

# Research Internship (PRe)

**Field of Study: Applied Maths**
**Scholar Year : 2023-2024**

# Hierarchical Matrices and Inexact GMRES

**Confidentiality Notice**
Non-confidential and publishable report


**Author: GUIMARAES Eduardo** **Promotion: 2025**

**ENSTA Paris Tutor: FARIAS Luiz**
**MARCHAND Pierre**
**Host Organism Tutor: CHAPOUTOT Alexandre**


**Internship from 13/05/2024 to 15/08/2024**


**POEMS**
**Address: 828, Boulevard des Maréchaux, 91762 Palaiseau France**

This page was intentionally left blank.

# Confidentialy Notice

This present document is not confidential. It can be communicated outside in paper format or distributed in electronic format.

**Abstract**

In this project we will focus on combining a recently develop inexact GMRES algorithm, which differs from the classic GMRES algorithm in that the underlying linear system is allowed to change at each iteration, and hierarchical matrix approximation to a boundary integral discretisation. The main goal is to (i) understand the inner workings of GMRES and how changing the linear system at each iteration affects the convergence properties, and (ii) explore whether inexact GMRES has practical interest when combined with a variable precision -matrix compression. This project will be carried out in the context of two existing libraries: HMatrices.jl and Inti.jl (under development). The internship will take place at the POEMS laboratory, and will be supervised by Luiz M. Faria (chercheur INRIA) and Pierre Marchand (chercheur INRIA).

***Keywords***— GMRES, BEM, applied maths, acceleration, Julia

# Acknowledgments

I am deeply grateful to all those who have contributed to the successful completion of this research project during my internship at Stark Industries.

First and foremost, I would like to express my sincere gratitude to Mr. Tony Stark for providing me with this remarkable opportunity. His visionary guidance and insightful feedback have been invaluable throughout the research process.

I would also like to extend my heartfelt thanks to Dr. Bruce Banner for his mentorship and expert advice. His profound knowledge in the field and his willingness to share it have significantly enriched my understanding and approach to the research.

My appreciation extends to the entire team at Stark Industries for their support and collaboration. Special thanks to Ms. Pepper Potts and Happy Hogan for theirs exceptional administrative support and ensuring a conducive working environment.

I am also grateful to my academic advisor at Midtown School of Science and Technology, Dr. Curt Connors, for his continuous encouragement and academic support.

Finally, I wish to thank my family and friends, especially my Aunt May, for their unwavering support and encouragement, which have been a constant source of motivation.

Thank you all for your invaluable contributions to this research.

# Contents

# List of Figures

# Chapter 1

# Iterative Methods and Krylov's Subspace

## 1.1 Iterative Methods and motivation

Iterative methods appear as an alternative to direct solution methods, where the direct solution to a problem usually scales with $\mathcal{O}(m^3)$ complexity, where $m$ is the dimension of the input matrix. Since larger matrices are usually employed in practice, the direct algorithms become inefficient, and we require a more reliable approach.

The idea of the iterable methods is to find, after a certain number of iterations, a sequence $x_k$ that converges to $x$, the exact solution of the problem 1.1. This should be done while making the large-scale computations faster, i.e., obtaining a complexity smaller than $\mathcal{O}(m^3)$, and keeping a maximum tolerance between the iterable solution and the exact one.

$$x = \lim_{k \to \infty} x_k \tag{1.1}$$

The method stops after $k$ iterations, where $x_k$ is the fist element of the sequence to satisfy the condition 1.2.

$$\frac{||Ax_k - b||}{||b||} \leq \epsilon \tag{1.2}$$

We define $\epsilon$ as the tolerance given to the algorithm.

To achieve a smaller complexity than $\mathcal{O}(m^3)$, Iterative Methods employ matrix vector products, complexity $\mathcal{O}(m)$, instead of the product between matrices found in direct methods. So, considering an Iterative Method finds a solution in $k$ steps, its complexity would be $\mathcal{O}(km^2)$.

Therefore, guaranteeing that the convergence rate of the method is sufficiently fast gives a $k << m$, and the Interative Method can be way more efficient thatn its counterpart.

The method we employ in our problems is the GMRES, explained later in the report. Its main ideia involves the projection of a high dimensional problem, as large as $A$ in 1.2, in a lower dimensional *Krylov Space*:

$$x_k \in \text{span}(b, Ab, A^2b, ..., A^{k-1}b) \tag{1.3}$$

Explained in more detail below.

## 1.2 Krylov's Subspace

Be $A \in \mathbb{K}^{n\times n}$ a matrix and $b \in \mathbb{K}^n$. To each $k \leq n$ the Krylov's Subspace $\mathcal{K}_k = \mathcal{K}_k(A, b)$ associated to A, b is defined as 1.4.

$$\mathcal{K}_k(A, b) = \text{span}(b, Ab, A^2b, \ldots, A^{k-1}b) \tag{1.4}$$

These Subspaces also have the following property: $k < l \rightarrow \mathcal{K}^k \subset \mathcal{K}^l$ [5].

