers commonly experience high-frequency overvoltages, which last a short moment. As is expected, it is not ideal for transformers to be reaching and surpassing their voltage capacity, so understanding how best to simulate these bursts is necessary. The paper in [10] aims to do just this. In section 3.1, we looked at the acoustic load on the exterior of a car model and how to predict its effects. The authors compared the BEM technique with a multipole method to show the standard method wasn't the most efficient. Similarly, this paper will be comparing BEM with ACA to attempt to find a better simulator for a transformer experiencing an overvoltage.

The way that the two methods will be compared will be based on the results of the finite element method (FEM), which is a technique for solving PDEs as stated in [11]. Using these methods, the authors built a mesh over the model of the transformer. For the FEM portion, they had a mesh consisting of 6,063,470 tetrahedra with it taking 18,060 seconds, or a little over 5 hours, to solve. Conversely, the BEM mesh had 115,216 elements and took 11,338 seconds, or a little over 3 hours, to solve. From this, we can already see that BEM is a cheaper technique than FEM. This is because BEM places its emphasis on the boundary of the object, which minimizes the number of elements in the matrix.

The authors then note the capacitance of the outer coil of the top disk in nanofarads nF, which is a unit of capacitance. For BEM, FEM, and the new BEM-ACA methods the authors got the results of 0.11134 nF,

[1] The symbol † represents the Hermitian conjugate of the matrix C.

0.11094 nF, and 0.11121 nF respectively. This shows that for this particular test, the BEM-ACA method arrives at a comparable result to the other two. Next, they display the calculated capacitance between the outer coil of the top disk and the inner coil in the same phase and found the results 0.02784 pF, 0.03304 pF, and 0.02697 pF, where pF represents picofarads. Once again, these results are all very close to each other, showing that BEM-ACA is as accurate as the methods of the time.

The authors do two more of these calculations and find similar results. This shows that when calculating the capacitance between certain coils in a given phase, the three methods give very similar results. This means the ACA method appears to be just as accurate as the other two. However, we need more compelling evidence of ACA being a useful method. This means it needs to outperform BEM and FEM in some metric.

To show the benefit of this new method, we can look at matrix compression rates CR, which are defined as the ratio between the uncompressed size of the matrix with the compressed size. Doing so gives the following image.

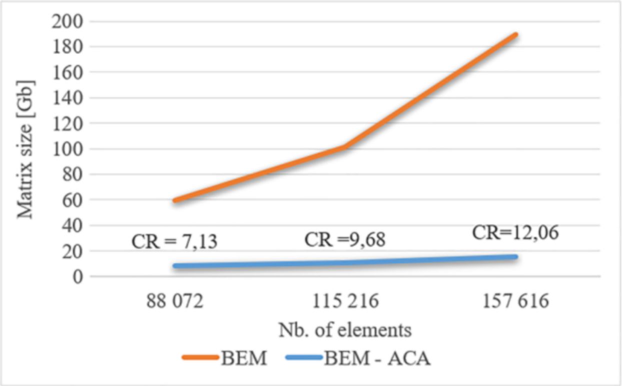


Figure 4: Comparison of matrix sizes for methods BEM and BEM-ACA

This shows that BEM has a change in slope that is not linear as the slopes to the left and right of 115,216 elements are clearly very different. This is because BEM requires space of complexity $O(n^2)$. Alternatively, BEM-ACA doesn't have as drastic of a change at that point. Overall, this method appears to be near linear. As it turns out, BEM-ACA has a complexity of $O(n \cdot log(n))$. This is drastically more efficient than BEM. We can also observe that not only does BEM grow much faster than BEM-ACA, but it also starts much higher than it from the start. That means BEM will always require a significantly larger matrix, which means it requires more memory. We then have that BEM-ACA requires less memory to store its information, making it more efficient, without losing any of the accuracy.

5.2.2 Non-Invasive Magnetic Brain Simulation

In paper [12], the authors were interested in transcranial magnetic stimulation (TMS). This procedure involves placing a coil with a low-frequency pulse on the patient's scalp. This pulse then induces an electric field, or E-field, which has previously been shown to correlate well when the outside layer of the brain is stimulated. What this means is that it's very important to know information about the current E-field to determine the placement of the coil and the intensity of the pulses. The reason that TMS is so important for us to understand is its connection with mental health problems and addiction. TMS can be used to treat symptoms for depression, migraines, OCD and is approved to help people quit smoking, see [13]. It also has potential in other conditions, such as epilepsy. This means if medical professionals better understand the E-fields, they can better treat these ailments.

The overall goal of this study was to see if ACA was a valid method to measure the E-field. While other methods do exist and are accurate, it is important to have the computations completed as quickly as possible. This will lead to faster adjustments of the coil as needed. The authors measured the E-field and put each of the results into a matrix. Each of the columns of this matrix represent the E-field measurement at a given coil placement. The rows represent the E-field measurement for all possible coil placements. Then, given a few sample rows and columns, the authors use ACA to attempt to approximate the whole matrix.

