Hierarchical Matrices and Inexact GMRES

Sudent:
GUIMARÃES Eduardo
Advisors:
FARIA Luiz
MARCHAND Pierre
Enseignant-Référent:
CHAPOUTOT Alexandre

Palaiseau, 13th May 2024

Contents

1 Iterative Methods and Krylov's Subspace **2**
1.1 Iterative Methods and motivation . . . 2
1.2 Krylov's Subspace . . . 3
1.3 Arnoldi's Method . . . 3

2 GMRES **5**
2.1 Givens's Rotation . . . 5
2.2 Inexact GMRES . . . 6

3 Hierarchical Matrices and ACA Method **10**
3.1 Low-rank Matrices . . . 10
3.2 ACA Method(Adaptative Cross Approximation) . . . 11

4 First results **12**
4.1 Laplace's results . . . 13
4.2 Helmholtz's results . . . 14

Bibliography **16**

Chapter 1

Iterative Methods and Krylov's Subspace

1.1 Iterative Methods and motivation

Iterative methods appear as an alternative to exact solution methods, where the true solution is not desired and a good approximation is enough.
The idea is to find, after a certain number of iterations, a sequence x_k that converges to x, the correct solution of the problem 1.1.

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

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

$$\frac{||x_k - x||}{||x||} \leq \epsilon \tag{1.2}$$

Where ϵ is a tolerance defined by who is applying the algorithm.
Usually x isn't known, so 1.2 gets modified for 1.3, where A is the system's matrix and b is the RHS(right hand side).

$$\frac{||Ax_k - b||}{||b||} \leq \epsilon \tag{1.3}$$

The first iterative methods used a decomposition of A as a combination of two matrices 1.4, where A_1 isn't singular, and each iteration is defined as 1.5.

$$A = A_1 - A_2 \tag{1.4}$$

$$A_1 x_{k+1} = b + A_2 x_k \tag{1.5}$$

With a substitution of the others x_k, 1.5 gives 1.6, which converges for every initial solution if and only Iff $\rho(A_2 A_1^{-1}) < 1$, or $\rho(X)$ is the spectral radius of X [5].

$$x_{k+1} = A_1^{-1}(b + A_2x_k) = A_1^{-1}(b + A_2A_1^{-1}(b + A_2x_{k-1}))... = A_1^{-1}\left[\sum_{i=0}^{k}(A_2A_1^{-1})^i b\right] \tag{1.6}$$

If $A_1 = I$ and $A_2 = I - A$ in 1.4, the sequence found in 1.6 is: $x_1 = b$,$x_2 = 2b - Ab$, $x_3 = 3b - 3Ab + A^2b$, ...

Even if the condition $\rho(A - I) \leq 1$ is strong [5], it shows that one approximation x_k could be represented as 1.7.

$$x_k \in span(b, Ab, A^2b, ..., A^{k-1}b) \tag{1.7}$$

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 1.8.

$$\mathcal{K}_k(A, b) = span(b, Ab, A^2b, \ldots, A^{k-1}b) \tag{1.8}$$

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 colinears*, since those are really close of the eigenvector with the biggest eigenvalue of A.

1.3 Arnoldi's Method

With the task of obtaining an orthonormal basis to $\mathcal{K}_k(A, b)$, the method searches for a unitary matrix Q for which the expression 1.9 is valid. $H_k = h_{ij}$ is an Hessenberg's matrix.

$$AQ_k = Q_{k+1}H_k \tag{1.9}$$

For each column-vector of Q, q_i, 1.9 could be written as 1.10, where the representation of $\mathcal{K}_k(A, b)$ with an orthonormal basis becomes more evident. In a pratical application, Q est initialized with $q_1 = \frac{b}{||b||}$.

$$Aq_m = h_{1m}q_1 + h_{2m}q_2 + \ldots h_{m+1,m}q_{m+1} \tag{1.10}$$

An algorithm for the method can be found in 1.

Algorithm 1 Arnoldi's iteration

$A \in \mathbb{K}^{n\times n}$ et $b \in \mathbb{K}^n$
$x = 0, \beta = \|b\|, q_1 = \frac{b}{\beta}$
for $j = 1, 2, \dots k$ **do**
  $q_{j+1} = Aq_j$
  for $i = 1, 2, \dots j$ **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

Chapter 2

GMRES

A projection in $\mathcal{K}_k(A, b)$, where the different approximations are taken as in 2.1, where Q_m is the vector in 1.9.