The subspace $\mathcal{K}_k(A, b)$ is also the subspace of all the vectors from $\mathbb{R}^m$ which could be written as $x = p(A)b$, where $p(A)$ is a polynom of degree less than $k-1$ which $p(0) = 1$.

The problem with using $A^kb, k \in 0, 1, 2, \ldots$ as a base comes from the fact that successive products of $A$ make vectors that are *approximately colinear*, since those are really close of the eigenvector with the largest eigenvalue of $A$.

## 1.3 Arnoldi's Method

Arnoldi's Method is an orthogonal projection method used to find an orthonormal basis $q_1, \ldots, q_k$ to $\mathcal{K}_k(A, b)$. An algorithm for the method can be found in 1.

**Algorithm 1** Arnoldi's iteration

1: $A \in \mathbb{K}^{n\times n}$ et $b \in \mathbb{K}^n$
2: $x = 0, \beta = \|b\|, q_1 = \frac{b}{\beta}$
3: **for** $j = 1, 2, \ldots k$ **do**
4: $\quad q_{j+1} = Aq_j$
5: $\quad$ **for** $i = 1, 2, \ldots j$ **do**
6: $\quad\quad h_{ij} = q_{j+1}^t q_i$
7: $\quad\quad q_{j+1} = q_{j+1} - h_{ij}q_i$
8: $\quad$ **end for**
9: $\quad h_{j+1,j} = \|q_{j+1}\|$
10: $\quad q_{j+1} = \frac{q_{j+1}}{h_{j+1,j}}$
11: **end for**

As we can see, at each step in 1, the previous vector $q_j$ is multiplied by $A$ and then orthonormalized in relation to all previous $q_i$'s with a Gram-Schmidt procedure. If $q_{j+1}$ ever vanishes during the inner loop between lines 5 and 8, the algorithm stops.

What is left is to show the $q_i$ generated by 1 form an othornormal basis for $\mathcal{K}_k(A, b)$.

*Proof.* By construction $q_j$, $j = 1, 2, \ldots, k$ are othornormal. To show they span $\mathcal{K}_k(A, b)$ we prove $q_j$ has the form $p_{j-1}(A)b$, where $p_j(A)$ is a polynomial of degree $j-1$ in A. Using induction the result is true for $j = 1$ since $q_1 = b$. We assume the result is true for all integers $\leq j$ and consider $q_{j+1}$. Using the definition of $q_{j+1}$ in 1 we have:

$$h_{j+1,j}q_{j+1} = Aq_j - \sum_{i=1}^{j} h_{ij}q_i = Ap_{j-1}(A)b - \sum_{i=1}^{j} h_{ij}p_{i-1}(A)b \tag{1.5}$$

Since, by the induction step above, $q_i = p_{i-1}(A)b$.

This shows $q_{j+1}$ can be written as $p_j(A)b$ and completes the proof. □

We also make note of the fact $q_1 = \frac{b}{||b||}$.

If we denote by $Q$ the $nxk$ matrix with column vectors $q_1, \ldots, q_k$ found in 1 and $H_k$ the $(k+1)xk$ Hessenberg matrix whose nonzero entries $h_{ij}$ are given just as in 1, we have 1.6.

$$AQ_k = Q_{k+1}H_k \tag{1.6}$$

*Proof.* For each column-vector of $Q$, $q_i$, 1.6 could be written as 1.7, where the representation of $\mathcal{K}_k(A, b)$ with an orthonormal basis becomes more evident.

$$Aq_m = h_{1m}q_1 + h_{2m}q_2 + \dots h_{m+1,m}q_{m+1} \tag{1.7}$$

This relation can be directly seen in 1 by using line 10 and the inner loop between lines 5 and 8:

$$\begin{aligned} q_{m+1}h_{m+1,m} &= Aq_m - \sum_{i=1}^{m} h_{im}q_i \\ Aq_m &= \sum_{i=1}^{m+1} h_{im}q_i \end{aligned} \tag{1.8}$$

□

# Chapter 2

# GMRES

A projection in $\mathcal{K}_k(A, b)$, where we take the different approximations as in 2.1, where $Q_m$ is the vector in 1.6.

$$x = x_0 + Q_m y \tag{2.1}$$

With 2.1 and 1.6 the residual becomes 2.2, where $x_0 = 0$, $\beta = \|b\|$ and $Q_{m+1}^t b = (\|b\| \, 0 \, 0 \ldots)^t$ since the columns of $Q_{m+1}$ are orthonormal vectors and $q_1 = \frac{b}{\|b\|}$.

$$\begin{aligned} r(y) &= \|b - Ax\| \\ &= \|b - A(Q_m y)\| \\ &= \|b - Q_{m+1} H_m y\| \\ &= \left\|Q_{m+1}(Q_{m+1}^t b - H_m y)\right\| \\ &= \|\beta e_1 - H_m y\| \end{aligned} \tag{2.2}$$

Thus, $y$ which appears in 2.1, is found as the solution of the residual's minimisation problem in 2.2.

$$y = \min_y \|\beta e_1 - H_m y\| \tag{2.3}$$

An initial version of the GMRES is in 2. The lines 4 to 12 contain the Arnoldi's Method presented in 1.