The researchers had two models of a head. One was spherical and the other was MRI-based. They were focused on finding the E-field distribution in a 2 cm region of the head. For the spherical option, the method looked at about 1,000 coil positions and 10 simulations of the "head". For the MRI-derived model, the method had only 360 coil positions and 20 simulations of the head. In both cases, the method had a margin of error that was less than 2%. Furthermore, the time spent on the MRI-derived model was less than 20 minutes. When compared with the results of another method the team has proposed in the past, auxiliary dipole method (ADM), ACA is dramatically faster. This is demonstrated in the graph below.

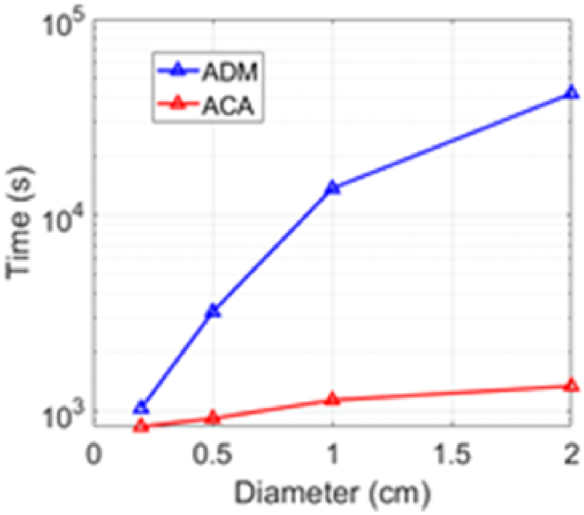


Figure 5: Comparison of time (s) taken on calculating the E-field distribution for methods ACA and ADM

We then see that for diameters arbitrarily close to 0 cm, there isn't much difference in the time taken. However, as we get to 0.5 cm, the gap starts to widen. By the time we get to the tested 2 cm, ADM takes over 10 times as long to calculate as ACA does. This led to the paper's findings, which showed ACA was a valid method for fast computation of coil placement as well as intensity. It is also able to give the E-field distribution within a desired radius for every coil placement.

6 Conclusion

We have introduced the concepts of the multipole methods as well as giving practical examples of why they may come in use. We discussed how car manufacturers need to account for outside acoustics and need to be able to predict how they affect the body. We also mentioned the traditional method of BEM is not as efficient as a multipole method. We also talked about how multipole methods can be used to solve a spherical harmonics transform, which is a type of PDE. This shows that not only is the multipole method effective in real world problems, but it also is applicable in a wide variety of areas.

We then built up the definition of a hierarchical matrix by discussing partitionings, both tensor and hierarchical partitionings, admissible functions, and trees. We mentioned how effective these $\mathcal{H}-$matrices can be if we can convert our target matrix into one of them. However, because most matrices can't be represented as an $\mathcal{H}-$matrix, we had to discuss singular value decomposition methods. For this, we saw how it could have applications in understanding politicians' voting records, crystal growth rates, and even quantum computing. We then mentioned that while singular value decomposition is effective, it becomes very expensive for very large matrices. As a result, we needed to bring in adaptive cross approximations. We then showed how ACA is more effective than BEM at modeling a transformer's capacity as it yields a smaller matrix without losing the accuracy. We also showed how ACA was used to more quickly get information about the E-field to better know where to place the coil during treatment.

With all of these topics dealt with, we have seen how solving large dense matrices can be useful for car manufacturing, PDEs, political science, geology, computer science, electrical engineer, and medical professionals.

7 Sources

[1] *Mercedes Marqués, Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Solving "Large" Dense Matrix Problems on Multi-Core Processors, 2009*

[2] *https://cs357.cs.illinois.edu/textbook/notes/sparse.html, Gives a definition of sparse matrices*

[3]*Holger Wendland, Numerical Linear Algebra, An Introduction, 2018*

[4]*Sébastien Chaigne, et al, On the Use of Fast Multipole Methods for Accurate Automotive Body Panel Acoustic Load Predictions, 2007*

[5]*Reiji Suda,Asayasu Takami, A Fast Spherical Harmonics Transform Algorithm, 2001*

[6]*Charles Martin, A Gentle Introduction to Harmonic Functions, April 19 2010*

[7]*Carla D. Martin and Mason A. Porter, The Extraordinary SVD, 2012*

[8]*Claude Cohen-Tannoudji, Bernard Diu, and Franck Laloe, Quantum Mechanics, 1973*

[9]*University of Oregon, Mixed states and pure states, 2009*

[10]*Ana Drandíc and Bojan Trkulja, Transformer capacitance matrix computation using 3D boundary element method and adaptive cross approximation, 2020*

[11]*https://www.comsol.com/multiphysics/finite-element-method, Defines FEM*

[12]*Dezhi Wang et al, ADAPTIVE CROSS APPROXIMATION FOR E-FIELD-GUIDED NONINVASIVE MAGNETIC BRAIN STIMULATION, 2021*

[13]*https://www.mayoclinic.org/tests-procedures/transcranial-magnetic-stimulation/about/pac-20384625, Gives an understanding of the importance of TMS*