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

With 2.1 and 1.9 the residue 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 bring the Arnoldi's Method presented in 1.
However, 2 doesn't bring 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 problem in 2.3, a transformation is applied to H_m, turning it into a triangular matrix.

2.1 Givens's Rotation

Givens's operator, $G(i, i+1)$, is an 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 $k = 1, 2, \dots$ **do**
  for $j = 1, 2, \dots k$ **do**
    $q_{j+1} = Aq_j$
    for $i = 1, 2, \dots j$ **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 βe_1.

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

It can be shown that g_m contains the residual of each iteration [7].
Thus, the new problem 2.5 can be solved with a simple backwards substitution.
(écrire le nouvel algorithme)

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 2.6.

$$\mathcal{A}q = (A + E)q \tag{2.6}$$

Where E in 2.6 is a *pertubation matrix* that changes with each iteration and will be written as E_k

for iteration k.
When the inexact matrix-vector product is the one being made, the left side of 1.9 must be changed by 2.7.

$$
\begin{aligned}
[(A+E_1)q_1,(A+E_2)q_2,\ldots,(A+E_k)q_k] = Q_{k+1}H_k \\
(A+\mathcal{E}_k)Q_k = Q_{k+1}H_k, \quad \mathcal{E}_k = \sum_{i=1}^{k} E_i q_i q_i^t \\
\mathcal{A}Q_k = W_k
\end{aligned}
\tag{2.7}
$$

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. The expression 2.7 also shows that Q_k becomes a basis for a new Krylov's subspace, $\mathcal{K}_k(A+\mathcal{E}_k,b)$, made by a big pertubation in A, that gets updated in each iteration.
A new distinction should also be made between the two types of residues appearing in the process: r_k, the exact residue 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 2.8.

$$
\begin{aligned}
r_k &= r_0 - AQ_k y_k \\
&= r_0 - (Q_{k+1}H_k - [E_1q_1,\ldots,E_kq_k])y_k \\
&= \tilde{r}_k + [E_1q_1,\ldots,E_kq_k]y_k \\
\rightarrow \delta_k &= \|r_k - \tilde{r}_k\| = \|[E_1q_1,\ldots,E_kq_k]y_k\|
\end{aligned}
\tag{2.8}
$$

Considering $y_k = [\eta_1^{(k)} \ldots \eta_n^{(k)}]$, upper index to clarify the iteration, an upper bound for δ_k can be found, but before we go through 2.9.

$$
\begin{aligned}
\|[E_1q_1,\ldots,E_kq_k]y_k\| = \left\| \sum_{i=1}^{k} E_i q_i \eta_i^{(k)} \right\| \\
\left\| \sum_{i=1}^{k} E_i q_i \eta_i^{(k)} \right\| \leq \sum_{i=1}^{k} \|E_i\| \left\| q_i \eta_i^{(k)} \right\| \\
\left\| \sum_{i=1}^{k} E_i q_i \eta_i^{(k)} \right\| \leq \sum_{i=1}^{k} \|E_i\| \, |\eta_i^{(k)}|
\end{aligned}
\tag{2.9}
$$

Where the fact that q_i are unitary was used between the last two lines. The bound on δ is then found in 2.10.

$$
\delta_k = \|r_k - \tilde{r}_k\| \leq \sum_{i=1}^{k} \|E_i\| \left\| \eta_i^{(k)} \right\| \tag{2.10}
$$

2.10 tells us that in order to keep both residues 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 minimization of $\|H_k y_k - e_1\beta\|$, we consider $V_k^t = \Omega_k \Omega_{k-1} \dots \Omega_1$ where each Ω represents a Givens rotation as shown in 2.4, so $\Omega_k = G(k, k+1)$.
The aplication of V_k in either side of $H_k y_k = e_1\beta$ gives us 2.11.

$$\begin{aligned} V_k H_k y_k &= V_k e_1 \beta \\ R_k y_k &= g_k \\ y_k &= R_k^{-1} g_k \end{aligned} \tag{2.11}$$