However, 2 doesn't present an efficient way of finding the residual in each iteration. To solve this problem and also to find a more efficient way of solving the least squares in 2.3, we apply a transformation to $H_m$, turning it into a triangular matrix.

## 2.1 Givens's Rotation

Givens's operator, $G(i, i+1)$, is a unitary matrix such that the column vector $a = Gb$ has the elements $a(i) = r \in \mathbb{R}$ and $a(i+1) = 0$. It has a structure as in 2.4. The coefficients $c_i, s_i$ only appear in the rows $i$ et $i+1$.

**Algorithm 2** Initial GMRES

$A \in \mathbb{K}^{n\times n}$ and $b \in \mathbb{K}^n$
$x = 0, \beta = \|b\|, q_1 = \frac{b}{\beta}$
**for** $k = 1, 2, \ldots$ **do**
  **for** $j = 1, 2, \ldots k$ **do**
    $q_{j+1} = Aq_j$
    **for** $i = 1, 2, \ldots j$ **do**
      $h_{ij} = q_{j+1}^t q_i$
      $q_{j+1} = q_{j+1} - h_{ij} q_i$
    **end for**
    $h_{j+1,j} = \|q_{j+1}\|$
    $q_{j+1} = \frac{q_{j+1}}{h_{j+1,j}}$
  **end for**
  Find $y = \min_y \|\beta e_1 - H_m y\|$
  $x = Q_k y$
  **Stop** if the residual is smaller than the tolerance
**end for**

$$G(i, i+1) = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \tag{2.4}$$

This operator offers a way to transform the columns in $H_m$, *zeroing* the elements outside the main diagonal. Since a product of unitary operators is still unitary, 2.3 can be written as 2.5, where $R_m$ and $g_m$ are the results from the application of multiple Givens's operators to $H_m$ and $\beta e_1$.

$$y = \min_y \|\beta e_1 - H_m y\| = \min_y \|g_m - R_m y\| \tag{2.5}$$

Thus, the new problem 2.5 can be solved with a simple backwards substitution. If $g_m = [\gamma_1 \ldots \gamma_{m+1}]^t$, an $m+1$ column vector, and $\{R_m\}_{ij} = r_{ij}$ an $m+1$ by $m$ upper triangular matrix with $r_{ii} \neq 0$ and its last row filed with zeros, each element of $y_m = [y_1 \ldots y_m]$ is given by 2.6.

$$\begin{aligned} \gamma_k &= \sum_{i=k}^{m} r_{ki} y_i \\ y_m &= \frac{\gamma_m}{r_{mm}} \\ y_i &= \frac{1}{r_{ii}} \left( \gamma_i - \sum_{j=i+1}^{m} r_{ij} \gamma_j \right) \end{aligned} \tag{2.6}$$

A simple algorithm to this end can be written as 3.

**Algorithm 3** Backwards substitution

1: $A \in \mathbb{K}^{n\times n}, \{A\}_{ij} = a_{ij}$ and $b \in \mathbb{K}^n$
2: **for** $k = n, n-1, \ldots$ **do**
3: $\quad y_k = b_k$
4: $\quad$ **for** $j = n, n-1, \ldots k+1$ **do**
5: $\quad\quad y_k = y_k - a_{kj}y_j$
6: $\quad$ **end for**
7: $\quad y_k = \frac{y_k}{a_{kk}}$
8: **end for**

It can be shown that $g_m$ also contains the residual of each iteration [7]. Since it's an $m+1$ column vector, we have, with $\Omega_m$ being the necessary Givens's Rotations to make $H_m$ upper triangular 2.7.

$$\begin{aligned} \|b - Ax_m\| &= \left\|Q_{m+1}^t(\beta e_1 - H_m y_m)\right\| \\ \|\beta e_1 - H_m y_m\| &= \left\|\Omega_m^t(g_m - R_m y_m)\right\| \end{aligned} \tag{2.7}$$

Since $y_m$ is a solution to the system, the norm in 2.7 is defined by $\|\gamma_{m+1} - (R_m)_{m+1,1:m}y_m\|$.

But since the last row in $R_m$ is composed by zeros, we have that $\|b - Ax_m\| = |\gamma_{m+1}|$, which gives a more efficient way to obtain the residuals during each iteration.

## 2.2 Inexact GMRES

The heaviest part in the code is in the matrix-vector product 1, line 4. Therefore, one approach to accelerate the iterations involves an approximation of $Aq$, instead of using the exact answer, as shown in 2.8.

$$\mathcal{A}q = (A + E)q \tag{2.8}$$

Where $E$ in 2.8 is a *pertubation matrix* that changes with each iteration and will be written as $E_k$ for iteration k.

When we realise the inexact matrix-vector product, instead of the regular one, the left side of 1.6 must be changed by 2.9.

$$\begin{aligned} [(A + E_1)q_1, (A + E_2)q_2, \ldots, (A + E_k)q_k] &= Q_{k+1}H_k \\ (A + \mathcal{E}_k)Q_k = Q_{k+1}H_k, \quad \mathcal{E}_k &= \sum_{i=1}^{k} E_i q_i q_i^t \\ \mathcal{A}Q_k &= W_k \end{aligned} \tag{2.9}$$

Where $W_m = Q_{m+1}H_m$ from this point foward.

Now the subspace spawn by the vectors of $Q_k$ is not the Krylov's subspace $\mathcal{K}_k(A, b)$, but these are still orthonormal. To see what kind of subspace our new $Q$ spams, 2.9 is looked into in 2.10.

$$(A + E_k)q_k = \mathcal{A}_k q_k = h_{1,k}q_1 + h_{2,k}q_2 + \cdots + h_{k+1,k}q_{k+1} \tag{2.10}$$

For $k = 1$, we have that $q_2$ is a combination of the vectors $\mathcal{A}_1 b$ and $b$ (since $q_1 = $ b). For $k = 2$ we see that $q_3$ is a combination that involves $\mathcal{A}_2\mathcal{A}_1 b$ and so forth.

Expression 2.9 then shows that $Q_k$ becomes a basis for a new Krylov's subspace, $\mathcal{K}_k(A+\mathcal{E}_k, b)$ = span$\{b, \mathcal{A}_1 b, \ldots, \mathcal{A}_k \ldots \mathcal{A}_1 b\}$, made by a large pertubation in $A$, that gets updated in each iteration.

A new distinction should also be made between the two types of residuals appearing in the process: $r_k$, the exact residual of an iteration, and $\tilde{r}_k$, the one that will really be calculated. A detailed definition for both and a measure of how distant they are is in 2.11.

$$\begin{aligned} r_k &= r_0 - AQ_k y_k \\ &= r_0 - (Q_{k+1}H_k - [E_1 q_1, \ldots, E_k q_k])y_k \\ &= \tilde{r}_k + [E_1 q_1, \ldots, E_k q_k] y_k \\ \rightarrow \delta_k &= \|r_k - \tilde{r}_k\| = \|[E_1 q_1, \ldots, E_k q_k] y_k\| \end{aligned} \tag{2.11}$$

Considering $y_k = [\eta_1^{(k)} \ldots \eta_n^{(k)}]$, upper index to clarify the iteration, an upper bound for $\delta_k$ can be found, but before we go through 2.12.

$$\begin{aligned} \|[E_1 q_1, \ldots, E_k q_k] y_k\| &= \left\| \sum_{i=1}^{k} E_i q_i \eta_i^{(k)} \right\| \\ \left\| \sum_{i=1}^{k} E_i q_i \eta_i^{(k)} \right\| &\leq \sum_{i=1}^{k} \|E_i\| \left\| q_i \eta_i^{(k)} \right\| \\ \left\| \sum_{i=1}^{k} E_i q_i \eta_i^{(k)} \right\| &\leq \sum_{i=1}^{k} \|E_i\| \, |\eta_i^{(k)}| \end{aligned} \tag{2.12}$$

We use the fact that $q_i$ are unitary between the last two lines. The bound on $\delta_k$ is then found in 2.13.

$$\delta_k = \|r_k - \tilde{r}_k\| \leq \sum_{i=1}^{k} \|E_i\| \left\| \eta_i^{(k)} \right\| \tag{2.13}$$

2.13 tells us that in order to keep both residuals close, either the pertubation of $A$, somewhat measured by $\|E_i\|$, or the elements of $y_i$ should be kept small. Since we expect to use more *relaxed* approximations of $A$ as the iterations go on, a greater tolerance in $E_k$ could be compensated with a sufficiently small $y_k$.

The problem is $y_k$ is only found after the construction of $E_k$, so an upper bound must be also found for its value.

Knowing $y_k$ is the solution of the minimisation of $\|H_k y_k - e_1 \beta\|$, we consider $\Omega_k = G(k, k+1)G(k-1, k) \ldots G(1, 2)$ where each $G$ represents a Givens rotation as shown in 2.4, so $\Omega_k$ is the matrix that transforms $H_k$ into an upper triangular matrix.