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 2.12

$$(R_k^{-1})_{i,1:k}(g_k)_{1:k} = (R_k^{-1})_{i,i:k}(g_k)_{i:k} \tag{2.12}$$

Using this last result in 2.11 gives

$$\begin{aligned} |\eta_i^{(k)}| &= \left\|(R_k^{-1})_{i,i:k}(g_k)_{i:k}\right\| \\ |\eta_i^{(k)}| &\leq \left\|e_k R_k^{-1}\right\| \|(g_k)_{i:k}\| \\ |\eta_i^{(k)}| &\leq \left\|e_k R_k^{-1}\right\| \|(g_k)_{i:k}\| \end{aligned} \tag{2.13}$$

Since $\left\|e_k R_k^{-1}\right\| \leq \left\|R_k^{-1}\right\| = \sigma_k(H_k)^{-1}$ and $\|(g_k)_{i:k}\| \leq \|\tilde{r}_{i-1}\|$ [8], the bound is given by 2.14.

$$\left\|\eta_i^{(k)}\right\| \leq \frac{1}{\sigma_k(H_k)} \|\tilde{r}_{i-1}\| \tag{2.14}$$

Putting 2.14 in 2.10 gives the results 2.15. Setting $\delta_k \leq \epsilon$ and determining a bound for each $\|E_i\|$ gets us 2.16.

$$\delta_k \leq \sum_{i=1}^{k} \frac{\|E_i\|}{\sigma_k(H_k)} \|\tilde{r}_{i-1}\| \tag{2.15}$$

$$\|E_i\| \leq \frac{\sigma_k(H_k)\epsilon}{k \|\tilde{r}_{i-1}\|} \tag{2.16}$$

Since H_k is also one of the matrices being constructed throuhout 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 a ℓ_k that will be estimated empirically [8], obtaining 2.17.

$$\|E_i\| \leq \ell_k \frac{1}{\|\tilde{r}_{i-1}\|} \epsilon \tag{2.17}$$

It should be noted [8] that some of the initial bounds found aren't really sharp, mainly 2.9 and

2.14, and further empirical analysis of these bounds could show a better theoretical bound can be found for both.

Chapter 3

Hierarchical Matrices and ACA Method

3.1 Low-rank Matrices

In reality, most matrices are big, 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 see 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 Σ 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 Σ (creating the matrix Σ_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 σ_{k+1}. For Frobenius's norm, $\|.\|_F$, the error becomes $\sum_{l=k+1}^{n} \sigma_l^2$.

Instead of approximating big 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 the precision are related in a logarithmic manner, and the rank required by a certain ϵ 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 ϵ, through the relation in 3.2. Nevertheless, this is an expensive method, where the complexity becomes too big for some calculations.
The algorithm for the method is in 3, 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 residue.

Algorithm 3 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.

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

Chapter 4

First results

Before using the inexact product in big 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 Δ 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 Γ 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

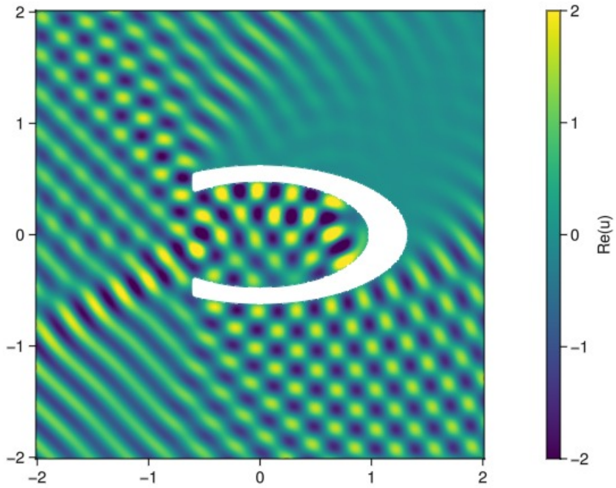


Figure 4.1: Geometry used in the test.

doing that, the product tolerance is changed to *Infinity*, and the product will be 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.