The aplication of $\Omega_k$ in either side of $H_k y_k = e_1 \beta$ gives us 2.14.

$$\begin{aligned} \Omega_k H_k y_k &= \Omega_k e_1 \beta \\ R_k y_k &= g_k \\ y_k &= R_k^{-1} g_k \end{aligned} \tag{2.14}$$

Since $R_k$, the transformation of a Hessenberg matrix by a series of Givens rotations, is upper triangular, then its inverse also is. Being an upper triangular matrix, the first $i-1$ elements of its ith line are zeros, so using Matlab index notation in 2.15.

$$(R_k^{-1})_{i,1:k} (g_k)_{1:k} = (R_k^{-1})_{i,i:k} (g_k)_{i:k} \tag{2.15}$$

Using this last result in 2.14 gives 2.16.

$$\begin{aligned} |\eta_i^{(k)}| &= \left\|(R_k^{-1})_{i,i:k}(g_k)_{i:k}\right\| \\ |\eta_i^{(k)}| &\leq \left\|e_k R_k^{-1}\right\| \left\|(g_k)_{i:k}\right\| \\ |\eta_i^{(k)}| &\leq \left\|e_k R_k^{-1}\right\| \left\|(g_k)_{i:k}\right\| \end{aligned} \tag{2.16}$$

Since $\left\|e_k R_k^{-1}\right\| \leq \left\|R_k^{-1}\right\| = \sigma_k(H_k)^{-1}$ and $\left\|(g_k)_{i:k}\right\| \leq \|\tilde{r}_{i-1}\|$ [8], the bound is given by 2.17.

$$\left\|\eta_i^{(k)}\right\| \leq \frac{1}{\sigma_k(H_k)} \|\tilde{r}_{i-1}\| \tag{2.17}$$

Putting 2.17 in 2.13 gives the results in 2.18. Setting $\delta_k \leq \epsilon$ and determining a bound for each $\|E_i\|$ gets us 2.19.

$$\delta_k \leq \sum_{i=1}^{k} \frac{\|E_i\|}{\sigma_k(H_k)} \|\tilde{r}_{i-1}\| \tag{2.18}$$

$$\|E_i\| \leq \frac{\sigma_k(H_k)\epsilon}{k \|\tilde{r}_{i-1}\|} \tag{2.19}$$

Since $H_k$ is also one of the matrices being constructed throughout the method, a workaround is necessary to apply find these bounds in a pratical situation. Either using an estimation of $\sigma_k(H_k)$ with the singular values of $A$ or grouping all uncalculated terms in an $\ell_k$ that will be estimated empirically [8], obtaining 2.20.

$$\|E_i\| \leq \ell_k \frac{1}{\|\tilde{r}_{i-1}\|} \epsilon \tag{2.20}$$

It should be noted [8] that among te initial bounds, some aren't really sharp, mainly 2.12 and 2.17, and further empirical analysis of these bounds could show a better theoretical bound can be found for both. A plot of these bounds for the cavity problem that will be studied is shown later.

It should also be noted that $\tilde{r}_k$, after the transformation of $H_k$ into an upper triangular matrix, is also found in the $i+1$'th element of $g_m$ in 2.14. The demonstration follow the same proof as for $g_m$ in 2.5, given that $y_m$ is a solution to a linear system that involves an upper triangular matrix.

The remaining theory in this report also explains the basics of Hierarchical Matrices, the structure that will be used to compress the matrices used in the algorithm, since $A$ appears in the discretization of integral operators and uses large dimensions. It's also though these strucures each $E_k$ will be made. As it will be explained later, at each iteration an $E_k$ will be indirectly constructed during the inexact product $\mathcal{A}_k q$, mainly using the residues that appear during this structure's construction, in the iterations of the *ACAMethod*.

# Chapter 3

# Hierarchical Matrices and ACA Method

## 3.1 Low-rank Matrices

In reality, most matrices are large, so storing each element is not efficient, or even possible. If $A \in \mathbb{C}^{n\times m}$ has a rank $k$ such that $k \leq m$ and $k(n+m) < n*m$ ($A$ is low-rank), $A$ can be written in outer product form, as a product between the matrices $U \in \mathbb{C}^{n\times k}$ and $V \in \mathbb{C}^{m\times k}$, which can be seen in 3.1, where $u_i, v_i$ are the column vectors of $U$ and $V$.

$$A = UV^H = \sum_{i=1}^{k} u_i v_i^* \tag{3.1}$$

Therefore, storing $k(n+m)$ elements to write $A$, and not $n \times m$. A matrix $A$ that can be represented as 3.1 is an element of $\mathbb{C}_k^{n\times m}$.

The representation in 3.1 also facilitates other operations with $A$, like matrix-vector products $Ab$ that are always present in methods like GMRES [4] and different kinds of norms, like $\|A\|_F, \|A\|_2$ [4].

However, even full rank matrices can be approximated by matrices with lower rank. A theorem [4] establishes that the closest matrix from $\mathbb{C}_k^{n\times m}$ of a matrix from $\mathbb{C}^{n\times m}$ can be obtained from the SVD $A = U\Sigma V^H$, where $\Sigma$ contains the singular valuers $\sigma_1 \geq \sigma_2 \ldots \sigma_m \geq 0$ and $U, V$ are unitary.

If $A_k$ is the approximation obtained after taking the first k elements of $\Sigma$ (creating the matrix $\Sigma_k$), the error between $A$ and $A_k$ is 3.2.

$$\|A - A_k\| = \left\|U\Sigma V^H - U'\Sigma_k V_H\right\| = \|\Sigma - \Sigma_k\| \tag{3.2}$$

If the spectral norm, $\|.\|_2$ is used instead, the error in 3.2 is given by $\sigma_{k+1}$. For Frobenius's norm, $\|.\|_F$, the error becomes $\sum_{l=k+1}^{n} \sigma_l^2$.

Instead of approximating large matrices entirely, it's better to think in approximations made to each of their blocks. Blocks that appear after the discretization of elliptic operators also have the possibility of being approximated by matrices that decay exponentially with k, $S_k$, as in 3.3.

$$\|A - S_k\|_2 < q^k \|A\|_2 \tag{3.3}$$

That way the rank, and precision are related logarithmically, and the rank required by a certain $\epsilon$ is 3.4.

$$k(\epsilon) = \min\{k \in \mathbb{N} : \sigma_{k+1} < \epsilon\sigma_1\} \tag{3.4}$$

## 3.2 ACA Method(Adaptative Cross Approximation)

As shown in the last section, the SVD methods gives us an approximation of $A$ given a certain $\epsilon$, through the relation in 3.2. Nevertheless, this is an expensive method, where the complexity becomes too heavy for some calculations.

The algorithm for the method is in 4, where $a_{ij}$ are the elements of a matrix $A \in \mathbb{R}^{n\times m}$. The main objective is to approximate $A$ as $A = S_k + R_k$, $S_k = \sum_{l=1}^{k} u_l v_l^t$ and $R_k$ is the residual.

**Algorithm 4** ACA Method

$k = 1$ et $\mathbf{Z} = \emptyset$
**repeat**
  TFind $i_k$
  $\hat{v}_k = a_{i_k,1:m}$
  **for** $l = 1, \ldots, k-1$ **do**
    $\hat{v}_k = \hat{v}_k - (u_l)_{i_k} v_l$
  **end for**
  $Z = Z \bigcup \{i_k\}$
  **if** $\hat{v}_k$ doesn't disappear **then**
    $j_k = argmax_j |(\hat{v}_k)_j|$ ; $v_k = (\hat{v}_k)_{j_k}^{-1} \hat{v}_k$
    $u_k = a_{1:n,j_k}$
    **for** $l = 1, \ldots, k-1$ **do**
      $u_k = u_k - (v_l)_{j_k} u_l$
    **end for**
    $k = k + 1$
  **end if**
**until** $\|u_k\| \, \|v_k\| \leq \epsilon$

Considering $I, J \in \mathbb{N}$ the index set of a given matrix and $\mathbf{T}_{I\times J}$ the cluster block tree that contains an admissible partition $P$ of $I \times J$ in its leaves, $\mathfrak{L}(\mathbf{T}_{I\times J})$. The set of hierarchical matrices in $\mathbf{T}_{I\times J}$ rank k for each block $A_b$ defined in 3.5.

$$\mathfrak{H}(\mathbf{T}_{I\times J}, k) = \left\{ A \in \mathbb{C}^{I\times J} : rank A_b \leq k, \forall b \in P \right\} \tag{3.5}$$

In practice, 4 works with $\epsilon$ given by the user during the assembling of the Hierarchical Matrix, and the compression acts in each of the admissible blocks that will then be represented as low rank matrices, shown in 3.1. We also store each one of the residuals obtained in the outer loop of 4.

After giving a tolerance $\sigma$ for an inexact product, the algorithm, for each admissible block of the cluster tree, uses the right amount of columns of the outer product representation of the block as to reach the desired tolerance. The number of columns can be infered by using the list of residuals of the ACA Method.

So, the different pertubations $E_k$ used in 2.9 are the matrices that, when added to $A$, leave each admissible block with the right amount $k'$ of columns in 3.1, so the product can be approximated by the tolerance $\sigma$. Calling the approximation of each block $B_k$ as $\tilde{B}_k$, we have 3.6.

$$\frac{\left\| B_k q - \tilde{B}_k q \right\|}{\|B_k q\|} \leq \sigma, \qquad \tilde{B}_k = \sum_{i=1}^{k'} u_i v_i^* \tag{3.6}$$

# Chapter 4

# First results

(Not results in the cluster, left to remake them after finishing the discussions we had about residuals) (I thought about changing the beggining of this chapter as a whole small chapter about BEM)

(Also, I thought about using the graphs with the bounds of each expression in the article, but I'm not shure if we put them here or with the Inexact theory chapter above)

Before using the inexact product in more complex problems, simpler examples are used to validate the approach and fix minor parameters in the scheme.