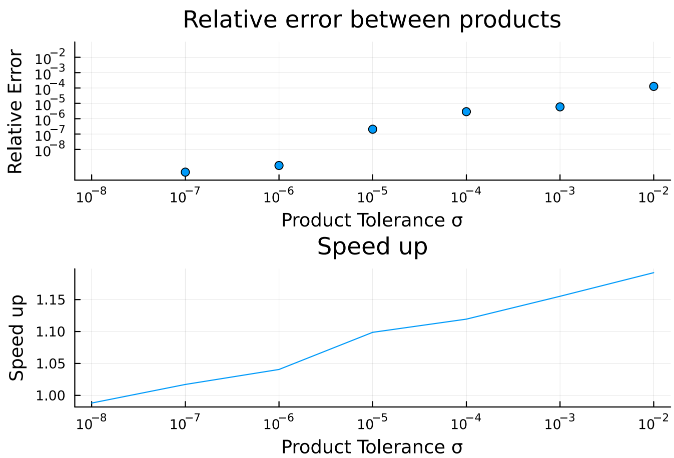
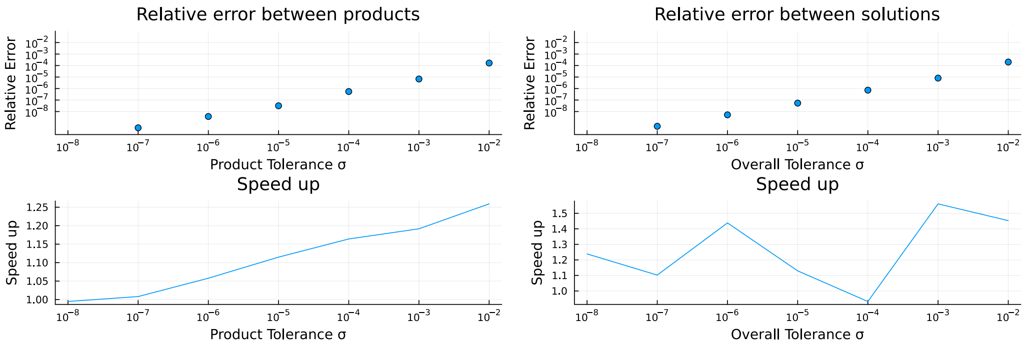


Figure 4.2: Speedup and residual evolution for the product between a 8000x8000 HMatrix and a vector.

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.

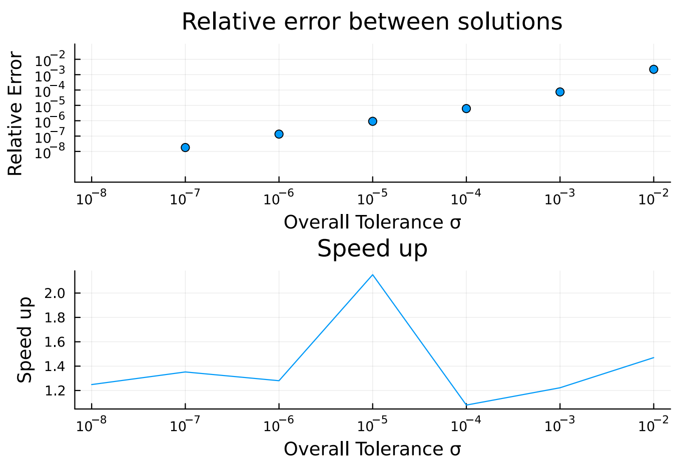


Figure 4.4: Speedup witnessed in the application of the Inexact GMRES in a 50000x50000 matrix.

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.

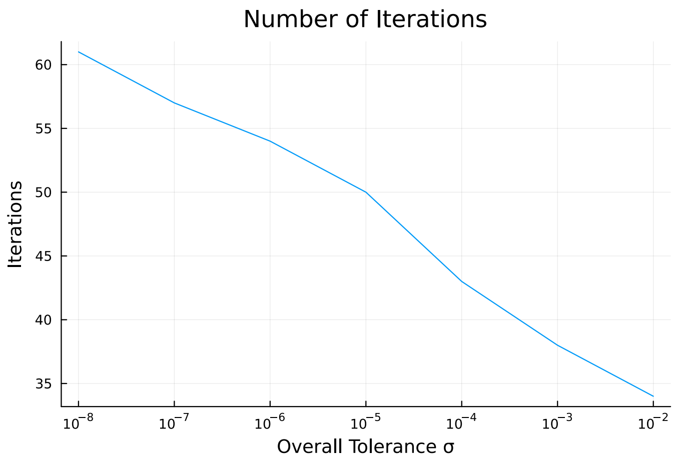


Figure 4.5: Evolution of the quantity of iterations needed for convergence and overall tolerance passed as an argument.

Bibliography

[1] Dépôt github. `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.