The two firsts tests evaluate the speedup in the product of a Hierarchical Matrix and a vector and an exection of the Inexact GMRES with few iterations, using the operators obtained through 2nd type Equations of Laplace and Helmholtz 4.1, where the last one is a scaterring problem, where $\Delta$ is the Laplace operator and everything is suposed to be solved in two dimentions. The last test will use a cavity problem to test the speedup of the algorithm in a situation with more iterations.

$$\begin{aligned} \Delta u &= 0 \\ \Delta u + k^2 u &= 0 \end{aligned} \tag{4.1}$$

Reformulationg both equations as a Boundary Integral Equation, the simple *direct* formulation is used to write the solution as 4.2, where $\Gamma$ is boudnary of the domain, $S$ and $D$ are the single and double layer operators, defined as 4.3, and $G(x, y)$ is the fundamental solution of the desired PDE.

$$-\frac{u(x)}{2} + D[u](x) = S[\partial_\nu u](x), \qquad x \in \Gamma \tag{4.2}$$

$$\begin{aligned} S[\sigma](x) &= \int_\Gamma G(x,y)\sigma(y)\, ds(y) \\ D[\sigma](x) &= \int_\Gamma \frac{\partial G}{\partial \nu_y}(x,y)\sigma(y)\, ds(y) \end{aligned} \tag{4.3}$$

For the first to examples, a unit circle around the origin is used as the boundary to generate the operators, with the mesh being created with the Inti library [3].

For the last test, the mesh is made from a cavity *.geo* file avaiable in [1]. A view of the figure can be seen in 4.1. The incident wave' angle is chosen to be $\frac{\pi}{4}$ rad.

A good way to infer the maximum acceleration possible for the inexact products would be using only admissible rank 1 blocks and measuring its execution time. Although such thing would not happen in a practical situation, it gives a maximum bound for the speed up we should expect. For doing that, the product tolerance is changed to *Infinity*, and the product will be

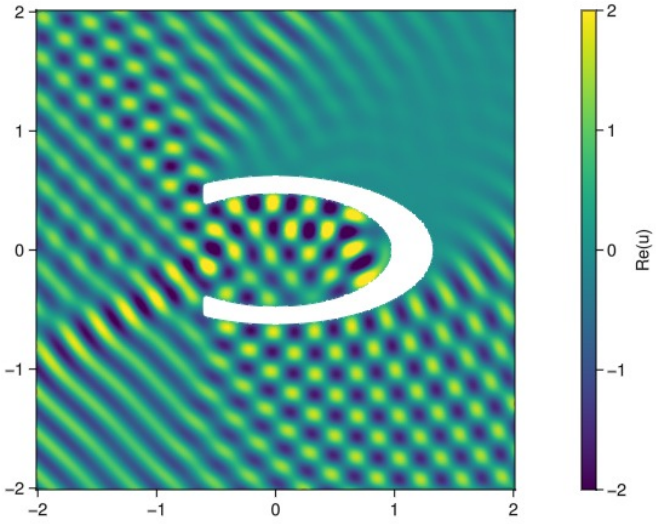



Figure 4.1: Geometry used in the test.

realised with only rank-1 blocks, since it's programmed to get the first aproximation lower than its given tolerance.

## 4.1 Laplace's results

Setting our product tolerance to *Infinity* and using only rank-1 blocks in the product, we got a () speedup.

All results are contained in 4.2, showing the evolution of the residual with the product tolerance aswell as the speedup to each of these values.

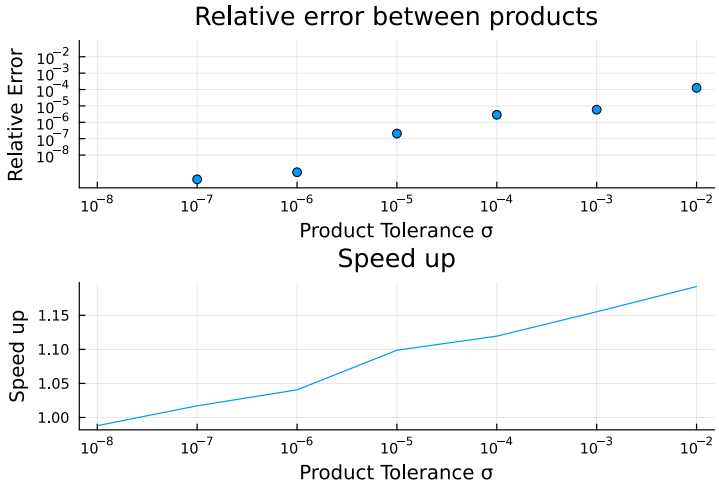
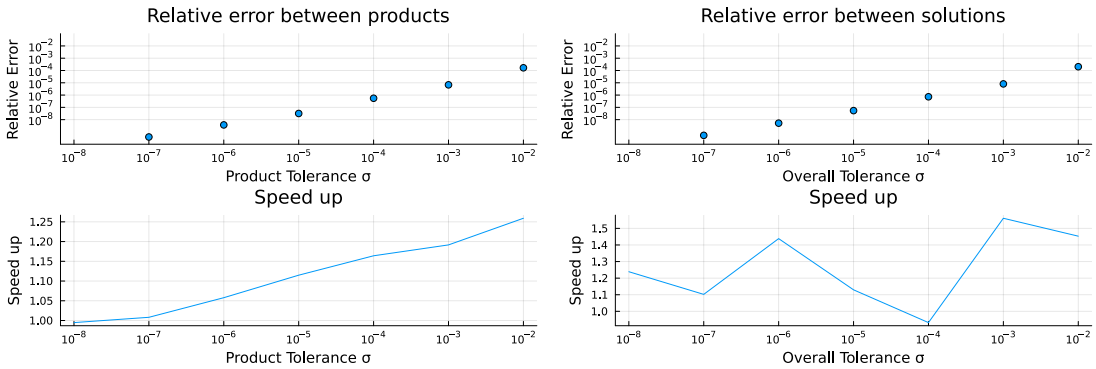



Figure 4.2: Speedup and residual evolution for the product between a 8000x8000 HMatrix and a vector.

## 4.2 Helmholtz's results

For a maximum speedup bound in the product, the infinity tolerance brought a () speedup.

For the unitary circle boundary the results can be seen in 4.3.




(a) Results for the product of a 70000x 70000 HMatrix and a vector.

(b) Results for an initial application of the Inexact GMRES algorithm.

Figure 4.3: Results for the application of the Inexact GMRES algorithm with a 70000x70000 HMatrix.

For the cavity, 4.4.

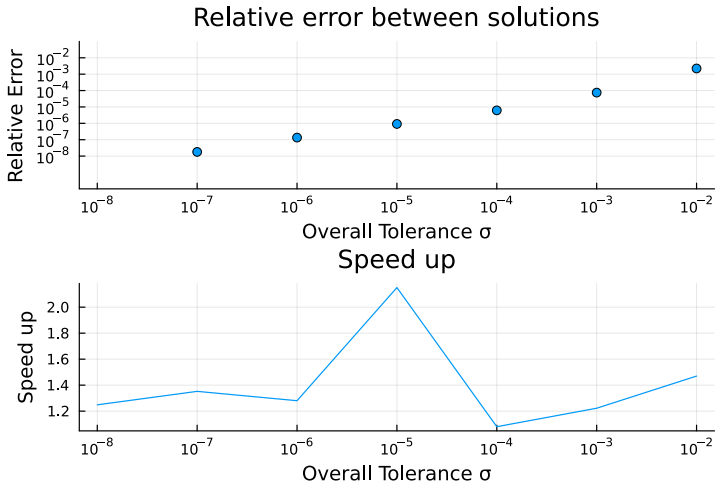



Figure 4.4: Speedup witnessed in the application of the Inexact GMRES in a 50000x50000 matrix.

An evolution of the number of iterations in face of the different tolerances passed to the algorithm is in 4.5.

To start assessing the maximum gain possible, we start by initiating the product tolerance as infitine and seeing the result. Choosing an infinite tolerance grants us the all admissible block used in the products will have rank 1.

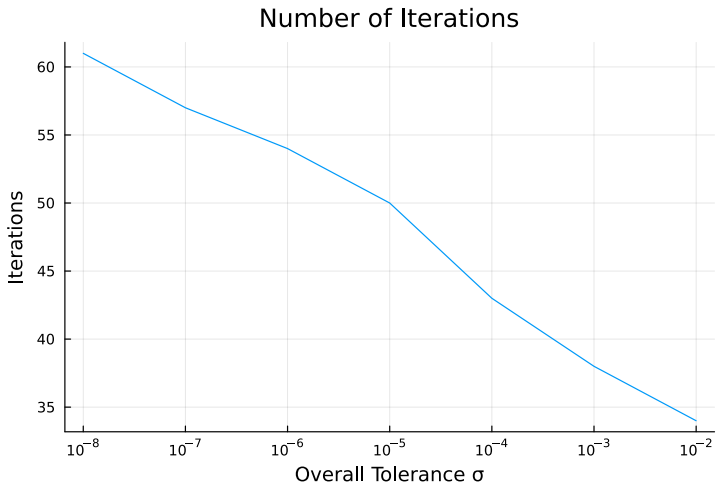



Figure 4.5: Evolution of the quantity of iterations needed for convergence and overall tolerance passed as an argument.

# Chapter 5

# Bibliography


[1] Github's repository. `https://github.com/DuduGuima/InexactGMRES.git`.

[2] Hmatrices.jl. `https://github.com/IntegralEquations/HMatrices.jl`.

[3] Inti.jl. `https://github.com/IntegralEquations/Inti.jl`.

[4] Mario Bebendorf. *Hierarchical matrices*. Springer, 2008.

[5] Marc Bonnet. *Lecture notes on numerical linear algebra*. ENSTA Paris - POEMS Group, 2021.

[6] Eric Darve and Mary Wootters. *Numerical linear algebra with Julia*. SIAM, 2021.

[7] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[8] Valeria Simoncini and Daniel B Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM Journal on Scientific Computing*, 25(2):454–477, 2003.

[9] Lloyd N Trefethen, David Bau III, and Ricardo D Fierro. Numerical linear algebra. *SIAM Review*, 40(3):735–738, 1998.